ROMA TRE
UNIVERSITÀ DEGLI STUDI

Dipartimento di Matematica e Fisica
Corso di Laurea Magistrale in Scienze Computazionali

Information Theoretic Methods of Inference for Gene Regulatory Networks

Candidato:
Pietro Cestola

Relatore:
Prof. Marco Pedicini

Academic Year: 2022/2023

MSC AMS: 94A15, 92-08, 62P10
KEYWORDS: Information Theory, Inference, Systems Biology.

Contents

List of Algorithms

Introduction

Majority of cells present in our body, with few exceptions, share the same genetic material. Despite this fact, these cells are able to perform completely different tasks, differentiating themselves into many classes. This differentiation has been biologically explained through *gene regulation*: a concept introduced in [20] and then exploited for the first time in a concrete case in [9]. After these first steps, gene regulation was widely recognized as a landmark in understanding the mechanisms of transcriptional control, explaining that even if cells share the same genetic material, in each of them genes are expressed in a different way, this variability in gene expression leads to differentiation at the level of task performed within the organism; in biological terms we would say that they assume different phenotypes. The study of gene regulatory networks is based on mathematical models capable of describing existing relationships between genes involved in the process of regulation. The object of this thesis is to analyze some of the mathematical methods introduced for the inference problem of networks: inference from samples of gene expression data taken from a cell population, paying particular attention to methods based on information theory.

To this end, we provide python implementations of algorithms and we apply them to the analysis of a dataset of interest in the last chapter. In the context of regulatory relationships between genes, it is important to pay particular attention to methods used to di stinguish between direct and indirect relationships between elements of the network, i.e., between those regulation processes that are mediated or not by at least one other element. From a mathematical point of view, it is therefore very important to provide a good characterization of this distinction. In this regard, many approaches using different mathematical perspectives have been presented in literature, among the various approaches capable of providing this classification, we present:

- Information-theoretical-based classification methods [17], [6], [32], [10]
- Regression-based classification methods [18], [11]
- ODE-based classification methods [21].

For a list of existing methods, we refer the reader to [7] or to the more recent reference [5].

1

Mathematical prerequisites

In this first chapter we introduce some notations, furthermore we introduce the mathematical prerequisites needed for a description of inference methods of gene regulatory networks covered in the thesis.

1.1 Notations

For numerical sets, we adopt the canonical notations: $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ and $\mathbb{R}$. In addition we write $\mathbb{R}^+$ for the positive real numbers and $\mathbb{R}_0^+$ for the non-negative real numbers.
When writing analytic expressions we use

Lower case $a, b, c, \dots$ for scalar quantities
Bold lower case $\mathbf{a}, \mathbf{b}, \mathbf{c}, \dots$ for vector quantities
Upper case $A, B, C, \dots$ for matrix quantities
∇ and ∇^2 respectively for Jacobian and Hessian
$\mathcal{M}_{n,m}(\mathbb{R})$ vector space of matrices with real coefficients with n rows and m columns
M^T for the transpose of a matrix M
C^k for the space of functions k times differentiable with continuity

For random variables we use

Uppercase $X, Y, Z, \dots$ for the representation of random variables
Calligraphic $\mathcal{X}, \mathcal{Y}, \mathcal{Z}, \dots$ to denote support of the respective probability distribution functions, for example $\mathcal{X} = \{x : f_X(x) > 0\}$
To omit the subscript reference to the random variable in writing the associated probability distribution function, when this does not cause misunderstandings, we write

$$p(x) := f_X(x) \qquad p(x,y) := f_{X,Y}(x,y) \qquad p(x|y) := f_{X|Y}(x,y)$$

We also use the following standard notations, where S is a set, $\mathbf{v} \in \mathbb{R}^d$ and $f : D \subseteq \mathbb{R}^d \to \mathbb{R}$

$$\mathbb{1}_S(x) = \begin{cases} 1 & x \in S \\ 0 & x \notin S \end{cases} \quad ||\mathbf{v}||_p := \left(\sum_{i=1}^{n} |\mathbf{v}_i|^p \right)^{\frac{1}{p}} \quad ||f||_p := \left(\int_D (f(\mathbf{x}))^p d\mathbf{x} \right)^{\frac{1}{p}} \quad ||f||_\infty := \sup_{\mathbf{x} \in D} f(\mathbf{x})$$

1.2 Statistics prerequisites

The object of this thesis is a problem of statistical inference, i.e. a procedure which aims at estimating certain characteristics of a set of objects of interest (statistical units) being able to observe only a subset of them called *sample*. The object of the inference treated in this thesis are the regulatory relationships between the genes of a cell. In this section we provide the necessary definitions and results, which concern topics related to statistics, for the formulation of the problem of interest.

Definition 1.2.1 (Random sample)**.** A random sample is a set of independent and identically distributed random variables $\{X_1, X_2, \dots, X_n\}$.

In the formulation of a statistical inference problem, the probability distribution of the random variables of the sample is unknown and the interest of the process is focused on the estimation of some parameters or the distribution itself in order to describe some characteristics of the sample and at measuring the confidence of the estimation made. More formally we define

Definition 1.2.2 (Statistical model)**.** Let $\{X_1, X_2, \dots, X_n\}$ be a random sample whose probability distribution is not known. A parametric statistical model is given by

$\varOmega \subseteq \mathbb{R}^m$: for some $m \in \mathbb{N}$, the image of the $\{X_i\}$ called *sample space*
$\varTheta \subseteq \mathbb{R}^d$: for some $d \in \mathbb{N}$ called *parameter support*
$\mathcal{F} = \{f(\cdot, \theta) : \theta \in \varTheta\}$: a family of distributions whose elements are the possible densities describing the random sample

We refer to θ as *model parameter.* A non parametric statistical model is given by

$\varOmega \subseteq \mathbb{R}^m$: for some $m \in \mathbb{N}$, the image of the $\{X_i\}$ called sample space
$\mathcal{F}$: a family of distributions whose elements are the possible densities describing the random sample

In the case of a non parametric model, no specific assumptions are made about the family of distributions describing the sample.

Definition 1.2.3 (Statistic)**.** Given a statistical model, a statistic is a function of the data $T : \varOmega^n \to \mathbb{R}^d$ (for some $d \in \mathbb{N}$) not depending on model parameters. Given a sample $X = \{X_1, X_2, \cdots, X_n\}$ some examples of statistics are the sample mean and the sample variance, respectively:

$$\bar{X}(X) := \frac{1}{n}\sum_{i=1}^{n} X_i \qquad\qquad S^2(X) := \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2.$$

Definition 1.2.4 (Sufficient statistic)**.** Let $X = \{X_1, X_2, \dots, X_n\}$ be a random sample of a parametric statistical model $(\varOmega, \varTheta, \mathcal{F})$ with parameter θ. A statistic $T(X)$ is said to be sufficient for θ if the distribution of the sample does not depend on the parameter θ given $T(X)$, i.e., $\mathbb{P}(X = x|T(X))$ does not depend on θ.

Definition 1.2.5 (Estimator)**.** Let $\{X_1, X_2, \dots, X_n\}$ be a random sample of a statistical model. For a parametric statistical model $(\Omega, \Theta, \mathcal{F})$, an estimator $\hat{\theta}$ of the parameter θ is a statistic such that:

1. $\mathbb{E}(\hat{\theta}) < +\infty$
2. $\hat{\theta}(x) \in \Theta,$ for all $x \in \Omega^n$
3. $\hat{\theta}$ is sufficient for θ

Where in order to make explicit the dependence of the estimator on the cardinality of the random sample we write $\hat{\theta}_n$. For a non parametric statistical model $(\Omega, \mathcal{F})$, an estimator $\hat{f}$ of the probability distribution function f is a statistic such that:

1. $\mathbb{E}(\hat{f}) < +\infty$
2. $\hat{f}(x) \in \mathcal{F},$ for all $x \in \Omega^n$

In general it is not possible to identify a universal criterion to define the goodness of an estimator, in fact, there are multiple approaches that do not always produce comparable results. Despite this, some characteristics have been identified in statistics which form the basis for the formulation of fundamental theorems concerning estimators. The notion of a good estimator extends far beyond the scope of this thesis, therefor the definitions that follow, which constitute only a part of the whole topic, have been chosen with the specific intention of facilitating the reading of the results reported in the thesis. For a more in-depth reading we refer the reader to James et al. [12].

Definition 1.2.6 (Bias)**.** Let $\hat{\theta}$ be an estimator of a parameter θ of a statistical model. The bias of the estimator $\hat{\theta}$ is the expected value of the difference between the value of the estimator and the real value of the parameter

$$\text{bias}(\hat{\theta}) := \mathbb{E}(\hat{\theta} - \theta)$$

Definition 1.2.7 (Unbiased and asymptotically unbiased estimator)**.** Given an estimator $\hat{\theta}_n$ of a parameter θ of a statistical model we say that the estimator is unbiased (vice versa biased) if

$$\text{bias}(\hat{\theta}_n) = 0, \text{ for all } n \in \mathbb{N}$$

on the other hand we say that the estimator is asymptotically unbiased if

$$\lim_{n \to +\infty} (\text{bias}(\hat{\theta}_n)) = 0.$$

A recurring idea in statistics consists in defining the concept of "optimum value" by means of a process of minimization of a non-negative quantity, called *cost function*, which must be proportional to the error made by the estimator. This type of approach is used for the definition of a measure of goodness of an estimator through the following definition:

Definition 1.2.8 (Mean squared error)**.** Given an estimator $\hat{\theta}_n$ of a parameter θ of a statistical model, the mean squared error of $\hat{\theta}_n$ is defined as:

$$\text{MSE}(\hat{\theta}_n) := \mathbb{E}((\hat{\theta}_n - \theta)^2).$$

This quantity possesses all the features mentioned above: it is a positive quantity proportional to the error committed by the estimator. this definition leads us to the following criterion: among alternative estimators, we choose the one with the smallest mean squared error.

Remark 1.2.1. For a generic estimator $\hat{\theta}_n$ holds that:

$$\mathrm{MSE}(\hat{\theta}_n) = (\mathrm{bias}(\hat{\theta}_n))^2 + Var(\hat{\theta}_n)$$

In fact from the definition of mean squared error we get:

$$\begin{aligned}
\mathrm{MSE}(\hat{\theta}_n) &= \mathbb{E}((\hat{\theta}_n - \theta)^2) \\
&= \mathbb{E}((\hat{\theta}_n - \mathbb{E}(\hat{\theta}_n) + \mathbb{E}(\hat{\theta}_n) - \theta)^2) \\
&= \mathbb{E}((\hat{\theta}_n - \mathbb{E}(\hat{\theta}_n))^2) + 2(\mathbb{E}(\hat{\theta}_n) - \theta)\mathbb{E}(\hat{\theta}_n - \mathbb{E}(\hat{\theta}_n)) + \mathbb{E}((\mathbb{E}(\hat{\theta}_n) - \theta)^2) \\
&= \mathbb{E}((\hat{\theta}_n - \mathbb{E}(\hat{\theta}_n))^2) + 2(\mathbb{E}(\hat{\theta}_n) - \theta)(\mathbb{E}(\hat{\theta}_n) - \mathbb{E}(\hat{\theta}_n)) + (\mathbb{E}(\hat{\theta}_n) - \theta)^2 \\
&= \mathrm{Var}(\hat{\theta}_n) + (\mathrm{bias}(\hat{\theta}_n))^2.
\end{aligned}$$

Trying to minimise the mean square error, bias and variance often turn out to be inversely proportional as the number of elements n in the sample (or any additional parameter) changes, which is why we refer to this problem as the bias-variance tradeoff.

Definition 1.2.9 (MLE estimator)**.** Given a random sample $\{X_1, X_2, \dots, X_n\}$ of a parametric statistical model $(\Omega, \Theta, \mathcal{F})$ the maximum likelihood function is defined as

$$\mathcal{L}(\theta|X) := \prod_{i=1}^{n} f(X_i|\theta)$$

The maximum likelihood estimator for the parameter θ is then defined as

$$\hat{\theta}_{\mathrm{MLE}}(X) := \arg\max_{\theta \in \Theta} \mathcal{L}(\theta|X)$$

Informally, we are looking for the value of the parameter θ that maximises the probability of observing a given random sample.

Proposition 1.2.1 (Gaussian MLE estimator)**.** The MLE estimators for a random sample $X = \{X_1, X_2, \dots, X_n\}$ of Gaussian random variables of mean $\mu \in \mathbb{R}$ and variance $\sigma \in \mathbb{R}^+$ are

$$\hat{\mu}_{\mathrm{MLE}}(X) = \frac{1}{n}\sum_{i=1}^{n} X_i \qquad \hat{\sigma}_{\mathrm{MLE}}(X) = \frac{1}{n}\sum_{i=1}^{n} (X_i - \hat{\mu}_{\mathrm{MLE}}(X))^2$$

Proof. Let us first write the maximum likelihood function for a sample of Gaussian random variables of mean μ and variance σ

$$\mathcal{L}(\mu, \sigma|X) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma}(x_i - \mu)^2\right) = (2\pi\sigma^2)^{-n/2} \exp\left(-\frac{1}{2\sigma}\sum_{i=1}^{n}(x_i - \mu)^2\right)$$

We note that since the natural logarithm is a strictly increasing function, it holds that

$$\arg\max_{(\mu,\sigma)\in\Theta} \mathcal{L}(\mu, \sigma|X) = \arg\max_{(\mu,\sigma)\in\Theta} \ln(\mathcal{L}(\mu, \sigma|X))$$

We then develop the expression on the right

$$\ln(\mathcal{L}(\mu, \sigma|X)) = \ln\left((2\pi\sigma^2)^{-n/2} \exp\left(-\frac{1}{2\sigma}\sum_{i=1}^{n}(x_i - \mu)^2\right)\right) = -\frac{n}{2}\ln(2\pi\sigma^2) - \frac{1}{2\sigma}\sum_{i=1}^{n}(x_i - \mu)^2$$

Whose critical point is given by

$$0 = \frac{\partial}{\partial \mu}(\ln(\mathcal{L}(\mu, \sigma|X))) = \sum_{i=1}^{n}(x_i - \mu) \Rightarrow \mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$0 = \frac{\partial}{\partial \sigma}(\ln(\mathcal{L}(\mu, \sigma|X))) = -\frac{n}{\sigma} + \frac{1}{\sigma^2}\sum_{i=1}^{n}(x_i - \mu)^2 \Rightarrow \sigma = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$$

Such critical point is the absolute maximum since

$$\lim_{\mu \to \pm\infty} \ln(\mathcal{L}(\mu, \sigma|X)) = \lim_{\sigma \to +\infty} \ln(\mathcal{L}(\mu, \sigma|X)) = \lim_{\sigma \to 0^+} \ln(\mathcal{L}(\mu, \sigma|X)) = -\infty.$$

□

Definition 1.2.10 (Empirical distribution function). Let $\{X_1, X_2, \ldots, X_n\}$ be a random sample of a non parametric statistical model with sample space $\Omega \subseteq \mathbb{R}$. The Empirical distribution function is an estimator of the cumulative distribution F of the variables of the sample defined as

$$\hat{F}_n(x) := \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{(-\infty,x]}(X_i)$$

We observe that it is an asymptotically unbiased estimator, in fact

$$\mathbb{E}(\hat{F}_n(x)) = \mathbb{E}\left(\frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{(-\infty,x]}(X_i)\right) = \mathbb{E}(\mathbb{1}_{(-\infty,x]}(X_1)) = \int_{\mathcal{X}} f(y)\mathbb{1}_{(-\infty,x]}(y)dy = F(x).$$

Definition 1.2.11 (Kolmogorov-Smirnov distance). Let $\{X_1, X_2, \ldots, X_n\}$ be a random sample of a non parametric statistical model and let $\bar{F}$ be a cumulative distribution function over $\mathcal{X}$. The Kolmogorov-Smirnov distance is a statistic defined as

$$D_n(x) := \sup_{x \in \mathcal{X}} |\hat{F}_n(x) - \bar{F}(x)|.$$

Definition 1.2.12 (Rank of a sample). The rank $R(X)$ of a sample $X = \{X_1, X_2, \ldots, X_n\}$ is the vector of random variables $R_1, R_2, \ldots, R_n$ obtained by the following procedure

1. Let $\sigma : \{1, \ldots, n\} \to \{1, \ldots, n\}$ a bijective map such that $i \leq j \Rightarrow X_{\sigma(i)} \leq X_{\sigma(j)}$
2. For every $i \in \{1, 2, \ldots, n\}$ let $r_i := \{j : X_j = X_i\}$
3. For every $i \in \{1, 2, \ldots, n\}$ let $R_i := \frac{1}{|r_i|}\sum_{j \in r_i} \sigma(j)$

Finally

$$R(X) := \{R_1, R_2, \ldots, R_n\}$$

Definition 1.2.13 (Spearmann's correlation coefficient). Let $\{X_1, X_2, \ldots, X_n\}$ and $\{Y_1, Y_2, \ldots, Y_n\}$ be two random samples of a statistical model, the Spearmann correlation coefficient ρ is defined as

$$\rho := \frac{\operatorname{cov}(R(X), R(Y))}{\sigma_{R(X)}\sigma_{R(Y)}} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}(R(X)_i - \bar{R}(X))(R(Y)_j - \bar{R}(Y))}{\sqrt{\sum_{i=1}^{n}(R(Y)_i - \bar{R}(X))}\sqrt{\sum_{i=1}^{n}(R(Y)_i - \bar{R}(Y))}}$$

Where $\bar{R}(X), \bar{R}(Y)$ are the sample means of the rank variables.

1.2.1 Probability density estimation

Some formulations of the problem of inference of gene regulatory networks need to provide an estimate of the probability distribution function of one or more random variables. The problem arises from the need to perform calculations involving probabilistic quantities and measurements which require at least partial knowledge of the distribution of a given set of variables. Consider a random sample $\{X_1, X_2, \ldots, X_n\}$ with density f and sample space Ω of a non parametric statistical model, the goal of a density estimation method is to define an estimator $\hat{f}$ of the probability density function. We note that for the purpose of this thesis it is sufficient to restrict ourselves to the case of random samples having as sample space a subset of $\mathbb{R}^d$, for a given $d \in \mathbb{N}$, obtained as the product of unit intervals $[0, 1]$ of $\mathbb{R}$, the extension to products of arbitrary intervals is straightforward. For reasons strictly related to the formulation of the algorithms in the third chapter we pay particular attention to the case $d \leq 2$.

Lemma 1.2.1. Let $d \in \mathbb{N}$ and $U \subseteq \mathbb{R}^d$ compact. Let moreover $f : U \to \mathbb{R}$ be continuous over U, then there exists $x^* \in U$ such that

$$\frac{1}{\mathrm{Vol}(U)} \int_U f(x)dx = f(x^*) \qquad\qquad \mathrm{Vol}(U) := \int_U dx$$

Proof. We first note that

$$\begin{aligned}
&\inf_U f \leq f(x) \leq \sup_U f \\
&\Rightarrow \mathrm{Vol}(U) \inf_U f \leq \int_U f(x)dx \leq \mathrm{Vol}(U) \sup_U f \\
&\Rightarrow \inf_U f \leq \frac{1}{\mathrm{Vol}(U)} \int_U f(x)dx \leq \sup_U f
\end{aligned}$$

By hypothesis f is continuous and U compact, therefore $f(U) \subset \mathbb{R}$ is compact, in particular[1] $f(U) = [\inf_U f, \sup_U f]$, so there exists $x^* \in U$ such that

$$f(x^*) = \frac{1}{\mathrm{Vol}(U)} \int_U f(x)dx$$

□

Histogram density estimation

The first step to define this estimator is to discretize the sample space by providing a partition whose elements are called bins. The basic idea is to estimate the probability at a point in the sample space belonging to a given bin as the ratio between the fraction of samples falling inside the bin and the volume of the bin itself. In this section we are only interested in the case of partitions formed by bins obtained as product of intervals of equal size.

[1] Since it is a compact of $\mathbb{R}$ with respect to the Euclidean topology it is a closed interval, moreover it must necessarily contain *inf* and *sup*

Definition 1.2.14 (Histogram estimator)**.** Let $\{X_1, X_2, \dots, X_n\}$ be a random sample having probability distribution f and sample space $[0,1]^d$ with $d \in \mathbb{N}$. Let also be $m \in \mathbb{N}$, we define:

$$B_{i_1,i_2,\dots,i_d} := \left[\frac{i_1}{m}, \frac{i_1+1}{m}\right) \times \left[\frac{i_2}{m}, \frac{i_2+1}{m}\right) \times \dots \times \left[\frac{i_d}{m}, \frac{i_d+1}{m}\right], i_1, i_2, \dots, i_d \in \{0, 1, \dots, m-1\} \quad (1.1)$$

Which are the subsets forming the sample space partition:

$$\bigsqcup_{i_1,i_2,\dots,i_n} B_{i_1,i_2,\dots,i_d} = [0,1]^d$$

For a generic $\mathbf{x} \in B_{i_1,i_2,\dots,i_d}$, the histogram density estimator is defined as

$$\hat{f}_n(\mathbf{x}) := \frac{m^d}{n} \sum_{j=1}^{n} \mathbb{1}_{B_{i_1,i_2,\dots,i_d}}(X_j).$$

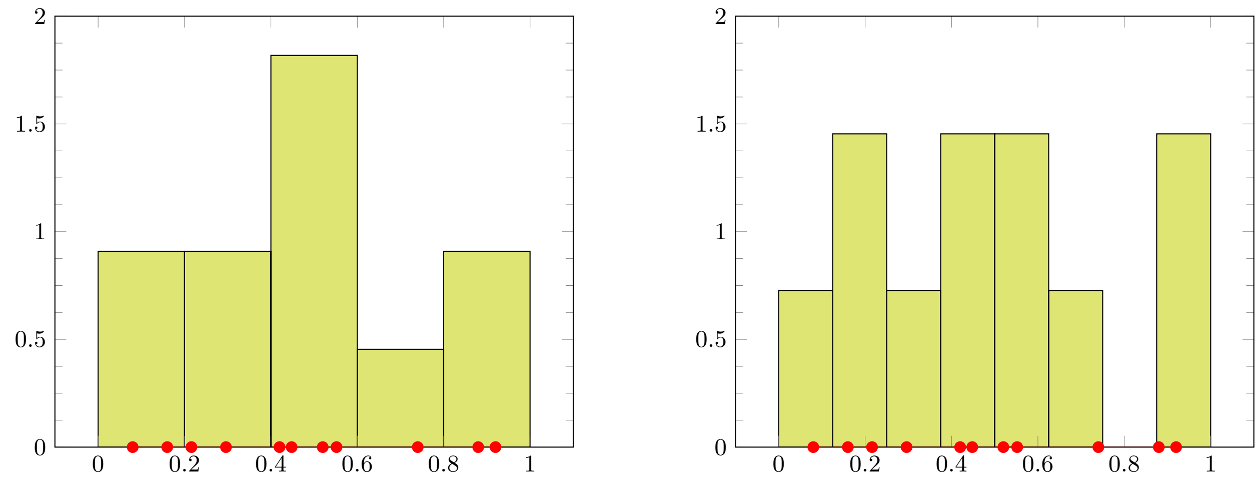


Fig. 1.1: Example of histogram density estimation with $d = 1$. The same sample is represented in the two figures, the method is therefore calculated first with $m = 5$ bins then with $m = 8$

Proposition 1.2.2. Let $\{X_1, X_2, \dots, X_n\}$ be a random sample having sample space $[0,1]^d$ and continuous and Lipschitz of constant L probability density function f. Then the bias of the histogram density estimator decreases as the number of bins m increases.

Proof. Let $\mathbf{x} \in B_{i_1,i_2,\dots,i_d}$ for some $i_1, i_2, \dots, i_n \in \{0, 1, \dots, m-1\}$. First of all we note that from the linearity of the expected value, the continuity of f and by Lemma 1.2.1 $(*)$ follows that

$$\begin{aligned}\mathbb{E}(\hat{f}_n(\mathbf{x})) &= \frac{m^d}{n} \sum_{i=1}^{n} \mathbb{E}(\mathbb{1}_{B_{i_1,i_2,\dots,i_d}}(X_i)) = \frac{m^d}{n} \sum_{i=1}^{n} \int_{\mathbb{R}} \mathbb{1}_{B_{i_1,i_2,\dots,i_d}}(\mathbf{x}) f(\mathbf{x}) d\mathbf{x} \\ &= m^d \int_{B_{i_1,i_2,\dots,i_d}} f(\mathbf{x}) d\mathbf{x} = m^d \int_{\bar{B}_{i_1,i_2,\dots,i_d}} f(\mathbf{x}) d\mathbf{x} \overset{(*)}{=} f(\mathbf{x}^*)\end{aligned}$$

Where $\bar{B}$ denotes the closure of the set. Due to the definition of the partition 1.1 it follows that for two generic $x, y \in B_{i_1,i_2,\dots,i_d}$ holds:

$$\|\mathbf{x}-\mathbf{y}\|_2 = \sqrt{\sum_{i=1}^{d}(\mathbf{x}_i-\mathbf{y}_i)^2} \leq \sqrt{\sum_{i=1}^{d}\frac{1}{m^2}} = \frac{\sqrt{d}}{m}$$

Moreover, by hypothesis, the function has Lipschitz constant L, so

$$|\text{bias}(\hat{f}_n)| = |\mathbb{E}(\hat{f}_n - f(\mathbf{x}))| = |f(\mathbf{x}^*) - f(\mathbf{x})| \leq L\|\mathbf{x}^*-\mathbf{x}\|_2 \leq \frac{L\sqrt{d}}{m}$$

□

Proposition 1.2.3. Let $\{X_1, X_2, \dots, X_n\}$ be a random sample having sample space $[0,1]^d$ and continuous and Lipschitz of constant L probability density function f. Then the variance of the histogram density estimator increases as the number of bins m increases and decreases as the number of samplings n increases.

Proof. Let $\mathbf{x} \in B_{i_1,i_2,\dots,i_d}$ for some $i_1, i_2, \dots, i_d \in \{0,1,\dots,m-1\}$, from the previous proposition we know that $\mathbb{E}(\hat{f}_n(\mathbf{x})) = f(\mathbf{x}^*)$ for some $\mathbf{x}^* \in B_{i_1,i_2,\dots,i_d}$, therefore

$$Var(\hat{f}_n(\mathbf{x})) = \mathbb{E}((\hat{f}_n(\mathbf{x}))^2) - \mathbb{E}(\hat{f}_n(\mathbf{x}))^2 = \mathbb{E}((\hat{f}_n(\mathbf{x}))^2) - (f(\mathbf{x}^*))^2$$

It remains to calculate the first addendum. Recalling that X_i and X_j are independent if $i \neq j$ we obtain that

$$\begin{aligned}
\mathbb{E}\big((\hat{f}_n(x))^2\big) &= \mathbb{E}\left((\frac{m^d}{n}\sum_{i=1}^{n} \mathbb{1}_{B_{i_1,i_2,\dots,i_d}}(X_i))^2 \right) \\
&= \frac{m^{2d}}{n^2}\left(\sum_{i\neq j}^{n} \mathbb{E}(\mathbb{1}_{B_{i_1,i_2,\dots,i_d}}(X_i)\mathbb{1}_{B_{i_1,i_2,\dots,i_d}}(X_j) + \sum_{i=1}^{n}\mathbb{E}((\mathbb{1}_{B_{i_1,i_2,\dots,i_d}}(X_i))^2) \right) \\
&= \frac{m^{2d}}{n^2}\left(\sum_{i\neq j}^{n} \mathbb{E}(\mathbb{1}_{B_{i_1,i_2,\dots,i_d}}(X_i))\mathbb{E}(\mathbb{1}_{B_{i_1,i_2,\dots,i_d}}(X_j)) + \sum_{i=1}^{n}\mathbb{E}(\mathbb{1}_{B_{i_1,i_2,\dots,i_d}}(X_i)) \right) \\
&= \frac{m^{2d}}{n^2}\left(\sum_{i\neq j}^{n} \frac{f(\mathbf{x}^*)}{m^d}\frac{f(\mathbf{x}^*)}{m^d} + \sum_{i=1}^{n}\frac{f(\mathbf{x}^*)}{m^d} \right) \\
&= \frac{n-1}{n}(f(\mathbf{x}^*))^2 + \frac{m^d}{n}f(\mathbf{x}^*)
\end{aligned}$$

So

$$Var(\hat{f}_n(x)) = \frac{n-1}{n}(f(\mathbf{x}^*))^2 + \frac{m^d}{n}f(\mathbf{x}^*) - f(\mathbf{x}^*)^2 = \frac{f(\mathbf{x}^*)}{n}(m^d - f(\mathbf{x}^*))$$

□

Remark 1.2.2. Informally, assuming the required hypotheses, from Propositions 1.2.2 and 1.2.3 we have that

$$\text{MSE}(\hat{f}_n(\mathbf{x})) = (\text{bias}(\hat{f}_n(\mathbf{x})))^2 + \text{Var}(\hat{f}_n(\mathbf{x})) \approx \frac{L^2 d}{m^2} + m^d\frac{f(\mathbf{x}^*)}{n}$$

It can be shown that the optimal choice of m is

$$m_{\text{opt}} = \left(\frac{nL^2}{f(\mathbf{x}^*)}\right)^{\frac{1}{2+d}} \approx n^{\frac{1}{2+d}}$$

We can therefore see that the mean square error increases as the dimension of the sample space increases

$$\text{MSE}(\hat{f}_n(\mathbf{x})) \approx 2n^{-\frac{2}{2+d}}.$$

Kernel density estimation

Definition 1.2.15 (Kernel function)**.** A kernel function $K : \mathbb{R}^n \to \mathbb{R}$ is a function that respects the following properties

$$K(\mathbf{x}) \geq 0 \text{ for all } \mathbf{x} \in \mathbb{R}^n \qquad \int_{\mathbb{R}^n} K(\mathbf{x})d\mathbf{x} = 1 \qquad \int_{\mathbb{R}^n} \mathbf{x}_i K(\mathbf{x})d\mathbf{x} = 0.$$

Definition 1.2.16 (Gaussian kernel)**.** The n-dimensional Gaussian kernel is defined as the following kernel function $K_n : \mathbb{R}^n \to \mathbb{R}$

$$K_n(\mathbf{x}) := (2\pi)^{-\frac{n}{2}} \exp\left(-\frac{\|\mathbf{x}\|_2^2}{2}\right).$$

Definition 1.2.17 (Epanechnikov kernel)**.** The Epanechnikov kernel function $K_n : \mathbb{R}^n \to \mathbb{R}$ is defined as

$$K_n(\mathbf{x}) = \frac{\Gamma\left(2 + \frac{n}{2}\right)}{\pi^{\frac{n}{2}}} \max\left(0, 1 - \|\mathbf{x}\|_2^2\right).$$

Definition 1.2.18 (Kernel density estimator)**.** Let $\{X_1, X_2, \ldots, X_n\}$ be a random sample with probability distribution f and sample space $\mathbb{R}^d$ for some $d \in \mathbb{N}$. Let $K : \mathbb{R}^d \to \mathbb{R}$ be a kernel function and $h \in \mathbb{R}^+$ a non-negative real value called bandwidth. The KDE estimator of kernel K and bandwidth h is defined as:

$$\hat{f}_n(\mathbf{x}) := \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{1}{h}(\mathbf{x} - X_i)\right)$$

We note that the estimator has unitary integral over $\mathbb{R}^d$, in fact

$$\int_{\mathbb{R}^n} \hat{f}_n(\mathbf{x})d\mathbf{x} = \int_{\mathbb{R}^n} \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{1}{h}(\mathbf{x} - X_i)\right)d\mathbf{x} = \frac{1}{nh^d} \sum_{i=1}^{n} \int_{\mathbb{R}^n} K\left(\frac{1}{h}(\mathbf{x} - X_i)\right)d\mathbf{x}$$

Finally, by apply the variable change $\mathbf{x} = X_i + h\mathbf{y}$.

$$= \frac{1}{nh^d} \sum_{i=1}^{n} h^d \int_{\mathbb{R}^n} K(\mathbf{y})d\mathbf{y} = \frac{1}{n} \sum_{i=1}^{n} 1 = 1.$$

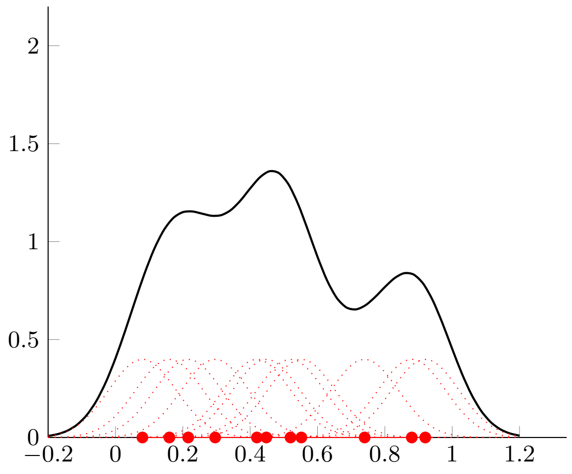


Fig. 1.2: Example of density estimation with KDE with 1-dimensional Gaussian kernel (red dotted lines).

Proposition 1.2.4. Given a random sample $\{X_1, X_2, \dots, X_n\}$ having probability density function $f \in C^2(\mathbb{R}^d, \mathbb{R})$ with bounded second derivative, given a kernel function $K : \mathbb{R}^d \to \mathbb{R}$ such that

$$\|K\|_\infty < +\infty \qquad \int_{\mathbb{R}^d} \mathbf{z}_i \mathbf{z}_j K(\mathbf{z}) d\mathbf{z} < +\infty \text{ for all } i, j \in \{1, 2, \dots, n\}$$

The bias of the KDE estimator having bandwidth $h \in \mathbb{R}^+$ is in $O(h^2)$ as $h \to 0$.

Proof.

$$\mathbb{E}(\hat{f}_n(\mathbf{x})) = \mathbb{E}\left(\frac{1}{nh^d}\sum_{i=1}^n K\left(\frac{X_i - \mathbf{x}}{h}\right)\right) = \frac{1}{h^d}\mathbb{E}\left(K\left(\frac{X_1 - \mathbf{x}}{h}\right)\right) = \frac{1}{h^d}\int_{\mathbb{R}^d} K\left(\frac{\mathbf{y} - \mathbf{x}}{h}\right) f(\mathbf{y}) d\mathbf{y}$$

We note that since the kernel function is bounded by hypothesis and $\|f\|_1 = 1$ by definition, from Holder's inequality follows that

$$\int_{\mathbb{R}^d} K\left(\frac{\mathbf{y} - \mathbf{x}}{h}\right) f(\mathbf{y}) d\mathbf{y} \leq \|K\|_\infty \|f\|_1 = \|K\|_\infty < +\infty.$$

Now applying the following variable change $\mathbf{y} = \mathbf{x} + h\mathbf{z}$ and developing in Taylor series with the remainder in Lagrange form, the expected value of the estimator becomes

$$\begin{aligned}
&= \int_{\mathbb{R}^d} K(\mathbf{z}) f(\mathbf{x} + h\mathbf{z}) d\mathbf{z} \\
&= \int_{\mathbb{R}^d} K(\mathbf{z})(f(\mathbf{x}) + \nabla f(\mathbf{x})(h\mathbf{z}) + \frac{1}{2}\nabla^2 f(\xi)(h\mathbf{z}, h\mathbf{z})) d\mathbf{z} \\
&= f(\mathbf{x}) \underbrace{\int_{\mathbb{R}^d} K(\mathbf{z}) d\mathbf{z}}_{=1 \text{ (By definition of } K)} + h \sum_{i=1}^n \frac{\partial f(\mathbf{x})}{\partial x_i} \underbrace{\int_{\mathbb{R}^d} \mathbf{z}_i K(\mathbf{z}) d\mathbf{z}}_{=0 \text{ (By definition of } K)} + \frac{h^2}{2}\int_{\mathbb{R}^d} K(\mathbf{z}) \nabla^2 f(\xi)(\mathbf{z}, \mathbf{z}) d\mathbf{z} \\
&= f(\mathbf{x}) + \frac{h^2}{2}\int_{\mathbb{R}^d} K(\mathbf{z}) \nabla^2 f(\xi)(\mathbf{z}, \mathbf{z}) d\mathbf{z} \\
&= f(\mathbf{x}) + O(h^2)
\end{aligned}$$

where by Holder's inequality and the fact that f has bounded second derivative follows that

$$\int_{\mathbb{R}^d} K(\mathbf{z}) \nabla^2 f(\xi)(\mathbf{z}, \mathbf{z}) d\mathbf{z} = \int_{\mathbb{R}^d} \sum_{i,j=1}^n \frac{\partial^2 f(\xi)}{\partial \mathbf{x}_i \partial \mathbf{x}_j} \mathbf{z}_i \mathbf{z}_j K(\mathbf{z}) d\mathbf{z} \leq \sum_{i,j=1}^n \sup_{\eta \in \mathbb{R}^d} \left|\frac{\partial^2 f(\eta)}{\partial \mathbf{x}_i \partial \mathbf{x}_j}\right| \int_{\mathbb{R}^d} \mathbf{z}_i \mathbf{z}_j K(\mathbf{z}) d\mathbf{z} < +\infty$$

so the bias of the estimator is

$$\text{bias}(\hat{f}_n(\mathbf{x})) = \mathbb{E}(\hat{f}_n(\mathbf{x})) - f(\mathbf{x}) = O(h^2).$$

□

Proposition 1.2.5. Given a random sample $\{X_1, X_2, \dots, X_n\}$ having probability density function $f \in C^2(\mathbb{R}^d, \mathbb{R})$ with bounded derivatives, given a kernel function $K : \mathbb{R}^d \to \mathbb{R}$ such that for all $i, j \in \{1, 2, \dots, n\}$

$$\|K\|_\infty < +\infty \qquad \int_{\mathbb{R}^d} \mathbf{z}_i \mathbf{z}_j K(\mathbf{z}) d\mathbf{z} < +\infty \qquad \int_{\mathbb{R}^d} \mathbf{z}_i K^2(\mathbf{z}) d\mathbf{z} < +\infty$$

The variance of the KDE estimator is an $O(\frac{1}{nh^d})$ as $h \to 0$.

Proof.

$$\mathrm{Var}(\hat{f}_n(\mathbf{x})) = \mathrm{Var}\left(\frac{1}{nh^d}\sum_{i=1}^n K\big(\frac{X_i-\mathbf{x}}{h}\big)\right) = \frac{1}{nh^{2d}}\mathrm{Var}\left(K\big(\frac{X_1-\mathbf{x}}{h}\big)\right).$$

We first note that since the kernel function is bounded by hypothesis and $\|f\|_1 = 1$ by definition, then from Holder's inequality follows that

$$\int_{\mathbb{R}^d} K^2\big(\frac{\mathbf{y}-\mathbf{x}}{h}\big)f(\mathbf{y})d\mathbf{y} \leq \|K^2\|_\infty\|f\|_1 = \|K^2\|_\infty < +\infty \tag{1.2}$$

we then calculate the following by applying the variable change $\mathbf{y} = \mathbf{x} + h\mathbf{z}$ and by developing in Taylor series with the remainder in Lagrange form

$$\begin{aligned}
\mathbb{E}\left(K^2\big(\frac{X_i-\mathbf{x}}{h}\big)\right) &= \int_{\mathbb{R}^d} K^2\big(\frac{\mathbf{y}-\mathbf{x}}{h}\big)f(\mathbf{y})d\mathbf{y} = h^d\int_{\mathbb{R}^d} K^2(\mathbf{z})f(\mathbf{x}+h\mathbf{z})d\mathbf{z}\\
&= h^d\int_{\mathbb{R}^d} K^2(\mathbf{z})(f(\mathbf{x}) + \nabla f(\xi)(h\mathbf{z}))d\mathbf{z}\\
&= h^d f(\mathbf{x})\int_{\mathbb{R}^d} K^2(\mathbf{z})d\mathbf{z} + h^{d+1}\int_{\mathbb{R}^d} K^2(\mathbf{z})\nabla f(\xi)(\mathbf{z})d\mathbf{z}\\
&= O(h^d)
\end{aligned}$$

where we used the fact that from Holder's inequality

$$\int_{\mathbb{R}^d} K^2(\mathbf{z})d\mathbf{z} \leq \|K\|_\infty\|K\|_1 = \|K\|_\infty < +\infty$$

and the fact that by hypothesis

$$\int_{\mathbb{R}^d} K^2(\mathbf{z})\nabla f(\xi)(\mathbf{z})d\mathbf{z} = \sum_{i=1}^n\int_{\mathbb{R}^d}\frac{\partial f(\xi)}{\partial x_i}\mathbf{z}_iK^2(\mathbf{z})d\mathbf{z} \leq \sum_{i=1}^n\sup_{\eta\in\mathbb{R}^d}\left|\frac{\partial f(\eta)}{\partial x_i}\right|\int_{\mathbb{R}^d}\mathbf{z}_iK^2(\mathbf{z})d\mathbf{z} < +\infty.$$

By Proposition 1.2.4

$$\mathbb{E}\left(K\big(\frac{X_i-\mathbf{x}}{h}\big)\right) = h^d(f(\mathbf{x}) + O(h^2)). \tag{1.3}$$

Finally, substituting what we have seen, as $h \to 0$ we obtain

$$\begin{aligned}
\mathrm{Var}(\hat{f}_n(\mathbf{x})) &= \frac{1}{nh^{2d}}\left(\mathbb{E}\left(K^2\big(\frac{X_i-\mathbf{x}}{h}\big)\right) - \left(\mathbb{E}\left(K\big(\frac{X_i-\mathbf{x}}{h}\big)\right)\right)^2\right)\\
&= \frac{1}{nh^{2d}}\left(O(h^d) - \left(h^d(f(\mathbf{x}) + O(h^2))\right)^2\right)\\
&= \frac{1}{nh^{2d}}\left(O(h^d) + h^{2d}(f(\mathbf{x}))^2 + O(h^{2d+2}) + O(h^{2d+4})\right)\\
&= O\big(\frac{1}{nh^d}\big)
\end{aligned}$$

□

Remark 1.2.3. Using a kernel function with compact support such as Definition 1.2.17, Propositions 1.2.4 and 1.2.5 hold requiring only that $f \in C^2(\mathbb{R}^d, \mathbb{R})$ and $\|K\|_\infty < +\infty$ since it is always possible to bound f, it's first and second derivative on a compact, since they are all continuous functions.

Remark 1.2.4. Informally, assuming the required hypotheses, from Propositions 1.2.4 and 1.2.5 we have that

$$\mathrm{MSE}(\hat{f}_n(\mathbf{x})) = (\mathrm{bias}(\hat{f}_n(\mathbf{x})))^2 + \mathrm{Var}(\hat{f}_n(\mathbf{x})) \approx h^4 + \frac{1}{nh^d}$$

The optimal choice of h is therefore

$$0 = \frac{\partial}{\partial h}(h^4 + \frac{1}{nh^d}h^4 + \frac{1}{nh^d}) = 4h^3 - \frac{d}{nh^{d+1}} \Rightarrow h_{\mathrm{opt}} = \left(\frac{d}{4n}\right)^{\frac{1}{4+d}} \approx n^{-\frac{1}{4+d}}$$

We can therefore see that the mean square error increases as the dimension of the sample space increases

$$\mathrm{MSE}(\hat{f}_n(\mathbf{x})) \approx 2n^{-\frac{4}{4+d}}.$$

1.2.2 Regression

Regression is a statistical model designed to estimate the relationship between a *dependent variable* Y and a set of *independent variables* $X_1, X_2, \ldots, X_n$ also called features. We observe that the term variable in this context does not refer to its probabilistic meaning, in fact, while Y is a random variable X_i are not. This model provides for the assumption of the following probabilistic relationship between the dependent variable and the features by means of the parameters $\theta_0, \theta_1, \theta_2, \ldots, \theta_m$

$$Y \sim f(X_1, \ldots, X_n, \theta_0, \ldots, \theta_m)$$

Given a set of features and a random sample, the optimal estimators $\hat{\theta}_0, \ldots, \hat{\theta}_n$ of the parameters are therefore inferred in such a way that the relationship best fits, with respect to a given criterion, the data provided. For the formulation of inference algorithms of gene regulatory networks reported later it is sufficient to restrict ourselves to the case of linear regression that we introduce below.

Definition 1.2.19 (Simple linear regression)**.** With the term simple linear regression we refer to the statistical model whose sample space and whose support of the parameters are respectively $\Omega = \mathbb{R}$ and $\Theta = \mathbb{R}^{m+1}$, with $m \in \mathbb{N}$ and such that a generic random sample $\{Y_1, Y_2, \ldots, Y_n\}$ of the model has the following normal distribution.

$$Y_i \sim N(\theta_0 + \theta_1 X_{i,1} + \cdots + \theta_m X_{i,m}, \sigma^2)$$

Where the features $X_{i,j} \in \mathbb{R}$ for each i, j and $\sigma \in \mathbb{R}^+$. We note that in defining regression as a statistical model the requirement for identical distribution among variables of the random sample is dropped.

Remark 1.2.5. In order to make it easier to formulate the results of this section we introduce the following matrix notation for the elements of the regression model. The features matrix

$$X = \begin{pmatrix} 1 & X_{1,1} & \ldots & X_{1,m} \\ 1 & \vdots & & \vdots \\ 1 & X_{n,1} & \ldots & X_{n,m} \end{pmatrix} \in \mathcal{M}_{n,m+1}(\mathbb{R})$$

Where the column of 1's is needed in order to include the θ_0 parameter. The vector representing a generic sample of the model

$$Y = (Y_1, \ldots, Y_n)^T \in \mathbb{R}^n$$

The vector of model parameters

$$\theta = (\theta_0, \theta_1, \ldots, \theta_m)^T \in \mathbb{R}^{m+1}.$$

The idea behind this model for calculating parameter estimators is to associate the data and every choice of parameters with a value in $\mathbb{R}^+$, that is directly proportional to the error committed in predicting the data, so that the minimization of this value leads to the optimal choice of the parameter estimators. For this purpose we define

Definition 1.2.20 (Mean squared error estimator)**.** [2] Given a sample Y of Definition 1.2.19 model the mean squared error is defined as

$$\mathrm{MSE}(X,\theta) = \frac{1}{n}\|Y - X^T\theta\|_2^2$$

Following on from the above, we can therefore define the mean square error estimator as

$$\hat{\theta}_{\mathrm{MSE}}(X) := \min_{\theta\in\Theta} \mathrm{MSE}(X,\theta).$$

Proposition 1.2.6. Given a random sample $Y \in \mathbb{R}^n$ and a features matrix X of the simple linear regression model, if X has maximum rank, then the estimator in Definition 1.2.20 has the following explicit expression:

$$\hat{\theta}_{\mathrm{MSE}}(X) = (X^TX)^{-1}X^TY$$

Proof. First of all we calculate the gradient of the least square error

$$\begin{aligned}\nabla\|Y - X\theta\|_2^2 =&\nabla((Y - X\theta)^T(Y - X\theta))\\ =&\nabla(Y^TY - \theta^TX^TY - Y^TX\theta + \theta^TX^TX\theta)\\ =&-\nabla(\theta^TX^TY) - \nabla(Y^TX\theta) + \nabla(\theta^TX^TX\theta))\\ =&-(X^TY) - (Y^TX)^T + (X^TX + (X^tX)^T)\theta\\ =&-2X^TY + 2X^TX\theta\end{aligned}$$

Noting that the matrix X^TX is positive defined (since rank(X) is maximum) it is possible to make explicit the critical points of the least square error. By setting the gradient equal to zero and solving for theta we obtain

$$\theta = (X^TX)^{-1}X^TY$$

To establish whether this critical point is a local minimum we compute the Hessian of the least square error at that point. The Hessian turns out to be

$$\nabla^2(\|Y - X^T\theta\|^2) = \nabla(\nabla\|Y - X^T\theta\|^2) = \nabla(-2X^TY + 2X^tX\theta) = \nabla(2X^TX\theta) = 2X^TX$$

Then, recalling that X^TX is positive defined it follows that $\hat{\theta}$ is a local minimum, and since the least square error is a strictly convex function in theta, it is the global minimum. $\square$

[2] We note that despite the name, this is different from Definition 1.2.8

Proposition 1.2.7. The estimator $\hat{\theta}_{\text{MSE}}$ defined in 1.2.20 with a matrix of features X of maximum rank is an unbiased estimator of the parameter θ.

Proof. Denoting by X_i the i-th column of the matrix and recalling that $Y_i \sim N(X_i^T\theta, \sigma^2)$ we can compute the expected value of the random sample

$$\mathbb{E}(Y) = (\mathbb{E}(Y_1), \ldots, \mathbb{E}(Y_n)) = (X_1^T\theta, \ldots, X_n^T\theta) = X\theta$$

Under the assumption of maximality of the rank of X we can use the Proposition 1.2.6, so it immediately follows that

$$\begin{aligned}\text{bias}(\hat{\theta}_{\text{MSE}}) &= \mathbb{E}(\hat{\theta}_{\text{MSE}}) - \theta \\ &= \mathbb{E}((X^TX)^{-1}X^TY - \theta) \\ &= (X^TX)^{-1}X^T\mathbb{E}(Y) - \theta \\ &= (X^TX)^{-1}X^TX\theta - \theta \\ &= \theta - \theta = 0\end{aligned}$$

□

Definition 1.2.21 (Ridge regression estimator)**.** The ridge regression consists of a simple linear regression to which a penalty term in norm l_2 is added to the quantity to minimize in order to calculate the estimator. Let $\lambda \in \mathbb{R}^+$, the θ parameter estimator for ridge regressions for the features matrix X and the random sample Y is given by:

$$\hat{\theta}_{\text{ridge}}(X) := \min_{\theta \in \Theta} (\text{MSE}(X, \theta) + \lambda ||\theta||_2^2)$$

where λ is called the model regularization parameter.

Proposition 1.2.8. Given a random sample $Y \in \mathbb{R}^n$ and a feature matrix X of the ridge regression model[3], then the ridge estimator has the following explicit expression

$$\hat{\theta}_{\text{ridge}}(X) = (X^TX + \lambda I)^{-1}X^TY$$

Proof. We first calculate the gradient of the ridge error

$$\begin{aligned}\nabla(\|Y - X\theta\|^2 + \lambda\|\theta\|^2) =& \nabla((Y - X\theta)^t(Y - X\theta) + \lambda(\theta^t\theta)) \\ =& \nabla(Y^tY - \theta^tX^tY - Y^tX\theta + \theta^tX^tX\theta + \lambda(\theta^t\theta)) \\ =& - \nabla(\theta^tX^tY) - \nabla(Y^tX\theta) + \nabla(\theta^tX^tX\theta) + \lambda\nabla(\theta^t\theta) \\ =& - X^tY - (Y^tX)^t + (X^tX + (X^tX)^t)\theta + 2\lambda\theta \\ =& - 2X^tY + 2X^tX\theta + 2\lambda\theta\end{aligned}$$

We now want to show that $X^tX + \lambda I$ is positive definite, in fact for a generic vector $\mathbf{v} \in \mathbb{R}^n$

$$\mathbf{v}^t(X^tX + \lambda I)\mathbf{v} = (X\mathbf{v})^t(X\mathbf{v}) + \lambda\mathbf{v}^t\mathbf{v} \geq 0$$

[3] We note that unlike the mean squared error estimator, the ridge estimator does not require the rank of X to be maximum

And since it is a sum of non-negative terms we have

$$(X\mathbf{v})^t(X\mathbf{v}) + \lambda \mathbf{v}^t\mathbf{v} = 0 \Leftrightarrow \begin{cases} (X\mathbf{v})^t(X\mathbf{v}) = 0 \\ \lambda \mathbf{v}^t\mathbf{v} = 0 \end{cases} \Leftrightarrow \mathbf{v} = 0$$

Then the matrix is invertible and for all $\lambda \in \mathbb{R}^+$ it is possible to make explicit the critical points of the ridge error. By setting the gradient equal to zero and solving for theta we obtain

$$\theta = (X^T X + \lambda I)^{-1} X^T Y$$

To establish whether this critical point is a local minimum we compute the Hessian of the least square error at that point. The Hessian turns out to be

$$\begin{aligned} \nabla(\nabla(\|Y - X\theta\|^2 + \lambda\|\theta\|^2)) &= \nabla(-2X^tY + 2X^tX\theta + 2\lambda\theta) \\ &= \nabla(2X^tX\theta + 2\lambda\theta) \\ &= 2\nabla(X^tX\theta) + 2\lambda\nabla(\theta) \\ &= 2(X^tX)^t + 2\lambda I \\ &= 2(X^tX + \lambda I) \end{aligned}$$

Finally, recalling that $X^TX+\lambda I$ is positive definite it follows that $\hat{\theta}$ for all $\lambda \in \mathbb{R}^+$ is a local minimum, and since the least square error is a strictly convex function in theta, it is the global minimum. □

Remark 1.2.6. The ridge estimator is biased if $\lambda \neq 0$. Indeed

$$\begin{aligned} \text{bias}(\hat{\theta}_{\text{ridge}}) &= \mathbb{E}(\hat{\theta}_{\text{ridge}} - \theta) \\ &= (X^TX + \lambda I)^{-1}X^T\mathbb{E}(Y) - \theta \\ &= (X^TX + \lambda I)^{-1}X^TX\theta - \theta \\ &= ((X^TX + \lambda I)^{-1}X^TX - I)\theta \neq 0. \end{aligned}$$

Definition 1.2.22 (Lasso regression)**.** The lasso regression consists of a simple linear regression to which a penalty term in norm l_1 is added. Let $\lambda \in \mathbb{R}^+$, the θ parameter estimator for lasso regressions for the features matrix X and the random sample Y is given by:

$$\hat{\theta}_{\text{lasso}}(X) := \min_{\theta\in\Theta} (\text{MSE}(X, \theta) + \lambda||\theta||_1)$$

Where λ is called the model regularization parameter.

Definition 1.2.23 (Elastic net regression)**.** Elastic net regression consists of a union of the ridge and lasso regression, it is therefore a simple linear regression to which both penalty terms are added. Let $\lambda_1, \lambda_2 \in \mathbb{R}^+$, the θ parameter estimator for ridge regressions for the features matrix X and the random sample Y is given by:

$$\hat{\theta}_{\text{elastic}}(X) := \min_{\theta\in\Theta} (\text{MSE}(X, \theta) + \lambda_1||\theta||_2^2 + \lambda_2||\theta||_1).$$

Due to the presence of the term $||\cdot||_1$ in the regression methods lasso, Definition 1.2.22, and elastic net, Definition 1.2.23, it is not generally possible to provide an explicit form of the estimator, for this reason various minimization techniques are used, such as that of gradient descent which is reported below.

Definition 1.2.24 (Gradient descent)**.** Gradient descent is a method of minimization of convex and differentiable functions. Let $f \in C^1(\mathbb{R}^p, \mathbb{R})$ be a convex function, let $\mathbf{x}_0 \in \mathbb{R}^p$ be a point of the domain of f. Gradient descent consists in the calculation of the following sequence:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha \nabla f(\mathbf{x}_n)$$

where $\alpha \in \mathbb{R}^+$ is a free parameter called step. The iteration is continued until the following halting condition is reached

$$||\nabla f(\mathbf{x}_n)||_2 < \varepsilon$$

where $\varepsilon \in \mathbb{R}^+$ is a free parameter called tolerance.

Definition 1.2.25 (Lipschitz continuous)**.** Let $f : \mathbb{R}^p \to \mathbb{R}$. f is said to be Lipschitz continuous if $\exists L \in \mathbb{R}^+$ such that

$$|f(\mathbf{x}) - f(\mathbf{y})| \leq L\|\mathbf{x} - \mathbf{y}\|, \text{ for all } \mathbf{x}, \mathbf{y} \in \mathbb{R}^p$$

Lemma 1.2.2. Let $f \in C^1(\mathbb{R}^p, \mathbb{R})$ be a function with Lipschitz continuous gradient, then

$$f(\mathbf{y}) \leq f(\mathbf{x}) + (\mathbf{y} - \mathbf{x})^T \nabla f(\mathbf{x}) + \frac{L}{2}\|\mathbf{y} - \mathbf{x}\|_2^2, \text{ for all } \mathbf{x}, \mathbf{y} \in \mathbb{R}^p.$$

Proof. Let $\mathbf{x}, \mathbf{y} \in U$, we develop f in Taylor series with integral remainder in $\mathbf{x}$

$$f(\mathbf{y}) = f(\mathbf{x}) + \int_0^1 (\mathbf{y} - \mathbf{x})^T \nabla f(\mathbf{x} + t(\mathbf{y} - \mathbf{x}))dt$$

Then

$$\begin{aligned}
|f(\mathbf{y}) - f(\mathbf{x}) - (\mathbf{y} - \mathbf{x})^T \nabla f(\mathbf{x})| &= |\int_0^1 (\mathbf{y} - \mathbf{x})^T (\nabla f(\mathbf{x} + t(\mathbf{y} - \mathbf{x})) - \nabla f(\mathbf{x}))dt| \\
&\leq \int_0^1 \big|(\mathbf{y} - \mathbf{x})^T (\nabla f(\mathbf{x} + t(\mathbf{y} - \mathbf{x})) - \nabla f(\mathbf{x}))\big|dt \\
&\leq \int_0^1 \|\mathbf{y} - \mathbf{x}\|_2 \|\nabla f(\mathbf{x} + t(\mathbf{y} - \mathbf{x})) - \nabla f(\mathbf{x})\|_2 dt \\
&\leq L\|\mathbf{y} - \mathbf{x}\|_2 \int_0^1 \|\mathbf{x} + t(\mathbf{y} - \mathbf{x}) - \mathbf{x}\|_2 dt \\
&= L\|\mathbf{y} - \mathbf{x}\|_2^2 \int_0^1 t dt \\
&= \frac{L}{2}\|\mathbf{y} - \mathbf{x}\|_2^2
\end{aligned}$$

□

Proposition 1.2.9. Let $f \in C^1(\mathbb{R}^p, \mathbb{R})$ be a non-negative, convex function with Lipschitz continuous gradient. Then the Algorithm 1.2.24 with step $\alpha = \frac{1}{L}$ halts in at least $O(\varepsilon^{-2})$ iterations.

Proof. Setting $\alpha = \frac{1}{L}$, by Lemma 1.2.2 we obtain

$$\begin{aligned} f(\mathbf{x}_{n+1}) &\leq f(\mathbf{x}_n) - (\mathbf{x}_{n+1} - \mathbf{x}_n)^T \nabla f(\mathbf{x}_n) + \frac{L}{2} ||\mathbf{x}_{n+1} - \mathbf{x}_n||_2^2 \\ &= f(\mathbf{x}_n) - (\alpha \nabla f(\mathbf{x}_n))^T \nabla f(\mathbf{x}_n) + \frac{L}{2} ||\alpha \nabla f(\mathbf{x}_n)||_2^2 \\ &= f(\mathbf{x}_n) - \alpha ||\nabla f(\mathbf{x}_n)||_2^2 + \frac{L\alpha^2}{2} ||\nabla f(\mathbf{x}_n)||_2^2 \\ &= f(\mathbf{x}_n) + \left(\frac{L\alpha^2 - 2\alpha}{2}\right) ||\nabla f(\mathbf{x}_n)||_2^2 \\ &= f(\mathbf{x}_n) - \frac{1}{2L} ||\nabla f(\mathbf{x}_n)||_2^2 \end{aligned}$$

from which it immediately follows that

$$\frac{1}{2L} ||\nabla f(\mathbf{x}_n)||_2^2 \leq f(\mathbf{x}_n) - f(\mathbf{x}_{n+1}).$$

This inequality implies that f strictly decreases with each iteration of gradient descent until it reaches the optimal value $f(x^*) \geq 0$ which, due to the convexity of the function, will be a point of global minimum. Summing over the iterations up to the N-th we therefore obtain that for each $N \in \mathbb{N}$

$$\frac{1}{2L} \sum_{n=0}^{N} ||\nabla f(\mathbf{x}_n)||_2^2 \leq \sum_{n=0}^{N} (f(\mathbf{x}_n) - f(\mathbf{x}_{n+1})) = f(\mathbf{x}_0) - f(\mathbf{x}_{N+1}) \leq f(\mathbf{x}_0) - f(\mathbf{x}^*)$$

then for $N \to +\infty$ the series converges, in particular we must have that

$$\lim_{n \to +\infty} ||\nabla f(\mathbf{x}_n)||_2^2 = 0.$$

As a result of the previous inequality we note that

$$f(\mathbf{x}_0) - f(\mathbf{x}^*) \geq \frac{1}{2L} \sum_{n=0}^{N} ||\nabla f(\mathbf{x}_n)||_2^2 \geq \frac{1}{2L} \sum_{n=0}^{N} ||\nabla f(\mathbf{x}_N)||_2^2 \geq \frac{N+1}{2L} ||\nabla f(\mathbf{x}_N)||_2^2$$

from which it follows that

$$||\nabla f(\mathbf{x}_N)||_2^2 \leq \sqrt{\frac{2L(f(\mathbf{x}_0) - f(\mathbf{x}^*))}{N+1}}.$$

Finally

$$\sqrt{\frac{2L(f(\mathbf{x}_0) - f(\mathbf{x}^*))}{N+1}} = \varepsilon \Rightarrow N = O\left(\frac{1}{\varepsilon^2}\right)$$

□

1.3 Information theory

Information theory, formally introduced by Claude Shannon in *A Mathematical Theory of Communication*, introduces effective tools and methods for analysis of dependencies between random events with the benefit of be able to highlight dependency relationships even when these are not linear. This makes the concepts introduced by information theory suitable for use in the inference problem of gene regulatory networks in which the causal relationships between elements of the network present varied and complex dynamics. In this section we therefore introduce the fundamental definitions of the theory and an extension of these concepts introduced by Paul L. Williams, Randall D. Beer in [32]. We first report a well known results of the theory of probability.

Theorem 1.3.1 (Jensen's inequality)**.** Let X be a discrete random variable such that $\mathbb{E}(X) < +\infty$, φ a convex function and ψ a concave function, then

$$\varphi(\mathbb{E}(X)) \leq \mathbb{E}(\varphi(X)) \qquad \psi(\mathbb{E}(X)) \geq \mathbb{E}(\psi(X))$$

Equality holds if and only if X is constant.

1.3.1 General definitions

In the development of the entire section we deliberately use the term *log* without indicating its base, assuming it is in $\mathbb{N}$ and greater than one, since definitions that follows differ by a constant factor which depends on the base. We note however that in classical information theory the standard base is 2.

Definition 1.3.1 (Entropy)**.** Given a discrete random variable X we define the entropy of X, $H(X)$ as:

$$H(X) := -\sum_{x\in\mathcal{X}} p(x)\log p(x).$$

Definition 1.3.2 (Joint entropy)**.** Given two discrete random variables X and Y we define the joint entropy $H(X,Y)$ as:

$$H(X,Y) := -\sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x,y)\log p(x,y).$$

Definition 1.3.3 (Conditional entropy)**.** Given two discrete random variables X and Y we define the conditional entropy $H(X|Y)$ as:

$$H(Y|X) := -\sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x,y)\log p(y|x).$$

Definition 1.3.4 (Informational Divergence)**.** Given two discrete discrete probability distributions p and q having the same support $\mathcal{X}$, the informational divergence between them is defined as

$$D(p \parallel q) := \sum_{x\in\mathcal{X}} p(x)\log\frac{p(x)}{q(x)}.$$

Theorem 1.3.2 (Gibbs' inequality)**.** Given two discrete probability distributions $p = (p_1, \dots, p_n), q = (q_1, \dots, q_n) \in \mathbb{R}^n$ we have that

$$D(p \parallel q) \geq 0 \qquad \text{and} \qquad D(p \parallel q) = 0 \Leftrightarrow p_i = q_i, \text{ for all } i \in \{1, 2, \dots, n\}.$$

Proof. From the fact that *log* is a strictly concave function, using the Jensen's inequality we have

$$\begin{aligned}-D(p \parallel q) &= -\sum_{i=1}^{n} p_i \log\left(\frac{p_i}{q_i}\right) = \sum_{i=1}^{n} p_i \log\left(\frac{q_i}{p_i}\right) \leq \log \sum_{i=1}^{n} p_i \frac{q_i}{p_i} \\ &= \log \sum_{i=1}^{n} q_i = \log 1 = 0\end{aligned}$$

furthermore, from Jensen's inequality we have that

$$D(p \parallel q) = 0 \Leftrightarrow \frac{p_i}{q_i} \equiv c, \text{ for all } i \in \{1, 2, \ldots, n\}$$

but since they are both probability distributions we have $c = 1$, hence the thesis. □

Proposition 1.3.1 (Log sum inequality)**.** Let $p_1, p_2, \ldots, p_n$ and $q_1, q_2, \ldots, q_n$ be non-negative real numbers, then

$$\sum_{i=1}^{n} p_i \log \frac{p_i}{q_i} \geq \sum_{i=1}^{n} p_i \left(\log \frac{\sum_{j=1}^{n} p_j}{\sum_{j=1}^{n} q_j}\right).$$

Proof. Let us first introduce the following notations

$$p := \sum_{i=1}^{n} p_i \qquad q := \sum_{i=1}^{n} q_i \qquad \hat{q}_i := \frac{q_i}{q}$$

we now note that $\hat{q}_i$ is a discrete probability distribution. In fact, the non-negativity of $\hat{q}_i$ immediately follows from the non-negativity of q_i, it also follows immediately from the definition that $\sum_{i=1}^{n} \hat{q}_i = 1$. We denote by $f(x) := x \log x$. From the fact that f is a convex function and that $\hat{q}_i$ is a discrete probability distribution, applying Jensen's inequality we obtain the following

$$\sum_{i=1}^{n} p_i \log \frac{p_i}{q_i} = \sum_{i=1}^{n} q_i f\left(\frac{p_i}{q_i}\right) = q \sum_{i=1}^{n} \hat{q}_i f\left(\frac{p_i}{q_i}\right) \geq q f\left(\sum_{i=1}^{n} \hat{q}_i \frac{p_i}{q_i}\right) = q f\left(\frac{p}{q}\right) = p \log \frac{p}{q}$$

□

Proposition 1.3.2. The informational divergence $D(p \parallel q)$ is convex in the (p, q) pair of the two distributions. That is, given four distributions p_1, p_2, q_1, q_2 we have that

$$D(\lambda p_1 + (1-\lambda) p_2 \parallel \lambda q_1 + (1-\lambda) q_2) \leq \lambda D(p_1 \parallel q_1) + (1-\lambda) D(p_2 \parallel q_2).$$

Proof. Let p_1, p_2, q_1, q_2 be four discrete distributions having $\mathcal{X}$ as support and be $\lambda \in \mathbb{R}^+$, then

$$\begin{aligned}&D(\lambda p_1 + (1-\lambda) p_2 \parallel \lambda q_1 + (1-\lambda) q_2) \\ &= \sum_{x \in \mathcal{X}} (\lambda p_1(x) + (1-\lambda) p_2(x)) \log \frac{(\lambda p_1(x) + (1-\lambda) p_2(x))}{(\lambda q_1(x) + (1-\lambda) q_2(x))}\end{aligned}$$

applying the log sum inequality we obtain

$$\begin{aligned}&\leq \sum_{x \in \mathcal{X}} \left(\lambda p_1(x) \log \frac{\lambda p_1(x)}{\lambda q_1(x)} + (1-\lambda) p_2(x) \log \frac{(1-\lambda) p_2(x)}{(1-\lambda) q_2(x)}\right) \\ &= \lambda \sum_{x \in \mathcal{X}} \lambda p_1(x) \log \frac{p_1(x)}{q_1(x)} + (1-\lambda) \sum_{x \in \mathcal{X}} \lambda p_2(x) \log \frac{p_2(x)}{q_2(x)} \\ &= \lambda D(p_1 \parallel q_1) + (1-\lambda) D(p_2 \parallel q_2)\end{aligned}$$

□

Definition 1.3.5 (Mutual Information)**.** Given two discrete random variables X and Y, we define the mutual information between X and Y as:

$$I(X;Y) := D(p_{XY} \parallel p_X p_Y) = \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} p(x,y)\log\frac{p(x,y)}{p(x)p(y)}$$

where p_{XY} is the joint distribution of the two variables. From Gibbs' inequality it immediately follows that

$$I(X;Y) \geq 0 \qquad \text{and} \qquad I(X;Y) = 0 \Leftrightarrow p_{XY}(x,y) = p_X(x)p_Y(y),\ \text{for all } x\in\mathcal{X} \text{ and } y\in\mathcal{Y}$$

mutual information is therefore zero if and only if the variables are independent.

Definition 1.3.6 (Conditional mutual information)**.** Given three discrete random variables X, Y and Z we define the conditional mutual information between X and Y given Z as:

$$I(X;Y|Z) := D(p_{XY|Z} \parallel p_{X|Z}p_{Y|Z}) = \sum_{(x,y,z)\in\mathcal{X}\times\mathcal{Y}\times\mathcal{Z}} p(x,y,z)\log\frac{p(x,y|z)}{p(x|z)p(y|z)}$$

From Gibbs' inequality it immediately follows $I(X;Y|Z) \geq 0$.

Remark 1.3.1. We observe that both mutual information and conditional mutual information are by definition symmetrical in the first two variables

$$I(X;Y) = I(Y;X) \qquad\qquad I(X;Y|Z) = I(Y;X|Z).$$

Proposition 1.3.3. Given two discrete random variables X and Y we have that:

$$I(X;Y) = H(X) - H(X|Y).$$

Proof.

$$\begin{aligned}
I(X;Y) &= \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log\frac{p(x,y)}{p(x)p(y)} \\
&= \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log\frac{p(x|y)}{p(x)} \\
&= \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)(\log p(x|y) - \log p(x)) \\
&= \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log p(x|y) - \sum_{x\in X} p(x)\log p(x) \\
&= -H(X|Y) + H(X)
\end{aligned}$$

$\square$

Proposition 1.3.4. Given three discrete random variables X, Y and Z we have that:

$$I(X;Y|Z) = H(X|Z) - H(X|Y,Z).$$

Proof.

$$\begin{aligned}
I(X;Y|Z) &= \sum_{z\in\mathcal{Z}}\sum_{y\in\mathcal{Y}}\sum_{x\in\mathcal{X}} p(x,y,z)\log\frac{p(x,y|z)}{p(x|z)p(y|z)} \\
&= \sum_{z\in\mathcal{Z}}\sum_{y\in\mathcal{Y}}\sum_{x\in\mathcal{X}} p(x,y,z)\log\frac{p(y|z)p(x|y,z)}{p(x|z)p(y|z)} \\
&= \sum_{z\in\mathcal{Z}}\sum_{y\in\mathcal{Y}}\sum_{x\in\mathcal{X}} p(x,y,z)\log\frac{p(x|y,z)}{p(x|z)} \\
&= \sum_{z\in\mathcal{Z}}\sum_{y\in\mathcal{Y}}\sum_{x\in\mathcal{X}} p(x,y,z)\log p(x|y,z) - \sum_{z\in\mathcal{Z}}\sum_{y\in\mathcal{Y}}\sum_{x\in\mathcal{X}} p(x,y,z)\log p(x|z) \\
&= \sum_{z\in\mathcal{Z}}\sum_{y\in\mathcal{Y}}\sum_{x\in\mathcal{X}} p(x,y,z)\log p(x|y,z) - \sum_{z\in\mathcal{Z}}\sum_{x\in\mathcal{X}} p(x,z)\log p(x|z) \\
&= -H(X|Y,Z) + H(X|Z)
\end{aligned}$$

□

Proposition 1.3.5. Given three discrete random variables X, Y and Z we have that:

$$I(X;Z) + I(X;Y|Z) = I(X;Y) + I(X;Z|Y).$$

Proof. By applying Proposition 1.3.3 to X and the pair (Y,Z), i.e. to the random variable having the distribution p_{YZ}, we can write $I(X;(Y,Z))$ in two distinct way. In the first case

$$\begin{aligned}
I(X;(Y,Z)) =& H(X) - H(X|Y,Z) \\
=& H(X) + (-H(X|Z) + H(X|Z)) - H(X|Y,Z) \\
=& (H(X) - H(X|Z)) + (H(X|Z) - H(X|Y,Z)) \\
=& I(X;Z) + I(X;Y|Z)
\end{aligned}$$

where the last equality is obtained by applying respectively Proposition 1.3.3 and Proposition 1.3.4. Similarly, in the second case

$$\begin{aligned}
I(X;(Y,Z)) =& H(X) - H(X|Y;Z) \\
=& H(X) + (-H(X|Y) + H(X|Y)) - H(X|Y;Z) \\
=& (H(X) - H(X|Y)) + (H(X|Y) - H(X|Y;Z)) \\
=& I(X;Y) + I(X;Z|Y)
\end{aligned}$$

where the last equality is obtained by applying respectively Proposition 1.3.3 and Proposition 1.3.4.

□

Definition 1.3.7. Three random variables X, Y, Z form a Markov chain, in which case we write $X \to Y \to Z$, if

$$p(x,y|z) = p(x|z)p(y|z), \text{ for all } x \in \mathcal{X}, y \in \mathcal{Y}, z \in \mathcal{Z}.$$

Proposition 1.3.6. Let X, Y and Z be three random variables, then

$$I(X;Z|Y) = 0 \Leftrightarrow X \to Y \to Z.$$

Proof. Thesis immediately follows from definition of conditional mutual information and Gibbs' inequality

$$I(X;Z|Y) = 0 \Leftrightarrow D(p_{X,Z|Y} \parallel p_{X|Y}p_{Z/Y}) = 0 \Leftrightarrow p(x,z|y) = p(x|y)p(z|y) \Leftrightarrow X \to Y \to Z$$

□

Theorem 1.3.3 (Data Processing Theorem)**.** Given three random variables in Markov chain $X \to Y \to Z$, holds

$$I(X;Z) \leq \min\left(I(X;Y), I(Y;Z)\right)$$

Where equality holds if and only if $X \to Z \to Y$ and $Y \to X \to Z$.

Proof. Rearranging the variables in Proposition 1.3.5 in two different ways we obtain the following

$$\begin{cases} I(X;Y|Z) = I(X;Y) + I(X;Z|Y) - I(X;Z) \\ I(Z;Y|X) = I(Z;Y) + I(Z;X|Y) - I(Z;X) \end{cases}$$

We now observe that by hypothesis $X \to Y \to Z$, therefore by Proposition 1.3.6 and symmetry of conditional mutual information respect to the first two variables the two terms $I(X;Z|Y)$ and $I(Z;X|Y)$ cancels out and the system becomes

$$\begin{cases} I(X;Y|Z) = I(X;Y) - I(X;Z) \\ I(Z;Y|X) = I(Z;Y) - I(Z;X) \end{cases} \tag{1.4}$$

The thesis therefore follows from the non-negativity of conditional mutual information and the symmetry of mutual information

$$\begin{cases} 0 \leq I(X;Y|Z) = I(X;Y) - I(X;Z) \\ 0 \leq I(Z;Y|X) = I(Z;Y) - I(Z;X) \end{cases}$$

The equality case follows by applying Proposition 1.3.6 to equality (1.4). □

1.3.2 Partial information decomposition

In the context we are dealing with, the concept of mutual information previously introduced plays an important role, as it allows us to provide a measure of dependence between variables. In some cases, however, there is the need to describe this measure in case of a greater number of variables, for which it is necessary to extend the concept of mutual information. The main attempt to generalize mutual information in the case of multiple interactions between variables is interaction information

$$II(X_1; X_2; \ldots; X_n) := I(X_1; \ldots; X_{n-1}) - I(X_1; \ldots; X_{n-1}|X_n).$$

However, despite being symmetrical, interaction information may be negative, which makes it difficult to provide an interpretation since it is not clear what it means for one variable to provides "negative information" about another. To date, there is no generally accepted extension of information theory to the case of multiple interactions. An alternative proposal to this problem is the concept of partial information decomposition (PID for short), introduced in Williams et al. [32]. Let Y be a random variable and let $X = \{X_1, \dots, X_n\}$ be a set of random variables. The proposed generalization provide a measure capable of quantifying the information that X provides on Y by decomposing it into the contributions of single variables or subsets of X.

Partial information measures

In this thesis we limit ourselves to the case of three variables, X_1, X_2, Y, for a complete discussion of the subject, we refer the reader to William et al. [32]. We can first ask ourselves how much total information does X_1 and X_2 provides about Y, the answer to this question is given by mutual information

$$I(Y; (X_1, X_2))$$

We can now ask ourselves to what extent X_1 and X_2 contribute to the total information. For this purpose, three distinct possibilities are provided. The first is the *Unique information*, which expresses the concept of the 'portion of total information' provided by a single variable, without loss of generality, it could happen that X_1 provides information on Y that X_2 does not. The second is *Redundancy*, the contribution of the two variables to the total information may overlap to some extent, i.e. part of the total information due to X is provided by both X_1 and X_2, thus being redundant. The third is *Synergy*, which represents that portion of total information that would not be obtainable from the variables if taken individually, but only if considered concurrently.

Definition 1.3.8 (Specific information)**.** Given two discrete random variables Y and X, taking into consideration a value y, the specific information of y with respect to X is defined as

$$I_{\text{spec}}(Y = y; X) := D(p_{X|Y=y} \parallel p_X) = \sum_{x \in \mathcal{X}} p(x|y) \log \frac{p(y|x)}{p(y)}$$

In other words, specific information quantifies the information that X provides with respect to a particular outcome y of the variable Y. We note that from Gibbs' inequality immediately follows that

$$I_{\text{spec}}(Y = y; X) \geq 0.$$

Definition 1.3.9 (Redundancy)**.** Let Y, X_1 and X_2 be discrete random variables. Redundancy (or minimal information) is defined as

$$I_{\min}(Y; X_1, X_2) := \sum_{y \in \mathcal{Y}} p(y) \min_{i \in \{1,2\}} \left(I_{\text{spec}}(Y = y; X_i)\right)$$

this quantity is representative of the idea mentioned above, which sees redundancy as the information shared by th variables X_1 and X_2, which can be seen as the minimum information that any source provides for each outcome of Y. We note that from the non negativity of specific information it immediately follows that

$$I_{\min}(Y; X_1, X_2) \geq 0.$$

Proposition 1.3.7. Let Y, X_1 and X_2 be discrete random variables, then the following holds

$$I_{\min}(Y; X_1, X_2) \leq I(Y; X_i), \text{ for all } i \in \{1, 2\}$$

this proposition allows us to understand even better the meaning of redundancy. In fact, since the redundancy is the information shared between the variables, it is reasonable to expect that it is bounded by the mutual information provided by each variable.

Proof. For each $j \in \{1, 2\}$ it holds that

$$\begin{aligned} I_{\min}(Y; X_1, X_2) &= \sum_{y \in \mathcal{Y}} p(y) \min_{i \in \{1,2\}} (I_{\text{spec}}(Y = y; X_i)) \\ &\leq \sum_{y \in \mathcal{Y}} p(y) I_{\text{spec}}(Y = y; X_j) \\ &= \sum_{y \in \mathcal{Y}} p(y) \sum_{x \in \mathcal{X}_j} p(x|y) \log \frac{p(y|x)}{p(y)} \\ &= \sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}_j} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} = I(Y; X_j) \end{aligned}$$

□

Definition 1.3.10 (Unique information)**.** Let Y, X_1 and X_2 be discrete random variables. The unique information of the Y variable with respect to the X_i variable for any fixed $(i, j) \in \{(1, 2), (2, 1)\}$, is defined as

$$\text{Unique}_{X_j}(Y; X_i) := I(Y, X_i) - I_{\min}(Y; X_i, X_j).$$

Intuitively, taking into consideration the unique information of Y variable with respect to X_1, it can be seen as that part of the mutual information between them that is not shared with the X_2 variable. We note that, for what is shown in Proposition 1.3.7 it immediately follows that for all $(i, j) \in \{(1, 2), (2, 1)\}$

$$\text{Unique}_{X_j}(Y; X_i) \geq 0.$$

Definition 1.3.11 (Synergy)**.** Let Y, X_1 and X_2 be discrete random variables. The synergy of the Y variable with respect to X_1 and X_2, is defined as

$$\text{Synergy}(Y; X_1, X_2) := I(Y; (X_1, X_2)) - I_{\min}(Y; X_1, X_2) - \text{Unique}_{X_2}(Y; X_1) - \text{Unique}_{X_1}(Y; X_2).$$

As we had informally suggested previously, the synergy is that part of information obtainable only by taking into consideration the contribution of both variables X_1 and X_2.

Proposition 1.3.8. Let Y, X_1 and X_2 be discrete random variables, then $\text{Synergy}(Y; X_1, X_2) \geq 0$.

Proof. We note first of all that

$$\begin{aligned} &\text{Synergy}(Y; X_1, X_2) \\ &= I(Y; (X_1, X_2)) - I_{\min}(Y; X_1, X_2) - \text{Unique}_{X_2}(Y; X_1) - \text{Unique}_{X_1}(Y; X_2) \\ &= I(Y; (X_1, X_2)) - I_{\min}(Y; X_1, X_2) - I(Y; X_1) - I(Y; X_2) + 2I_{\min}(Y; X_1, X_2) \\ &= I(Y; (X_1, X_2)) - I(Y; X_1) - I(Y; X_2) + I_{\min}(Y; X_1, X_2). \end{aligned}$$

Let us develop the following quantity

$$\begin{aligned}
&I(Y;X_1) + I(Y;X_2) - I_{\min}(Y;X_1,X_2) \\
&= \sum_{y\in\mathcal{Y}} \sum_{x_1\in\mathcal{X}_1} p(y,x_1) \log \frac{p(y,x_1)}{p(y)p(x_1)} + \sum_{y\in\mathcal{Y}} \sum_{x_2\in\mathcal{X}_2} p(y,x_2) \log \frac{p(y,x_2)}{p(y)p(x_2)} \\
&- \sum_{y\in\mathcal{Y}} p(y) \min\left(I_{\text{spec}}(y;X_1), I_{\text{spec}}(y;X_2)\right) \\
&= \sum_{y\in\mathcal{Y}} p(y) \sum_{x_1\in\mathcal{X}_1} p(x_1|y) \log \frac{p(y,x_1)}{p(y)p(x_1)} + \sum_{y\in\mathcal{Y}} p(y) \sum_{x_2\in\mathcal{X}_2} p(x_2|y) \log \frac{p(y,x_2)}{p(y)p(x_2)} \\
&- \sum_{y\in\mathcal{Y}} p(y) \min\left(I_{\text{spec}}(y;X_1), I_{\text{spec}}(y;X_2)\right) \\
&= \sum_{y\in\mathcal{Y}} p(y) I_{\text{spec}}(y;X_1) + \sum_{y\in\mathcal{Y}} p(y) I_{\text{spec}}(y;X_2) - \sum_{y\in\mathcal{Y}} p(y) \min\left(I_{\text{spec}}(y;X_1), I_{\text{spec}}(y;X_2)\right) \\
&= \sum_{y\in\mathcal{Y}} p(y) \big(I_{\text{spec}}(y;X_1) + I_{\text{spec}}(y;X_2) - \min\left(I_{\text{spec}}(y;X_1), I_{\text{spec}}(y;X_2)\right)\big) \\
&= \sum_{y\in\mathcal{Y}} p(y) \max\left(I_{\text{spec}}(y;X_1), I_{\text{spec}}(y;X_2)\right)\big).
\end{aligned}$$

We also note that from Gibbs' inequality it follows that

$$\begin{aligned}
&\sum_{x_1\in\mathcal{X}_1} \sum_{x_2\in\mathcal{X}_2} p(y,x_1,x_2) \log \frac{p(y,x_1,x_2)}{p(y)p(x_1,x_2)} \\
&= \sum_{x_1\in\mathcal{X}_1} \sum_{x_2\in\mathcal{X}_2} p(y,x_1,x_2) \log \frac{p(y,x_1)p(x_2|y,x_1)}{p(y)p(x_1)p(x_2|x_1)} \\
&= \sum_{x_1\in\mathcal{X}_1} \sum_{x_2\in\mathcal{X}_2} p(y,x_1,x_2) \log \frac{p(y,x_1)}{p(y)p(x_1)} + \sum_{x_1\in\mathcal{X}_1} p(y,x_1) \sum_{x_2\in\mathcal{X}_2} p(x_2|y,x_1) \log \frac{p(x_2|y,x_1)}{p(x_2|x_1)} \\
&= \sum_{x_1\in\mathcal{X}_1} p(y,x_1) \log \frac{p(y,x_1)}{p(y)p(x_1)} + \sum_{x_1\in\mathcal{X}_1} p(y,x_1) D(p_{X_2|Y=y,X_1=x_1} \parallel p_{X_2|X_1=x_1}) \\
&\geq \sum_{x_1\in\mathcal{X}_1} p(y,x_1) \log \frac{p(y,x_1)}{p(y)p(x_1)} = p(y) \sum_{x_1\in\mathcal{X}_1} p(x_1|y) \log \frac{p(y,x_1)}{p(y)p(x_1)} = p(y) I_{\text{spec}}(y;X_1)
\end{aligned}$$

by symmetry

$$\sum_{x_1\in\mathcal{X}_1} \sum_{x_2\in\mathcal{X}_2} p(y,x_1,x_2) \log \frac{p(y,x_1,x_2)}{p(y)p(x_1,x_2)} \geq p(y) I_{\text{spec}}(y;X_2).$$

Putting together what has been shown so far, it follows that

$$\begin{aligned}
&\text{Synergy}(Y;X_1,X_2) \\
&= I(Y;(X_1,X_2)) - I(Y;X_1) - I(Y;X_2) + I_{\min}(Y;X_1,X_2) \\
&= \sum_{y\in\mathcal{Y}} \sum_{x_1\in\mathcal{X}_1} \sum_{x_2\in\mathcal{X}_2} p(y,x_1,x_2) \log \frac{p(y,x_1,x_2)}{p(y)p(x_1,x_2)} - \sum_{y\in\mathcal{Y}} p(y) \max\left(I_{\text{spec}}(y;X_1), I_{\text{spec}}(y;X_2)\right)\big) \\
&= \sum_{y\in\mathcal{Y}} \left(\sum_{x_1\in\mathcal{X}_1} \sum_{x_2\in\mathcal{X}_2} p(y,x_1,x_2) \log \frac{p(y,x_1,x_2)}{p(y)p(x_1,x_2)} - p(y) \max\left(I_{\text{spec}}(y;X_1), I_{\text{spec}}(y;X_2)\right)\big) \right) \geq 0
\end{aligned}$$

□

1.3.3 Information estimators

When approaching the problem of inference of gene regulatory networks with the tools of information theory, one has to estimate quantities such as entropy and mutual information from a gene expression dataset. Despite their discrete nature, due to the normalisation processes that take place on them, data from microarrays and RNA-Seq[4] are modelled as continuous random variables. For continuous random variables, there are equivalent definitions of entropy and mutual information, respectively.

Definition 1.3.12 (Differential entropy). Let X be a continuous random variable with density f and support $\mathcal{X} \subseteq \mathbb{R}$. Assuming the integrability of the function, the differential entropy of X is defined as

$$h(X) := -\int_{\mathcal{X}} f(x) \log f(x) dx.$$

Definition 1.3.13 (Differential mutual information). Let X, Y be continuous random variables with joint density f_{XY}, marginals f_X, f_Y and support $\mathcal{X}, \mathcal{Y} \subseteq \mathbb{R}$. Assuming the integrability of the function, the differential mutual information of X is defined as

$$I(X;Y) = \iint_{\mathcal{X}\times\mathcal{Y}} f(x,y) \log \frac{f(x,y)}{f(x)f(y)} dxdy.$$

The choice of representing gene expression values as continuous random variables makes the tools of classical information theory, which requires random variables to be discrete, seemingly useless. A possible solution to this problem is provided by the following theorem in Thomas M. Cover, Joy A. Thomas [27]. Let us first give the following definition

Definition 1.3.14 (Uniform discretisation of a continuous variable). Let X be a continuous random variable with density f and support $\mathcal{X} \subseteq \mathbb{R}^d$, be $\Delta \in \mathbb{R}^+$ and consider the partition of $\mathcal{X}$ into adjacent d-dimensional cubes $\mathcal{X}_i$ of volume Δ^d. If f is continuous on $\mathcal{X}_i$ for each i. From Lemma 1.2.1, for each i

$$f(x_i)\Delta^d = \int_{\mathcal{X}_i} f(x)dx \text{ exists } x_i \in \mathcal{X}_i.$$

A uniform discretisation of X of parameter Δ, denoted X^Δ, is defined as the discrete random variable defined by the following probability distribution

$$\mathbb{P}(X^\Delta = x_i) := \mathbb{P}(X \in \mathcal{X}_i) = f(x_i)\Delta.$$

Remark 1.3.2. Using the notation of the previous definition, we then note that the entropy of the uniform discretisation of a random variable X can be written as follows

$$\begin{aligned} H(X^\Delta) &= -\sum_{x_i \in \mathcal{X}} \mathbb{P}(X^\Delta = x_i) \log \mathbb{P}(X^\Delta = x_i) = -\sum_{x_i \in \mathcal{X}} f(x_i)\Delta \log f(x_i)\Delta \\ &= -\sum_{x_i \in \mathcal{X}} f(x_i)\Delta \log f(x_i) - \sum_{x_i \in \mathcal{X}} f(x_i)\Delta \log \Delta \\ &= -\sum_{x_i \in \mathcal{X}} \Delta f(x_i) \log f(x_i) - \log \Delta. \end{aligned}$$

[4] Definitions in Chapter 2

The reason why it is reasonable to approximate information measures of a continuous random variable by means of information measures of its discretisation is due to the following results in [27].

Theorem 1.3.4. Let X be a continuous random variable with density f and support $\mathcal{X}$. If f is Riemann-integrable then as $\varDelta \to 0$.

$$H(X^{\varDelta}) + \log \varDelta \to h(X).$$

Corollary 1.3.1. Let X, Y be continuous random variables with density f_X, f_Y and support $\mathcal{X}, \mathcal{Y}$. If f_X and f_Y are Riemann-integrable then as $\varDelta \to 0$.

$$I(X^{\varDelta}; Y^{\varDelta}) \to I(X; Y).$$

Discrete variable estimator

As a result of what has just been shown, we can therefore consider reasonable to use the discretisation of continuous random variables for the inference problem of gene regulatory networks. This approach allows discrete measures of information to be used by means of the histogram estimator introduced in Section 1.2.1. Using the definition of the discretization of a random variable, we can define the following estimators.

Proposition 1.3.9. Let $\{X_1, X_2, \dots, X_n\}$ be a random sample of random variables having discrete probability distribution p having the finite set $\mathcal{X}$ as support. The following is an unbiased estimator for $p(x)$.

$$\hat{p}_n(x) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\{x\}}(X_i).$$

Proof. For each $x \in \mathcal{X}$ it holds that

$$\mathbb{E}(\hat{p}_n(x)) = \mathbb{E}\left(\frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\{x\}}(X_i)\right) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}(\mathbb{1}_{\{x\}}(X_i)) = \frac{1}{n} \sum_{i=1}^{n} p(x) = p(x).$$

□

Definition 1.3.15 (Plug-in estimator for entropy)**.** Let $\{X_1, X_2, \dots, X_n\}$ be a random sample of random variables having discrete probability distribution p having the finite set $\mathcal{X}$ as support. The Plug-in estimator of the entropy $H(p)$ is defined as

$$\hat{H}(p) := H(\hat{p}_n) = \sum_{x \in \mathcal{X}} \hat{p}_n(x) \log(\hat{p}_n(x)).$$

Proposition 1.3.10. The Plug-in entropy estimator $\hat{H}$ in Definition 1.3.15, underestimates the entropy $H(p)$.

Proof. From the linearity of the expected value and Jensen's inequality follows that

$$\begin{aligned}\mathbb{E}(\hat{H}) = \mathbb{E}\left(\sum_{x \in \mathcal{X}} \hat{p}_n(x) \log(\hat{p}_n(x))\right) &= \sum_{x \in \mathcal{X}} \mathbb{E}(\hat{p}_n(x) \log(\hat{p}_n(x))) \leq \sum_{x \in \mathcal{X}} \mathbb{E}(\hat{p}_n(x)) \log(\mathbb{E}(\hat{p}_n(x)))) \\ &= \sum_{x \in \mathcal{X}} p(x) \log(p(x)) = H(p).\end{aligned}$$

□

We now show a result for the proof of which we refer to Aryaman Arora et al. [2].

Proposition 1.3.11. Let $\{X_1, X_2, \dots, X_n\}$ be a random sample of random variables having discrete probability distribution p having the finite set $\mathcal{X}$ as support. Then

$$\mathrm{bias}(\hat{H}) = -\frac{|\mathcal{X}|-1}{2n} + o(n^{-1}).$$

Definition 1.3.16 (Miller Madow estimator for entropy)**.** Let $\{X_1, X_2, \dots, X_n\}$ be a random sample of random variables having discrete probability distribution p having the finite set $\mathcal{X}$ as support. The Miller Madow estimator of the entropy $H(p)$ is defined as

$$\hat{H}_{MM} := \hat{H} + \frac{|\mathcal{X}|-1}{2n}.$$

Continue variable estimator

The following non-parametric differential entropy estimator (which we will name after the authors and which we use in the ARACNE algorithm that we introduce in the third chapter) was defined and studied by Ibrahim A. Ahmad and Pi-Erh Lin in [1]. The proof[5] of the consistency of this estimator uses tools that go far beyond the scope of this thesis and which we therefore do not report.

Definition 1.3.17 (Pi-Erh Lin estimator)**.** Let $\{X_1, X_2, \dots, X_n\}$ be a random sample of random variables of a non parametric statistical model having density f such that $h(f) < +\infty$. The Ibrahim - Pi-Erh Lin estimator is defined as

$$\hat{h}_n(X) := -\frac{1}{n}\sum_{i=1}^{n} \ln \hat{f}_n(X_i)$$

where $\hat{f}_n$ is the kernel estimator introduced in Section 1.2.1

$$\hat{f}_n(X_i) = \frac{1}{nh(n)}\sum_{i=1}^{n} K\left(\frac{x - X_i}{h(n)}\right).$$

Theorem 1.3.5. If the following are fulfilled

1. $\lim_{n\to+\infty} nh(n) = +\infty$
2. $\int (\ln f(x))^2 f(x)dx < +\infty$
3. f' is continuous and $\sup_{\mathbb{R}} |f'(y)| < +\infty$
4. $\int |x|K(x)dx < +\infty$
5. $\mathbb{E}\left(\left(\frac{f'}{f}\right)^2\right) < +\infty$

Then the mean quadratic error of the Pi-Erh Lin estimator of the differential entropy $h(X)$ tends to zero as n tends to infinity.

$$^{6}\lim_{n\to+\infty} \mathrm{MSE}(\hat{h}_n(X)) = 0.$$

[5] The proof we refer to can be found on page 374 of [1].

[6] We specify that from Definition 1.2.8, $\mathrm{MSE}(\hat{h}_n(X)) = \mathbb{E}((\hat{h}_n(X) - h(X))^2)$.

1.4 Graph theory prerequisites

Graphs are discrete mathematical structures of great interest for a large number of fields. There are multiple definitions of graph, more or less restrictive, we therefore note that the definition that we report here does not represent the most general case but is chosen to adapt to the requirements of the thesis.

Definition 1.4.1 (Undirected graph)**.** An undirected graph G is given by an ordered couple of finite sets (V, E) such that $E \subseteq \{\{u, v\} : u, v \in V\}$. An element $v \in V$ is called *vertex* of the graph, an element $e \in E$ is called *edge* of the graph.

Definition 1.4.2 (Degree of a node)**.** Let $G = (V, E)$ be an undirected graph. Given $v \in V$, we call degree of v the number of edges that are incident to the node counting loops twice.

Definition 1.4.3 (Directed graph)**.** A directed graph G is given by an ordered couple of finite sets (V, E) such that $V \subseteq V \times V$. It is important to note that unlike Definition 1.4.1, in this case, E is made up of ordered pairs.

Definition 1.4.4 (In and out degree of a node)**.** Given a directed graph $G = (V, E)$ and a vertex $v \in V$, we define respectively the incoming and outgoing degree as

$$\begin{aligned} d_{\text{in}}(v) &:= |\{e \in E(G) : e = (u, v), \exists u \in V\}| \\ d_{\text{out}}(v) &:= |\{e \in E(G) : e = (v, u), \exists u \in V\}|. \end{aligned}$$

Definition 1.4.5 (Subgraph)**.** Let $G = (V, E)$ be a graph. A graph $G' = (V', E')$ is said to be a subgraph of G if $V' \subseteq V$ and $E' \subseteq E'$. Moreover, given a subset $U \subseteq V$, let us denote by $G \setminus U$ the graph to which we remove the vertices of U and the arcs incident to them, i.e., $(V \setminus U, \{e \in E : u \notin e, \text{ for all } u \in U\})$.

Definition 1.4.6 (Edge-labelled graph)**.** An edgle-labelled graph $G = (V, E, \psi)$ is a graph (V, E) equipped with a map $\psi : E \to L$ where L (called set of labels) is a non empty set. Furthermore, if L is the subset of a numerical set, in a more specific way we refer to it as an edge-weighted graph.

Definition 1.4.7 (End points)**.** Let $G = (V, E)$ be a graph. Given $e \in E$, two vertices $u, v \in V$ are called end points of e respectively if

Undirected: $e = \{u, v\}$
Directed: $e = (u, v)$ or $e = (v, u)$

In either case we say that e is incident to u and v and that u and v are neighbours. For an undirected graph we denote with $N(u) := \{v \in V : \{u, v\} \in E\}$ the set of neighbours of the vertex u.

Definition 1.4.8 (Path)**.** Given a graph $G = (V, E)$, a path p is a set of edges $\{e_1, e_2, \ldots, e_n\}$ for witch exists a set of distinct vertices $\{v_1, v_2, \ldots, v_{n+1}\}$ such that

Undirected: $e_i = \{v_i, v_{i+1}\}$ for all $i \in \{1, 2, \ldots, n\}$
Directed: $e_i = (v_i, v_{i+1})$ for all $i \in \{1, 2, \ldots, n\}$

In this case we say that p is a path from v_1 to v_{n+1}.

Definition 1.4.9 (Clique)**.** Given an undirected graph $G = (V, E)$, a clique is a subset of vertices $C \subseteq V$ such that $(u, v) \in E$ for all $u, v \in C$ with $u \neq v$. A clique C is said to be maximal if for each $u \in V \setminus C$, $C \cup v$ is not a clique. We further denote by $\mathcal{C}_G$ the set of cliques of G.

Definition 1.4.10 (Connected graph)**.** Given an undirected graph $G = (V, E)$, we say that G is connected respectively if for all $u, v \in V$ such that $u \neq v$ exists a path p from u to v.

Definition 1.4.11 (Separation)**.** Let $G = (V, E)$ be an undirected graph. Let $U, W, Z \subset V$ mutually disjoint subsets of V, we will say that Z separates U and W, in which case we will write $U \perp W | Z$ if there are no paths from u to w in $G \setminus Z$ for all $u \in U$ and $w \in W$.

Definition 1.4.12 (Cycle)**.** Given a graph $G = (V, E)$, a cycle c is a sequence of edges $\{e_1, e_2, \dots, e_n\}$ for witch exists a sequence of distinct vertices $\{v_1, v_2, \dots, v_n, v_1\}$ such that

Undirected: $e_i = \{v_i, v_{i+1}\}$ for all $i \in \{1, 2, \dots, n-1\}$ and $e_n = \{v_n, v_1\}$
Directed: $e_i = (v_i, v_{i+1})$ for all $i \in \{1, 2, \dots, n-1\}$ and $e_n = (v_n, v_1)$.

Definition 1.4.13 (Tree)**.** A tree is an undirected graph $G = (V, E)$ which is connected and it has no cycles.

Proposition 1.4.1. Let $T = (V, E)$ be a tree and $u, v \in V$ then there is one and only one path in T from u to v.

Proof. A tree is by definition connected, so there is at least one path from u to v. Let us by contradiction that exist two distinct paths $p = \{e_1, e_2, \dots, e_n\}$ and $q = \{e'_1, e'_2, \dots, e'_m\}$ from u to v for which there exist respectively two sequences of vertices $\{v_1, v_2, \dots, v_{n+1}\}$ and $\{v'_1, v'_2, \dots, v'_{m+1}\}$ such that requirements in Definition 1.4.8 hold. Since the paths are distinct exists $e_i \notin q$. Now we can define $(j_1, j_2) := \max\limits_{l_1, l_2 \leq i} v_{l_1} = v'_{l_2}$ and $(k_1, k_2) := \min\limits_{l_1, l_2 \geq i+1} v_{l_1} = v'_{l_2}$. The definition is well posed since $v_1 = v'_1 = u$ and $v_{n+1} = v_{m+1} = v$. But then $\{e_{j_1}, \dots, e_{k_1}, e'_{j_2}, \dots, e'_{k_2}\}$ is a cycle of T and that is a contradiction since T is acyclic by definition. □

Definition 1.4.14 (Spanning tree)**.** Let $G = (V, E)$ be an undirected connected graph, a tree $T = (V_T, E_T)$ such that $V_T = V$ and $E_T \subseteq E$ is called a spanning tree of G.

Definition 1.4.15 (Minimum spanning tree)**.** Let $G = (V, E, \psi)$ be an undirected, connected and edge-weighted graph where $\psi : E \to \mathbb{R}$, a minimum spanning tree T is a spanning tree such that

$$T = \arg\min_{T'} \sum_{e \in E(T')} \psi(e).$$

1.5 Dimensionality reduction

In this section we describe some unsupervised learning methods widely used in literature for the visualization and processing of datasets of gene regulatory networks.

1. Given a random sample $\{X_1, X_2, \dots, X_n\}$ of elements of $\mathbb{R}^D$, we denote by $X \in \mathcal{M}_{D \times n}(\mathbb{R})$ the matrix obtained by stacking the elements of the random sample by columns, this matrix takes the name of data matrix.
2. Starting from a random sample of high-dimensional elements, the methods we introduce below aim to provide a representation $\{Y_1, Y_2, \dots, Y_n\} \in \mathbb{R}^d$ in higher dimension low, $d \ll D$ of the original random sample. Using the wording of the previous point, we refer to the transformed data matrix with the term Y.

3. The idea behind dimensionality reduction is not limited to reducing the size of the data space but at the same time seeks to preserve the information and useful characteristics of the original data, this concept however does not allow for a single interpretation, we will in fact see that will be formulated distinctly in the presented algorithms.

Definition 1.5.1 (Pseudoinverse)**.** Given $M \in \mathcal{M}_{n\times m}(\mathbb{R})$ the pseudo-inverse M^+ is defined as the matrix such that

$$MM^+M = M \qquad M^+MM^+ = M^+ \qquad (MM^+)^T = MM^+(M^+M) = M^+M.$$

Definition 1.5.2 (Covariance matrix)**.** Given a data matrix X, we define the covariance matrix as

$$\Sigma_X := \frac{1}{n}\sum_{i=1}^{n}(X_i - \boldsymbol{\mu}_X)(X_i - \boldsymbol{\mu}_X)^T \in \mathcal{M}_{D\times D}(\mathbb{R})$$

where μ_X is the vector of sample means

$$\boldsymbol{\mu}_X := \frac{1}{n}\sum_{i=1}^{n} X_i \in \mathbb{R}^D.$$

Remark 1.5.1. We note that the covariance matrix is symmetric and positive definite. Indeed

$$\begin{aligned}\Sigma_X^T &= \left(\frac{1}{n}\sum_{i=1}^{n}(X_i - \boldsymbol{\mu}_X)(X_i - \boldsymbol{\mu}_X)^T\right)^T = \frac{1}{n}\sum_{i=1}^{n}((X_i - \boldsymbol{\mu}_X)(X_i - \boldsymbol{\mu}_X)^T)^T \\ &= \frac{1}{n}\sum_{i=1}^{n}(X_i - \boldsymbol{\mu}_X)(X_i - \boldsymbol{\mu}_X)^T = \Sigma_X\end{aligned}$$

and for a generic $v \in \mathbb{R}^D$

$$\begin{aligned}v^t\Sigma_X^T v &= v^t\frac{1}{n}\sum_{i=1}^{n}(X_i - \boldsymbol{\mu}_X)(X_i - \boldsymbol{\mu}_X)^T v = \frac{1}{n}\sum_{i=1}^{n} v^t(X_i - \boldsymbol{\mu}_X)(X_i - \boldsymbol{\mu}_X)^T v \\ &= \frac{1}{n}\sum_{i=1}^{n}((X_i - \boldsymbol{\mu}_X)^T v)^T(X_i - \boldsymbol{\mu}_X)^T v = \frac{1}{n}\sum_{i=1}^{n}||(X_i - \boldsymbol{\mu}_X)^T v||_2^2 \geq 0.\end{aligned}$$

1.5.1 Principal component analysis

Principal component analysis, PCA for short, is a linear dimensionality reduction algorithm, i.e., it uses a linear transformation as a map to reduced dimensionality space.

Theorem 1.5.1 (Spectral theorem)**.** Let $M \in \mathcal{M}_{n,n}(\mathbb{R})$. Then M is a symmetrical matrix if and only if there exists an orthonormal basis of $\mathbb{R}^n$ of eigenvectors of M.

Definition 1.5.3 (Principal Components)**.** Let X be a data matrix and $d \leq D$. Principal components are defined as the set of row vectors of the matrix $A \in \mathcal{M}_{d\times D}(\mathbb{R})$ such that

1. $AA^T = I$
2. The transformed data $Y = AX$ are uncorrelated and have maximum variance

Remark 1.5.2. The covariance matrix of the transformed matrix Y turns out to be

$$\begin{aligned}\varSigma_Y &= \frac{1}{n}\sum_{i=1}^{n}(Y_i - \boldsymbol{\mu}_Y)(Y_i - \boldsymbol{\mu}_Y)^T = \frac{1}{n}\sum_{i=1}^{n}(AX_i - A\boldsymbol{\mu}_X)(AX_i - A\boldsymbol{\mu}_X)^T \\ &= \frac{1}{n}\sum_{i=1}^{n}A(X_i - \boldsymbol{\mu}_X)(X_i - \boldsymbol{\mu}_X)^T A^T = A\frac{1}{n}\sum_{i=1}^{n}(X_i - \boldsymbol{\mu}_X)(X_i - \boldsymbol{\mu}_X)^T A^T = A\varSigma_X A^T.\end{aligned}$$

Proposition 1.5.1. Let X be a data matrix. The principal components of X are the first d orthonormalised eigenvalues of $\varSigma_X$ ordered with respect to the corresponding eigenvalues taken in descending order.

Proof. Let us denote by $e_1, e_2, \dots, e_d$ the standard basis of $\mathbb{R}^d$ and by $a_i := A^T e_i$ the i-th row of A. We first develop the expression of the variance and covariance of the transformed data by Remark 1.5.2

$$\mathrm{Var}(Y_i) = e_i^T \varSigma_Y e_i = e_i^T A\varSigma_X A^T e_i = (A^T e_i)^T \varSigma_X (A^T e_i) = a_i^T \varSigma_X a_i$$

$$\mathrm{Cov}(Y_i, Y_j) = e_i^T \varSigma_Y e_j = e_i^T A\varSigma_X A^T e_j = (A^T e_i)^T \varSigma_X (A^T e_j) = a_i^T \varSigma_X a_j$$

noting also that $||a_i||_2^2 = ||A^T e_i||_2^2 = (A^T e_i)^T A^T e_i = e_i^T AA^T e_i = e_i^t e_i = 1$, the conditions required in the definition of the principal components are equivalent to the following constrained system

$$\begin{cases} a_i^T a_i = 1, \text{ for all } i \\ a_i = \underset{v\in\mathbb{R}^D}{\arg\max}(v^T \varSigma_X v), \text{ for all } i \\ a_i^T \varSigma_X a_j = 0, \text{ for all } i, j \end{cases} \rightarrow \begin{cases} a_i = \underset{||v||_2=1}{\arg\max}(v^T \varSigma_X v), \text{ for all } i \\ a_i^T \varSigma_X a_j = 0, \text{ for all } i, j \end{cases}$$

We then solve the maximisation problem by applying Lagrange's multiplier method

$$0 = \nabla(a_i^T \varSigma_X a_i + \lambda(1 - a_i^T a_i)) = \begin{pmatrix} 2\varSigma_X a_i - 2\lambda a_i \\ 1 - a_i^T a_i \end{pmatrix}$$

which leads us to the following system

$$\begin{cases} \varSigma_X a_i = \lambda a_i, \text{ for all } i \\ a_i^T a_i = 1, \text{ for all } i \\ a_i^T \varSigma_X a_j = 0, \text{ for all } i, j \end{cases}$$

which admits as solutions the first d orthonormalised eigenvectors of $\varSigma_X$ ordered with respect to the eigenvalues in decreasing order, whose existence is guaranteed by Theorem 1.5.1 due to the fact that as shown in Remark 1.5.1 the matrix $\varSigma_X$ is symmetrical and positively defined. ◻

Algorithm 1.1 PCA

```
Require: $X \in \mathcal{M}_{D,n}(\mathbb{R})$ data matrix in dimension D, $d$ desired low dimension value
Ensure: $Y \in \mathcal{M}_{d,n}(\mathbb{R})$ data matrix in dimension d
1: $\Sigma_X \leftarrow$ calculate covariance matrix from $X$          ▷ Definition 1.5.2
2: $A \leftarrow$ calculate orthonormal basis of eigenvectors of $\Sigma_X$
3: $\bar{A} \leftarrow$ matrix of the first d rows of $A$ taken in decreasing order with respect to the eigenvalues
4: $Y \leftarrow \bar{A}X$
5: return $Y$
```

1.5.2 t-Distributed Stochastic Neighbor Embedding

The t-Distributed Stochastic Neighbour Embedding, t-SNE for short, is a dimension reduction algorithm introduced in van der Maaten et al. [29]. The purpose of this algorithm is to provide a two- or three-dimensional visualisation of higher-dimensional data. The algorithm is stochastic, so two separate runs on the same set of data can lead to different results, but unlike the PCA algorithm, it does not assume linearity of the transformation. The idea behind the algorithm is to endow the high-dimensional and low-dimensional points respectively with two distinct probability distributions and then perform a minimisation of the informational divergence of the two by moving the points in low-dimensional space using gradient descent.

Definition 1.5.4 (Probability in high dimension)**.** Let $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ be a set of d-dimensional data for some $d \in \mathbb{N}$. We define

$$p_{j|i} := \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|_2^2 / 2\sigma_i^2)}$$

This measure represents the probability with which the element $\mathbf{x}_i$ would take the element $\mathbf{x}_j$ as its neighbour if the neighbours were chosen proportional to the probability density of a Gaussian centred in $\mathbf{x}_i$ and of variance σ_i. We then define the following probability distribution

$$p_{ij} := \frac{p_{j|i} + p_{i|j}}{2n}.$$

Definition 1.5.5 (Probability in low dimension)**.** Let $\{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n\}$ be a set of d-dimensional data for some $d \in \mathbb{N}$. Let us define the following probability distribution

$$q_{ij} := \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|_2^2)^{-1}}{\sum_{k \neq i} (1 + \|\mathbf{y}_k - \mathbf{y}_i\|_2^2)^{-1}}.$$

Definition 1.5.6 (Perplexity)**.** Let $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ be a set of d-dimensional data for some $d \in \mathbb{N}$. Perplexity is defined as

$$p := 2^{H(p_{j|i})}$$

where $H(p_{j|i})$ is the entropy of the conditional distribution

$$H(p_{j|i}) = -\sum_j p_{j|i} \log_2 p_{j|i}.$$

We note that, as suggested by the authors of the article[7] this quantity should be interpreted as a measure of the effective number of neighbours.

[7] Page 2582 of der Maaten et al. [29]

Given a data matrix X, the goal of the t-SNE algorithm is to produce a data matrix Y in dimension 2 or 3 in such a way as to preserve the probability distribution defined on the starting set. To this end, the t-SNE algorithm first calculate the values of σ_i by performing a binary search in order to find those values that reach a predefined perplexity value p_0, then minimises the informational divergence, acting as the distance between distributions, between the probabilities $P = p_{ij}$ computed on the data $\mathbf{x}_i$ and $Q = q_{ij}$ computed on the data Y_i, in formulae

$$\hat{y} = \arg\min_y D(P||Q).$$

Since the informational divergence is convex in its arguments, as shown in Proposition 1.3.2, the gradient descent is used for minimisation.

Algorithm 1.2 tSNE

```
Require: $X \in \mathcal{M}_{D,n}(\mathbb{R})$ data matrix in dimension D, $T$ maximum number of iterations, $\eta$ learning rate parameter, $p_0$ perplexity
Ensure: $Y \in \mathcal{M}_{d,n}(\mathbb{R})$ data matrix in dimension d
1: $P \leftarrow$ high-dimensional probability matrix with perplexity $\approx p_0$        ▷ Definition 1.5.4
2: $Y^{(0)} \leftarrow N(0, \sigma^2)$        ▷ sample initial positions in low dimension
3: for $t \leftarrow 1$ to $T$ do
4:     $Q \leftarrow$ evaluate low-dimensional probability matrix from $Y^{(t-1)}$        ▷ Definition 1.5.5
5:     $\nabla D \leftarrow$ evaluate gradient        ▷ Proposition 1.5.2
6:     $Y^{(t)} \leftarrow Y^{(t-1)} + \eta\nabla D$        ▷ gradient descent
7: end for
8: return $Y^{(t)}$
```

Proposition 1.5.2. The gradient elements of the informational divergence used in the t-SNE algorithm have the following analytical form

$$\frac{\partial D(P \parallel Q)}{\partial y_i} = 4\sum_{j\neq i}(p_{ij} - q_{ij})(1 + ||y_i - y_j||_2^2)^{-1}(y_i - y_j).$$

Proof. To simplify the following steps, we introduce the following notations

$$d_{ij} := \|y_i - y_j\|_2 \qquad Z := \sum_{k\neq i}(1 + d_{ki}^2)^{-1} \qquad D := D(P||Q)$$

we can then proceed with the calculation of the partial derivative. We first note that as y_i varies within D the only ones to vary are d_{ij} and d_{ji}, so:

$$\frac{\partial D}{\partial y_i} = \frac{\partial D}{\partial d_{ij}}\frac{\partial d_{ij}}{\partial y_i} + \frac{\partial D}{\partial d_{ji}}\frac{\partial d_{ji}}{\partial y_i} = \left(\frac{\partial D}{\partial d_{ij}} + \frac{\partial D}{\partial d_{ji}}\right)(y_i - y_j).$$

Step 2: only the distribution Q depends on d_{ij} $(*)$, therefore

$$\frac{\partial D}{\partial d_{ij}} \overset{(*)}{=} -\sum_{k\neq l}^{n} p_{kl}\frac{\partial}{\partial d_{ij}}\log q_{kl} = -\sum_{k\neq l}^{n} p_{kl}\left(\frac{\partial}{\partial d_{ij}}\log(1 + d_{kl}^2)^{-1} - \frac{\partial}{\partial d_{ij}}\log Z\right).$$

Step 3

$$\frac{\partial}{\partial d_{ij}}\log(1 + d_{ij}^2)^{-1} = -2d_{ij}(1 + d_{ij}^2)^{-2}(1 + d_{ij}^2) = -2d_{ij}(1 + d_{ij}^2)^{-1}.$$

Step 4: by the definition of q_{ij} $(**)$

$$\frac{\partial}{\partial d_{ij}} \log Z = \frac{1}{Z}\frac{\partial Z}{\partial d_{ij}} = \frac{1}{Z}\frac{\partial}{\partial d_{ij}} \sum_{k \neq l} (1+d_{kl}^2)^{-1} = \frac{-2d_{ij}(1+d_{ij}^2)^{-2}}{Z} \overset{(**)}{=} -2d_{ij}q_{ij}(1+d_{ij}^2)^{-1}.$$

Step 5: Substituting step 3 and 4 into step 2 we obtain

$$\frac{\partial D}{\partial d_{ij}} = 2d_{ij}p_{ij}(1+d_{ij}^2)^{-1} - 2d_{ij}q_{ij}(1+d_{ij}^2)^{-1}\sum_{k \neq l}^{n} p_{kl} = 2d_{ij}(p_{ij}-q_{ij})(1+d_{ij}^2)^{-1}.$$

Finally, by substituting step 5 into step 1, by symmetry of d_{ij}, q_{ij} and p_{ij}, we obtain the thesis

$$\frac{\partial D}{\partial y_i} = \sum_{j=1}^{n} (4d_{ij}(p_{ij}-q_{ij})(1+d_{ij}^2)^{-1}(y_i - y_j)).$$

□

2 Gene regulatory networks

In biology, the term gene refers to a specific region of DNA containing the information necessary for the expression of a protein. The biological process in which genes are involved in order to synthesize a protein is called *gene regulation*, simplifying we can divide this process into two parts: transcription and translation. During transcription, genes are used to produce strings of RNA which are subsequently transformed into strings of mRNA (messenger RNA), these strings are subsequently used during the translation phase for protein synthesis. We say that a gene is expressed if the information it contains is transformed into a product by means of the process of gene expression.

2.1 Gene regulation process

The gene regulation process consists of all the sub-processes that take place in the cell for the regulation of gene expression. Two cells containing the same genome may differ both functionally and morphologically due to the fact that genes are expressed differently in each of the two cells. In biology this mechanism is called differentiation. Fundamental proteins for the gene regulation process are transcription factors, TF for short, these particular proteins possess the ability to bind to specific regions of DNA and regulate the transcription rate of a target gene. In particular, a gene which codes for a transcription factor is called *activator*, respectively *inhibitor*, if amplifies, respectively inhibits, the transcription of the target gene. In general a gene that directly regulates another gene is called *regulator*. The function of transcription factors is therefore to regulate the process of gene expression within the cell so that the products of the target genes are present in the quantities and at the times necessary for carrying out the activities of the cell. When multiple transcription factors give rise to complex regulatory systems capable of carrying out specific tasks within the cell, we speak of gene regulatory networks.

2.2 Gene regulatory networks

In order to study gene regulatory networks, different mathemaitcal models have been identified over time, each of which describes in its own way, with different mathematical tools, the behavior of a target gene based on the activity of his regulators. Each of these models focuses on a different aspect of regulatory process, making possible simplifications in the description of the elements of the network or making more or less restrictive assumptions on the biological process itself. The main difference between the existing models lies in the choice of the mathematical representation of the elements of the network: if the interest in the study of the network is focused on the macro-state of the elements, one could choose to represent them using boolean variables that simply reflect the state of the node (transcribed/not transcribed for genes or present/not present for mRNA strands); otherwise, by relaxing this representation, variables can be allowed to take on a value in a finite set of states (such as 'absent'/'low' /'medium'/'high' in reference to the concentration of proteins or mRNA); it is also possible to study the network from a probabilistic point of view, representing the network elements as random variables whose distribution represents the gene expression of a population of cells; finally, it is possible to represent the elements of the network through real-valued continuous variables, which specify the instant-by-instant value of gene expression. Further assumptions are made on the nature of the data made available to the model, which can be considered free or affected by noise, respectively deterministic and stochastic models. Another distinguishing feature is the time dependence or independence of the variables, respectively dynamic and static model. However, in the formulation of each model, the ability to distinguish between direct and indirect regulations, i.e. mediated or not by at least one other network element, has particular relevance. Let us now introduce some notations[1] that we will use in the rest of the thesis unless otherwise specified.

gene expression value: by gene expression value we mean a non-negative real numerical value directly proportional to the presence of the mRNA strands of a transcribed gene
regulator: a gene r is called regulator of a gene g if it's expression produces transcription factors that directly regulate the expression of g
target: a gene g is called target of a gene r if r is a regulator of g
activator: a gene g is called activator if it is a regulator that increases transcription of the target gene
repressor: a gene g is called repressor if it is a regulator that decreases transcription of the target gene

2.2.1 Labelled graph representation

One of the simplest ways of representing a gene regulatory network is by means of a directed and edge-labeled graph $G = (V, E)$ where the set of vertices is constituted by the genes of the network and the set of edges reflects the regulatory relationships between the genes. In particular $(i, j) \in E$ if and only if gene i is a regulator of gene j. An edge (g, h) is labeled with $+$ (graphically with the arrow $\vdash$) if g is an activator of h, it is instead labeled with $-$ (graphically with the arrow $\leftarrow$) if it's a repressor of h. the topology of the graph therefore contains a generic description of the regulatory network. An example of such a representation is shown in Figure 2.1. Arc orientation and/or edge labeling requests can be omitted depending on the need, many algorithms present in the literature use representations that do not use either of these two features. Although this model is simple, it does not provide sufficient information to describe the time evolution of the state of genes or their gene expression. Subsequently, representations are introduced which make more specific assumptions about the nature of the genes of the network in order to provide the possibility of representing this evolution.

[1] These definitions are used for the sole purpose of simplifying the language of the thesis, and therefore have no scientific value

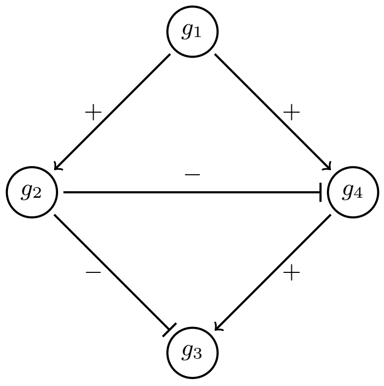


Fig. 2.1: Directed, labeled graph representation of a gene regulatory network involving four genes. From the representation we can deduce that g_1 acts as an activator against the g_2 and g_4 genes and that g_2 acts as a repressor against the g_3 and g_4 genes

.

2.2.2 Boolean network representation

The representation through boolean networks provides a qualitative description of the dynamics of the regulatory network, describing the macro-state of the elements of the network and the dynamics itself. The elements of the network, that we denote $x_1(t), x_2(t), \ldots, x_n(t)$, are modeled as time-dependent boolean variables, where 0/1 corresponds respectively to non-expressed/expressed gene or absent/present gene product inside the cell. The regulations between the elements of the network is therefore naturally represented by a boolean network in which the value at the next time of each variable representing a gene is determined solely by the value at the previous time of the variables representing the gene's regulators. In this model the dynamic of the network admits an interesting natural representation which constitutes a useful tool for studying the functional states of the cell.

Definition 2.2.1 (Boolean network)**.** An n-variable Boolean network is a tuple $f = (f_1, f_2, \ldots, f_n)$ of boolean functions in n variables, that is, $f_i : \{0,1\}^n \to \{0,1\}$ for each $i \in \{1, \ldots, n\}$.

A Boolean network can be viewed as a discrete dynamical system. In particular, given a boolean network f of n variables we can define the following dynamical system

$$\begin{cases} x_1(t+1) = f_1(x_1(t), x_2(t), \ldots, x_n(t)) \\ \cdots \\ x_n(t+1) = f_n(x_1(t), x_2(t), \ldots, x_n(t)) \end{cases}$$

where $x_i(t)$ represents the value assumed by the i-th boolean variable at time t. It is therefore straightforward to define the concept of attractor of the system.

Definition 2.2.2 (Attractors and basins)**.** Let $f = (f_1, \ldots, f_n)$ be a boolean network in n variables

1. A sequence $\mathbf{x}_1, \ldots, \mathbf{x}_l \in \{0,1\}^n$ of states such that
   $\mathbf{x}_i \neq \mathbf{x}_j$ for each $i \neq j$
   $f(\mathbf{x}_i) = \mathbf{x}_{i+1}$ for each $i \in \{1, \ldots, l-1\}$
   $f(\mathbf{x}_l) = \mathbf{x}_1$
   is called the attractor of the boolean network f.

2. An attractor $\mathbf{x}_1, \dots, \mathbf{x}_l$ is called steady state if $l = 1$, that is, if the sequence is composed of only one element
3. Given an attractor $X = (\mathbf{x}_1, \dots, \mathbf{x}_l)$, indicating with $f^{(m)}$ the composition of f with itself m times, the following set is called the basin of attractor X

$$\mathcal{B}(X) := \{\mathbf{x} \in \{0,1\}^n : f^{(m)}(\mathbf{x}) \in X, \exists m \in \mathbb{N}\}.$$

Definition 2.2.3 (State transition graph)**.** Given a genetic regulatory network containing n elements and represented by a boolean network whose transition function we call f, the state transition graph associated to the network is defined as the directed graph such that

1. $V = \{0,1\}^n$. The set of possible states of the network given by all possible n-uples of boolean values constitutes the set of vertices of the graph.
2. $E = \{(u,v) \in V \times V : f(u) = v\}$. The set of edges of the graph is given by the set of oriented edges whose endpoints represent consequential network states.

An example of such a representation of a network with three genes follows

$$\begin{cases} x_1(t+1) = x_1(t) \\ x_2(t+1) = \neg x_1(t) \\ x_3(t+1) = x_1(t) \wedge x_2(t) \end{cases}$$

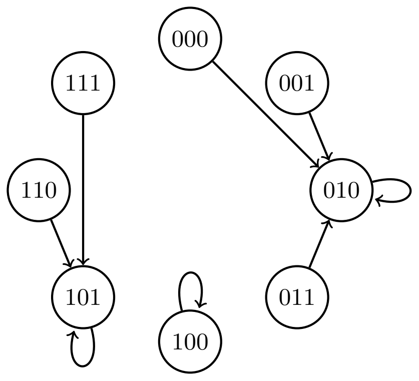


Fig. 2.2: State transition graph associated with the gene regulatory network previously reported. Where the three loops of the graph correspond to three steady states of the network.

It is interesting to note that the attractors of the boolean network representing the network are in one-to-one correspondence with the cycles of the state transition graph. In particular, given an attractor $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_l$ we can naturally associate to it the cycle $(\mathbf{x}_1, \mathbf{x}_2), \dots, (\mathbf{x}_{l-1}, \mathbf{x}_l), (\mathbf{x}_l, \mathbf{x}_1)$. Every other vertex of the graph are called transient states and necessarily belong to the basin of some attractor of the network. From a biological point of view, the attractors of the network can be considered as functional states which the cell reaches during their life cycle in response to external stimuli. Therefore the study of the dynamics of the boolean network (respectively, the topology of the state transition graph) provides a representation of the differentiation process a cell undergoes.

2.2.3 Probabilistic representation

The probabilistic representation of a gene regulatory network consists in a graph G whose vertices are random variables $\{X_1, \dots, X_n\}$ associated with the network elements and whose values represent the gene expression of the element with which they are associated. The regulatory relationships between elements are instead represented by a set of conditions on the probability distribution function of the network variables, which must reflect the graph topology given by the set of edges E. In literature, there are multiple ways of imposing such conditions, for the purpose of this thesis we are interested in Bayesian and Markov network models, which definitions are given below.

Definition 2.2.4 (Parents and non-descendants)**.** Given a direct acyclic graph $G = (V, E)$ we define the set of parents and non-descendants respectively as

$$\mathrm{pa}(u) := \{v \in V : (v, u) \in E\} \qquad\qquad \mathrm{nd}(u) := \{v \in V : (u, v) \notin E\}$$

Since G is acyclic it follows that $(u, v) \in E \Leftrightarrow (v, u) \notin E$ therefore $\mathrm{pa}(u) \subseteq \mathrm{nd}(u)$ for all $u \in V$.

Definition 2.2.5 (Bayesian network)**.** A Bayesian network is a pair (G, p) where G is a directed acyclic graph whose set of vertices $V = \{X_1, X_2, \ldots, X_n\}$ is a set of random variables with a joint distribution p such that for each $X_i \in V$ it is true that

$$p(\mathbf{x}_i|\mathrm{nd}(\mathbf{x}_i)) = p(\mathbf{x}_i|\mathrm{pa}(\mathbf{x}_i))$$

The joint distribution thus takes the following form

$$p(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) = \prod_{i=1}^{n} p(\mathbf{x}_i|\mathrm{pa}(\mathbf{x}_i)).$$

Definition 2.2.6 (Conditional independence)**.** Given three sets of random variables X, Y and Z. We say that X and Y are conditionally independent given Z, in which case we would write $X \perp\!\!\!\perp Y|Z$, if

$$p(\mathbf{x}, \mathbf{y}|\mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{y}|\mathbf{z}), \text{ for all } \mathbf{x} \in \mathcal{X}, \mathbf{y} \in \mathcal{Y}, \mathbf{z} \in \mathcal{Z}.$$

Definition 2.2.7 (Markov network)**.** A Markov network is a pair (G, p) where G is an undirected graph whose set of vertices V is a set of random variables with a joint distribution p such that for each $X, Y, Z \subseteq V$

$$X \perp Y|Z \Rightarrow X \perp\!\!\!\perp Y|Z$$

where $X \perp Y|Z$ is the Definition 1.4.11.

Proposition 2.2.1. $X \perp\!\!\!\perp Y|Z$ if and only if exists f, g such that $p(\mathbf{x}, \mathbf{y}, \mathbf{z}) = f(\mathbf{x}, \mathbf{z})g(\mathbf{y}, \mathbf{z})$ for all $\mathbf{x} \in \mathcal{X}, \mathbf{y} \in \mathcal{Y}, \mathbf{z} \in \mathcal{Z}$.

Proof. Let us first prove $(\Rightarrow)$.

$$p(\mathbf{x}, \mathbf{y}, \mathbf{z}) = p(\mathbf{z})p(\mathbf{x}, \mathbf{y}|\mathbf{z}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z})p(\mathbf{y}|\mathbf{z})$$

the thesis follows by defining $f(\mathbf{x}, \mathbf{z}) := p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$ and $g(\mathbf{y}, \mathbf{z}) = p(\mathbf{y}|\mathbf{z})$. Let us now prove $(\Leftarrow)$. We first note that

$$p(\mathbf{z}) = \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{X} \times \mathcal{Y}} p(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{X} \times \mathcal{Y}} f(\mathbf{x}, \mathbf{z})g(\mathbf{y}, \mathbf{z}) = \sum_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}, \mathbf{z}) \sum_{\mathbf{y} \in \mathcal{Y}} g(\mathbf{y}, \mathbf{z})$$

then the following applies

$$p(\mathbf{x}, \mathbf{y}|\mathbf{z}) = \frac{p(\mathbf{x}, \mathbf{y}, \mathbf{z})}{p(\mathbf{z})} = \frac{f(\mathbf{x}, \mathbf{z})g(\mathbf{y}, \mathbf{z})}{\sum_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}, \mathbf{z}) \sum_{\mathbf{y} \in \mathcal{Y}} g(\mathbf{y}, \mathbf{z})}$$

$$p(\mathbf{x}|\mathbf{z}) = \frac{p(\mathbf{x},\mathbf{z})}{p(\mathbf{z})} = \frac{\sum_{y\in\mathcal{Y}} p(\mathbf{x},\mathbf{y},\mathbf{z})}{p(\mathbf{z})} = \frac{f(\mathbf{x},\mathbf{z})\sum_{\mathbf{y}\in\mathcal{Y}} g(\mathbf{y},\mathbf{z})}{\sum_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x},\mathbf{z})\sum_{\mathbf{y}\in\mathcal{Y}} g(\mathbf{y},\mathbf{z})} = \frac{f(\mathbf{x},\mathbf{z})}{\sum_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x},\mathbf{z})}$$

$$p(\mathbf{y}|\mathbf{z}) = \frac{p(\mathbf{y},\mathbf{z})}{p(\mathbf{z})} = \frac{\sum_{x\in\mathcal{X}} p(\mathbf{x},\mathbf{y},\mathbf{z})}{p(\mathbf{z})} = \frac{g(\mathbf{y},\mathbf{z})\sum_{x\in\mathcal{X}} f(\mathbf{x},\mathbf{z})}{\sum_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x},\mathbf{z})\sum_{\mathbf{y}\in\mathcal{Y}} g(\mathbf{y},\mathbf{z})} = \frac{g(\mathbf{y},\mathbf{z})}{\sum_{\mathbf{y}\in\mathcal{Y}} g(\mathbf{y},\mathbf{z})}$$

from which it immediately follows that $p(\mathbf{x},\mathbf{y}|\mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{y}|\mathbf{z})$. □

Definition 2.2.8 (Gibbs distribution)**.** Let $G = (V, E)$ be an undirected graph with $n := |V|$. A probability distribution function $p : \mathcal{X}^n \to \mathbb{R}$ over the set of vertices V of G is said to be a Gibbs distribution over G if it can be factorized as

$$p(\mathbf{x}) = \frac{1}{Z}\prod_{C\in\mathcal{C}_G} \phi_C(\mathbf{x}_C)$$

Where $\mathcal{C}_G$ is the set of cliques of G, $\phi_C : \mathcal{X}^{|C|} \to \mathbb{R}^+$ are called potential functions, $\mathbf{x}_C$ is the vector obtained from $\mathbf{x}$ by taking only the elements in C and

$$Z := \sum_{\mathbf{x}\in\mathcal{X}}\prod_{C\in\mathcal{C}_G} \phi(\mathbf{x}_C)$$

We now quote a theorem whose complete proof can be found in *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* [22] by Pearl and Judea.

Theorem 2.2.1 (Hammersley-Clifford theorem)**.** Let $G = (V, E)$ be an undirected connected graph and p a probability distribution function over the sets of vertices V of G, then (G, p) is a Markov network if and only if p is a Gibbs distribution over G.

Proof. We only prove $(\Leftarrow)$. Let X, Y, Z be mutually disjoint subsets of V such that $X \perp Y|Z$. Let us first define

1. $\tilde{X} := \{v \in V \setminus Z : \exists x \in X, \exists p_{xv} \text{ path from } x \text{ to } v, p_{xv} \cap Z = \emptyset\}$
2. $\tilde{Y} := \{v \in V \setminus Z : \exists y \in Y, \exists p_{yv} \text{ path from } y \text{ to } v, p_{yv} \cap Z = \emptyset\}$

We observe that obviously $X \subset \tilde{X}$ and $Y \subset \tilde{Y}$ furthermore, since G is connected $\tilde{X}\cup\tilde{Y}\cup Z = V$. We now show that $C \subset \tilde{X} \cup Z$ or $C \subset \tilde{Y} \cup Z$ for all cliques $C \in \mathcal{C}_G$. In fact, if $C \subset Z$, there is nothing to prove. Let us therefore assume that $C \cap Z = \emptyset$. If absurdly exists $\tilde{x} \in \tilde{X} \cap C$ and $\tilde{y} \in \tilde{Y} \cap C$, then the following holds

1. $\{\tilde{x}, \tilde{y}\} \in E$ since C is a clique and $\tilde{x}, \tilde{y} \in C$
2. exists $p_{\tilde{x},x}$ path from $\tilde{x}$ to x that not intersect Z, for some $x \in X$, by definition of $\tilde{X}$
3. exists $p_{\tilde{y},y}$ path from $\tilde{y}$ to y that not intersect Z, for some $y \in Y$, by definition of $\tilde{Y}$

Therefore $p_{\tilde{x},x} \cup (\tilde{x}, \tilde{y}) \cup p_{\tilde{y},y}$ is a path from x to y that not intersect Z but this is a contradiction to the fact that $X \perp Y|Z$. We can therefore write that

$$p(\mathbf{x}) = \frac{1}{Z}\prod_{C\in\mathcal{C}_G} \phi_C(\mathbf{x}_C) = \frac{1}{Z}\prod_{C\subset Z\cup\tilde{X}} \phi_C(\mathbf{x}_C)\prod_{C\subset Z\cup\tilde{Y}, C\cap\tilde{Y}\neq\emptyset} \phi_C(\mathbf{x}_C) = f(\mathbf{x}_{\tilde{X}}, \mathbf{x}_Z)g(\mathbf{x}_{\tilde{Y}}, \mathbf{x}_Z)$$

Then it follows from Proposition 2.2.1 that

$$\tilde{X} \perp\!\!\!\perp \tilde{Y}|Z$$

But $X \subseteq \tilde{X}$ and $Y \subseteq \tilde{Y}$ thus

$$X \perp\!\!\!\perp Y|Z$$

□

2.2.4 ODE representation

The representation through ordinary differential equations, ODEs for short, provides a description of the gene regulatory networks of a quantitative type. In this formulation the mRNA and protein concentrations are represented as time-dependent real-valued variables $x_i : \mathbb{R}^+ \to \mathbb{R}^+$ whose variation over time is dictated by the regulatory relationships of the network which take the form of a system of ordinary differential equations. The dynamics of the gene regulatory network is therefore described by a system of differential equations in which the transcription rate of the product of each gene in the network is expressed as a function of the concentration of the products of the respective regulators and of any external products supplied to the cell. Denoting with $\mathbf{x}(t) = (x_1(t), x_2(t), \dots, x_n(t))$ the vector of the concentrations of the products of a set of genes of a regulatory network and denoting with $\mathbf{u}(t) = (u_1(t), u_2(t), \dots, u_m(t))$ the vector of concentrations of external products, the dynamics of the network has the following form[2]: for all $i \in \{1, 2, \dots, n\}$

$$\begin{cases} \frac{dx_i}{dt}(t) = f_i(\mathbf{x}(t), \mathbf{u}(t)) \\ x_i(t) \geq 0 \end{cases}$$

For a given set of functions $f_i : \mathbb{R}^{n+m} \to \mathbb{R}$ called regulation functions for which the necessary regularity for the description of each specific regulation network is assumed. In this type of representation of gene regulatory networks the object of interest is the expression rate of each gene, therefore the type of regulatory relationship $j \to i$ between two generic genes i and j is determined by the contribution that the expression rate of gene j makes to the expression rate of gene i, we note in particular that

$$\frac{d}{dt}\left(\frac{dx_i}{dt}\right) = \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}\frac{dx_j}{dt} + \sum_{k=1}^{m} \frac{\partial f_i}{\partial u_k}\frac{du_k}{dt}$$

That is, the partial derivative of the regulation function f_i with respect to the variable x_j determines the relation between the two expression rates, therefore we say that the gene j is an activator or respectively an inhibitor of the gene i if:

$$\frac{\partial f_i}{\partial x_j} > 0 \qquad\qquad \frac{\partial f_i}{\partial x_j} < 0$$

Generally this system of equations appears in less general forms such as the following: for all $i \in \{1, 2, \dots, n\}$

$$\begin{cases} \frac{dx_i}{dt}(t) = r_i(\mathbf{x}(t), \mathbf{u}(t)) - \gamma_i x_i(t) \\ x_i(t) \geq 0 \end{cases}$$

Where r_i represents the contribution of the regulators of the gene i and of the external products, $\gamma_i \in \mathbb{R}^+$ is called the degradation constant of the gene i and denotes the fact that in absence of contributions the concentration of the products of a gene decreases over time due to biological processes such as degradation and diffusion. In fact if there are no regulatory relationships, i.e. $r_i \equiv 0$, we obtain

$$\frac{dx_i}{dt} = -\gamma_i x_i$$

[2] Systems of order greater than one can be found in the literature, which are beyond the scope of this thesis and are therefore not reported

The solution of which, for each initial value x_0 compliant with the requirement of positivity of the variable, is well known

$$x_i(t) = x_0 e^{-\gamma_i t}$$

So $x_i(t) \to 0$ as $t \to +\infty$. In literature there are many regulation functions among which we report the following

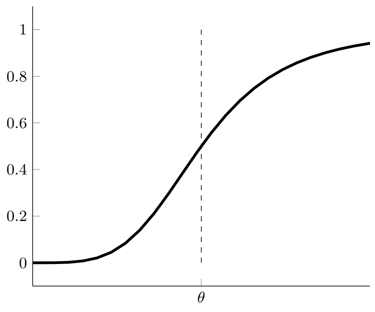


Fig. 2.3: Hill function

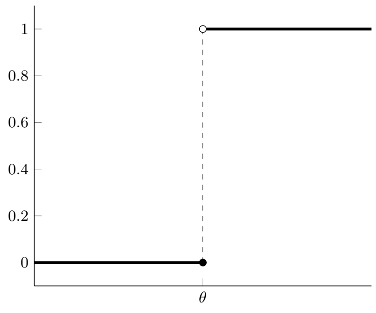


Fig. 2.4: Step function

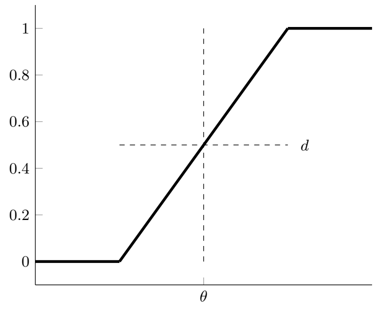


Fig. 2.5: Ramp function

$$r(x,\theta) = \frac{x^n}{x^n + \theta^n}$$

$$r(x,\theta) = \mathbb{I}_{(\theta,+\infty)}(x)$$

$$r(x,\theta,d) = \frac{1}{2} + \left|\frac{1}{4} + \frac{x+\theta}{2d}\right| - \left|\frac{1}{4} - \frac{x+\theta}{2d}\right|$$

An explicit example - the Lac operon

An example of gene regulatory process can be found in the genome of E.Coli, a bacterium that lives inside the intestines of warm-blooded animals like humans. It uses as principal source of energy glucose but since intestinal environment is constantly changing the bacterium is able to use other energy sources such as lactose. The bacterium possesses a process of gene regulation that allows it to change its metabolic activities depending on the environment. An operon is a unit of DNA containing a collection of genes which are expressed together or not at all. The process in question aims to regulate the gene expression of the Lac operon whose collection of genes, LacZ, LacY and LacA code for proteins necessary to break down lactose. This process is mediated by the LacI gene, which in absence of lactose is expressed and acts as a repressor inhibiting the transcription of Lac operon genes, in this case we say that the operon is repressed. When lactose is present in the cell it begins to be converted into allolactose which act as a repressor for the LacI gene which thus interrupts its role of repressor against the Lac operon, in this case we say that the operator is induced by lactose.

$$\begin{tikzcd}
 & \text{LacI} \arrow[dl, -|] & \\
\text{LacZ/Y/A} \arrow[d, -|] & & \text{Allolactose} \arrow[ul, -|] \\
\text{Lactose} \arrow[urr] & & \text{Source} \arrow[ll]
\end{tikzcd}$$

Fig. 2.6: Simplified gene regulatory network of the lac operon. Where $\to$ indicates an activation regulation and $\dashv$ indicates an inhibition regulation.

The process just mentioned is not the only one regulating the Lac operon, in fact, as previously mentioned, the bacterium tends to prefer glucose if it is present. The overall process is described by the following boolean newtork, whose exhaustive treatment is reported in Alan Veliz-Cuba and Brandilyn Stigler [30]. Each variable of this model represents a protein, a sugar or an mRNA strand, each of which can assume only one of the following two states: 0 to indicate its absence, 1 to indicate its presence.

Network element name	**Variable**	**Update function**
lac mRNA	M	$\neg R \wedge C$
lac permease	P	M
β-galactosidase	B	$\neg G_e$
catabolite activator protein	C	$L \wedge B$
repressor protein LacI	R	$\neg G_e \wedge P \wedge L_e$
lactose	L	M
allolactose	A	$\neg A \wedge \neg A_l$
low concentration of lactose	L_l	$A \vee L \vee L_l$
low concentration of allolactose	A_l	$\neg G_e \wedge (L \vee L_e)$
extracellular lactose	L_e	
extracellular glucose	G_e	

Table 2.1: Table of the gene regulatory network of the Lac operon in E.Coli.

G_e and L_e represent extracellular lactose and glucose respectively, and are considered free parameters of the model. With reference to article[3] we say that the operon is induced when $(M, P, B) = (1, 1, 1)$ and repressed when $(M, P, B) = (0, 0, 0)$. We observe that for each pair of values of external products the network possesses a unique steady state[4]: for $(G_e, L_e) \in \{(0, 0), (1, 0), (1, 1)\}$ the steady state corresponds to a state of repression of the Lac operon, when instead only lactose is supplied to the cells, the steady state correspond to a state of induction of the Lac operon.

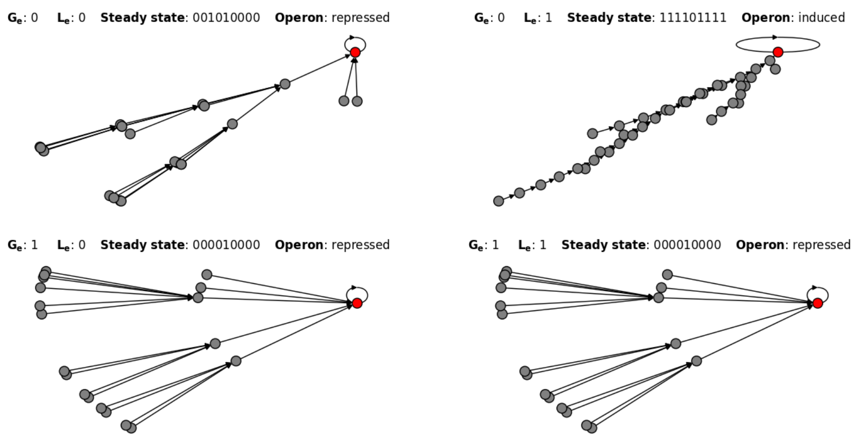

G_e: 0 L_e: 0 Steady state: 001010000 Operon: repressed
G_e: 0 L_e: 1 Steady state: 111101111 Operon: induced
G_e: 1 L_e: 0 Steady state: 000010000 Operon: repressed
G_e: 1 L_e: 1 Steady state: 000010000 Operon: repressed

Fig. 2.7: In the figure the state transition graph of the process from which zero in-degree vertices have been removed as external parameters vary. In red the steady state.

[3] Page 5 of Alan Veliz-Cuba and Brandilyn Stigler [30]
[4] Steady state calculated using Deep first search algorithm on the state transition graph of the boolean network

2.3 Gene regulatory network inference

The inference problem of a gene regulatory network consists in the reconstruction, by means of any representation, of the regulatory relationships existing between the elements of a regulatory network starting from experimental data containing the gene expression values of the latter . Due to the complex nature of the data taken into consideration, in formulating the problem it may happen that one has to choose to focus only on particular classes of elements of the network, from the most superficial level taking into consideration only the transcription factors involved in the process at a more detailed which also includes proteins and mRNA strands. The objective of such a process is therefore to estimate a value for each pair of elements of the network which serves as a measure of confidence of the effective presence of a regulatory relationship between them, given for example two genes of the network for which a regulator/target type relationship is assumed, a non-negative value is to be assigned to this pair, the higher the more plausible it is deemed that this relationship actually occurs inside the cell.

2.3.1 Data types

The types of data on which we draw attention in this thesis are a particular class of transcriptomic data called respectively microarray-data and RNA-Seq data, abbreviation for RNA sequencing data. They provide a measure of the quantity of transcript products of a given set of genes present in the cells of an organism. In each measurement, for each gene, the corresponding strands of RNA present in the samples are counted, the number of these strands represents an expression index of the corresponding gene: the more strands there are, the more plausible that the corresponding gene is currently activated, vice versa fewer strands are an indication of a potential inhibition process in progress. Depending on the case, these measurements can undergo a normalization process based on known a priori observations of a biological nature aimed at eliminating any bias due to external factors.

Microarray data: microarray data are older than RNA-Seq. These data provide a gene expression value for a specific set of genes as the sample varies, usually between samples the cells used differ in conditions such as the presence or absence of a specific disease. The gene expression value can be the signal intensity of the detection instrument or the quantity of gene products.

Bulk RNA-Seq data: The term Bulk RNA-Seq refers to those RNA-Seq data whose samples consist of a mixture of RNA strands from different sets of cells into which the population has been subdivided. What is obtained in this case is therefore an average value, per-sample, of the number of strands.

scRNA-Seq data: The term scRNA-Seq data, short for single cell RNA-Seq data, refers to those RNA-Seq data whose samples consist of all the RNA strands of the single cells of the population. Unlike bulk data, single cells allow us to appreciate the possible heterogeneity of the population, in fact single cells can present differences in the stages of development or in gene expression caused by various factors. The study of these differences makes it easier to identify gene expression patterns within the dataset that would otherwise not be observable. What is obtained in this case is therefore a per-cell and per-sample value of the number of strands.

From now on, unless otherwise specified, we will use the following matrix notation to refer to a microarray or to an RNA-Seq dataset: with the term $E_{i,j}$ we will indicate the gene expression value of gene i in sample j, in the case of a single cell dataset the j will therefore indicate the cell index. If the data depend on another parameter such as time, we will use the notation with superscript $E^k_{i,j}$ in which k acts as an index for the new parameter. An example of such a representation is shown in the following image

Gene \ Sample	s_1	s_2	...
g_1	120	142	176
g_2	34	52	76
g_3	0	13	5
g_4	75	63	57
...	...	...	...

2.3.2 Pseudotime

In this subsection we will give an intuitive idea of the concept of pseudotime introduced by Trapnell et al. In [28] for the analysis of an scRNA-Seq dataset. In this regard, we highlight the main mathematical tools used to define this quantity. The idea behind the concept of pseudotime is based on the observation of the fact that in the analysis of a single cell dataset it is not always convenient to choose to use a time scale for sorting the data, since in the same population of cells there could be subsets of cells belonging to different branches of the differentiation process and which therefore are not comparable if one looks only at the time: the data relating to two distinct cells of the dataset could present a large temporal difference, due for example to the natural slowness of the process of which they are part, although they belong to the same branch of the differentiation process. Among the known approaches for the definition of this quantity we highlight the Monocle package whose original version is based on the article mentioned above and whose latest version is updated to 2022. Within Monocle, for the definition of the concept of pseudotime the authors have chosen to assume that the differentiation process of the cells under examination can be represented by means of a tree, whose vertices represent the main functional states that the cell can assume during its life and whose edges highlight the possible paths the cell can take. Using this representation, given a vertex o of the graph as the origin of the differentiation process, for each functional state s of the connected component to which o belongs, it is possible to associate a distance considering the path connecting the vertices o and s, this distance is called pseudotime and represents a measure of the evolutionary state of a cell with respect to a branch of the differentiation process. Among the methods provided by Monocle for the calculation of pseudotime we present below the one based on information theory.

Data pre-processing The process starts from a scRNA-Seq dataset $E^0 \in \mathcal{M}_{n,c}(\mathbb{R})$ where $n \in \mathbb{N}$ represents the number of genes and $c \in \mathbb{N}$ the number of cells. During the first phase, the goal is to perform a first dimensionality reduction of the dataset in order to eliminate uninformative data by means of the PCA algorithm introduced in Section 1.5.1. We therefore obtain a new dataset $E^1 \in \mathcal{M}_{m,c}(\mathbb{R})$ with $m \ll n$.

Non-linear dimension reduction Starting from the dataset elaborated in the previous step E^1, monocle proceeds with a second dimensionality reduction aimed at bringing the dataset to a dimension of 2 or 3 to be able to display it. Monocle provides two different algorithms for this reduction among which t-SNE introduced in Section 1.5.2. At the end of the second step we therefore have a dataset $E^2 \in \mathcal{M}_{l,c}(\mathbb{R})$ with $l \in \{2, 3\}$.

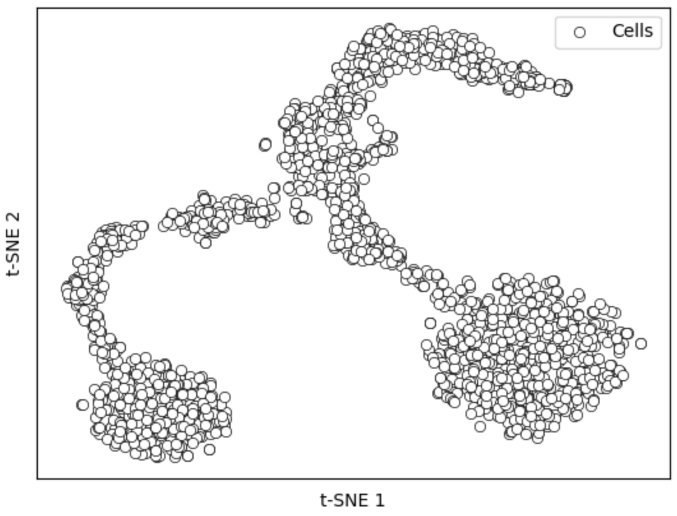
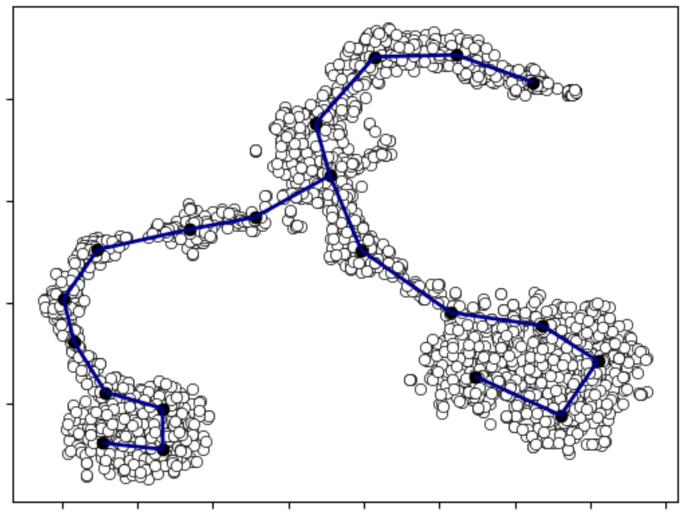


Fig. 2.8: tSNE algorithm applied to E.Coli dataset

Learn graph During the third phase, simplePPT[5] Algorithm is performed to search the tree $T = (V, E)$ that best fit the data according to the following criterion

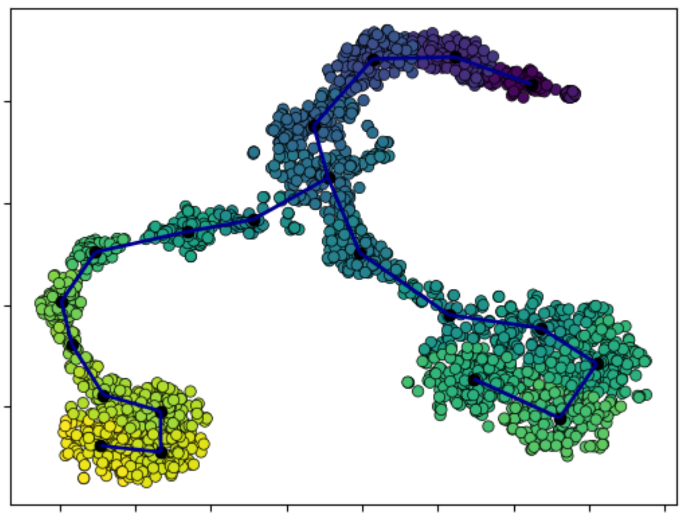
$$\text{minimize } \frac{1}{n}\sum_{i=1}^{n} \min_{\mathbf{z}\in V(T)} \|\mathbf{x}_i - \mathbf{z}\|_2^2 \text{ constrained to } \sum_{\{\mathbf{z},\mathbf{w}\}\in E(T)} \|\mathbf{z} - \mathbf{w}\|_2^2 \leq l$$

where $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\} \in \mathbb{R}^2$ are the points calculated with tSNE, $l \in \mathbb{R}^+$ is a parameter of the algorithm and the set of nodes of the tree is seen as a set of elements of $\mathbb{R}^2$. For For an in-depth discussion of the algorithm, we refer to the article by Qi Mao et al. [15]



[5] An efficient python implementation of simplePPT, used to produce the image in the figure, can be found at https://github.com/LouisFaure/simpleppt

Pseudo-time assignment Finally Monocle proceeds with the assignment of the pseudo-times. Once indicated a vertex as the origin, the value of the pseudotime of a cell, i.e. of a point $\mathbf{x}_i$ in the dataset, is then obtained by calculating the length of the shortest path between the projection of the cell on the tree and the origin. The length of the path is to be understood as the sum of the Euclidean distance between the extremes of each edge of the path.



Let us denote by $\{\mathbf{z}, \mathbf{w}\} \in E(T)$ the edge of the graph closest to $\mathbf{x}_i$. The projection $\pi(\mathbf{x}_i)$ of $\mathbf{x}_i$ onto the edge is defined as

$$\pi(\mathbf{x}_i) := \mathbf{w} + t(\mathbf{z} - \mathbf{w}) \qquad t := \begin{cases} \hat{t} & 0 \leq \hat{t} \leq 1 \\ 1 & \hat{t} > 1 \\ 0 & \hat{t} < 0 \end{cases} \qquad \hat{t} := \frac{(\mathbf{x}_i - \mathbf{z})^T(\mathbf{z} - \mathbf{w})}{\|\mathbf{z} - \mathbf{w}\|^2}$$

2.3.3 Workflow of GRN inference algorithms

Due to its complex nature caused by the huge variety of data types, the gene regulation network inference problem does not allow to establish with certainty which is the best setting for its formulation, however common strategies can be identified for specific types of data. In this thesis we have considered datasets of the RNA-Seq type, for which we introduce the workflow proposed in the project BEELINE by Pratapa et al. [23], which provides tools for the evaluation of the performance of algorithms for inference of gene regulatory networks from RNA-Seq data. The pattern identified for the formulation of the problem involves three fundamental steps.

Data processing

The first step consists of pre-processing the dataset, which is often reduced and normalised in order to make the next steps more functional. Normally, RNA-Seq datasets have a large number of genes, some of which, for various reasons are uninformative. To solve this problem, the size of the data is reduced by means of algorithms such as PCA; This step is particularly important if the regulatory network is modelled using a probabilistic represetnation, as in the case of algorithms based on information theory. The first step concludes by outputting a reduced matrix of the data $\hat{E}$ from the original matrix E.

Algorithm body

The second step consists in the application of the inference algorithm that will be required to provide a minimal set of output data for subsequent evaluation from the gene expression matrix $\hat{E}$ processed during the first step. In particular, it is required to generate a classification of the pairs of network elements in the form of a matrix of the type

$$R \in \mathcal{M}_{n,n}(\mathbb{R})$$

where, for a generic pair i, j the quantity $R_{i,j}$ stands for a confidence measure estimated by the algorithm regarding the actual presence of the corresponding adjustment ratio between the two elements. In other words, the higher the value $R_{i,j}$, the more plausible the algorithm must consider that there is an adjustment relationship between the i-second element and the j-second element of the network. Such a matrix will be symmetrical or not, depending on the algorithm's ability to infer the roles within the pair: regulator and target. If provided by the algorithm, it will also be required to provide a matrix

$$S \in \mathcal{M}_{n,n}(\mathbb{R})$$

where, for a generic pair i, j the quantity $S_{i,j}$ indicates the type of regulation: 0 for an inhibition and 1 for an activation. Example of output in table form:

Regulator i	Target j	confidence $R_{i,j}$	type $S_{i,j}$
g_5	g_4	2.3	1
g_3	g_4	1.2	1
g_6	g_1	0.8	0
...	...	...	...

Evaluation of the algorithm

The matrices produced in the previous step are then used for the evaluation of the algorithm, in particular they are used in order to formulate the inference problem as a binary classification and consequently use the measures typical of this class of problems. By varying a non-negative real parameter ε, which represents the threshold with which to discern between values that are actually reliable and those that are not. A classification is induced on the pairs of genes of the network starting from the associated confidence measures, setting equal to 1 the pairs whose confidence value is greater than or equal to ε and 0 the remaining ones. Once the classification has been carried out, we can proceed to measure the goodness of the algorithm as the threshold parameter ε changes.

Definition 2.3.1 (Confusion matrix)**.** Given a binary classification model, the confusion matrix is a table made up of four cells, each of which contains one of the possible predictions that such a model can perform:

1. True positives (TP): observations belonging to the positive class correctly classified by the model
2. True negatives (TN): observations belonging to the negative class correctly classified by the model
3. False positives (FP): observations belonging to the positive class misclassified by the model
4. False negatives (FN): observations belonging to the negative class misclassified by the model

If the binary classification model depends on a classification threshold ϵ, as in the case of interest, the quantities just introduced will also depend on this parameter, therefore we can make this dependence explicit by writing for example TP(ϵ).

Definition 2.3.2 (ROC curve)**.** The Receiver Operating Characteristic, ROC in short, is a graphical representation of the performance of a binary classification model as the threshold parameter varies. In particular it is a curve, ROC : $I \to [0,1] \times [0,1]$, with I interval of $\mathbb{R}_0^+$, defined by the following law

$$\mathrm{ROC}(\epsilon) = \left(\frac{\mathrm{TP}(\epsilon)}{\mathrm{TP}(\epsilon) + \mathrm{FN}(\epsilon)}, \frac{\mathrm{FP}(\epsilon)}{\mathrm{FP}(\epsilon) + \mathrm{TN}(\epsilon)} \right)$$

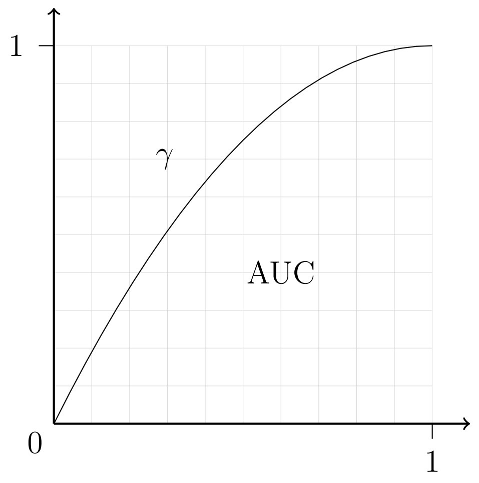
Where the first element is called *true positive rate* and the second *false positive rate* of the model respectively representing the fraction of elements correctly classified as positive (negative) among all those classified as positive (negative).

Definition 2.3.3 (PR curve)**.** The Precision Recall Characteristic, PR in short, is a graphical representation of the performance of a binary classification model as the threshold parameter varies. In particular it is a curve, PR : $I \to [0,1] \times [0,1]$, with I interval of $\mathbb{R}_0^+$, defined by the following law

$$\mathrm{PR}(\epsilon) = \left(\frac{\mathrm{TP}(\epsilon)}{\mathrm{TP}(\epsilon) + \mathrm{FN}(\epsilon)}, \frac{\mathrm{TP}(\epsilon)}{\mathrm{TP}(\epsilon) + \mathrm{FP}(\epsilon)} \right)$$

Where the first element is called *recall* and the second *precision* of the model respectively representing the fraction of elements correctly classified as positive among all those truly positive and among all those classified as positive.

Definition 2.3.4 (AUC measure)**.** Given a curve $\gamma : I \to [0,1] \times [0,1]$, with I interval of $\mathbb{R}_0^+$, evaluating a binary classification model, the Area Under the Curve, AUC for short, is defined as the value of the area between the curve and the axis of the abscissas. This measure varies in the interval $[0,1]$, where 1 indicates a perfect predictive ability of the model. If the curve is ROC, the associated measurement is denoted AUROC, if the curve is PR, it is denoted AUPRC.



3

GRN inference algorithms from data

In this chapter we introduce a selection of inference algorithms of gene regulatory networks. We chose the algorithms from literature mainly influenced by the desire to provide a sufficiently varied overview of the types of tools used, trying to cover the models and approaches introduced in the previous chapter while leaving the focus on algorithms based on information theory.

3.1 ARACNE

ARACNE is an inference algorithm for gene regulation networks based on information theory by Adam A. Margolin et al. [17]. It proposes an approach based on information theory.

Input: Microarray data
Output: Ranking of unsigned and unordered couples of genes
Model: Information measures, Probabilistic representation 2.2.3
Key concepts: Markov network 2.2.7, Gibbs distribution 2.2.8, Entropy estimators 1.3.3
Time complexity: $O(n^3 + n^2m^2)$, where n is the number of genes and m the number of samples

ARACNE uses a Markov network representation (G, p) 2.2.7 with the additional assumption that the maximal cliques of graph G are the sides of the graph itself and that the joint distribution of the gene expression of all genes in the network can be expressed as a Gibbs distribution 2.2.8. Such assumptions are well posed because of the theorem 2.2.1.

$$p(x_1, x_2, \ldots, x_n) = \frac{1}{Z} \exp\left(-\sum_{i=1}^{n} \phi_i(x_i) - \sum_{i,j=1}^{n} \phi_{i,j}(x_i, x_j) \right)$$

where

$$Z = \sum_{x_1, x_2, \ldots, x_n \in \mathcal{X}^n} \exp\left(-\sum_{i=1}^{n} \phi_i(x_i) - \sum_{i,j=1}^{n} \phi_{i,j}(x_i, x_j) \right)$$

ARACNE defines two genes X_i, X_j as non-interacting if and only if $\phi_{i,j} \equiv 0$. We note that this definition includes both genes that are actually independent ($\phi_{i,j} \equiv 0$ and $p(x,y) = p(x)p(y)$) and those that are not independent but interact by means of other genes in the network ($\phi_{i,j} \equiv 0$ and $p(x,y) \neq p(x)p(y)$). ARACNE's goal is therefore to try to identify which $\phi_{i,j}$ are non-zero, this is done in two steps. Recalling that mutual information between two random variables cancels if and only if they are independent

$$I(X;Y) = 0 \Leftrightarrow p(x,y) = p(x)p(y) \text{ for all } x \in \mathcal{X}, y \in \mathcal{Y}.$$

In the first step ARACNE aims to eliminate non-interacting and independent gene pairs, in this regard it removes genes couple for which the estimated mutual information is less than a given threshold I_0 previously calculated. The first step therefore fails to eliminate non-interacting gene pairs that are non-independent, i.e., those pairs that do not have a direct connection on the graph but interact via other genes and therefore still exhibit a sufficiently high mutual information value. Therefore during the second step an attempt is made to eliminate such couples. In particular, the proposed method involves the use of the data processing theorem 1.3.3.

Mutual information estimator

ARACNE estimates mutual information using the entropy estimator seen in Definition 1.3.17

$$\hat{h}_n(X) = -\frac{1}{n}\sum_{i=1}^{n} \ln \hat{f}_n(X_i) \tag{3.1}$$

Let $X_1, X_2, \dots, X_n$ and $Y_1, Y_2, \dots, Y_n$ be two random samples associated with two genes in the network. By the fact that $I(X;Y) = h(X) + h(Y) - h(X,Y)$ ARACNE uses the estimator obtained by approximating the entropies with Equation 3.1 using a multivariate Gaussian kernel K seen in Definition 1.2.16.

$$\hat{I}(X;Y) := \hat{h}_n(X) + \hat{h}_n(Y) - \hat{h}_n(X,Y) = \frac{1}{n}\sum_{i=1}^{n} \ln \frac{\hat{f}_n(X_i,Y_i)}{\hat{f}_n(X_i)\hat{f}_n(Y_i)} \tag{3.2}$$

Where

$$\hat{f}_n(x) := \frac{1}{nh}\sum_{j=1}^{n} K\left(\frac{1}{h}(x - X_j)\right) \qquad \hat{f}_n(x,y) := \frac{1}{nh^2}\sum_{i=1}^{n} K\left(\frac{x - X_j}{h}, \frac{y - Y_j}{h}\right) \tag{3.3}$$

3.1.1 Algorithm

First step

Definition 3.1.1 (p value)**.** Let $\{X_1, X_2, \dots, X_n\}$ be a random sample of variables with values in $\mathbb{R}_0^+$ and let $I_0 \in \mathbb{R}_0^+$ then the p-value of I_0 is defined as

$$p(I_0) := \mathbb{P}(\hat{I} \geq I_0 | I = 0)$$

for which the following estimator is defined

$$\hat{p}(I_0) := \frac{\sum_{i \neq j} \mathbb{1}_{[I_0,+\infty)}(\hat{I}(X_i, X_j))}{n(n-1)}$$

This parameter is used to estimate the value of I_0 to be used in the first stage of the algorithm. From a set of independent data, p is calculated as a function of I_0 and then I_0 is calculated from the p calculated on the set of input samples of the algorithm, reversing the relationship found previously. The authors randomly shuffled gene expression between the various samples in the microarray and evaluated the mutual information for all independent gene pairs. In particular, the estimation was performed on a sample of 10^5 gene pairs, in order to obtain reliable estimates of $I_0(p)$ up to p = 10^{-4}. The same simulation is performed here, with the same choice of bandwidth h using the python implementation of ARACNE reported in the last chapter and with the only difference being the number of gene pairs, which in this case is 10^4. We also note that contrary to the article we not use the estimate given for smaller values of p: $p(\hat{I} \geq I_0|I) \propto e^{-\alpha n I_0}$. Therefore starting from the change in concavity of the function in Figure 3.1, the value may no longer be reliable.

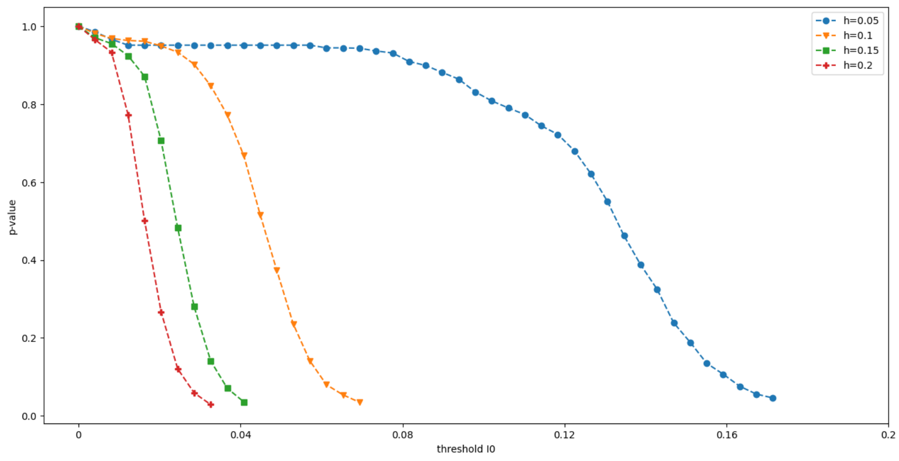


Fig. 3.1: Estimated p-value on the E.Coli dataset avaiable at https://github.com/Tchanders/network_inference_tutorials/tree/master/goldstandards

Second step

Let X, Y and Z be three genes of the network, such that Y belongs to every path from X to Z, i.e., such that X and Z interact only through Y. From the definition of a Markov network follows that

$$p(x, z|y) = p(x|y)p(z|y) \text{ for all } x \in \mathcal{X}, y \in \mathcal{Y}, z \in \mathcal{Z}$$

that is, $X \to Y \to Z$. We are therefore under the assumptions of the data processing Theorem 1.3.3 for which it holds that:

$$I(X;Z) \leq \min\left(I(X;Y), I(Y;Z)\right) \tag{3.4}$$

therefore the couple of genes to be removed has less or equal mutual information than the other two couples. During this step, for each possible triple of genes, the edge whose end points have the lowest value of mutual information is then removed. We note that without further assumptions it is not always possible to require the condition for which the strict inequality applies in 3.4 therefore it is not always clear which couple to remove. Despite this, according to the authors the "strictly less than" version of the data processing theorem is relevant whenever a system is lossy. i.e. whenever information is lost along each path. Biological systems are, by definition, lossy.

Theorem 3.1.1. If the mutual information between each pair of genes can be estimated without any error, the underlying regulatory network is a tree, it possesses exclusively pairwise interactions and the data processing theorem can be applied with the strict inequality, then ARACNE will exactly reconstruct the gene regulatory network.

Proof. Let's (T, p) be a Markov network representation of the network such that $T = (V, E)$ is a tree. Since T is a tree, by Proposition 1.4.1 for every $u, v \in V$ exists a unique path in T from u to v, whose associated set of vertices we denote by $p_{u,v}$. If there is at least one other vertex $z \in p_{u,v}$, the algorithm should remove the edge $\{u, v\}$ since $\{u, v\} \notin E$. In that case $u \perp v|z$ then by definition of Markov network holds that $u \to z \to v$ therefore by hypothesis we can apply the data processing theorem with the strict inequality:

$$I(u, v) < \min\left(I(u, z), I(z, v)\right)$$

therefore when the algorithm will consider the triple (u, v, z) it will correctly eliminate the edge (u, v). If instead the path $p_{u,v}$ consists only of the vertices u and v then the edge (u, v) should be kept by the algorithm. Considering any vertex $z \in V \setminus \{u, v\}$ there exists two unique paths $p_{u,z}$ and $p_{v,z}$, in particular it must hold that $v \in p_{u,z}$ or $u \in p_{v,z}$ since otherwise the tree would contain the cycle $p_{u,z} \cup p_{v,z} \cup p_{u,v}$ but this is not possible since T is a tree. Without loss of generality we can therefore assert that $v \in p_{u,z}$ which implies $u \to v \to z$ since $u \perp z|v$. Therefore by hypothesis we can apply by the data processing theorem with the strict inequality

$$I(u, z) < \min\left(I(u, v), I(v, z)\right)$$

thus the edge (u, v) will not be deleted when the algorithm will consider the triple (u, v, z). □

Algorithm 3.1 ARACNE

```
Require: $E$ network gene expression matrix, $I_0$ threshold value for mutual information
Ensure: $R$ matrix of regulatory relationships between gene pairs
 1: for $i = 1$ to n do
 2:     for $j = 1$ to n do
 3:         $\hat{f}(X_i, X_j) \leftarrow$ estimate density from $E$                ▷ Formula 3.3
 4:         $I_{i,j} \leftarrow$ estimate mutual information from $\hat{f}$        ▷ Formula 3.2
 5:         if $I_{i,j} \geq I_0$ then
 6:             $R_{i,j} \leftarrow 1$
 7:         else
 8:             $R_{i,j} \leftarrow 0$
 9:         end if
10:     end for
11: end for
12: for $i = 1$ to n do
13:     for $j = i + 1$ to n do
14:         for $k = j + 1$ to n do
15:             if $R_{i,j}, R_{i,k}, R_{j,k} = 1$ then
16:                 if $I_{i,j} < \min(I_{i,k}, I_{j,k})$ then     ▷ Apply data processing theorem 3.4
17:                     $R_{i,j} \leftarrow 0$
18:                 else if $I_{i,k} < \min(I_{i,j}, I_{j,k})$ then
19:                     $R_{i,k} \leftarrow 0$
20:                 else if $I_{j,k} < \min(I_{i,j}, I_{i,k})$ then
21:                     $R_{j,k} \leftarrow 0$
22:                 end if
23:             end if
24:         end for
25:     end for
26: end for
27: return $R$
```

3.2 PIDC

PIDC is an inference algorithm for gene regulatory networks based on information theory by Chan et al. [6]. This algorithm is based on information measures introduced in the partial information decomposition Section 1.3.2. The algorithm uses these measures to analyze the relationships between gene triples in the network in order to subsequently reconstruct the entire network. The use of such measures makes it necessary to estimate the joint distribution of two random variables, which is made easier by the fact that scRna-Seq datasets have high quantities of measurements.

Input: scRNa-Seq data
Output: Ranking of unsigned and unordered couples of genes
Model: Information measures, Probabilistic representation 2.2.3
Key concepts: Partial information decomposition 1.3.2, Entropy estimators 1.3.3
Time complexity: $O(n^3)$ where n is the number of genes

3.2.1 Algorithm

The basic idea of the algorithm is to define a measure that reflects the confidence in the presence of a given regulation, more precisely we want to associate each pair of genes X and Y with a non-negative value, creating a ranking of pairs in the network such that the higher up in the ranking a gene pair is, the more we are confident that there is a regulatory relationship between the pair. This algorithm does not provide information regarding the role of genes in regulation, in fact informations regarding who the regulatory gene is and whether it acts as an activator or inhibitor are not provided. The authors of the article first showed that by simulating networks of three genes, a particular pattern emerges for those networks with only one regulatory edge. For these networks in fact, the calculation of the partial information decomposition shows a peak in the value of unique information between connected variables that exceeds the value of redundant information. For all other triples types, on the other hand, it is seen that redundancy exceeds in value both unique informations. To test whether this pattern was respected for triples of genes belonging to larger networks the authors performed the calculation of the partial information decomposition on five different datasets of larger size. In this thesis, we repeated the same simulation on the same datasets by means of a code we implemented in Python. Within these five networks, we can find the following six types of triples.

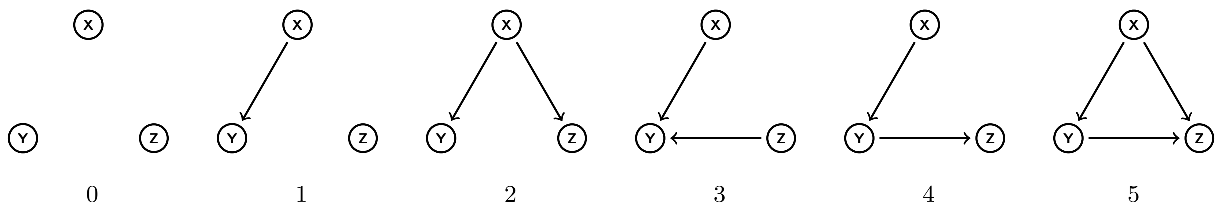


Fig. 3.2: Shown are the six configurations of oriented arcs contained within the datasets considered

For each of the networks in the article, the partial information decomposition was calculated for each triple of genes. The results were then divided and averaged over the previously shown classes, we can see that the pattern described above occurs again: the unique information of the connected genes is higher than the redundant information only in the case where the triple possesses a single directed edge.

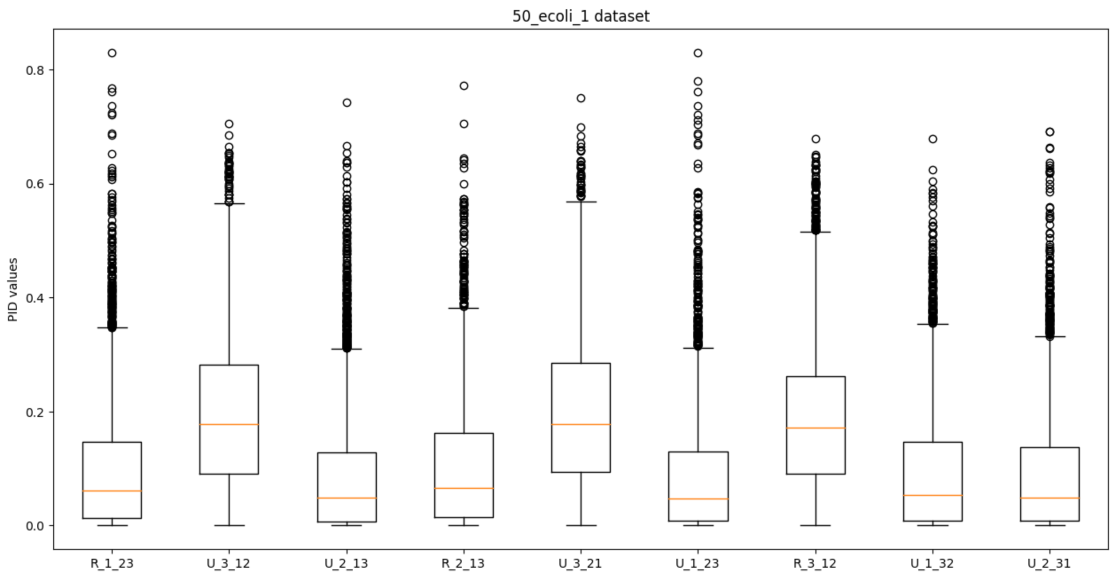


Fig. 3.3: PID values calculated for the triples of class 1 elements in the 50_ecoli_1 dataset. Triples are ordered in such a way that in the last three values the target is the unconnected gene.

Let us now consider three discrete random variables representing the gene expression of three distinct genes of a gene regulatory network X, Y and Z. We rewrite for these three variables the Definition 1.3.10 of unique information

$$\text{Unique}_Z(X;Y) + I_{\min}(X;Y,Z) = I(X;Y)$$

Suppose now that there is a direct regulating relation between X and Y, in which case we have $I(X;Y) \neq 0$. At this point, by dividing both sides of the equality given above by $I(X;Y)$ we obtain

$$\frac{\text{Unique}_Z(X;Y)}{I(X;Y)} + \frac{I_{\min}(X;Y,Z)}{I(X;Y)} = 1$$

The first addend on the left therefore captures the portion of mutual information between X and Y due to the contribution of the unique information of the two variables X and Y, in opposition to the second addend which instead reflects the portion of mutual information due to redundancy of all the variables including Z. Finally, we show the percentage count of the types of triples present in the datasets of the article[1] obtained using the python implementation given in the last chapter.

[1] Link to the github page containing the simulated networks: https://github.com/Tchanders/network_inference_tutorials/tree/master/goldstandards

	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6
50_ecoli_1	86.41%	12.05%	1.18%	0.11%	0.21%	0.05%
50_ecoli_2	82.81%	14.37%	2.42%	0.24%	0.1%	0.06%
50_yeast_1	82.67%	15.95%	0.48%	0.22%	0.53%	0.15%
50_yeast_2	66.61%	28.24%	1.7%	1.23%	1.56%	0.65%
50_yeast_3	64.43%	29.65%	2.48%	0.99%	1.55%	0.89%

Table 3.1: Percentages of items per class of the datasets used

We can now explain what the idea behind PIDC consists of. We note that in all networks, most of the triples belong to the class of unconnected or to the class with only one edge. Thus by fixing two random variables X and Y, whose respective associated genes are connected in the network and considering all the possible triples including them, we expect most of these triples to be of class 1, similarly if X and Y are associated to unrelated genes we expect most triples to be of class 0. Thus by what we observed at the beginning, we expect that the portion of unique information between two variables is on average higher when they are connected. Based on these observations, the following measure will provide a high value for connected pairs.

Definition 3.2.1 (Proportional unique contribution)**.** Let S denote the set of all random variables of the regulatory network. Given two random variables X and Y in S such that $I(X;Y) \neq 0$, representing two distinct genes of the network, the proportional unique contribution, PUC for short, is defined as

$$u_{X,Y} := \sum_{Z \in S \setminus \{X,Y\}} \frac{\text{Unique}_Z(X;Y)}{I(X;Y)} + \sum_{Z \in S \setminus \{Y,X\}} \frac{\text{Unique}_Z(Y;X)}{I(X;Y)}$$

If $I(X;Y) = 0$ then $u_{X,Y} := 0$. We note that although the unique information is not symmetrical in its arguments, it holds that $u_{X,Y} = u_{Y,X}$.

Now that we have all the necessary definitions we can bring attention to the phases in which the algorithm is divided.

PUC score evaluation

We denote by $E \in \mathcal{M}_{n_g,n_c}(\mathbb{R})$ the scRNA-Seq dataset provided as input to the algorithm, where $n_g, n_c \in \mathbb{N}$ are the number of genes in the network and the number of cells, respectively. In order to estimate information measures, PIDC needs both to discretize the values in the data matrix E and the joint distributions between the pairs of elements in the dataset. This is achieved using the histogram estimator introduced in Section 1.2.1 with which the joint distribution is calculated for each pair of genes in the network using as the number of bins the lower integer approximation of the root of the number of cells in the network $m = \lfloor \sqrt{n_c} \rfloor$ and using the rows of the matrix E as samples of the genes in the network. Specifically, given two distinct genes of the network, denoting by X the random variable having as density function the joint probability of the two genes, we consider the discretisation X^{Δ}, Definition 1.3.14, having parameter $\Delta = \frac{1}{m}$. Therefore, for a generic couple of indices $i, j \in \{1, 2, \ldots, m\}$, the estimator takes the following form:

$$\hat{p}_{i,j}^{(g,h)} := \frac{\lfloor\sqrt{n_c}\rfloor^2}{n_c}\sum_{k=1}^{n_c} \mathbb{1}_{B_{i,j}}((E_{g,k}, E_{h,k})) \tag{3.5}$$

In summary, this estimator is obtained by considering the natural discretisation induced by the histogram bins on the sample space. Once this discrete distribution has been calculated for each pair of variables in the network, the PUC scores are calculated.

Ranking

Due to the variability of gene expression in dateset, the PUC scores obtained may not be directly comparable to one another, and are therefore insufficient to provide a confidence estimate of the actual presence of a regulatory relationship between genes. To solve this problem, a probability-based score is provided that takes into account the distribution of PUC scores over the entire network. In this way, values derived from the distribution, such as the mean and variance, can be used to compare scores more accurately with each other.

Definition 3.2.2 (Confidence)**.** Given two genes X and Y of the network, we define the confidence of the edge between them as

$$c_{X,Y} := F_X(u_{X,Y}) + F_Y(u_{X,Y})$$

Where $F_X(\cdot)$ is the cumulative distribution function of all PUC scores involving gene X obtained by assuming a Gaussian cumulative distribution having as parameters the estimators $\hat{\mu}_{\text{MLE}}(X)$ and $\hat{\sigma}_{\text{MLE}}(X)$ defined in 1.2.1.

The scores thus obtained constitute the final ranking that the algorithm returns in output.

Algorithm 3.2 PIDC

```
Require: $E$ network gene expression matrix
Ensure: $R$ matrix of regulatory relationships between gene pairs
 1: for $g = 1$ to $n_g$ do
 2:     for $h = g + 1$ to $n_g$ do
 3:         $P^{(g,h)} \leftarrow$ Calculate discrete probability estimation        ▷ Definition 3.5
 4:     end for
 5: end for
 6: for $g = 1$ to $n_g$ do
 7:     for $h = g + 1$ to $n_g$ do
 8:         for $l = h + 1$ to $n_g$ do
 9:             $U_{g,h} \leftarrow$ Calculate PUC scores using $P^{(g,h)}, P^{(g,l)}, P^{(h,l)}$        ▷ Definition 3.2.1
10:         end for
11:     end for
12: end for
13: for $g = 1$ to $n_g$ do
14:     for $h = g + 1$ to $n_g$ do
15:         $R_{g,h} \leftarrow$ Calculate confidence score using U        ▷ Definition 3.2.2
16:     end for
17: end for
18: return $R$
```

3.3 SCODE

SCODE is an inference algorithm for gene regulatory networks by Matsumoto et al. [18]. It proposes an ODEs-based approach to infer a gene regulatory network starting from scRNA-Seq data and pseudotimes. The algorithm consists of the iteration of two phases, one of regression and one of numerical integration. The algorithm outputs for each ordered pair of genes (i, j) a value $A_{ij} \in \mathbb{R}$ whose sign distinguishes between activators and inhibitors.

Input: scRNa-Seq dataset, pseudotimes
Output: Ranking of directed, signed network edges
Model: ODEs system, regression, ODEs representation 2.2.4
Key concepts: regression 1.2.2
Time complexity: $O(nd^3 + n^2d + n^3)$ where n is the number of genes and d the dimension of the reduced system

3.3.1 Algorithm

SCODE describe the dynamics of genes'products through the use of the following system of linear ODE

$$\frac{dx_i}{dt}(t) = \sum_{k=1}^{n} A_{ik}x_k(t), \forall i \in \{1, 2, \ldots, n\}$$

According to the criterion given in the second chapter, the relationship between two generic elements i and j of the network is determined by the sign of the partial derivative of f_i with respect to x_j, which in this case, denoting by e_i the elements of the standard basis of $\mathbb{R}^n$, becomes

$$\frac{\partial}{\partial x_j}\frac{dx_i}{dt}(t) = \frac{\partial}{\partial x_j}\sum_{k=1}^{n} A_{ik}x_k(t) = \sum_{k=1}^{n}\frac{\partial}{\partial x_j}A_{ik}x_k(t) = A_{ij}$$

Therefore A_{ij} is the parameter that mediates the regulation interaction $j \to i$. In matrix form we therefore have

$$\frac{d\mathbf{x}}{dt}(t) = A\mathbf{x}(t)$$

Where $\mathbf{x}(t) = (x_1(t), x_2(t), \ldots, x_n(t))$ is the gene expression vector of the genes in the network. Instead of solving this system directly, SCODE rewrite the system in the following form

$$\frac{d\mathbf{z}}{dt}(t) = B\mathbf{z}(t) \tag{3.6}$$

Where $\mathbf{z} \in \mathbb{R}^m$ and $B \in \mathcal{M}_{m,m}(\mathbb{R})$ is a diagonal matrix with $m \ll n$. This linear system, due to the fact that the matrix B is diagonal, admits a known analytic solution for a given initial value $\mathbf{z}(0) \in \mathbb{R}^m$, in particular

$$z_i(t) = z_i(0)\exp(B_{ii}t), \forall i \in \{1, 2, \ldots, m\} \tag{3.7}$$

To reduce to such a system it is necessary to find the matrix $W \in \mathcal{M}_{n,m}(\mathbb{R})$ such that $W\mathbf{z} = \mathbf{x}$, in fact we note that assuming the knowledge of this matrix and denoting its pseudo-inverse, Definition 1.5.1 with W^+, we can write

$$WB\mathbf{z}(t) = WW^+WB\mathbf{z}(t) = WB^T(W^+W)^T\mathbf{z}(t) \overset{(*)}{=} WBW^+W\mathbf{z}(t) = WBW^+(W\mathbf{z}(t)) = WBW^+\mathbf{x}(t)$$

Where in $(*)$ we use the fact that both B and W^+W are symmetric by definition. Therefore

$$\frac{d\mathbf{z}}{dt}(t) = B\mathbf{z}(t) \Leftrightarrow W\frac{d\mathbf{z}}{dt}(t) = WB\mathbf{z}(t) \Leftrightarrow \frac{dW\mathbf{z}}{dt}(t) = WBW^+W\mathbf{z}(t) \Leftrightarrow \frac{d\mathbf{x}}{dt}(t) = WBW^+\mathbf{x}(t)$$

So starting from WBW^+ we can obtain the matrix A which solves the original system. The calculation of B and W is performed by means of a linear regression, whose sample is generated by random sampling the matrix B, in two distinct phases which alternate iteratively.

Sampling

During the first step we want to calculate the sample of vectors $\{\mathbf{z}(t_1), \mathbf{z}(t_2), \ldots, \mathbf{z}(t_m)\}$, solution of the reduced system, where t_i is the pseudotime of the i-th gene. First of all, a random diagonal matrix B is sampled from the set of diagonal matrices whose elements along the diagonal are drawn uniformly in the interval $[-10, 2]$, then to evaluate $\mathbf{z}(t_i)$ we proceed by solving the reduced linear system 3.6 using Formula 3.7, this concludes the first phase.

Regression

We then proceed with the estimation of the W matrix by means of a linear regression. In particular, we set up the following regressions $\mathbf{x}(t_i) \sim N(W\mathbf{z}(t_i), \sigma^2)$ for every $i \in \{1, 2, \ldots, m\}$. For the estimation of W we use the MSE estimator introduced in Definition 1.2.20 of which we know the analytical form, we therefore obtain

$$\hat{W}_{MSE} = (Z^T Z)^{-1} Z X \tag{3.8}$$

Where X and Z are the matrices whose i-th column is respectively $\mathbf{x}_i$ and $\mathbf{z}_i$. An appropriate value of B must allow to obtain an accurate prediction of X starting from Z, therefore the mean squared error of the model is used as a score to obtain the best matrix in the sampling just carried out. Precisely, denoting with B_i Z_i and W_i the matrices of the i-th iteration, we choose the index $j \in \{1, 2, \ldots, n\}$ in such a way that

$$\text{MSE}(W_j, Z_j) = \min_i \text{MSE}(W_i, Z_i),\ A \text{ is then inferred using } W_j B_j W_j^+.$$

Algorithm 3.3 SCODE

```
Require: $E$ network gene expression matrix, maxiter, $\sigma$
Ensure: $R$ matrix of regulatory relationships between gene pairs
 1: Min $\leftarrow +\infty$
 2: $B \leftarrow$ random diagonal matrix                                  ▷ $B_{ii} \in [-10, 2]$
 3: for $k = 1$ to maxiter do
 4:     $Z \leftarrow$ Solution of the reduced linear system $d\mathbf{z} = B\mathbf{z}dt$    ▷ Formula 3.7
 5:     $W \leftarrow$ Linear regression solution                        ▷ Formula 3.8
 6:     if MSE$(W, E)$ < Min then
 7:         Min $\leftarrow$ MSE$(W, E)$
 8:         $\hat{B} \leftarrow B$
 9:         $\hat{W} \leftarrow W$
10:     end if
11:     $i \leftarrow$ value uniformly extracted from $\{1, 2, \ldots, m\}$
12:     $B_{ii} \leftarrow$ value uniformly extracted from $[-10, 2]$
13: end for
14: $\bar{W} \leftarrow$ evaluate pseudoinverse of $\hat{W}$
15: $R \leftarrow \bar{W}\hat{B}\bar{W}^{-1}$
16: return $R$
```

3.4 SINCERITIES

SINCERITIES, Papili Gao et al. [21], (acronym of SINgle CEll Regularized Inference using TIme stamped Expression profileS) in order to calculate the ranking of gene pairs uses regularized linear regression, with additional constraint on the parameters, calculated on the temporal variation of gene expression distributions, while to infer the type of regulation between gene pairs uses partial correletion analyses. The algorithm looks at gene expression as a random variable whose distribution function varies over time, in order to obtain this representation it requires multiple time series sharing the same time scale and each belonging to a distinct single cell as input data. The data have therefore the form of n_t tables containing the gene expression values as the gene and the belonging cell vary, where n_t indicates the number of instants in time. We denote by $E^{(t)}$ the table at the instant t and by $E^{(t)}_{i,j}$ the value of the expression of the gene i, in the cell j at the instant t. The time-dependent distribution for each gene is obtained from the values of the expression as the cell varies, we indicate the random variable associated with the gene j at time t by contracting the cell index, thus denoting it $E^{(t)}_j$

Input: scRNa-Seq dataset, time points / pseudotimes
Output: Ranking of directed, signed network edges
Model: Regression, correlation
Key concepts: Kolmogorov-Smirnov distance 1.2.11, correlation 1.2.13, regression 1.2.2

3.4.1 Algorithm

The algorithm is divided into three different phases, a data processing phase during which the normalized distributions of the gene expression of each gene are calculated for each of the times, then we proceed with the calculation of the distances between these distributions by means of the distance of Kolmogorov-Smirnov, during the second phase a measure of confidence of the presence of a regulation ratio between each ordered pair of genes is provided through the use of a regularized linear regression, the algorithm allows to choose between the ridge, the lasso and the elastic-net, finally during the third phase the type of regulation (activation or repression) of all the ordered pairs of the network is inferred, independently from the value of the measure previously calculated. At the end of the execution of the algorithm there we have a ranking, with respect to the measurement of phase two, of ordered pairs of genes in the network. The algorithm is based on the assumption that changes in the expression of transcription factors of regulatory genes will alter the expression of target genes in subsequent instants, the calculation of the distances between the distributions is then used to provide a measure of this alteration.

Distances evaluation

The genes of the network are modeled by means of random variables whose distribution varies over time, therefore we introduce a series of notations that we use in the rest of the section

n_g : number of genes in the dataset
n_c : number of cells in the dataset
n_t : number of time instants of the dataset
t_k : k-th time instant of the dataset

$E_{i,j}^{(k)} \in \mathbb{R}^+$: the gene expression value of gene i, in cell j at time t_k
$X_i^{(k)} : \mathcal{X}_i \to \mathbb{R}^+$: the random variable representing the i-th gene at time t_k
F_i^k : the cumulative distribution of the variable $X_i^{(k)}$
$D \in \mathcal{M}_{n_g,n_t}(\mathbb{R})$: the matrix of distances between cumulative distributions of gene expression from sequential instants in time

As a first step, the elements of the distance matrix D are calculated using the Kolmogorov-Smirnov distance, in particular

$$D_{i,k} := \max_{x \in \mathcal{X}_i} |F_i^{(k+1)}(x) - F_i^{(k)}(x)| \tag{3.9}$$

Where, according to the definition provided in the chapter on mathematical preliminaries, we have

$$F_i^k(x) := \frac{1}{n_c} \sum_{j=1}^{n_c} \mathbb{I}_{(-\infty,x]}(E_{i,j}^{(k)})$$

However, the values just calculated do not take into consideration the fact that the time which elapses between two generic consecutive instants of the dataset is not necessarily the same, therefore we proceed with the normalization with respect to the time duration of each element of the matrix D, obtaining a array of normalized values that we denote by $\hat{D}$, in formulae

$$\hat{D}_{i,k} := \frac{D_{i,k}}{t_{k+1} - t_k} \tag{3.10}$$

The first step is therefore completed.

Regression

The idea behind the second phase of the algorithm, for calculating the confidence measure of the presence of directed arcs in the network, is to highlight the fact that under the hypothesis made at the beginning of the session, according to which the variation of gene expression of a target gene directly depends on the variation of the gene expression of the transcription factors of the corresponding regulatory genes in the immediately preceding instants it is reasonable to think of predicting the variation of the gene expression of a given gene at a fixed instant using the variations in the gene expression of all genes in the network at the previous instant. Mathematically, a linear dependence is assumed between the variations of the gene expression distributions of the genes in the network whose parameters are obtained by solving $l-2$ linear regressions. Specifically, for each gene j a regularized linear regression is set up whose dependent variables are $\hat{D}_{j,k+1}$ and whose features are $\hat{D}_{1,k}, \dots, \hat{D}_{n_g,k}$ for each $k \in \{1, 2, \dots, l-2\}$. It is important to note that the regression parameters act as a measure of the influence that one gene exerts on another, therefore in formulating the regression the non-negativity of these parameters is imposed as a constraint. into formulas

$$\begin{cases} \hat{D}_{j,k+1} \sim N(\alpha_{1,j}\hat{D}_{1,k} + \alpha_{2,j}\hat{D}_{2,k} + \dots + \alpha_{n_g,j}\hat{D}_{n_g,k}, \sigma^2), \forall k \in \{1, 2, \dots, l-2\} \\ \alpha_{p,q} \geq 0, \forall p, q \in \{1, 2, \dots, n_g\} \end{cases} \tag{3.11}$$

Where the parameter $\alpha_{p,q}$ describes the influence of gene p on gene q. As the authors note in the article, this regression needs regularization as it is often under-determined due to the fact that the number of genes often exceeds the number of time windows. An example of a system follows

$$\begin{pmatrix} \hat{D}_{1,2} \\ \hat{D}_{1,3} \\ \hat{D}_{1,4} \end{pmatrix} = \begin{pmatrix} \hat{D}_{1,1} & \hat{D}_{2,1} & \hat{D}_{3,1} & \hat{D}_{4,1} \\ \hat{D}_{1,2} & \hat{D}_{2,2} & \hat{D}_{3,2} & \hat{D}_{4,2} \\ \hat{D}_{1,3} & \hat{D}_{3,3} & \hat{D}_{3,3} & \hat{D}_{4,3} \end{pmatrix} \begin{pmatrix} \alpha_{1,1} \\ \alpha_{2,1} \\ \alpha_{3,1} \\ \alpha_{4,1} \end{pmatrix}$$

The effective resolution of the regression is performed using the GLMNET[2] algorithm, based on a constrained version of gradient descent which is not covered in this thesis.

Correlation

Once the confidence measures have been calculated, the types of regulation between pairs of genes must be inferred. As previously mentioned this is done by calculating the sign of the Spearmann correlation coefficient 1.2.13 for each pair of genes using the distributions obtained by considering the combined expression of all the instants in time.

Algorithm 3.4 SINCERITIES

```
Require: $E^{(1)}, \dots, E^{(n_t)}$ gene expression matrices
Ensure: $\alpha, \rho$ regulations and sign matrices
1: $D \leftarrow$ Kolmogorov-Smirnov distance calculation        ▷ Definition 3.9
2: $\hat{D} \leftarrow$ normalize distance matrix $D$        ▷ Definition 3.10
3: for $j = 1$ to $n_g$ do
4:     $\alpha_j \leftarrow$ solve system 3.11        ▷ done using GLMNET
5: end for
6: $\rho \leftarrow$ evaluate Spearmann correlation rank between couple
7: return $\alpha, \rho$
```

[2] The R package we refer to is avaiable at https://cran.r-project.org/web/packages/glmnet/index.html, while the python library we is available at https://pypi.org/project/glmnet/

4

Applications

4.1 Inference

In this section we show the results obtained using PIDC to infer the gene regulatory network presented in Pedicini Marco, Barrenäs et al. [16] using the dataset GSE36842[1] containing microarray gene expression values of Homo sapiens. As mentioned in Section 3.2, PIDC makes an assumption about the triples of genes in the network, in particular it is necessary that almost all triples are concentrated in the first two classes, preferably with a high value of class 2 elements. Therefore we checked that the percentages of elements per class were comparable to those used by the authors of the article in Figure 3.1. Almost all triples belong to the classes of disconnected and one-edge triples, which together account for 98.8% of the total, making it possible to proceed with the inference.

	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6
Number	17233	3330	246	15	1	0
Percent	82.8%	16.0%	1.2%	0.07%	0.0048%	0%

Table 4.1: Percentages of items per class of the dataset

Definition 4.1.1 (Core network)**.** Given a gene regulatory network represented by an oriented graph $G = (V, E)$, the core network is defined as the graph obtained by removing from G all the vertices that do not belong to any cycle of the graph.

The idea behind the analysis of the network is to use the core network as a training set, i.e. as a data set on which to perform the tuning of model parameters. We used the histogram estimator 1.2.1 as the joint probability distribution estimator, and the plug-in estimator 1.3.15 as estimator of information measures. For mutual information we therefore have

$$\hat{I}(X,Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}}\hat{p}(x,y)\log\frac{\hat{p}(x,y)}{\hat{p}(x)\hat{p}(y)}$$

Within the dataset used, there are 39 samples for a total of 20 genes. We know from Remark 1.2.2 that the optimal number of bins for the estimator is

[1] Dataset avaiable at the following link https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE36842

$$m_{\text{opt}} \approx n^{\frac{1}{d+1}} = 39^{\frac{1}{3}} \approx 3.4$$

This result is found in the numerical simulation performed below. By varying the number of bins, we evaluate the boxplot graph of the plug-in estimator of mutual information over a sample of 10^3 elements having the same random joint distribution.

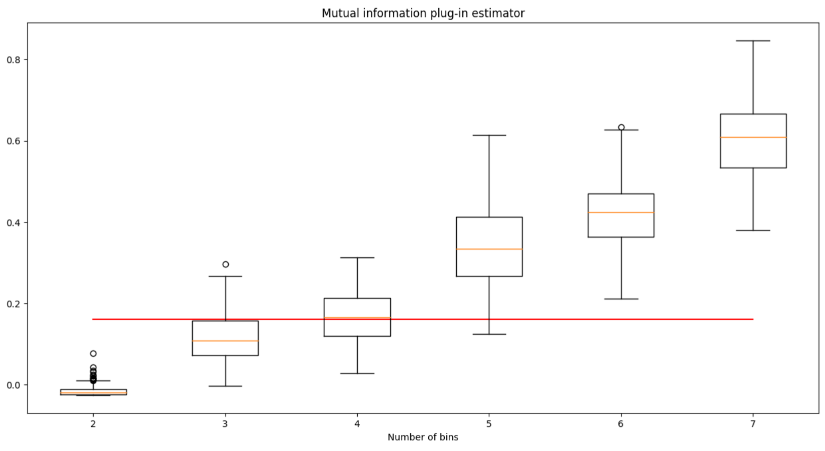


Fig. 4.1: In Figure the boxplot plot of the plug-in estimator of the mutual information as the number of bins varies. In red is the exact value of the mutual information of the random distribution.

For the partial information decomposition measurements, we again use the plug-in estimator.

$$\hat{I}_{\text{spec}}(y, X) = \sum_{x\in\mathcal{X}} \frac{\hat{p}(y,x)}{\hat{p}(y)} \log \frac{\hat{p}(y,x)}{\hat{p}(y)\hat{p}(x)} \qquad \hat{I}_{\min}(Z;X,Y) = \sum_{z\in\mathcal{Z}} \hat{p}(z) \min_{S\in\{X,Y\}} \hat{I}_{\text{spec}}(y,S)$$

$$\hat{U}_Y(Z;X) = \hat{I}(X;Z) - \hat{I}_{\min}(Z;X,Y)$$

For these measures we repeated the same numerical simulation performed in Figure 4.1.

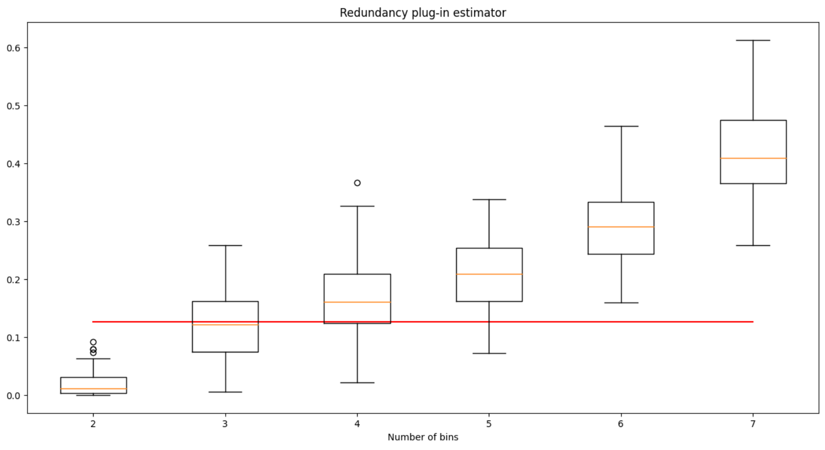
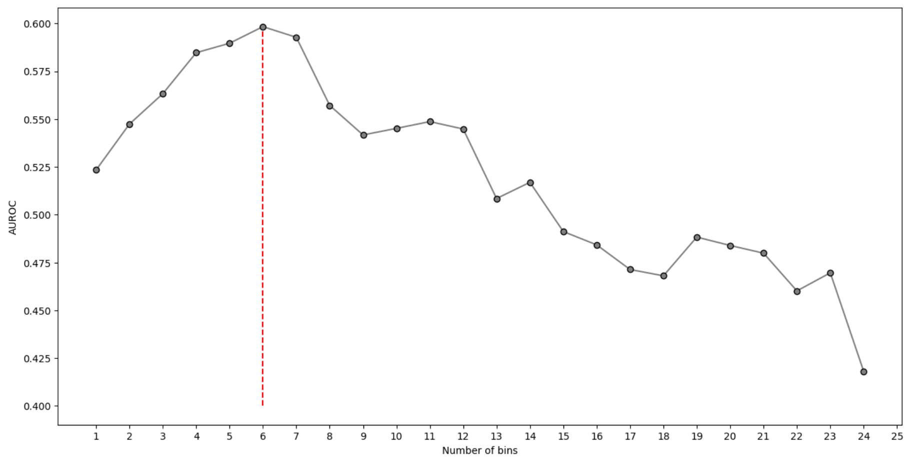


Fig. 4.2: In Figure the boxplot plot of the plug-in estimator of Redundancy as the number of bins varies. In red is the exact value of the Redundancy of the random distribution.

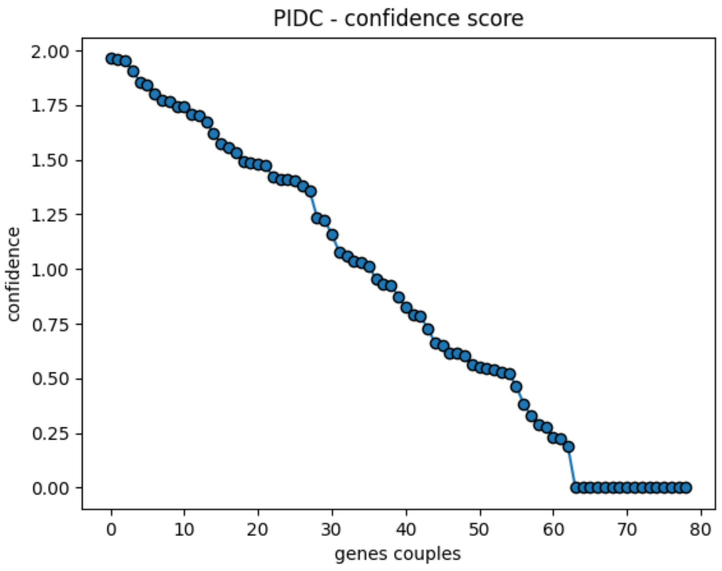
Based on these observations, we expect that the optimal value of the number of bins is not high since the theoretical result ensures that the optimal value is independent of the probability distribution of the sample. Therefore we run the PIDC algorithm on the dataset by varying the number of bins in the range $\{2, 3, \ldots, 25\}$ to get an overview of the inference trend. The following image shows the value of the AUROC score obtained by varying the number of bins.



The optimal number of bins found on the core network is therefore $m_{\text{opt}} = 6$. We then proceed to run the PIDC algorithm on the overall network consisting of 46 genes. The AUROC scores obtained are as follows

AUROC score on the core network for 6 bins: 0.5983483483483483
AUROC score on the overall network for 6 bins: 0.598467395548614

The following image shows the confidence scores obtained for all pairs of genes in the network sorted in descending order. We note that from Definition 3.2.2 the maximum value is 2.



4.2 Implementation of algorithms

In this section, we present Python implementations of the algorithms discussed in the thesis. The libraries required for all of them are: scikit-learn 1.0, numpy 1.21.6 and pandas 1.3.5.

4.2.1 ARACNE

Python implementation of ARACNE 3.1. To execute the code from command line there are the following alternatives. Specify only the mutual information threshold I_0 in which case gene expression and output file are set by default to *expr.txt* and *out.txt* respectively.

```
python aracne.py <I0>
```

Alternatively, it is possible to indicate the paths to the gene expression and output files.

```
python aracne.py <I0> <expfile> <outfile>
```

The file containing the gene expression values must be a Tab-spaced table along whose rows the gene varies and along whose columns the sample varies. Next, the body of the python code, excluding the input data handling functions.

```
# ===========[MUTUAL INFORMATION ESTIMATION]==========
# data : gene expression value
# ==================================================
def get_mutual_information(data):
    # kernel density estimator
    kernel = gaussian_kde(data)
    joint_p = kernel(data)
    kernel = gaussian_kde(data[0,:])
    p1 = kernel(data[0,:])
    kernel = gaussian_kde(data[1,:])
    p2 = kernel(data[1,:])
    # mutual information estimator
    MI = np.sum(np.log(joint_p/p1/p2))/data.shape[1]
    return MI

# ===========[MUTUAL INFORMATION]==========
# data : gene expression value
# ========================================
def get_mutual_information_matrix(data):
    genesnum = data.shape[0]
    MI = np.zeros((genesnum,genesnum))
    for i in range(genesnum):
        for j in range(i+1,genesnum):
            temp = get_mutual_information(data[[i,j],:])
            MI[i,j] = temp
            MI[j,i] = temp
    return MI

# =============[ARACNE]=============
# data : gene expression value
# I0 : mutual information threshold
# =================================
def aracne(data, I0):
    MI = get_mutual_information_matrix(data)
    genesnum = data.shape[0]
    # Evaluate MI
    couples = list()
    for i in range(genesnum):
        for j in range(i+1,genesnum):
            if MI[i,j] > I0:
                couples.append((i,j))
    # Apply data processing theorem
    R = np.zeros((genesnum,genesnum))
    for c in couples:
        i = c[0]
        j = c[1]
        R[i,j] = 1
        R[j,i] = 1
        for k in range(genesnum):
            if k != i and k != j:
                if MI[i,j] < MI[i,k] and MI[i,j] < MI[j,k]:
                    R[i,j], R[j,i] = 0, 0
                elif MI[i,k] < MI[i,j] and MI[i,k] < MI[j,k]:
                    R[i,k], R[k,i] = 0, 0
                else:
                    R[j,k], R[k,j] = 0, 0
    return R
```

4.2.2 PIDC

Python implementation of PIDC 3.2. To execute the code from the command line there are the following alternatives. In the first one the number of bins to be used must be specified. In this case, the gene expression and output file are set by default to *expr.txt* and *out.txt* respectively

```
python pidc.py <bins>
```

In the second, it is possible to additionally indicate the paths to the gene expression and output files

```
python pidc.py <bins> <expfile> <outfile>
```

The file containing the gene expression values must be a table along whose rows the gene varies and along whose columns the sample varies, additionally the values must be spaced using the Tab character. Next, the body of the python code, excluding the input data handling functions.

```
# =========[EVALUATE PUC SCORE]==========
# puc : puc score matrix
# pij : estimator of density distribution
# pi : estimator of density distribution
# i1,i2,i3 : index oredering
# bins : bins number
# =======================================
def evaluatePUC(puc, p12, p13, p23, p1, p2, p3, i1, i2, i3, bins):
    # Redundancy
    R_1_23, R_2_13, R_3_12 = 0, 0, 0
    MI_12, MI_13, MI_23 = 0, 0, 0
    for i in range(bins):
        s12, s21, s13, s31, s23, s32 = 0, 0, 0, 0, 0, 0
        for j in range(bins):
            if p12[i,j] != 0:
                s12 += p12[i,j] * np.log(p12[i,j]/p1[i]/p2[j])
            if p12[j,i] != 0:
                s21 += p12[j,i] * np.log(p12[j,i]/p1[j]/p2[i])
            if p13[i,j] != 0:
                s13 += p13[i,j] * np.log(p13[i,j]/p1[i]/p3[j])
            if p13[j,i] != 0:
                s31 += p13[j,i] * np.log(p13[j,i]/p1[j]/p3[i])
            if p23[i,j] != 0:
                s23 += p23[i,j] * np.log(p23[i,j]/p2[i]/p3[j])
            if p23[j,i] != 0:
                s32 += p23[j,i] * np.log(p23[j,i]/p2[j]/p3[i])
        MI_12, MI_13, MI_23 = MI_12+s12, MI_13+s13, MI_23+s23
        R_1_23, R_2_13, R_3_12 = R_1_23+min(s12, s13), R_2_13+min(s21, s23), R_3_12+min(s31, s32)
    # Unique contribution
    U_3_12, U_3_21= max(MI_12 - R_1_23,0), max(MI_12 - R_2_13,0)
    U_2_13, U_2_31 = max(MI_13 - R_1_23,0), max(MI_13 - R_3_12,0)
    U_1_23, U_1_32 = max(MI_23 - R_2_13,0), max(MI_23 - R_3_12,0)
    # Puc scores
    if MI_12 > 0:
        puc[i1,i2] += U_3_12/MI_12 + U_3_21/MI_12
        puc[i2,i1] += U_3_12/MI_12 + U_3_21/MI_12
    if MI_13 > 0:
        puc[i1,i3] += U_2_13/MI_13 + U_2_31/MI_13
        puc[i3,i1] += U_2_13/MI_13 + U_2_31/MI_13
    if MI_23 > 0:
        puc[i2,i3] += U_1_23/MI_23 + U_1_32/MI_23
        puc[i3,i2] += U_1_23/MI_23 + U_1_32/MI_23

# ==================[PIDC]======================
# data : gene expression data matrix
# bins : bins number for MLE estimator
# =============================================
def pidc(data, bins):
    genesnum = data.shape[0]
    # ======[Evaluate PUC score]======
    # puc : matrix of puc scores
    # ==============================
    puc = np.zeros((genesnum,genesnum))
    for i in range(genesnum):
        for j in range(i+1,genesnum):
            X = data[i,:]
            Y = data[j,:]
            # Estimate joint density function
            pXY = get_probabilities(X,Y,bins)
            pX = np.sum(pXY, 1)
            pY = np.sum(pXY, 0)
            for k in range(j+1,genesnum):
                Z = data[k,:]
                # Estimate joint density functions
                pXZ = get_probabilities(X,Z,bins)
                pYZ = get_probabilities(Y,Z,bins)
                pZ = np.sum(pXZ, 0)
```

```
                    # get puc scores for X, Y, Z
                    evaluatePUC(puc, pXY, pXZ, pYZ, pX, pY, pZ, i, j, k, bins)
        # ======[FIT DISTRIBUTION]======
        # args : list of MLE estimators
        # =============================
        args = list()
        for i in range(genesnum):
            args.append(gamma.fit(np.delete(puc[i,:], i)))
        # =======[RELATIONSHIP MATRIX]======
        # R : (genesnum x genesnum) matrix
        # ==================================
        R = np.zeros((genesnum,genesnum))
        for i in range(genesnum):
            for j in range(i,genesnum-1):
                val = gamma.cdf(puc[i,j],args[i][0],args[i][1],args[i][2])
                val += gamma.cdf(puc[i,j],args[j][0],args[j][1],args[j][2])
                R[i,j+1] = val
                R[j+1,i] = val
        return R
```

4.2.3 SCODE

Python implementation of SCODE 3.3. To execute the code from the command line there are the following alternatives. In the former, the size of the reduced system, the maximum number of iterations per run and the number of runs to be performed to calculate the average matrix. In this case, the gene expression, pseudo-times and output file are set by default to *expr.txt*, *time.txt* and *out.txt* respectively

```
python scode.py <dim> <maxite> <execnum>
```

In the second, it is possible to additionally indicate the paths to the gene expression, pseudo-times and output files

```
python scode.py <dim> <maxite> <execnum> <expfile> <timefile> <outfile>
```

The file containing the gene expression values must be a table along whose rows the gene varies and along whose columns the sample varies, additionally the values must be spaced using the Tab character. The file containing the pseudotime values must be a table with one column along whose rows the sample varies. Next, the body of the python code, excluding the input data handling functions.

```
# ============[Sampling of Z]==============
# Sampling of the Z matrix by means of the
# analytical solution
# ========================================
def sampleZ(Z, dimension, cellsnum, B, pseudotime):
    for i in range(dimension):
        for j in range(cellsnum):
            Z[i,j] = np.exp(B[i]*pseudotime[j]) + np.random.uniform(0.001,0.002)

# ============[Data reading]===============
# INPUT
# expfile  :  scRna-Seq expressions file
# timefile : pseudotimes file
# OUTPUT
# X : matrix of expressions
# pseudotime : normalized pseudotime vector
# ========================================
def getData(expfile, timefile):
    X = read_csv("./"+expfile,sep='\t',header=None).to_numpy()
    cellsnum = X.shape[1]

    pseudotime = read_csv("./"+tempfile,sep='\t',header=None).to_numpy()
    pseudotime = pseudotime[0:cellsnum,1]
    pseudotime = pseudotime/np.max(pseudotime)
    return X,pseudotime

# ==================[SCODE]====================
# dimension : reduced system dimension
# maxite : maximum number of iterations
# mode : regression solver, standard = analytic
# ============================================
def scode(X, pseudotime, dimension, maxite, mode=0):
    # ===================[VALUES]======================
    # genesnum : genes number
```

```python
    # cellsnum : cells number
    # X : (genesnum x cellsnum) matrix of gene expression
    # W : (genesnum x dimension) matrix of gene expression
    # Z : (dimension x cellsnum) matrix of reduced system
    # ==================================================
    genesnum = X.shape[0]
    cellsnum = X.shape[1]
    W = np.zeros((genesnum,dimension))
    Z = np.zeros((dimension,cellsnum))
    bestW = None

    # ================[INITIALIZATION]==================
    # RSS : Current value of the mean squared error
    # newB : current matrix of the reduced system
    # oldB : previous matrix of the reduced system
    # ==================================================
    RSS = np.inf
    new_B = np.random.uniform(minB,maxB,size=dimension)
    old_B = np.copy(new_B)

    # ===================[OPTIMIZATION]===================
    # Optimizing W and B matrices
    # reg : instance of the numpy class of linear regressors
    # ====================================================
    if mode == 1:
        reg = LinearRegression()
    for i in range(maxite):
        # Sampling B matrix
        target = np.random.randint(0,dimension)
        new_B[target] = np.random.uniform(minB, maxB)

        if i == maxite-1:
            new_B = np.copy(old_B)

        # Get Z from B
        sampleZ(Z, dimension, cellsnum, new_B, pseudotime)

        # Evaluate W with regression
        # mode = 0 : analytical form
        # mode = 1 : gradient descent
        if mode == 0:
            tempo = np.linalg.inv(np.dot(Z,Z.transpose()))
            for i in range(genesnum):
                W[i,:] = np.dot(tempo,np.dot(Z,X[i,:]))
        elif mode == 1:
            reg.fit(X.transpose(),Z.transpose())
            W = reg.coef_.transpose()

        # Calculate mean squared error
        deviation = X - np.dot(W,Z)
        tmp_RSS = 0
        for i in range(genesnum):
            tmp_RSS += np.dot(deviation[:,i].transpose(),deviation[:,i])
        if tmp_RSS < RSS:
            RSS = tmp_RSS
            bestW = W.copy()
            old_B[target] = new_B[target]
        else:
            new_B[target] = old_B[target]
    W = bestW.copy()

    # =============[INFERENCE]==============
    # Inference of originial system matrix A
    # invW : pseudoinverse of W matrix
    # ======================================
    B = np.diag(new_B)
    invW = np.linalg.pinv(W)
    A = np.dot(W,np.dot(B,invW))
    return A
```

4.2.4 SINCERITIES

Python implementation of SINCERITIES 3.4. To execute the code from the command line must be specified: the gene expression and output files. If not specified, the gene expression and output file are set by default to *expr.txt* and *out.txt* respectively

```
python sincerities.py <expfile> <outfile>
```

The gene expression file must contain the juxtaposition, along the columns, of the tables as time varies. Thus, along the rows the genes vary while along the columns both the sample and the time vary. The file containing the pseudotime values must be a table with one column along whose rows the sample varies. Next, the body of the python code, excluding the input data handling functions.

```python
# ============[DISTANCE EVALUATION]=============
# exprdata : gene expression value matrix
# (cells_num x time_poins_num x genes_num)
# timedata : time / pseudotime matrix
# time_points_num : number of time points
# genes_num : number of genes
# ============================================
def get_normalized_distances(exprdata, timedata, time_points_num, genes_num):
    distance_matrix = np.zeros((time_points_num-1, genes_num))
    for t in range(time_points_num-1):
        for g in range(genes_num):
            p1 = exprdata[:,t,g]
            p2 = exprdata[:,t+1,g]
            distance_matrix[t,g] = ks_2samp(p1,p2).statistic
    delta_time = timedata[1:] - timedata[:-1]
    return distance_matrix/delta_time

# ==========[LINEAR REGRESSION]===========
# D : normalized distance matrix
# time_points_num : number of time points
# genes_num : number of genes
# =======================================
def solve_linear_regression(D, time_points_num, genes_num):
    X = D[:time_points_num-2,:]
    # GLMNET parameters
    nfolds = X.shape[0]
    foldid = np.arange(0,nfolds,1)
    keep = True
    alpha = 0
    # Cross-validation and solution of regression
    R = np.zeros((genes_num,genes_num))
    for g in range(genes_num):
        Y = D[1:time_points_num-1,g]
        cvfit = cvglmnet(x = X.copy(), y = Y.copy(), alpha = alpha, exclude=[g], nfolds = nfolds, foldid = foldid, keep =
        R[g,:] = cvglmnetCoef(cvfit, s = 'lambda_min')[1:,0]
    return R

# ===========[SPEARMANN RANK]============
# D : normalized distance matrix
# time_points_num : number of time points
# =======================================
def get_spearmann_rank(data, cells_num, time_points_num, genes_num):
    rho = np.zeros((genes_num,genes_num),dtype=int)
    for i in range(genes_num):
        for j in range(i+1,genes_num):
            r1 = np.reshape(data[:,:,i],cells_num*time_points_num)
            r2 = np.reshape(data[:,:,j],cells_num*time_points_num)
            temp = spearmanr(r1,r2).correlation
            rho[i,j] = np.sign(temp)
            rho[j,i] = np.sign(temp)
    return rho

# =============[SINCERITIES]==============
# exprdata : gene expression value matrix
# timedata : time / pseudotime matrix
# =======================================
def sincerities(exprdata, timedata):
    cells_num = exprdata.shape[0]
    time_points_num = exprdata.shape[1]
    genes_num = exprdata.shape[2]
    D = get_normalized_distances(exprdata, timedata, time_points_num, genes_num)
    R = solve_linear_regression(D, time_points_num, genes_num)
    rho = get_spearmann_rank(exprdata, cells_num, time_points_num, genes_num)
    return R, rho
```

5 Conclusions

In this paper we have introduced the problem of inference of gene regulatory networks by giving an outline of some of the main methods of approaching this problem. Particular attention has been paid to methods based on information theory because of the ability of its tools to capture the interactions between network elements without assuming a linearity constraint. It is also observed how the limitations on the number of genes considered in examining the relationships between genes, imposed by classical information theory, can be overcomed by the use of partial information decomposition, which in this paper is restricted to the $n = 3$ case. The general case is dealt within the reference article [1] and led us to consider as a future work the possibility of extending the idea seen in PIDC to the case of more than three variables, we note however that due to the limitations imposed by the results on the estimation of the distribution of random variables, Remark 1.2.2 and Remark 1.2.4, this extension may not necessarily lead to an improvement in the performance of the algorithm.

As part of this thesis we provided Python implementations of the four algorithms discussed which we tested and compared with the original algorithms on the datasets analysed in the respective articles. Currently, the implementation of the PIDC algorithm does not present all the estimators used in the original article. We would like to draw attention to the fact that in its current state SINCERITIES algorithm is not executable on Windows operating system due to an incompatibility of one of the library used.

[1] Nonnegative decomposition of multivariate information [32]

References

1. Ibrahim A. Ahmad and Pi-Erh Lin. A nonparametric estimation of the entropy for absolutely continuous distributions (corresp.). *IEEE Trans. Inf. Theory*, 22:372–375, 1976.
2. Aryaman Arora, Clara Meister, and Ryan Cotterell. Estimating the entropy of linguistic distributions, 2022.
3. Shohag Barman and Yung-Keun Kwon. A Boolean network inference from time-series gene expression data using a genetic algorithm. *Bioinformatics*, 34(17):i927–i933, 09 2018.
4. Junyue Cao, Malte Spielmann, Xiaojie Qiu, Xingfan Huang, Daniel M Ibrahim, Andrew J Hill, Fan Zhang, Stefan Mundlos, Lena Christiansen, Frank J Steemers, Cole Trapnell, and Jay Shendure. The single-cell transcriptional landscape of mammalian organogenesis. *Nature*, 566(7745):496–502, February 2019.
5. Filippo Castiglione, Emiliano Mancini, Marco Pedicini, and Abdul Salam Jarrah. Quantitative modelling approaches. In Shoba Ranganathan, Michael Gribskov, Kenta Nakai, and Christian Schönbach, editors, *Encyclopedia of Bioinformatics and Computational Biology*, pages 874–883. Academic Press, Oxford, 2019.
6. Thalia Chan, Michael Stumpf, and Ann Babtie. Gene regulatory network inference from single-cell data using multivariate information measures. *Cell Systems*, 5:251–267.e3, 09 2017.
7. Hidde de Jong. Modeling and simulation of genetic regulatory systems: a literature review. *J Comput Biol*, 2002.
8. Michael R DeWeese and Markus Meister. How to measure the information gained from one symbol. *Network: Computation in Neural Systems*, 10(4):325, 1999.
9. Ellis Englesberg, Joseph Irr, Joseph Power, and Nancy Lee. Positive control of enzyme synthesis by gene c in the l-arabinose system. *Journal of bacteriology*, 90(4):946–957, 1965.
10. F. Fabris. *Teoria dell'informazione, codici, cifrari.* Bollati Boringhieri, 2001.
11. Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction.* Springer, 2 edition, 2009.
12. Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: with Applications in R.* Springer, 2013.
13. Warren M. Lord, Jie Sun, Nicholas T. Ouellette, and Erik M. Bollt. Inference of causal information flow in collective animal behavior. *IEEE Transactions on Molecular, Biological and Multi-Scale Communications*, 2(1):107–116, 2016.
14. David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms.* Copyright Cambridge University Press, 2003.
15. Qi Mao, Le Yang, Li Wang, Steve Goodison, and Yijun Sun. *SimplePPT: A Simple Principal Tree Algorithm*, pages 792–800. 2015.
16. Pedicini Marco, Fredrik Barrenäs, Trevor Clancy, Filippo Castiglione, Eivind Hovig, Kartiek Kanduri, Daniele Santoni, and Mikael Benson. Combining network modeling and gene expression microarray analysis to explore the dynamics of th1 and th2 cell regulation. *PLOS Computational Biology*, 6(12):1–8, 12 2010.
17. Adam A Margolin, Ilya Nemenman, Katia Basso, Chris Wiggins, Gustavo Stolovitzky, Riccardo Dalla Favera, and Andrea Califano. ARACNE: An algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics*, 7(S1), mar 2006.
18. Hirotaka Matsumoto, Hisanori Kiryu, Chikara Furusawa, Minoru S H Ko, Shigeru B H Ko, Norio Gouda, Tetsutaro Hayashi, and Itoshi Nikaido. SCODE: an efficient regulatory network inference algorithm from single-cell RNA-Seq during differentiation. *Bioinformatics*, 33(15):2314–2321, 04 2017.
19. Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2020.
20. Jacques Monod and François Jacob. General conclusions: teleonomic mechanisms in cellular metabolism, growth, and differentiation. In *Cold Spring Harbor symposia on quantitative biology*, volume 26, pages 389–401. Cold Spring Harbor Laboratory Press, 1961.

21. Nan Papili Gao, S M Minhaz Ud-Dean, Olivier Gandrillon, and Rudiyanto Gunawan. SINCERITIES: inferring gene regulatory networks from time-stamped single cell transcriptional expression profiles. *Bioinformatics*, 34(2):258–266, 09 2017.
22. Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.
23. Aditya Pratapa, Amogh P Jalihal, Jeffrey N Law, Aditya Bharadwaj, and T M Murali. Benchmarking algorithms for gene regulatory network inference from single-cell transcriptomic data. *Nat Methods*, 17(2):147–154, January 2020.
24. Xiaojie Qiu, Qi Mao, Ying Tang, Li Wang, Raghav Chawla, Hannah A Pliner, and Cole Trapnell. Reversed graph embedding resolves complex single-cell trajectories. *Nat Methods*, 14(10):979–982, August 2017.
25. Jie Sun, Abd AlRahman R. AlMomani, and Erik Bollt. Data-driven learning of boolean networks and functions by optimal causation entropy principle. *Patterns*, 3(11):100631, 2022.
26. Jie Sun, Dane Taylor, and Erik M. Bollt. Causal network inference by optimal causation entropy. *SIAM Journal on Applied Dynamical Systems*, 14(1):73–106, 2015.
27. Joy A. Thomas Thomas M. Cover. *Differential Entropy*, chapter 8, pages 243–259. John Wiley & Sons, Ltd, 2005.
28. Cole Trapnell, Davide Cacchiarelli, Jonna Grimsby, Prapti Pokharel, Shuqiang Li, Michael Morse, Niall J Lennon, Kenneth J Livak, Tarjei S Mikkelsen, and John L Rinn. The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. *Nature Biotechnology*, 32(4):381–386, April 2014.
29. Laurens van der Maaten and Geoffrey Hinton. Viualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 11 2008.
30. Alan Veliz-Cuba and Brandilyn Stigler. Boolean models can explain bistability in the *lac* operon. *Journal of Computational Biology*, 18(6):783–794, jun 2011.
31. Qing Wang. Multivariate kernel smoothing and its applications. *Journal of the American Statistical Association*, 115(529):486–486, 2020.
32. Paul L Williams and Randall D Beer. Nonnegative decomposition of multivariate information. *arXiv preprint arXiv:1004.2515*, 2010.