UNIVERSIDAD DE GRANADA

MUTUAL INFORMATION IN UNSUPERVISED MACHINE LEARNING

FRANCISCO JAVIER SÁEZ MALDONADO

Bachelor's Thesis

Computer Science and Mathematics

Tutor

Nicolás Pérez de la Blanca Capilla

FACULTY OF SCIENCE

H.T.S. OF COMPUTER ENGINEER AND TELECOMMUNICATIONS

Granada, Thursday 15th July, 2021

LICENSE PAGE

ABSTRACT

RESUMEN

INTRODUCTION

MAIN GOALS AND RESULTS ACHIEVED

CONTENTS

Part I

BASIC CONCEPTS

1 PROBABILITY

Underneath each experiment involving any grade of uncertainty there is a *random variable*. This is no more than a *measurable* function between two *measurable spaces*. A probability space is composed by three elements: $(\Omega, \mathscr{A}, \mathcal{P})$. We will define those concepts one by one.

1.1 BASIC NOTIONS

Definition 1.1.1. Let Ω be a non empty sample space. $\mathscr{A}$ is a *$\sigma-$algebra* over Ω if it is a family of subsets of Ω that verify that the emptyset is in $\mathscr{A}$, and it is closed under complementation and countable unions. That is:

- $\emptyset \in \mathscr{A}$.
- If $A \in \mathscr{A}$, then $\Omega \backslash A \in \mathscr{A}$.
- If $\{A_i\}_{i \in \mathbb{N}} \in A$ is a numerable family of $\mathscr{A}$ subsets, then $\cup_{i \in \mathbb{N}} A_i \in \mathscr{A}$.

The pair $(\Omega, \mathscr{A})$ is called a *measurable space* To get to our probability space, we need to define a *measure* on the *measurable space*.

Definition 1.1.2. Given $(\Omega, \mathscr{A})$ a measurable space, a *measure* $\mathcal{P}$ is a countable additive, non-negative set function on this space. That is: $\mathcal{P} : \mathscr{A} \to \mathbb{R}_0^+$ satisfying:

- $\mathcal{P}(A) \geq \mathcal{P}(\emptyset) = 0$ for all $A \in \mathscr{A}$,
- $P(\cup_n A_n) = \sum_n P(A_n)$ for any countable collection of disjoint sets $A_n \in \mathscr{A}$.

If $\mathcal{P}(\Omega) = 1$, $\mathcal{P}$ is a *probability measure* or simply a *probability*. With the concepts that have just been explained, we get to the following definition:

Definition 1.1.3. A *measure space* is the tuple $(\Omega, \mathscr{A}, \mathcal{P})$ where $\mathcal{P}$ is a *measure* on $(\Omega, \mathscr{A})$. If $\mathcal{P}$ is a *probability measure* $(\Omega, \mathscr{A}, \mathcal{P})$ will be called a *probability space*.

Throughout this work, we will be always in the case where $\mathcal{P}$ is a probability measure, so we will always be talking about probability spaces and we will note $\mathcal{P}$ simply as P. Some notation for these measures must be introduced. Let A and B be two events. The notation $P(A, B)$ reffers to the probability of the intersection of the events A and B, that is: $P(A, B) := P(A \cap B)$. It is clear that since $A \cap B = B \cap A$, then $P(A, B) = P(B, A)$. We remark the next definition since it will be important.

Definition 1.1.4. Let A, B be two events in Ω. The *conditional probability* of B given A is defined as:

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

There is an alternative way to state the definition that we have just made.

Theorem 1.1.1 (Bayes' Theorem)**.** *Let A, B be two events in Ω, given that $P(B) \neq 0$. Then*

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

Proof. Straight from the definition of the conditional probability we obtain that:

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

We also see from the definition that

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

Hence, since $P(A, B) = P(B, A)$,

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

□

However, events might not give any information about another event occurring. When this happens, we call those events to be *independent*. Mathematically, if A,B are independent events:

$$P(A, B) = P(A)P(B)$$

and as a consequence of this, the conditional probabilty of those events is $P(A|B) = P(A)$. For a finite set of events $\{A_i\}_{i=1}^n$, we say that they are mutually independent if, and only if, every event is independet of any intersection of the other events. That is, if $\{B_i\} \subset \{A_i\}$, then

$$P\left(\cap_{i=1}^k B_i\right) = \prod_{i=1}^k P(B_i) \qquad \text{for all } k \leq n.$$

Random variables (RV) can now be introduced. Their first property is that they are measurable functions. This kind of functions are defined as it follows:

Definition 1.1.5. Let $(\Omega_1, \mathscr{A}), (\Omega_2, \mathcal{B})$ be measurable spaces. A function $f : \Omega_1 \to \Omega_2$ is said to be *measurable* if, $f^{-1}(B) \in \mathscr{A}$ for every $B \in \mathcal{B}$.

As a quick note, we can affirm that if f, g are real-valued measurable functions, and $k \in \mathbb{R}$, it is true that kf, $f + g$, fg and f/g (if g is not the identically zero function) are also *measurable functions*.

We are now ready to define one of the concepts that will lead us to the main objective of this thesis.

Definition 1.1.6. Let $(\Omega, \mathscr{A}, \mathcal{P})$ be a probability space, and $(E, \mathcal{B})$ be a measurable space. A *random variable* is a measurable function $X : \Omega \to E$, from the probability space to the measurable space. This means: for every subset $B \in (E, \mathcal{B})$, its preimage

$$X^{-1}(B) = \{\omega : X(\omega) \in B\} \in \mathscr{A}.$$

Using that sums, products and quotients of measurable functions are measurable functions, we obtain that *sums, products and quotients of random variables are random variables*.

Let now X be a R.V. The *probability* of X taking a concrete value on a measurable set contained in E, say, $S \in E$, is written as:

$$P_X(S) = P(X \in S) = P(\{a \in \Omega : X(a) \in S\}).$$

We use the notation P_X to remark that P is related to the distribution of X. Most of the times, when it is clear from the context, the subindex will be omitted and we will simply refer to it $p(x)$. A very simple example of random variable is the following:

Example 1. Consider tossing a coin. The possible outcomes of this experiment are *Heads or Tails*. Those are our random events. We can give our random events a possible value. For instance, let *Heads* be 1 and *Tails* be 0. Then, our random variable looks like this:

$$X = \begin{cases} 1, & \text{if we obtain heads,} \\ 0, & \text{if we obtain tails.} \end{cases}$$

In the last example, our random variable is *discrete*, since the set $\{X(\omega) : \omega \in \Omega\}$ is finite. A random variable can also be *continuous*, if it can take any value within an interval.

1.2 EXPECTATION OF A RANDOM VARIABLE

Definition 1.2.1. The *cumulative distribution function* F_X of a real-valued random variable X is its probability of taking value below or equal to x. That is:

$$F_X(x) = P(X \leq x) = P(\{\omega : X(\omega) \leq x\}) = P_X((-\infty, x]) \quad \text{for all } x \in \mathbb{R}.$$

We can difference between certain types of random variables. If the image, $\mathcal{X}$, of X is countable, we call it a *discrete* random variable. Its *probability mass function p* gives the probability of the R.V. being equal to a certain value:

$$p(x) = P(X = x).$$

If the cumulative distribution function of our random variable X is continuous everywhere, then X is a *continuous* random variable. In this case there might exist a non-negative Lebesgue-integrable function f such that:

$$F_X(x) = \int_{\infty}^{x} f(t)dt,$$

called the *probability density function* of X.

During this document, distributions $p(x)$ will be adressed many times. We will use $p(x)$ to denote the probability mass function of a discrete random variable or the probability density function of a continuous random variable.

We are now ready to introduce the *expectation* of a random variable. Imagine observing a wide number of outcomes from our random variable, and taking the average of these random values. The expectation is the value of this average when we take *infinite* outcomes of our random variable.

> **Definition 1.2.2.** Let X be a non negative random variable on a probability space $(\Omega, \mathscr{A}, \mathcal{P})$. The *expectation* $E[X]$ of X is defined as:
>
> $$E[X] = \int_\Omega X(\omega)\, dP(\omega).$$

Sometimes we might be referring to multiple random variables. In these cases, in order to make reference to the variable (or distribution function, that will be presented later) for which we calculate the expectation, we will denote it as E_X (or E_P, in the case that we are adressing a distribution).

The expectation of a random variable will be also denoted as μ. Now, if X is generic $R.V$, the expectation is defined as:

$$E[X] = E[X^+] - E[X^-],$$

where X^+, X^- are defined as it follows:

$$X^+(\omega) = \max(X(\omega), 0), \qquad X^-(\omega) = \min(X(\omega), 0).$$

The expectation $E[X]$ of a random variable is a linear operation. That is, if Y is another random variable, and $\alpha, \beta \in \mathbb{R}$, then

$$E[\alpha X + \beta \mathcal{Y}] = \alpha E[X] + \beta E[\mathcal{Y}].$$

This is a trivial consequence of the linearity of the *Lebesgue integral*.

As a note, if X is a *discrete* random variable and $\mathcal{X}$ is its image, its expectation can be computed as:

$$E[X] = \sum_{x \in \mathcal{X}} x P_X(x),$$

where x is each possible outcome of the experiment, and $P_X(x)$ the probability under the distribution of X of the outcome x. The expression given in Def. 1.2.2 generalizes this particular case.

Using the definition of the expectation of a random variable, we can approach to the concept of the *moments* of a random variable.

> **Definition 1.2.3.** If $k \in \mathbb{N}$, then $E[X^k]$ is called the $k-th$ moment of X.

If we take $k = 1$, we have the definition of the *expectation*. It is sometimes written as $m_X = E[X]$, and called the *mean*. We use the *mean* in the definition of the variance:

Definition 1.2.4. Let X be a random variable. If $E\left[X^2\right] < \infty$, then the *variance* of X is defined to be

$$\operatorname{Var}(X) = E\left[(X - m_X)^2\right] = E\left[X^2\right] - m_X^2.$$

Thanks to the linearity of the *expectation* of a random variable, it is easy to see that, if $a, b \in \mathbb{R}$, then

$$\operatorname{Var}(aX + b) = E[(aX + b) - E[aX + b])^2] = a^2 E[(X - m_X)^2] = a^2 \operatorname{Var}(X).$$

Usually, when it comes to applying these concepts to a real problem, we will be observing multiple features that a phenomenon in nature presents. We would like to have a collection of random variables each one representing one of this features. In order to set the notation for these kinds of situations, we will introduce *random vectors*.

Definition 1.2.5. A random vector is a row vector $\mathbf{X} = (X_1, \ldots, X_n)$ whose components are real-valued random variables on the same probability space $(\Omega, \mathscr{A}, P)$.

The probability distribution of a random variable can be extended in to the *joint probability distribution* of a random vector.

Definition 1.2.6. Let $\mathbf{X} = (X_1, \ldots, X_n)$ be a random vector. The *cumulative distribution function* (or simply, the *distribution function*) $F_{\mathbf{X}} : \mathbb{R}^n \to [0, 1]$ of $\mathbf{X}$ is defined as:

$$F_{\mathbf{X}}(x) = P(X_1 \leq x_1, \ldots, X_n \leq x_n).$$

We also name it *multivariate distribution*. Before, we presented the concept of independence between a pair of events. Using the cumulative distribution function, we can now define the independence between random variables.

Definition 1.2.7. A finite set of n random variables $\{X_1, \ldots, X_n\}$ is mutually independent if, and only if, for any sequence $\{x_1, \ldots, x_n\}$, the events $\{X_1 \leq x_1\}, \ldots, \{X_n \leq x_n\}$ are mutually independent. Equivalently, this finite set is mutually independent if, and only if,:

$$F_{X_1,\ldots,X_n}(x_1, \ldots, x_n) = F_{X_1}(x_1) \ldots F_{X_n}(x_n), \qquad \text{for all } x_1, \ldots, x_n.$$

We can also extend the notion of expectation to a random vector. Let $\mathbf{X} = (X_1, \ldots, X_n)$ be a random vector and assume that $E[X_i]$ exists for all $i \in \{1, \ldots, n\}$. The expectation of $\mathbf{X}$ is defined as the vector containing the expectations of each individual random vector, that is:

$$E[\mathbf{X}] = \begin{bmatrix} E[X_1] \\ \vdots \\ E[X_n] \end{bmatrix}.$$

To generalize the variance of a random variable, we have to build the following matrix.

Definition 1.2.8. Let $\mathbf{X} = (X_1, \ldots, X_n)$ be a random vector. Then, the *covariance matrix* of $\mathbf{X}$ is defined as:

$$\Sigma = \mathrm{Cov}(\mathbf{X}) = E[(\mathbf{X} - \mu_{\mathbf{X}})(\mathbf{X} - \mu_{\mathbf{X}})^T] = \begin{pmatrix} \sigma_{11} & \cdots & \sigma_{1n} \\ \vdots & \ddots & \vdots \\ \sigma_{n1} & \cdots & \sigma_{nn} \end{pmatrix},$$

where $\sigma_{ij} = \mathrm{Cov}(X_i, X_j) = E[(X_i - \mu_i)(X_j - \mu_j)] = \sigma_{ji}$.

It can also happen that, given a random vector, we would like to know the probability distribution of some of its components. That is called the *marginal distribution*.

Definition 1.2.9. Let $\mathbf{X} = (X_1, \ldots, X_n)$ be a random vector. The *marginal distribution* of a subset of $\mathbf{X}$ is the probability distribution of the variables contained in the subset.

In the simple case of having two random variables, e.g. $X = (X_1, X_2)$, then the marginal distribution of X_1 is:

$$P(x) = \int_{x_2} P(x_1, x_2) dx_2.$$

2 DISTRIBUTIONS

We have introduced the concepts of *random variable*, *random vector* and its *probability distribution*. We will explain some concepts related to the latter ones.

Definition 2.0.1. The *mode* of a distribution is the value at which the probability mass function takes its maximum value. That is, the value that is most likely to be sampled.

Distributions can be *unimodal*, when their distribution has a single peak, *bimodal* when their distribution has two peaks, and *multimodal* when the number of peaks is equal or greater to 2.

Example 2. We can simulate the two following distributions:

1. The distribution of the marks obtained in a test by the students of certain class.
2. The distribution of the height of the plants from three different species.

The result is the following:

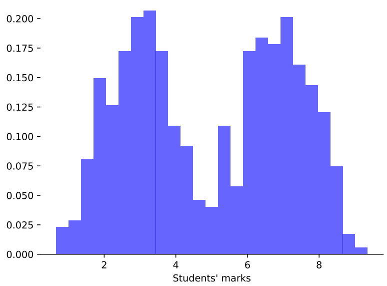

Figure 1: Bimodal Distribution

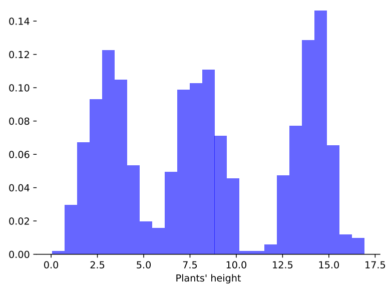

Figure 2: Plants' height

Figure 3: Examples of bimodal and multimodal distributions.

Now, given two distributions, we would like to determine how different they are from each other. In order to compare them, we enunciate the definition of the Kullback-Leibler divergence.

Definition 2.0.2. Let P and Q be probability distributions over the same probability space Ω. Then, the Kullback-Leibler divergence is defined as:

$$D_{KL}(P \,||\, Q) = E_P \left[\log \frac{P(x)}{Q(x)} \right].$$

It is defined if, and only if, P is *absolutely continuous with respect to* Q, that is, if $P(A) = 0$ for any A subset of Ω where $Q(A) = 0$. There are some properties of this definition that must be stated.

> **Proposition 1.** *If P, Q are two probability distributions over the same probability space, then $D_{KL}(P|Q) \geq 0$.*

Proof. Firstly, note that if $a \in \mathbb{R}^+$, then $\log\ a \leq a - 1$. Then:

$$\begin{aligned} -D_{KL}(P \mid\mid Q) &= -E_P\left[\log \frac{P(x)}{Q(x)}\right] \\ &= E_P\left[\log \frac{Q(x)}{P(x)}\right] \\ &\leq E_P\left[\left(\frac{Q(x)}{P(x)} - 1\right)\right] \\ &= \int P(x) \frac{Q(x)}{P(x)}\, dx - 1 \\ &= 0. \end{aligned}$$

So we have obtained that $-D_{KL}(P \mid\mid Q) \leq 0$, which implies that $D_{KL}(P \mid\mid Q) \geq 0$. $\square$

As a corollary of this proposition, we can affirm that $D_{KL}(P \mid\mid Q)$ equals zero if and only if $P = Q$ almost everywhere. We will also remark the discrete case, as it will be used later. Let P, Q be discrete probability distributions defined on the same probability space Ω. Then,

$$D_{KL}(P \mid\mid Q) = \sum_{x \in \Omega} P(x) \log\left(\frac{P(x)}{Q(x)}\right).$$

2.1 EXAMPLES OF DISTRIBUTIONS

Let us present some examples o common distributions. They will be used further in this document.

Bernoulli

Think for a moment that you want to model the possible outcomes of an experiment with two possibilites: sucess or failure. Imagine also that you already know that in your experiment there is a probability p of achieving success. That is the intuitive idea of a Bernoulli distribution. We can define it more formally as follows:

The *Bernoulli distribution* is a discrete probability distribution of a random variable that takes two values, $\{0, 1\}$, with probabilities p and $q = 1 - p$, respectively. We will say that our distribution is a $Bern(p)$.

If k is a possible outcome, we can define the probability mass function f of a Bernoulli distribution as:

$$f(k,p) = \begin{cases} p, & \text{if } k = 1, \\ 1-p, & \text{if } k = 0. \end{cases}$$

Using the expression of the mean for discrete random variables, we obtain that $E[X] = p$ and

$$\mathrm{Var}[X] = E[X^2] - E[X]^2 = E[X] - E[X]^2 = p - p^2 = p(1-p) = pq.$$

As a note, this is just a particular case of the *Binomial distribution* with $n = 1$.

Gaussian Distribution

The Gaussian (or normal) distribution is used to represent real-valued random variables whose distributions are not known. Its importance relies in the fact that, using the *central limit theorem*, we can assume that the average of many samples of a random variable with finite mean and variance is a random variable whose distribution converges to a normal distribution as the number of samples increases.

> **Definition 2.1.1.** We say that the real valued random variable X follows a *normal distribution* of parameters $\mu, \sigma \in \mathbb{R}$ if, and only if, its probability density function exists and it is determined by
>
> $$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2},$$
>
> where μ is the mean and σ is its standard deviation. We denote this normal distribution as $X \sim \mathcal{N}(\mu, \sigma)$.

The particular case where $\mu = 0$ and $\sigma = 1$ is widely used in statistics. In this case, the density function is simpler:

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

A remarkable property of these distributions is that, if $f : \mathbb{R} \to \mathbb{R}$ is a real-valued function defined as $f(x) = ax + b$, then $f(X) \sim \mathcal{N}(a\mu + b, |a|\,\sigma)$.

In the same way that we extended random variables to random vectors, we can extend the normal distribution to a multivariate random distribution.

> **Definition 2.1.2.** We say that a random vector $\boldsymbol{X} = (X_1, \ldots, X_n)$ follows a multivariate normal distributions of parameters $\mu \in \mathbb{R}^n$, $\Sigma \in \mathcal{M}_N(\mathbb{R})$ if, and only if, its probability density function is:
>
> $$f(x) = \frac{1}{\sqrt{\det(2\pi\Sigma)}} e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}.$$
>
> It is denoted $X \sim \mathcal{N}(\mu, \Sigma)$. In this case, μ is the mean vector of the distribution and Σ denotes the covariance matrix.

3 STATISTICAL INFERENCE

Statistical inference is the process of deducing properties of an underlying distribution by analyzing the data that it is available. With this purpose, techniques like deriving estimates and testing hypotheses are used.

Inferential statistics are usually contrasted with descriptive statistics, which are only concerned with properties of the observed data. The difference between these two is that in inferential statistics, we assume that the data comes from a larger population that we would like to know.

In *machine learning*, subject that concerns us the most, the term inference is sometimes used to mean *make a prediction by evaluating an already trained model*, and in this context, inferring properties of the model is refered as *training or learning*.

3.1 PARAMETRIC MODELING

In the following chapters, we will be trying to estimate density functions in a dataset. To do this we will be using *parametric models*. We say that a *parametric model*, $P_\theta(x)$, is a family of density functions that can be described using a finite numbers of parameters θ. We can get to the concept of *log-likelihood* now.

> **Definition 3.1.1.** The *likelihood* $\mathcal{L}(\theta|x)$ of a parameter set θ is a function that measures how plausible is θ, given an observed point x in the dataset $\mathcal{D}$. It is defined as the value of the density function parametrized by θ at x. That is:
> $$\mathcal{L}(\theta|x) = P_\theta(x).$$

In a finite dataset $\mathcal{D}$ consisting of independent observations, we can write:

$$\mathcal{L}(\theta|X) = \prod_{x \in D} P_\theta(x).$$

This can be computationally hard to work with, so the log-likelihood is often used instead.

> **Definition 3.1.2.** Let $\mathcal{D}$ be a dataset of independent observations and θ a set of parameters. Then, we define the *log-likelihood* ℓ as the sum of the logarithms of the evaluations of p_θ in each x in the dataset. That is:
> $$\ell(\theta|X) = \sum_{x \in \mathcal{D}} \log P_\theta(x).$$

Our goal would be to find the optimal value $\hat{\theta}$ that maximizes the likelihood of observing the dataset $\mathcal{D}$. We get to the following definition:

Definition 3.1.3. We say that $\hat{\theta} = \hat{\theta}(\mathcal{D})$ is a *maximum likelihood estimator*(MLE) for θ if

$$\hat{\theta} \in \arg\max_{\theta} \mathcal{L}(\theta|\mathcal{D})$$

for every observation $\mathcal{D}$.

3.2 MINIMAL SUFFICIENT STATISTICS

In parametric modeling, the goal was to determine the density function under a distribution. Another interesting task can be determining specific parameters or quantitys related to a distribution, given a sample $X = (x_1, \cdots, x_n)$.

Definition 3.2.1. A *statistic* is a measurable function of the data. That is, if $T : \Omega \to \mathbb{T}$ is measurable, then $T(X)$ is a statistic.

Remark 1. A statistic is also a random variable.

However, not all statistics will provide useful information for the statistical inference problem. We would like to find statistics that provide relevant information.

Definition 3.2.2. Let $X \sim P_\theta$. Then, the statistic $T(X) = T : (\Omega, \mathscr{A}) \to (\mathbb{T}, \mathcal{B})$, is sufficient for a family of parameters $\{P_\theta \ : \ \theta \in \Theta\}$ if the conditional distribution of X, given $T = t$, is indepentent of θ.

Example 3. The simplest example of a sufficient statistic is the mean μ of a gaussian distribution with known variance. Oppositely, the *median* of an arbitrary distribution is not sufficient for the mean since, even if the median of the sample is known, more information about the mean of the population can be obtained from the mean of the sample itself.

Although it will not be shown in this document, sufficient statistics are not unique. In fact, if T is sufficient, $\psi(T)$ is sufficient for any bijective mapping ψ. It would be interesting to find a sufficient statistic T that is *the smallest* of them.

Definition 3.2.3. A sufficient statistic T is minimal if, for every sufficient statistic U, there exists a mapping f such that $T(x) = f(U(x))$ for any $x \in \Omega$.

4 NOISE CONTRASTIVE ESTIMATION

Our problem now is, to estimate the density function (p.d.f.) of some observed data.

A sample $X = \{x_1, \ldots, x_{T_d}\}$ of a random vector is observed. It follows an unknown p.d.f. P_d. We assume that the data p.d.f. belongs to a parametrized family of functions, that is

$$P_d \in \{P_m(.;\theta)\}_\theta,$$

where θ is a vector of parameters. This means that, in fact,

$$P_d(.) = P_m(.;\theta^*) \quad \text{for some } \theta^*,$$

so our problem is to find θ^*.

Any estimate $\hat{\theta}$ must meet the constraints that a normalized p.d.f. should satisfy, that is:

$$\int P_m(u;\hat{\theta})du = 1, \qquad P_m(.;\hat{\theta}) \geq 0.$$

If the constraints are satisfied for any θ in the set of parameters, we say that the model is normalized, and then we can use the maximum likelihood principle to estimate θ.

Let us assume that the noisy data Y is an i.i.d. sample $\{y_1, \ldots, Y_{T_n}\}$ of a random variable with p.d.f. P_n. The ratio P_d/P_n of the density functions that generate X and Y respectively, can give us a relative description of the data X. If P_n is known, then we can obtain P_d using the ratio that we have just mentioned.

In order to discriminate between elements of X and Y, it is needed to compare their properties. We will show that we can provide a relative description of X in the form of an estimate of the ratio P_d/P_n.

Let $U = \{u_1, \cdots, u_{T_d+T_n}\}$ be the union of the sets X and Y. We assign to each u_t a binary class label:

$$C_t(u_t) = \begin{cases} 1 & if\ u_t \in X \\ 0 & if\ u_t \in Y \end{cases}$$

We will now make use of logistic regression, where the posterior probabilities of the classes given the data are estimated. We know that P_d is unknown, we want to model $P(.|C = 1)$ with $P_m(.;\theta)$. Note that θ may include a parameter for the normalization of the model, if it is not normalized. Hence, we have:

$$P(u|C = 1, \theta) = P_m(u;\theta), \qquad P(u|C = 0) = P_n(u),$$

with

$$P(C=1) = \frac{T_d}{T_d + T_n}, \qquad P(C=0) = \frac{T_n}{T_d + T_n}.$$

Hence, if $\nu = T_n/T_d$, the posterior probabilities for the classes are:

$$P(C=1|u;\theta) = \frac{P_m(u;\theta)}{P_m(u;\theta) + \nu P_n(u)}, \qquad P(C=0|u;\theta) = \frac{\nu P_n(u)}{P_m(u;\theta) + \nu P_n(u)}.$$

Denote $G(.;\theta)$ to the log ratio between $P_m(.;\theta)$ and P_n:

$$G(u;\theta) = \log P_m(u;\theta) - \log P_n(u) = \log \frac{P_m(u;\theta)}{P_n(u)}. \tag{1}$$

Also, let r_ν the logistic function parametrized by ν, that is:

$$r_\nu(u) = \frac{1}{1 + \nu exp(-u)}. \tag{2}$$

Using 1 and 2, we can write

$$h(u;\theta) := P(C=1|u;\theta) = r_\nu(G(u;\theta)) = \frac{1}{1 + \nu exp(\log \frac{P_m(u,\theta)}{P_n(u)})}.$$

Since the class labels C_t are assumed Bernoulli distributed and independent, the conditional log-likelyhood has the form:

$$\ell(\theta) = \sum_{t=1}^{T_d+T_n} C_t \log P(C_t = 1|u_t;\theta) + (1 - C_t) \log P(C_t = 0|u_t;\theta). \tag{3}$$

Now, in the t such that u_t in X,then $u_t = x_t$ an we have that $P(C_t = 0|x_t;\theta) = 0$, so we obtain that the term that adds to the sum in that certain t is:

$$1 \cdot \log P(C_t = 1|u_t;\theta) = \log h(x_t;\theta).$$

Using the same argument for t such that $u_t \in Y$, we obtain the following form of the log-likelyhood in **??**:

$$\ell(\theta) = \sum_{t=1}^{T_d} \log[h(x_t;\theta)] + \sum_{t=1}^{T_n} \log[1 - h(y_t,\theta)]. \tag{4}$$

Now, optimizing $\ell(\theta)$ with respecto to θ leads to an estimate $G(.;\hat{\theta})$ of the log-ratio $\log(P_d/P_n)$, so we get an approximate description of X relative to Y by optimizing **??**.

Remark 2. If we consider $-\ell(\theta)$, this is known as the *cross entropy function*.

Remark 3. Here, we have achieved the estimation of a p.d.f. , which is an unsupervised (not labeled data) learning problem, logistic regression, which is supervised learning (labeled data).

Now, if we consider $P_m^0(.;\alpha)$ an unnormalized (doest not integrate 1) model, we can add a normalization parameter to it in order to normalize it. We can consider

$$\log P_m(.;\theta) = \log P_m^0(.;\alpha) + c, \qquad \text{with } \theta = (\alpha, c).$$

With this model, a new estimator is defined. Considering X as before and Y an artificially generated set with $T_n = \nu T_d$ independent obvservations extracted from P_n, known. as the argument $\hat{\theta}_T$ which maximizes

$$J_T(\theta) = \frac{1}{T_d}\{\sum_{t=1}^{T_d} \log[h(x_t;\theta)] + \sum_{t=1}^{T_n} \log[1 - h(y_t;\theta)]\}.$$

We have to remark that in this case, we have fixed ν before T_n, so T_n will increase as T_d increases. Now, using the weak law of large numbers, $J_T(\theta) \to J$ in probability, where

$$J(\theta) = E\{\log[h(x;\theta)]\} + \nu E\log[1 - h(y;\theta)].$$

Let us rename some terms before announcing a theorem. We want to see J as a function of $\log P_m(.;\theta)$ instead of only θ. In order to do this, let $f_m(.) = \log P_m(.;\theta)$, and consider

$$\tilde{J}(f_m) = E\{\log[r_\nu(f_m(x) - \log P_n(x))]\} + \nu E\{\log[1 - r_\nu(f_m(y) - \log P_n(y))]\}.$$

The following theorem states that the probability density function P_d of the data can be found by maximizing $\tilde{J}$, that is, learning a nonparametric classifier in *infinite data*.

> **Theorem 4.0.1.** *The objective $\tilde{J}(f_m)$ achieves a maximum at $f_m = \log P_d$. Furthermore, there are not other extrema if the noise density P_n is chosen such that it is nonzero whenever P_d is nonzero.*

Part II

INFORMATION THEORY

5 ENTROPY

Intuitively, if x is a datapoint in a dataset $\mathcal{D} \subset \mathbb{R}^d$, a *representation* of x is a vector $r \in \mathbb{R}^n$ (usually, $n \leq d$), that shares information with the datapoint x. Representations are very often used in Machine Learning.

Obtaining good representations of data is one of the most important tasks in machine learning (ML). Recently, it has been discovered that maximizing *mutual information* between two elements in our data can give us good representations for our data. In this section, *information theory* notions will be presented, in order to use them in our ML models. This will provide a theoretical solid base for the notions explained later.

The *mutual information* concept is based on the *Shannon entropy*, which we will introduce first, along with some basic properties of it. The Shannon entropy is a way of measuring the uncertainty in a random variable. Given an event $\mathcal{A} \in \mathscr{A}$, P a probability measure and $P[\mathcal{A}]$ the probability of $\mathcal{A}$, we can affirm that

$$\log \frac{1}{P[\mathcal{A}]}$$

describes *"how surprising is that $\mathcal{A}$ occurs"*. For instance, if $P[\mathcal{A}] = 1$, then the last expression is zero, which means that it is not a surprise that $\mathcal{A}$ occurred. With this motivation, we get to the following definition.

Definition 5.0.1. Let X be a discrete random variable with image $\mathcal{X}$. The *Shannon entropy*, or simply *entropy* $H(X)$ of X is defined as:

$$H(X) = E_X\left[\log \frac{1}{P_X(X)}\right] = \sum_{x \in \mathcal{X}} P_X(x) \log \frac{1}{P_X(x)}.$$

The *entropy* can trivially be expressed as:

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

This simple example, (Cover & Thomas, 1991), even though it is very simple, it is very illustrative for our definition:

Example 4. Let $X \sim Bern(p)$. Then, the entropy of X is:

$$H(X) = -p \log p - (1-p)\log(1-p)) = H(p),$$

since H only depends on p. In Fig. ?? we can see a representation of this function. We appreciate that in this case, H is concave and equals 0 if $p \in \{0, 1\}$, which are the values of p that give us no uncertainty. The maximum uncertainty is obtained when $p = \frac{1}{2}$, where we do not know what to expect as an outcome from our random variable X.

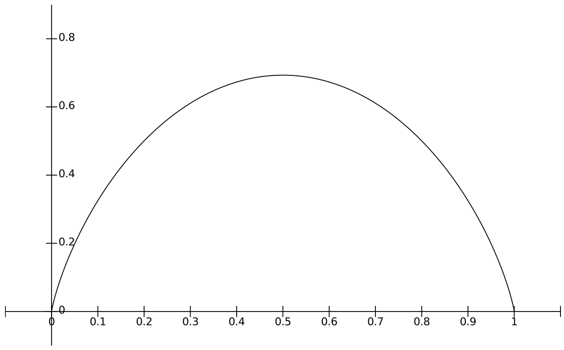

Figure 4: Representation of $H(p)$ in the example 4.

It can also be proven that, in general, the entropy is concave. There are some properties of the *entropy* that must be remarked.

> **Proposition 2.** *Let X be a random variable with image $\mathcal{X}$. If $|\mathcal{X}|$ is the cardinal of $\mathcal{X}$, then*
> $$0 \leq H(X) \leq \log(|\mathcal{X}|).$$

Proof. Since $\log y$ is concave on $\mathbb{R}^+$, by Jensen's inequality (see Appendix 11, Prop. 8), we obtain:

$$H(X) = -\sum_{x\in\mathcal{X}} P_X(x)\log P_X(x) \leq \log\left(\sum_{x\in\mathcal{X}} 1\right) = \log(|\mathcal{X}|).$$

For the lower bound we see that, since $P_X(x) \in [0,1]$ for all $x \in \mathcal{X}$ then $\log P_X(x) \leq 0 \quad \forall x \in \mathcal{X}$. Hence , $-P_X(x)\log P_X(x) \geq 0$ for all $x \in X$, so $H(X) \geq 0$. □

We can also see that the equality on the left holds if , and only if , exists x in X such that its probability is exactly one, that is $P_X(x) = 1$. The right equality holds if and only if , for all $x \in \mathcal{X}$, its probability is $P_X(x) = \frac{1}{|X|}$.

Conditional entropy

We have already said that entropy measures how surprising is that an event occurs. Usually, we will be looking at two random variables and it would be interesting to see how likely is that one of them, say $X(x)$, occurred, if we already know that $Y(y)$ occurred. This leads us to the definition of *conditional entropy*. Let us see a simpler case first:

Let A be an event, and X a random variable. The conditional probability $P_{X|A}$ defines the entropy of X conditioned to A:

$$H(X|A) = \sum_{x\in\mathcal{X}} P_{X|A}(x)\log\frac{1}{P_{X|A}(x)}.$$

If Y is another random variable and $\mathcal{Y}$ is its image, intuitively we can sum the conditional entropy of an event with all the events in $\mathcal{Y}$, and this way we obtain the conditional entropy of X given Y.

Definition 5.0.2 (Conditional Entropy)**.** Let X, Y be random variables with images $\mathcal{X}, \mathcal{Y}$. The *conditional entropy* $H(X|Y)$ is defined as:

$$\begin{aligned} H(X|Y) &:= \sum_{y \in \mathcal{Y}} P_{\mathcal{Y}}(y) H(X|Y=y) \\ &= \sum_{y \in \mathcal{Y}} P_{\mathcal{Y}}(y) \sum_{x \in \mathcal{X}} P_{X|Y}(x|y) \log \frac{1}{P_{X|Y}(x|y)} \\ &= \sum_{x \in X, y \in \mathcal{Y}} P_{XY}(x,y) \log \frac{P_Y(y)}{P_{XY}(x,y)}. \end{aligned}$$

The interpretation of the conditional entropy is simple: the uncertainty in X when Y is given. Since we know about an event that has occurred (Y), intuitively the conditional entropy , or the uncertainty of X occurring given that Y has occurred, will be lesser than the entropy of X, since we already have some information about what is happening. We can prove this:

Proposition 3. *Let X, Y be random variables with images $\mathcal{X}, \mathcal{Y}$. Then:*

$$0 \leq H(X|Y) \leq H(X).$$

Proof. The inequality on the left was proved on Proposition 2. The characterization of when $H(X|Y) = 0$ was also mentioned after it. Let us look at the inequality on the right. Note that restricting to the (x, y) where $P_{XY}(x,y) > 0$ and using the definition of the conditional probability we have:

$$\begin{aligned} H(X|Y) &= \sum_{y \in \mathcal{Y}} P_Y(y) \sum_{x \in \mathcal{X}} P_{X|Y}(x|y) \log \frac{1}{P_{X|Y}(x|y)} \\ &= \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} P_Y(y) P_{X|Y}(x,y) \log \frac{P_Y(y)}{P_{XY}(x,y)} \\ &= \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} P_{XY}(x,y) \log \frac{P_Y(y)}{P_{XY}(x,y)}, \end{aligned}$$

and

$$H(X) = \sum_x P_X(x) \log \frac{1}{P_X(x)} = \sum_{x,y} P_{XY}(x,y) \log \frac{1}{P_X(x)}.$$

Hence,

$$\begin{aligned} H(X|Y) - H(X) &= \sum_{x,y} P_{XY}(x,y) \left(\log \frac{P_Y(y)}{P_{XY}(x,y)} - \log \frac{1}{P_X(x)} \right) \\ &= \sum_{x,y} P_{XY} \log \frac{P_Y(y) P_X(x)}{P_{XY}(x,y)}. \end{aligned} \tag{5}$$

So, using Jensen's inequality, we obtain:

$$\begin{aligned} \sum_{x,y} P_{XY} \log \frac{P_Y(y) P_X(x)}{P_{XY}(x,y)} &\leq \log \left(\sum_{x,y} \frac{\cancel{P_{XY}(x,y)} \; P_Y(y) P_X(x)}{\cancel{P_{XY}(x,y)}} \right) \\ &= \log \left(\left(\sum_x P_X(x) \right) \left(\sum_y P_Y(y) \right) \right) = \log 1 = 0, \end{aligned}$$

and this leads us to:

$$H(X|Y) - H(X) \leq 0 \quad \text{then} \quad H(X|Y) \leq H(X) \tag{6}$$

as we wanted. $\square$

It must be noted that the inequality the state of the proposition,

$$0 \leq H(X|Y) \leq H(X),$$

in the inequality of the left, equality holds if, and only if, $P_{XY}(x,y) = P_X(x)P_Y(y)$ for all (x,y) with $P_{XY}(x,y) > 0$, as it is said in Jensen's inequality. For the inequality on the right, equality holds if and only if $P_{XY}(x,y) = 0$, which implies $P_X(x)P_Y(y) = 0$ for any $x \in \mathcal{X}$, $y \in \mathcal{Y}$. It follows that $H(X|Y) = H(X)$ if and only if $P_{XY}(x,y) = P_X(x)P_Y(y)$ for all $(x,y) \in \mathcal{X} \times \mathcal{Y}$

6 MUTUAL INFORMATION

Using the entropy of a random variable we can directly state the definition of *mutual information* as follows:

Definition 6.0.1. Let X, Z be random variables. The *mutual information (MI)* between X and Z is expressed as the difference between the entropy of X and the conditional entropy of X and Z, that is:

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

Since the entropy of the random variable $H(X)$ explains the uncertainty of X occurring, the intuitive idea of the *MI* is to determine the decrease of uncertainty of X occurring when we already know that Z has occurred. We also have to note that, using the definition of the *entropy* and the same argument that we used to obtain the expression in Eq. 5, we can rewrite the *MI* it follows:

$$\begin{aligned} I(X,Z) &= \sum_{x\in\mathcal{X}} P_X(x)\log\frac{1}{P(x)} - \sum_{x\in\mathcal{X},z\in\mathcal{Z}} P_{XZ}(x,z)\log\frac{P_Z(x)}{P_{XZ}(x,z)} \\ &= \sum_{x,z} P_{XZ}\log\frac{P_{XZ}(x,z)}{P_Z(z)P_X(x)} \end{aligned} \tag{7}$$

and if we compare it to the formula of the KL-Divergence, we obtain:

$$I(X,Z) = \sum_{x,z} P_{XZ}\log\frac{P_{XZ}(x,z)}{P_Z(z)P_X(x)} = D_{KL}(P_{XZ} \,||\, P_X P_Z),$$

so we have obtained an expression of the mutual information using the *Kullback-Leibler* divergence. This provides with the following immediate consequences:

Corollary 1. (i) *Mutual information is non-negative. That is :* $I(X,Z) \geq 0$.
(ii) *If* X, Z *are random variables, then its mutual information equals zero if, and only if, they are independent.*
(iii) *Mutual information is symmetric. That is:* $I(X,Z) = I(Z,X)$.

Proof. (i) This is trivial using Prop 3 and the definition of the mutual information.
(ii) We can use the KL-Divergence formulation to see that since

$$D_{KL}(P_{XZ} \,||\, P_X P_Z) = 0 \implies P_{XZ} = P_X P_Z,$$

almost everywhere then X and Z are independent.
(iii) It is a consequence of the fact that $P_{XZ} = P_{ZX}$ and $P_X P_Z = P_Z P_X$.

□

Later in this document, we will have some sort of random variable X and would like it to maintain the mutual information with itself after being applied a function. The following proposition will be useful:

> **Proposition 4.** *Let X, Z be random variables. Then, $I(X, Z)$ is invariant under homeomorphism.*

Proof. Let $\phi(x)$ be an homeomorphism, i.e., a continuous, monotonic function with $\phi^{-1}(x)$ also continuous and monotonic. Let X be a random variable and Y another one such $y = \phi(x)$ if $x = X(\omega)$ for some $\omega \in \Omega$. Then, if S is a particular subset we have

$$P(Y \in S) = \int_S P_Y(y)dy = \int_{\phi^{-1}(S)} P_X(x)dx \overset{(1)}{=} \int_S P_X(\phi^{-1}(y)) \left|\frac{d\phi^{-1}}{dy}\right| dy,$$

where in (1) we have changed from x to y. Hence,

$$P_Y(y) = P_X(\phi^{-1}(y)) \left|\frac{d\phi^{-1}}{dy}\right|.$$

As a consequence of this, $I(X, Z) = I(\phi(X), Z)$ for any homeomorphism ϕ. By symmetry, the same holds for Z.

□

Remark 4. We can set a connection between the mutual information and sufficient statistics. Let $T(X)$ be a statistic. We say that $T(X)$ is sufficient for θ if its mutual information with θ equals the mutual information between X and θ, that is:

$$I(\theta, X) = I(\theta, T(X)).$$

This means that sufficient statistics preserve mutual information and conversely.

6.1 LOWER BOUNDS ON MUTUAL INFORMATION

Although mutual information seems like a relatively intuitive concept, it is most of the times extremely hard to compute it in real life problems in which the distributions $P(x, z), P(x), P(z)$ are not known.

Example 5. Let x represent an image of size $n \times m$ pixels. Then, the dimension of the single image is $n \cdot m \cdot 3$, for RGB color channels. In these cases, there is no easy way of calculating $P(x)$.

Due to this problem related to the *Curse of Dimensionality*, we can try to compute lower bounds of it that are generally easier to calculate. We will now expose two general lower bounds, and we will focus on a third one that will be explained later in this work.

Variational Lower Bound

Using the expression of the mutual information in terms of entropy, $I(x,z) = H(z) - H(z|x)$, we can give a lower bound on $I(x,z)$ as a function of a probability distribution $Q_\theta(z|x)$.

> **Proposition 5.** *Let X, Z be random variables and $Q_\theta(z|x)$ be an arbitrary probability distribution. Then,*
>
> $$I(x,z) \geq H(z) + E_{P_X}\left[E_{P_{X|Z}}\left[\log Q_\theta(z|x)\right]\right]$$

Proof. Recalling that

$$H(z|x) = -E_{P_{XZ}}\left[\log P(x,z) - \log P(x)\right],$$

and that

$$\begin{aligned}
E_{P(x,z)}\left[\log \frac{P(x,z)}{P(x)}\right] &= \sum_{x,z} P(x,z)\log\frac{P(x,z)}{P(x)} \\
&= \sum_{x,z} P(x)P(z|x)\log P(z|x) = \sum_{x,z} P(x)E_{P(z|x)}[\log P(z|x)] \\
&= E_{P(x)}\left[E_{P(z|x)}[\log P(z|x)]\right],
\end{aligned}$$

we only have to use the definition of the conditional probability to see that:

$$\begin{aligned}
I(x,z) &= H(z) - H(z|x) \\
&= H(z) + E_{P(x,z)} = H(z) + E_{P(x,z)}\left[\log\frac{P(x,z)}{P(x)}\right] \\
&= H(z) + E_{P(x)}\left[E_{P(x|z)}\left[\log P(z|x)\right]\right] \\
&= H(z) + E_{P(x)}\left[E_{P(x|z)}\left[\log\frac{P(z|x)}{Q_\theta(z|x)}\right] + E_{P(z|x)}\left[\log Q_\theta(z|x)\right]\right] \\
&= H(z) + E_{P(x)}\left[\underbrace{D_{KL}(P(z|x)||Q_\theta(z|x))}_{\geq 0} + E_{P(z|x)}\left[\log Q_\theta(z|x)\right]\right] \\
&\geq H(z) + E_{P(x)}\left[E_{P(z|x)}\left[\log Q_\theta(z|x)\right]\right].
\end{aligned}$$

We have taken advantage of the non-negativity of the KL-Divergence. □

Using this bound, and combining this theoretical knowledge with machine learning methods, such as *backpropagation*, we can make Q_θ be a neural network and maximize this lower bound.

Donsker-Varadhan Representation

We can also give a lower bound on the mutual information using its KL-Divergence formulation. Firstly, we have to

Theorem 6.1.1 (Donsker-Varadhan)**.** *The KL divergence admits the following dual representation:*

$$D_{KL}(P||Q) = \sup_T E_P[T] - \log E_Q[e^T],$$

where the supremum is taken over all functions $T : \Omega \to \mathbb{R}$ such that both expectations exist.

Proof. TODO □

Using this representation, we reach this lower bound. Let $\mathcal{F}$ be any class of functions $T : \Omega \to \mathbb{R}$ satisfying the integrability constraints of the theorem. Then,

$$I(P, Q) = D_{KL}(P||Q) \geq \sup_{T \in \mathcal{F}} E_P[T] - \log E_Q[e^T].$$

Contrastive Lower Bound

In chapter 4 we presented Noise Contrastive Estimation, that tried to discriminate between elements of two different sets. One was composed of data, X, and the other one was composed of noise, Y.

Let (x, z) be a data representation drawn from a distribution $P(x, z)$ and x' be some other data drawn from the distribution $P(x)$. Using NCE, we should be able to say that (x, z) was drawn from the distribution $P(x, z)$ (which was P_d in the NCE theory) while (x', z) was drawn from the product of the marginal distributions $P(x)P(z)$ (which was P_n in the explanation of NCE). Let h_θ be a model that helps us to do this discrimination, with parameters θ.

As we did before, we want to estimate the ratio P_d/P_n of the different distributions, in this case the ratio would be $P(x, z)/P(x)P(z)$. Let (x^*, z) be a pair drawn from $P(x, z)$ and $X = \{x^*, x_1, \cdots, x_{N-1}\}$, where the rest of the $N - 1$ points form pairs (x_j, z) drawn from $P(x)P(z)$ the product of the marginal distribution. We can rewrite the loss 4 in a simpler expression:

$$l(\theta) = E_X \left[\log \frac{h_\theta(x^*, z)}{\sum_{x \in X} h_\theta(x, z)} \right]. \tag{8}$$

If we maximize this objective, h_θ learns to discriminate (x^*, z) from (x_j, z) for $1 \leq j < N$ and, thus, we are learning to estimate the ratio $P(x, z)/P(x)P(z)$. Let us see how maximizing $\ell(\theta)$ we are maximizing a lower bound for $I(x, z)$.

Proposition 6. *Let $X = \{x^*, x_1, \cdots, x_{N-1}\}$, where $x^* \sim P(x, z)$ and the rest of them were sampled from $P(x)P(z)$. Then,*

$$I(x, z) \geq \ell(\theta) + \log N$$

Proof. Firstly, using Bayes' rule, $P(x^*, z) = P(x^*|z)P(z)$. Hence, since h_θ estimates $P(x^*, z)/P(x)P(z)$, it also estimates

$$\frac{P(x^*, z)}{P(x)P(z)} = \frac{P(x^*|z)P(z)}{P(x)P(z)} = \frac{P(x^*|z)}{P(x)}.$$

Using the definition of the log-likelyhood that we see in 8, we see that

$$E_X\left[\log\frac{h_\theta(x^*,z)}{\sum_{x\in X}h_\theta(x,z)}\right] = E_X\left[\log\frac{h_\theta(x^*,z)}{h_\theta(x^*,z)+\sum_{j=1}^{N-1}h_\theta(x_j,z)}\right]$$
$$\approx E_X\left[\log\frac{\frac{P(x^*|z)}{P(x)}}{\frac{P(x^*|z)}{P(x)}+\sum_{j=1}^{N-1}\frac{P(x_j|z)}{P(x)}}\right].$$

Now, using that $\log(a) = -log(a^{-1})$,

$$E_X\left[\log\frac{\frac{P(x^*|z)}{P(x)}}{\frac{P(x^*|z)}{P(x)}+\sum_{j=1}^{N-1}\frac{P(x_j|z)}{P(x)}}\right] = E_X\left[-\log\left(\frac{\frac{P(x^*|z)}{P(x)}+\sum_{j=1}^{N-1}\frac{P(x_j|z)}{P(x)}}{\frac{P(x^*|z)}{P(x)}}\right)\right]$$
$$= E_X\left[-\log\left(1+\frac{\sum_{j=1}^{N-1}\frac{P(x_j|z)}{P(x)}}{\frac{P(x^*|z)}{P(x)}}\right)\right]$$
$$= E_X\left[-\log\left(1+\frac{(N-1)E_{X-\{x^*\}}\left[\frac{P(x|z)}{P(x)}\right]}{\frac{P(x^*|z)}{P(x)}}\right)\right]$$

Now, since $E_{X-\{x^*\}}\left[\frac{P(x|z)}{P(x)}\right] = \sum_{x_j\in X-\{x^*\}}P(x_j)\frac{P(x_j|z)}{P(x_j)} = 1$, then

$$E_X\left[-\log\left(1+\frac{(N-1)E_{X-\{x^*\}}\left[\frac{P(x|z)}{P(x)}\right]}{\frac{P(x^*|z)}{P(x)}}\right)\right] = E_X\left[-\log\left(1+\frac{(N-1)}{\frac{P(x^*|z)}{P(x)}}\right)\right].$$

Lastly, using that if $k>0$, then $-\log a(k+1)\geq -\log(1+ak)$, we obtain:

$$E_X\left[-\log\left(1+\frac{(N-1)}{\frac{P(x^*|z)}{P(x)}}\right)\right] = E_X\left[\log\frac{1}{1+\frac{P(x^*)}{P(x^*|z)}(N-1)}\right]$$
$$\leq E_X\left[\log\left(\frac{1}{\frac{P(x^*)}{P(x^*|z)}}\frac{1}{N}\right)\right]$$
$$= E_X\left[\log\left(\frac{P(x^*|z)}{P(x^*)}\frac{1}{N}\right)\right]$$
$$= E_X\left[\log\left(\frac{P(x^*|z)}{P(x^*)}\right)\right] - \log N$$
$$\overset{(1)}{=} E_X\left[\log\left(\frac{P(x^*,z)}{P(x^*)P(z)}\right)\right] - \log N$$
$$\overset{(2)}{=} I(x,z) - \log N,$$

where, in (1), we have use Bayes' rule again and in (2) we have used the definition of the MI that we found in equation 7. Looking at the first and last equations used in this proof, and seeing that we have $\leq$ in the middle of the chain of equalities, we have proven

$$E_X\left[\log\frac{h_\theta(x^*,z)}{\sum_{x\in X}h_\theta(x,z)}\right] = \ell(\theta)\leq I(x,z)-\log N \implies I(x,z)\geq \ell(\theta)+\log N,$$

as we wanted. $\square$

Part III

REPRESENTATION LEARNING

7 CONTEXT

Machine learning is the field of computer science that studies algorithms that improve automatically through experience from examples. These algorithms allow computers to discover how to perform tasks without being explicitly programmed to do them. For the computers to learn, it is mandatory that a finite set of data (or dataset) $\mathcal{D}$ is available.

Whenever a computer is providen with data, the data can be *labeled* or *unlabeled*. Labeled data is the one that, each point $x_i \in \mathcal{D}$ is related to a tag $y_i \in Y$, where Y is a set of classes. Unlabeled data is the kind of data that does not have a label or class associated to it, so it is just $x_i \in \mathbb{R}^d$.

Depending on how the data (*or signal*) is given to the computer, the machine learning approaches can be divided into three broad categories:

1. *Supervised learning*. In this category the goal is to use the labeled data in order to find a function that maps the dataset to the set of classes. That is a function $g : \mathcal{D} \to Y$. An example of supervised learning is image classification: giving a label to an image.
2. *Unsupervised learning*. In ths case, the data is unlabeled, so the approach is completely different. Usually, the goal here is to discover hidden patterns in data or to learn features from it. An example of this kind of learning is K means, which consistis in clustering the data in k groups. It is also known as *self-supervised* learning.
3. *Reinforcement learning*. This is the area concerned with how intelligent agents take decisions in a specific environment in order to obtain the best reward in their objective. It focuses on finding a balance between exploration of uncharted territory and explotation of the current knowledge.

In this work, we will focus on unsupervised learning. Particularly, in representation learning.

There are many different tasks that can be performed with the data, such as linear regression, logistic regression, or classification. In any of these tasks, computers might need to do intermediate steps before giving a label to the input example. Sometimes, they must create a *representation* that contains the data's key qualities. Here is where *representation learning* is born.

Having a representation $\tilde{x}$ of a datapoint $x \in \mathbb{R}^d$, if a machine learning model tries to make a posterior task, such as classification, the input x must be transformed to a, usually lower dimensional, vector r in order to perform the final label.

Features are parts or patterns of an datapoint $x \in \mathcal{D}$ that help to identify it. In fact, it is desiderable that this attribute is shared by all independent units that represent the same object. For instance, if we consider an image of any square, we should be able to identify 4 corners and 4 edges. These could be features of a square. When we mention feature detection, we are adressing the methods for detecting these features of a datapoint.

Representation learning is a set of techniques that allow a system to discover the representations needed for feature detection or classification. In contrast to manual feature engineering (which involves manually exploring the data and finding relationships in it), feature learning allows a machine to learn the features and to use them to perform a task.

Feature learning can be supervised or unsupervised.

- In supervised feature learning, representations are learned using labeled data.Examples of this kind of feature learning are supervised neural networks and multilayer perceptron.
- In unsupervised learning, the features are learned using unlabeled data. There are many examples of this, such as independent component analysis (ICP) and autoencoders.

In this work, we will be working with unsupervised feature learning, so we will have unlabeled data that we would like to find a representation for.

The performance of machine learning methods is heavily dependent on the choice of data features (Bengio *et al.*, 2014). This is why most of the current effort in machine learning focuses on designing preprocessing and data transformation that lead to good quality representations. A representation will be of good quality when its features produce good results when we evaluate the *accuracy* of our model.

The main goal in representation learning is to obtain features of the data that are generally good for either of the supervised tasks. These tasks are usually called *downstream tasks*. That is, we would like to obtain a representation that is either good for image classification (giving an image a label of what we can see in it) or image captioning (producing a text that describes the image).

Data's features that are invariant through time are very useful for machine learning models. In (Wiskott & Sejnowski, 2002), *slow features* are presented. Slow features are defined as features of a signal (which can be the input of a model) that vary slowly during time. That means, if $\boldsymbol{X}$ is a *time series*[1], we will try to find any number of features in $\boldsymbol{X}$ that vary the most slowly. These kind of features are the most interesting ones when creating representations, since they give an abstract view of the original data.

Example 6. In computer vision, the value of the pixels in an image can vary fast. For instance, if we have a zebra on a video and the zebra is moving from

1 A time series is an ordered sequence of values of a random variable at, usually, equally spaced time intervals.

one side of the image to the other, due to the black stripes of this animal, the pixels will fast change from black to white and viceversa, so value of pixels is probably not a good feature to choose as an slow feature. However, there will always be a zebra on the image, so the feature that indicates that there is a zebra on the image will stay positive throughout all the video, so we can say that this is a slow feature.

We will be studying different models that try to learn representations from raw data without labels, as we have mentioned. We usually need a function that measures what is the penalty that the model gets for a choice of a parameter. This is called a *loss function*, that we will want to optimize.

For instance, in a regression problem, a good example of loss function is *mean squared error*, which is expressed as follows:

$$\text{MSE} = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}.$$

In a classification problem, each datapoint x_i has a correct classification y_i. In this case, the score of the correct category y_i should be greater than the sum of the scores of all incorrect categories y_j with $j \neq i$, so we could use a function like *support vector machine (SVM) loss*:

$$\text{SVMLoss} = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

In the context of machine learning, a *model* is the result of running a maching learning algorithm in data. This model will represent what the computer has learned from the data using this algorithm. As an easy example, when we use a linear regression algorithm, we obtain a model that is a vector of coefficients with specific values.

In the following chapters, we will explain different kinds of neural networks and which kind of loss functions they use.

8 GENERATIVE MODELS

The vast majority of the problems in ML are usually of a discriminative nature, which is almost a synonym of supervised learning. However, there also exist problems that involve learning how to generate new examples of the data. More formally:

Definition 8.0.1.
1. *Discriminative models* estimate $p(y|x)$, the probability of a label y given an observation x.
2. *Generative models* estimate $p(x)$, the probability of observing the datapoint x. If the dataset is labeled, a generative model can also estimate the distribution $p(x|y)$.

From now on, let $\mathcal{D}$ be any kind of observed data. This will always be a finite subset of samples taken from a probability distribution p_{data}. There are models that, given $\mathcal{D}$, try to approximate the probability distribution that lies underneath it. These are called *generative models (G.M.)*.

Generative models can give parametric and non parametric approximations to the distribution p_{data}. In our case, we will focus on parametric approximations where the model searches for the parameters that minimize a chosen metric (which can be a distance or other kind of metric such as K-L divergence) between the model distribution and the data distribution.

We can express our problem more formally as follows. Let θ be a generative model within a model family $\mathcal{M}$. The goal of generative models is to optimize:

$$\min_{\theta \in \mathcal{M}} d(p_{\text{data}}, p_\theta),$$

where d stands for the distance between the distributions. We can use, for instance, K-L divergence.

Generative models have many useful applications. We can however remark the tasks that we would like our generative model to be able to do. Those are:

- Estimate the density function: given a datapoint, $x \in D$, estimate the probability of that point $p_\theta(x)$.
- Generate new samples from the model distribution $x \sim p_\theta(x)$.
- Learn useful features of the datapoints.

If we have a look again at the example of the zebras, if we make our generative model learn about images of zebras, we will expect our $p_\theta(x)$ to be high for zebra's images. We will also expect the model to generate new images of this animal and to learn different features of the animal, such as their big size in comparison with cats.

8.1 AUTOREGRESSIVE MODELS

In time-series theory, autoregressive models use observations from previous time steps to predict values at the current time. Fixing an order of the variables $x_1, \dots, x_n$, the distribution for the i-th random variable depends on all the preceding values in the particular chosen order. We will make use of the name of these models to define the machine learning approach.

A very first definition of *autoregressive models (AR)* would be the following one: *autoregressive models are feed-forward models that predict future values using past values*. Let us go deeper into this concept and explain how it behaves.

Again, let $\mathcal{D}$ be a set of $n-$dimensional datapoints x. We can assume that $x \in \{0,1\}^n$ for simplicity, without losing generality. If we choose any $x \in \mathcal{D}$, using the chain rule of probability, we obtain

$$p(x) = \prod_{i=1}^{n} p(x_i | x_1, \dots, x_{i-1}) = \prod_{i=1}^{n} p(x_i | \boldsymbol{x}_{<i}),$$

where $\boldsymbol{x}_{<i} \in \mathbb{R}^{i-1}$ is a vector whose componentes are the previous x_j for $j = 1, \dots, i-1$, that is: $\boldsymbol{x}_{<i} = [x_1, \dots, x_{i-1}]$.
It is known that given a set of discrete and mutually dependent random variables, they can be displayed in a table of conditional probabilities. If K_i is the number of states that each random variable can take then $\prod K_i$ is the number of cells that the table will have. If we represent $p(x_i|\boldsymbol{x}_{<i})$ for every i in tabular form, we can represent any possible distribution over n random variables.

This, however, will cause an exponential growth on the complexity of the representation, due to the need of specifying 2^{n-1} possibilities for each case. In terms of neural networks, since each column must sum 1 because we are working with probabilities, we have $2^{n-1} - 1$ parameters for this conditional, and the tabular representation becomes impractical for our network to learn when n increases.

In autoregressive generative models, the conditionals are specified as we have mentioned before: parameterized functions with a fixed numbers of parameters. More precisely, we assume the conditional distributions to be Bernoulli random variables and learn a function f_i that maps these random variables to the mean of the distribution. Mathematically, we have to find

$$p_{\theta_i}(x_i | \boldsymbol{x}_{<i}) = \text{Bern}(f_i(x_1, \dots, x_{i-1})),$$

where θ_i is the set of parameters that specify the mean function $f_i : \{0,1\}^{i-1} \to [0,1]$.

Then, the number of parameters is reduced to $\sum_{i=1}^{n} |\theta_i|$ so we can not represent all possible distributions as we could when using the tabular form of the conditional probabilities. We are now setting the limit of its expressiveness because we are setting the conditional distributions $p_{\theta_i}(x_i|\boldsymbol{x}_{<i})$ to be *Bernoulli* random variables with the mean specified by a restricted class of parametrized functions.

Let us see a very simple case first in order to understand it better. Let σ be a *sigmoid*[1] non linear function and $\theta_i = \left\{\alpha_0^{(i)}, \alpha_1^{(i)}, \ldots, \alpha_{i-1}^{(i)}\right\}$ the parameters of the mean function. Then, we can define our function f_i as :

$$f_i(x_1, \ldots, x_{i-1}) = \sigma(\alpha_0^{(i)} + \alpha_1^{(i)} x_1 + \cdots + \alpha_{i-1}^{(i)} x_{i-1}).$$

In this case, the number of parameters would be $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$, so using *Big O* notation, we would be in the case of $O(n^2)$. We will state now a more general and useful case, giving a more interesting parametrization for the mean function: *multi layer perceptrons*[2](MLP).

For this example we will consider the most simple MLP: the one with one hidden layer. Let $h_i = \sigma(\boldsymbol{A}_i \boldsymbol{x}_{<i} + c_i)$ be the hidden layer activation function. Remember that $h_i \in \mathbb{R}^d$. Let $\theta_i = \{\boldsymbol{A}_i \in \mathbb{R}^{d\times(i-1)},\ c_i \in \mathbb{R}^d,\ \alpha^{(i)} \in \mathbb{R}^d,\ b_i \in \mathbb{R}\}$ the set of parameters for the mean function f_i, that we define as:

$$f_i(\boldsymbol{x}_{<i}) = \sigma(\alpha^{(i)} h_i + b_i).$$

In this case, the number of parameters will be $O(n^2 d)$.

This is the simplest example. Currently, there are alternative parametrization models , such as the *Neural Autoregressive Density Estimator* (Larochelle & Murray, n.d.), that provide a more statistically and computationally efficient solution. In fact, the number of parameters is reduced from $O(n^2 d)$ to $O(nd)$. Also, *RNADE* (Uria *et al.*, 2014) extends NADE to learn generative models over real-valued data, generalizing the case that we have just exposed. However, these models are out of the scope of this project so no further explanation will be given.

1 A sigmoid function is a bounded,differentiable, real function which derivative is non-negative at each point and it has exactly one inflection point.

2 Multi layer perceptrons are feed-forward neural networks with at least 3 layers: input, hidden and output layers; each one using an activation function.

9 THE INFONCE LOSS

We are now ready to connnect the concepts of mutual information and generative models that we have presented. In unsupervised learning, it is a common strategy to predict future information and to try to find out if our predictions are correct. In *natural language processing*, for instance, representations are learned using neighbouring words (Mikolov *et al.*, 2013). In the field of computer vision, some studies have been able to predict color from grey-scale (Doersch *et al.*, 2016).

When we work with high-dimensional data, it is not useful to make use of an unimodal loss function to evaluate our model. If we did it like this, we would be assuming that there is only one peak in the distribution function and that it is actually similar to a Gaussian. This is not always true, so we can not assume it for our models. Generative models can be used for this purpose: they will model the relationships in the data x. However, they ignore the context c in which the data x is involved. As an easy example of this, an image contains thousands of bits of information, while the label that classifies the image contains much less information , say, 10 bits for 1024 categories. Because of this, modeling $P(x|c)$ might not be the best way to proceed if we want to obtain the real distribution that generates our data.

During the last few years, the representation learning problem has been approached using different machine learning frameworks. The most competitive ones have been self-supervised contrastive representation learning Oord *et al.* (2019); Tian *et al.* (2020); Hjelm *et al.* (2019); Gutmann & Hyvarinen (n.d.); Chen *et al.* (2020); He *et al.* (2020) using *contrastive losses*, and they have empirically outperformed other approaches.

In contrastive learning, different "views" of the same input are created. These are also called *positive samples*. Then, they are compared with *negative samples*, which are views created from an input that does not share information with the input of the positive sample. A very interesting idea would be to try and maximize the mutual information between positive samples and push appart the views taken from negative samples.

There are many ways of creating samples, both positive and negative, of an input. For instance:

- Randomly cropping different parts of an image. These would be examples of positive examples.
- Rotating or flipping images or crops of them would also be examples of positive samples.
- Taking different time-steps of a video would create positive samples.
- Selecting different parts of the same text would also be a positive example.

- Negative samples are created by applying one of the previous techniques to images that have nothing in common with the positive input.

In fact, if v_1, v_2 are two views of an input, we can think of the positive pairs as points coming from a joint distribution over the views $P(v_1, v_2)$, and negative samples coming from the product of the marginals $P(v_1)P(v_2)$, (Tian *et al.*, 2020).

It is important to find a way to determine how much shared information between the views is needed, in order to make the representations obtained good enough for any downstream task. Here is where the *InfoMin principle* is born.

> **Definition 9.0.1** (The InfoMin principle)**.** A good set of views are those that share the minimal information neccesary to perform well at the downstream task.

Our goal here will be to seek for a way of extracting shared information between the context c and the data x. Here is where we link the mutual information with representation learning. Remember that the mutual information of two random variables, say x and c in this case, is:

$$I(x, c) = \sum_{x,c} P(x, c) \log \frac{P(x|c)}{P(x)} \tag{9}$$

Maximizing the MI between x and c, we extract the latent variables that the inputs have in common.

9.1 CONTRASTIVE PREDICTIVE CODING

We can apply these concepts in a concrete framework, presented firstly in Oord *et al.* (2019). Let us see what information is used and how it is treated in order to train a model that tries to obtain useful representations for downstream tasks.

In this section, if x is an input signal for our network, x_t will be the value of the input at instant t. We will also make reference to x_{t+k}, meaning that x_{t+k} is k steps ahead of time to x_t

Firstly, an *encoder* is used. An encoder is a model that, given an input x, provides a feature map or vector that holds the information that the input x had.

So, we will use an encoder g_{enc} that transforms the input sequence of observations x_t to a sequence of latent representations

$$z_t = g_{enc}(x_t).$$

After we have obtained z_t, we use it as input of an autoregressive model, explained before, to produce a context latent representation:

$$c_t = g_{ar}(z_{\leq t}).$$

In this case, c_t will summarize the information of z_i for $i \leq t$. Following the argument that we gave before, predicting the future x_{t+k} using only a generative model (say $p_k(x_{t+k}|c)$) might not be correct, since we would be ignoring the context.

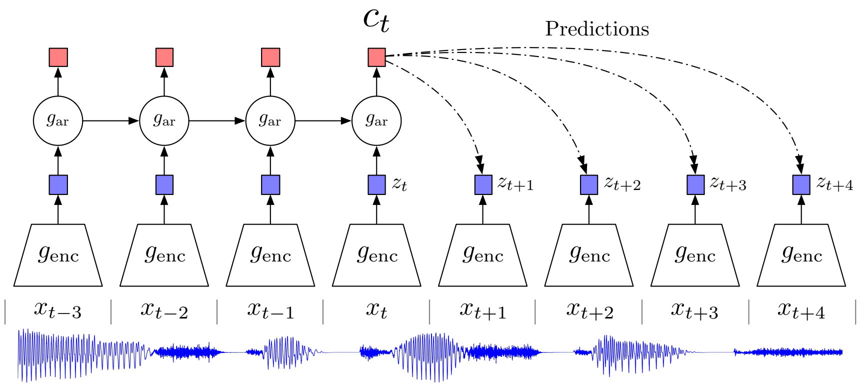

Figure 5: Image from (Oord *et al.*, 2019). Overview of Contrastive Predictive Coding framework using audio signal as input.

Let us see how we train the encoder g_{enc} and the autoregressive model g_{ar}.

In Chapter 4, we gave the notions of the general idea of Noise Contrastive Estimation. Now, we can apply those ideas to a particular case, in which one of the subsets, say X only has one element, and the other one has $N-1$ elements. We combine both sets in X for the following argument.

Let $X = \{x_1, \ldots, x_N\}$ be a set of N random samples. X will contain a positive sample taken from the distribution $P(x_{t+k}|c_t)$ and $N-1$ negative samples from the distribution proposed $P(x_{t+k})$. With this set, we would like to optimize the following loss function, which is an alternative expression of 4:

$$\mathcal{L}_N = -\underset{X}{E}\left[\log \frac{f_k(x_{t+k}, c_t)}{\sum_{x_j \in X} f_k(x_j, c_k)}\right]. \tag{10}$$

This is known as the *InfoNCE* (Information Noise Contrastive Estimation) loss, which is based in Noise Contrastive Estimation (Gutmann & Hyvarinen, n.d.).

Let us have a look at the *categorical cross-entropy* loss function:

$$\mathcal{L}(y, s) = -\sum_{i}^{C} y_i \log(s_i)$$

where C is the number of possible classes in a classification problem, y_i are the groundtruth of each class and s_i is the score of each class.

We can say that $\mathcal{L}_N$ is no more than the categorical cross-entropy of classifying the positive sample of X correctly, with the argument of the logarithm being the prediction of the model. If we note with $[d = i]$ as an indicator of the sample x_i being the positive sample in X, the optimal probability for this loss can be written as $P(d = i|X, c_t)$.

Now, the probability that x_i was drawn from the conditional distribution $P(x_{t+k}|c_t)$ that has the context in account, rather than the proposal distribution $P(x_{t+k})$ that does not have c_t in account, leads us to the following expression:

$$P(d=i|X,c_t) = \frac{\frac{P(x_i|c_t)}{P(x_i)}}{\sum_{j=1}^{N} \frac{P(x_j|c_t)}{P(x_j)}}.$$

This is the optimal case for (10).

In fact, if we denote $f(x_{t+k}, c_t)$ as the density ratio that preserves the mutual information between x_{t+k} and c_t in the mutual information definition (9), if x_{t+k} is k steps ahead on time respect to x_t, then

$$f_k(x_{t+k}, c_t) \propto \frac{P(x_{t+k}|c_t)}{P(x_{t+k})}, \tag{11}$$

where $\propto$ means that the member on the left is proportional to the member on the right. We can see that the optimal value $f_k(x_{t+k}, c_t)$ does not depend on $N-1$, the number of negative samples in X. Using this density ratio, we are relieved from modeling the high dimensional distribution x_{t_k}. In Oord *et al.* (2019), for instance, the following log-bilinear model expression is used:

$$f_k(x_{t+k}, c_t) = exp(z_{t+k}^T W_k c_t).$$

In the proposed model, we can either use the representation given by the encoder (z_t) or the representation given by the autoregressive model (c_t) for downstream tasks. Clearly, the representation that aggregates information from past inputs will be more useful if more information about the context is needed. Furthermore, any type of models for the encoder and the autoregressive models can be used in this kind of framework.

It is clear how, using (10) we are using exactly the same function that we were using in Equation 8 in Chapter 6. This way, if we maximize this loss, we are also maximizing the mutual information between x_{t+k} and the context c_t.

9.2 GOOD VIEWS FOR CONTRASTIVE LEARNING

We have presented a framework in which a set $X = \{x_1, \cdots, x_n\}$ contains a sample from the distribution $P(x_{t+k}, ct)$ and the rest are samples from the distribution $P(x_{t+k})$. This samples are different views of the data.

The choice of the views affects the results in the downstream tasks (Tian *et al.*, 2020). The views will affect on the training, hence, it will affect to the representations that are obtained. We would like to have some guarantees that the views that we are choosing provide us with good examples for our training. Let us formalize this idea.

Given two random variables v_1, v_2, our goal was to learn a function to discriminate the samples from the joint distribution and the product of the marginal distributions, resulting on a mutual information estimator between v_1 and v_2. In practice, v_1 and v_2 are two views of the same input x, using one of the methods that we mentioned in the introduction. We would like to have that, if y is a downstream task, the mutual information between both the inputs and the downstram task, is the same as the mutual information between the input x and the downstream task, i.e.:

$$I(v_1, y) = I(v_2, y) = I(x, y).$$

Also, we would like to remove the information that is not relevant for our downstream task. This is done by obtaining the pair of views (v_1^*, v_2^*) such that the mutual information between them is the minimum of the mutual information between all the possible views (v_i, v_j). Formally, that is obtaining

$$(v_1^*, v_2^*) = \min_{v_1, v_2} I(v_1, v_2),$$

These two ideas form the *InfoMin Principle* that we mentioned before in Definition 9.0.1.

Usually, the views are encoded using an encoder f, not having it to be the same for both views. We can say that $z_i = f_i(v_i)$ for $i \in \{1, 2\}$. If an encoder is sufficient, then it has to maintain the mutual information between the random variables after one of them has been encoded. More formally,

> **Definition 9.2.1.** We say an encoder f_i of a view v_i, with $i \in \{1, 2\}$ is *sufficient* in the contrastive learning framework if, and only if it maintains the mutual information between the pairs (v_i, v_j) and $(f_i(v_i), v_j)$ with $j \in \{1, 2\}$. That is
> $$I(v_i, v_j) = I(f_i(v_i), v_j).$$

This usually means that no information was lost in the process of encoding. We want to extract only the most essential information and do not learn the "extra" information between the views.

> **Definition 9.2.2.** We say that a sufficient encoder f_i of v_i is *minimal* if, and only if, the mutual information between $(f_i(v_i), v_j)$ is lesser than the mutual information between $(f(v_i), v_j)$ for any other sufficient encoder f. That is:
> $$I(f_i(v_i), v_j) \leq I(f(v_i), v_j) \quad \text{for all sufficient } \ f.$$

With these notions already presented, we would like to define what representations are good for a downstream task. We get to the following definition (Tian *et al.*, 2020):

> **Definition 9.2.3.** For a task $\mathcal{T}$, whose goal is to predict a label y from the input data x, the optimal representation z^* encoded from x, that is $z^* = f(x)$ for some encoder f, is the minimal sufficient statistic with respect to y.

This means that if we use z^* to make a prediction using a machine learning model, we are using the same information that we would be using the whole input x. What is more, since we are following the InfoMin principles and we are dismissing all the non relevant information, z^* provides with the smallest complexity.

Proposition 7. *Let f_1, f_2 be minimal sufficient encoders, and $\mathcal{T}$ be a downstream task with label y. Then, the optimal views (v_1^*, v_2^*) from the data x are the ones that have minimal mutual information*

$$(v_1^*, v_2^*) = \underset{v_1, v_2}{\arg\min}\, I(v_1, v_2),$$

subjecto to $I(v_1, y) = I(v_2, y) = I(x, y)$.

Given the optimal views (v_1^*, v_2^*), the representation z_1^* learned by contrastive learning is optimal for $\mathcal{T}$.

The proof of this proposition is out of the scope of this work so no further information will be provided. The last statement of the Proposition 7 is a consequence of the minimality and sufficency of f_1 and f_2.

This proposition carries the most important mathematical conclusion from this section. It will serve to prove sufficient conditions for views to be effective for contrastive learning, where effectiveness is measured as effectiveness in downstream tasks.

10 TRIPLET LOSSES

We have seen how the framework presented in **?** uses a generative approach as a part of the representation learning process. Let us set in the case of learning representations of images. In this case, generative models must *generate* each pixel on the image. This can be extremely computationally expensive.

Until now, we had been trying to minimize the loss in 10, which we proved that maximizes a lower bound in the Mutual Information. However, some papers such as Chen *et al.* (2020), Tschannen *et al.* (2020), suggest that it is unclear if the success of their methods is caused by the maximization of mutual information between the latent representations, or by the specific form that the constrastive loss has.

In fact, in Tschannen *et al.* (2020) they provide empirical proof for the loose connection between the success of the methods that use MI maximization and the utility of the MI maximization in practice. They also empirically proof that the encoder architecture can be more important than the estimator used to determine the MI.

Even with the empirically proved disconnection between MI maximization and representation quality, recent works that have used the loss function 10 have obtained state-of-art results in practice. There is an explanation for this, connecting the recently mentioned loss with a popular triplet loss.

10.1 FROM DEEP METRIC LEARNING TO TRIPLET LOSSES

Distance metric learning aims to learn an embedding representation of an input data x that preserves the distance between similar data points close and also makes de distance between different datapoints far on the embedding space (Sohn, 2016).

We will consider sets of triplets (x, x^+, x^-) where:

- The element x is an anchor point,
- The element x^+ is a positive instance,
- The element x^- is a negative instance.

Example 7. Let us present a very simple example. If our input image is a cat, that would be the anchor x. Clearly, a positive instance would be an image of another cat or even the same cat seen from another perspective. A negative instance would be a photo of any other animal, in this case we use a dog.

Anchor Positive example Negative example

Figure 6: Example of an anchor x, a positive instance x^+ and a negative instance x^-. Images obtained from *Google*.

The main idea is to learn a representation of x, say $g(x)$, such that the distance of the representation of the input is closer in distance to the representation of the positive sample x^+ than the representation of the negative sample x^-. Using the norm[1], we can formally express that as follows:

$$\left\|g(x) - g(x^+)\right\|_2 \leq \left\|g(x) - g(x^-)\right\|_2,$$

for each triplet in the set.

Support-vector machines (SVMs) are supervised learning models used for classification or regression problems. They are one of the most robust predicion methods. They search for a hyperplane h in high or infinite dimensional space that separates the data as much as possible, making use of *support vectors*, the datapoints that are closest to the hyperplane. If the data is linearly separable, we can select two hyperplanes h_1, h_2 that are parallel to h and making the distance from them to h as large as possible. That region is called the *margin*.

Coming back to our triplets problem, we also want to introduce a margin between the distances of the elements of the triplets, in order to separate positive examples from negative examples as much as possible. This way, we introduce a *margin* term α, rewriting our last equation as follows:

$$\left\|g(x) - g(x^+)\right\|_2 + \alpha < \left\|g(x) - g(x^-)\right\|_2.$$

Using this inequality, we can define a hinge loss function for each triplet in the set:

$$\ell^\alpha(x, x^+, x^-) = \max\left(0, \left\|g(x) - g(x^+)\right\|_2 - \left\|g(x) - g(x^-)\right\|_2 + \alpha\right). \quad (12)$$

This loss has been defined for a single triplet. Now, we can define a global loss that accumulates the loss in Equation (**??**) using all the triplets in set.

1 A definition of the norm can be found on Appendix 11, Definition 11.0.1.

Definition 10.1.1. Given a set of triplets, each containing an anchor, a positive example and a negative example, $\mathcal{T} = \{(x_i, x_i^+, x_i^-)\}_{i \in \Lambda}$, we define a triplet loss as follows:

$$\mathcal{L} = \sum_{i \in \Lambda} \ell^{\alpha}(x_i, x_i^+, x_i^-). \tag{13}$$

It would be interesting to present the model non-trivial metric to the learning algorithm. When the representation g improves, this is harder to do.

The InfoNCE loss on Equation 10 has proved to be useful in representation learning. Let us consider a reformulation on it. Firstly, since f_k was an exponential, we can also consider e^f and remove the exponential from f, this is just notation. Now, we can rewrite the InfoNCE objective as follows

$$I_{NCE} = E\left[\frac{1}{N}\sum_{i=1}^{N} \log \frac{e^{f(x_i, y_i)}}{\frac{1}{N}\sum_{j=1}^{N} e^{f(x_i, y_j)}}\right] = \log N - E\left[\frac{1}{N}\sum_{i=1}^{N} \log\left(1 + \sum_{j \neq i} e^{f(x_i, y_j) - f(x_i, y_i)}\right)\right].$$

Part IV

APPENDIX

11 APPENDIX A

This appendix will be used to set forth some theoretical results that might not always be relevant but are needed to understand some details during this thesis. Not all of them will be proven.

Proposition 8 (Jensen's Inequality). *Let $f : \mathcal{D} \to \mathbb{R}$ be a concave function and $n \in \mathbb{N}$. For any $p_1, \dots, p_n \in \mathbb{R}_0^+$ with $\sum p_i = 1$ and any $x_1, \dots, x_n \in \mathcal{D}$, it holds that:*

$$\sum_{i=1}^{n} p_i f(x_i) \leq f\left(\sum_{i=1}^{n} p_i x_i\right).$$

Furthermore, if f is strictly *concave and $p_i \geq 0$ for all $i = 1, \dots, n$, then the equality holds if, and only if, $x_1 = \cdots = x_n$.*

In Chapter 10, the norm $\|\cdot\|_2$ is mentioned. Norm theory is a very extense field, so we will only mention the definition and the norm that we will use in the text.

Definition 11.0.1. Given a vector space X over a subfield F of the complex numbers $\mathbb{C}$, a *norm* is a real valued function $\|\cdot\| : X \to \mathbb{R}$ with the following properties:

1. Triangle inequality, that is: $\|x + y\| \leq \|x\| + \|y\|$ for all $x, y \in X$.
2. Absolute homogeinity, that is: $\|sx\| = |s| \, \|x\|$ for all $x \in X$ and any scalar s.
3. Positive definiteness, that is $\|x\| \geq 0$ for all $x \in X$ and $\|x\| = 0$ if, and only if, $x = 0$.

In particular, the $\|\cdot\|_2$ that we used in the euclidean space $\mathbb{R}^n$, is defined as follows:

$$\|x\|_2 := \left(\sum_{i=1}^{n} x_i^2\right)^{\frac{1}{2}}, \qquad \forall x \in \mathbb{R}^n.$$

BIBLIOGRAPHY

Bengio, Yoshua, Courville, Aaron, & Vincent, Pascal. 2014. Representation Learning: A Review and New Perspectives. *arXiv:1206.5538 [cs]*, Apr. arXiv: 1206.5538.

Chen, Ting, Kornblith, Simon, Norouzi, Mohammad, & Hinton, Geoffrey. 2020. A Simple Framework for Contrastive Learning of Visual Representations. *arXiv:2002.05709 [cs, stat]*, June. arXiv: 2002.05709.

Cover, T. M., & Thomas, Joy A. 1991. *Elements of information theory*. Wiley series in telecommunications. New York: Wiley.

Doersch, Carl, Gupta, Abhinav, & Efros, Alexei A. 2016. Unsupervised Visual Representation Learning by Context Prediction. *arXiv:1505.05192 [cs]*, Jan. arXiv: 1505.05192.

Gutmann, Michael U, & Hyvarinen, Aapo. Noise-Contrastive Estimation of Unnormalized Statistical Models, with Applications to Natural Image Statistics. 55.

He, Kaiming, Fan, Haoqi, Wu, Yuxin, Xie, Saining, & Girshick, Ross. 2020. Momentum Contrast for Unsupervised Visual Representation Learning. *arXiv:1911.05722 [cs]*, Mar. arXiv: 1911.05722.

Hjelm, R. Devon, Fedorov, Alex, Lavoie-Marchildon, Samuel, Grewal, Karan, Bachman, Phil, Trischler, Adam, & Bengio, Yoshua. 2019. Learning deep representations by mutual information estimation and maximization. *arXiv:1808.06670 [cs, stat]*, Feb. arXiv: 1808.06670.

Larochelle, Hugo, & Murray, Iain. The Neural Autoregressive Distribution Estimator. 9.

Löwe, Sindy, O'Connor, Peter, & Veeling, Bastiaan. 2019. Putting an End to End-to-End: Gradient-Isolated Learning of Representations. *Pages 3039–3051 of: Advances in Neural Information Processing Systems*.

Mikolov, Tomas, Chen, Kai, Corrado, Greg, & Dean, Jeffrey. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781 [cs]*, Sept. arXiv: 1301.3781.

Oord, Aaron van den, Li, Yazhe, & Vinyals, Oriol. 2019. Representation Learning with Contrastive Predictive Coding. *arXiv:1807.03748 [cs, stat]*, Jan. arXiv: 1807.03748.

Sohn, Kihyuk. 2016. Improved Deep Metric Learning with Multi-class N-pair Loss Objective. *In: NIPS*.

Tian, Yonglong, Sun, Chen, Poole, Ben, Krishnan, Dilip, Schmid, Cordelia, & Isola, Phillip. 2020. What Makes for Good Views for Contrastive Learning? *arXiv:2005.10243 [cs]*, Dec. arXiv: 2005.10243.

Tschannen, Michael, Djolonga, Josip, Rubenstein, Paul K., Gelly, Sylvain, & Lucic, Mario. 2020. On Mutual Information Maximization for Representation Learning. *arXiv:1907.13625 [cs, stat]*, Jan. arXiv: 1907.13625.

Uria, Benigno, Murray, Iain, & Larochelle, Hugo. 2014. RNADE: The real-valued neural autoregressive density-estimator. *arXiv:1306.0186 [cs, stat]*, Jan. arXiv: 1306.0186.

Wiskott, Laurenz, & Sejnowski, Terrence J. 2002. Slow Feature Analysis: Unsupervised Learning of Invariances. *Neural Computation*, **14**(4), 715–770.