Mathematical foundations for ML - UM6P class

Clustering and Latent Variable Models
with an application in Feature Importance Analysis

06 February 2023

Contents

1 Clustering methods

1.1 Motivation

Given a data set $X = (x_1, \dots, x_n) \in \mathbb{R}^{n \times p}$ where n is the number of observation and p is the number of features, we want to separate these data into K classes (clusters), i.e.we want to learn :

- the centroid (center) of each cluster $\{c_1, \dots, c_K\} \in \mathbb{R}^{p \times K}$
- an assignation function $\Psi : \{x_1, \dots, x_n\} \in \mathbb{R}^{n \times p} \to \{1, \dots, K\}$, meaning "sample x_i belongs to class $\Psi(x_i)$ ".

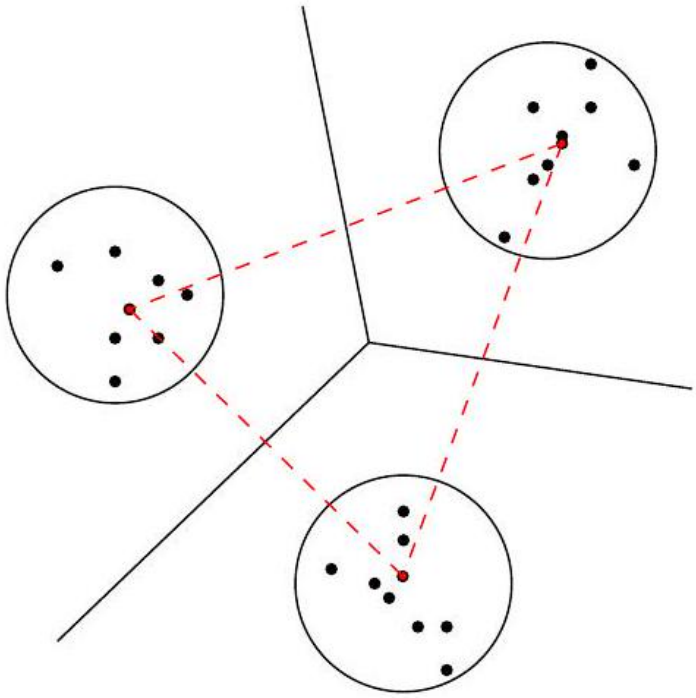


Figure 1: A simple representation of the situation $(n = 25, p = 2, K = 3)$

1.2 The K-means Clustering Algorithm

Definition 1.1. *The algorithm 1 is called the K-means algorithm. It's an iterative algorithm that provides an assignation function Ψ^* and the associated centroids $c_1^*, \ldots, c_K^*$.*

Algorithm 1 The K-means Algorithm

Require: A data set $X = \{x_1, \ldots, x_n\}$ $(x_i \in \mathbb{R}^p)$
Ensure: An assignation function Ψ^* and the associated centroids $c_1^*, \ldots, c_K^*$.

```
Initialization: Choose c_1, ..., c_K in X at random
repeat
    Assignment step:
    for i = 1 ... n do
        Ψ(x_i) ← arg min_{k∈{1,...,K}} ||x_i − c_k||^2
    end for
    Re-estimation step:
    for k = 1 ... K do
        c_j ← 1 / (Σ_{i=1}^n 1(Ψ(x_i)=k)) Σ_{i=1}^n 1 (Ψ(x_i) = k) x_i
    end for
until convergence
return Ψ*, c_1^*, ..., c_K^*
```

Definition 1.2. *Let Ψ be an assignation function and $c = (c_1, \ldots, c_K)$ be the corresponding centroids. We define the distortion $J(\Psi, c)$ as follows:*

$$J(\Psi, c) = \frac{1}{n} \sum_{i=1}^{n} \left\| x_i - c_{\Psi(x_i)} \right\|^2$$

Theorem 1.2.1. *The K-means algorithm 1 monotonically decreases the distortion*

Exercise:

For each iteration t of the algorithm 1, we define the distortion at time t as :

$$J(\Psi^{(t)}, c^{(t)}) = \frac{1}{n} \sum_{i=1}^{n} \left\| x_i - c^{(t)}_{\Psi^{(t)}(x_i)} \right\|^2$$

Show that for all t:

$$J(\Psi^{(t)}, c^{(t)}) \geq J(\Psi^{(t+1)}, c^{(t+1)})$$

Solution:

We have:

$$\Psi^{t+1}(x_i) = \underset{k \in \{1,\dots,K\}}{\arg\min} \left\| x_i - c_k^{(t)} \right\|^2$$

So,

$$\begin{aligned} J(\Psi^{(t)}, c^{(t)}) &= \frac{1}{n} \sum_{i=1}^{n} \left\| x_i - c^{(t)}_{\Psi^{(t)}(x_i)} \right\|^2 \\ &\geq \frac{1}{n} \sum_{i=1}^{n} \left\| x_i - c^{(t)}_{\Psi^{(t+1)}(x_i)} \right\|^2 \\ &= \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} z_i^k \left\| x_i - c_k^{(t)} \right\|^2 \end{aligned} \tag{1}$$

where $z = (z_i^k)_{(i,k) \in \{1,\dots,n\} \times \{1,\dots,K\}}$ is such that:

$$\forall (i,k) \in \{1,\dots,n\} \times \{1,\dots,K\} \quad z_i^k = \mathbf{1}\left(\Psi^{(t+1)}(x_i) = k\right)$$

We define:

$$\forall c = (c_1, \dots, c_K) \in \mathbb{R}^{K \times p} \quad \mathcal{L}(c) := \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} z_i^k \left\| x_i - c_k \right\|^2$$

It's straightforward that:

$$\mathcal{L}(c^{(t)}) = \frac{1}{n} \sum_{i=1}^{n} \left\| x_i - c^{(t)}_{\Psi^{(t+1)}(x_i)} \right\|^2 \tag{2}$$

and

$$\mathcal{L}(c^{(t+1)}) = \frac{1}{n}\sum_{i=1}^{n}\left\|x_i - c^{(t+1)}_{\Psi^{(t+1)}(x_i)}\right\|^2 \tag{3}$$

We wish to minimize $\mathcal{L}$ w.r.t c.

We have:

$$\forall k \in \{1,\dots,K\} \quad \nabla_{c_k}\mathcal{L}(c) = \frac{1}{n}\sum_{i=1}^{n} z_i^k \nabla_{c_k} \underbrace{\left(\|x_i - c\|^2\right)}_{g\circ f(c)} \tag{4}$$

where:

$$f : c \mapsto x_i - c \quad \text{and} \quad g : y \mapsto \|y\|^2$$

We have

$$\begin{aligned} d(g\circ f)_a(h) &= dg_{f(a)}\left(df_a(h)\right) \\ &= dg_{f(a)}(-h) \\ &= \langle 2f(a), -h\rangle \\ &= \langle -2(x_i - a), h\rangle \end{aligned}$$

Thus,

$$\nabla_{c_k} g\circ f(c) = -2(x_i - c) \tag{5}$$

From 4 and 5, we conclude that:

$$\forall k \in \{1,\dots,K\} \quad \nabla_{c_k}\mathcal{L}(c) = \frac{-2}{n}\sum_{i=1}^{n} z_i^k (x_i - c)$$

Therefore,

$$\begin{aligned}
\nabla_c \mathcal{L}(c) = 0 &\iff \forall k \in \{1, \dots, K\} \quad \nabla_{c_k} \mathcal{L}(c) = 0 \\
&\iff \forall k \in \{1, \dots, K\} \quad c_k = \frac{\sum_{i=1}^{n} z_i^k x_i}{\sum_{i=1}^{n} z_i^k} \\
&\iff \forall k \in \{1, \dots, K\} \quad c_k = c_k^{(t+1)} \\
&\iff c = c^{(t+1)}
\end{aligned}$$

We conclude that

$$\forall c = (c_1, \dots, c_K) \in \mathbb{R}^{K \times p} \quad \mathcal{L}(c) \geq \mathcal{L}(c^{(t+1)})$$

And therefore:

$$\mathcal{L}(c^{(t)}) \geq \mathcal{L}(c^{(t+1)}) \tag{6}$$

From equations 1, 2, 3 and 6, we conclude that:

$$J(\Psi^{(t)}, c^{(t)}) \geq \frac{1}{n} \sum_{i=1}^{n} \left\| x_i - c^{(t+1)}_{\Psi^{(t+1)}(x_i)} \right\|^2 = J(\Psi^{(t+1)}, c^{(t+1)})$$

Corollary 1.2.2. *The K-means algorithm 1 stops after a finite number of steps.*

Proof. The number of possible assignations is finite.

Thus, there exists t such that

$$J(\Psi^{(t)}, c^{(t)}) = J(\Psi^{(t+1)}, c^{(t+1)})$$

□

2 Gaussian Mixture Model

2.1 Introduction

Gaussian Mixture Models (GMMs) are a probabilistic model for representing normally distributed subpopulations within an overall population. Unlike single Gaussian models, which assume that all observations are drawn from a single distribution, GMMs consider a mixture of several Gaussian distributions, each with its own mean and variance, thus providing a more flexible approach to modeling data distributions. This flexibility makes GMMs particularly useful for modeling complex data sets with hidden or latent variables—where observations may originate from one of several unknown subpopulations.

Let's present a simple example to illustrate what we just said. The probability density represented on Figure 2 is akin to an average of two Gaussians. Thus, it is natural to use a mixture model and to introduce an hidden variable z, following a Bernoulli distribution defining which Gaussian the point is sampled from.

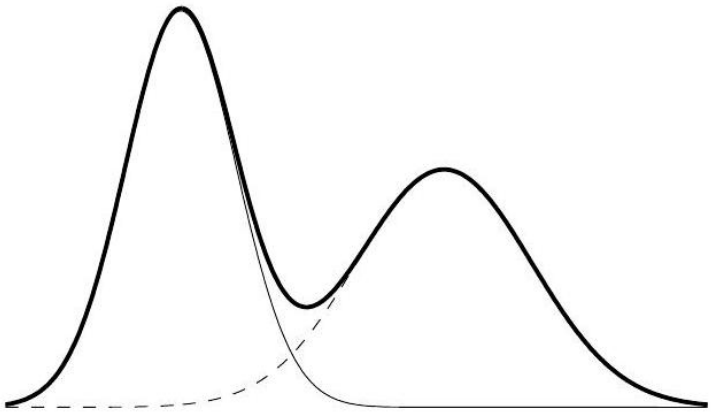


Figure 2: Average of two probability distributions of two Gaussian for which it is natural to introduce a mixture model

Thus we have : $z \in \{1, 2\}$ and $x \mid z = i \sim \mathcal{N}(\mu_i, \Sigma_i)$. The density $p(x)$ is a convex combination of normal density:

$$p(x) = p(x, z = 1) + p(x, z = 2) = p(x \mid z = 1)p(z = 1) + p(x \mid z = 2)p(z = 2)$$

It is a mixture model. It represents a simple way to model complicated phenomena.

2.2 Expectation Maximization Algorithm

2.2.1 Introducing the context

The Expectation-Maximization (EM) algorithm is an iterative method used for obtaining maximum likelihood estimates of parameters within statistical models. These models are characterized by their reliance on unobserved latent variables or hidden variables. Latent variables, denoted as z, are not directly observed but are inferred through the variables that are observed, denoted as x.

Within the context of the EM algorithm, we operate under the following framework:

- **Assumption:** We consider (x, z) to be random variables, where x represents the observed data, and z represents the hidden or latent variables (for example, unknown cluster centers in a clustering problem). The joint density function of x and z, $p_\theta(x, z)$, is parameterized by θ, indicating the model's parameters.
- **Objective:** The primary goal is to maximize the marginal likelihood of the observed data x with respect to the parameters θ, expressed as:

$$\max_\theta p_\theta(x) = \sum_z p_\theta(x, z)$$

This objective highlights the challenge posed by the presence of latent variables: maximizing the marginal likelihood is not straightforward due to the summation over the latent variable z. The summation introduces complexities, making the problem more challenging than optimizing a likelihood function without latent variables.

Specifically, taking the logarithm of the marginal likelihood does not lead to a simple convex optimization problem. The EM algorithm provides a robust method for addressing this challenge, facilitating the estimation of model parameters in the presence of latent variables.

2.2.2 The EM algorithm

The Dataset is composed of the pairs $(x_i, z_i)_{1 \leq i \leq n}$ where x_i is the observed data and z_i is the hidden data.

We make the assumption that the $(x_i, z_i)_{1 \leq i \leq n}$ are i.i.d.

The aim is to maximize the log likelihood:

$$\log p_\theta(x) = \sum_{i=1}^{n} \log \sum_{z_i} p_\theta(x_i, z_i)$$

We will use the following properties :

Proposition 2.2.1. *Jensen Inequality:*

1. *if $f : \mathbb{R} \to \mathbb{R}$ is convex and if X is an integrable random variable :*

$$\mathbb{E}_X(f(X)) \geq f(\mathbb{E}_X(X))$$

2. *if $f : \mathbb{R} \to \mathbb{R}$ is strictly convex, we have equality in the previous inequality if and only if $X =$ constant a.s.*

The EM algorithm is an iterative method for finding maximum likelihood estimates of parameters in statistical models, where the models depend on unobserved latent variables.

Consider for instance n observations $x_1, \ldots, x_n$ and the latent variables associateed with them $z_1, \ldots, z_n$.

We assume the pairs (x_i, z_i) to be independent and identically distributed.

For $(x, z) = (x_1, z_1, \ldots, x_n, z_n)$, the objective is to maximize:

$$\log(p(x; \theta)) = \sum_{i=1}^{n} \log \left(\sum_{z_i} p(x_i, z_i; \theta) \right)$$

For each $i \in \{1, \ldots, n\}$, we introduce a function $z_i \mapsto q(z_i)$ such that $q(z_i) \geq 0$ and $\sum_{z_i} q(z_i) = 1$ in the expression of the likelihood.

By conditioning on a latent variable z_i and using the Jensen inequality, we get a lower bound $\mathcal{L}(q, \theta)$ that depends on both q and θ.

$$\begin{aligned}
\log(p(x;\theta)) &= \sum_{i=1}^{n} \log\left(\sum_{z_i} p(x_i, z_i; \theta)\right) \\
&= \sum_{i=1}^{n} \log\left(\sum_{z_i} q(z_i) \frac{p(x_i, z_i; \theta)}{q(z_i)}\right) \\
&\geq \sum_{i=1}^{n} \sum_{z_i} q(z_i) \log\left(\frac{p(x_i, z_i; \theta)}{q(z_i)}\right) \\
&= \sum_{i=1}^{n} \underbrace{\mathbb{E}_{q(z_i)}\left[\log\left(\frac{p_\theta(x_i, z_i)}{q(z_i)}\right)\right]}_{\mathcal{L}(q(z_i),\theta)}
\end{aligned}$$

The EM algorithm can then be summarized as depicted in 2.

Algorithm 2 EM Algorithm

Require: Data set $X = \{x_1, \ldots, x_n\}$
Ensure: Optimal θ

1: **Initialization:** Choose initial parameters $\theta^{(0)}$.
2: Set iteration counter $i = 0$.
3: **while** not converged **do**
4: **E-step:** Update q to maximize the lower bound with respect to q.

$$q_{t+1} \in \arg\max_{q} \left(\mathcal{L}(q, \theta_t)\right)$$

5: **M-step:** Update θ to maximize the lower bound with respect to θ.

$$\theta_{t+1} \in \arg\max_{\theta} \left(\mathcal{L}(q_{t+1}, \theta)\right)$$

6: Check for convergence criterion (e.g., change in θ below a threshold).
7: $i \leftarrow i + 1$
8: **end while**
9: **return** Optimized parameters θ^*.

Exercise:

Show that the gap beween the marginal log-likelihood and the **lower bound** $\sum_{i=1}^{n} \mathcal{L}(q(z_i), \theta)$ is reduced to 0 when $q(z_i) = p_\theta(z_i \mid x_i)\ \forall i \in \{1, \dots, n\}$.

$p_\theta(z_i \mid x_i)$ is called the **posterior distribution**

Solution: Let $d = \log(p_\theta(x)) - \sum_{i=1}^{n} \mathcal{L}(q(z_i), \theta)$.

We have:

$$
\begin{aligned}
\mathrm{d} &= \log(p_\theta(x)) - \sum_{i=1}^{n} \mathcal{L}(q(z_i), \theta) \\
&= \sum_{i=1}^{n} \left(\log(p_\theta(x_i)) - \mathcal{L}(q(z_i), \theta)\right) \\
&= \sum_{i=1}^{n} \left(\sum_{z_i} q(z_i) \log(p_\theta(x_i)) - \sum_{z_i} q(z_i) \log\left(\frac{p_\theta(x_i, z_i)}{q(z_i)}\right)\right) \\
&= \sum_{i=1}^{n} \sum_{z_i} q(z_i) \left(\log(p_\theta(x_i)) - \log\left(\frac{p_\theta(x_i, z_i)}{q(z_i)}\right)\right) \\
&= \sum_{i=1}^{n} \sum_{z_i} q(z_i) \log\left(\frac{q(z_i)}{p_\theta(z_i|x_i)}\right) \\
&= \sum_{i=1}^{n} D_{\mathrm{KL}}\left(q(z_i) \,\|\, p_\theta(z_i|x_i)\right)
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
d = 0 &\iff \sum_{i=1}^{n} \underbrace{D_{\mathrm{KL}}\left(q(z_i) \,\|\, p_\theta(z_i|x_i)\right)}_{\geq 0} \\
&\iff \forall i \in \{1, \dots, n\} \quad D_{\mathrm{KL}}\left(q(z_i) \,\|\, p_\theta(z_i|x_i)\right) \\
&\iff \forall i \in \{1, \dots, n\} \quad q(z_i) = p_\theta(z_i|x_i)
\end{aligned}
$$

Therefore, maximizing the lower bound $\log(p_\theta(x))$ with respect to q consists in taking the posterior distributions $\forall i \in \{1, \ldots, n\} \quad q(z_i) = p_\theta(z_i|x_i)$.

Let's recall the expression of the lower bound:

$$\mathcal{L}(q, \theta) = \sum_{i=1}^{n} \left(\sum_{z_i} q(z_i) \log p_\theta(x_i, z_i) - \sum_{z_i} q(z_i) \log q(z_i) \right)$$

Since $\sum_{z_i} q(z_i) \log q(z_i)$ doesn't depend on θ, maximizing the lower bound with respect to θ is equivalent to maximizing w.r.t θ the expected value of the complete log likelihood function $\log(p_{\theta_t}(x, z))$.

The final recipe is given in algorithm 3. It consists in the following steps:

1. Compute the probability of Z given X : $p_{\theta_t}(z \mid x)$ (Corresponding to $q_{t+1} = \arg\max_q \mathcal{L}(q, \theta_t)$)
2. Write the complete loglikelihood $l_c = \log(p_{\theta_t}(x, z))$.
3. **E-Step**: Calculate the expected value of the complete log likelihood function, with respect to the conditional distribution of Z given X under the current estimate of the parameter $\theta_t : \mathbb{E}_{Z|X}(l_c)$.
4. **E-Step**: Find θ_{t+1} by maximizing $\mathcal{L}(q_{t+1}, \theta)$ with respect to θ.

Algorithm 3 EM algorithm

Require: Observations $x_1, \ldots, x_n$
Ensure: Optimal θ

Initialize $\theta^{(0)}$
while not converged **do**
    E-step: $q(z) = p(z|x; \theta^{(i-1)})$
    M-step: $\theta^{(i)} = \arg\max_\theta \mathbb{E}_q[\log p(x, z; \theta)]$
    $i \leftarrow i + 1$
end while

Remarks:

- It is an ascent algorithm, indeed it goes up in term of likelihood (compare to before where we were descending along the distortion).
- The sequence of log-likelihoods converges.

- It does not converge to a global maximum but rather to a local maximum because we are dealing here with a non-convex problem. An illustration is given in Figure 3

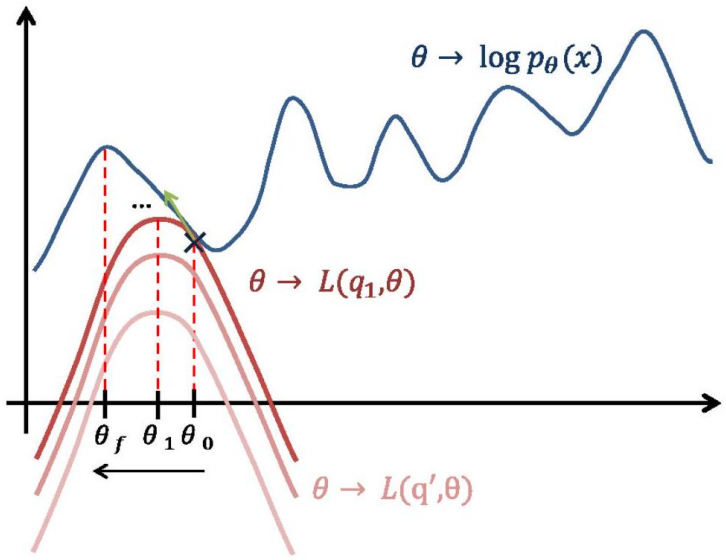


Figure 3: An illustration of the EM algorithm that converges to a local minimum.

- As it was already the case for K-means, we reiterate the result in order to be more confident. Then we keep the one with the highest likelihood.
- Because EM gives a local maximum, it is clever to choose a θ_0 relatively close to the final solution. For Gaussian mixtures, it is quite usual to initiate EM by a K-means.

Exercise:

Suppose we have observations n observations $x_1, \dots x_n$ in $\mathbb{R}^p$.

We make the assumption of the existence of latent variables $z_1, \dots, z_n$ from a multinomial distribution with K possible outcomes.

i.e:

$\forall i \in \{1, \dots, n\} \quad x_i \in \mathbb{R}^p, z_i \sim \mathcal{M}(1, \pi_1, \dots, \pi_K)$ and $(x_i \mid z_i = j) \sim \mathcal{N}(\mu_j, \Sigma_j)$.

Here we have $\theta = (\pi, \mu, \Sigma)$.

Use the EM algorithm to estimate θ.

Solution:

1. **Calculation of the posterior distributions** $p_\theta(z_i \mid x_i)$:
   We write $p_\theta(x_i)$:

$$\begin{aligned} p_\theta(x_i) &= \sum_{z_i} p_\theta(x_i, z_i) = \sum_{z_i} p_\theta(x_i \mid z_i)\, p_\theta(z_i) \\ &= \sum_{j=1}^{K} p_\theta(x_i \mid z_i = j)\, p_\theta(z_i = j) \end{aligned}$$

   Then we use the Bayes formula to estimate $p_\theta(z \mid x)$:

$$\begin{aligned} p_\theta(z_i = j \mid x_i) &= \frac{p_\theta(x_i \mid z_i = j)\, p_\theta(z_i = j)}{p_\theta(x_i)} \\ &= \frac{\pi_j \mathcal{N}(x_i \mid \mu_j, \Sigma_j)}{\sum_{j'} \pi_{j'} \mathcal{N}\left(x_i \mid \mu'_j, \Sigma'_j\right)} \\ &= \tau_i^j(\theta). \end{aligned}$$

   We recall that $\mathcal{N}(x_i \mid \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{p}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$.
   Suppose that we are at the t-th iteration of the algorithm.

2. **Complete likelihood**
   Let's write the complete likelihood of the problem.

$$
\begin{aligned}
l_{c,t} = \log p_{\theta_t}(x, z) &= \sum_{i=1}^{n} \log p_{\theta_t}\left(x_i, z_i\right) \\
&= \sum_{i=1}^{n} \log\left(p_{\theta_t}\left(z_i\right) p_{\theta_t}\left(x_i \mid z_i\right)\right) \\
&= \sum_{i=1}^{n} \log\left(p_{\theta_t}\left(z_i\right)\right) + \log\left(p_{\theta_t}\left(x_i \mid z_i\right)\right) \\
&= \sum_{i=1}^{n} \sum_{j=1}^{K} z_i^j \log\left(\pi_{j,t}\right) \\
&\quad + \sum_{i=1}^{n} \sum_{j=1}^{K} z_i^j \log\left(\mathcal{N}\left(x_i \mid \mu_{j,t}, \Sigma_{j,t}\right)\right)
\end{aligned}
$$

where $z_i^j \in \{0, 1\}$ with $z_i^j = 1$ if $z_i = j$ and 0 otherwise.

3. **E-Step** In the E-step, we compute the expectation of the complete log-likelihood with respect to the conditional distribution of the latent variables Z given the observed data X. This involves replacing the indicator variables z_i^j with their expected values:

$$
\mathbb{E}_{Z|X}\left(z_i^j\right) = p_{\theta_t}(z = j|x_i) = \tau_i^j(\theta_t),
$$

where τ_i^j represents the posterior probability that observation x_i belongs to component j, given the current parameter estimates. By substituting z_i^j with τ_i^j, we obtain the expected complete log-likelihood:

$$
\mathbb{E}_{Z|X}(l_{c,t}) = \sum_{i=1}^{n} \sum_{j=1}^{K} \tau_i^j \log(\pi_{j,t}) + \sum_{i=1}^{n} \sum_{j=1}^{K} \tau_i^j \log\left(\mathcal{N}(x_i|\mu_{j,t}, \Sigma_{j,t})\right).
$$

4. **M-Step**

   For the M-step, we this need to maximize:

$$
\begin{aligned}
&\sum_{i=1}^{n} \sum_{j=1}^{K} \tau_i^j \log\left(\pi_{j,t}\right) + \sum_{i=1}^{n} \sum_{j=1}^{K} \tau_i^j \left[\log\left(\frac{1}{(2\pi)^{\frac{p}{2}}}\right) + \log\left(\frac{1}{|\Sigma_{j,t}|^{\frac{1}{2}}}\right)\right. \\
&\left. -\frac{1}{2}\left(x_i - \mu_{j,t}\right)^T \Sigma_{j,t}^{-1}\left(x_i - \mu_{j,t}\right)\right]
\end{aligned}
$$

We want to maximize the previous equation with respect to $\theta_t = (\Pi_t, \mu_t, \Sigma_t)$

As the sum is separated into two terms independent along the variables we can first maximize with respect to π_t :

$$\max_{\Pi} \sum_{j=1}^{k} \sum_{i=1}^{n} \tau_i^j \log \pi_j \quad \Rightarrow \quad \pi_{j,t+1} = \frac{\sum_{i=1}^{n} \tau_i^j}{\sum_{i=1}^{n} \sum_{j'=1}^{k} \tau_i^{j'}} = \frac{1}{n} \sum_{i=1}^{n} \tau_i^j$$

We can now maximize with respect to μ_t and Σ_t. By computing the gradient along the $\mu_{j,t}$ and along the $\Sigma_{j,t}$, we obtain :

$$\mu_{j,t+1} = \frac{\sum_i \tau_i^j x_i}{\sum_i \tau_i^j}$$

$$\Sigma_{j,t+1} = \frac{\sum_i \tau_i^j \left(x_i - \mu_{j,t+1}\right)\left(x_i - \mu_{j,t+1}\right)^T}{\sum_i \tau_i^j}$$

The M-step in the EM algorithm corresponds to the estimation of means step in K-means. Note that the value of τ_i^j in the expressions above are taken for the parameter values of the previous iterate, i.e., $\tau_i^j = \tau_i^j(\theta_t)$.

Possible forms for Σ_j

- isotropic: $\Sigma_j = \sigma_j^2 \mathrm{Id}, 1$ parameter, the cluster is a sphere.
- diagonal: Σ_j is a diagonal matrix, d parameters, the cluster is an ellipse oriented along the axis.
- general: $\Sigma_j, \frac{d(d+1)}{2}$ parameters, the cluster is an ellipse.

3 Distance Metrics

3.1 Motivation:

Despite all of its virtues, correlation suffers from some critical limitations as a measure of codependence. In this section, we overcome those limitations by reviewing information theory concepts that underlie many modern marvels, such as the internet, mobile phones, file compression, video streaming, or encryption. None of these inventions would have been possible if researchers had not looked beyond correlations to understand codependency.

As it turns out, information theory in general, and the concept of Shannon's entropy in particular, also have useful applications in finance. The key idea behind entropy is to quantify the amount of uncertainty associated with a random variable.

In this section, we review concepts that are used throughout ML in a variety of settings, including:

- Defining the objective function in decision tree learning.
- Defining the loss function for classification problems.
- Evaluating the distance between two random variables.
- Comparing clusters
- Feature selection

3.2 A correlation-based metric

Correlation is a useful measure of linear codependence. For example, we could use correlations to identify clusters of highly interrelated securities. But before we can do that, we need to address a technical problem: correlation is not a metric, because it does not satisfy nonnegativity and triangle inequality conditions. Metrics are important because they induce an intuitive topology on a set. Without that intuitive topology, comparing non-metric measurements of codependence can lead to rather incoherent outcomes. For instance, the difference between correlations $(0.9, 1.0)$ is the same as $(0.1, 0.2)$, even though the former involves a greater difference in terms of codependence.

Proposition 3.2.1. *Consider two random vectors* X, Y *of size* T*, and a correlation estimate* $\rho[X, Y]$*, with the only requirement that:*

$\sigma[X, Y] = \rho[X, Y]\sigma[X]\sigma[Y]$*, where* $\sigma[X, Y]$ *is the covariance between the two vectors and* $\sigma[.]$ *is the standard deviation.*

(Pearson's correlation is one of several correlation estimates that satisfy these requirements.)

The measure $\mathrm{d}_\rho[X, Y] = \sqrt{\frac{1}{2}(1 - \rho[X, Y])}$ *is a metric.*

Exercise:

Prove that $\mathrm{d}_\rho[X,Y] = \sqrt{\frac{1}{2}(1-\rho[X,Y]))}$ is a metric.

Solution: To prove that statement:

- First consider that the Euclidean distance between the two vectors is $d[X,Y] = \sqrt{\sum_{t=1}^{T}(X_t - Y_t)^2}$.
- Second, we z-standardize those vectors as $x = (X - \bar{X})/\sigma[X], y = (Y - \bar{Y})/\sigma[Y]$, where $\bar{X}$ is the mean of X, and $\bar{Y}$ is the mean of Y. Consequently, $\rho[x,y] = \rho[X,Y]$.
- Third, we derive the Euclidean distance $d[x,y]$ as

$$\begin{aligned} d[x,y] &= \sqrt{\sum_{t=1}^{T}(x_t - y_t)^2} \\ &= \sqrt{\sum_{t=1}^{T}x_t^2 + \sum_{t=1}^{T}y_t^2 - 2\sum_{t=1}^{T}x_t y_t} = \sqrt{T + T - 2T\sigma[x,y]} \\ &= \sqrt{2T(1 - \underbrace{\rho[x,y]}_{=\rho[X,Y]})} = \sqrt{4T}d_\rho[X,Y]. \end{aligned}$$

The implication is that $d_\rho[X,Y]$ is a linear multiple of the Euclidean distance between the vectors $\{X,Y\}$ after z-standardization ($d[x,y]$), hence it inherits the true-metric properties of the Euclidean distance.

The metric $d[x,y]$ has the property that it is normalized, $d_\rho[X,Y] \in [0,1]$, because $\rho[X,Y] \in [-1,1]$. Another property is that it deems more distant two random variables with negative correlation than two random variables with positive correlation, regardless of their absolute value. This property makes sense in many applications. For example, we may wish to build a long-only portfolio, where holdings in negatively-correlated securities can only offset risk, and therefore should be treated as different for diversification purposes.

3.3 Information Theory based metrics

The notion of correlation presents three important caveats. First, it quantifies the linear codependency between two random variables. It neglects nonlinear relationships. Second, correlation is highly influenced by outliers. Third, its application beyond the multivariate Normal case is questionable. We may compute the correlation between any two real variables, however that correlation is typically meaningless unless the two variables follow a bivariate Normal distribution. To overcome these caveats, we need to introduce a few information-theoretic concepts.

1. **Marginal and Joint Entropy**

Definition 3.1. *Let* X *be a discrete random variable that takes a value x from the set S_X with probability $p[x]$. The entropy of* X *is defined as*

$$H[X] = -\sum_{x \in S_X} p[x] \log[p[x]]$$

We will follow the convention that $0 \log[0] = 0$, since $\lim_{p \to 0^+} p \log[p] = 0$. The value $1/p[x]$ measures how surprising an observation is, because surprising observations are characterized by their low probability. Entropy is the expected value of those surprises, where the $\log[\cdot]$ function prevents that $p[x]$ cancels $1/p[x]$ and endows entropy with desirable mathematical properties. Accordingly, entropy can be interpreted as the amount of uncertainty associated with X. Entropy is zero when all probability is concentrated in a single element of S_X. Entropy reaches a maximum at $\log[\|S_X\|]$ when X is distributed uniformly, $p[x] = 1/\|S_X\|, \forall x \in S_X$.

Definition 3.2. *Let* Y *be a discrete random variable that takes a value y from the set S_Y with probability $p[y]$. Random variables* X *and* Y *do not need to be defined on the same probability space. The joint entropy of* X *and* Y *is*

$$H[X, Y] = -\sum_{x,y \in S_X \times S_Y} p[x, y] \log[p[x, y]]$$

In particular, we have that $H[X, Y] = H[Y, X]$, $H[X, X] = H[X]$, $H[X, Y] \geq \max\{H[X], H[Y]\}$, and $H[X, Y] \leq H[X] + H[Y]$.

It is important to recognize that Shannon's entropy is finite only for discrete random variables. In the continuous case, one should use the limiting density of discrete points (LDDP), or discretize the random variable.

2. **Conditional Entropy**

Definition 3.3. *The conditional entropy of* X *given* Y *is defined as*

$$\begin{aligned} H[X \mid Y] &= H[X, Y] - H[Y] \\ &= -\sum_{y \in S_Y} p[y] \sum_{x \in S_X} p[x \mid Y = y] \log[p[x \mid Y = y]] \end{aligned}$$

where $p[x \mid Y = y]$ is the probability that X takes the value x conditioned on Y having taken the value y.

Proposition 3.3.1. *$H[X \mid X] = 0$, and $H[X] \geq H[X \mid Y]$.*

3. **Kullback-Leibler Divergence**

Definition 3.4. *Let* p *and* q *be two discrete probability distributions defined on the same probability space. The Kullback-Leibler (or KL) divergence between* p *and* q *is:*

$$D_{KL}[p \| q] = -\sum_{x \in S_X} p[x] \log \left[\frac{q[x]}{p[x]}\right] = \sum_{x \in S_X} p[x] \log \left[\frac{p[x]}{q[x]}\right]$$

where $q[x] = 0 \Rightarrow p[x] = 0$.

Intuitively, this expression measures how much p diverges from a reference distribution q.

The KL divergence is not a metric: although it is always nonnegative ($D_{KL}[p \| q] \geq 0$), it violates the symmetry ($D_{KL}[p \| q] \neq D_{KL}[q \| p]$) and triangle inequality conditions.

KL divergence is widely used in variational inference.

Exercise:

Let p and q be two finite distributions on $\mathcal{Y}$.

The **Kullback-Leibler divergence Kullback-Leibler divergence**

between p and q is:

$$\begin{aligned} D_{\mathrm{KL}}(p \parallel q) &= \mathbb{E}_{Y\sim p}\left[\log \frac{p(Y)}{q(Y)})\right] \\ &= \sum_{y\in\mathcal{Y}} p(y) \log \frac{p(y)}{q(y)} \\ &= \sum_{y\in\mathcal{Y}} \frac{p(y)}{q(y)} \left(\log \frac{p(y)}{q(y)}\right) q(y) \\ &= \mathbb{E}_{Y\sim q}\left[\frac{p(Y)}{q(Y)} \log \frac{p(Y)}{q(Y)})\right] \end{aligned}$$

Show that

$$D_{\mathrm{KL}}(p \parallel q) \geq 0$$

Solution:

$$\begin{aligned} D_{\mathrm{KL}}(p \parallel q) &= \mathbb{E}_{Y\sim q}\left[\frac{p(Y)}{q(Y)} \log \frac{p(Y)}{q(Y)})\right] \quad \text{(definition)} \\ &\geq \mathbb{E}_{Y\sim q}\left[\frac{p(Y)}{q(Y)}\right] \log \mathbb{E}\left[\frac{p(Y)}{q(Y)})\right] \quad \text{(Jensen Inequality)} \\ &= 0 \quad (\text{since} \quad \mathbb{E}_{Y\sim q}\left[\frac{p(Y)}{q(Y)}\right] = 1) \end{aligned}$$

with $D_{\mathrm{KL}}(p \parallel q) = 0$ iff there is an equality in Jensen's inequality, which means $p = q$.

4. **The Cross-Entropy**

Definition 3.5. *Let p and q be two discrete probability distributions defined on the same probability space. Cross-entropy between p and q is*

$$H_C[p\|q] = -\sum_{x\in S_X} p[x] \log[q[x]] = H[X] + D_{KL}[p\|q]$$

Cross-entropy can be interpreted as the uncertainty associated with X, where we evaluate its information content using a wrong distribution q rather than the true distribution p. Cross-entropy is a popular scoring function in classification problems, and it is particularly meaningful in financial applications.

5. **Mutual Information**

Definition 3.6. *Mutual information is defined as the decrease in uncertainty (or informational gain) in* X *that results from knowing the value of* Y*:*

$$I[X,Y] = H[X] - H[X \mid Y] = H[X] + H[Y] - H[X,Y]$$

We have the following property:

Proposition 3.3.2.

$$I[X,Y] = \mathrm{E}_X\left[D_{KL}[p[y \mid x] \| p[y]]\right]$$

Exercise:

Prove that:

$$I[X,Y] = \mathrm{E}_X\left[D_{KL}[p[y \mid x] \| p[y]]\right]$$

Solution:

$$\begin{aligned}
I[X,Y] &= H[X] - H[X \mid Y] = H[X] + H[Y] - H[X,Y] \\
&= \sum_{x \in S_X} \sum_{y \in S_Y} p[x,y] \log \left[\frac{p[x,y]}{p[x]p[y]}\right] \\
&= D_{KL}[p[x,y] \| p[x]p[y]] = \sum_{y \in S_Y} p[y] \sum_{x \in S_X} p[x \mid y] \log \left[\frac{p[x \mid y]}{p[x]}\right] \\
&= \mathrm{E}_Y\left[D_{KL}[p[x \mid y] \| p[x]]\right] = \sum_{x \in S_X} p[x] \sum_{y \in S_Y} p[y \mid x] \log \left[\frac{p[y \mid x]}{p[y]}\right] \\
&= \mathrm{E}_X\left[D_{KL}[p[y \mid x] \| p[y]]\right]
\end{aligned}$$

From the above we can deduce some properties:

Proposition 3.3.3. *We have the following properties:*

- $I[X,Y] \geq 0, I[X,Y] = I[Y,X]$
- $I[X,X] = H[X]$
- *When* X *and* Y *are independent,* $p[x,y] = p[x]p[y]$*, hence* $I[X,Y] =$

0.

- *An upper boundary is given by* $I[X,Y] \leq \min\{H[X], H[Y]\}$.
- *The grouping property:*

$$I[X,Y,Z] = I[X,Y] + I[(X,Y),Z]$$

where (X,Y) *represents the joint distribution of* X *and* Y*. Since* X*,* Y*, and* Z *can themselves represent joint distributions, the above property can be used to decompose mutual information into simpler constituents. This makes mutual information a useful similarity measure in the context of agglomerative clustering algorithms and forward feature selection.*

Remark:

Mutual information is not a metric, because it does not satisfy the triangle inequality:

$I[X,Z]I[X,Y] + I[Y,Z]$.

6. **Variation of Information**

Definition 3.7. *The Variation of information is defined as:*

$$VI[X,Y] = H[X \mid Y] + H[Y \mid X]$$

Proposition 3.3.4.

$$VI[X,Y] = H[X,Y] - I[X,Y]$$

Proof.

$$\begin{aligned} VI[X,Y] &= H[X,Y] - I[X,Y] \\ &= H[X] + H[Y] - 2I[X,Y] \\ &= 2H[X,Y] - H[X] - H[Y] \\ &= H[X,Y] - I[X,Y] \end{aligned}$$

□

This measure can be interpreted as the uncertainty we expect in one variable if we are told the value of the other. It has a lower bound in $VI[X,Y] = 0 \Leftrightarrow X = Y$, and an upper bound in $VI[X,Y] \leq H[X,Y]$. Variation of information is a metric, because it satisfies the axioms:

- Nonnegativity: $VI[X,Y] \geq 0$
- Symmetry,:$VI[X,Y] = VI[Y,X]$
- Triangle inequality: $VI[X,Z] \leq VI[X,Y] + VI[Y,Z]$

However, because $H[X,Y]$ is a function of the sizes of S_X and S_Y, $VI[X,Y]$ does not have a firm upper bound.

This is problematic when we wish to compare variations of information across different population sizes.

The following quantity is a metric bounded between zero and one for all pairs (X,Y)

Definition 3.8. *We define the normalized variation of information as follows:*

$$\widetilde{VI}[X,Y] = \frac{VI[X,Y]}{H[X,Y]} = 1 - \frac{I[X,Y]}{H[X,Y]}$$

Following [?], the following quantity is an alternative metric:

Definition 3.9. *We define* $\widetilde{\widetilde{VI}}$ *as follows:*

$$\widetilde{\widetilde{VI}} = \frac{\max\{H[X \mid Y], H[Y \mid X]\}}{\max\{H[X], H[Y]\}}$$

We have:

Proposition 3.3.5. *For all pairs* (X,Y)*:*

$$\widetilde{\widetilde{VI}} = 1 - \frac{I[X,Y]}{\max\{H[X], H[Y]\}} \leq \widetilde{VI}[X,Y]$$

As a summary, figure 5 provides a visual representation of how these concepts are interrelated:

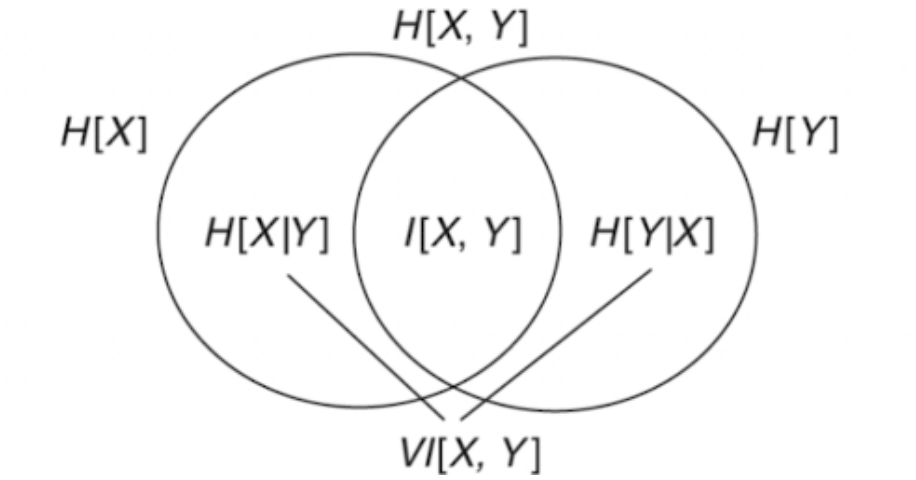


Figure 4: Correspondence between joint entropy, marginal entropies, conditional entropies, mutual information, and variation of information.

4 Application: Solving the substitution effect in Feature Importance Analysis

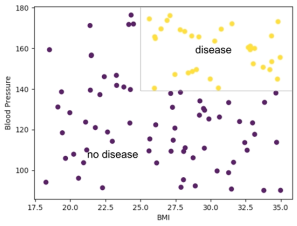
4.1 The Decision Tree algorithm

4.1.1 A brief introduction

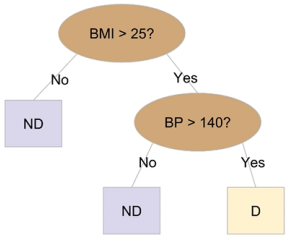
1. **The intuition:**

   The **Decision Tree** (DT) algorithm is an attractive model if we care about interpretability.

   As the name decision tree suggests, we can think of this model as breaking down our data by making a decision based on asking a series of questions.



(a) Features and Labels



(b) Decision Tree graph

2. **A high level description of the algorithm**

   The DT algorithm is basically just a bunch of nested if-statements on the input features (also called **attributes**) in the training dataset.

   The decision algorithm:

   - We start at the tree root (with the whole dataset)
   - Then we split the dataset on the attribute that results in the largest **Information Gain** (IG).
   - We iterate the splitting procedure at each child node until the leaves are pure (which means that the samples at the leaves belong to the same class)
   - A very deep tree is prone to overfitting. To avoid that, we set a limit for the maximal depth of the tree.

3. **Building the decision tree**

   - First, we need to define an objective function that we want to optimize (Information Gain).
   - Then, at each iteration, two challenges arise when trying to choose the best split.
     - How do we choose the best attribute responsible for the split ?
     - How do we choose the threshold when splitting based on the "best attribute" ?

4.1.2 Introducing the Information Gain

1. **The entropy of a Bernoulli distribution**

   Let Y be a random variable taking values in the finite set $\mathcal{Y}$.

   Let's denote $p(y) = \mathbb{P}(Y = y)$

In informarion theory, the quantity $I(y) = \log_2(\frac{1}{p(y)})$ can be interpreted as a quantity of information carried by the occurence of y (sometimes called *self-information*)

So, the **entropy** is defined as the expected amount of information of the random variable Y:

$$H(Y) = \mathbb{E}_{p(y)}[I(Y)] = -\sum_{y \in \mathcal{Y}} p(y) \log_2(p(y))$$

Let's now focus on a Bernoulli distribution $Y \sim \mathcal{B}(p)$

As we have defined earlier, the entropy of Y as a function of p is:

$$H(p) = -p \log_2(p) - (1-p) \log_2(1-p)$$

As show in figure 5, $p = 0.5$ yields maximum entropy.

So, the entropy is a measure of how much information we get from finding out the value of the random variable.

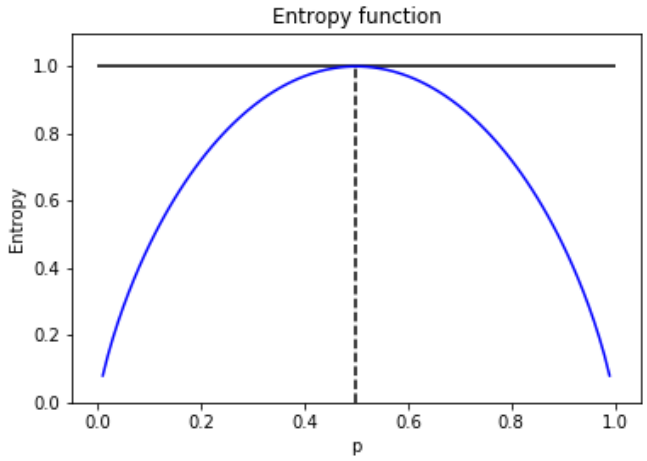


Figure 5: Entropy of the Bernoulli distribution

Exercise:

Prove the following inequalities:

$$H(Y) \geq 0 \quad \text{with equality if Y is constant a.s}$$
$$H(Y) \leq \log_2(\text{Card}(\mathcal{Y}))$$

Solution:

First point:

- Since $\forall y \in \mathcal{Y} \quad p(y) \leq 1$ then:

$$H(Y) = \sum_{y \in \mathcal{Y}} \underbrace{-p(y)\log_2(p(y))}_{\geq 0} \geq 0$$

- With equality iff $\forall y \in \mathcal{Y} \quad p(y)\log_2(y) = 0$, which proves the first point.

Second point:

- We have for all distributions p and q:

$$\begin{aligned} D_{\mathrm{KL}}(p \,\|\, q) &= \sum_{y \in \mathcal{Y}} p(y) \log \frac{p(y)}{q(y)} \quad \text{(definition)} \\ &= -\sum_{y \in \mathcal{Y}} p(y)\log(q(y)) - \left(-\sum_{y \in \mathcal{Y}} p(y)\log(p(y)) \right) \\ &= -\sum_{y \in \mathcal{Y}} p(y)\log(q(y)) - H(Y) \end{aligned}$$

- Hence, by choosing $\forall y \in \mathcal{Y} \quad q^*(y) = \frac{1}{\mathrm{Card}(\mathcal{Y})}$:

$$H(Y) = \log(\mathrm{Card}(\mathcal{Y})) - \underbrace{D_{\mathrm{KL}}(p \,\|\, q^*)}_{\geq 0} \leq \log_2(\mathrm{Card}(\mathcal{Y}))$$

2. **The information gain**

Definition 4.1. *We define the* ***information gain*** *at a split on the attribute c as follows:*

$$IG(D_p, c) = I(D_p) - \frac{N_{left}}{N_p} I(D_{left}) - \frac{N_{right}}{N_p} I(D_{right})$$

- D_p *refers to the dataset of the parent.*
- D_{left} *and* D_{right} *are the datasets of the left and right child nodes. (For simplicity, most libraries only implement binary decision trees).*
- I *is the* ***impurity*** *measure.*

- *N_p is the total number of samples at the parent node.*
- *N_{left} and N_{right} are the total number of samples at the right and left child nodes.*

3. **Different impurity measures:**

   The three impurity measures or splitting criteria that are commonly used in binary decision trees are **Gini Impurity** (I_G), **Entropy** (I_H), and the **Classification Error** (I_E).

   Let's denote p_k^m the proportion of the samples that belong to the class $m \in \{1, \ldots, M\}$ for a particular node k. We define the above stated impurities, shown in figure 6, as follows:

$$I_G(k) = -\sum_{m=1}^{M} p_k^m (1 - p_k^m)$$
$$I_H(k) = -\sum_{m=1}^{M} p_k^m \log_2(p_k^m)$$
$$I_E(k) = 1 - \max_{1 \leq m \leq M} p_k^m$$

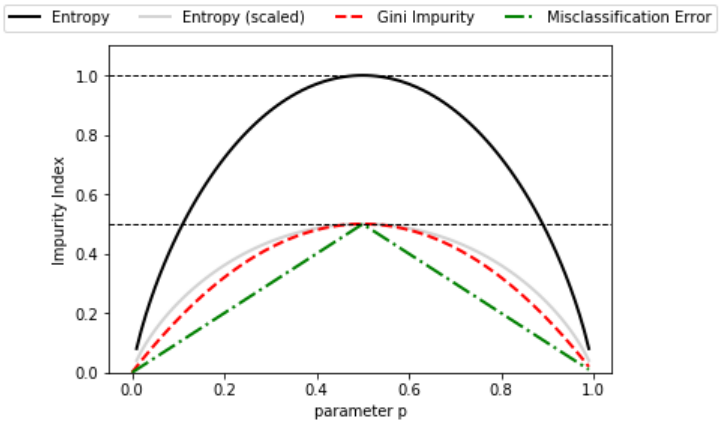


Figure 6: Different impurities for Information Gain

4.2 Application: Feature Selection on synthetic data

4.2.1 The problem

Consider a binary random classification problem composed of forty features, where five are informative, thirty are redundant, and five are noise.

- **Informative features** (marked with the I prefix) are those used to generate labels.
- **Redundant features** (marked with the R prefix) are those that are formed by adding Gaussian noise to a randomly chosen informative feature.
- **Noise features** (marked with the N prefix) are those that are not used to generate labels.

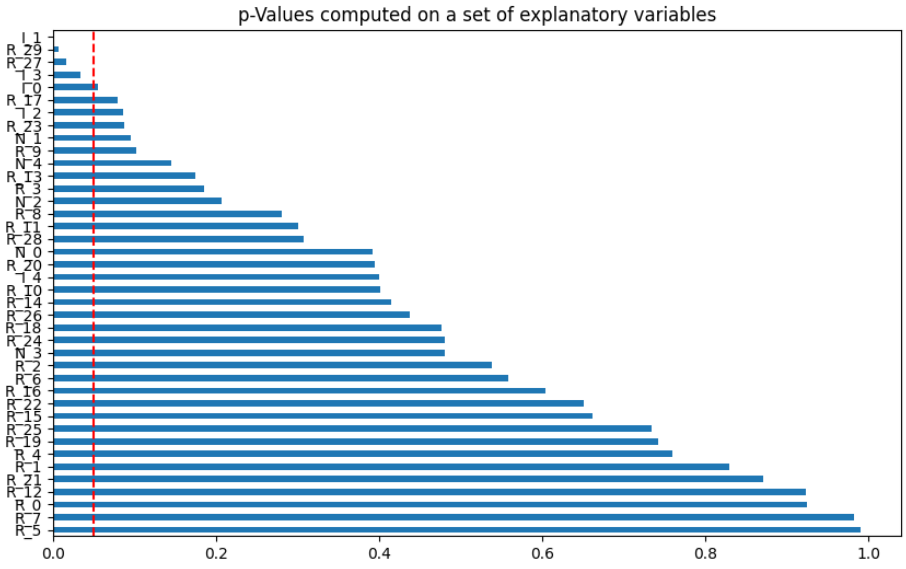


Figure 7: p-values from a logit regression

Figure 7 presents the p-values from a logit regression on selected features, where horizontal bars indicate the p-values, and a vertical dashed line denotes the 5% significance threshold. Remarkably, only I_1, R_{29}, R_{27}, I_3 out of thirty-five non-noise features are statistically significant at this level.

Interestingly, noise features are often misinterpreted as important, with fourteen non-noise features ranked among the least important, challenging the reliability of these p-values due to several caveats:

1. **Assumption Sensitivity:** P-values depend heavily on underlying assumptions. Inaccuracies here can lead to false positives (low p-value with a true coefficient of zero) or false negatives (high p-value with a nonzero true coefficient).
2. **Multicollinearity Issues:** In the presence of multicollinearity among explanatory variables, p-values become unreliable. Traditional regression struggles to differentiate among correlated variables, distorting the interpretation of their significance.
3. **Irrelevant Probability Estimation:** P-values estimate the probability of observing data as extreme as or more than the estimated coefficient $\hat{\beta}$ under the null hypothesis H_0. This contrasts with the often more relevant probability of H_0 being true given $\hat{\beta}$, which requires Bayesian approaches and additional assumptions.
4. **In-Sample Bias:** P-values assess significance using the same sample for both coefficient estimation and significance testing, potentially overfitting and misrepresenting out-of-sample explanatory power. This in-sample testing can lead to p-hacking and false discoveries.

In essence, p-values necessitate multiple assumptions (#1), produce estimates sensitive to multicollinearity (#2), calculate a probability of limited practical interest (#3), and may lack out-of-sample generalizability (#4).

The theoretical transparency offered by classical methods in attributing significance is compromised in practice by these issues. It suggests a potential benefit from integrating modern computational techniques to address these challenges.

4.2.2 Feature Importance Analysis

4.2.3 Introduction

In our approach to assessing feature importance, we employ two distinct methods: one that depends on in-sample measures, such as mean decrease impurity, and another that utilizes out-of-sample measures, like mean decrease accuracy. Both methods, however, encounter challenges due to substitution effects, where the importance of one feature is overshadowed by the presence of other, related features.

To address the issue of substitution effects, we implement two strategies:

- We could employ Principal Component Analysis (PCA) to project the features onto an orthogonal basis. This technique helps reduce the effects of linear substitution by transforming the features into a set of orthogonal vectors.
- As an alternative, clustering the observation matrix and then applying both Mean Decrease Accuracy (MDA) and Mean Decrease Impurity (MDI) at the cluster level is viable. This method enables the evaluation of feature importance within homogeneous groups, thereby enhancing our insight into the significance of features across various data segments.

4.2.4 Feature Importance with substitution effects

- **Mean Decrease Impurity**

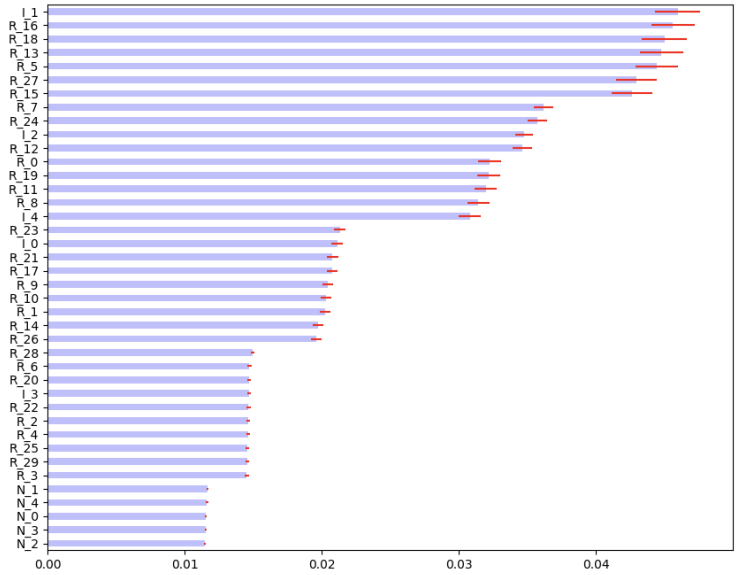


Figure 8: MDI

Mean decrease impurity (MDI) is a fast, explanatory-importance (in-sample, IS) method specific to tree-based classifiers, like RF. At each node of each decision tree, the selected feature splits the subset it received in such a way that impurity is decreased. Therefore, we can derive for each decision tree how much of the overall impurity decrease can be assigned to each feature. And given that we have a forest of trees, we can average those

values across all estimators and rank the features accordingly. See [**?**] for a detailed description.

There are some important considerations we must keep in mind when working with MDI:

1. The procedure is obviously In Sample. Every feature will have some importance, even if they have no predictive power whatsoever.
2. MDI cannot be generalized to other non-tree based classifiers.
3. By construction, MDI has the nice property that feature importances add up to 1, and every feature importance is bounded between 0 and 1.
4. The method does not address substitution effects in the presence of correlated features. MDI dilutes the importance of substitute features, because of their interchangeability: The importance of two identical features will be halved, as they are randomly chosen with equal probability.
5. [**?**] show experimentally that MDI is biased towards some predictor variables. [**?**] argue that, in case of single decision trees, this bias is due to an unfair advantage given by popular impurity functions toward predictors with a large number of categories.

- **Mean Decrease Accuracy**

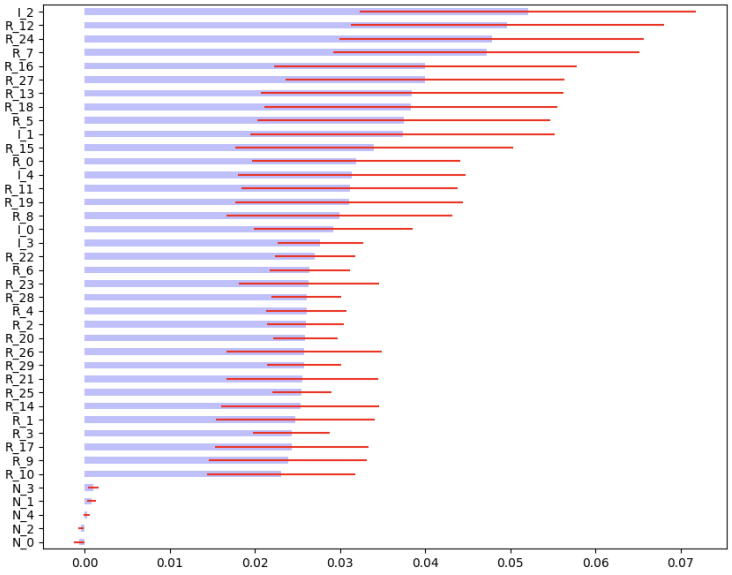


Figure 9: MDA

Mean decrease accuracy (MDA) is a slow, predictive-importance (out-of-sample, OOS) method. First, it fits a classifier; second, it derives its

performance OOS according to some performance score (accuracy, negative log-loss, etc.); third, it permutates each column of the features matrix (X), one column at a time, deriving the performance OOS after each column's permutation. The importance of a feature is a function of the loss in performance caused by its column's permutation. Some relevant considerations include:

1. This method can be applied to any classifier, not only tree-based classifiers.
2. MDA is not limited to accuracy as the sole performance score. For example, in the context of meta-labeling applications, we may prefer to score a classifier with F1 rather than accuracy. That is one reason a better descriptive name would have been "permutation importance." When the scoring function does not correspond to a metric space, MDA results should be used as a ranking.
3. Like MDI, the procedure is also susceptible to substitution effects in the presence of correlated features. Given two identical features, MDA always considers one to be redundant to the other. Unfortunately, MDA will make both features appear to be outright irrelevant, even if they are critical.
4. Unlike MDI, it is possible that MDA concludes that all features are unimportant. That is because MDA is based on OOS performance.

4.2.5 Addressing substituions effects using Clustering Methods

- **Substitution effects**

  Substitution effects emerge when two variables share redundant predictive content, potentially distorting feature importance evaluation outcomes. In the context of Mean Decrease Impurity (MDI), the contribution of duplicate variables is effectively halved due to their equiprobable selection. Concurrently, within the Mean Decrease Accuracy (MDA) paradigm, analogous variables may be erroneously deemed inconsequential despite their criticality. This misjudgment arises as the perturbation induced by permuting one variable could be neutralized by the remaining variable, thus obscuring their authentic relevance.

  To counteract substitution effects, a methodological remedy is to aggregate analogous variables into clusters and execute feature importance analysis at the cluster level. Constructed to ensure inter-cluster dissimilarity, this technique attenuates substitution effects. Conducting the analysis on a partitioned feature space, without basis transformation, yields results that are inherently more lucid and intuitive. We use the ONC algorithm [**?**]

- **Clustering the Observation Matrix using the ONC algorithm**

We create a matrix representing our observations in a metric space. For instance, the matrix could be derived from the variation of information between random variables, as discussed in section 3.

We now turn our attention to the clustering algorithm. A viable option is to apply the k-means algorithm to our observation matrix. Although k-means is straightforward and often effective, it has two significant drawbacks:

- Firstly, the algorithm mandates the specification of the number of clusters K, which may not be optimal beforehand;
- Secondly, its random initialization means the algorithm's performance can also be unpredictable.

To overcome these limitations, we propose modifications to the k-means algorithm, introduced in [**?**], called the ONC algorithm.

The core clustering process includes three enhancements, as shown in algorithm 4:

- The first enhancement introduces an objective function to determine the ideal number of clusters, denoted as "K." We adopt the silhouette score for this, as initially proposed by [**?**].

  The silhouette score for an element i in a clustering scenario is defined by the formula:

  $$S_i = \frac{b_i - a_i}{\max\{a_i, b_i\}}; \quad i = 1, \ldots, n$$

  Here, a_i represents the mean distance from the i^{th} element to other elements within the same cluster, while b_i is the mean distance to elements in the nearest other cluster. The silhouette score ranges from 1, indicating perfect clustering, to -1, indicating poor clustering. We define the overall partition quality, q, as:

  $$q = \frac{E[\{S_i\}]}{\sqrt{V[\{S_i\}]}}$$

  where $E[\{S_i\}]$ and $V[\{S_i\}]$ are the mean and variance of the silhouette scores, respectively.
- The second enhancement improves the k-means initialization process. Our modified algorithm begins by evaluating the observation matrix, followed by a two-stage for-loop. The initial stage tests different k values (2 to n) for clustering using k-means with a single initialization, calculating the quality q for each. The subsequent stage repeats this process with multiple initializations, selecting the clustering with the highest q value from both stages.

– The third refinement addresses the detection of clusters of varying distinctiveness within the k-means framework. While effective at identifying prominent clusters, the algorithm might miss more nuanced ones. To address this, we evaluate the quality q_k of each cluster $(k = 1, \ldots, K)$ using scores from the initial clustering. We calculate the average quality $\bar{q}$ and identify clusters below this threshold, marked as $\{q_k \mid q_k < \bar{q}, k = 1, \ldots, K\}$. We denote the count of these lower-quality clusters as K_1, where $K_1 < K$. If K_1 is one or less, we maintain the original clustering. If K_1 is two or more, we reapply the clustering algorithm to just those items in the K_1 clusters, treating the rest as properly clustered.

  This involves constructing a new observation matrix for the K_1 clusters and reapplying the base clustering algorithm. This re-clustering may result in a different configuration for the K_1 items. We then assess the re-clustering's success by comparing the average quality of clusters before and after. Should the re-clustering demonstrate improved quality, we integrate this new clustering with the original. If not, we revert to the clustering determined by the initial algorithm.

4.2.6 Empirical Results

Applying the Optimal Number of Clusters (ONC) algorithm 4 enables the use of Mean Decrease Impurity (MDI) and Mean Decrease Accuracy (MDA) metrics on a cluster level, effectively avoiding substitution effects. This strategic approach ensures a more precise analysis of feature importance by evaluating MDI and MDA within distinct clusters, thereby reducing the risk of not selecting an important feature.

Presented below (figures 10, 11 and 12) are visual representations of clustered correlation, alongside the application of MDI and MDA metrics at the cluster level.

Importantly, the MDA on a cluster level has successfully isolated the cluster of noisy features, demonstrating the method's efficacy in distinguishing between informative and non-informative attributes within the clustered dataset.

Algorithm 4 ONC algorithm

Input: Observation matrix, number of elements n
Output: Optimal clustering configuration
Step 1: Base Clustering
for each k from 2 to n, and for each initialization **do**
    Perform k-means clustering with the current k and initialization.
    For each element i, calculate silhouette score $S_i = \frac{b_i - a_i}{\max\{a_i, b_i\}}$, where:
- a_i represents the mean distance from the i^{th} element to other elements within the same cluster.
- b_i is the mean distance to elements in the nearest other cluster.

    Compute the average silhouette score for this configuration and the overall partition quality q as $q = \frac{E[\{S_i\}]}{\sqrt{V[\{S_i\}]}}$, where $E[\{S_i\}]$ and $V[\{S_i\}]$ are the mean and variance of the silhouette scores, respectively.
end for
Select K and initialization with the highest average silhouette score.
Step 2: Higher-Level Clustering
Evaluate quality q_k of each cluster $k = 1$ to K, and calculate $\bar{q}$, the average quality.
Identify clusters with $q_k < \bar{q}$, denote count as K_1.
if $K_1 > 1$ **then**
    Recluster items in K_1 clusters, focusing on optimizing within this subset.
    Compare quality before and after reclustering.
    if reclustering improves quality **then**
        Merge refined clustering with original configuration.
    else
        Maintain original configuration as determined in Step 1.
    end if
else
    Maintain initial configuration; no reclustering needed.
end if

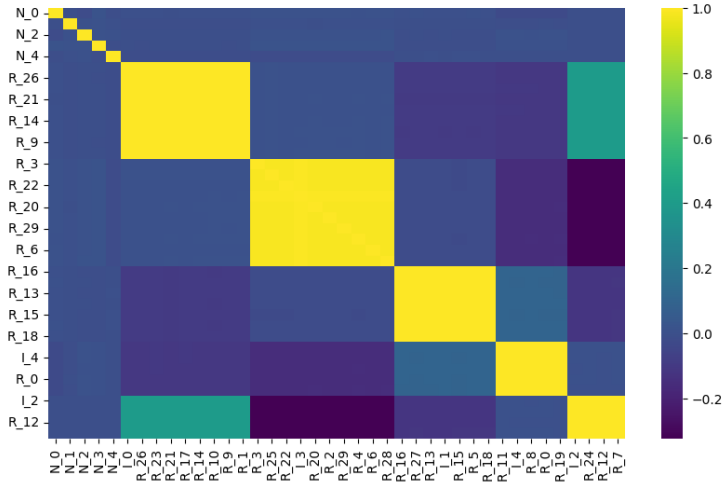


Figure 10: Clustered correlation using the ONC algorithm

- **Clustered MDI**

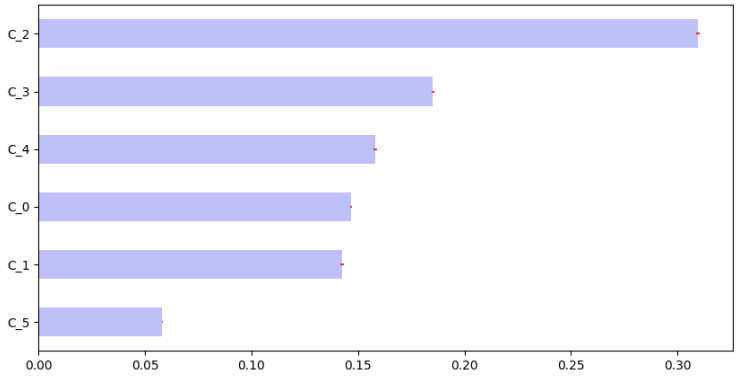


Figure 11: Clustered MDI

- **Clustered MDA**

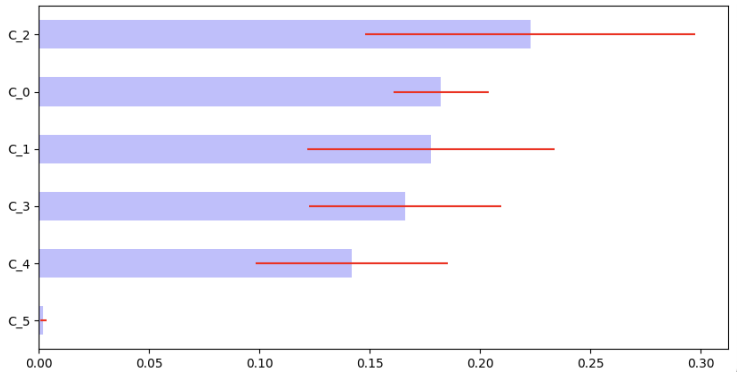


Figure 12: Clustered MDA