University of Trieste

International School for Advanced Studies

The Abdus Salam International Centre for Theoretical Physics

Probabilistic Machine Learning

Lectures Notes

Author:
Marco Sciorilli

Tuesday 20th April, 2021

Abstract

This document contains my notes on the course of Probabilistic Machine Learning held by Prof. Luca Bortolussi for the Master Degree in Data Science and Scientific Computing at Trieste University in the year 2020/2021. As they are a work in progress, every correction and suggestion is welcomed. Please, write me at: marco.sciorilli@gmail.com .

Contents

Chapter 1

Introduction to the course

1.1 Introduction

A model is a kind of a step of abstraction: it describes a complex object with a simpler one understandable by us. More specifically a mathematical model is one that use the language of mathematics, describes systematically relations between objects. A stochastic model is a mathematical model which has probability distributions as objects. We use a set of parameters to build the model, and we try to find the best set of parameters to describe the observed object. We want to automatically chose the best model among all possibles, in a thought through way. A major difference with statistic is that we focus on the algorithm, rather than the data. This allows us to use model usually far more complicated then the statistical one. The course is about how to effectively find a way to describe data.

1.2 Kind of learning

Different kind of machine learning, explained in a probabilistic prospective:

- Supervised learning: the models are functions, from a input it gives an output. There can be categorical output. Most of nowadays machine learning is of this kind.
- Unsupervised learning: find pattern in data
- Reinforcement learning: making the best decision in a certain scenario.

In this course we will mostly stick to supervised learning.

1.3 Generative and discriminative models

Generative models aim at describing the full probability distribution of input using a joint probability distribution of a input and an output. Probabilities are a reasonable way to describe the world. Discriminative learning: wants to describe the conditional probability, in order to discriminate among inputs.

We will try to work mostly on joint probability distribution.

1.4 Machine learning

Supervised learning: learn the joint distribution.
Unsupervised: learn the input probability and its property p(x).
Data generation: learn how to sample. Or maybe create new input from the probability distribution of input.

1.5 Inference and estimation

Inference: starting with a complex distribution, and compute simpler conditional distributions. we want doing inference in an effective way.
Estimation: given a set of parameters, with wants to find the best value of such parameter in the model, even a very complex one, to best describe the input.
In Bayesian statistic, estimation became an operation of inference. Bayesian machine learning has the downside that the computation is way harder (and don't scale very well), but it usually get the most out of data. The asymptotic equalness with the frequentist approach depends on the complexity of the model, usually not suitable for the input data-set available.

Chapter 2

Empirical Risk Minimization

2.1 Basic definitions

Let's create a generalize framework to formulate machine learning problems.

Definition 1. *Supervised Learning: predicting a function linking an input to an output.*

Definition 2. *Input space $X \subseteq \mathbb{R}^n$: the features considered are* **real** *features (could also be 0 and 1, or categorical etc.)*

Definition 3. *Output space $Y \subseteq \mathbb{R}$: can be a real number (could also be 0 and 1, or categorical etc.)*

Empirical risk minimization framework is based on probability, what we have in general is that are trying to describe the world (or the part of the world considered in our problem) in terms of probability. The true system we are trying to capture is a joint probability distribution of the input and the output, which we call "the data distribution".

Definition 4. *Data distribution $p(x, y)$, where $x \in X$ is an element of the input space, $y \in Y$ is an element of the output space. Data distribution $p(x, y) \in Dist(X \times Y)$ a distribution over x time y.*

What we are trying to do is to describe a link, a function link, between input and output, which rather than being strictly deterministic, is allowed to be probabilistic in nature. E.g. given a certain input we may observe several outcomes depending on some random process which is out of the detailed description we want to capture (so it is represented as a probability distribution). Maybe there is a way to understand the details of the problem we are studying, but those are likely too difficult to describe mathematically.
Sometimes we have a functional relationship between input and output (true functional relationship) $f : X \to Y$.
In this case we typically have the joint probability distribution which we can factorize as $p(x, y) = p(x)p(y|x)$ and $p(y|x) = p(y|f(x))$ i.e. the conditional distribution of y depends only on the value of $f(x)$ rather then on x itself. This is typically the case of classification problems.

In machine learning we want to learn this world, and we want to learn it from observations.

Definition 5. *Dataset D, with a certain carnality $|D| = N$, $D \sim p^n(x, y)$ i.e. D is sampled from a probability distribution. This means that $D = \{(x_i, y_i)|i = 1, ..., N\}$, $(x_i, y_i) \sim p(x, y)$ i.e. D is composed by n pairs of inputs and outputs, and each of this pair is sampled from $p(x, y)$ independently from the other pairs.*

This is the scenario in which we are working. We would like to recover at least, if it is there, the function f which map input to output.
Whenever we want to learn, we make hypothesis on which are good candidates functions of our model i.e. we choose a set of functions that for us should contain a good model, maybe even the true model, for our data. This set of functions is our hypothesis class.

Definition 6. *Hypothesis class $H = \{h : X \to Y\}$ is a set of functions from input to output. Typically this set of function is depending on a parameter $h = h_\theta$, $\theta \in \Theta \subseteq \mathbb{R}^k$.*

The assumption we make on the hypothesis class are the crucial ones. This assumption is what allows us to learn.
H **encodes our inductive bias**. I.e. in our induction process, we are biasing out induction itself in a certain direction, which is defined by the hypothesis class.
We need a way to determine how good our choice of hypothesis match the date we observe, and what we predict to observe.

Definition 7. *Loss function $l(x, y, h) \in \mathbb{R}_{\geq 0}$: takes an input x, an output y and a function h, end return how much error we commit using h to predict y. The higher the value, the worst is our prediction.*

Example 1. *Here some examples of loss functions:*

- *0-1 loss: $l(x, y, h) = I(h(x) \neq y)$, for a classification problem ($y \in \{0, 1\}$), is the indicator function that tell us if $h(x) \neq y$. There is no error if we are correctly predicting the class.*
- *Square loss: $l(x, y, h) = (h(x) - y)^2$, for a regression problem ($y \in \mathbb{R}$), is the square difference between our prediction and the observed value. It is always positive and differentiable.*

2.2 Risk and Empirical risk

The loss function acts on a single input-output pair, but we want a function h which work well on any pair according to the joint probability distribution $p(x, y)$. We can encode it with the use of risk.

Definition 8. *Risk, or Generalization error, $R(h) = \mathbb{E}_{x,y \in p(x,y)}[l(x, y, h)]$: every function h comes with a risk, defined as the expectation over the pairs x, y sample over the true data generating distribution of the loss of the pair according to the hypothesis h. e.q. What is the fraction of pair we are going to misclassify adopting the hypothesis h?*

Generalization error depends on the true data generating distribution, which we usually do not know. So in practice, we can replace the risk with something computable, called the empirical risk.

Definition 9. *Empirical risk, also called training error on $D \sim p^N$, $\hat{R}_D(h)$:it is the empirical approximation according to the sample of the actual risk.* $\hat{R}_D(h) = \frac{1}{N}\sum_{i=1} l(x_i, y_i, h)$, *i.e. the average of the loss, sampling N time from the distribution.*

Definition 10. *Risk minimization principle, find $h^* \in H$ s.t. $h^* = \arg\min_{h\in H} R(h)$: we want to find a function h^* in our hypothesis set such that it is the one with the minimum risk among the ones of our hypothesis set.*

If l is the 0-1 loss and $\exists h \in H$ *s.t.* $p(h(x) = f(x)) = 1$, where $f(x)$ is the true class, than H has the **realizability** property, so $R(h^*) = 0$. This means that we can actually find the true model. This is typically not the case.

Definition 11. *Empirical risk minimization, find $h^* \in H$ s.t. $h^*_D = \arg\min_{h\in H} \hat{R}_D(h)$: the same but with empirical risk.*

2.3 Bias Variance Tradeoff

Sometime finding the minimum of the empirical risk is very calculus intensive, so we are just satisfied with a good approximation. Also, what is $R(h^*_D)$ associated with my solution? We can understand the relationship between the true and the empirical risk through the bias-variance trade-off, and we can see if we can give error bound the true generalization error (probably approximately correct learning).

Bias-Variance trade-off in a regression problem

Let's consider the case of a regression problem. We are therefore considering the square loss $h \in H$. So the explicit expression of the generalization error choosing an hypothesis h is:

$$R(h) = \mathbb{E}_p[l(x, y, h)] = \int\int (h(x) - y)^2 p(x, y)dxdy \tag{2.1}$$

We can define a minimizer of R ($g = \arg\min_h R(h)$, if $g \in H$) as:

$$g(x) := \mathbb{E}[y|x] = \int yp(y|x)dy \tag{2.2}$$

Proof. Let's write the generalisation error as:

$$R(h) = \int\int (h(x) - y)^2 p(x, y)dxdy \tag{2.3}$$

We can rewrite the inner term of the integral to make $g(x)$ appear

$$\begin{aligned}(h(x) - y)^2 &= (h(x) - g(x) + g(x) - y)^2 \\ &= (h(x) - g(x))^2 + (g(x) - y)^2 + 2(h(x) - g(x)(g(x) - y)\end{aligned}$$

The lower grade term in the integral is going to cancel out, because in the moment we integrate in respect to y, as is not appearing in the first part, became

$$\int 2(h(x)-g(x)(g(x)-y)p(y, x)dydx = \int 2(h(x)-g(x) \int (g(x)-y)p(y|x)dyp(x)dx \tag{2.4}$$

The second integral is encoding for the conditional expectation:

$$\int (g(x)-y)p(y|x)dy = g(x) - \int yp(y|x)dy = 0 \tag{2.5}$$

So the lower grade term really cancel out.

$$R(h) = \int (h(x)-g(x))^2 p(x)dx + \int\int (g(x)-y)^2 p(x,y)dxdy \tag{2.6}$$

The second term depends only on the conditional expectation, not on the function we are evaluating, so it is essentially a constant, equal for all hs. It tells us how much y is varying across its conditional expectation (the conditional variance of y integrated over all possible xs). This is a sort of term expressing the idea of how noisy is our regression problem, i.e. how much we can deviate from the conditional expectation when we actually observe our process.
The first term is 0 if and only if $h(x) = g(x)$, which shows that $g(x)$ is a minimizer.

$\square$

Now we are going to evaluate our error for a specific h function, the one we obtain by empirical risk minimization principle when we observe dataset D of size N generated according to the distribution p ($D \sim p^n$):

$$R(h^*_D) = \int (h^*_d(x) - g(x))^2 p(x)dx + noise \tag{2.7}$$

We want to study now the expected value of the generalization error with respect to our data. It is the average error we are going to make by adopting the empirical risk minimization to replace the true error.

$$\mathbb{E}_D[R(^*_D)] = \int \mathbb{E}_D[(h^*_D - g(x))^2]p(x)dx \tag{2.8}$$

If we add and subtract in the square parenthesis the expectation with respect to D of our function ($\pm\mathbb{E}_D(h^*_D(x))$). We can work out the square as before, and observe that $\mathbb{E}_D(h^*_D - \mathbb{E}_D[h^*_D(x)]) = 0$ (the expected deviation of an element from its mean is 0). So:

$$\begin{aligned}\mathbb{E}_D[R(h^*_D)] = &\int \mathbb{E}_D(h^*_D - g(x))^2 p(x)dx \\ &+ \int \mathbb{E}_D[(h^*_D(x) - \mathbb{E}_D[h^*_D])^2 p(x)dx \\ &+ \int\int (g(x)-y)^2 p(x,y)dxdy\end{aligned}$$

The term on top captures the square differences between the average predictor across all dataset, obtained from empirical risk minimisation, and the optima predictor, which is the conditional expectation. If the difference is small, on average across all datasets our empirical risk minimisation is going to work, but if this difference is large, it means that across all datasets our empirical risk minimisation is not going to work well. We call this term $bias^2$ (squared-bias) because it essentially captures the intrinsic distortion of our

empirical risk minimization predictor, and so, of our set of hypothesis, and how well we can reconstruct the true function.
The second term encodes an expectation across all datasets of the variance of our predictors, i.e. how far each single instance of the empirical risk minimisation framework differs from the average predictor. It is called *variance* term. The larger the variance, the more the noisy are going to be the single instances of our framework, i.e. how the intrinsic variability of our dataset is going to impact on what we can reconstruct. High variance means we are in a region of over-fitting, low variance the opposite. High bias at the opposite means we are in a region of under-fitting, where we are not able to capture the true dynamic of what is going on.
The last term is the noise. What we observe is that the sum of bias and variance in relation to the complexity of the model, has a kind of convex shape, with a minimum where there is an optima trade-off between this two values.

Chapter 3

PAC learning

3.1 Definition of PAC learning

We fix from now on the 0-1 loss function, $y = \{0, 1\}$. The hypothesis set function has the reliazability property, that for our case can be written as

$$\exists \overline{h} \in H \ s.t. \ p_{x,y}(\overline{h}(x) = y) = 1 \tag{3.1}$$

Definition 12. *H is PAC learnable if and only if*

$$\begin{aligned}
&\forall \epsilon, \delta \in (0,1), \ \forall p(x,y) \\
&\exists m_{\epsilon,\delta} \in \mathbb{N} \ s.t. \ \forall m \geq m_{\epsilon,\delta}, \ \forall D \sim p^m, \ |D| = m \\
&P_D(R(h_D^*) \leq \epsilon) \geq 1 - \delta
\end{aligned}$$

This means: take a scenario (a data-generating distribution), and take an ϵ and a δ (two parameters governing our precision). Once we fix this parameters, we find a number m as a function of ϵ and δ that is telling us that we can learn with error bounded by ϵ the true function (assuming that the true function, the closest to the real one, belong to the set H). If we have data which is more than $1 - \delta$ points, the this is going to happen with high probability.
PAC literally means: probably approximately correct.

Definition 13. *H, A (the learning algorithm, i.e. the mechanism which returns the optimal function) is agnostic PAC learnable if and only if*

$$\begin{aligned}
&\forall \epsilon, \delta \in (0,1), \ \forall p(x,y) \\
&\exists m_{\epsilon,\delta} \in \mathbb{N} \ s.t. \ \forall m \geq m_{\epsilon,\delta}, \ \forall D \sim p^m, \ |D| = m \\
&R(h_D^A) \leq \min_{h \in H} R(h') + \epsilon \ \textit{with probability} \ 1 - \delta
\end{aligned}$$

Where h_D^A is the result of A on H, D. Observations:

- $m_{\epsilon,\delta}$: *typically we require it to depend polynomially from* $\frac{1}{\epsilon}, \frac{1}{\delta}$. *This is because we do not want a bound to depends on a huge number of observation: it has to increase moderately with the complexity*
- *A should run in polynomial time. This is because we want to limit the complexity of the algorithm.*

3.2 Learning finite hypothesis sets

Definition 14. *Class of finite hypothesis: H s.t.* $|H| < \infty$

The true solution may of may not be contained inside. Because we have a finite set of hypothesis function, there is always a way in which we can discriminate, or assign classes to points: our representation power is finite, and prior that we have enough samples, we should be able to cover up all possibilities.

Definition 15. *H s.t.* $|H| < \infty$ *is agnostic PAC learnable with:*

$$m_{\epsilon,\delta} \leq \left\lceil \frac{2\log\left(\frac{2|H|}{\delta}\right)}{\epsilon^2} \right\rceil \tag{3.2}$$

Where $\log|H|$ *is the measure of complexity of H.*

Remark. If H is described by d parameters of type `DOUBLE` (64 bits), then $|H| \leq 2^{d \cdot 64}$, then

$$m_{\epsilon,\delta} \leq \frac{128 \cdot d + 2\log\left(\frac{2}{\delta}\right)}{\epsilon^2} \tag{3.3}$$

3.3 VC dimension

In most of the cases, at least from a theoretical prospective, we would like to reason on a infinite classes of functions (e.g. lines on a plane, boxes in space etc.). We need to define a proper notion of complexity for a class of functions to be able to say something meaningful.

Example 2. *Let's consider the plane* $\mathbb{R}^2$*, and two class: axis lined lines and boxes. How many configurations of data points are those classes able to separate (to classify them correctly)?*

Let's take n points, and see if I can correctly classify them. To do so, I have to consider that all possible assignment could happen. So we have to essentially relativize a set of hypothesis function to a finite set of point. Here is a formal definition:

Definition 16. *Given a class of hypothesis functions* ($H = \{h : Z \leftarrow 0, 1\}$)*, a subset* C *of* X *of finite cardinality* ($C \subseteq X$, $|C| = m$, $C = \{c_1, ..., c_m\} \subseteq X$)*. We define H relativize to C (or H to C) as:*

$$H_C = \{(h(c_1), ..., h(c_m)) | h \in H\} \tag{3.4}$$

i.e. takes this points and apply one of my functions to this points and check which are the tuples of $\{0,1\}$ *we can generate.*

Remark. We say that H **shatters** C if and only if $|H_C| = 2^m$.

So we can define

Definition 17. *VC dimension:*

$$VCdim(H) = \max\{m | \exists C \subseteq X, |C| = m \ s.t. \ H \ shatters \ C\} \tag{3.5}$$

Its possible that the VC dimension is infinite.

3.4 VC dimension and PAC learning

Proposition 1. *If* H *shatters* C, $|C| \geq 2m$, *then we cannot learn* H *with* m *samples. So if* $VCdim(H) = \infty$, H *in not PAC learnable.*

i.e. there will be a specific true function that assign classes to n points such as we commit an error with high probability. This allow to state that if a set has infinite VC dimension, is not PAC learnable.
Infinite VC dimension means that whenever we have n points, whenever their assignment of class, we will always find a function which will captures exactly this assignment. Basically, every effect of noise will captured with empirical loss 0, so we can over-fit as much as we want.

Theorem 2. *H is (agnostic) PAC learnable if and only id the VC dimension (of H) is finite* ($< \infty$). *In this case* $\exists c_1, c_2$*:*

$$c_1 \frac{VCdim(H) + \log \frac{1}{\delta}}{\epsilon^2} \leq m_{\epsilon,\delta} \leq c_2 \frac{VCdim(H) + \log \frac{1}{\delta}}{\epsilon^2} \tag{3.6}$$

i.e. the VC dimension is ruling the bound.

3.5 Rademacher Complexity

Complexity of a set of functions, more general then the VC dimension.

Definition 18. *Given* $p(x,y)$, $D \sim p^m$, $H = \{h : H \to \{-1,1\}\}$, *and defined the rademacher distribution as*

$$\sigma = (\sigma_1, ..., \sigma_m), \ \sigma_1 \in \{-1,1\}, \ p(\sigma_1 = +1) = 0.5 \tag{3.7}$$

The rademacher complexity for dataset D is defined as

$$\hat{\mathcal{R}}_D(H) = \mathbb{E}_\sigma \Big[\sup_{h \in H} \frac{1}{m} \sum_{i=1}^{m} \sigma_i \cdot h(x_i) \Big] \tag{3.8}$$

i.e. $\sigma_i \cdot h(x_i)$ is the scalar product of the rademacher distribution with $h(x)$ evaluated on our dataset ($\frac{1}{m}\sigma_i \cdot h(x_i) \in [-1, 1]$) and is a measure of correlation between h and the random noise. If this value is 1, for the specific sample of random noise considered, we can obtain a perfect correlation, perfectly representing what we see.
For a specific σ we are going to choose the best h that correlates with that noise, and then take the expectation over the possible values of σ.
This is a property of both the function and of the dataset observed. We can define the rademacher complexity independent from the data, both rather dependent on their size.

Definition 19. *Rademacher complexity data independent:*

$$\mathcal{R}_m(H) = \mathbb{E}_{D\sim p^m}[\hat{\mathcal{R}}_D(H)] \tag{3.9}$$

Still depends on H and on p.

Let's state the PAC learning bounds on the rademacher complexity.
We fix H and $p(x, y)$; given $\forall\delta > 0$, with probability $\geq 1 - \delta$, for any $D \sim p^m$, $|D| = m$ and $\forall h \in H$, with have the following bound on the generalization error

$$R(h) \leq \hat{R}_D(h) + \mathcal{R}_m(H) + \sqrt{\frac{\log \frac{1}{\delta}}{2m}} \tag{3.10}$$

Where m is the number of data points. The last two terms play the role of ϵ. And the bound on the generalization error where we fix the data on the data-dependent rademacher complexity

$$R(h) \leq \hat{R}_D(h) + \mathcal{R}_D(H) + 3\sqrt{\frac{\log \frac{1}{\delta}}{2m}} \tag{3.11}$$

Where also the last two term play the role of ϵ.

3.6 Rademacher Complexity ad VC dimension

The links with VC dimension comes through the so called growth function. This function is telling us how the complexity grows increasing the cardinality of the number of data points we have.

Definition 20. *Grow function:*

$$\Pi_H : \mathbb{N} \to \mathbb{N} \qquad \Pi_H(m) = \max_{C\subseteq X, |C|=m} |H_C| \tag{3.12}$$

where $H_C = \{(h(c_1), ..., h(c_m)) | h \in H, C = \{c_1, ..., c_m\}\}$

We have that $VCdim(H) = \max\{m|\Pi_H(m) = 2^m\}$. If this is infinite, it means that the complexity of the function h allows us to find a set of an arbitrary large number of points that can be classified in an arbitrary way. The growth function is an intermediate step, we can bound growth function by an expression depending on the VC dimension, and than bound the rademacher complexity with an expression depending on the growth function. We can combine this bounds and see that there is a dependency depending on the relationship between the two.

Lemma 3. *The relationship between VC dimension and rademacher complexity is give by the Sauer Lemma:*

$$\Pi_H(m) \leq \sum_{i=0}^{d} \binom{m}{i} \leq \left(\frac{e \cdot m}{d}\right)^d = O(m^d) \tag{3.13}$$

where $d = VCdim(H)$

i.e. when d is finite, the growth function grows exponentially until d, then it start grows polynomially with a degrees that depends on the VC dimension set.
It holds that

$$\mathcal{R}_m(H) \leq \sqrt{\frac{2 \log \Pi_H(m)}{m}} \tag{3.14}$$

This different ways of measuring complexity are roughly similar, they gives us consistent results.

$$\mathcal{R}_m(H) \approx VCdim(H) \tag{3.15}$$

3.7 Empirical Risk Minimization and Maximum Likelihood

Quick Max-Likelihood recap

Given $D = \{(x_i, y_i)\}_{i=1,\dots,m}$, $D \sim p^m$ and $p = p(x, y)$, in the maximum likelihood scenario what we do is consider the data generating distribution, factorize it, and we typically make some hypothesis on the conditional distribution function (i.e. try to express it in a parametric form depending on a certain set of parameters θ, $p(y|x) = p(y|x, \theta)$).
In maximum likelihood we write the log-likelihood of the probability of the data given θ(i.e. observing y_i given x_i and a certain θ):

$$\mathcal{L}(\theta; D) = \sum_{i=1}^{m} \log p(y_i|x_i, \theta) \tag{3.16}$$

And we choose the parameter so that

$$\theta_{ML} = \arg\max_\theta \mathcal{L}(\theta; D) = \arg\min_\theta -\mathcal{L}(\theta; D) \tag{3.17}$$

Playing around a bit

$$\begin{aligned} \arg\min -\mathcal{L}(\theta; D) &= \arg\min_\theta -\frac{1}{m}\sum_{i=1}^{m} \log p(y_i|x_i, \theta) \\ &\approx \arg\min_\theta \mathbb{E}_{p(x,y)}[-\log p(y|x, \theta)] \end{aligned}$$

The expression $-\frac{1}{m}\sum_{i=1}^{m} \log p(y_i|x_i, \theta)$ is called cross-entropy.

Empirical Risk Minimization and Maximum Likelihood bridge

In the maximum likelihood framework we have a loss function

$$l(x, y, \theta) = -\log p(y|x, \theta) \tag{3.18}$$

But what is H?

- For regression $h(x, \theta) = \mathbb{E}_y[p(y|x, \theta)]$.
- For classification $h(x, \theta) = \arg\max_y p(y|x, \theta)$. This is the Bayes decision rule.

Choice of $p(y|x, \theta)$

- Regression $p(y|x, \theta) = \mathcal{N}(h_\theta(x), \sigma^2)$ (i.e. a normal distribution centered in some function, i.e. the expected value).
  This implies that
  $$-\log p(y|x, \theta) \propto (y - h_\theta(x))^2 \tag{3.19}$$
  Essentially the loss based on the sum of square comes out of a probabilistic assumption that there is a Gaussian noise that generates the observation, which mean is equal to the true value, with a certain standard deviation.

3.8 Introduction to Information Theory

Entropy: the idea is to use it to describe the average amount of information that is conveyed by something which has a probabilistic nature.

Definition 21. *Given a probability distribution $p(x)$, we define $-\log p(x)$ as a measure of self information.*

i.e. Let's imagine that $p(x) = 1$, so x is always happening. There is no information conveyed by $p(x)$ because we know that it is always happening. If $p(x) = 0$, then if x happens there should be an infinite quantity of information: something is very wrong in our model. When the probability distribution is very small, the self information carried is very large: the event is very unexpected. We want to attach a measure of information to the distribution itself: the expectation of the self information, the entropy.

Definition 22. *Entropy: $\mathcal{H}[p] = \mathbb{E}_p[-\log p(x)] = -\int p(x) \log p(x) dx$*
Entropy is maximum in the continuum case comes for a given fix minimum variance, the normal of Gaussian random variable.
In the discrete case: $\mathcal{H}[p] = -\sum_i p(x_i) \log p(x_i)$
Entropy is maximum in the discrete case for the uniform distribution, when $= \log K$, where K is the amount of different events that can happen

Definition 23. *Kullback Leibler divergence, or relative entropy: Given p,q*

$$KL[q\|p] = \int q(x) \log \frac{q(x)}{p(x)} dx \tag{3.20}$$

The discrete case is the same but with a sum.

i.e. it is a sort of expected difference between p and q. If they are the same p and q will always be equal, and the ration will always be equal to 1, so $KL = 0$:

$$KL[q\|p] = 0 \text{ iff } q = p \tag{3.21}$$

Also KL is a convex functional and $KL \geq 0$. The larger the KL divergence, the more the two distributions are different. The worst case scenario is when one of the two distribution is 0, so KL divergence explodes to infinity.
We can rewrite the KL divergence by splitting the logarithm in a difference.

$$KL[q\|p] = \int q(x) \log \frac{q(x)}{p(x)} dx = -\mathcal{H}[q] - \mathbb{E}_q[\log p] \tag{3.22}$$

KL divergence is important because it is essentially a measure of distance of the probability distributions. The best match of two probability distribution can be obtained by minimizing the KL divergence.
Given a fixed but known p, and a $q = q_\theta$ (depending on a set of parameters θ) that can vary, one could ask himself what is the best q_θ which could approximate p. And answer could be

$$\theta^* = \arg\min_\theta KL[q_\theta\|p] \tag{3.23}$$

This is the base of what is called **variational inference**.
If we have two random variables x,y and we compute the KL divergence between the joint distribution and the product of marginal distribution, than it will be 0 when the two variables are independent, and the more dependent the are (the more information x carries about y and vice versa), the great it will be. This is called the mutual information between x and y.

Definition 24. *Mutual information between x and y*

$$\mathcal{I}[x, y] = KL[p(x, y)\|p(x)p(y)] \tag{3.24}$$

Now let's consider a set of data points $x = x_1, ..., x_N$.

Definition 25. *Empirical distribution:*

$$p_{emp}(x) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(x = x_i) \tag{3.25}$$

i.e. give probability mass on the observation (kind of constructing an histogram). We want a nice q to approximate x. We can compute the KL divergence between the empirical distribution, and the distribution q

$$KL[p_{emp}\|q] = \overbrace{\mathbb{E}_{p_{emp}}[-\log q(x)]}^{-\frac{1}{N}\sum_i \log q(x_i)} - \mathcal{H}[p_{emp}]$$

If $q = q_\theta$

$$-\frac{1}{N}\sum_i \log q(x_i) = -\frac{1}{N}\mathcal{L}''(\theta) \tag{3.26}$$

So maximising $\mathcal{L}(\theta)$ is equivalent to minimizing the KL divergence between p_{emp} and q_θ

Chapter 4

Probabilistic Graphical Models

4.1 Introduction

Talking about generative model, we have a certain set of random variates (e.g. $x = (x_1, ..., x_n)$, $z = (z_1, ...z_m)$). Having a probabilistic model of the world means having a joint probability distribution (e.g.$p(x, z)$, known).
Typically in practical scenario, we want to reason on this joint distribution. We either want to compute **marginal probability distribution**: $p(x) = \int p(x, z)dz$ if z are discrete $p(x) = \sum_z p(x, z)$. From a larger amount of information, we are interested in a specific subset of it.
Another typical operation we perform is **conditioning**: $p(z|x = \underline{x}) = \frac{p(\underline{x},z)}{p(\underline{x})}$, with $\underline{x}$ fixed values.
Or we can do **both** (marginalization and conditioning):

$$p(z_j|x_i = \underline{x}_i) = \frac{p(\underline{x}_i, z_j)}{p(\underline{x}_i}) = \frac{\int p(x, z)dx_{-i}dz_{-j}}{\int p(x, z)dx_{-i}dz}$$

This procedures are generalizing in a certain sense the concept of logical inference for boolean variables, to the probabilistic world of probability distributions.
What is the cost in terms of computation (considering discrete random variables, $z = z_1, ..., z_m$ with $z_i \in 0, 1$ boolean) of: $\int p(x, z)dz = p(x) = \sum_{z \in Z} p(x, z)$?
Let's remember that $z \in Z$ which has cardinality $|Z| = 2^m$, which is not an innocuous number (e.g. $m = 100 \leftarrow 2^m \approx 10^{30}$). We need then a more clever way to deal with data.
Let's introduce the concept of **factorization**, applying basic laws of probability:

$$\begin{aligned} p(x_1, .., x_n) &= p(x_2, ..., x_n|x_1)p(x_1) \\ &= p(x_3, ..., x_n|x_1, x_2) \cdot p(x_2|x_1)p(x_1) \\ &= p(x_4, ..., x_n|x_1, x_2, x_3)p(x_3|x_1, x_2)p(x_2|x_1)p(x_1) \\ &= p(x_n|x_1, .., x_{n-1})p(x_{n-1}|x_1, ...x_{n-1})...p(x_3|x_1, x_2)p(x_2|x_1)p(x_1) \end{aligned}$$

That is a factorization: representing a full joint probability distribution as the product of factors. It may happen that not all the terms in the conditioning set are relevant to

describe the conditional distribution we are interested in.
In some cases factorization can be simpler:

$$p(x_1, ..., x_n) = p(x_n|x_{n-1})p(x_{n-1}|x_{n-2})...p(x_3|x_2)p(x_2|x_1)p(x_1) \tag{4.1}$$

That for example is particularly lucky factorization.
Suppose our variables are categorical, and can assumes value in the range $x_i \in 1, ..., k$. What is the cost of storing the values of a single probability distribution on x?
Using an array, storing the probability for each values of $p(x_i)$ (i.e. 1 with probability p_1, 2 with probability p_2, so on until $p_k = 1 - \sum_{i=1}^{k-1} p_i$, we need $k - 1 = O(k)$ doubles.
Suppose now we want to store $p(x_2|x_1)$. We are building this time a table, in which we have probability $p_{i,i}$ for each one of the categorical values. Each rows in this configuration costs $k - 1$, and in total we have k rows. So we need $k(k.1) = O(k^2)$ doubles.
So, the cost for every representation is:

- $p(x_1, .., x_n)$: cost $O(k^n)$
- $p(x_n|x_1, .., x_{n-1})p(x_{n-1}|x_1, ...x_{n-1})...p(x_3|x_1, x_2)p(x_2|x_1)p(x_1)$: each of the terms is of the order $O(k^n)$ where n is the number of variables considered for the evaluation, so again the total number of space required is of the order of $O(k^n)$
- $p(x_n|x_{n-1})p(x_{n-1}|x_{n-2})...p(x_3|x_2)p(x_2|x_1)p(x_1)$: we always have only two variates for all of the n conditional distributions, so the total number of space required is of the order $O(nk^2)$

The main kind of models that we are going to consider:

- Bayesian networks
- Markov random fields
- (Factor graph)

They are all graphs: they are composed of nodes and edges that can be directed or undirected. If all edges are directed and the graphic does not have cycle, we are talking about a Directed Acyclic Graphs used in Bayesian Network, if at the opposite the graph is undirected and can have cycle, we are talking about Random Markov fields. They are different ways to factorize a probability distribution function, based on different assumptions:Bayesian usually are more applicable, but usually inference between all of them are the same.
E.g. of a Bayesian Network:

$$p(x_1, x_2, x_3, x_4) = p(x_4|x_3)p(x_3|x_1, x_2)p(x_2)p(x_1) \tag{4.2}$$

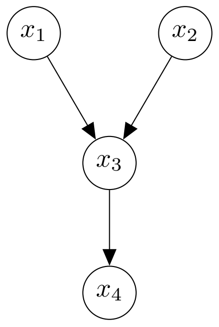


Figure 4.1: Bayesian network

4.2 Bayesian Networks

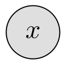
Definition 26. *pa_k: parents of x_k, i.e. $pa_k \subseteq x_n, ..., x_{k-1}$ are in the conditioning set of x_k, and $pa_k \Rightarrow p(x_k|pa_k)$ which is the factor of x_k*

Definition 27. *Bayesian network is a DAG with:*

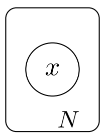
- *$\{x_1, ..., x_n\}$ vertices*
- *$(x_i, x_j), i < j$ and $x_i \in pa_j$ edge. We cannot have cycle.*

There are three conventions for the graphic representation of Bayesian Network:

- *Observed nodes are drawed colored.*



- *Plated node (or subgraph)*



Where N represent the number of different instances of the node.

- *Deterministic quantities (or fixed) are solid circles (but I will use only the name of the variable).*

4.3 Sampling and Reasoning in Bayesian Networks

Definition 28. *Ancestral sampling: if the factorization is feasible, it means sampling follows the bayesian network from top to bottom (i.e. sample the ancestor, then progressively the descendants). It is possible only if we know how to sample effectively from the probability of x given its parents ($p(x|pa_x)$*

Example 3. *Let's consider a Bayesian network with the aim of predicting the grade of an exam, in which we define the node:*

- *D = difficulty*
- *I = intelligence*
- *G = grade*
- *P = passed a hard exam*

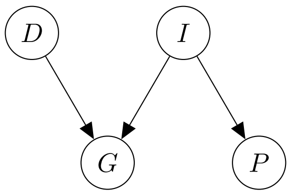


Figure 4.2: Bayesian network for grading prevision

Here are different kids of reasoning:

- *Casual reasoning: we observe the nodes on top, and want to compute the marginal probability of the variables which depends on what we observe. Information is flowing from top to bottom.*

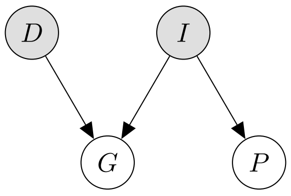


Figure 4.3: Bayesian network for grading prevision, causal reasoning.

- *Evidential reasoning: information flow from bottom to top. We observe the consequenses of something and try to infer the probability of the causes of the observation.*

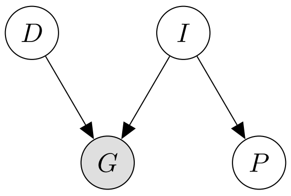


Figure 4.4: Bayesian network for grading prevision, evidential reasoning.

- *Intercausal reasoning: a sum of the two. The information goes both up to down and viceversa.*

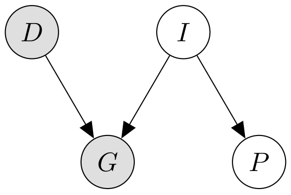


Figure 4.5: Bayesian network for grading prevision, intercausal reasoning

.

4.4 Naive Bayes

We have a certain set of features $x_1, ..., x_n$ that we observe. This observation are going to depend on class (the class of the observation we are going to have) $p(x_1, ..., x_n|c)$. This is a generative model which is **class conditional** (probability of observing certain data if it comes from a certain class. Solving a classification task, we want to compute $p(c|x_1, ..., x_n)$. Now $x_1, ..., x_n$ are no longer independent observations, but the features of our model.
Modelling this probability is potentially very difficult, so we make a simple assumption: $x_1, ..., x_n$ are independent among them given the class, which corresponds to the following network

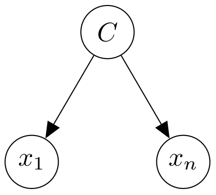


Figure 4.6: Naive Bayes structure (for all nodes of x

Which can be written factorize as

$$p(x_1, ..., x_n|c) = \prod_{i=1}^{n} p(x_i|c) \tag{4.3}$$

This makes training the model much easier: we have observation for class C, and a certain number of observation in the train set. For each of this observation we focus only on the feature x_i, so given c, we fix x_i consider observations from the data we have on feature x_i for points of class c and fit a model parametric of $p(x_i|c, \theta)$ (i.e. it depends on a certain number of parameters), and we fit our parameter θ. Because we are fitting probabilistic models of individual variables, usually the fit is pretty easy.
We can compute now $p(c|x_1, ..., x_n) \propto p(c) \cdot p(x_1, ..., x_n|c) = p(c)\prod_i p(x_i|c)$, but not precisely (we still need some marginalization, and an estimate of $p(c)$).
Once we have fitted the model, and we have a certain observation, for example of two classes, we can compute the ration:

$$\frac{p(c = 0|x_1, ..., x_n)}{p(c = 1|x_1, ..., x_n)} \tag{4.4}$$

In the classification we choose the class which is dominating good for classification but not for estimating the probability of c given the observation.

4.5 Conditional Independence

Definition 29. *Let's consider a, b, c random variables. We say that a is conditional independent of b given c ($a \perp\!\!\!\perp b|c$) if and only if*

$$p(a|b,c) = p(a|c) \quad or \quad p(a,b|c) = p(a|c)p(b|c) \tag{4.5}$$

There are three scenario that we need to consider depending on how the arrows connect the three nodes (a, b and c).

First scenario: tail to tail

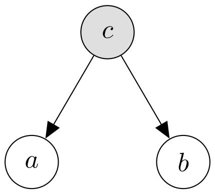


Figure 4.7: Bayesian network first scenario

In the path from a to b passing through c the arrows are tail to tail facing c. Let's compute

- $p(a,b) = \sum_c p(a|c)p(b|c)p(c) \neq p(a)p(b)$ so a and b are not conditional independent (if c is not observed).
- $p(a,b|c) = \frac{p(a,b,c)}{p(c)} = \frac{p(a|c)p(b|c)p(c)}{p(c)} = p(a|c)p(b|c)$ so a and b are conditional independent (if c is observed).

c is the root cause of a and b, so if we do not know c, there may be some relationship between them. Once we observe c, all the dependency is contained in c.

Second scenario: head to tail

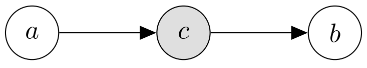


Figure 4.8: Bayesian network second scenario

So as before

- $p(a,b) = \sum_c p(b|c)p(c|a)p(a) = p(b|a)p(a) \neq p(a)p(b)$ so a and b are not conditional independent (if c is not observed).
- $p(a,b|c) = \frac{p(a,b,c)}{p(c)} = \frac{p(b|c)p(c|a)p(a)}{p(c)} = p(b|c)p(a|c)$ so a and b are conditional independent (if c is observed).

a causes c which causes b. Once we observe something in the middle, b is not going to be conditional independent on anything before c, because c already bring all the information necessary for b.

Third scenario: head to head

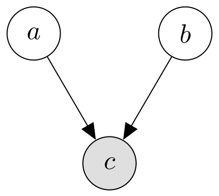


Figure 4.9: Bayesian network third scenario

So as before

- $p(a,b)=\sum_c p(a)p(b)p(c|a,b)=p(a)p(b)$ so a and b are conditional independent (if c is not observed).
- $p(a,b|c)=\frac{p(a,b,c)}{p(c)}=\frac{p(a)p(b)p(c|a,b)}{p(c)}\neq p(b|c)p(a|c)$ so a and b are not conditional independent (if c is observed). Also a and b are not conditional independent given d for d a descendent of c.

a and b are independent, but if we observe something downstream, that it is consequence of both a and b how may introduce some information which may be propagated back.

Exploring graph structure of the Bayesian network

Let's consider three node a, b and c. A path from a to b is blocked by c if and only if

- c is observed and the path is head to tail or tail to tail in c.
- c is not observed nor any descendent of c and the path is head to head in c.

Now let A, B and C subsets of nodes disjointed, if all path from a node in A to a node in B are blocked by a node in C than

$$A \perp\!\!\!\perp B|C \tag{4.6}$$

Definition 30. *Markov blanket: let's consider a node x_i, we will condition all the rest x_{-i}, which nodes remain in the conditional set (this nodes are the markov blanket)?*

$$p(x_i|x_{-i})=\frac{p(x_1,...,x_i,...,x_n)}{p(x_1,...,x_{i-1},x_{i+1},...,n_n)}=\frac{\prod_i p(x_i|pa_j)}{\sum_{x_i}\prod_j p(x_j|pa_j)} \tag{4.7}$$

Summing over x_i (denominator), which of the terms will depend on x_i? Either x_i is x_j, or x_i belongs to the parents set $x_i\in pa_j$.
So the Markov blanket of x_i is made of:

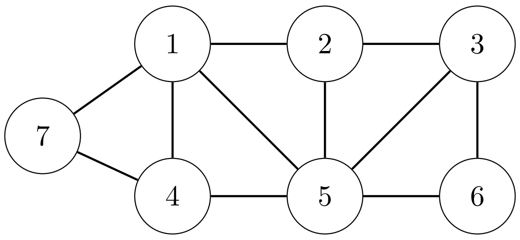
- pa_j

- *Children*
- *Co-parents*

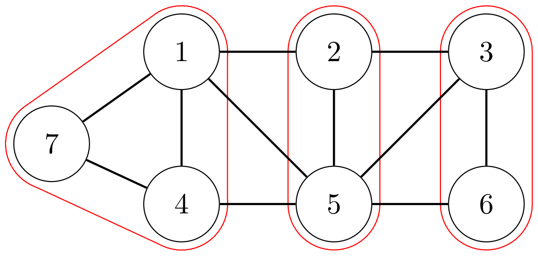
The Markov blanket "covers" x_i, and make it independent.

4.6 Markov Random Fields

Markov Random Fields are undirected graphs (i.e. graph in which edges has no arrow)



Where nodes are variables, and edges represent a sort of dependency (there is no direct correspondence between graph structure and the factorization implied). Let's see how conditional independence is represented in a markov random fields.



Let's considered three subset of nodes: $A = \{1, 4, 7\}$, $B = \{3, 4\}$ and $C = \{2, 5\}$. We can say that: $A \perp\!\!\!\perp B|C \Leftrightarrow$ all paths from a node $a \in A$ to any node $b \in B$ pass from C (are blocked by a node in C).
Markov blanket of x_i in this context is the st of neighbours of x_i.
When two nodes are conditionally independent given the rest of the nodes ($x_i \perp\!\!\!\perp x_j | x_{-\{i,j\}}$)?
When there is no path from x_i to x_j passing from any other node.
So $x_i \perp\!\!\!\perp x_j | x_{-\{i,j\}}$ if and only if there is no edge between them. From this we can deduce that

$$p(x_i, x_j | x_{-\{i,j\}}) = p(x_i | x_{-\{i,j\}}) p(x_j | x_{-\{i,j\}}) \tag{4.8}$$

i.e. joint probability factorize as the product of marginal probability conditioning on all the other nodes.
But that means that x_i and x_j must belong to different factors. Nodes which are not connected by edges should belong to different factors.
From this derives that factors in a random markov field is equivalent to a maximal cliques in the graph.

Definition 31. *Cliques: subgraph fully connected.*
Maximal Cliques: cliques that cannot be extended

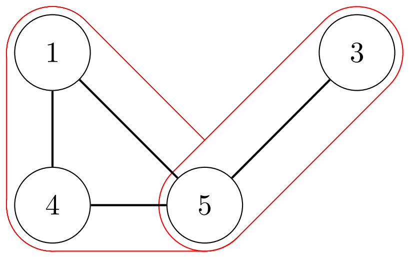


Figure 4.10: Markov Random Fields with two cliques overlapping on node 5

The reason why factors corresponds to cliques is that if we are connected on an edge, even if they are conditioning on all the rest, we remain independent.

Definition 32. *Defined $C = \{$set of maximal cliques$\}$ and $x_c = \{x | x \in C \in \mathcal{C}\}$ (where $\mathcal{C}$ is the set of maximal cliques). Given $x = x_1, .., x_n$*
Factorization in a MRF:

$$p(x) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \psi_c(x_c) \tag{4.9}$$

i.e. the product over the maximal cliques of some function ψ_c evaluated on the subset of the variables of x that belong the the clique; over a constant Z which guarantees the normalization into a probability density.

*$\psi_c(x_c) \geq 0$ and have a finite integral $\int \psi_c(x_c) dx_c < \infty$, ψ_c is a **potential function**.*

*$Z = \int \prod_c \psi_c(x_c) dx$ is called **partition function** (but it is actually a constant).*

Z usually is hard to compute. If variables are discrete with k states $x_i \in \{1, ..., k\}$, then computing Z means:

$$Z = \sum_x \prod_c \psi_c(x_c) \tag{4.10}$$

with the sum on a subset of k^n states. Practically impossible. And in the case of the integral it is the same. But

- If we want to compute conditionals, then Z cancels out.
- If we want to compute marginal of few variables, we can work with potentials and normalize at the end (i.e. let's imagine that when we are working with potentials we may compute this one dimensional marginal. This one dimensional is an unnormalized marginal, so we have potentials, and numbers which does not sum up to 1, but we have to sum k numbers, which is easy).

Potentials can be defined in any way, they just have to be positive. A way to guarantee that something is positive, is to model it as the exponential of a real number. So by convention we usually model the potential function as

$$\psi_c(x_c) = exp(-E(x_c)) \tag{4.11}$$

which is known as the **Boltzmann distribution**, where $E(x_c)$ is the **energy**. States of low energy corresponds to states of high potential. It is convenient to work with Boltzmann models per the easiness of factorization

$$\begin{aligned}\psi_{c_1}(x_{c_1})\psi_{c_2}(x_{c_2}) &= exp(-E_1(x_{c_1}))exp(-E_2(x_{c_2})) \\ &= exp(-E_1(x_{c_1}) - E_2(x_{c_2}))\end{aligned}$$

Chapter 5

Inference in PGM

5.1 Introduction

5.2 Factor Graphs

5.3 Sum-Product algorithm

5.4 Max-Plus algorithm

5.5 Inference for general PMGs