UniTs - University of Trieste

Faculty of Scientific and Data Intensive Computing

Department of mathematics informatics and geosciences

Probabilistic Machine Learning

Lecturer:
Prof. Luca Bortolussi

Authors:
Andrea Spinelli
Christian Faccio

April 15, 2025

github.com/Spina02 andreaspinelli2002@gmail.com

Preface

As a student of Scientific and Data Intensive Computing, I've created these notes while attending the **Probabilistic Machine Learning** course.

This collection introduces ideas and instruments of machine learning from a probabilistic perspective—an approach that continues to grow in importance and serves as the foundation for many recent successes in generative Artificial Intelligence. The notes begin with a discussion on the fundamental role of probability and mathematics in machine learning, providing a framework for understanding ML concepts through this powerful lens.

Throughout the course, we will focus on the following topics:

- Basics of probability and probabilistic inference
- Probabilistic formulation of learning (Empirical Risk Minimization and PAC Learning)
- Graphical Models
- Inference with graphical models: belief propagation
- Hidden Markov Models for sequential data
- Bayesian Linear Regression and Classification, Laplace approximation, Model Selection
- Kernel Regression and Kernel functions, Gaussian Processes for regression (hints)
- Monte Carlo sampling
- Expectation Maximization and Variational Inference
- Bayesian Neural Networks
- Generative Modelling: Variational Autoencoders and Diffusion Processes

The structure of these notes follows the natural progression from fundamental probabilistic concepts to advanced generative models, emphasizing both theoretical foundations and practical applications. While these notes were primarily created for my personal study, they may serve as a valuable resource for fellow students and professionals interested in probabilistic machine learning.

Contents

1 Introduction

1.1 Models and Probability

Machine learning is a field of computer science about **learning models**.

Definition: *Machine learning*

Machine learning *explores the study and construction of algorithms that can learn from and make predictions on data.* *Wikipedia*

If we dig into this statement, we may wonder what precisely do ML algorithms learn.

The answer to this question is (apparently) simple: they learn models of the observed data. Models that can then be used to make predictions or to extract information from such data.

Definition: *Model*

- A ***Model*** is a hypothesis that certain features of a system of interest are well replicated in another, simpler syetem.
- ***Mathematical Model*** is a model where the simpler system consists of a set of mathematical relations between objects (equations, inequalities, etc).
- A ***Stochastic Model*** is a mathematical model where the objects are probability distributions.

All modelling usually starts by defining a family of models indexed by some parameters, which are tweaked to reflect how well the feature of interest are replicated.

Machine learning deals with algorithms for automatic selecrion of a model from observations of the system.

Generative and Discriminative Learning

- **Generative Learning** im at describing the full probability distribution of inputs x or input/output pairs (x,y).
$$p(x,y) = p(x)p(y|x)$$
- **Discriminative Learning** aims at describing the conditional probability of output given the input, or a statistics/function of such probability
$$p(y|x) \quad or \quad y = f(x)$$

[to fix:]

- **Supervised Learning**: The algorithm learns from labeled data by mapping inputs to outputs.
- **Unsupervised Learning**: The algorithm identifies patterns or structures in unlabeled data.
- **Data Generatiion**: The algorithm generates new data points.

Inference and Estimation

Two central concepts for probabilistic machine learning are:

- **Inference**: Compute marginals and contitionals probability distributions applying the laws of probability.
- **Estimation**: Given data and a family of models, find the best parameters/models that explains the data.

In the Bayesian world: estimation $\approx$ inference.

Probability

Probability is a mathematical theory that deals with **uncertainty**

When a certain problems has to face practical difficulties due to it's complexity, we can use probability to model the ***aleatorical uncertainty***, which is the uncertainty due to the randomness of the system.

More often, we have a limited knowledge of the system, and we can use probability to model the ***epistemic uncertainty***, which is the uncertainty due to the lack of knowledge.

Tip: ***Everything is a probability distribution***

In machine learning **everything is a probability distribution**, even if not explicitly stated.

1.2 Probability basics

1.2.1 Random Variables

Random Variables are functions mapping outcomes of an experiment to real numbers. They serve as abstract representations of the outcomes in randomized experiments. Note that what we observe are the *realizations* (values resulting from an observed outcome) of these random variables.

We consider a ***Sample Space*** Ω, which is the set of all possible outcomes of a random experiment. A random variable X is a function:

$$X : \Omega \to E, \qquad \text{where } E \subseteq \mathbb{R} \quad (\text{or } E \subseteq \mathbb{N})$$

with the probability measure

$$P(X \in S) = P(\{\omega \in \Omega \mid X(\omega) \in S\}), \quad S \subseteq E.$$

A model for our random outcome is the probability distribution of X. In particular, if the sample space is finite or countable the **probability mass function (pmf)** is given by:

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

If the sample space is infinite, we use the **probability density function (pdf)** where

$$P(a \leq X \leq b) = \int_a^b p(x)dx \quad \text{and} \quad \int_{\mathbb{R}} p(x)dx = 1.$$

1.2.2 Notable Probability Distributions

Below are some of the most common probability distributions.

Discrete Distributions

Distribution	pmf	Mean	Variance
Binomial $\text{Bin}(n,p)$	$\binom{n}{x} p^x (1-p)^{n-x}$	np	$np(1-p)$
Bernoulli $\text{Bern}(p)$	$p \quad (x=1), \quad 1-p \quad (x=0)$	p	$p(1-p)$
Discrete Uniform $\mathcal{U}(a,b)$	$\frac{1}{b-a+1}$	$\frac{a+b}{2}$	$\frac{(b-a+1)^2-1}{12}$
Geometric $\text{Geom}(p)$	$(1-p)^{x-1}p$	$\frac{1}{p}$	$\frac{1-p}{p^2}$
Poisson $\text{Pois}(\lambda)$	$\frac{\lambda^x e^{-\lambda}}{x!}$	λ	λ

Continuous Distributions

Distribution	pdf	Mean	Variance
Continuous Uniform $\mathscr{U}(a,b)$	$\begin{cases} \frac{1}{b-a} & x \in [a,b] \\ 0 & \text{otherwise} \end{cases}$	$\frac{a+b}{2}$	$\frac{(b-a)^2}{12}$
Exponential $\text{Exp}(\lambda)$	$\lambda e^{-\lambda x}$	$\frac{1}{\lambda}$	$\frac{1}{\lambda^2}$
Gaussian $\mathscr{N}(\mu,\sigma^2)$	$\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$	μ	σ^2
Beta $\text{Beta}(\alpha,\beta)$	$\frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}$	$\frac{\alpha}{\alpha+\beta}$	$\frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$
Gamma $\text{Gamma}(\alpha,\beta)$	$\frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}$	$\frac{\alpha}{\beta}$	$\frac{\alpha}{\beta^2}$
Dirichlet $Dir(\alpha)$	$\frac{1}{B(\alpha)} \prod_{i=1}^{K} x_i^{\alpha_i - 1}$	$\tilde{\alpha}_i$	$\frac{\tilde{\alpha}_i(1-\tilde{\alpha}_i)}{\alpha_0+1}$
Student's t $St(\nu)$	$\frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\,\Gamma\left(\frac{\nu}{2}\right)} \left(1+\frac{x^2}{\nu}\right)^{-\frac{\nu+1}{2}}$	0	$\begin{cases} \frac{\nu}{\nu-2} & \nu > 2 \\ \infty & 1 < \nu \leq 2 \end{cases}$

Notes:

- For discrete distributions, $n \in \{0,1,2,\dots\}$, $p \in [0,1]$, and x runs over the support.
- For continuous distributions, parameters such as λ, μ, σ, α, and β belong to $\mathbb{R}$ (with appropriate restrictions) and $x \in \mathbb{R}$.
- In the Dirichlet distribution, $\tilde{\alpha}_i = \frac{\alpha_i}{\sum_{h=1}^{K} \alpha_h}$ and $\alpha_0 = \sum_{i=1}^{K} \alpha_i$.
- For Student's t-distribution, $\nu > 1$.

2 ERM and PAC Learning

In this chapter, we will introduce the concept of **Empirical Risk Minimization** (ERM) in which to frame learning problems, the notion of inductive bias, and the main results of algorithmic learnability, encapsulated in the definition of **Probably Approximately Correct** (PAC) Learning and of complexity of a set of hypothesis, namely VC-dimension and Rademacher complexity.

2.1 Empirical Risk Minimization

We begin by considering a supervised learning setting in which the **input space** X is a subset of $\mathbb{R}^n$, and the **output space** Y can be real-valued (e.g., $Y = \mathbb{R}$), binary (e.g., $Y = \{0,1\}$), or a finite set of classes (e.g., $Y = \{0,1,\ldots,K\}$). In this probabilistic framework, each input-output pair (x,y) is drawn from a joint probability distribution

$$p(x,y) \in \text{Dist}(X \times Y),$$

often referred to as the ***data generating distribution***.

By definition, this distribution factors into the marginal $p(x)$ and the conditional $p(y \mid x)$, so that

$$p(x,y) = p(x)\,p(y \mid x).$$

Because $p(x)$ and $p(y \mid x)$ describe how inputs and outputs are related, it is helpful to write them explicitly. The marginal distribution of x is

$$p(x) = \int p(x,y)\,dy,$$

while the conditional distribution of y given x is

$$p(y \mid x) = \frac{p(x,y)}{p(x)}.$$

A typical dataset D in supervised learning consists of N input-output pairs drawn independently from $p(x,y)$. We denote this as

$$D \sim p^N(x,y),$$

which means

$$D = \{(x_i,y_i) \mid i = 1,\ldots,N\},$$

where each (x_i,y_i) is sampled according to the joint distribution $p(x,y)$.

In many cases, we assume that $p(y \mid x)$ depends on some unknown function of x. Formally, one might write

$$p(y \mid x) = p\big(y \mid f(x)\big),$$

where f is the function we aim to learn. The central objective in supervised learning—through methods such as empirical risk minimization—is to find or approximate this function f by using the observed data D.

2.2 Risk and Empirical Risk

$h \in \mathscr{H} \quad x,y \sim p(x,y)$

loss function $l(x,y,h) \in \mathbb{R}_{\geq}$,

- 0-1 loss: $l(x,y,h) = \mathbb{I}(h(x) \neq y)$, eith $y \in \{0,1\}$.

- squared loss: $l(x,y,h) = (h(x)-y)^2$, with $y \in \mathbb{R}$.

We have a probabilistic process, so we have some inputs that are more likely than others. If a model makes a mistake on a more likely input, it should be penalized more.

Definition: ***Risk***

The ***risk*** (or ***generalization error***) is defined as:

$$R(h) = E_{x,y\sim p(x,y)}[l(x,y,h)]$$

Risk minimization principle:

The goal is to find the hypothesis h that minimizes the risk.

$$\text{find } h^* \in \mathscr{H} \text{ such that } h^* = \arg\min_{h\in\mathscr{H}} R(h)$$

Definition: ***Empirical Risk***

The ***empirical risk*** (or ***training error***) is defined as:

$$\hat{R} = \frac{1}{N}\sum_{i=1}^{N} l(x_i,y_i,h)$$

Empirical risk minimization principle:

The goal is to find the hypothesis h that minimizes the empirical risk.

$$\text{find } h_D^* = \arg\min_{h\in\mathscr{H}} \hat{R}(h)$$

2.2.1 Bias Variance Trade-off

In this section, we want to analyze the generalization error and decom- pose it according to the sources of error that we are going to commit.

In what follows, we will use the squared loss (hence we will focus on regression problems). Considering $h \in \mathscr{H}$, an explicit expression of the generalization error committed when choosing hypothesis h is:

$$R(h) = E_p[l(x,y,h)] = \int\int (h(x)-y)^2 p(x,y)dxdy$$

Theorem 1. *The minimizer of the generalization error R is:*

$$\boxed{g(x) = E[y|x] = \int y p(y|x)dy}$$

so that $g = \arg\min_h R(h)$, *if* $g \in \mathscr{H}$

We can rewrite the risk as:

$$R(h) = \underbrace{\int (h(x)-g(x))^2 p(x)dx}_{=0 \quad iff \quad h(x)=g(x)} + \overbrace{\iint (g(x)-y)^2 p(x,y)dxdy}^{\text{independent of } h\text{: intrinsic noise}}$$

$$\begin{aligned} E_D[R(h_D^*)] &= \underbrace{\int (E_D[h_D^*(x)-g(x)])^2 p(x)dx}_{bias^2} \\ &+ \underbrace{\int E_D[(h_D^*(x)-E_D[h_D^*(x)])^2 p(x)dx]}_{variance} \\ &+ \underbrace{\iint (g(x)-y)^2 p(x,y)dxdy}_{noise} \end{aligned}$$

2.3 ERM and Maximum Likelihood

Given a dataset $D = \{(x_i, y_i)\}_{i=1,\dots,m}$ s.t. $D \sim p^m, p = p(x,y)$

We factorize the data generating distributions as: $p(x,y) = p(x)p(y|x)$ and we make an hypothesis on $p(y|x)$, trying to express this conditional probability in a parametric form:

$$p(y|x) = \underbrace{p(y|x,\theta)}_{\text{parametric family of distributions}}$$

where θ is the parameter of the model.

We consider the log Likelihood:

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

Then we apply the maximum likelihood principle, according to which:

$$\Theta_{\text{ML}} = \arg\max_{\theta} L(\theta;D) = \arg\min_{\theta} -L(\theta;D)$$

It holds that:

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

since the avarage is an empirical approximation of the expectation.

Definition: ***Cross-Entropy***

The ***cross-entropy*** is defined as:

$$-\frac{1}{m}\sum_{i=1}^{m} \log p(y_i|x_i,\theta)$$

2.4 KL Divergence

From a physical point of view, entropy is a measure of disorder of a system, while from a probabilistic point of view is a measure of "surprise".

A measure, called **self-information**, of a probability distribution p(x) is given by the negative of the logarithm of the probability of the event:

$$I(x) = -\log p(x)$$

Indeed, if $p(x) = 1$, then $I(x) = 0$, while if $p(x) = 0$, then $I(x) = \infty$. In general, the more rare the event is, i.e. the lower is $p(x)$ the higher is the self-information, i.e. the larger is $-\log p(x)$.

In an information-theoretic sense, the **entropy** is a measure of the information that is carried by a random phenomenon, expressed as the expected amount of self-information that is conveyed by a realization of the random phenomenon.

Entropy is formally defined as:

$$\mathbb{H}[p] = \mathbb{E}_p[-\log p(x)] = \begin{cases} -\int p(x)\log p(x)dx & \text{(if } x \text{ is continuous)} \\ -\sum_i p(x)\log p(x_i) & \text{(if } x_i \text{ is discrete)} \end{cases}$$

In the discrete case, the maximum entropy is achieved for the uniform distribution and it is equal to $\log K$, with K number of events that can happen. In the continuous case, for a fixed variance, the distribution that maximizes entropy is the Gaussian. The entropy is always 0 if we have a deterministic distribution.

Definition:

The **Kullback-Leibler divergence** is a measure of how one probability distribution diverges from a second, expected probability distribution. It is defined as:

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

Intuitively, we are taking a sort of expected difference between p and q, expressed in terms of a log odds ratio. It tells us how different the two distributions are. The larger the KL divergence, the more different the two distributions are.

Properties of KL divergence

- $\mathbb{KL}[p||q] = 0$ iff $p = q$.
- $\mathbb{KL}[p||q]$ is a convex function of q and p and $\mathbb{KL}[p||q] \geq 0$
- $\mathbb{KL}$ is non-symmetric: $\mathbb{KL}[p||q] \neq \mathbb{KL}[q||p]$
- $\mathbb{KL}[p||q] = -H[q] - \mathbb{E}_p[\log q(x)]$, where the first term is the entropy of q and the second term is the cross-entropy.

…

2.4.1 KL Divergence and Maximum Likelihood

Consider a dataset: $\underline{x} : x_1, ..., x_N$:

Definition:

The **empirical distribution** is defined as:

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

It is an approximation of the input data generating function $p(x)$. Practically, the more observations we have, the better the approximation.

Given a distribution q, we can compute:

$$\mathbb{KL}[p_{emp}||q] = \mathbb{E}_{p_{emp}}[-\log q(x)] - \mathbb{H}[p_{emp}] = \underbrace{-\frac{1}{N}\sum_{i=1}^{N} \log q(x_i)}_{-\frac{1}{N}L(q,D)} - \mathbb{H}(p_{emp})$$

if $q = q_0$, this is $-\frac{1}{N}L(\theta)$ plus a constant. Hence, maximising $L(\theta)$ is equivalent to minimizing the KL divergence between the empirical distribution and the model distribution. This means that we can always rephrase maximum likelihood in terms of cross-entropy.

2.5 PAC Learning

Consider an hypothesis set $\mathscr{H}$ with the realizability property, i.e. $\exists \bar{h} \in H$ s.t. $p_{x,y}(\bar{h}(x) = y) = 1$, since $y \in \{0,1\}$ then $\exists f : X \to Y$ s.t. $p_{x,y}(\bar{h}(x)) = f(x)$ (that is, our hypothesis set contains the true function).

Definition:

A realizable hypothesis set $\mathscr{H}$ is **PAC-learnable** *iff* $\forall \varepsilon, \delta \in (0,1), \forall p(x,y)$, $\exists m_{\varepsilon,\delta} \in \mathbb{N}$ s.t. $\forall m \geq m_{\varepsilon,\delta}, \forall D \sim p^m, |D| = m$, then:

$$p_D(R(h_D^*) \leq \varepsilon) \geq 1 - \delta$$

…

Definition:

Given an hypothesis set $\mathscr{H}$ (not necessarily realizable) and an algorithm A, $\mathscr{H}$ is **agnostic PAC-learnable** iff $\forall \varepsilon, \delta \in (0,1), \forall p(x,y), \exists m_{\varepsilon,\delta} \in \mathbb{N}$ s.t. $\forall D \sim p^m, |D| = m \geq m_{\varepsilon,\delta}$, then:

$$p_D(R(h_D^A) \leq \min_{h \in \mathscr{H}} R(h) + \varepsilon) \geq 1 - \delta$$

being h_D^A the result of applying A to $\mathscr{H}$ and D.

In other words, there exists a number of samples $m_{\varepsilon,\delta}$ such that the probability of the algorithm A to find a hypothesis h whose risk is close to the minimum risk ($\leq \varepsilon$) is at least $1 - \delta$.

We have a bound of generalization error in terms of ε and δ and, in order to achieve this bound, we need to have enaugh data points. Tipically:

- $m_{\varepsilon,\delta}$ depends polinomially on $\frac{1}{\varepsilon}$ and $\frac{1}{\delta}$ (since we want the number of observations to increase moderately with the complexity of the problem)
- A should run in polinomial time.

…

2.6 VC Dimension

Consider a class of hypothesis functions $\mathscr{H} = \{h : X \to \{0,1\}\}$, and a set of points $C = \{c_1, ..., c_N\} \subseteq X$ of input points.

Define $\mathscr{H}_C = \{h(c_1), ..., h(c_N) \mid h \in \mathscr{H}\}$, the set of all tuples of Booleans obrained by applying all possible hypothesis functions $h \in \mathscr{H}$ to all points in C. We say that $\mathscr{H}$ **shatters** C iff $|\mathscr{H}_C| = 2^m$.

Practically, this means that for any label assignment to points in C, we havce a function in our hypothesis set which is able to match such an assignment. Namely, we can exactly describe every possible dataset with inputs in C.

Definition: ***VC Dimension***

The **Vapnik-Chervonenkis (VC) dimension** of $\mathscr{H}$ is the size of the largest set C that can be shattered by $\mathscr{H}$, i.e. the largest set C such that $\forall \{0,1\}^N$ can be realized by $\mathscr{H}$:

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

Tip: ***Just one point***

In calculating the VC dimension, it is enaugh that *we find one set of* m points that can be shattered, it is not necessary to prove that all sets of m points can be shattered.

2.6.1 VC dimension and PAC learning

In what follows, we will explore the reasons why VC dimension is crucial for PAC learnability.

Proposition 1. *If* $\mathscr{H}$ $shatters C, |C| \geq 2m$, *then we cannot learn* $\mathscr{H}$ *with m samples.*

Hence, there will be an assignment of m samples to classes in which we are going to commit a large error.

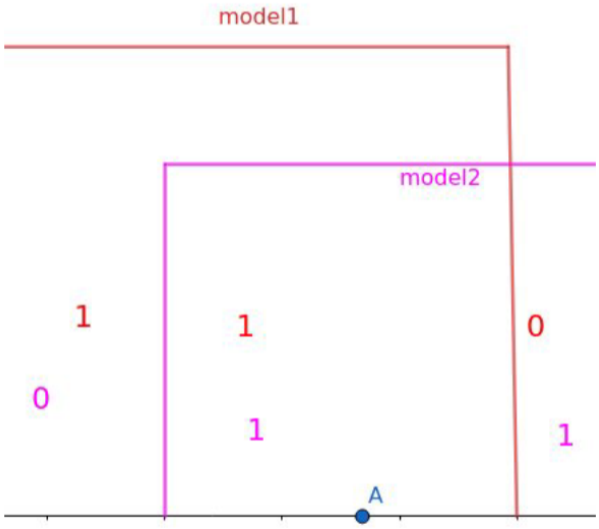


Figure 2.1: Visual interpretation of the theorem: it is impossible to train a model of type $\mathscr{H}_{\dashv+}$, with only a point A with known classification (suppose 1) because the points differ from A could have any classification.

…

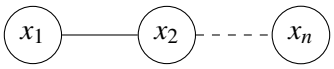
3
Lecture 24/03/2025

3.1 Max Plus

In what follows our task will be, given a joint density □(□), to find the tuple □□ = argmax□ □(□) (i.e. find a setting of the variables that has the largest probability and the value of that probability).

To do so, we will use the max-plus algorithm.

As an example, consider a chain structure described by a Markov Random Field:



whose factorization reads as

$$p(x) = \frac{1}{Z}\psi_{1,2}(x_1,x_2)\dots\psi_{n-1,n}(x_{n-1},x_n)$$

Our goal is to mazimise p(x) w.r.t. x_n.

It is possible to distribute the factorization so that only local computations are required:

$$\max_x p(x) = \max_{x_1}\dots\max_{x_2} p(x) = \frac{1}{Z}\max_{x_1}\max_{x_2}\psi_{1,2}(x_1,x_2)\cdots\max_{x_{n-1}}\Psi_{n-2,n-1}(x_{n-2},x_{n-1})\max_{x_n}\psi_{n-1,n}(x_{n-1},x_n)$$

This happens because of the distributive properties of the max, indeed it holds that, given $a > 0$:

- the max distributes over the product:

$$\max(ab,ac) = a\max(b,c)$$

- the max distributes over the sum:

$$\max(a+b,a+c) = a+\max(b,c)$$

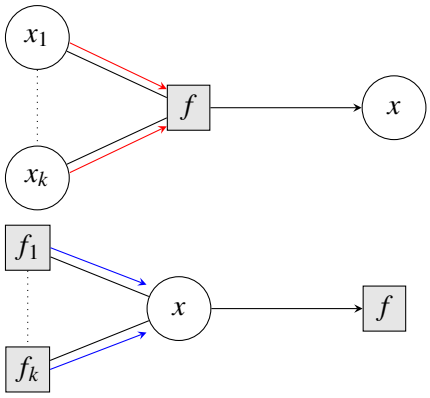
This allows us to take an approach similar to the one used in the sum-product algorithm.

Actually we will maximize $\log p(x)$, hence turning the product into sum of logarithms (and this justifies the name max-plus), i.e. our objective is to maximize

$$\log p(x) = \sum_i \log\psi_i(x_i,x_{i+1}) - \log Z$$

This saves us from product of factors that are possibly very small, since they are probabilities.

Max-plus algorithm is very similar to sum-product: essentially max replaces sum and plus replaces product. This leads to the following scenarios:



3.2 Inference in general Probabilistic Graphical Models

In what follows, we want to extend what we have seen so far to general probabilistic graphical models, meaning Factor Graphs which contain loops.

In these cases, we cannot identify the root and the leaves, hence we don't have well defined forward and backward directions.

There are several possibilities:

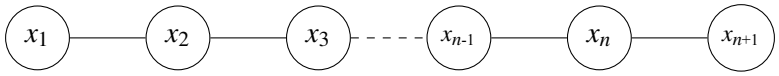
- **Junction Tree Algorithm**: it roughly builds a tree over the cliques of the Factor Graph, then exact inference is done in this tree. The worst case complexity of this algorithm is exponential in the sizeof the largest clique (so possibly very heavy computationally);
- **Loopy Belief Propagation**: forward and backward pass of the sum-product algorithm are iterated several times, until a fix point is reached. Unfortunately there is no guarantee that the algorithm will converge. When it does, it provides an approximate answer to the inference problem;
- **Monte Carlo Sampling**: a general strategy for approximate inference based on Sampling
- **Variational Inference**: it approximates the posterior distribution with a simpler distribution belonging to a pre-specified (parametric) class, which is the closer one to the true posterior, minimizing the KL-divergence.

4 Hidden Markov Models

In what follows our goal is to model ***sequential data*** (i.e. ***time series***). This kind of data are observed from a process evolving in time, typically at different time steps $x_1, x_2, ..., x_N$ (i.e. assuming a discrete model of time). Since sequential data often arise through measurement of time series, there is correlation between observations at different time steps.

These data can come from very different domains: financial data, weather forecast data, speech data, epidemiological data.

Markov Chains are natural models for sequential data, the following is a Markov chain of order 1:



Since all the points (up to a certain step N) are observed, the factorization implied by the model is:

$$p(x_1,\dots,x_N) = p(x_1)p(x_2|x_1)\dots p(x_N|x_{N-1})$$

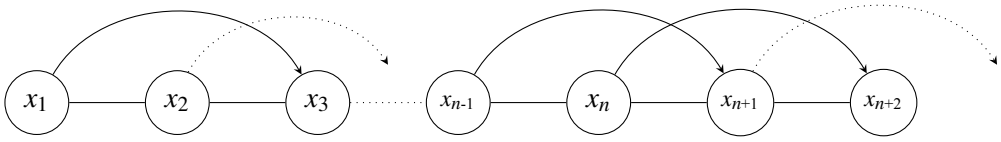
It holds that future observations are independent of all but the most recent observation:

$$x_{n+1} \perp\!\!\!\perp x_{n-1}|x_n$$

A time homogeneous process is a process whose transition probability does not change in time, i.e. such that $p(x_n|x_{n-1}) = p(x_2|x_1)$.

These chains are not always the best model for describing sequential observations, indeed often there is a deeper dependency on the past, and first-order Markov chains suffer from too short memory.

In these cases, we can move to Markov models of order k, where the dependency of x_n is on the previous k steps. The following is a second-order Markov chain:



In this case the factorization of the joint probability distribution is:

$$p(x_1,\dots,x_N) = p(x_1)p(x_2|x_1)p(x_3|x_1,x_2)\dots p(x_N|x_{N-2},x_{N-1})$$

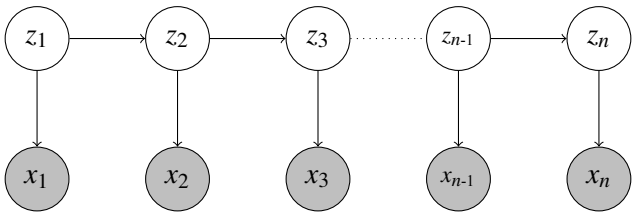
And it holds that:

$$x_{n+2} \perp\!\!\!\perp x_{n-1}|x_n,x_{n+1}$$

Observation:

If x_i are discrete, we talk about ***Markov Chains***. If x_i are continuous and $p(x_n|\dots)$ are Gaussian, we talk about ***autorergressive models*** (of order k).

If we want to build a model for sequential data that is not limited by the Markov assumption of any order, we can rely on state space models, which introduce latent variables. In terms of graphical model, we have that latent variables form a Markov chain, and each of them corresponds to an observation:



5 Sampling-based Inference

...

5.1 Markov chain

5.1.1 Introduction

A Markov Chain is a stochastic process, denoted as

$$\{X_t\}_{t\geq 0}$$

with $t \in \mathbb{N}, X_0, X_1, ..., X_t \in \mathscr{X}$, where $\mathscr{X}$ can be discrete or continuous.

It can be described as a *dynamical system*, i.e. a system in which we start from a certain state x_0 with a given probability $p_0(x)$ and that changes state according to a dynamic described by the *transition kernel* $p(x_n|x_{n-1})$.

Tip: ***Remark - memoryless property***

Markov Chains satisfy the memoryless property, which corresponds to the assumptions encoded into the following probabilistic graphical model:

Therefore we have that:

$$p(x_n|x_{n-1},...,x_0) = p(x_n|x_{n-1})$$

which is known as the ***memoryless*** or ***Markov property***.

We also require the ***time homogeneity*** property that states that:

$$p(x_n|x_{n-1}) = p(x_1|x_0), \quad \forall n \geq 1$$

which means that the probability to jump from a certain state to another state stays the same for every time step.

Definition: ***Transition Kernel***

$p(x_n|x_{n-1})$ satisfying the time homogeneity property is called the (one step) ***transition kernel***.

For a discrete state space we would have a transition matrix, while for a continuous state space, $p(x_n|x_{n-1})$ is a probability density on x_n depending continuously on x_{n-1}.

Since we are interested at the behaviour of the Markov Chain as the index n progresses, i.e. for large times, we need to define few more concepts.

Definition: ***Ergodicity***

A *Markov Chain is* ***erodic*** *iff*:

$$\forall x, y \in \mathscr{X}, \exists t \geq 0 : p(x_t = y|x_0 = x) > 0$$

If a Markov Chain is ergodic, it means that there is always the possibility of going from a given state x to another given state y if we are patient enough. This means that the entire state space is reachable, no matter where we start exploring.

MISSING: MCMC, Gibbs Sampling, PGM

5.1.2 Convergence Diagnostic

How can we check that our Markov Chain has reached the steady state? We will put forward a set of tools that can monitor one or more trajectories and roughly tell us whether we reached it or not. Notice that since we are interested in sampling the stationary probability distributions, we shall start keeping the samples only when we actually reached the steady state.

Let's define some notation for the rest of the section. We are going to denote a general function over a state of the MCMC trajectory $(x_t)_{t\geq 0}$ as $\Psi : \mathscr{X} \to \mathbb{R}$. This function can be a lot of different things, depending on what we are interested in computing, e.g. a projection on single coordinate.

We will assume that Ψ has values in $\mathbb{R}$, and not just in a subset of it, and, if it does, we transform the function Ψ to make it compliant to this assumption (taking the logarithms of quantities in between $(0,\infty)$ for example).

Let's fix some further notation:

- $x_1,\dots,x_n$ is our sampled trajectory
- $\Psi_j := \Psi(x_j)$
- $\bar{\Psi} := \frac{1}{N}\sum_j \Psi_j$ is the estimate of $\mathbb{E}[\Psi] = \int \Psi(x)p(x)\mathrm{d}x$

On a high level, the idea is to look at more than one chain and compare the distribution of the samples that we obtain and see if they look more or less the same.

Pratically,

1. we sample $\frac{m}{2} \geq 1$ trajectories from overdispersed initial points. We try to start from different states that are far away in our state space;
2. Sample for $4n$ steps;
3. We throw away the first half of every trajectory so that we have only $2n$ points left. This phase is known as the ***burn-in*** or ***warm-up*** phase. We do this becouse it takes time to reach the steady state (notice that is an heuristic: convergence to the stationary distribution can happen faster or slower than $2n$ steps)
4. Then we split in 2 parts the remain trajectories, so that we are left with m different trajectories each of length n.

From now on we will denote each sample as x_{ij} where $i \in [1,n]$ and $j \in [1,m]$, where this notation describes the i_{th} smaple of the j_{th} trajectory. We will also denote $\Psi(x_{ij}) = \Psi_{ij}$ and define:

$$\begin{cases} \bar{\Psi}_j := \frac{1}{n}\sum_{i=1}^n \Psi_{ij} \\ \bar{\Psi} := \frac{1}{m}\sum_{j=1}^m \bar{\Psi}_j \end{cases}$$

Hence, $\bar{\Psi}_j$ is the average within the trajectory j and $\bar{\Psi}$ is the average over all the trajectories. We are also interested in the variance of $\bar{\Psi}$, but since our samples are not independent, we don't have that $\mathrm{VAR}[\bar{\Psi}] = \frac{1}{n}\mathrm{VAR}[\Psi]$, i.e. the ariance of the estimator in this case is not just the variance of our random variable divided by the number of samples. If you think about two consecutive points in the chain, there is a high chance that the correlation between them is higher than that of two points which are sampled distantly in time from one another.

Let's define these two quantities:

$$W := \frac{1}{m}\sum_{j=1}^{m} s_j^2, \qquad s_j^2 = \frac{1}{n-1}\sum_{i=1}^{n}(\Psi_{ij} - \bar{\Psi}_j)^2$$

$$B := \frac{n}{m-1}\sum_{j=1}^{m}(\bar{\Psi}_j - \bar{\Psi})^2$$

W is called the ***within variance*** while B is called the ***between variance***.

We know by definition that:

$$W \leq \text{VAR}[\Psi]$$

because when sampling single trajectories we have not necessarily explored and visited the full space. Increasing the number of samples we will converge to the true variance. Moreover, as long as the initial states are overdispersed, it can be shown that:

$$\text{VAR}[\Psi] \leq \text{VAR}^+[\Psi] := \frac{n-1}{m}W + \frac{1}{n}B$$

Then we have both a lower and an upper bound for our variance, both converging to the true variance. Therefore we can compute the statistics:

$$\hat{R} := \sqrt{\frac{\text{VAR}^+[\Psi]}{W}}$$

which can be monitor while running the MCMC simulations. Notice that $\hat{R} > 1$ and that $\hat{R} \xrightarrow{n\to\infty} 1$. Heuristically, we can say that when $\hat{R} \leq 1.1$ we have converged to our stationary distribution.

Effective Smaple Size

…

Hamiltonian Monte Carlo

Hamiltonian Monte Carlo can be considered as the state of the art method for doing Markov Chain Monte Carlo. It falls into the category of ***augmented variables*** Monte Carlo methods.

The idea is to turn the problem into an Hamiltonian, augmenting the state space with momentum variables that provide a sort of ”kinetic energy” that allows the algorithm to move along the surface of the energy corresponding to our probability distribution. This scheme improves a lot the mixing time, hence the efficiency of MCMC algorithms. Moreover, it can be used in the case of high-dimensional multimodal distributions because it does not remain stuck in a single mode.

As usual, we start in a situation in which we want to sample from a distribution $p(x) = \frac{1}{Z}\tilde{p}(x)$, and now we express our distribution as:

$$\frac{1}{Z}\tilde{p}(x) = \frac{1}{Z_x}\exp(H_x(x))$$

This procedure is called the ***Boltzmann Trick*** and it is always possible (just take the logarithm of the distribution) and turns our probability distribution in the fotm of an energy.

Then, we introduce momentum variables y, as many as the number of x variables that we have, and we assign to them the probability distribution $p(y) = \frac{1}{Z_y}\exp(H_y(y))$, where is typical to make the assumption:

$$H_y(y) = \frac{1}{2} y^\top y$$

i.e. to consider $p(y)$ a standard Gaussian.

We are going to sample from the joint distribution, exploiting the independence of our variables:

$$p(x,y) = p(x)p(y) = \frac{1}{Z_x Z_y} \exp(H_x(x) + H_y(y)) = \frac{1}{Z} \exp(H(x,y))$$

The idea of the algorithm is to sample from $p(x,y)$ and then forget about y.

We are going to sample according to the force field defined by the Hamiltonian, i.e. along lines which keep the energy constant. This means that, given some velocity, we follow a trajectory on the probability distribution space accordingly to the equations of motion in order to explore the space without losing energy. In fact, we would like to preserve energy because we want to move between regions with high probability.

Hence we have the following algorithm:

Algorithm 1 Hamiltonian Monte Carlo

1: We start fromn point x_i
2: we sample $y \sim p(y)$, i.e. we randomize the momentum
3: we choose a random direction in time, i.e. we sample from $-1, 1$ uniformly. This makes our problem reversible and provides ergodicity
4: we move according to Hemiltonian dynamics from (x_i, y) to a candidate (x', y'):