ML-MDS - Machine Learning

Jose Antonio Lorencio Abril

Spring 2023

Professor: Marta Arias

Student e-mail: jose.antonio.lorencio@estudiantat..upc.edu

This is a summary of the course *Machine Learning* taught at the Universitat Politècnica de Catalunya by Professor Marta Arias in the academic year 22/23. Most of the content of this document is adapted from the course notes by Arias, [1], so I won't be citing it all the time. Other references will be provided when used.

Contents

List of Figures

List of Algorithms

1 Introduction

2 Linear regression

2.1 Introduction

In Figure 1, we can observe a dataset of the population of different states plotted against the number of fatal accidents in each of the states. Here, each blue circle corresponds to a row of our data, and the coordinates are the $(population, \#accidents)$ values in the row. The red line is the linear regression model of this data. This means it is the line that 'best' approximates the data, where best refers to minimizing some kind of error: the squared error between each point to its projection on the y axis of the line, in this case. This approach is called the **least squares method**.

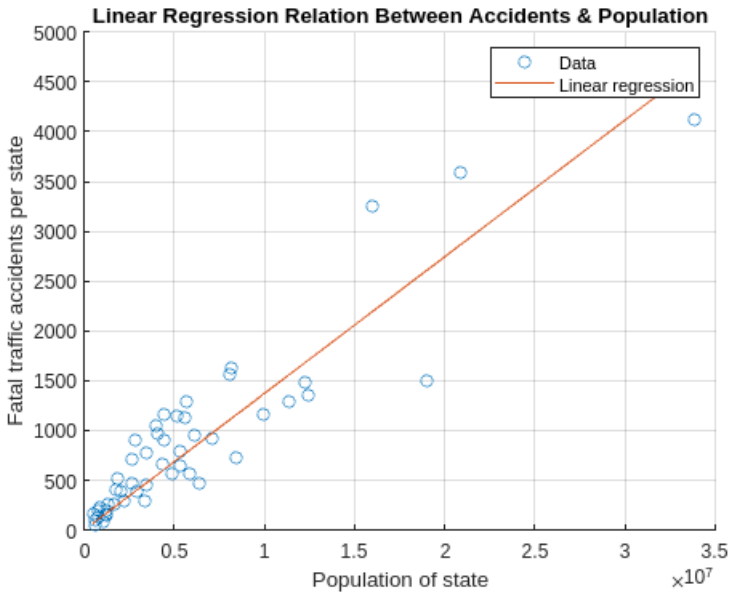


Figure 1: Matlab's accidents dataset and best linear fit.

2.2 Least squares method

2.2.1 Least squares in 2D

In 2D, we have a dataset $\{(x_i, y_i), i = 1, ..., n\}$ and we want to find the line that best approximates y as a function of x. As we want a line, we need to specify its slope, θ_1, and its intercept, θ_0. So, our estimations are:

$$\hat{y}(x_i) = \hat{y}_i = \theta_0 + \theta_1 x_i.$$

The least squares linear regression method chooses θ_0 and θ_1 in such a way that the **error function**

$$J(\theta_0, \theta_1) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - \theta_0 - \theta_1 x_i)^2$$

is minimized.
Note that this function only depends on the parameters θ_0 and θ_1, since the data is assumed to be fixed (they are observations).
To compute them, we just need to find the minimum of J, by taking partial derivatives and setting them to 0. Let's do this optimization. First, we can develop the square:

$$J(\theta_0, \theta_1) = \sum_{i=1}^{n} y_i^2 + \theta_0^2 + \theta_1^2 x_i^2 - 2\theta_0 y_i - 2\theta_1 x_i y_i + 2\theta_0 \theta_1 x_i.$$

Thus:

$$\frac{\partial J}{\partial \theta_0} = \sum_{i=1}^{n} 2\theta_0 - 2y_i + 2\theta_1 x_i = 2n\theta_0 - 2\sum_{i=1}^{n} y_i + 2\theta_1 \sum_{i=1}^{n} x_i$$

and

$$\frac{\partial J}{\partial \theta_1} = \sum_{i=1}^{n} 2\theta_1 x_i^2 - 2x_i y_i + 2\theta_0 x_i = 2\theta_1 \sum_{i=1}^{n} x_i^2 - 2\sum_{i=1}^{n} x_i y_i + 2\theta_0 \sum_{i=1}^{n} x_i.$$

We have now to solve the system given by

$$\begin{cases} 2n\theta_0 - 2\sum_{i=1}^{n} y_i + 2\theta_1 \sum_{i=1}^{n} x_i = 0 \\ 2\theta_1 \sum_{i=1}^{n} x_i^2 - 2\sum_{i=1}^{n} x_i y_i + 2\theta_0 \sum_{i=1}^{n} x_i = 0 \end{cases}$$

which is equivalent to

$$\begin{cases} n\theta_0 - \sum_{i=1}^{n} y_i + \theta_1 \sum_{i=1}^{n} x_i = 0 \\ \theta_1 \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} x_i y_i + \theta_0 \sum_{i=1}^{n} x_i = 0 \end{cases}.$$

We can isolate θ_0 from the first equation:

$$\theta_0 = \frac{\sum_{i=1}^{n} y_i - \theta_1 \sum_{i=1}^{n} x_i}{n},$$

and substitute it in the second one

$$\theta_1 \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} x_i y_i + \frac{\sum_{i=1}^{n} y_i - \theta_1 \sum_{i=1}^{n} x_i}{n} \sum_{i=1}^{n} x_i = 0,$$

which is equivalent to

$$\theta_1 \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} x_i y_i + \frac{\sum_{i=1}^{n} y_i \sum_{i=1}^{n} x_i - \theta_1 \left[\sum_{i=1}^{n} x_i\right]^2}{n} = 0$$

or

$$\theta_1 \left[\sum_{i=1}^{n} x_i^2 - \frac{\left[\sum_{i=1}^{n} x_i\right]^2}{n}\right] - \sum_{i=1}^{n} x_i y_i + \frac{\sum_{i=1}^{n} y_i \sum_{i=1}^{n} x_i}{n} = 0.$$

At this point, we can divide everything by n, yielding:

$$\theta_1 \left[\frac{\sum_{i=1}^{n} x_i^2}{n} - \frac{\left[\sum_{i=1}^{n} x_i\right]^2}{n^2}\right] - \frac{\sum_{i=1}^{n} x_i y_i}{n} + \frac{\sum_{i=1}^{n} y_i \sum_{i=1}^{n} x_i}{n^2} = 0.$$

If we now assume that the observations are equiprobable, i.e., $P(x_i) = \frac{1}{n}$, and we call X the random variable from which observations x_i are obtained and the same for the observations y_i, obtained from Y, then:

$$\frac{\sum_{i=1}^{n} x_i^2}{n} = E\left[X^2\right], \ \frac{\left[\sum_{i=1}^{n} x_i\right]^2}{n^2} = E\left[X\right]^2, \ \frac{\sum_{i=1}^{n} x_i y_i}{n} = E\left[XY\right], \ \frac{\sum_{i=1}^{n} y_i \sum_{i=1}^{n} x_i}{n^2} = E\left[X\right] E\left[Y\right].$$

This means that the previous equation can be rewritten as:

$$\theta_1 \left(E\left[X^2\right] - E\left[X\right]^2\right) - \left(E\left[XY\right] - E\left[X\right] E\left[Y\right]\right) = 0 \iff \theta_1 Var\left[X\right] - Cov\left[X, Y\right] = 0$$

So

$$\theta_1 = \frac{Cov\left[X, Y\right]}{Var\left[X\right]},$$

$$\theta_0 = E\left[Y\right] - \theta_1 E\left[X\right].$$

2.2.2 Least squares regression: multivariate case

Now, we can assume that we have m independent variables $X_1, ..., X_m$ which we want to use to predict the dependent variable Y. Again, we have n observations of each variable. Now, we want to construct an hyperplane in $\mathbb{R}^{m+1}$, whose predictions would be obtained as

$$\hat{y}(X_i) = \theta_0 + \theta_1 x_{i1} + ... + \theta_m x_{im} = \theta_0 + \sum_{j=1}^{m} \theta_j x_{ij} = \sum_{j=0}^{m} \theta_j x_{ij},$$

where we define $x_{i0} = 1$, for all i. We can represent

$$X = (x_{ij})_{i=1,...,n;\ j=1,...,m}, \ Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}, \ \theta = \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_m \end{bmatrix},$$

so that we can write

$$\hat{Y} = X\theta.$$

The error function is defined as in the simple case

$$J(\theta) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2,$$

but now we can rewrite this as

$$J(\theta) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \left(Y - \hat{Y}\right)^T \left(Y - \hat{Y}\right) = (Y - X\theta)^T (Y - X\theta).$$

Again, to obtain θ we need to optimize this function using matrix calculus.

Lemma 2.1. *If* $A = \begin{bmatrix} a_{11} & ... & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & ... & a_{nm} \end{bmatrix} \in \mathcal{M}_{n\times m}(\mathbb{R})$, $\theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_m \end{bmatrix} \in \mathbb{R}^m$ *and* $B = \begin{bmatrix} b_{11} & ... & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{n1} & ... & b_{nn} \end{bmatrix} \in \mathcal{M}_{m\times m}(\mathbb{R})$ *is a symmetric matrix, it holds:*

$$\frac{\partial A\theta}{\partial \theta} = A,$$

$$\frac{\partial \theta^T A^T}{\partial \theta} = A,$$

and

$$\frac{\partial \theta^T B\theta}{\partial \theta} = 2\theta^T B^T.$$

Proof. First, notice that $A\theta = \begin{bmatrix} \sum_{j=1}^{m} a_{1j}\theta_j \\ \vdots \\ \sum_{j=1}^{m} a_{nj}\theta_j \end{bmatrix} \in \mathbb{R}^n$,so it is

$$\frac{\partial A\theta}{\partial \theta} = \begin{bmatrix} \frac{\partial \sum_{j=1}^{m} a_{1j}\theta_j}{\partial \theta_1} & ... & \frac{\partial \sum_{j=1}^{m} a_{1j}\theta_j}{\partial \theta_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial \sum_{j=1}^{m} a_{nj}\theta_j}{\partial \theta_1} & ... & \frac{\partial \sum_{j=1}^{m} a_{nj}\theta_j}{\partial \theta_m} \end{bmatrix} = \begin{bmatrix} a_{11} & ... & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & ... & a_{nm} \end{bmatrix} = A.$$

For the second result, the procedure is the same.
Lastly, notice that $\theta^T B\theta = \sum_{k=1}^{m}\sum_{j=1}^{m} \theta_k b_{kj} \theta_j \in \mathbb{R}$, so

$$\frac{\partial \theta^T B\theta}{\partial \theta} = \begin{bmatrix} \frac{\partial \sum_{k=1}^{m}\sum_{j=1}^{m} \theta_k b_{kj}\theta_j}{\partial \theta_1} & ... & \frac{\partial \sum_{k=1}^{m}\sum_{j=1}^{m} \theta_k b_{kj}\theta_j}{\partial \theta_m} \end{bmatrix} = \begin{bmatrix} 2\sum_{j=1}^{m} b_{1j}\theta_j & ... & 2\sum_{j=1}^{m} b_{mj}\theta_j \end{bmatrix} = 2[B\theta]^T = 2\theta^T B^T.$$

□

Now, we can proceed and minimize J:

$$\begin{aligned}\frac{\partial J(\theta)}{\partial \theta} =& \frac{\partial (Y - X\theta)^T (Y - X\theta)}{\theta} \\ =& \frac{\partial}{\partial \theta}\left[Y^T Y - Y^T X\theta - \theta^T X^T Y + \theta^T X^T X\theta\right] \\ =& 0 - Y^T X - Y^T X + 2X^T X\theta \\ =& -2Y^T X + 2\theta^T X^T X,\end{aligned}$$

setting this to be 0, we get

$$\theta^T X^T X = Y^T X \iff X^T X\theta = X^T Y \iff \theta = \left(X^T X\right)^{-1} X^T Y.$$

Thus, the 'best' linear model is given by

$$\theta_{lse} = \left(X^T X\right)^{-1} X^T Y.$$

Once we have this model, if we have an observation of X, $x' = (x'_1, ..., x'_m)$ and we want to make a prediction, we compute

$$y' = x'\theta_{lse}.$$

The approach that we have followed here is the **optimization** view of learning, which basically consists of the steps:

1. Set up an error function as a function of some parameters.
2. Optimize this function to find the suitable values for this parameters, assuming the data as given.
3. Use incoming values to make predictions.

2.2.3 Computation of least squares solution via the singular values decomposition (SVD)

Inverting $X^T X$ can entail numerical problems, so the SVD can be used instead.

Theorem 2.1. *Any matrix $A \in \mathbb{R}^{m\times n}$, $m > n$, can be expressed as*

$$A = U\Sigma V^T,$$

where $U \in \mathbb{R}^{m\times n}$ has orthonormal columns $\left(U^T U = I\right)$, $\Sigma \in \mathbb{R}^{n\times n}$ is diagonal and contains the singular values in its diagonal and $V \in \mathbb{R}^{n\times n}$ is orthonormal $\left(V^{-1} = V^T\right)$.

Proof. Let $C = A^T A \in \mathbb{R}^{n\times n}$. C is square, symmetric and positive semidefinite. Therefore, C is diagonalizable, so it can be written as

$$C = V\Lambda V^T,$$

where $V = (v_i)_{i=1,...,n}$ is orthogonal and $\Lambda = diag(\lambda_1, ..., \lambda_d)$ with $\lambda_1 \geq ... \geq \lambda_r > 0 = \lambda_{r+1} = ... = \lambda_n$ and r is $rank(A) \leq n$.
Now, define $\sigma_i = \sqrt{\lambda_i}$, and form the matrix

$$\Sigma = \begin{bmatrix} diag(\sigma_1, ..., \sigma_r) & 0_{r\times(n-r)} \\ 0_{(m-r)\times r} & 0_{(m-r)\times(n-r)} \end{bmatrix}.$$

Define also

$$u_i = \frac{1}{\sigma_i} X v_i \in \mathbb{R}^m, i = 1, ..., r.$$

Then, this vectors are orthonormal:

$$u_i^T u_j = \left(\frac{1}{\sigma_i} X v_i\right)^T \left(\frac{1}{\sigma_j} X v_j\right) = \frac{1}{\sigma_i\sigma_j} v_i^T X^T X v_j = \frac{1}{\sigma_i\sigma_j} v_i^T C v_j \overset{*}{=} \frac{1}{\sigma_i\sigma_j} v_i^T (\lambda_j v_j) \overset{(\lambda_j=\sigma_j^2)}{=} \frac{\sigma_j}{\sigma_i} v_i^T v_j \overset{**}{=} 0,$$

where $(*)$ is because V is formed with the eigenvectors of C, and $(**)$ is because V is orthonormal.
Now, we can complete the base with $u_{r+1}, ..., u_n$ (using Gram-Schmidt) in such a way that

$$U = [u_1, ..., u_r, u_{r+1}, ..., u_n] \in \mathbb{R}^{n \times n}$$

is column orthonormal.
Now, if it is the case that $XV = U\Sigma$, then

$$X = XVV^T = U\Sigma V^T,$$

so it is only left to see that indeed this holds. Consider two cases:

- $1 \leq i \leq r$: $Xv_i = u_i\Sigma$ by the definition of u_i.
- $r+1 \leq i \leq n$: It is $Xv_i = 0$, because $X^TXv_i = Cv_i = \lambda_i v_i \overset{i \geq r}{=} 0$. As $X, v_i \neq 0$, it must be $Xv_i = 0$. On the other side of the equation we also have 0 because $u_i\Sigma = u_i\sigma_i = 0$ as $i > r$.

□

This, added to the fact that if X has full rank, X^TX is invertible and all its eigenvalues non null, gives us:

$$\begin{aligned}
\theta_{lse} &= \left(X^TX\right)^{-1} X^Ty \\
&= \left(\left(U\Sigma V^T\right) U\Sigma V^T\right)^{-1} \left(U\Sigma V^T\right)^T y \\
&= \left(V\Sigma U^T U\Sigma V^T\right)^{-1} V\Sigma U^T y \\
&= \Sigma^{-2} V\Sigma U^T y = V\Sigma^{-1}U^T y \\
&= V \cdot diag\left(\frac{1}{\sigma_1}, ..., \frac{1}{\sigma_n}\right) \cdot U^T y.
\end{aligned}$$

Intuitive interpretation

The intuition behind the SVD is summarized in Figure 2. Basically, every linear transformation can be decomposed into a rotation, a scaling and a simpler transformation (column orthogonal).

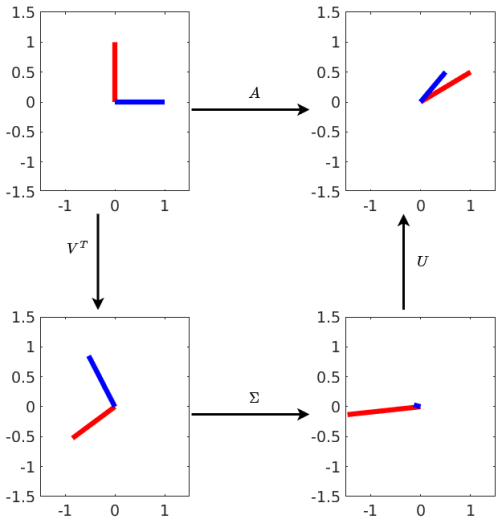


Figure 2: SVD visual.

The intuition behind SVD lies in the idea of finding a low-rank approximation of a given matrix. The rank of a matrix is the number of linearly independent rows or columns it contains. A high-rank matrix has many linearly independent rows or columns, which makes it complex and difficult to analyze. On the other hand, a low-rank matrix has fewer linearly independent rows or columns, which makes it simpler and easier to analyze.

SVD provides a way to find the best possible low-rank approximation of a given matrix by decomposing it into three components. The left singular vectors represent the **direction of maximum variance** in the data, while the right singular vectors represent the **direction of maximum correlation** between the variables. The singular values represent the **magnitude of the variance or correlation** in each direction.
By truncating the diagonal matrix of singular values to keep only the top-k values, we can obtain a low-rank approximation of the original matrix that retains most of the important information. This is useful for reducing the dimensionality of data, compressing images, and solving linear equations, among other applications.

Example 2.1. How to use SVD in Python and Matlab.

```python
import numpy as np

U, d, Vt = np.linalg.svd(X, full_matrices=False)
D = np.diag(1/d)
theta = Vt.T @ D @ U.T @ y
```

Algorithm 1: SVD in Python.

```matlab
[U, d, V] = svd(X)
D = diag(diag(1./d))
theta = V'*D*U'*y
```

Algorithm 2: SVD in Matlab.

2.3 Things that could go wrong when using linear regression

2.3.1 Our independent variable is not enough

It is possible that our variable X does not provide enough information to predict Y.

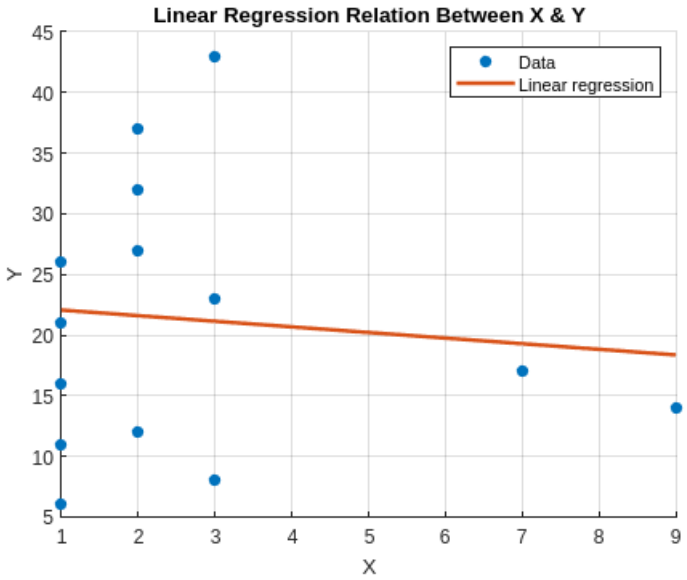


Figure 3: The independent variable does not provide enough information.

2.3.2 The relationship between the variables is not linear (underfitting)

It is also possible that the variables are related in non-linear ways.

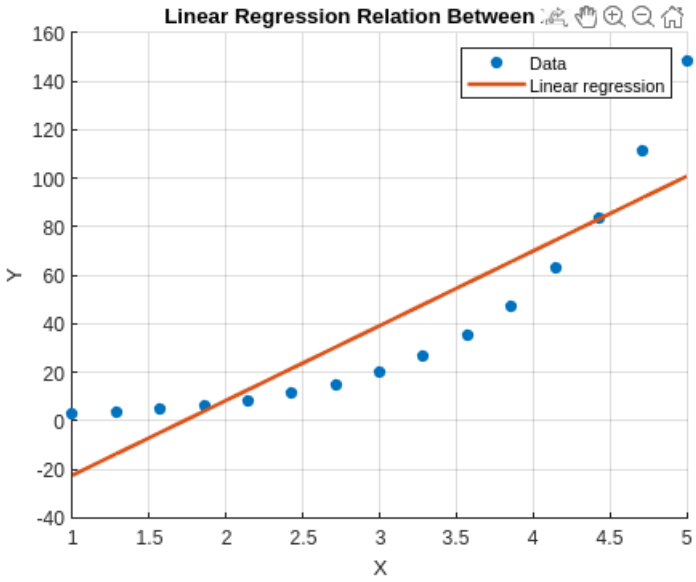


Figure 4: The variables are not linearly related.

2.3.3 Outliers affect the fit

In the presence of outliers, the model obtained can be distorted, leading to bad results.

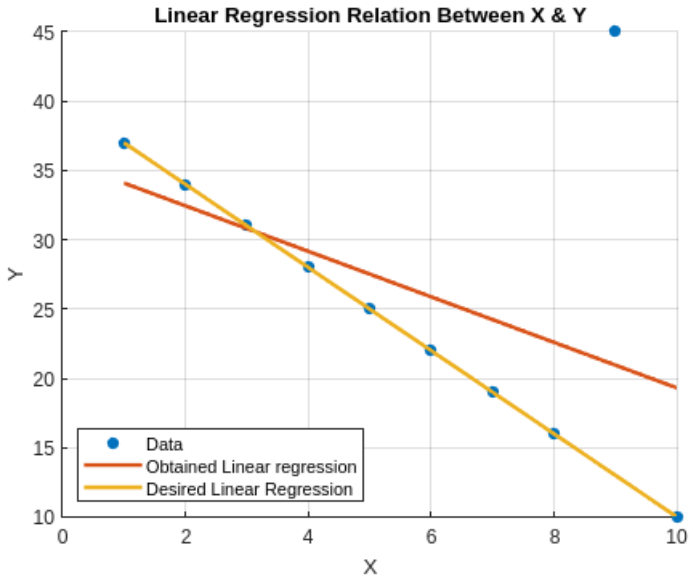


Figure 5: The outlier distort the fit.

2.4 Basis Functions

In order to fix the second problem (Subsection 2.3.2), we can make use of basis functions. The idea is to apply different transformations to the data, so that we can extend the expressive power of our model.

Definition 2.1. A **feature mapping** is a non-linear transformation of the inputs $\phi : \mathbb{R}^m \to \mathbb{R}^k$. The resulting **predictive function** or model is $y = \phi(x)\theta$.

Example 2.2. For example, we can consider the **polynomial expansion of degree** k, which is a commonly used feature mapping that approximates the relationship between the independent variable x and the dependent variable y to be polynomial of degree k, i.e.:

$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + ... + \theta_k x^k.$$

The feature mapping is $\phi(x) = \begin{pmatrix} 1 & x & x^2 & ... & x^k \end{pmatrix}$.

Note that the idea is to transform the data so that the fit is still linear, even if the relationship is not. Of course, this requires to apply the same transformation whenever we receive an input for which we want to make predictions. Also, the resulting model is more complex, so **complexity control** is necessary to avoid overfitting.
When we apply ϕ to the input matrix X, we get a new input matrix, given by

$$\Phi = \begin{pmatrix} \phi(x_1) \\ \phi(x_2) \\ \vdots \\ \phi(x_n) \end{pmatrix} = \begin{pmatrix} \phi_1(x_1) & \phi_2(x_1) & \dots & \phi_m(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \dots & \phi_m(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(x_n) & \phi_2(x_n) & \dots & \phi_m(x_n) \end{pmatrix},$$

and we obtain the optimal solution as before:

$$\theta_{min} = \arg\min_{\theta} (y - \Phi\theta)^T (y - \Phi\theta) = \left(\Phi^T \Phi\right)^{-1} \Phi^T y.$$

Example 2.3. A MATLAB example

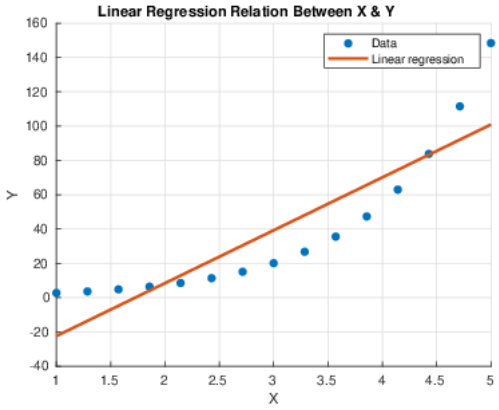
Example 2.3

First, we define the dataset. In this case $y = e^x$.

```
x=linspace(1,5,15)';
y=exp(x);
```

Now, we first see what happens with linear regression:

```
b1 = X\y;
yCalc1 = X*b1;
figure;

scatter(x,y,'filled')
hold on
plot(x,yCalc1,'LineWidth',2)
xlabel('X')
ylabel('Y')
title('Linear Regression Relation Between X & Y')
legend('Data','Linear regression')
grid on
hold off
```



As we can see, the model does not fit the data addecuately. We can use a feature mapping and repeat the process with the transformed input:

```
X = feat_map(x)
```

```
X = 15x3
    1.0000    1.0000    2.7183
    1.0000    1.2857    3.6173
    1.0000    1.5714    4.8135
    1.0000    1.8571    6.4054
    1.0000    2.1429    8.5238
    1.0000    2.4286   11.3427
    1.0000    2.7143   15.0938
    1.0000    3.0000   20.0855
    1.0000    3.2857   26.7281
    1.0000    3.5714   35.5674
```

```
b2 = X\y;
yCalc2 = X*b2;

scatter(x,y,'filled')
hold on
plot(x,yCalc2,'LineWidth',2)
xlabel('X')
ylabel('Y')
title('Linear Regression Relation Between exp(X) & Y')
legend('Data','Linear regression','Location','north')
grid on
hold off
```

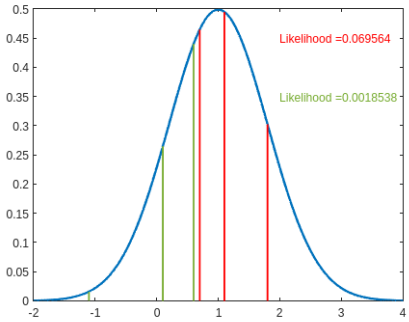
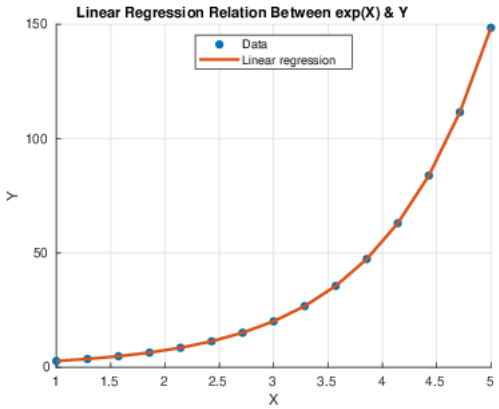


Figure 6: Likelihood example.



As we can see, the model is now perfectly fitting the data!

```
function X=feat_map(x)
    X = [ones(size(x)) x exp(x)];
end
```

2.5 Probabilistic approach

A review on probability theory, Bayes theorem and Bayesian Learning is in Appendix A.

2.5.1 Least squares regression from a probabilistic perspective

We are noe going to derive the linear regression estimates using the principle of maximum likelihood and the univariate Gaussian distribution, whose probability density function is given by

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}.$$

Given a sample $D = \{x_1, ..., x_n\}$, where $x_i \sim \mathcal{N}(\mu, \sigma)$, we define its likelihood as the function

$$\mathcal{L}(\mu, \sigma, D) = P(D; \mu, \sigma) = \prod_i p(x_i; \mu, \sigma).$$

In Figure 6, we can see how the likelihood relates to '*how likely it is that our points have been created froma certain distribution*', because the red outcomes are more likely to appear from the blue distribution than the green ones.
Now, this can be used to select the distribution that best matches our data. As an easy approach, suppose we want to decide between two distributions F_1 and F_2, and we have a dataset D. To decide, we can compute

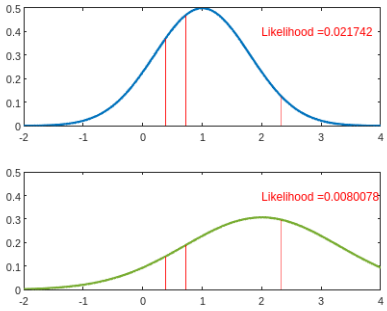


Figure 7: Using likelihood to select the distribution.

$\mathcal{L}(F_1, D)$ and $\mathcal{L}(F_2, D)$ and select the distribution whose likelihood is greater. This is visually exemplified in Figure 7, where we can see that given the three red outcomes and the two distributions (blue and green), the blue one should be preferred, because it mazimizes the likelihood.
This way, we can think of the likelihood of a function of the unknown parameters of the distribution, with the dataset fixed, and we can maximize this function to obtain the parameters that best describe our data.
In the probabilistic setting of linear regression, we assume that each label y_i we observe is normally distributed, with mean $\mu = x_i\theta$ and variance σ^2:

$$y_i = x_i\theta + \varepsilon_i,$$

where $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$. This way, we seek to obtain θ and σ^2 that best describe our data. Remember that

$$y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \; X = \begin{pmatrix} 1 & x_{11} & \dots & x_{1d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \dots & x_{nd} \end{pmatrix},$$

so that the likelihood of the parameter vector θ is given by

$$\mathcal{L}(\theta, \sigma) = P(y|X; \theta, \sigma^2) = \prod_{i=1}^{n} p(y_i|x_i; \theta, \sigma^2).$$

It is usual to maximize the log-likelihood instead, basically because the likelihood tends to give values too close to zero (we may be multiplying thousands of small values), so numerical problems may arise. Thus:

$$\begin{aligned} l(\theta, \sigma^2) = \log \mathcal{L}(\theta, \sigma^2) &= \log \prod_{i=1}^{n} p(y_i|x_i; \theta, \sigma^2) \\ &= \sum_{i=1}^{n} \log p(y_i|x_i; \theta, \sigma^2) \\ &= \sum_{i=1}^{n} \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(y_i - x_i\theta)^2} \right] \\ &= \sum_{i=1}^{n} \left(\log \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2}(y_i - x_i\theta)^2 \right) \\ &= -\frac{n}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - x_i\theta)^2 \\ &= -\frac{n}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y - X\theta)^T (y - X\theta). \end{aligned}$$

At this point, we differentiate and set equal to 0:

$$\frac{\partial}{\partial\theta} l(\theta, \sigma^2) = -\frac{1}{2\sigma^2}(-2X^T y + 2X^T X\theta) = 0$$

$$\frac{\partial}{\partial \sigma^2} l\left(\theta, \sigma^2\right) = -\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4}\left(y - X\theta\right)^T \left(y - X\theta\right) = 0,$$

obtaining

$$\theta_{ML} = \left(X^T X\right)^{-1} X^T y$$

and

$$\sigma^2_{ML} = \frac{1}{n}\left(y - X\theta\right)^T \left(y - X\theta\right) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - x_i\theta_{ML}\right)^2 = MSE.$$

It is noticeable that the maximum likelihood estimates concide with the estimates we found minimizing the squared error. This is a consequence of assuming gaussian noise, and other types of distribution would give us the same estimates as minimizing a different error function.

2.6 Bias-Variance decomposition

Let $f : \mathbb{R}^d \to \mathbb{R}$ be the true function that we are trying to approximate and $D = \{(x_1, y_1), ..., (x_n, y_n)\}$ a finite training dataset, where $y_i = f(x_i) + \varepsilon_i$ and $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$. Let $x \in \mathbb{R}^d$ be a test data point. The setup is using D to train a model $\hat{f}$ that we want to use to make predictions

$$\hat{y}_D = \hat{f}(x).$$

We are going to see how the expected squared error $(y - \hat{y}_D)^2$, where y is the real value, can be decomosed as a sum of the following components:

- **Irreducible error**: given by σ^2.
- **Bias**: the systematic limitation that the modelling assumptions impose. For example, if we choose linear approximations, we will never be able to model non-linear data well enough.
- **Variance**: refers to the sensitivity of the model to the training set D. The more the model varies when the training set is changed, the higher variance it has.

Let's do this:

$$\begin{aligned}
E\left[(y - \hat{y}_D)^2\right] =& E\left[(f(x) + \varepsilon - \hat{y}_D)^2\right] = E\left[(f(x) + \varepsilon - \hat{y}_D + E[\hat{y}_D] - E[\hat{y}_D])^2\right] \\
=& E\left[((f(x) - E[\hat{y}_D]) + \varepsilon + (E[\hat{y}_D] - \hat{y}_D))^2\right] \\
=& E\left[(f(x) - E[\hat{y}_D])^2\right] + E\left[\varepsilon^2\right] + E\left[(E[\hat{y}_D] - \hat{y}_D)^2\right] \\
&+ 2E\left[(f(x) - E[\hat{y}_D])\varepsilon\right] + 2E\left[(f(x) - E[\hat{y}_D])(E[\hat{y}_D] - \hat{y}_D)\right] \\
&+ 2E\left[\varepsilon(E[\hat{y}_D] - \hat{y}_D)\right] \\
=& E\left[(f(x) - E[\hat{y}_D])^2\right] + \sigma^2 + E\left[(E[\hat{y}_D] - \hat{y}_D)^2\right] \\
&+ 2E\left[(f(x) - E[\hat{y}_D])\right] \cancelto{0}{E[\varepsilon]} + 2E\left[(f(x) - E[\hat{y}_D])\right] \cancelto{0}{E\left[(E[\hat{y}_D] - \hat{y}_D)\right]} \\
&+ 2E\left[(E[\hat{y}_D] - \hat{y}_D)\right] \cancelto{0}{E[\varepsilon]} \\
=& E\left[(f(x) - E[\hat{y}_D])^2\right] + \sigma^2 + E\left[(E[\hat{y}_D] - \hat{y}_D)^2\right].
\end{aligned}$$

And we now define

$$\begin{aligned}
Bias[\hat{y}_D] &= E\left[(f(x) - E[\hat{y}_D])^2\right], \\
Variance[\hat{y}_D] &= E\left[(E[\hat{y}_D] - \hat{y}_D)^2\right].
\end{aligned}$$

The Bias reflects the expected difference between our assumed model and the real function, while the variance reflects the difference between the assumed model and the obtained model. In Figure 8, we can see:

- The linear model has high bias and low variance.
- The polynomial of degree 3 has low bias and moderate variance.
- The polynomial of degree 8 has low bias but high variance.

A summary of commonly used errors used for regression is shown in Table 1.

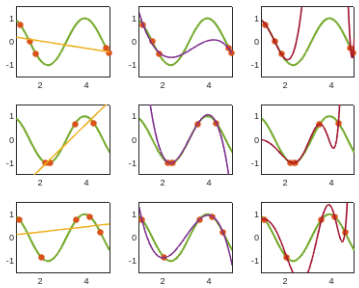


Figure 8: Visual representation of the Bias-Variance errors.

Name	Abbreviation	Formula
mean squared error	MSE	$\frac{1}{n}\sum_{i=1}^{n}(y_i - x_i\theta)^2$
root mean squared error	RMSE	$\sqrt{MSE}$
normalized root mean squared error	NRMSE	$\sqrt{\frac{MSE}{Var(y)}}$
coefficient of determination	R^2	1-NRMSE^2
mean absolute error	MAE	$\frac{1}{n}\sum_{i=1}^{n}\lvert y_i - x_i\theta\rvert$

Table 1: Common error functions.

2.7 Ridge Regression from Gaussian prior

We are going to consider MAP estimates in this section, which are explained in the Annex A.
Assume isotropic Gaussian prior on d-dimensional θ, i.e., $\theta \sim \mathcal{N}\left(\mu = 0, \Sigma = \tau^2 I\right)$, so that $\Sigma^{-1} = \frac{1}{\tau^2} I$ and $\det \Sigma = \tau^{2d}$. Then, on one hand, we have

$$\begin{aligned} P\left(\theta; \mu = 0, \Sigma = \tau^2 I\right) =& \frac{1}{\det\left(\Sigma\right)^{\frac{1}{2}} \left(2\pi\right)^{\frac{d}{2}}} \exp\left\{-\frac{1}{2}\left(y-\mu\right)^T \Sigma^{-1}\left(y-\mu\right)\right\} \\ =& \frac{1}{\left(2\pi\tau^2\right)^{\frac{d}{2}}} \exp\left\{-\frac{1}{2\tau^2}\theta^T\theta\right\} \\ =& \frac{1}{\left(2\pi\tau^2\right)^{\frac{d}{2}}} \exp\left\{-\frac{\|\theta\|^2}{2\tau^2}\right\}. \end{aligned}$$

On the other hand, it is

$$\begin{aligned} P\left(\theta|y,X\right) \propto& P\left(y|X,\theta\right) P\left(\theta\right) \\ \propto& \exp\left\{-\frac{1}{2\sigma^2}\left(y-X\theta\right)^T\left(y-X\theta\right)\right\} \exp\left\{-\frac{\|\theta\|^2}{2\tau^2}\right\} \\ =& \exp\left\{-\frac{1}{2\sigma^2}\left(y-X\theta\right)^T\left(y-X\theta\right) - \frac{\|\theta\|^2}{2\tau^2}\right\}. \end{aligned}$$

To obtain the MAP, we maximize the log of this expression:

$$\begin{aligned} \theta_{MAP} =& \arg\max_\theta \log\left[P\left(y|X,\theta\right) P\left(\theta\right)\right] \\ =& \arg\max_\theta \left[-\frac{1}{2\sigma^2}\left(y-X\theta\right)^T\left(y-X\theta\right) - \frac{\|\theta\|^2}{2\tau^2}\right] \\ =& \arg\min_\theta \left[\left(y-X\theta\right)^T\left(y-X\theta\right) + \frac{\sigma^2}{\tau^2}\|\theta\|^2\right] \\ =& \arg\min_\theta \left[\left(y-X\theta\right)^T\left(y-X\theta\right) + \lambda\|\theta\|^2\right] \end{aligned}$$

which is the **ridge regression** estimate with $\lambda = \frac{\sigma^2}{\tau^2}$. We can now differenciate this expression to find its minimum:

$$\begin{aligned}\frac{\partial}{\partial\theta}\left[\|y - X\theta\|^2 + \lambda\|\theta\|^2\right] &= -y^T X - X^T y + 2X^T X\theta + 2\lambda\theta = -2X^T y + 2X^T X\theta + 2\lambda\theta = 0 \\ &\iff \left(2X^T X + 2\lambda I\right)\theta = 2X^T y \\ &\iff \theta_{MAP} = \theta_{ridge} = \left(X^T X + \lambda I\right)^{-1} X^T y.\end{aligned}$$

Remark 2.1. There are a few remarks here:

- λ controls the complexity of the solution θ, the bigger λ is, the smaller θ tends to be, leading to simpler solutions.
- $X^T X + \lambda I$ is guaranteed to be non-singular and behaves better numerically than $X^T X$, specially if there is high correlation in the columns of X, or if there are few observations relative to the amount of variables.
- A general approach when we have a regularized objective function is to use models that are potentially more complex than needed, and then adjust λ until obtaining good result and simpler models.

2.7.1 Tuning λ

Cross-validation

Given a dataset D, the k-cross-validation approach starts by separating D into two subsets:

- The **training dataset**, D_{train}.
- The **test dataset**, D_{test}.

These are obtained in such a way that:

1. $D_{train} \cap D_{test} = \emptyset$.
2. $D_{train} \cup D_{test} = D$.

Now, we divide D_{train} into k subsets of equal size, $\{D_{train,i}\}_{i=1}^{k}$, called **folds**, and imagine we want to decide on a set of values for our model's parameter $\lambda \in \Lambda = \{\lambda_1, ..., \lambda_l\}$.

1. For each $\lambda \in \Lambda$:
   (a) For each $i = 1, ..., k$:
      i. Train the model in $D_{train} \setminus D_{train,i}$.
      ii. Evaluate the model in $D_{train,i}$.
   (b) Average the k evaluations to obtain an estimation of the performance of the model.
2. Select λ that gives us the best estimation.

Leave-one-out cross-validation (LOOCV)

Is a special case of cross-validation, in which $k = n$, i.e., each data point is a fold.

2.7.2 LOOCV for Ridge regression

As a particularity of linear and ridge regression, for a given value of λ, only one training is necessary for LOOCV, so we can proceed as follows:

1. For each $\lambda \in \Lambda$:
   (a) Compute the optimal solution
   $$\hat{\theta}_\lambda = \left(X^T X + \lambda I\right)^{-1} X^T y.$$

(b) Compute the *hat matrix* or smoothing matrix

$$H_\lambda = (h_{ij})_{ij} = X\left(X^T X + \lambda I\right)^{-1} X^T.$$

(c) Compute LOOCV directly for each λ, without folding

$$LOOCV\left(\lambda\right) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - f\left(x_i\right)\theta_\lambda}{1 - h_{ii}}\right)^2.$$

2. Return λ with minimum LOOCV.

2.7.3 Generalized Cross-Validation (GCV)

Generalized Cross-Validation (GCV) is a model selection technique used to estimate the performance of a model in terms of prediction accuracy. GCV is particularly useful for choosing the optimal parameters in regularization or smoothing methods, where the goal is to balance model complexity and goodness of fit. Examples of such methods include ridge regression, LASSO, and smoothing splines.
The main idea of GCV is to approximate the leave-one-out cross-validation (LOOCV) error without actually performing the computationally expensive process of fitting the model to all but one data point multiple times. Also, it is more computationally stable than the previous approach.
The GCV score is defined as follows:

$$GCV\left(\lambda\right) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - f\left(x_i\right)\theta_\lambda}{1 - \frac{Tr(H_\lambda)}{n}}\right)^2 = \frac{MSE_\lambda}{\left(1 - \frac{Tr(H_\lambda)}{n}\right)},$$

where $Tr\left(H_\lambda\right)$ is the trace of H_λ.

2.8 LASSO regression

Definition 2.2. The ***Lp*−norm** of a vector θ is

$$\left\|\theta\right\|_p = \left(\sum_{i=1}^{n}\left|\theta_d\right|^p\right)^{\frac{1}{p}}.$$

Remark 2.2. As we have seen, assuming an isotropic Gaussian prior on the parameters leads to ridge regression, which minimizes the L2-norm of θ and the squared error.

Another very common choice is $p = 1$, which leads to **LASSO regression**. Thus, LASSO regression minimizes the L1-norm of parameters and squared error:

$$\theta_{LASSO} = \arg\min_\theta\left[\left\|y - X\theta\right\|_2^2 + \lambda\left\|\theta\right\|_1\right].$$

In fact, LASSO regression arises assuming a Lapace distribution prior over the parameters.
Some characteristics:

- LASSO regularized cost function is not quadratic anymore, and it has no close solution, so an approximation procedure is used: the **least angle regression**, which provides an efficient way to compute the solutions for a list of possible values for $\lambda > 0$, giving the **regularization path**.
- LASSO regression gives sparse solutions, in which many coefficients/coordinates of θ might be 0. This means that LASSO performs feature selection automatically.

2.9 The full-Bayesian perspective

Both maximum likelihood (ML) and maximum a priori (MAP) produce point estimates of the parameters, while in Bayesian Learning[1] we want the full posterior distribution of the parameters. The idea is that if we know the posterior distribution of the parameters, then we can use all the information provided by this distribution for our predictions, instead of just a single point-estimate that summarizes it. For instance, if the probability function of the posterior is $p(\theta|D)$ and we receive a new input point x, then we can compute the probability of $f(x) = y$ by

$$p(y|x, D) = \int_{\Theta} p(y|x, \theta, D)\, p(\theta|D)\, d\theta.$$

Now, when we do this for all possible values of y, we obtain the full distribution of the predictions, instead of just one estimation. Nonetheless, computing this integral is usually too hard, so it needs to be approximated, but in the context of linear regression all these expressions have close-form formulas[2].
Technically, ML and MAP assume that $Y \sim \mathcal{N}\left(x^T\theta, \sigma^2\right)$, so a prediction for a new test point $\overline{x}$ is going to have a distribution $\mathcal{N}\left(\overline{x}^T\hat{\theta}, \sigma^2\right)$. Note now the lack of flexibility of this approach, since the width of the normal distribution is going to be the same for any new test point, which may be a dangerous assumption.
Let $D = \{(x_i, y_i)\}_{i=1}^n$ be our dataset, with $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, and assume:

- $Y_i, ..., Y_n$ are independent given θ.
- $Y_i \sim \mathcal{N}\left(x_i^T\theta, \frac{1}{a}\right)$, with $a > 0$ being the prevision of the noise in the observations $\left(a = \frac{1}{\sigma^2}\right)$.
- A spherical or isotropic Gaussian for the parameter's prior, $p(\theta) \sim \mathcal{N}\left(0, b^{-1}I\right)$.
- a and b are known.
- The only parameter variables are the coefficients $\theta = (\theta_0, ..., \theta_d)^T$.

Then, the likelihood function is

$$p(D|\theta) \propto \exp\left\{-\frac{a}{2}(y - X\theta)^T(y - X\theta)\right\}.$$

As usual, using Bayes, we can derive the posterior distribution

$$\begin{aligned} p(\theta|D) \propto & p(D|\theta)\, p(\theta) \\ & \propto \exp\left\{-\frac{a}{2}(y - X\theta)^T(y - X\theta)\right\} \exp\left\{-\frac{b}{2}\theta^T\theta\right\} \\ & \propto \exp\left\{-\frac{a}{2}(y - X\theta)^T(y - X\theta) - \frac{b}{2}\theta^T\theta\right\}. \end{aligned}$$

Notice here that the exponent of this expression is quadratic on θ, so it is going to be a multivariate Gaussian[3]. We need to turn the exponent into something resembling

$$(\theta - \mu)^T Q (\theta - \mu),$$

so that we can derive what the mean μ and the precision Q of the posterior density is. For this, we are going to complete squares so that we can 'match the terms' between $(\theta - \mu)^T Q (\theta - \mu)$ and $a(y - X\theta)^T(y - X\theta) + b\theta^T\theta$. On one hand:

$$\begin{aligned} (\theta - \mu)^T Q (\theta - \mu) =& \theta^T Q\theta - \theta Q\mu - \mu^T Q\theta + \mu^T Q\mu \\ =& \theta^T Q\theta - 2\theta^T Q\mu + const. \end{aligned}$$

[1] Refer to Appendix A for more details.
[2] For computational speed we may use approximations as well.
[3] The idea is that the product of two Gaussians is also Gaussian. Think of it as if you are measuring height and IQ: these variables can be assumed independent, and have a normal distribution. You can multiply them and obtain a combined normal distribution in 2-D. The idea here is basically the same.

We don't mind about constant terms with respect to θ, since we only care about proportionality. On the other hand, we have:

$$\begin{aligned} a\left(y-X\theta\right)^T\left(y-X\theta\right)+b\theta^T\theta =& ay^Ty-ay^TX\theta-a\theta^TX^Ty+a\theta^TX^TX\theta+b\theta^T\theta \\ =& ay^Ty-2a\theta^TX^Ty+\theta^T\left(aX^TX+bI\right)\theta. \end{aligned}$$

From here, we obtain that

$$Q=aX^TX+bI,$$

and we now match $-2a\theta^TX^Ty$ with $-2\theta^TQ\mu$[4]:

$$aX^Ty=Q\mu \iff \mu=aQ^{-1}X^Ty.$$

Thus, the posterior probability is $p\left(\theta|D\right)\sim\mathcal{N}\left(\mu,Q^{-1}\right)$ with

$$Q=aX^TX+bI,\ \mu=aQ^{-1}X^Ty.$$

The MAP estimate can be directly obtained from here, since the maximum density is obtained at the mean in any Gaussian distribution. Additionally, in ridge regression we let $\lambda=\frac{b}{a}$ and turn it into a parameter that we can tune to control complexity against training error.

2.9.1 Using the posterior distribution for predictions

Let's now see how to compute the **predictive distribution**, i.e.,

$$p\left(y|x,D\right)=\int_\Theta p\left(y|x,\theta,D\right)p\left(\theta|D\right)d\theta.$$

For this, we substitute the densities

$$\begin{aligned} p\left(y|x,D\right) &= \int_\Theta \mathcal{N}\left(y|x^T\theta,\frac{1}{a}\right)\mathcal{N}\left(\theta|Q^{-1}\right)d\theta \\ &\overset{w.r.t.\ y}{\propto} \int_\Theta \exp\left\{-\frac{a}{2}\left(y-x^T\theta\right)^2\right\}\exp\left\{-\frac{1}{2}\left(\theta-\mu\right)^TQ\left(\theta-\mu\right)\right\}d\theta \\ &= \int_\Theta \exp\left\{-\frac{a}{2}\left(y^2-2\left(x^T\theta\right)y+\left(x^T\theta\right)^2\right)-\frac{1}{2}\left(\theta^TQ\theta-2\theta^TQ\mu+\cancel{\mu^TQ\mu}\right)\right\}d\theta \\ &\propto \int_\Theta \exp\left\{-\frac{a}{2}\left(y^2-2\left(x^T\theta\right)y+\left(x^T\theta\right)^2\right)-\frac{1}{2}\left(\theta^TQ\theta-2\theta^TQ\mu\right)\right\}d\theta. \end{aligned}$$

Now, our objective is to set this integral to equate (or be proportional to) another of the form

$$\int_\Theta \mathcal{N}\left(\theta|...\right)g\left(y\right)d\theta=g\left(y\right)\int_\Theta \mathcal{N}\left(\theta|...\right)d\theta=g\left(y\right),$$

and finally to see that $g\left(y\right)\propto\mathcal{N}\left(y|...\right)$. We then have

$$\begin{aligned} p\left(y|x,D\right) &\propto \int_\Theta \exp\left\{-\frac{a}{2}\left(y^2-2\left(x^T\theta\right)y+\left(x^T\theta\right)^2\right)-\frac{1}{2}\left(\theta^TQ\theta-2\theta^TQ\mu\right)\right\}d\theta \\ &= \int_\Theta \exp\left\{-\frac{1}{2}\left[ay^2-2ax^T\theta y+a\theta^Txx^T\theta+\theta^TQ\theta-2\theta^TQ\mu\right]\right\}d\theta \\ &= \int_\Theta \exp\left\{-\frac{1}{2}\left[\theta^T\left(axx^T+Q\right)\theta-2\theta^T\left(xya+Q\mu\right)+ay^2\right]\right\}d\theta. \end{aligned}$$

Again, we want to match to something of the form $\left(\theta-m\right)^TL\left(\theta-m\right)=\theta^TL\theta-2\theta^TLm+m^TLm$, so

$$L=axx^T+Q$$

[4] Q is invertible because it is positive definite, thanks to the $+bI$, with $b>0$.

and

$$Lm = xya + Q\mu \iff m = L^{-1}\left(xya + Q\mu\right),$$

assuming L^{-1} exists for now. Then:

$$\begin{aligned} p\left(y|x,D\right) &\propto \int_{\Theta} \exp\left\{-\frac{1}{2}\left[\theta^T\left(axx^T + Q\right)\theta - 2\theta^T\left(xya + Q\mu\right) + ay^2\right]\right\} d\theta \\ &= \int_{\Theta} \exp\left\{-\frac{1}{2}\left[\theta^T L\theta - 2\theta^T Lm + m^T Lm - m^T Lm + ay^2\right]\right\} d\theta \\ &= \int_{\Theta} \exp\left\{-\frac{1}{2}\left[\left(\theta - m\right)^T L\left(\theta - m\right) - m^T Lm + ay^2\right]\right\} d\theta. \end{aligned}$$

Notice here that m is independent of θ so that we have

$$\begin{aligned} p\left(y|x,D\right) &\propto \int_{\Theta} \exp\left\{-\frac{1}{2}\left[\left(\theta - m\right)^T L\left(\theta - m\right) - m^T Lm + ay^2\right]\right\} d\theta \\ &= \int_{\Theta} \exp\left\{-\frac{1}{2}\left[\left(\theta - m\right)^T L\left(\theta - m\right)\right]\right\} \exp\left\{-\frac{1}{2}\left\{ay^2 - m^T Lm\right\}\right\} d\theta \\ &= \int_{\Theta} \exp\left\{-\frac{1}{2}\left[\left(\theta - m\right)^T L\left(\theta - m\right)\right]\right\} g\left(y\right) d\theta \\ &= g\left(y\right) \int_{\Theta} \exp\left\{-\frac{1}{2}\left[\left(\theta - m\right)^T L\left(\theta - m\right)\right]\right\} d\theta \\ &\propto g\left(y\right) = \exp\left\{-\frac{1}{2}\left\{ay^2 - m^T Lm\right\}\right\}. \end{aligned}$$

And now... we complete squares again :D

$$\begin{aligned} m^T Lm &= \left(ayx + Q\mu\right)^T L^{-1}\cancel{LL^{-1}}\left(ayx + Q\mu\right) \\ &= ayx^T L^{-1} ayx + 2ayx^T L^{-1}Q\mu + \cancel{\mu^T Q^T L^{-1} Q\mu} \quad \text{indep of } y \\ &= \left(a^2 x^T L^{-1} x\right) y^2 + 2\left(ax^T L^{-1} Q\mu\right) y + const. \end{aligned}$$

So

$$ay^2 - m^T Lm = \left(a - a^2 x^T L^{-1} x\right) y^2 - 2\left(ax^T L^{-1} Q\mu\right) y + const$$

If $g\left(y\right)$ is a Gaussian, then this should look something like

$$\lambda\left(y - u\right)^2 = \lambda y^2 - 2\lambda uy + \lambda u^2,$$

so

$$\lambda = a - a^2 x^T L^{-1} x$$

and

$$\lambda u = ax^T L^{-1} Q\mu,$$

so

$$u = \frac{1}{\lambda} ax^T L^{-1} Q\mu.$$

And then, we have

$$\begin{aligned} p\left(y|x,D\right) &\propto g\left(y\right) \\ &\propto \exp\left\{-\frac{\lambda}{2}\left(y - u\right)^2\right\}, \end{aligned}$$

so we have that $p\left(y|x,D\right) = \mathcal{N}\left(y|u, \frac{1}{\lambda}\right)$.
Not only this, but[5]

$$u = \mu^T x$$

[5] The derivation for these values is a bit involved. It can be consulted in https://www.youtube.com/watch?v=LCISTY9S6SQ&t=287s.

and

$$\frac{1}{\lambda} = \frac{1}{a} + x^T Q^{-1} x.$$

We note now that the predictive distribution's mean prediction equals the point-prediction of the MAP. However, the variance of the prediction does depend on x, which is good, since the unvertainty of out predictions depends on how far observed samples are:

- If observed samples are near from our new inputs, then we should be more certain.
- If they are far, then we should be less certain.

3 Clustering

The goal of clustering is to partition a dataset into groups called **clusters**, in such a way that observations in the same cluster tend to be more similar than observations in different clusters. The input data is embedded in a d-dimensional space with a similarity/dissimilarity function defined among elements in the space, which should capture relatedness among elements in this space. Two elements are understood to be related when they are close in the space. Thus, a cluster is a compact group that is separated from other groups or elements outside the cluster.
There is a large variety of clustering algorithms, such as hierarchical bottom-up or top-down clustering, probabilistic clustering, possibilistic clustering, algorithmic clustering, sprectral clustering or density-based clustering. The problem of clustering is quite complex, as if we have N data points which we want to separate into K clusters, then there are

$$S(N,K) = \frac{1}{K!}\sum_{i=1}^{K}(-1)^i \binom{K}{i}(K-i)^N$$

possibilities. This is the **stirling number of the second king**.
If in addition we don't know how many clusters we want to use, we have to add all possible $K = 1, ..., N$, summing up to

$$B(N) = \sum_{K=1}^{N} S(N,K)$$

possibilites. This number is the **Bell number**, which is really gigantic[6].

3.1 k-Means

The k-Means clustering algorithm takes a dataset $D = \{x_1, ..., x_n\}$ and an integer $k > 1$ as input, and separates D into k disjoint clusters. It is a **representative-based clustering**, meaning that each cluster is represented by one single point. In the case of k-means, the representative is the **cluster center**, $\mu_k \in \mathbb{R}^d, k = 1, ..., K$, i.e., the average of all the points in that cluster. Each point in thus assigned to its closest representative point. If C_k is the k-th cluster, then we consider it a better cluster when the value

$$\sum_{x \in C_k} \|x - \mu_k\|^2$$

is smaller.
Now, let's formalize all this. First, we introduce an indicator variable

$$r_{ik} = \begin{cases} 1 & if\ x_i \in C_k \\ 0 & otherwise \end{cases},$$

and the objective function

$$\mathcal{J}(\mu, r) = \sum_{k=1}^{K}\sum_{i=1}^{n} r_{ij}\|x_i - \mu_k\|^2,$$

which we aim to minimize by selecting appropriate $\{\mu_k\}_k$ and $\{r_{ik}\}_{ik}$. The issue is that this problem is NP-hard, so we use an heuristic method that is only guaranteed to find local minima. This method relies in two facts:

1. For fixed cluster centers μ_k, it is easy to optimize cluster assignments r_{ik}.

   Proof. Assume fixed $\{\mu_k\}_k$, then we assign x_i to the closest μ_k, because if we assign it to a different center, μ_j, then we can minimize the sum as

   $$\|x_i - \mu_j\|^2 > \|x_i - \mu_k\|^2.$$

   □

[6] For example, $B(71) \approx 4 \times 10^{71}$.

2. For fixed cluster assignments r_{ik}, it is easy to optimize cluster centers μ_k.

   Proof. Assume fixed $\{r_{ij}\}_{ij}$, then

$$\begin{aligned}
\frac{\partial}{\partial \mu_j}\mathcal{J}\left(\mu_1,...,\mu_K\right) &= \sum_{k=1}^{K}\sum_{i=1}^{n}\frac{\partial}{\partial \mu_j} r_{ij}\left\|x_i-\mu_k\right\|^2 \\
&\overset{\frac{\partial}{\partial \mu_j}\mu_k=0,\forall k\neq j}{=} \sum_{i=1}^{n} r_{ij}\frac{\partial}{\partial \mu_j}\left(x_i-\mu_j\right)^T\left(x_i-\mu_j\right) \\
&= \sum_{i=1}^{n} r_{ij}\frac{\partial}{\partial \mu_j}\left(x_i^T x_i - 2\mu_j^T x_i + \mu_j^T\mu_j\right) \\
&= \sum_{i=1}^{n} r_{ij}\left(-2x_i+2\mu_j\right) \\
&= -2\sum_{i=1}^{n} r_{ij}x_i + 2\mu_j\sum_{i=1}^{n} r_{ij}.
\end{aligned}$$

   Thus, the minimum is obtained at

$$\mu_j = \frac{\sum_{i=1}^{n} r_{ij}x_i}{\sum_{i=1}^{n} r_{ij}} = \frac{\sum_{x\in C_j} x}{card\left(C_j\right)}.$$

   This is the average of the points of the cluster. □

The pseudocode is illustrated in Algorithm 3.

```
Initialize cluster centers μ_1,...,μ_K

repeat until convergence
  assign each point to the cluster with closest center
```

$$r_{ik} = \begin{cases} 1 & if\ \ k = \arg\min_j ||x_i-\mu_j||^2 \\ 0 & otherwise \end{cases}$$

```
  recompute cluster centers for all k=1,...,K
```

$$\mu_k = \frac{\sum_{x\in C_j} x}{card\left(C_j\right)}$$

Algorithm 3: k-Means.

The characteristics of k-Means are:

Advantages :

(a) Easy implementation.
(b) Fast, even for large datasets.

Limitations :

(a) Can converge to local minimum.
(b) Needs the number of clusters, K, as input.
(c) Hard cluster assignments: meaning that each point corresponds to a single cluster, which may not always be what we want.
(d) Sensitive to ourliers and clusters of different sizes and densities.
(e) Sensitive to initialization, so it is usual to run it many times and keep the best run.
(f) Biased towards rounded clusters, because it uses the Euclidean distance.

3.2 k-Means++

k-Means++ is a variant of k-Means that uses a heuristic for initializing cluster centers as in Algorithm 4.

```
choose first center μ_1 uniformly at random from all available examples

for k=2,...,K do
  choose next center μ_k at random, with probability proportional to ||x_i − μ_l||
  here, μ_l is the closest center picked so far

proceed with standard k-Means
```

Algorithm 4: k-Means++.

3.3 Choosing the number of cluster K

The number of clusters is a hyper-parameter that has to be set by the user, and there is no obvious way to choose an optimal K, since it may not even exist. This is due to the fact that there is no such thing as a true clustering against to compare. Nevertheless, there are reasonable cluster quality criteria, that can be used to select K. These criteria measure a balance between separation of clusters and their compactness. Depending on the problem, one criterion or another should be chosen.

3.3.1 Calinski-Harabasz index

The **CH index** uses Euclidean distances to measure cluster quality, so it is usually used with k-means. It measures the ratio between:

- Separation of cluster centers: sum of distances of cluster centers to the overall mean.
- Cluster compactness: sum of distances from each point to its assigned cluster center.

Thus, it is:

$$CH = \frac{N-K}{K-1} \frac{\sum_{k=1}^{K} n_k \left\|\mu_k - \overline{x}\right\|^2}{\sum_{k=1}^{K} \sum_{i=1}^{n} r_{ik} \left\|x_i - \mu_k\right\|^2},$$

where $\overline{x} = \frac{\sum x_i}{n}$. Notice that the quantities are normalized by $\frac{N-K}{K-1}$ to avoid larger K having better values. The usual approach is run k-Means with different values of K, and then select the K that maximizes the index.

Example 3.1. A k-Means example in Matlab.

k-Means algorithm example

Introduce the data

```
X1 = [2,10];
X2 = [2,5];
X3 = [8,4];
X4 = [5,8];
X5 = [7,5];
X6 = [6,4];
X7 = [1,2];
X8 = [4,9];

D = [X1;X2;X3;X4;X5;X6;X7;X8];
Y = [X5; X6; X8];
```

Compute the distance between all points

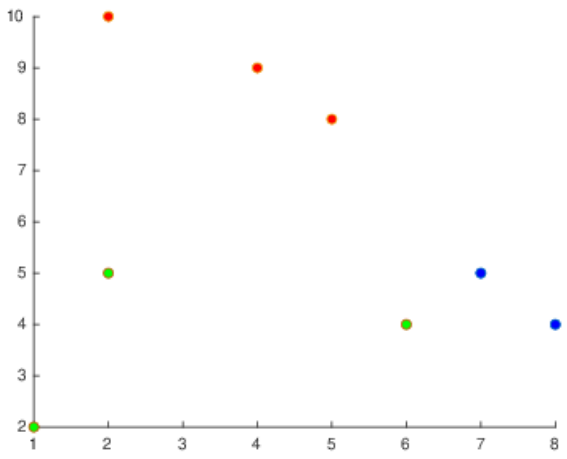
```
dist = zeros(3);

for j=1:3
    for i=1:8
        dist(i,j)=distance(D(i,:),Y(j,:),2);
    end
end
```

Select, for each point, the point that minimizes the distance

```
[v, idx] = min(dist, [], 2);
D = [D,idx];
```

Plot the clusters:

```
c1 = D(:,3) == 1;
c2 = D(:,3) == 2;
c3 = D(:,3) == 3;
scatter(D(c1,1),D(c1,2), 'MarkerFaceColor', 'b');
hold on
scatter(D(c2,1),D(c2,2), 'MarkerFaceColor', 'g');
scatter(D(c3,1),D(c3,2), 'MarkerFaceColor', 'r');
hold off
```



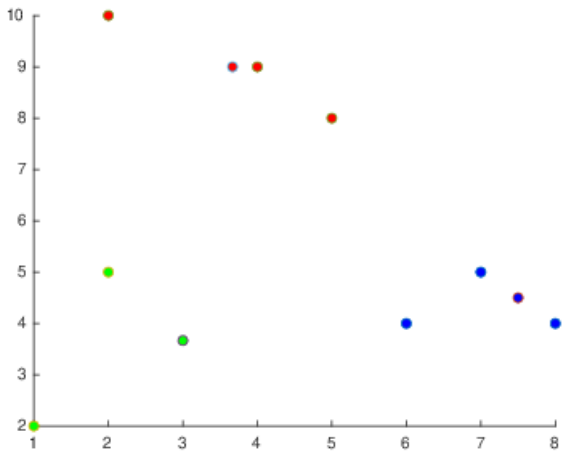
Compute the new `Y`:

```
for i=1:3
    ci = D(:,3) == i;
    x = mean(D(ci,1));
    y = mean(D(ci,2));
    Y(i,:)=[x,y];
end
Y
```

```
Y = 3x2
    7.5000    4.5000
    3.0000    3.6667
    3.6667    9.0000
```

Repeat the process:

```
%Distances
for j=1:3
    for i=1:8
        dist(i,j)=distance(D(i,1:2),Y(j,:),2);
    end
end
%Assign
[v, idx] = min(dist, [], 2);
D(:,3) = idx;
%Plot
c1 = D(:,3) == 1;
c2 = D(:,3) == 2;
c3 = D(:,3) == 3;
scatter(D(c1,1),D(c1,2), 'MarkerFaceColor', 'b');
hold on
scatter(Y(1,1),Y(1,2), 'MarkerFaceColor', 'b');
scatter(D(c2,1),D(c2,2), 'MarkerFaceColor', 'g');
scatter(Y(2,1),Y(2,2), 'MarkerFaceColor', 'g');
scatter(D(c3,1),D(c3,2), 'MarkerFaceColor', 'r');
scatter(Y(3,1),Y(3,2), 'MarkerFaceColor', 'r');
hold off
```



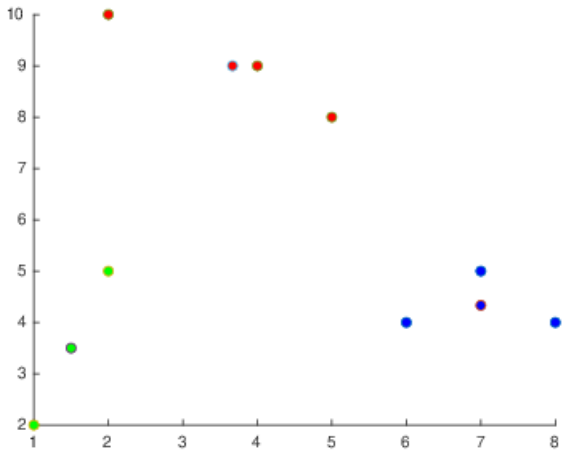

```
%Optimize
for i=1:3
    ci = D(:,3) == i;
    x = mean(D(ci,1));
    y = mean(D(ci,2));
    Y(i,:)=[x,y];
```

```
end
Y
```

```
Y = 3x2
    7.0000    4.3333
    1.5000    3.5000
    3.6667    9.0000
```

Again:

```
%Distances
for j=1:3
    for i=1:8
        dist(i,j)=distance(D(i,1:2),Y(j,:),2);
    end
end
%Assign
[v, idx] = min(dist, [], 2);
D(:,3) = idx;
%Plot
c1 = D(:,3) == 1;
c2 = D(:,3) == 2;
c3 = D(:,3) == 3;
scatter(D(c1,1),D(c1,2), 'MarkerFaceColor', 'b');
hold on
scatter(Y(1,1),Y(1,2), 'MarkerFaceColor', 'b');
scatter(D(c2,1),D(c2,2), 'MarkerFaceColor', 'g');
scatter(Y(2,1),Y(2,2), 'MarkerFaceColor', 'g');
scatter(D(c3,1),D(c3,2), 'MarkerFaceColor', 'r');
scatter(Y(3,1),Y(3,2), 'MarkerFaceColor', 'r');
hold off
```



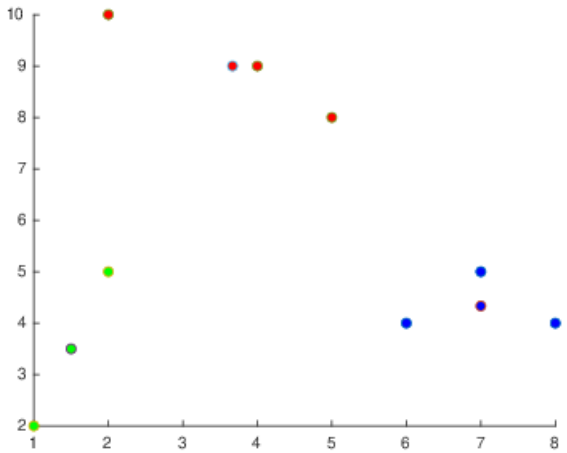

```
%Optimize
for i=1:3
    ci = D(:,3) == i;
    x = mean(D(ci,1));
    y = mean(D(ci,2));
```

```
    Y(i,:)=[x,y];
end
Y
```

```
Y = 3x2
    7.0000    4.3333
    1.5000    3.5000
    3.6667    9.0000
```

At this point we see how the clusters are the ones that we see naturally with our eyes. If we execute it again, we can see that the changes are very slight now:

```
%Distances
for j=1:3
    for i=1:8
        dist(i,j)=distance(D(i,1:2),Y(j,:),2);
    end
end
%Assign
[v, idx] = min(dist, [], 2);
D(:,3) = idx;
%Plot
c1 = D(:,3) == 1;
c2 = D(:,3) == 2;
c3 = D(:,3) == 3;
scatter(D(c1,1),D(c1,2), 'MarkerFaceColor', 'b');
hold on
scatter(Y(1,1),Y(1,2), 'MarkerFaceColor', 'b');
scatter(D(c2,1),D(c2,2), 'MarkerFaceColor', 'g');
scatter(Y(2,1),Y(2,2), 'MarkerFaceColor', 'g');
scatter(D(c3,1),D(c3,2), 'MarkerFaceColor', 'r');
scatter(Y(3,1),Y(3,2), 'MarkerFaceColor', 'r');
hold off
```



```
%Optimize
for i=1:3
    ci = D(:,3) == i;
    x = mean(D(ci,1));
    y = mean(D(ci,2));
    Y(i,:)=[x,y];
end
Y
```

```
Y = 3x2
    7.0000    4.3333
    1.5000    3.5000
    3.6667    9.0000
```

In fact, Y is not changing anymore!

Functions

```
function d = distance(X, Y, m)
    n = length(X);
    d = 0;
    for i=1:n
        d = d + (X(i)-Y(i))^m;
    end
    d = sqrt(d);
end
```

3.4 Gaussian Mixtures

A **mixture of Gaussians** is a distributions that is built using a convex sum of Gaussians, making it more flexible than a single Gaussian distribution. If the **components** of the mixture are $\mathcal{N}(\mu_k, \Sigma_k), k = 1, ..., K$ and $\pi_k, k = 1, ..., K$ are the **mixing coefficients**, with $0 \leq \pi_k \leq 1, \sum_k \pi_k = 1$, then, the density function of the mixture is given by

$$p(x|\theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x; \mu_k, \Sigma_k),$$

where $\theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1,...,K}$ represents the parameters of the distribution.
The key assumption is that each data point has been generated from only one component, we just don't know which one.
In Figure 9, we can see different Gaussian Mixtures that arise from the same components, choosing different mixing coefficients.

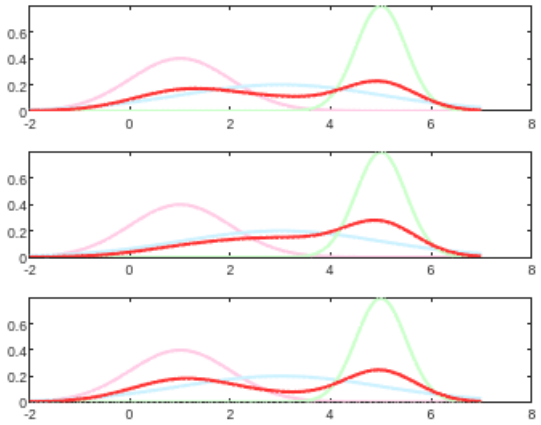


Figure 9: Different Gaussian mixtures with the same components.

3.4.1 Clustering with a Gaussian mixture

The idea is to identify each component with a cluster, so we want to determine the component from which each point in the dataset is more likely to have been created, as well as the parameters of each of the distribution. Thus, to cluster a dataset $D = \{x_1, ..., x_n\}$ into K clusters, the approach is the following:

1. Use Expectation-Maximization (EM) to estimate the mixture, obtaining approximations $\hat{\pi}_k, \hat{\mu}_k$ and $\hat{\Sigma}_k$ for each $k = 1, ..., K$.

2. Find assignments for each x_i to the clusters.

In this case, the clustering is **soft**, in opposition to the hard clustering of k-Means. This means that we will obtain the probability for each point belonging to each cluster.

3.4.2 A generative mixture of Gaussians

To sample from a mixture of Gaussians, we use a **generative model** that uses a latent variable $z = (z_1, ..., z_K)$ whose components are all 0, except one which denotes the component from which we sample, and we do:

1. Pick component k with probability π_k. This means that we set $z_k = 1$ with probability π_k.

2. Generate a sample x according to $\mathcal{N}(\mu_k, \Sigma_k)$.

The probability of generating a sample x using this generative model is

$$p(x) = \sum_z p(x, z) = \sum_k p(x, z_k = 1) = \sum_k p(x|z_k = 1)\, p(z_k = 1) = \sum_k \pi_k \mathcal{N}(x; \mu_k, \Sigma_k),$$

the joint distribution of x and z is given by

$$p(x, z) = \prod_k \pi_k^{z_k} \mathcal{N}(x; \mu_k, \Sigma_k)^{z_k},$$

and the marginal distribution over x is

$$\begin{aligned} p(x) &= \sum_k \pi_k \mathcal{N}(x; \mu_k, \Sigma_k) \\ &= \sum_z p(x, z) = \sum_z \prod_{k'} \pi_{k'}^{z_{k'}} \mathcal{N}(x; \mu_{k'}, \Sigma_{k'})^{z_{k'}}. \end{aligned}$$

Therefore, we can use Bayes to compute the conditional distribution of z given x:

$$\begin{aligned} p(z_k = 1|x) &= \frac{p(x|z_k = 1)\, p(z_k = 1)}{p(x)} \\ &= \frac{p(x|z_k = 1)\, p(z_k = 1)}{\sum_{k'} \pi_{k'} \mathcal{N}(x; \mu_{k'}, \Sigma_{k'})} \\ &= \frac{\pi_k \mathcal{N}(x; \mu_k, \Sigma_k)}{\sum_{k'} \pi_{k'} \mathcal{N}(x; \mu_{k'}, \Sigma_{k'})} \qquad = \gamma_k(x). \end{aligned}$$

The quantity $\gamma_k(x)$ indicates how probable it is that a particular data point x has been generated by the mixture component k. Or, in the context of clustering: *how probable it is that x belongs to cluster k.* We use these quantities as the **soft membership** to each cluster. If a hard membserhip is needed, then we assign x to cluster j, where $j = \arg\max_{k'} \gamma_{k'}(x)$.

3.4.3 Learning Gaussian mixtures with Expectation-Maximization

We have a dataset of unlabelled observations $D = \{x_1, ..., x_n\}$ and we want to model it as a Gaussian mixture, with unknown parameters $\theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1,...,K}$, with a fixed K. For this we use the maximum likelihood approach. First, we compute the loglikelihood of θ:

$$\begin{aligned} l(\theta) = \log \mathcal{L}(\theta) &= \log \prod_{i=1}^{n} p(x_i; \theta) \\ &= \log \prod_i \sum_k \pi_k \mathcal{N}(x; \mu_k, \Sigma_k) \\ &= \sum_i \log \sum_k \pi_k \mathcal{N}(x; \mu_k, \Sigma_k). \end{aligned}$$

This is hard to optimize... so we use the Expectation-Maximization approach. First, we can differenciate it to see what conditions must hold for local maxima, with $\frac{\partial}{\partial \mu_k} l(\theta) = 0$ leading to

$$\hat{\mu}_k = \frac{\sum_i \gamma_k(x_i)\, x_i}{\sum_i \gamma_k(x_i)} = \frac{\sum_i p(z_k = 1|x_i)\, x_i}{\sum_i p(z_k = 1|x_i)},$$

which is a weighted average of the points in our data, with weights being the soft assignments of each point to cluster k.
The problem now, is we cannot know $\gamma_k(x)$ without μ_k, Σ_k and π_k. Now, $\frac{\partial}{\partial \Sigma_k} l(\theta) = 0$ gives us

$$\hat{\Sigma}_k = \frac{\sum_i \gamma_k(x_i)(x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T}{\sum_i \gamma_k(x_i)} = \frac{\sum_i p(z_k = 1|x_i)(x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T}{\sum_i p(z_k = 1|x_i)},$$

which is the sample covariance matrix of all x_i, weigthed by the soft assignments of each point to cluster k. We have the same problem!
Since we have the constraint $\sum_k \pi_k = 1$, we now maximize the Lagrangian

$$\mathcal{L} = l(\theta) - \lambda \left(\sum_k \pi_k - 1 \right),$$

obtaining

$$\hat{\pi}_k = \frac{1}{n} \sum_i \gamma_k(x_i),$$

which is the average of all soft assignments for each point x. Again the same problem is present.
Therefore, we are in a situation in which we can estimate π_k, Σ_k and μ_k if we know γ_k, and we can compute γ_k from the estimates $\hat{\pi}_k, \hat{\Sigma}_k$ and $\hat{\mu}_k$; and we can use this in our benefit by following the pseudocode depicted in Algorithm 5. Commonly, the initializations are done using the result of k-Means in the following manner:

- Run k-Means with $k = K$.

- Set $\hat{\mu}_k$ to the mean of cluster k.
- Set $\hat{\Sigma}_k$ to the sample covariance of cluster k.
- Set $\hat{\pi}_k$ as the fraction of examples assigned to cluster k.

Initialize parameters $\hat{\pi}_k, \hat{\Sigma}_k, \hat{\mu}_k$

repeat until convergence

E-step: recompute soft assignments $\gamma_k(x_i)$

$$\gamma_k(x_i) = \frac{\pi_k \mathcal{N}(x_i; \mu_k, \Sigma_k)}{\sum_{k'} \pi_{k'} \mathcal{N}(x_i; \mu_{k'}, \Sigma_{k'})}$$

M-step: recompute ML estimates

$$\hat{\mu}_k = \frac{\sum_i \gamma_k(x_i) x_i}{\sum_i \gamma_k(x_i)}$$

$$\hat{\Sigma}_k = \frac{\sum_i \gamma_k(x_i)(x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T}{\sum_i \gamma_k(x_i)}$$

$$\hat{\pi}_k = \frac{1}{n} \sum_i \gamma_k(x_i)$$

Algorithm 5: EM algorithm.

Example 3.2. An example of the EM algorithm using Matlab

Gaussian Mixture: EM algorithm

1- Create the mixture

```
mu = [1; 9];
sigma = zeros(1,1,2);
sigma(1,1,1) = 0.5;
sigma(1,1,2) = 0.5;

x = linspace(-2,15,300)';
p1 = [0.3,0.7];

M1 = gmdistribution(mu,sigma,p1);
MM1 = M1.pdf(x);
```

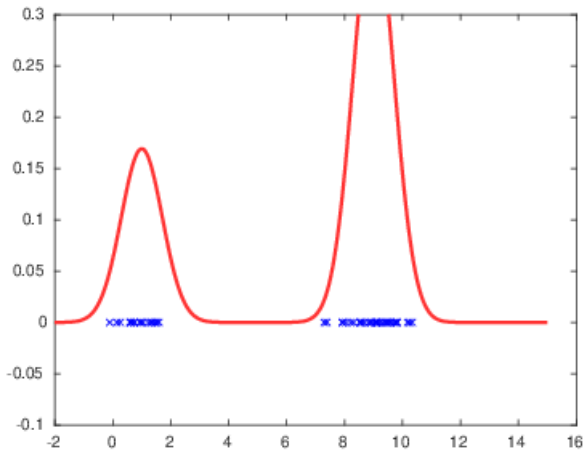
2- Sample the mixture

```
rng(3)
Y = random(M1,50)'
```

```
Y = 1x50
    8.2075    9.8171    0.1442    8.8333    9.4084    9.0987    0.7269    1.0454   -0.1108    9.4905
```

3- See what we got

```
figure(1)
plot(x,MM1,'LineWidth',2,'Color',[1 0.2 0.2 1])
hold on
ylim([-0.1 0.3])
scatter(Y,zeros(size(Y)), 'xb')
```



4- Initialize variables.

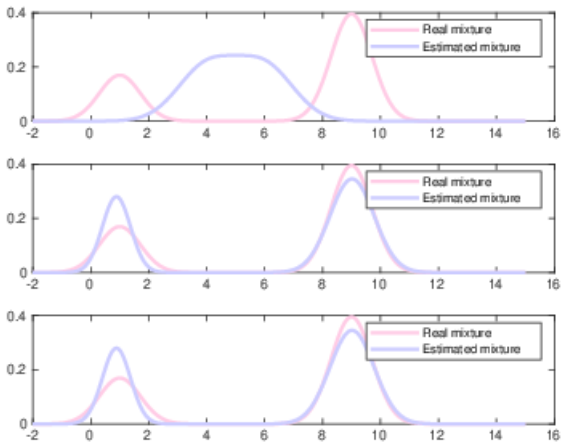
```
muhat = [4; 6];
sigmahat = zeros(1,1,2);
sigmahat(1,1,1) = 1;
sigmahat(1,1,2) = 1;
phat = [0.5 0.5]
```

```
phat = 1x2
    0.5000    0.5000
```

5- Iterate the E-step and M-step (see functions below)

```
niter = 6;
rem = 3;

figure(2)

subplot(niter/rem + 1, 1, 1)
Mhat = gmdistribution(muhat,sigmahat,phat);
MMhat = Mhat.pdf(x);
plot(x,MM1,'LineWidth',2,'Color',[1 0.8 0.9 1])
hold on
plot(x,MMhat, 'LineWidth', 2, 'Color', [0.8 0.8 1 1])
legend('Real mixture','Estimated mixture')
hold off

for i=1:niter
    g = e_step(Y,muhat,sigmahat,phat);
```

```
    [muhat,sigmahat,phat] = m_step(g,Y);

    Mhat = gmdistribution(muhat,sigmahat,phat);
    MMhat = Mhat.pdf(x);

    if mod(i,rem) == 0
        subplot(niter/rem+1,1,i/rem+1)
        plot(x,MM1,'LineWidth',2,'Color',[1 0.8 0.9 1])
        hold on
        plot(x,MMhat, 'LineWidth', 2, 'Color', [0.8 0.8 1 1])
        legend('Real mixture','Estimated mixture')
        hold off
    end
end
```




```
muhat
```

```
muhat = 2x1
    0.8838
    9.0175
```

```
sigmahat(1,1,:)
```

```
ans =
ans(:,:,1) =

    0.2352


ans(:,:,2) =

    0.5825
```

```
phat
```

```
phat = 1x2
    0.3400    0.6600
```

```
function g = e_step(Y,muhat,sigmahat,pihat)
    g = zeros(length(Y),length(muhat));
    denom = zeros(length(Y),1);
    for k=1:length(muhat)
        g(:,k) = pihat(k)*normpdf(Y(:),muhat(k),sigmahat(1,1,k));
        denom = denom + pihat(k)*normpdf(Y(:),muhat(k),sigmahat(1,1,k));
    end
    g = g./denom;
end

function [muhat,sigmahat,pihat] = m_step(g,Y)
    s = size(g);
    muhat = zeros(s(2),1);
    for k=1:s(2)
        muhat(k) = Y*g(:,k) / sum(g(:,k));
    end

    sigmahat = zeros(1,1,s(2));
    for k=1:s(2)
        sigmahat(1,1,k) = (Y-muhat(k)).^2*g(:,k) / sum(g(:,k));
    end

    pihat = zeros(1,s(2));
    for k=1:s(2)
        pihat(k) = sum(g(:,k)) / length(Y);
    end
end
```

3.4.4 Special cases

The shape of the Gaussians can be restricted, obtaining special cases of mixtures:

- No restrictions on Σ_k: the general case, in which each cluster can have general Gaussian shape.
- Σ_k are diagonal: each Gaussian component is forced to have no correlation among input dimensions.
- $\Sigma_k = \sigma^2 I$ are isotropic or spherical: each Gaussian component is forced to be spherical, so no correlation among input variables and same scaling across each input variable.

A Notes on probability theory, Bayes theorem and Bayesian learning

These notes are adapted from [2].

A.1 Probability theory basic

Let Ω be a sample space, i.e., the set of possible outcomes of an event, and $A \subset \Omega$ an event. A probability measure is a function

$$P : \mathcal{P}(\Omega) \to \mathbb{R},$$

that assigns a real number to every event, $P(A)$. This represents how likely it is that the experiment's outcome is in A.
For (Ω, P) to be a probability space[7] we have to impose three axioms:

A1) $P(A) \geq 0, \forall A \subset \Omega$.

A2) $P(\Omega) = 1$.

A3) $P(A \cup B) = P(A) + P(B)$ if $A \cap B = \emptyset$.

From these axioms, some consequences can be derived:

- $P\left(\overline{A}\right) \stackrel{def}{:=} P(\Omega \setminus A) = 1 - P(A)$.

  Proof. We can write $\Omega = A \cup (\Omega \setminus A)$ and $A \cap (\Omega \setminus A) = \emptyset$, so we have

  $$1 \stackrel{A2}{=} P(\Omega) = P(A \cup (\Omega \setminus A)) \stackrel{A3}{=} P(A) + P(\Omega \setminus A),$$

  thus:

  $$P\left(\overline{A}\right) = P(\Omega \setminus A) = 1 - P(A).$$

  □

- $P(\emptyset) = 0$.

  Proof. $P(\emptyset) = P\left(\overline{\Omega}\right) = 1 - P(\Omega) = 0$. □

- If $A \subset B$, then $P(A) \leq P(B)$.

  Proof. In this case, we can write $B = A \cup (B \setminus A)$, so $P(B) = P(A) + P(B \setminus A)$ and we have the inequality. □

- $P(A \cup B) = P(A) + P(B) - P(A \cap B)$.

  Proof. We have $A = (A \setminus B) \cup (A \cap B)$, $B = (B \setminus A) \cup (A \cap B)$ and $A \cup B = (A \setminus B) \cup (A \cap B) \cup (B \setminus A)$, so

  $$\begin{aligned} P(A \cup B) =& P(A \setminus B) + P(A \cap B) + P(B \setminus A) \\ =& P(A) - P(A \cap B) + P(A \cap B) + P(B) - P(A \cap B) \\ =& P(A) + P(B) - P(A \cap B). \end{aligned}$$

  □

- $P(A \cup B) \leq P(A) + P(B)$.

  Proof. This is obvious from the previous result. □

[7] In fact, we need one more ingredient, $\mathcal{F}$, which is a σ-algebra of subsets of Ω. This is just a formalization, but we can abuse notation here to simplify some things.

A.1.1 Joint probability

It is usal to be interested in the probability of two events happening simultaneously. This is called the **joint probability** of events A and B:

$$P(A,B) \overset{def}{:=} P(A \cap B).$$

The join probability is useful when an event can be decomposed in simpler disjoint events, i.e., we have $A \subset B_1 \cup ... \cup B_n$, with $B_i \cap B_j = \emptyset, \forall i \neq j$. In this situation, we can use the **sum rule**:

$$P(A) = \sum_{i=1}^{n} P(A, B_i).$$

This is also known as **marginalization**: when we know the joint probability $p(x,y)$ and we want to compute $p(x)$, we marginalize out y. This basically means that if we know the probabilities of all possible pairs (x,y), we can know the probability of x by exploring all the possibilities. Here, 'exploring' is using the sum rule:

$$p(x) = \sum_{y} p(x,y)$$

or

$$p(x) = \int_{y} p(x,y)\, dy,$$

if y is continuous.

Example A.1. We have two events:

- x: earns more than 100k or earns less than 100k.
- y: is a professor, a software engineer or a data scientist.

We have some sources of information, and are able to determine that

$$p(>100, prof) = 0.05,\ p(>100, seng) = 0.1,\ p(>100, dsci) = 0.2,$$

then we can conclude that

$$p(>100) = 0.05 + 0.1 + 0.2 = 0.35.$$

A.1.2 Conditional probability

The **conditional probability** of B given A is the probability that B occurs, knowing that A has occurred. This means that we have to restrict the space of possible outcomes, from Ω to A[8]:

$$P(B|A) = \frac{P(A \cap B)}{P(A)}.$$

If we rearrange the terms, we can obtain the **product rule**:

$$P(A,B) = P(B|A)\, P(A).$$

This formula can be generalized to an arbitrary number of events, the **general product rule**, given by[9]

$$P(A_1, ..., A_n) = \prod_{i=1}^{n} P(A_i | A_1, ..., A_{i-1}).$$

Exercise A.1. Prove that

$$P(A,B,C) = P(A)\, P(B|A)\, P(C|A,B) = P(C)\, P(B|C)\, P(A|B,C).$$

Then, prove the general product rule.

[8] Note that $P(A) > 0$ is needed.

[9] Note that here it is needed that all the intersections $A_1 \cap ... \cap A_i$ have non-zero probability, for $i = 1, ..., n-1$.

Basically, we are asked to prove the general product rule by induction.
For $n = 2$, we have already proven it.
For $n = 3$, we hav

$$\begin{aligned} P((A_1 \cap A_2) \cap A_3) =& P(A_3|A_1 \cap A_2) P(A_1 \cap A_2) \\ =& P(A_3|A_1 \cap A_2) P(A_2|A_1) P(A_1), \end{aligned}$$

which is what we wanted.
Now, assume it is true for $n-1$. Then, we have

$$\begin{aligned} P(A_1 \cap ... \cap A_n) =& P((A_1 \cap ... \cap A_{n-1}) \cap A_n) \\ =& P(A_n|A_1 \cap ... \cap A_{n-1}) P(A_1 \cap ... \cap A_{n-1}) \\ \overset{induction}{=}& P(A_n|A_1 \cap ... \cap A_{n-1}) \prod_{i=1}^{n-1} P(A_i|A_1, ..., A_{i-1}) \\ =& \prod_{i=1}^{n} P(A_i|A_1, ..., A_{i-1}). \end{aligned}$$

A.1.3 Bayes rule

Bayes theorem gives an alternative formula for the conditional probability:

$$P(A|B) = \frac{P(B|A) P(A)}{P(B)}.$$

Proof. Assuming all involved probabilities are non-zero:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \overset{prod\ rule}{=} \frac{P(B|A) P(A)}{P(B)}.$$

□

This rule is known to be useful to update the probability of an event happening, when we are able to gather new information of related events. $P(A)$ is usually called the **prior probability**, and $P(A|B)$ is the **a posteriori probability**, which means that we have observed B, and want to update the probability estimate for A.

Example A.2. Example of the Bayes rule in action
An English-speaking tourist visits a city whose language is not English. A local friend tells him that 1 in 10 natives speak English, 1 in 5 people in the streets are tourists and that half of the tourists speak English. Our visitor stops comeone in the street and finds that this person speaks English. What is the probability that this person is a tourist?
We have

$$P(EN|Tourist) = \frac{1}{2},\ P(EN|Local) = \frac{1}{10},\ P(Tourist) = \frac{1}{5}$$

We want to update our knowledge about the event of this person being a tourist. The prior probability is $\frac{1}{5}$, but since we know that this person speaks english, we have new information useful for updating the probability. First, the total probability of someone speaking english is

$$P(EN) \overset{sum\ rule+product\ rule}{=} P(EN|Tourist) P(Tourist) + P(EN|Local) P(Local) = \frac{1}{2}\frac{1}{5} + \frac{1}{10}\frac{4}{5} = \frac{9}{50}.$$

Now, the a posteriori probability of the person being a tourist, now that we know that he speaks english is

$$P(tourist|EN) = \frac{P(EN|Tourist) P(Tourist)}{P(EN)} = \frac{\frac{1}{2}\frac{1}{5}}{\frac{9}{50}} = \frac{5}{9}.$$

As we can see (and as we should expect), knowing that the person speaks english, our confidence that he is a tourist increases.

A.2 Bayes rule in the context of learning

As have been explained, Bayes rule allows us to reason about hypotheses from data:

$$P\left(hypothesis|data\right)=\frac{P\left(data|hypothesis\right)P\left(hypothesis\right)}{P\left(data\right)}.$$

In the jargon of parameters and datasets, this is: *let θ be a random variable with support Θ, and let D be the data that has been observed. Then, it is*

$$P\left(\theta|D\right)=\frac{P\left(D|\theta\right)P\left(\theta\right)}{P\left(D\right)}=\frac{P\left(D|\theta\right)P\left(\theta\right)}{\int_{\Theta}P\left(D|\theta\right)P\left(\theta\right)d\theta}.$$

Here, $P\left(\theta\right)$ is the **prior distribution of** θ. This means it is the distribution that we assume, before observing D. $P\left(D|\theta\right)$ is the **likelihood** of θ: the probability of observing D if the parameters are θ. $P\left(D\right)$ is the **evidence** or expected likelihood. $P\left(\theta|D\right)$ is the **posterior distribution of** θ, our quantity of interest, expressing what we know about θ after having observed D.
Thus, we can continue this line of thought to tackle a new way of creating a model: given some data D, find the best possible values for the unknown parameters θ, so that the posterior or its likelihood is maximized. There are, basically, two different approaches:

- **Maximum likelihood**: in this case, we want to choose θ in order to maximize its likelihood:

$$\theta_{ML}=\arg\max_{\theta}P\left(D|\theta\right).$$

- **Maximum a posteriori**: in this case, we take into account a prior distribution for θ and estimate its value maximizing its posterior distribution:

$$\theta_{MAP}=\arg\max_{\theta}P\left(D|\theta\right)P\left(\theta\right).$$

A.3 Maximum likelihood estimation

Given a sample $D=\{x_1,...,x_n\}$, where x_i are independent are identically distributed observations from a random variable X, following a distribution $p\left(X;\theta\right)$, with θ the paremeters of the distribution. Our objective will be to obtain the best values for θ, according to our data, assuming some special form for p.
For this, the likelihood can be used: since the x_i are independent, the probability of having the sample D is

$$p\left(D;\theta\right)=\prod_{i=1}^{n}p\left(x;\theta\right).$$

The **likelihood function** is thus defined as

$$L\left(\theta\right)=p\left(D;\theta\right).$$

Note that this is done with a fixed data D, so it is not a probability distribution, but a function of θ. This way, the maximum likelihood estimator for θ is given by

$$\theta_{ML}=\arg\max_{\theta}L\left(\theta\right).$$

There is a numerical issue here, though: as we are multiplying probabilities, which are values between 0 and 1, and we are likely to be multiplying many of them, we expect to obtain values very close to 0, which can lead to underflow in computations. Thus, it is convenient to use the **log-likelihood**:

$$\theta_{ML}=\arg\max_{\theta}L\left(\theta\right)=\arg\max_{\theta}\left[\log L\left(\theta\right)\right]=\arg\min_{\theta}\left[-\log L\left(\theta\right)\right].$$

Example A.3. Compute the maximum likelihood estimator (MLE) for univariate Gaussian distribution.

For this, we assume the data $D = \{x_1, ..., x_n\}$, where each $x_i \sim \mathcal{N}\left(\mu, \sigma^2\right)$. In this situation, the parameters are μ and σ^2. The likelihood function is

$$\begin{aligned} L\left(D; \mu, \sigma^2\right) =& \prod_{i=1}^{n} f\left(x; \mu, \sigma^2\right) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{(x_i-\mu)^2}{\sigma^2}} \\ =& \frac{1}{\left(2\pi\sigma^2\right)^{\frac{n}{2}}} e^{-\frac{1}{2\sigma^2}\sum (x_i-\mu)^2}. \end{aligned}$$

Now, the log-likelihood is

$$\begin{aligned} logL\left(D; \mu, \sigma^2\right) = \log\left(L\left(D; \mu, \sigma^2\right)\right) =& \log\left(\frac{1}{\left(2\pi\sigma^2\right)^{\frac{n}{2}}} e^{-\frac{1}{2\sigma^2}\sum (x_i-\mu)^2}\right) \\ =& \log\left(\frac{1}{\left(2\pi\sigma^2\right)^{\frac{n}{2}}}\right) + \log\left(e^{-\frac{1}{2\sigma^2}\sum (x_i-\mu)^2}\right) \\ =& \log(1) - \log\left(\left(2\pi\sigma^2\right)^{\frac{n}{2}}\right) - \frac{1}{2\sigma^2}\sum (x_i-\mu)^2 \log(e) \\ =& 0 - \frac{n}{2}\log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}\sum (x_i-\mu)^2. \end{aligned}$$

At this point, to obtain the MLE, we need to maximize this function with respect to μ and σ^2:

$$\frac{\partial}{\partial\mu} logL = -\frac{1}{\sigma^2}\sum (x_i-\mu) = 0 \iff \sum (x_i-\mu) = 0 \iff \sum x_i - n\mu = 0 \iff \mu_{MLE} = \frac{\sum x_i}{n},$$

$$\frac{\partial}{\partial\sigma^2} logL = -\frac{n}{2}\frac{1}{2\pi\sigma^2}2\pi + \frac{1}{2\sigma^4}\sum (x_i-\mu)^2 = -\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4}\sum (x_i-\mu)^2 = 0$$

$$\iff \frac{1}{2\sigma^4}\sum (x_i-\mu)^2 = \frac{n}{2\sigma^2} \iff \sum (x_i-\mu)^2 = n\sigma^2 \iff \sigma^2 = \frac{\sum (x_i-\mu)^2}{n}.$$

Now, we substitute the value obtained for μ.

$$\sigma^2_{MLE} = \frac{\sum \left(x_i - \mu_{MLE}\right)^2}{n}.$$

Example A.4. Compute the MLE for a Bernoulli distribution.
Now the observations are the results of n coin tosses. We have to compute the parameter p_{ML} of the Bernoulli random variable, whose probability function is given by

$$f(x) = p^x (1-p)^{1-x}, \ p \in (0,1), \ x \in \{0,1\}.$$

Now, we have $D = \{x_1, ..., x_n\} \subset \{0,1\}^n$. The likelihood function is

$$L(D; p) = \prod_{i=1}^{n} f(x_i) = \prod_{i=1}^{n} p^{x_i}(1-p)^{1-x_i} = p^{\sum x_i}(1-p)^{n-\sum x_i}.$$

We differentiate:

$$\begin{aligned} \frac{\partial}{\partial p} L =& \sum x_i p^{\sum x_i - 1}(1-p)^{n-\sum x_i} - p^{\sum x_i}\left[n - \sum x_i\right](1-p)^{n-\sum x_i - 1} \\ =& p^{\sum x_i - 1}(1-p)^{n-\sum x_i - 1}\left[\sum x_i (1-p) - p\left(n - \sum x_i\right)\right] \\ =& p^{\sum x_i - 1}(1-p)^{n-\sum x_i - 1}\left[\sum x_i - \cancel{p\sum x_i} - pn + \cancel{p\sum x_i}\right] \\ =& p^{\sum x_i - 1}(1-p)^{n-\sum x_i - 1}\left[\sum x_i - pn\right]. \end{aligned}$$

This derivative is zero if and only if

$$\sum x_i - pn = 0 \iff p = \frac{\sum x_i}{n} = \overline{x}.$$

A.4 Properties of estimators

Definition A.1. If we have a dataset $D = \{x_1, ..., x_n\}$ where $x_i \sim X$ and X is a random variable, we call **estimator** a function $h : \mathbb{R}^n \to \mathbb{R}^k$.

Usually, we focus on estimators that tell us something about the underlying distribution X. For example, it is usual to assume that X belongs to a certain family of random variables $X \in F(\theta)$, so that we need to estimate the parameters $\theta = (\theta_1, ..., \theta_k)$.
There are some properties that are considered desirable for an estimator to have:

- **Unbiasedness**: an estimator $\hat{\theta}$ is unbiased if in the long run it takes the value of estimated parameter. If we define the **bias** of an estimator as

$$Bias\left[\hat{\theta}\right] = E\left[\hat{\theta}\right] - \theta,$$

  then, the estimator is unbiased if

$$Bias\left[\hat{\theta}\right] = 0.$$

- **Low variance**: the variance of an estimator tells us how sensitive it is to variations in the input data D:

$$Var\left[\hat{\theta}\right] = E\left[\left(\hat{\theta} - E\left[\hat{\theta}\right]\right)^2\right] = E\left[\hat{\theta}^2\right] - E\left[\hat{\theta}\right]^2.$$

  In the case that

$$Var\left[\hat{\theta}_1\right] < Var\left[\hat{\theta}_2\right],$$

  we say that $\hat{\theta}_1$ is **more efficient** than $\hat{\theta}_2$.

  The **Cramer-Rao bound** gives a theoretical lower bound to the variance of an estimator, under some hypotheses that are usually assumed true[10]:

$$Var\left(\hat{\theta}\right) \geq \frac{\left(\frac{\partial E[\hat{\theta}]}{\partial \theta}\right)^2}{E\left[\left(\frac{\partial \ln L}{\partial \theta}\right)^2\right]}.$$

- **Efficiency**: an estimator is efficient if:
  - It is unbiased.
  - There is no other unbiased estimator with lower variance.
- **Consistency**: a sequence of estimators $\left\{\hat{\theta}_n\right\}_{n\in\mathbb{N}}$ is consistent if it converges in probability to the true value of the parameter θ:

$$\forall \varepsilon > 0, \lim_{n\to\infty} P\left(\left|\theta - \hat{\theta}\right| < \varepsilon\right) = 1.$$

  Remark A.1. If the bias and the variance of an estimator tend to 0 with n, then it is consistent.

- **Mean squeared error**: the mean squared error of an estimator is

$$MSE\left(\hat{\theta}\right) = E\left[\left(\theta - \hat{\theta}\right)^2\right],$$

  which is a value that we seek to minimize.

[10] For further information one could read my notes from a course in Statistics (in Spanish), [3].

Example A.5. Show that

$$MSE\left(\hat{\theta}\right) = Bias\left[\hat{\theta}\right]^2 + Var\left[\hat{\theta}\right].$$

Let's start from the definition:

$$\begin{aligned}
MSE\left(\hat{\theta}\right) = E\left[\left(\theta-\hat{\theta}\right)^2\right] =& E\left[\theta^2 - 2\theta\hat{\theta} + \hat{\theta}^2\right] = E\left[\theta^2\right] - 2\theta E\left[\hat{\theta}\right] + E\left[\hat{\theta}^2\right] \\
=& \theta^2 - \theta E\left[\hat{\theta}\right] - \theta E\left[\hat{\theta}\right] + E\left[\hat{\theta}^2\right] - E\left[\hat{\theta}\right]^2 + E\left[\hat{\theta}\right]^2 \\
=& \theta\left(\theta - E\left[\hat{\theta}\right]\right) - \theta E\left[\hat{\theta}\right] + E\left[\hat{\theta}^2\right] - E\left[\hat{\theta}\right]^2 + E\left[\hat{\theta}\right]^2 \\
=& Var\left[\hat{\theta}\right] - \theta \cdot Bias\left[\hat{\theta}\right] + E\left[\hat{\theta}\right]\left(E\left[\hat{\theta}\right] - \theta\right) \\
=& Var\left[\hat{\theta}\right] + Bias\left[\hat{\theta}\right]\left(E\left[\hat{\theta}\right] - \theta\right) \\
=& Var\left[\hat{\theta}\right] + Bias\left[\hat{\theta}\right]^2.
\end{aligned}$$

Example A.6. Compute the bias and the variance of the ML estimates (μ, σ^2) of an univariate Gaussian. Show that σ_{ML} is biased and that we can correct its biasedness by using a different estimator

$$\hat{\sigma}^2 = \frac{n}{n-1}\sigma^2_{ML}.$$

Compute the bias and the variance of this new estimator.
Let's start with μ:

$$\begin{aligned}
Bias\left(\mu_{MLE}\right) =& Bias\left(\frac{\sum x_i}{n}\right) = E\left[\frac{\sum x_i}{n}\right] - E\left[X\right] = \frac{1}{n}E\left[\sum x_i\right] - E\left[X\right] = \frac{1}{n}\sum E\left[x_i\right] - E\left[X\right] \\
=& \frac{1}{n}\sum E\left[X\right] - E\left[X\right] = \frac{n}{n}E\left[X\right] - E\left[X\right] = 0,
\end{aligned}$$

$$\begin{aligned}
Variance\left(\mu_{MLE}\right) =& E\left[\left(\frac{\sum x_i}{n}\right)^2\right] - E\left[\frac{\sum x_i}{n}\right]^2 \\
=& \frac{1}{n^2}E\left[\left(\sum x_i\right)^2\right] - \frac{1}{n^2}E\left[\sum x_i\right]^2 \\
=& \frac{1}{n^2}E\left[\sum x_i^2 + \sum_{i\neq j} x_i x_j\right] - \frac{1}{n^2}\left(\sum E\left[x_i\right]\right)^2 \\
=& \frac{1}{n^2}\left(\sum E\left[x_i^2\right] + \sum_{i\neq j} E\left[x_i x_j\right]\right) - \frac{1}{n^2}\left(\sum E\left[X\right]\right)^2 \\
=& \frac{1}{n^2}\left(\sum E\left[X^2\right] + \sum_{i\neq j} E\left[x_i\right]E\left[x_j\right]\right) - \frac{1}{n^2}n^2 E\left[X\right]^2 \\
=& \frac{1}{n^2}\left(nE\left[X^2\right] + \sum_{i\neq j} E\left[X\right]^2\right) - E\left[X\right]^2 \\
=& \frac{E\left[X^2\right] + n\left(n-1\right)E\left[X\right]^2 - n^2 E\left[X\right]^2}{n^2} \\
=& \frac{E\left[X^2\right] - E\left[X\right]^2}{n^2} = \frac{Var\left[X\right]}{n^2}.
\end{aligned}$$

Now σ^2_{MLE}:

$$\begin{aligned}
Bias\left(\sigma^2_{MLE}\right) =& Bias\left(\frac{\sum\left(x_i-\mu_{MLE}\right)^2}{n}\right) = E\left[\frac{\sum\left(x_i-\mu_{MLE}\right)^2}{n}\right] - Var\left[X\right] \\
=& \frac{1}{n}E\left[\sum x_i^2 - 2\mu_{MLE}\sum x_i + n\mu^2_{MLE}\right] - Var\left[X\right] \\
=& E\left[X^2\right] - \frac{2}{n}E\left[\mu_{MLE}\right]nE\left[X\right] + E\left[\mu^2_{MLE}\right] - Var\left[X\right] \\
=& E\left[X^2\right] - 2E\left[X\right]^2 + E\left[\left(\frac{\sum x_i}{n}\right)^2\right] - Var\left[X\right] \\
=& \cancel{E\left[X^2\right] - E\left[X\right]^2} - E\left[X\right]^2 + E\left[\left(\frac{\sum x_i}{n}\right)^2\right] - \cancel{Var\left[X\right]} \\
=& \frac{1}{n^2}\left(nE\left[X^2\right] + \sum_{i\neq j}E\left[X\right]^2\right) - E\left[X\right]^2 \\
=& \frac{nE\left[X^2\right] + n\left(n-1\right)E\left[X\right]^2 - n^2E\left[X\right]^2}{n^2} \\
=& \frac{nE\left[X^2\right] - nE\left[X\right]^2}{n^2} = \frac{Var\left[X\right]}{n}.
\end{aligned}$$

$$\begin{aligned}
Bias\left(\hat{\sigma}^2\right) = Bias\left(\frac{n}{n-1}\sigma^2_{MLE}\right) =& Bias\left(\frac{\sum\left(x_i-\mu_{MLE}\right)^2}{n-1}\right) = E\left[\frac{\sum\left(x_i-\mu_{MLE}\right)^2}{n-1}\right] - Var\left[X\right] \\
=& \frac{1}{n-1}E\left[\sum x_i^2 - 2\mu_{MLE}\sum x_i + n\mu^2_{MLE}\right] - Var\left[X\right] \\
=& \frac{n}{n-1}E\left[X^2\right] - \frac{2}{n-1}E\left[\mu_{MLE}\right]nE\left[X\right] + \frac{n}{n-1}E\left[\mu^2_{MLE}\right] - Var\left[X\right] \\
=& \frac{n}{n-1}E\left[X^2\right] - \frac{2n}{n-1}E\left[X\right]^2 + \frac{\cancel{n}}{n-1}\frac{1}{n^{\cancel{2}}}\left(nE\left[X^2\right] + \sum_{i\neq j}E\left[X\right]^2\right) - Var\left[X\right] \\
=& \frac{n^2E\left[X^2\right] - 2n^2E\left[X\right]^2 + nE\left[X^2\right] + n\left(n-1\right)E\left[X\right]^2 - n\left(n-1\right)Var\left[X\right]}{n\left(n-1\right)} \\
=& \frac{n^2Var\left[X\right] - n^2E\left[X\right]^2 + nE\left[X^2\right] + n^2E\left[X\right]^2 - nE\left[X\right]^2 - n^2Var\left[X\right] + nVar\left[X\right]}{n\left(n-1\right)} \\
=& 0.
\end{aligned}$$

And we see how this one is, in fact, unbiased.

A.5 Maximum a posteriori estimation

MAP (maximum a posteriori) estimation is a method of estimating the parameters of a statistical model by finding the parameter values that maximize the posterior probability distribution of the parameters, given the observed data and a prior probability distribution over the parameters. In contexts where the amount of data is limited or noisy, incorporating prior knowledge or beliefs can help to produce more stable and accurate estimates. The prior distribution serves as a regularization term, allowing us to control the degree of influence that the prior has on the estimate:

$$\hat{\theta}_{MAP} = \arg\max_\theta P\left(\theta|D\right) = \arg\max_\theta \frac{P\left(D|\theta\right)P\left(\theta\right)}{P\left(D\right)} = \arg\max_\theta P\left(D|\theta\right)P\left(\theta\right),$$

where the denominator can be ignored because it is constant for all possible θ, so it does not change the arg max.

Example A.7. Find the MAP estimate for μ of an univariate Gaussian $X \sim \mathcal{N}\left(\mu, \sigma^2\right)$ with Gaussian prior distribution for $\mu \sim \mathcal{N}\left(\mu_0, \sigma_0^2\right)$, where σ, σ_0 and μ_0 are assumed to be known.
Let $D = \{x_1, ..., x_n\}$ where $x_i \sim X$, then

$$\begin{aligned} P\left(\mu|D\right) =& \frac{P\left(D|\mu\right)P\left(\mu\right)}{P\left(D\right)} = \frac{\prod_{i=1}^{n} P\left(x_i|\mu\right)P\left(\mu\right)}{P\left(D\right)} = \frac{\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{\left(x_i-\mu\right)^2}{\sigma^2}} \cdot \frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-\frac{1}{2}\frac{\left(\mu-\mu_0\right)^2}{\sigma_0^2}}}{P\left(D\right)} \\ =& \frac{\frac{1}{\left(2\pi\sigma^2\right)^{\frac{n}{2}}} \frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(x_i-\mu\right)^2} \cdot e^{-\frac{1}{2}\frac{\left(\mu-\mu_0\right)^2}{\sigma_0^2}}}{P\left(D\right)}, \end{aligned}$$

so we want to maximize

$$f\left(\mu\right) = e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(x_i-\mu\right)^2} \cdot e^{-\frac{1}{2}\frac{\left(\mu-\mu_0\right)^2}{\sigma_0^2}},$$

or, its log function

$$g\left(\mu\right) = \log f\left(\mu\right) = -\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(x_i-\mu\right)^2 - \frac{1}{2}\frac{\left(\mu-\mu_0\right)^2}{\sigma_0^2}.$$

Thus,

$$\begin{aligned} \frac{d}{d\mu}g\left(\mu\right) =& \frac{1}{2\sigma^2}\sum_{i=1}^{n} 2\left(x_i-\mu\right) - \frac{1}{2}\frac{2\left(\mu-\mu_0\right)}{\sigma_0^2} \\ =& \frac{\sum\left(x_i-\mu\right)}{\sigma^2} - \frac{\mu-\mu_0}{\sigma_0^2} \\ =& \frac{\sum x_i - n\mu}{\sigma^2} - \frac{\mu-\mu_0}{\sigma_0^2} \\ =& \frac{\sum x_i}{\sigma^2} - \frac{n\mu\sigma_0^2+\sigma^2\mu}{\sigma^2\sigma_0^2} + \frac{\mu_0}{\sigma_0^2} \\ =& \frac{\sum x_i}{\sigma^2} - \mu\frac{n\sigma_0^2+\sigma^2}{\sigma^2\sigma_0^2} + \frac{\mu_0}{\sigma_0^2} \end{aligned}$$

and this is zero if and only if

$$\mu_{MAP} = \frac{\sigma^2\sigma_0^2}{\sigma^2+n\sigma_0^2}\left(\frac{\sum x_i}{\sigma^2} + \frac{\mu_0}{\sigma_0^2}\right) = \frac{\sigma^2\sigma_0^2}{\sigma^2+n\sigma_0^2}\left(\frac{\sigma_0^2\sum x_i + \sigma^2\mu_0}{\sigma^2\sigma_0^2}\right) = \frac{\sigma_0^2\sum x_i + \sigma^2\mu_0}{\sigma^2+n\sigma_0^2}.$$

A different way to do this is that we have

$$P\left(\mu|D\right) = \frac{\frac{1}{\left(2\pi\sigma^2\right)^{\frac{n}{2}}} \frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(x_i-\mu\right)^2} \cdot e^{-\frac{1}{2}\frac{\left(\mu-\mu_0\right)^2}{\sigma_0^2}}}{P\left(D\right)} \propto e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(x_i-\mu\right)^2} \cdot e^{-\frac{1}{2}\frac{\left(\mu-\mu_0\right)^2}{\sigma_0^2}},$$

and we want (μ_n, σ_n) such that

$$P\left(\mu|D\right) \propto e^{-\frac{\left(\mu-\mu_n\right)^2}{2\sigma_n^2}}$$

and $P\left(\mu|D\right) \sim N\left(\mu_n, \sigma_n^2\right)$, so we can solve

$$e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(x_i-\mu\right)^2} \cdot e^{-\frac{1}{2}\frac{\left(\mu-\mu_0\right)^2}{\sigma_0^2}} = e^{-\frac{\left(\mu-\mu_n\right)^2}{2\sigma_n^2}},$$

obtaining

$$\mu_n = \frac{\sigma_0^2\sum x_i + \sigma^2\mu_0}{\sigma^2+n\sigma_0^2}$$

and

$$\sigma_n = \frac{\sigma^2\sigma_0^2}{\sigma^2+n\sigma_0^2}.$$

Example A.8. The maximum likelihood estimate of p for a Bernoulli r.v. X is given by

$$p_{ML} = \frac{\sum x_i}{n}.$$

If we have $K > 2$ outcomes, then we have the categorical distribution, also known as multinoulli or generalized Bernoulli, which has support $\{1, ..., K\}$. Its parameters are $p = (p_1, ..., p_K)$, representing the probability of observing each of the possible outcomes, with $p_i \in [0, 1], \forall i$ and $\sum p_i = 1$.
It is convenient to use the *one-of-K encoding* (also called one-hot encoding) for each outcome. Thus, the pmf of this distribution becomes

$$p(x) = \prod_{i=1}^{K} p_i^{x_i}.$$

Now, given a sample $D = \{x_1, ..., x_n\}$ of possible outcomes for a multinoulli r.v. X, the maximum likelihood estimate for p is

$$\hat{p}_k = \frac{1}{n} \sum x_{ik},$$

for each $k \in \{1, ..., K\}$. We can write this compactely as

$$\hat{p} = \frac{1}{n} \sum x_i.$$

If some category k is not present in our sample, then its corresponding ML estimate is going to be 0. These 0-estimates are problematic in predictive applications, because unseen outcomes in the training data are considered to be impossible to happen, and thus can never be predicted. To avoid this, **pseudocounts** are used instead. These represent prior knowledge in the form of (imagined) counts c_k for each category k. The idea is to assume that the data is augmented with our pseudocounts, and then we estimate using maximum likelihood over the augmented data, namely

$$\hat{p} = \frac{c + \sum_i x_i}{n + \sum_k c_k},$$

so the k-th parameter is

$$\hat{p}_k = \frac{c_k + \sum_i x_{ik}}{n + \sum_k c_k}.$$

As an example, imagine that we obtain a sample from a die: $\{1, 3, 5, 4, 4, 6\}$. If the vector of pseudocounts is $c = (1, 1, 1, 1, 1, 1)$, then the estimates are

$$\hat{p}_1 = \frac{1 + 1}{6 + 6} = \frac{1}{6},$$

$$\hat{p}_2 = \frac{1}{6 + 6} = \frac{1}{12},$$

and so on. Notice that although 2 has not been observed in D, its probability estimate is not 0. This special case where all pseudocounts are 1 is known as **Laplace smoothing**.
Prove that using maximum likelihood with pseudocounts corresponds to a MAP estimate with Dirichlet prior with parameters $(c_1 + 1, ..., c_K + 1)$.
A Dirichlet distribution, $Dir(c_1 + 1, ..., c_K + 1)$ has the density function:

$$f(x) = \frac{\Gamma\left(\sum (c_i + 1)\right)}{\prod \Gamma(c_i + 1)} \prod x_i^{c_i} = \frac{\Gamma\left(\sum c_i + K\right)}{\prod c_i!} \prod x_i^{c_i} = \frac{\left(\sum c_i + K - 1\right)!}{\prod c_i!} \prod x_i^{c_i}.$$

To compute the MAP estimate, we use

$$P(D|p) P(p) = \prod_i P(x_i|p) P(p) = \prod_i \prod_k p_k^{x_{ik}} \cdot \frac{\left(\sum c_i + K - 1\right)!}{\prod c_i!} \prod_k p_k^{c_k}.$$

Thus, we want to minimize

$$f(p_1, ..., p_K) = \prod_{i,k} p_k^{x_{ik}} \prod_k p_k^{c_k},$$

or, equivalently, its log

$$g\left(p_1,...,p_K\right)=\log f\left(p_1,...,p_K\right)=\sum_{ik}x_{ik}\log\left(p_k\right)+\sum_k c_k\log\left(p_k\right).$$

We have

$$\begin{array}{ll}\min & g\left(p_1,...,p_K\right)\\ s.a. & \sum p_i=1\\ & p_i\in[0,1],\forall i\end{array}.$$

The lagrangian is

$$L=g-\lambda\left(\sum p_i-1\right),$$

so that

$$\frac{\partial}{\partial p_k}L=\frac{\partial}{\partial p_k}g-\lambda=\sum_i x_{ik}\frac{1}{p_k}+c_k\frac{1}{p_k}-\lambda=0\iff\lambda=\frac{\sum_i x_{ik}+c_k}{p_k}\iff p_k=\frac{\sum_i x_{ik}+c_k}{\lambda},$$

and then

$$1=\sum_k p_k=\sum_k\frac{\sum_i x_{ik}+c_k}{\lambda}=\frac{1}{\lambda}\left(n+\sum_k c_k\right)\iff\lambda=n+\sum_k c_k.$$

Finally, substituting back, we obtain

$$p_k=\frac{c_k+\sum_i x_{ik}}{n+\sum_k c_k},$$

as we wanted!

A.6 Bayesian Learning

In the Bayesian Learning framework, instead of working with point estimates of our unknown parameter variables θ, we work with the whole posterior distribution $P\left(\theta|D\right)$. In this case, learning is the process by which starting with some prior belief about the parameters and when facing some observations in the form of a dataset D, we update our belief about the possible values for our parameters in the form of the posterior distribution. This process is iterated over newly received data, so it can be viewed as a sequential process, in which Bayes is invoked each time we need a new posterior $P\left(\theta|D_1,D_2,...\right)$.

Example A.9. Bayesian learning in Matlab

Bayesian Learning

We are going to obtain data from a distribution $\mathcal{N}(5,2)$.

```
mu_true = 5;
sigma = 2;

num_samples = 10;

rng('default') % For reproducibility
data = mu_true + sigma_true * randn(num_samples, 1);

% Set up the x-axis for plotting
x = linspace(-5, 10, 1000);

% Plot the true distribution
figure;
hold on;
plot(x, normpdf(x, mu_true, sigma_true), 'LineWidth', 2);
title('Bayesian Learning');
xlabel('x');
```

```
ylabel('Probability Density');
xlim([-5, 10]);
legend('N(5, 2)');
```

Our prior is going to be a standard normal $\mu \sim \mathcal{N}(0, 1)$ and we are going to assume that σ is well approximated by the sample variance.

```
mu_prior = 0;
sigma_prior = 1;
```

Now, we are going to update our beliefs with the obtained datapoints, and the formulas for updating are:

$$\mu_n = \frac{\sigma_0^2 \sum x_i + \sigma^2 \mu_0}{\sigma^2 + n\sigma_0^2}$$

and

$\sigma_n = \frac{\sigma^2 \sigma_0^2}{\sigma^2 + n\sigma_0^2}$.

```
mu_posterior = (sigma_prior^2*(sum(data))+var(data)*mu_prior)/(var(data)+num_samples*sigma_prior^2)
```

```
mu_posterior = 2.7734
```

```
sigma_posterior = (var(data)*sigma_prior^2)/(var(data)+num_samples*sigma_prior^2)
```

```
sigma_posterior = 0.5561
```

```
plot(x, normpdf(x, mu_posterior, sqrt(var(data))), 'LineWidth', 2);
xlabel('x');
ylabel('Probability Density');

sigma_prior = sigma_posterior;
mu_prior = mu_posterior;

data = mu_true + sigma_true * randn(num_samples, 1);

mu_posterior = (sigma_prior^2*(sum(data))+var(data)*mu_prior)/(var(data)+num_samples*sigma_prior^2)
```

```
mu_posterior = 4.0179
```

```
sigma_posterior = (var(data)*sigma_prior^2)/(var(data)+num_samples*sigma_prior^2)
```

```
sigma_posterior = 0.2034
```

```
plot(x, normpdf(x, mu_posterior, sqrt(var(data))), 'LineWidth', 2);
xlabel('x');
ylabel('Probability Density');

sigma_prior = sigma_posterior;
mu_prior = mu_posterior;

num_samples = 500;
```

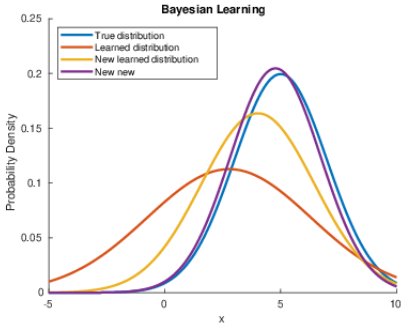
```
data = mu_true + sigma_true * randn(num_samples, 1);

mu_posterior = (sigma_prior^2*(sum(data))+var(data)*mu_prior)/(var(data)+num_samples*sigma_prior^2)
```

```
mu_posterior = 4.7770
```

```
sigma_posterior = (var(data)*sigma_prior^2)/(var(data)+num_samples*sigma_prior^2)
```

```
sigma_posterior = 0.0064
```

```
plot(x, normpdf(x, mu_posterior, sqrt(var(data))), 'LineWidth', 2);
xlabel('x');
ylabel('Probability Density');
leg = legend('True distribution', 'Learned distribution', 'New learned distribution', 'New new');
leg.Location = "northwest"
```




```
leg =
  Legend (True distribution, Learned distribution, New learned distribution, New new) with properties

         String: {'True distribution'  'Learned distribution'  'New learned distribution'  'New new'}
       Location: 'northwest'
    Orientation: 'vertical'
       FontSize: 9
       Position: [0.1483 0.7446 0.3565 0.1545]
          Units: 'normalized'

  Show all properties
```

A.6.1 Predictive posterior

When doing prediction in this framework, the whole distribution $P(\theta|D)$ is used. We view a prediction as a weighted average of all predictions each value for θ can make, weighted by its posterior probability (its expected value):

$$P(x'|D) = E_{\theta|D}[x] = \int_{\Theta} p(x'|\theta, D)\, P(\theta|D)\, d\theta.$$

Example A.10. An insightful example

Bayesian Learning: an insightful example

Suppose we have a dataset with X and Y values representing the relationship between the number of hours studied and test scores. Our true model is $y = 2X + 1$.

```
X = [1,2,3,4,4.5]';
y = [3,5,7,9,10]';
```

1) **Prior beliefs**: To start, we are searching for a linear relationship y=a*X+b. First, we need to define prior distributions for a and b. These distributions represent our initial beliefs about the values of a and b before seeing any data. A common choice is to use normal distributions with a mean of 0 and a large variance (e.g., 1000) to indicate a high level of uncertainty. For example, a~N(0,1000) and b~N(0,1000).

```
mu_prior = [0;0];
sigma_prior = 1000;
V_prior = sigma_prior^2*eye(2) %Covariance matrix for prior distributions
```

```
V_prior = 2x2
     1000000           0
           0     1000000
```

```
X_design = [X, ones(size(X))] %Design matrix [X,1]
```

```
X_design = 5x2
    1.0000    1.0000
    2.0000    1.0000
    3.0000    1.0000
    4.0000    1.0000
    4.5000    1.0000
```

2) **Likelihood function**: We need to define a likelihood function, which describes the probability of the observed data given the parameters a and b. In linear regression, we typically assume that the errors are normally distributed. Therefore, the likelihood function can be represented as $y_i \sim \mathcal{N}(aX_i + b, \sigma^2)$, where σ^2 is the variance of the error. We will assume it known, because the approaches to get it are a bit involved.

```
sigma_error = 1;
```

3) **Posterior distribution**: We update our beliefs about the coefficients a and b given the data by computing the posterior distribution. This is done by multiplying the prior distributions with the likelihood function and applying Bayes' theorem:

$P(a, b|X, y) = \frac{P(y|X,a,b)*P(a,b)}{P(Y|X)}$.

Note that the denominator is a normalizing constant that ensures the posterior distribution sums to 1.

$$V_{post} = (V_{prior}^{-1} + \frac{1}{\sigma_{error}^2} \cdot X_{design}^T * X_{design})^{-1}$$

$$\mu_{post} = V_{post} \cdot (V_{prior}^{-1} \cdot \mu_{prior} + \frac{1}{\sigma_{error}^2} \cdot X_{design}^T \cdot y)$$

```
V_posterior = inv(inv(V_prior) + (1 / sigma_error^2) * (X_design' * X_design))
```

```
V_posterior = 2x2
    0.1220   -0.3537
   -0.3537    1.2256
```

```
mu_posterior = V_posterior * (inv(V_prior) * mu_prior + (1 / sigma_error^2) * (X_design' * y))
```

```
mu_posterior = 2x1
    2.0000
    1.0000
```

4) **Making predictions**: To make predictions for new X values, we can use the posterior predictive distribution, which accounts for the uncertainty in the estimates of a and b.

```
% Number of samples
n_samples = 1000;

% Sample from the posterior distribution of a and b
ab_samples = mvnrnd(mu_posterior, V_posterior, n_samples);

% Prediction for new X value
X_new = 3.5;

% Generate predictions using the samples of a and b
Y_new_samples = zeros(n_samples, 1);
for i = 1:n_samples
    a_sample = ab_samples(i, 1);
    b_sample = ab_samples(i, 2);

    Y_new_mean_sample = [X_new, 1] * [a_sample; b_sample];
    Y_new_samples(i) = Y_new_mean_sample + sigma_error * randn;
end

% Compute summary statistics for the predictions
Y_new_mean = mean(Y_new_samples);
Y_new_std = std(Y_new_samples);

fprintf('Prediction for X = %d: Y = %.2f +/- %.2f\n', X_new, Y_new_mean, Y_new_std);
```

```
Prediction for X = 3.500000e+00: Y = 7.98 +/- 1.13
```

The true value according to the model is 8, and we have obtained 7.98 +- 1.13, which is pretty good!

References

[1] Marta Arias. Machine learning. Lecture Notes.

[2] Marta Arias. Notes on probability theory, bayes theorem and bayesian learning. Lecture Notes.

[3] Jose Antonio Lorencio Abril. Apuntes de inferencia estadÃstica, 2021.