UNIVERSIDAD
DE GRANADA

STATISTICAL MODELS WITH VARIATIONAL METHODS

LUIS ANTONIO ORTEGA ANDRÉS

Bachelor's Thesis

Computer Science and Mathematics

Tutor
Serafín Moral Callejón

FACULTY OF SCIENCE
H.T.S. OF COMPUTER ENGINEER AND TELECOMMUNICATIONS

Granada, Thursday 3rd September, 2020

The source code of this text and developed programs are available in the Github repository Ludvins/Statistical-Models-with-Variational-Methods

ABSTRACT

In this document, the theoretical fundamentals of *statistical inference*, more precisely, *variational inference* are reviewed, making special emphasis on how the application of *graphical models* affects inference.

Statistical inference is the process of inferring the underlying properties of a dataset or population. Two main paradigms are discussed, *Bayesian inference* and *likelihoodist inference* which differ in that the former uses *Bayes' theorem* during the inference task, assuming a *prior distribution* over the model parameters.

Variational Bayesian methods are a class of techniques that among with *Bayes' theorem*, transform the inference task in a optimization one. Which might then be approached through machine learning algorithms, such as *gradient or coordinate descent*. Specific algorithms do also rise, as the case of *expectation maximization*.

The combination of *variational inference*, the *exponential family* and *graphical models* do highly simplify the optimization task, automatizing it in some models. *Variational message passing* is an example of this.

Some common models who are usually approached using variational inference are *Gaussian mixtures*, *latent Dirichlet allocation* and *principal components analysis*.

Keywords: *statistical inference, variational inference, exponential family, graphical models, expectation-maximization algorithm, variational Bayes, Gaussian mixture, variational auto-encoders*.

RESUMEN EXTENDIDO EN ESPAÑOL

El campo de la *inferencia variacional* se encuentra en la intersección entre la *inferencia estadística*, la *teoría de grafos* y el *aprendizaje automático*. El trabajo se divide en un estudio **teórico** de la materia (partes 1 a 6), y una aplicación **práctica** de los modelos y técnicas estudiados (parte 7).

La parte teórica describe los conceptos y técnicas utilizadas en la inferencia variacional, apoyándose en un razonamiento matemático de los mismos. En concreto:

Parte 1: En esta parte se describen los conceptos básicos de la teoría de la probabilidad (capítulo 1), entre los cuales se encuentran los conceptos de *variable aleatoria*, *distribución de probabilidad* e *independencia*. También se definen aquellas distribuciones de probabilidad, discretas y continuas, que se vayan a utilizar a lo largo del documento (capítulo 2), resaltando aquellas propiedades que vayan a ser de utilidad.

Se introduce la definición de la *divergencia de Kullback-Leibler* que juega un papel central en este documento. Finalmente, el tercer capítulo introduce la teoría de grafos, dirigidos y no dirigidos, junto con aquellas propiedades necesarias en *modelos gráficos*.

Parte 2: En esta parte se discute el concepto de *inferencia estadística*, destacando las principales diferencias entre dos enfoques distintos: el de *estimación de máxima verosimilitud* (capítulo 5), que busca encontrar aquellos valores de los parámetros que maximizan la probabilidad de los datos se generen a partir de estos, y la *inferencia Bayesiana* (capítulo 6) que se basa en la utilización del teorema de Bayes y una distribución de probabilidad a "priori" sobre dichos parámetros.

En el primer enfoque llegamos a la conclusión de que la distribución de máxima verosimilitud coincide con la distribución empírica cuando no consideramos restricciones. Además, vemos que la estimación de máxima verosimilitud es un caso concreto de una técnica de inferencia Bayesiana, *máximo a posteriori*, la cual busca aquellos valores de los parámetros que mejor se ajustan a los datos.

Como último, en el capítulo 7 se describen las tres formas de modelar *datos perdidos*, a saber, *completamente aleatorio*, *aleatorio* y *no aleatorio*, expresando matemáticamente como afecta cada uno de ellos a la inferencia.

Parte 3: Esta parte está enfocada al estudio de la *inferencia variacional Bayesiana*, aplicada a modelos probabilísticos con variables ocultas en las cuales se incluyen los parámetros. La inferencia variacional transforma el problema de inferencia en un problema de optimización, afrontado mediante técnicas de *machine learning*. Se introduce brevemente el concepto de métodos de *cadenas de Markov Monte Carlo* como principal contrapartida a los métodos variacionales.

Los métodos variacionales se apoyan en la utilización de distribución de probabilidad variacional sobre las variables ocultas $Q(\boldsymbol{z})$ y minimizar su divergencia de Kullback-Leibler

con la distribución condicionada sobre los datos $P(\boldsymbol{z} \mid \boldsymbol{x})$. Para ello se deriva una cota inferior al logaritmo de la evidencia $\log P(\boldsymbol{x})$, cambiando el problema de inferencia por uno de optimización.

Se estudian dos algoritmos principales: el algoritmo *EM*, estudiado en el capítulo 9, resulta un método parcialmente Bayesiano, pues no considera distribuciones de probabilidad en los parámetros pese a ser un método variacional. Por otro lado, el algoritmo *CAVI* (capítulo 10), surge como una generalización totalmente Bayesiana del anterior donde sí se consideran distribuciones a "priori" en los parámetros y el problema se afronta realizando *descenso coordinado*. En el capítulo 11, se introduce la *familia exponencial* y una clase de modelos que simplifican la inferencia, los *modelos condicionalmente conjugados*. Para terminar, se explica el modelo de la *mixtura de Gaussianas* y su resolución mediante el algoritmo CAVI (capítulo 12).

El modelo de *mixtura de Gaussianas* supone que los datos se generan a partir de la suma ponderada de distintas distribuciones Gaussianas, donde se consideran distribuciones de probabilidad sobre las variables correspondientes a los pesos, las medias y las varianzas de cada una de ellas.

Parte 4: En esta parte se introducen las ideas fundamentales de los *modelos gráficos*. Se describen dos modelos principales: las *redes Bayesianas* como principal modelo gráfico dirigido, donde se incluyen los conceptos de *D-separación* y *D-conexión*, útiles para la construcción de redes a partir de un conjunto de datos, y los *campos aleatorios de Markov* como principal modelo gráfico no dirigido. El algoritmo de *eliminación de variables* permite calcular de forma eficiente una distribución de probabilidad condicionada (problema $\mathcal{NP}-$difícil), ayudándose de la estructura del modelo gráfico y *programación dinámica*.

Aunque el resto del documento está enfocado a la utilización de redes Bayesianas, se pueden aplicar a los campos aleatorios de Markov métodos de inferencia similares.

Parte 5: En esta parte se estudia el aprendizaje de modelos de redes Bayesianas. Se introduce el *algoritmo PC* (capítulo 16) como método de construcción de redes Bayesianas a partir de un conjunto de datos. Para ello, es necesario calcular la independencia condicionada de un conjunto de variables, donde surge el concepto de *información mutua*.

Como ya dijimos antes, la distribución de probabilidad sin restricciones que maximiza la función de verosimilitud es la distribución empírica, sin embargo, las distribuciones ligadas a una red Bayesiana presentan una clara restricción (su factorización). En el capítulo 17 se comprueba como esta restricción no es suficiente para cambiar la distribución de máxima verosimilitud, donde cada factor del modelo gráfico $P(x \mid pa(x), \theta^{ML})$ corresponde con la distribución empírica $Q(x \mid pa(x))$.

Se exponen las ventajas que provee las estructura de red Bayesiana a la tarea de inferencia, haciendo especial énfasis en cómo se ve alterado el algoritmo *EM* (capítulo 19), donde uno de sus pasos se ve completamente alterado y simplificado gracias a la red.

En el capítulo 20, se introduce el algoritmo de *paso de mensajes variacional*, como aplicación del algoritmo *CAVI* a un modelo probabilístico ligado a una red Bayesiana, donde cada factor se encuentra en la familia exponencial. De esta forma, un procedimiento de paso de mensajes entre los nodos permite simplificar y automatizar la inferencia.

Parte 6: En esta parte se revisa el modelo gráfico correspondiente a la *mixtura de Gaussianas* (capítulo 21) y se estudian 2 modelos gráficos conocidos, *asignación latente de Dirichlet* (capítulo 22) y *análisis de componentes principales probabilístico* (capítulo 23).

En el primer modelo, que estudia la ocurrencia de palabras y temas en un conjunto de documentos, se establece el modelo generativo de los datos y el modelo variacional, estudiando las actualizaciones de parámetros que el algoritmo *CAVI* lleva a cabo en cada iteración.

El segundo modelo, basado en una reducción de dimensionalidad, extiende el *análisis de componentes principales* clásico, donde se estudia una transformación lineal entre el espacio observado y el espacio donde se supone la reducción, a un ámbito probabilístico. Normalmente, el problema clásico se resuelve calculando la descomposición de valores propios de la matriz de covarianza de los datos. Por otro lado, el modelo probabilístico se resuelve mediante el algoritmo EM.

El modelo probabilístico se extiende al caso no lineal mediante el uso de redes artificiales neuronales en lugar de aplicaciones lineales entre los espacios.

Finalmente se puede considerar que el modelo variacional presenta otra red neuronal, que transforma los datos del espacio observado al oculto, dando lugar al modelo conocido como *codificadores automáticos variacionales*, formado por dos redes neuronales, un *codificador* y un *decodificador*.

Por otro lado, la parte práctica estudia la aplicación de las técnicas estudiadas a algunos modelos vistos, concretamente aquellos relacionados con la reducción de dimensionalidad y la mixtura de Gaussianas. Esto se desarrolla en la parte 7:

Parte 7: El objetivo de esta parte es probar las limitaciones y funcionalidades de tres *frameworks* de `Python`, a saber `InferPy`, `BayesPy` y `Scikit-Learn` al usarlos en conjuntos de datos reales. El primero de estos se caracteriza por permitir la integración de modelos con redes neuronales artificiales, lo cual permite modelar análisis de componentes principales no lineal y codificadores automáticos variacionales de forma sencilla. Por otro lado, `BayesPy` se caracteriza por la utilización de *paso de mensajes variacional* como método de inferencia. Por último, `Scikit-Learn` es una librería de *machine learning* de propósito general, donde la clase *BayesianGaussianMixture* permite aprender modelos de mixtura de Gaussianas y utilizar el algoritmo EM para realizar inferencia.

Se han probado modelos de reducción de dimensionalidad con `InferPy` en dos bases de datos diferentes (*Mnist* y *Breast Cancer Wisconsin*). Esta última se ha intentado modelar utilizando `BayesPy` y `Scikit-Learn` mediante un modelo de mixtura de Gaussianas.

Palabras clave: *inferencia estadística*, *inferencia variacional*, *familia exponencial*, *modelos gráficos*, *algoritmo EM*, *Bayes variacional*, *mixtura de Gaussianas*, *codificadores automáticos variacionales*.

INTRODUCTION

Variational inference concepts, which are adapted from statistical physics, first appeared in Anderson & Peterson (1987), in which the authors used them to fit a neural-network. More precisely, they used *mean-field* methods to achieve it.

In the coming years, several studies were done on variational inference, such as Hinton & Van Camp (1993), who used further mean-field methods in neural networks, and Jordan *et al.* (1999) that generalized variational inference to many models.

Today, variational inference is more scalable and easy to derive, in some cases it is even automated. It has been applied to many different models and types of learning, such as document topic learning (Blei (2012)).

This document attempts to give an overview of some results in *Bayesian variational inference* as well as to test some frameworks for probabilistic modeling. In these tests an effort to apply variational inference techniques to real databases is made.

The theoretical part of this document, which is encompassed by chapters 1 to 23, describes the basic concepts of *statistical inference*, from classical to *variational*. After this, the *exponential family* and *graphical models* are reviewed together with their influence in variational inference, focusing on how the inference task is simplified by their usage.

The last chapters, focus on the utilization of different frameworks to experiment different models, which involve *Gaussian mixture* and dimensionality reduction via *principal components analysis* and *variational auto-encoders*.

The main sources used for writing this documents were Barber (2007), Bishop (2006), Koller & Friedman (2009), Masegosa *et al.* (2019) and Blei *et al.* (2017). Citations are made in each section within the text.

MAIN GOALS ANS RESULTS ACHIEVED

The main goals of this bachelor's thesis were:

1. to study statistical graphical models and the use of variational methods to solve inference and estimation problems,
2. describe how these can be used to build statistical models and resolve computational learning problems and
3. install variational inference frameworks in order to test their features and limitations, and apply them to real practical problems.

The first two were completely successful. For the latter, different generative models are reviewed, such as, Gaussian mixture, latent Dirichlet allocation and principal component analysis. Explicit computational resolutions are given for the first two, whereas generalizations of the third are build using neural networks.

The third goal was also successful. Three different frameworks were selected, `InferPy`, `BayesPy` and `Scikit-Learn`, based on their capabilities for modeling and making inference. Even though some experiments could not be executed due to limitations of the frameworks themselves, I got to familiarize myself with these limitations. In this way, I gained a deeper understanding of what can and can not be done with these state-of-the-art frameworks, so that a plausible future path in this regard would be to try and implement the capabilities that are missing from this software.

CONTENTS

Part I

BASIC CONCEPTS

In this part the underlying concepts of probability and graph theory that will be needed are reviewed. *Probability distributions* and *random variables* are defined, and *Bayes' theorem* is stated. The *Kullback-Leibler divergence* is defined as it plays a central role in all the following theory.

Any *discrete and continuous distributions* used later on in the document is also reviewed.

1

PROBABILITY

All our theory will be made under the assumption that there is a *referential set* Ω, set of all possible outcomes of an experiment. Any subset of Ω will be called an *event*.

Definition 1. Let $\mathcal{P}(\Omega)$ be the power set of Ω. Then, $\mathcal{F} \subset \mathcal{P}(\Omega)$ is called a *σ-algebra* if it satisfies three conditions:

- It contains the full referential set, $\Omega \in \mathcal{F}$.
- $\mathcal{F}$ is closed under complementation.
- $\mathcal{F}$ is closed under countable unions.

From these properties it follows that $\emptyset \in \mathcal{F}$ and that $\mathcal{F}$ is closed under countable intersections.

The tuple $(\Omega, \mathcal{F})$ is called a *measurable space*.

Definition 2. A *probability* P over a measurable space $(\Omega, \mathcal{F})$ is a mapping $P : \mathcal{F} \to [0, 1]$ which satisfies:

- $P(\alpha) \geq 0 \ \ \forall \alpha \in \mathcal{F}$.
- $P(\Omega) = 1$.
- P is countably additive, that is, if $\{\alpha_n\}_{n \in \mathbb{N}} \subset \mathcal{F}$ is a countable collection of pairwise disjoint sets, then

$$P\Big(\bigcup_{n \in \mathbb{N}} \alpha_n \Big) = \sum_{n \in \mathbb{N}} P(\alpha_n).$$

The first condition guarantees non negativity. The second one states that the *trivial event* has the maximal possible probability of 1. The third condition implies that given a set of pairwise disjoint events, the probability of either one of them occurring is equal to the sum of the probabilities of each one.

These conditions imply the following two:

- $P(\emptyset) = 0$.
- $P(\alpha \cup \beta) = P(\alpha) + P(\beta) - P(\alpha \cap \beta)$.

The triple $(\Omega, \mathcal{F}, P)$ is called a *probability space*.

Definition 3. Given two events $\alpha, \beta \in \mathcal{F}$, with $P(\beta) \neq 0$, the conditional probability of α given β is defined as the quotient of the probability of the joint events and the probability of β:

$$P(\alpha \mid \beta) = \frac{P(\alpha \cap \beta)}{P(\beta)}.$$

Theorem 1. ***(Bayes' theorem)***. *Let α, β be two events of an experiment, given that $P(\beta) \neq 0$. Then*

$$P(\alpha \mid \beta) = \frac{P(\beta \mid \alpha)P(\alpha)}{P(\beta)}.$$

Example 1. Consider a study where the relation of a disease d and an habit h is being investigated. Suppose that $P(d) = 10^{-5}$, $P(h) = 0.5$ and $P(h \mid d) = 0.9$. What is the probability that a person with habit h will have the disease d?

$$P(d \mid h) = \frac{P(d \cap h)}{P(h)} = \frac{P(h \mid d)P(d)}{P(h)} = \frac{0.9 \times 10^{-5}}{0.5} = 1.8 \times 10^{-5}.$$

If the probability of having habit h is set to a much lower value as $P(h) = 0.001$, then the above calculation gives approximately $1/100$. Intuitively, a smaller number of people have the habit and most of them have the desease. This means that the relation between having the desease and the habit is stronger now compared with the case where more people had the habit.

Definition 4. Two events $\alpha, \beta \in \mathcal{F}$ are said *independent* if knowing one of them does not give any extra information about the other. Mathematically,

$$P(\alpha \cap \beta) = P(\alpha)P(\beta) \quad \text{and} \quad P(\alpha \mid \beta) = P(\alpha).$$

Let $\alpha, \beta, \gamma \in \mathcal{F}$, α and β are said to be *conditionally independent* on γ, expressed as $\alpha \perp\!\!\!\perp \beta \mid \gamma$, if and only if

$$P(\alpha \cup \beta \mid \gamma) = P(\alpha \mid \gamma)P(\beta \mid \gamma).$$

Otherwise, they are said to be *conditionally dependent* on γ, expressed as $\alpha \not\perp\!\!\!\perp \beta \mid \gamma$.

RANDOM VARIABLES

Definition 5. A function $f : \Omega_1 \to \Omega_2$ between two measurable spaces $(\Omega_1, \mathcal{F}_1)$ and $(\Omega_2, \mathcal{F}_2)$ is said to be *measurable* if $f^{-1}(\alpha) \in \mathcal{F}_1$ for every $\alpha \in \mathcal{F}_2$.

Definition 6. A *random variable* is a measurable function $X : \Omega \to E$ from a probability space $(\Omega, \mathcal{F}, P)$ to a measurable space $(E, \mathcal{F}')$ verifying $X(\omega) \in \mathcal{F}' \ \forall \omega \in \Omega$.

The probability of X taking a value on a measurable set $S \in E$ is written as

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

Where the sub-index is usually omitted. A *probability distribution* P_X of a random variable X over the probability space $(\Omega, \mathcal{F}, P)$ is defined as the push-forward measure of it, that is, $P_X = P \circ X^{-1}$.

Questions like "How likely is that the value of X equals a?" are equivalent to ask for the probability (measure) of the set $\{\omega \in \Omega \mid X(w) = a\}$.

The following notation is going to be used: random variables will be denoted with an upper case letter like X and a set of variables with a bold symbol like $\boldsymbol{X}$. The meaning of $P(state)$ will be clear without a reference to the variable. Otherwise $P(X = state)$ will be used. Using a lower case letter like $P(x)$ will denote the probability of the corresponding upper case variable X taking a specific value x.

Definition 7. The *cumulative distribution function* of a real-valued random variable X is defined as

$$F_X(x) = P(X \leq x),$$

where the right-hand side represents the probability of the random variable taking value below or equal to x.

Definition 8. When the image of a random variable X is countable, the random variable is called a *discrete random variable* and its *probability mass function* p gives the probability of it being equal to some value:

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

In case the image is uncountable and real, then X is called a *continuous random variable* and if there exists is a non-negative Lebesgue-integrable f such that

$$F_X(x) = P(X \leq x) = \int_{-\infty}^{x} f(u)du,$$

then it is called its *probability density function*.

A *mixed random variable* is a random variable who is neither discrete nor continuous, it can be realized as the sum of a discrete and continuous random variables. An example of a random variable of mixed type would be based on an experiment where a coin is flipped and a random positive number is chose only if the result of the coin toss is heads, -1 otherwise.

From now on, $P(x)$ will denote $f(x)$ when X is a continuous random variable.

Integrate notation will be used in both continuous and discrete cases, where the last one can be interpreted as integration with respect to the *counting measure* defined as

$$\#(dx) = \sum_{n \in \mathcal{I}} \delta(x - n)dx,$$

where $\mathcal{I}$ is the set of values X can take, and δ is the Dirac measure. Given this measure, integration corresponds to summation as

$$\int_x P(x)\#(dx) = \sum_{n \in \mathcal{I}} \int_x P(x)\delta(x - n)dx = \sum_{n \in \mathcal{I}} P(n).$$

Where $\int_x f(x)\delta(x - x_0) = f(x_0)$ is used. Given this, from now on, integration notation will be used for both discrete and continuous variables given that the integrals will be respect to the counting measure when needed.

Definition 9. The *conditional probability* might be defined over random variables, let X, Y be two random variables, then

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

It is required that $P(y) \neq 0$ for the conditional probability to be defined.

The *Bayes' theorem* may be enunciated as

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

Clearly, an arbitrary number of variables can be considered in both cases.

Definition 10. The *marginal distribution* of a subset of random variables is the probability distribution of the variables contained in that subset.

Let X, Y be two random variables, the marginal distribution of X is:

$$P(x) = \int_y P(x,y).$$

Definition 11. Let $\boldsymbol{X} = (X_1, X_2, \ldots, X_n)$ be a set of random variables, the *joint probability distribution* for $\boldsymbol{X}$ is function that gives the probability of each random variable X_n falling in a particular range or discrete set of values for that variable. It is called a *multi-variate distribution*.

When using only two random variables, then is called a *bi-variate distribution*.

This distribution can be expressed either in terms of a joint cumulative distribution function

$$F_{\boldsymbol{X}}(\boldsymbol{x}) = F_{X_1,\ldots,X_n}(x_1, \ldots, x_n) = P(X_1 \leq x_1, \ldots, X_n \leq x_n)^1,$$

or using a probability density or mass function.

Definition 12. Two random variables X and Y are said to be *independent* if knowing one of them doesn't give any extra information about the other. Mathematically,

$$P(x,y) = P(x)P(y).$$

From this it follows that if X and Y are independent, then $P(x \mid y) = P(x)$.

Definition 13. Let X, Y and Z be three random variables, then X and Y are *conditionally independent* given Z if and only if

$$P(x,y \mid z) = P(x \mid z)P(y \mid z),$$

in that case we will denote $X \perp\!\!\!\perp Y \mid Z$. If X and Y are not conditionally independent, they are *conditionally dependent* $X \not\perp\!\!\!\perp Y \mid Z$

Both independence definitions can be made over sets of variables $\boldsymbol{X}, \boldsymbol{Y}$ and $\boldsymbol{Z}$ in a straightforward way.

1 Where $\boldsymbol{x} = (x_1, \ldots, x_n)$

Definition 14. A set of N random variables $\{X_1, \ldots, X_N\}$ defined to assume values in $I \subset \mathbb{R}$ are said *independent and identically distributed (i.i.d)* if and only if they are independent, i.e,

$$F_{X_1,\ldots,X_N}(x_1, \ldots, x_n) = F_{X_1}(x_1) \ldots F_{X_N}(x_N) \ \forall x_1, \ldots, x_N \in I,$$

and are identically distributed

$$F_{X_1}(x) = F_{X_n}(x) \ \forall n \in \{2, \ldots, N\} \text{ and } \forall x \in I.$$

Definition 15. A *multi-variate random variable* or *random vector* is a column vector $\boldsymbol{X} = (X_1, \ldots, X_N)^T$ whose components are random variables that can be defined over different probability spaces.

Note that the same symbol $\boldsymbol{X}$ is used for random vectors and sets of variables, but the meaning will be clear within the context.

2

DISTRIBUTIONS

In this section some concepts concerning probability distributions among with some of the most used ones are summarized.

From now on, let X be a random variable and P its probability distribution.

Definition 16. The *mode* X_* of the probability distribution P is the state of X where the distribution takes it's highest value

$$X_* = \arg\max_x P(x).$$

A distribution could have more than one mode, in this case we say it is *multi-modal*.

Definition 17. The notation $\mathbb{E}\left[X\right]$ is used to denote the *average* or *expectation* of the values a real-valued variable takes respect to its distribution. It is worth mentioning that it might not exists. If X is non-negative, it is defined as

$$\mathbb{E}\left[X\right] = \int_\Omega X(\omega) dP(\omega)^1.$$

For a general variable X, it is defined as $\mathbb{E}\left[X\right] = \mathbb{E}\left[X^+\right] - \mathbb{E}\left[X^-\right]$. Where

$$X^+(\omega) = \max(X(\omega), 0) \quad \text{and} \quad X^-(\omega) = \min(X(\omega), 0).$$

In case the variable is continuous, the expectation is

$$\mathbb{E}\left[X\right] = \int_{-\infty}^{+\infty} x f(x) dx.$$

In case it is discrete, let x_i be the values X can take, the expectation takes the form

$$\mathbb{E}\left[X\right] = \sum_i x_i p(x_i) dx.$$

Let $g : \mathbb{R} \to \mathbb{R}$ be a measurable function, then $g \circ X$ is another random variable and we can talk about $\mathbb{E}\left[g(X)\right]$, so in case X is continuous, we have that

$$\mathbb{E}\left[g(X)\right] = \int_{-\infty}^{+\infty} g(x) f(x) dx.$$

1 P is a measure over Ω

In the above definition, the expectation is calculated over the probability distribution P of the random variable, in future sections this distribution is unknown and a guessed distribution Q will be used. In this cases when the distribution is not clear from the context the notation $\mathbb{E}_Q\Big[X\Big]$ will be used.

Definition 18. We define the k^{th} moment of a distribution as the average of X^k over the distribution

$$\mu_k = \mathbb{E}\Big[X^k\Big]$$

For $k = 1$ it is typically denoted as μ. Note μ_k can also denote the k^{th} element in the mean vector of a multi-variate variable.

Definition 19. The *variance* of a distribution is defined as

$$Var(X) = \sigma^2 = \mathbb{E}\Big[(X - \mathbb{E}\Big[X\Big])^2\Big] = \mathbb{E}\Big[X\Big]^2 - \mathbb{E}\Big[X^2\Big]$$

The positive square root of the variance σ is called the *standard deviation*.

When using a multi-variate distribution $\boldsymbol{X} = (X_1, \ldots, X_N)^T$ we can talk about the *covariance matrix* $\boldsymbol{\Sigma}$ whose elements are

$$\begin{aligned}\boldsymbol{\Sigma}_{ij} &= \mathbb{E}\Big[\Big(X_i - \mathbb{E}\Big[X_i\Big]\Big)\Big(X_j - \mathbb{E}\Big[X_j\Big]\Big)\Big] = \mathbb{E}\Big[(X_i - \mu_i)(X_j - \mu_j)\Big] \\ &= \mathbb{E}\Big[X_i X_j\Big] - \mathbb{E}\Big[X_i\Big]\mathbb{E}\Big[X_j\Big]\end{aligned}$$

The following result will be helpful later on.

Proposition 1. *Let $\boldsymbol{X} = \{X_1, \ldots, X_N\}$ be a set of random variables, $\boldsymbol{X}_0 \subset \boldsymbol{X}$ and $P(\boldsymbol{X}), P(\boldsymbol{X}_0)$ their probability distributions. It follows that the expectation of a function g over $\boldsymbol{X}_0$, verifies*

$$\mathbb{E}_{P(\boldsymbol{X})}\Big[g(\boldsymbol{X}_0)\Big] = \mathbb{E}_{P(\boldsymbol{X}_0)}\Big[g(\boldsymbol{X}_0)\Big].$$

That is, we only need to know the marginal distribution of the subset in order to carry out the average.

Proof. Let $\mathcal{I} = (i_1, \ldots, i_k)$ be the indexes corresponding to $\boldsymbol{X}_0$, then

$$\begin{aligned}\mathbb{E}_{P(\boldsymbol{X})}\Big[g(\boldsymbol{X}_0)\Big] &= \int_{x_1} \cdots \int_{x_N} g(x_{i_1}, \ldots, x_{i_k}) f(x_1, \ldots, x_N) \\ &= \int_{x_{i_1}} \cdots \int_{x_{i_k}} g(x_{i_1}, \ldots, x_{i_k}) \int \cdots \int f(x_1, \ldots, x_N) d_{x_1}, \ldots, d_{x_N} \\ &= \int_{x_{i_1}} \cdots \int_{x_{i_k}} g(x_{i_1}, \ldots, x_{i_k}) f(x_{i_1}, \ldots, x_{i_k}) = \mathbb{E}_{P(\boldsymbol{X}_0)}\Big[g(\boldsymbol{X}_0)\Big].\end{aligned}$$

Where in the last equality we used marginalization. □

We are going to discuss now some examples of probability distributions that are going to be used from now on.

2.1 DISCRETE DISTRIBUTIONS

Bernoulli distribution

The Bernoulli distribution describes a discrete binary variable X that takes the value 1 with probability p and the value 0 with probability $1 - p$.

$$P(x) = \begin{cases} p & \text{if } x = 1 \\ 1 - p & \text{if } x = 0 \end{cases}.$$

Categorical distribution

A generalization of the Bernoulli distribution when the variable can take more than two states is the *Categorical distribution*. Let $Dom(X) = \{1, \ldots, N\}$, then X follows a categorical distribution of parameters $\theta = (\theta_1, \ldots, \theta_N)$ if and only if

$$P(x \mid \theta) = \prod_{i=1}^{N} \theta_i^{\mathbb{I}[x=i]} \text{ and } \sum_{i=1}^{N} \theta_i = 1.$$

Binomial distribution

The binomial distribution describes the number of successes in a sequence of independent Bernoulli trials. A discrete binary random variable X follows a *binomial distribution* of parameters $n \in \mathbb{N}$ and $p \in [0, 1]$, denoted as $X \sim B(n, p)$ if and only if

$$P(x = k) = \binom{n}{k} p^k (1 - p)^{n-k}.$$

That is, X models the probability of getting k times an 1 outcome in a Bernoulli trial with parameter p in n total trials.

Multinomial distribution

The multinomial distribution is a generalization of the binomial distribution which describes the result of a sequence of independent trials in a categorical distribution. A discrete random variable X follows a *multinomial distribution* of parameters $n \in \mathbb{N}$, $\boldsymbol{p} = (p_1, \ldots, p_K)$ such that $\sum p_k = 1$ if and only if

$$P(\boldsymbol{x} = (x_1, \ldots, x_K)) = \begin{cases} \frac{n!}{x_1! \ldots x_K!} \prod_{k=1}^{K} p_k^{x_k} & \text{if } \sum_k x_k = n \\ 0 & \text{otherwise} \end{cases}$$

2.2 CONTINUOUS DISTRIBUTIONS

Uni-Variate Gaussian distribution

The *normal* or *Gaussian distribution* is a type of continuous probability distribution for real-valued random variables.

Definition 20. We say the real valued random variable X follows a *normal distribution* of parameters $\mu, \sigma \in \mathbb{R}$, denoted as $X \sim \mathcal{N}(\mu, \sigma)$ if and only if, its probability density function exists and is

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

The parameter μ is the mean or expectation of the distribution and σ is its standard deviation.

The simplest case of a normal distribution is known as *standard normal distribution*. It is a special case where $\mu = 0$ and $\sigma = 1$, then its density function is

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

One of the properties of the normal distribution is that if $X \sim \mathcal{N}(\mu, \sigma)$, $a, b \in \mathbb{R}$ and $f : \mathbb{R} \to \mathbb{R}$ be defined as $f(x) = ax + b$, then $f(X) \sim \mathcal{N}(\mu + b, a^2\sigma)$.

Multi-Variate Gaussian distribution

This distribution plays a fundamental role in this project so we will discuss its properties in more detail. It is an extension of the uni-variate one when having a multi-variate random variable.

Definition 21. We say that a random vector $\boldsymbol{X} = (X_1, \ldots, X_N)$ follows a *multi-variate normal or Gaussian distribution* of parameters $\boldsymbol{\mu} \in \mathbb{R}^N$ and $\boldsymbol{\Sigma} \in \mathbb{M}_N(\mathbb{R})$, denoted as $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ if and only if its probability density function is

$$f(\boldsymbol{x}) = \frac{1}{\sqrt{det(2\pi\boldsymbol{\Sigma})}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}.$$

Where $\boldsymbol{\mu}$ is the mean vector of the distribution, and $\boldsymbol{\Sigma}$ the *covariance matrix*. The inverse matrix $\boldsymbol{\Sigma}^{-1}$ is called *precision matrix*. It also satisfies that

$$\boldsymbol{\mu} = \mathbb{E}[\boldsymbol{X}], \quad \boldsymbol{\Sigma} = \mathbb{E}\Big[(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^T\Big].$$

As $\boldsymbol{\Sigma}$ is a real symmetric matrix, it can be eigen-decomposed as

$$\boldsymbol{\Sigma} = \boldsymbol{E}\boldsymbol{\Delta}\boldsymbol{E}^T,$$

where $\boldsymbol{E}^T\boldsymbol{E} = \boldsymbol{I}$ and $\boldsymbol{\Delta} = diag(\lambda_1, \ldots, \lambda_n)$.

Using the transformation

$$\boldsymbol{y} = \boldsymbol{\Delta}^{\frac{1}{2}}\boldsymbol{E}^T(\boldsymbol{x} - \boldsymbol{\mu}),$$

we get that

$$(\boldsymbol{x} - \boldsymbol{\mu})^T\boldsymbol{\Sigma}(\boldsymbol{x} - \boldsymbol{\mu}) = \boldsymbol{y}^T\boldsymbol{y}.$$

Using this, the multi-variate Gaussian distribution reduces to a product of n uni-variate standard Gaussian distributions.

Gamma distribution

Definition 22. We say that a continuous random variable X defined on $\mathbb{R}^+$ follows a *gamma distribution* of parameters $\alpha, \beta > 0$, denoted as $X \sim Gamma(\alpha, \beta)$ if and only if its density function is

$$f(x) = \frac{\beta^\alpha x^{\alpha-1} e^{-\beta x}}{\Gamma(\alpha)},$$

where Γ is the Gamma function defined as

$$\Gamma(\alpha) = \int_0^\infty x^{\alpha-1} e^{-x} dx.$$

The mean is given by $\mathbb{E}\Big[X\Big] = \frac{\alpha}{\beta}$.

Definition 23. The *inverse gamma distribution* is defined by the density function

$$f(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{-\alpha-1} e^{-\beta/x}.$$

Wishart distribution

The Wishart distribution is a generalization of the gamma distribution to multiple dimensions. Consider G a $p \times \nu$ matrix where each column $G_i \sim \mathcal{N}_p(0, \boldsymbol{V})$ is independent from the others, with V a common vector.

Then $S = GG^T$ follows a Wishart distribution with ν degrees of freedom,

$$S \sim \mathcal{W}(\nu, \boldsymbol{V}).$$

The Wishart distribution is characterized by its probability density function

$$f(x) = \frac{1}{2^{\nu p/2} |\boldsymbol{V}|^{\nu/2} \Gamma_p(\frac{\nu}{2})} |x|^{(\nu-p-1)/2} e^{-(1/2)tr(\boldsymbol{V}^{-1}x)},$$

where

$$\Gamma_p(\frac{\nu}{2}) = \pi^{p(p-1)/4} \prod_{j=1}^{p} \Gamma\Big(\frac{n}{2} - \frac{j-1}{2}\Big).$$

Gaussian-Wishart distribution

The Gaussian-Wishart distribution is a four parameter distribution. Let $\boldsymbol{\mu}$ follow a Gaussian distribution of mean $\boldsymbol{\mu}_0$ and covariance matrix $(\boldsymbol{\lambda\Lambda})^{-1}$

$$\boldsymbol{\mu} \sim \mathcal{N}(\boldsymbol{\mu}_0, (\boldsymbol{\lambda\Lambda})^{-1}),$$

where $\boldsymbol{\Lambda}$ follows a Wishart distribution with parameters $\nu, \boldsymbol{W}$:

$$\boldsymbol{\Lambda} \sim \mathcal{W}(\nu, \boldsymbol{W}).$$

Then their joint distribution is a Gaussian-Wishart distribution:

$$(\boldsymbol{\mu}, \boldsymbol{\Lambda}) \sim \mathcal{NW}(\boldsymbol{\mu}_0, \boldsymbol{\lambda}, \nu, \boldsymbol{W}).$$

Their probability density function equals a product of the corresponding Gaussian and Wishart distribution:

$$f(\boldsymbol{\mu}, \boldsymbol{\Lambda}) = \mathcal{N}(\boldsymbol{\mu}_0, (\boldsymbol{\lambda}\boldsymbol{\Lambda})^{-1})\mathcal{W}(\nu, \boldsymbol{W})^{2}.$$

Beta distribution

Definition 24. We say that a continuous random variable X defined on the interval $[0, 1]$ follows a *Beta distribution* of parameters $\alpha, \beta > 0$, denoted as $X \sim Beta(\alpha, \beta)$ if and only if its density function is

$$f(x) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1}(1-x)^{\beta-1},$$

where $B(\alpha, \beta)$ is the *beta function* defined as

$$B(\alpha, \beta) = \int_0^1 x^{\alpha-1}(1-x)^{\beta-1} dx.$$

Its mean is given by $\mathbb{E}\left[X\right] = \frac{\alpha}{\alpha+\beta}$.

Dirichlet distribution

The Dirichlet distribution is a family of continuous multi-variate probability distributions parameterized by a vector $\boldsymbol{\alpha}$ of positive reals. It is a multi-variate generalization of the Beta distribution.

Definition 25. We say that a continuous random multi-variate variable $\boldsymbol{X}$ with order $K \geq 2$, follows a *Dirichlet distribution* with parameters $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_K)$, if and only if its density function is defined as

$$f(\boldsymbol{x}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{k=1}^{K} x_k^{\alpha_k - 1},$$

and it satisfies that

$$\sum_{k=1}^{K} x_k = 1 \text{ and } x_k > 0 \; \forall k = 1, \ldots, K.$$

Where the normalization constant is the multi-variate beta function

$$B(\boldsymbol{\alpha}) = \frac{\prod_k \Gamma(\alpha_k)}{\Gamma\left(\sum_k \alpha_k\right)}.$$

When the vector parameter $\boldsymbol{\alpha}$ is filled with the same value α_0, the distribution is called Symmetric-Dirichlet with parameter α_0.

2 Each distribution symbolyze its density funcion.

Proposition 2. *Let* $(X_0, \dots, X_n) \sim$ *Dirichlet*$(\alpha_0, \dots, \alpha_n)$*, then* $X_0 \sim$ *Beta*$(\alpha_0, \alpha_1 + \cdots + \alpha_n)$.

Proof. Following Farrow (2008), we can write the joint probability as

$$f(x_1, \dots, x_n) = f_1(x_1) f_2(x_2 \mid x_1) \dots f_{n-1}(x_{n-1} \mid x_1, \dots, x_{n-2}).$$

We do not need the last term because it is fixed given the others. In fact, let $A = \sum_i \alpha_i$, we can write it as

$$\left(\frac{\Gamma(A)}{\Gamma(\alpha_1)\Gamma(A-\alpha_1)} x_1^{\alpha_1 - 1}(1-x_1)^{A-\alpha_1-1}\right)\left(\frac{\Gamma(A-\alpha_1)}{\Gamma(\alpha_2)\Gamma(A-\alpha_1-\alpha_2)} \frac{x_2^{\alpha_2-1}(1-x_1-x_2)^{A-\alpha_2-\alpha_1-1}}{(1-\alpha_1)^{A-\alpha_1-1}}\right)$$

$$\dots \left(\frac{\Gamma(A-\alpha_1-\cdots-\alpha_{n-2})}{\Gamma(\alpha_{n-1})\Gamma(A-\alpha_1-\cdots-\alpha_{n-1})} \frac{x_{n-1}^{\alpha_n-1} x_n^{\alpha_n-1}}{(1-x_1-\cdots-x_{n-2})^{\alpha_{n-1}+\alpha_n-1}}\right).$$

From this, we get that

$$f_1(x_1) = \frac{\Gamma(A)}{\Gamma(\alpha_1)\Gamma(A-\alpha_1)} x_1^{\alpha_1-1}(1-x_1)^{A-\alpha_1-1} \implies X_1 \sim \text{Beta}(\alpha_1, A-\alpha_1).$$

Making the decomposition over any other X_j, results on $X_j \sim \text{Beta}(\alpha_j, A-\alpha_j)$. □

2.3 KULLBACK-LEIBLER DIVERGENCE

Definition 26. Let P and Q be two probability distributions over the same probability space Ω, the *Kullback-Leibler divergence* $KL\Big(Q \mid P\Big)$ measures the "difference" between both distributions as

$$KL\Big(Q \mid P\Big) = \mathbb{E}_Q\Big[\log Q(x) - \log P(x)\Big].$$

The Kullback-Leibler divergence is defined if and only if for all $x \in \Omega$ such that $P(x) = 0$, then $Q(x) = 0$. In measure terms, Q is absolutely continuous with respect to P.

Proposition 3. *The Kullback-Leibler divergence is always non-negative.*

Proof. As the logarithm is bounded by $x - 1$, we can bound $\log \frac{P(x)}{Q(x)}$

$$\log x \leq x - 1 \implies \frac{P(x)}{Q(x)} - 1 \geq \log \frac{P(x)}{Q(x)}.$$

Since probabilities are non-negative, we can multiply by $Q(x)$ in the last inequality

$$P(x) - Q(x) \geq Q(x) \log \frac{P(x)}{Q(x)} = Q(x) \log P(x) - Q(x) \log Q(x).$$

Now we integrate (sum in case of discrete variables) both sides

$$0 \geq \mathbb{E}_Q\Big[\log P(x) - \log Q(x)\Big] \implies \mathbb{E}_Q\Big[\log Q(x) - \log P(x)\Big] \geq 0.$$

□

As a result, the Kullback-Leibler divergence is 0 if and only if the two distributions are equal almost everywhere.

3

GRAPH THEORY

Definition 27. A *graph* $G = (V, E)$ is a set of vertices or nodes V and edges $E \subset V \times V$ between them. If V is a set of ordered pairs then the graph is called a *directed graph*, otherwise if V is a set of unordered pairs it is called an *undirected graph*.

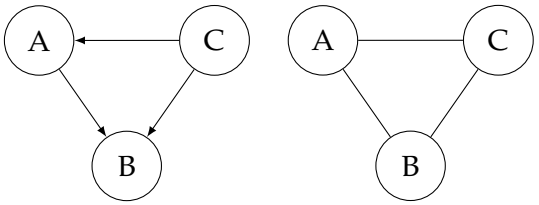


Figure 1: Example of directed and undirected graph, respectively.

Definition 28. In a directed graph $G = (V, E)$, a *directed path* $A \to B$ is a sequence of vertices $A = A_0, A_1, \ldots, A_{n-1}, A_n = B$ where $(A_i, A_{i+1}) \in E \ \forall i \in 0, \ldots, n-1$.

If G is a undirected graph, $A \to B$ is an *undirected path* if $\{A_i, A_{i+1}\} \in E \ \forall i \in 0, \ldots, n-1$

Definition 29. Let A, B be two vertices of a directed graph G. If $A \to B$ is a directed path and $B \not\to A$ (meaning there isn't a directed path from B to A), then A is called an *ancestor* of B and B is called a *descendant* of A.

For example, in the Figure 1, C is an ancestor of B.

Definition 30. A *directed acyclic graph (DAG)* is a directed graph such that no directed path between any two nodes revisits a vertex.

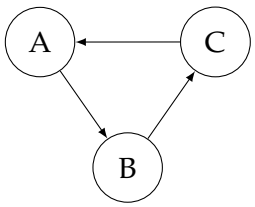


Figure 2: Example of graph which is not acyclic.

As we can see in the Figure 2, $A \to B \to C \to A \to B$ is a path from A to B that revisits A.

Now where are going to define some relations between nodes in a DAG.

Definition 31. The *parents* of a node A is the set of nodes B such that there is a directed edge from B to A. The same applies for the *children* of a node.

The *Markov blanket* of a node is composed by the node itself, its children, its parents and the parents of its children. The latter are usually called co-parents.

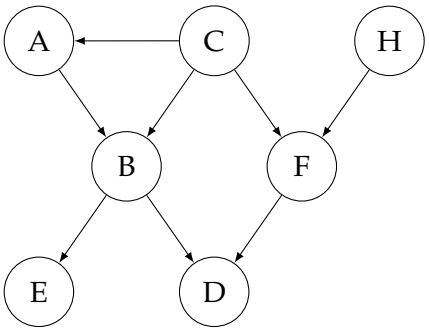


Figure 3: Directed acyclic graph

Definition 32. In a graph, the *neighbors* of a node are those directly connected to it.

We can use Figure 3 to reflect on these definitions. The parents of B are $pa(B) = \{A, C\}$ and its children are $ch(B) = \{E, D\}$. Taking this into account, its neighbors are $ne(B) = \{A, C, E, D\}$ and its Markov blanket is $\{A, B, C, D, E, F\}$.

Definition 33. Let G be a DAG, U be a path between two vertex and $A \in U$

- A is called a *collider* if $\forall B \in ne(A) \cap U, (B, A) \in E$.
- A is called a *fork* if $\forall B \in ne(A) \cap U, (A, B) \in E$.

Notice, a vertex can be a collider for a path but not for others. A vertex is said to be a collider or a fork without any reference to the path when it is for any path that goes through it. This happens when the edge direction condition is satisfied for every neighbor.

For example in Figure 3, D is a collider and C is a fork.

Definition 34. Let G be an undirected graph, a *clique* is a maximally connected subset of vertices. That is, all the members of the clique are connected to each others and there is no bigger clique that contains another.

Formally, $S \subset V$ is a *clique* if and only if $\forall A, B \in S,\ \{A, B\} \in E$ and $\nexists C \in V \backslash S$ such that $\forall A \in S,\ \{A, C\} \in E$.

Part II

STATISTICAL INFERENCE

Statistical inference is defined as the process of deducing properties of an underlying distribution using data analysis.

Bayesian inference is an statistical inference method in which Bayes' theorem is applied. The *posterior probability* is derived as a consequence of a *prior probability* and a *likelihood function*. Both of these are derived from a statistical model for the observed data.

Maximum likelihood estimation is a parameter estimation method which maximizes the *likelihood function* of the underlying parametric distribution. This method maximizes the parameters under which the observed data is most probable.

4

INTRODUCTION

Inferential statistical infer the underlying properties of a dataset or population, using different techniques as deriving estimates and testing hypotheses. This analysis are made under the assumption that the observed data is sampled from a larger population (Upton & Cook (2014)).

Compared to *inferential statistics*, *descriptive analysis* uniquely concerns about properties of the observed data and does not assume that this data comes from a larger population.

In the field of *machine learning*, rather than inference, the procedure for deducing properties of the model is commonly referred to as *training or learning*, in contrast, *inference* refers to using a model for prediction.

A *statistical model* is a set of assumptions concerning how the observer data was generated (Cox (2006)). There are different levels of modeling assumptions, which differ on whether the process that generates the data, is fully, partially or minimally described by a family of probability distributions. These distributions involve a finite amount of unknown parameters. This study's approach is *fully parametric*, which assumes this generation is fully described by those parameters.

Different schools or paradigms of statistical inference have become established (Bandyopadhyay & Forster (2011)). These are not mutually exclusive, meaning that methods that work well on a given paradigm might have interpretations on other paradigms. In this chapter we are reviewing two of them: the *Bayesian paradigm* and the *likelihoodist paradigm*.

As purely Bayesian or likelihoodist methods are beyond the scope of this study, we are just reviewing the needed definitions to later understand related *variational methods*, a few example are analyzed in each section.

5

MAXIMUM LIKELIHOOD

Given a set of observations $\boldsymbol{x}$ and parameters $\boldsymbol{\theta}$ that model the obtained samples via $P(\boldsymbol{x} \mid \boldsymbol{\theta})$, *maximum likelihood estimation* (Rossi (2018)) is a *classical inference* method of estimating the maximum likelihood parameters, i.e, the ones that maximizes the likelihood $P(\boldsymbol{x} \mid \boldsymbol{\theta})$:

$$\boldsymbol{\theta}^{ML} = \arg\max_{\boldsymbol{\theta}} P(\boldsymbol{x} \mid \boldsymbol{\theta}).$$

This value symbolizes the *value of the parameter to which the data is most probable to be generated with*. There are several techniques for finding this value, for example, if the likelihood function is differentiable, its derivative can be used to determine the maxima.

Remark 1. Maximum likelihood estimation does not consider a probability distribution over the parameter, whereas Bayesian estimation does.

5.1 MAXIMUM LIKELIHOOD AND THE EMPIRICAL DISTRIBUTION

Consider a set of i.i.d random variables $\boldsymbol{X} = (X_1, \ldots, X_N)$ and their observations $\boldsymbol{x} = (x_1, \ldots, x_N)$, we are showing the relation between the maximum likelihood and the Kullback-Leibler divergence of the empirical and the underlying distribution. The empirical distribution is defined as the distribution whose probability mass function Q is

$$Q(x) = \frac{1}{N} \sum_{n=1}^{N} \mathbb{I}[x = x_n].$$

Where X is i.i.d with the rest of variables we are considering. That is, it establish the probability of an outcome as the proportional times it appears in the set ob observations.

Proposition 4. *In case $P(x \mid \boldsymbol{\theta})$ is unconstrained, maximum likelihood distribution corresponds to the empirical distribution, i.e, $P(x \mid \boldsymbol{\theta}^{ML}) = Q(x)$.*

Proof. The Kullback-Leibler divergence between the empirical and our considered model $P(x \mid \boldsymbol{\theta})$ is:

$$KL\Big(Q \mid P\Big) = \mathbb{E}_Q\Big[\log Q(x)\Big] - \mathbb{E}_Q\Big[\log P(x \mid \boldsymbol{\theta})\Big].$$

Notice the term $\mathbb{E}_Q\Big[\log Q(\boldsymbol{x})\Big]$ is a constant and the log likelihood under Q takes the form

$$\mathbb{E}_Q\Big[\log P(x \mid \boldsymbol{\theta})\Big] = \frac{1}{N}\int_x \sum_{n=1}^{N} \mathbb{I}[x = x_n] \log P(x \mid \boldsymbol{\theta}) = \frac{1}{N}\sum_{n=1}^{N} \log P(x_n \mid \boldsymbol{\theta}).$$

As the logarithm is a strictly increasing function, maximizing the log likelihood equals to maximize the likelihood itself, in conclusion, it is equivalent to minimize the Kullback-Leibler divergence between the empirical distribution Q and our distribution P.

$$\arg\min_{\boldsymbol{\theta}} KL\Big(Q \mid P\Big) = \arg\min_{\boldsymbol{\theta}} -\mathbb{E}_Q\Big[\log P(\boldsymbol{x} \mid \boldsymbol{\theta})\Big] = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_Q\Big[\log P(\boldsymbol{x} \mid \boldsymbol{\theta})\Big] =$$
$$\arg\max_{\boldsymbol{\theta}} \frac{1}{N}\sum_{n=1}^{N} \log P(x_n \mid \boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \frac{1}{N}\sum_{n=1}^{N} P(x_n \mid \boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \sum_{n=1}^{N} P(x_n \mid \boldsymbol{\theta}) = \boldsymbol{\theta}^{ML}.$$

□

6

BAYESIAN INFERENCE

Bayesian inference considers a probability distribution over the set of parameters. This distribution is then governed by so called *hyper-parameters* $\boldsymbol{\alpha}$, these are typically omitted when using $P(\boldsymbol{\theta} \mid \boldsymbol{\alpha})$, where $P(\boldsymbol{\theta})$ is written instead.

Bayesian inference attempts to determine the *posterior distribution* $P(\boldsymbol{\theta} \mid \boldsymbol{x})$ using a *prior belief* $P(\boldsymbol{\theta})$ and the *likelihood function* $P(\boldsymbol{x} \mid \boldsymbol{\theta})$ on the basis of Bayes' theorem:

$$P(\boldsymbol{\theta} \mid \boldsymbol{x}) = \frac{P(\boldsymbol{x} \mid \boldsymbol{\theta})P(\boldsymbol{\theta})}{P(\boldsymbol{x})}.$$

6.1 EXAMPLE: DISCRETE PRIOR

For this example, consider a set of i.i.d variables $\boldsymbol{X} = (X_1, \ldots, X_N)$ with their corresponding set of observations be $\boldsymbol{x} = (x_1, \ldots, x_N)$, where each X models the results of coin-tossing experiment, let 1 symbolize *heads* and 0 *tails*.

Bayesian inference attempts to estimate the probability distribution of θ given $\boldsymbol{x}$. This parameter models the probability of the tossing resulting in heads as

$$P(x_n = 1 \mid \theta) = \theta \quad \forall n \in \{1, \ldots, N\}.$$

The joint probability factorizes using the i.i.d assumption

$$P(\boldsymbol{x}, \theta) = P(\theta) \prod_{n=1}^{N} P(x_n \mid \theta).$$

The objective of Bayesian inference is then to compute the posterior distribution,

$$P(\theta \mid \boldsymbol{x}) = \frac{P(\boldsymbol{x} \mid \theta)P(\theta)}{P(\boldsymbol{x})},$$

to do so, we need to specify the prior distribution $P(\theta)$. For now, we are using a discrete variable that verifies:

$$P(\theta = 0.2) = 0.1, \quad P(\theta = 0.5) = 0.7 \quad \text{and} \quad P(\theta = 0.8) = 0.2.$$

This means that we have a 70% belief that the coin is fair, a 10% belief that is biased to tails and 20% that is biased to heads. Let n_h be the number of heads in our observed data and n_t the number of tails, mathematically:

$$n_h = \#\{x \in \boldsymbol{x} \ : \ x = 1\} \quad \text{and} \quad n_t = \#\{x \in \boldsymbol{x} \ : \ x = 0\}.$$

Given that $X_n \mid \theta$ follows a Bernoulli distribution $\forall n \in \{1, \ldots, N\}$, where the posterior is

$$P(\theta \mid \boldsymbol{x}) = \frac{P(\theta)}{P(\boldsymbol{x})} \theta^{n_h} (1-\theta)^{n_t}.$$

Suppose that $n_h = 2$ and $n_t = 8$, the posterior might be calculated up to a normalization factor:

$$P(\theta = 0.2 \mid \boldsymbol{x}) = \frac{1}{P(\boldsymbol{x})} \times 0.1 \times 0.2^2 \times 0.8^8 = \frac{1}{P(\boldsymbol{x})} \times 6.71 \times 10^{-4},$$
$$P(\theta = 0.5 \mid \boldsymbol{x}) = \frac{1}{P(\boldsymbol{x})} \times 0.7 \times 0.5^2 \times 0.5^8 = \frac{1}{P(\boldsymbol{x})} \times 6.83 \times 10^{-4},$$
$$P(\theta = 0.8 \mid \boldsymbol{x}) = \frac{1}{P(\boldsymbol{x})} \times 0.2 \times 0.2^2 \times 0.8^8 = \frac{1}{P(\boldsymbol{x})} \times 3.27 \times 10^{-7}.$$

We can compute the normalizing factor as

$$P(\boldsymbol{x}) = \sum_{\theta \in \{0.2, 0.5, 0.8\}} P(\boldsymbol{x}, \theta) = 6.71 \times 10^{-4} + 6.83 \times 10^{-4} + 3.27 \times 10^{-7} = 0.00135.$$

Therefore, the posterior is

$$P(\theta = 0.2 \mid \boldsymbol{x}) = 0.4979,$$
$$P(\theta = 0.5 \mid \boldsymbol{x}) = 0.5059,$$
$$P(\theta = 0.8 \mid \boldsymbol{x}) = 0.00024.$$

6.2 EXAMPLE: CONTINUOUS PRIOR

In the previous example, we have used a discrete prior for the parameter distribution, a continuous prior might be chosen instead. Suppose an uniform prior distribution:

$$P(\theta) = k \implies \int_0^1 P(\theta) d\theta = k = 1$$

due to normalization.

Using the previous calculations we have

$$P(\theta \mid \boldsymbol{x}) = \frac{1}{P(\boldsymbol{x})} \theta^{n_h} (1-\theta)^{n_t},$$

where

$$P(\boldsymbol{x}) = \int_0^1 \theta^{n_h} (1-\theta)^{n_t} d\theta.$$

This implies that

$$P(\theta \mid \boldsymbol{x}) = \frac{\theta^{n_h} (1-\theta)^{n_t}}{\int_0^1 u^{n_h} (1-u)^{n_t} du} \implies \theta \mid \boldsymbol{x} \sim Beta(n_h + 1, n_t + 1).$$

A Beta distribution could be also considered as the prior distribution:

$$\theta \sim \text{Beta}(\alpha, \beta) \implies P(\theta) = \frac{1}{B(\alpha, \beta)} \theta^{\alpha - 1} (1-\theta)^{\beta - 1}.$$

in this case, the posterior is:

$$P(\theta, \boldsymbol{x}) = \frac{1}{B(\alpha + n_h, \beta + n_t)} \theta^{\alpha + n_h - 1} (1-\theta)^{\beta + n_t - 1} \implies \theta \mid \boldsymbol{x} \sim Beta(n_h + \alpha, n_t + \beta)$$

6.3 UTILITY FUNCTION

The Bayesian posterior says nothing about how to benefit from the beliefs it represents, in order to do this we need to specify the utility of each decision.

With this idea we define an utility function over the parameters

$$U(\theta, \theta_{true}) = \alpha \mathbb{I}[\theta = \theta_{true}] - \beta \mathbb{I}[\theta \neq \theta_{true}],$$

where $\alpha, \beta \in \mathbb{R}$. This symbolizes the gains or looses of choosing the parameter θ, when the true value of the parameter is supposed to be θ_{true}. Therefore, the expected utility of a parameter θ_0 is calculated as

$$U(\theta = \theta_0) = \int_{\theta_{true}} U(\theta = \theta_0, \theta_{true}) P(\theta = \theta_{true} \mid \boldsymbol{x}).$$

We might as well define an utility function over the previous example:

$$U(\theta, \theta_{true}) = 10 \mathbb{I}[\theta = \theta_{true}] - 20 \mathbb{I}[\theta \neq \theta_{true}],$$

where we interpret that the loss of choosing the wrong parameter is twice as important as the gains from doing it right.

The expected utility of the decision that the parameter is $\theta = 0.2$ in our discrete example would be

$$\begin{aligned}
U(\theta = 0.2) &= U(\theta = 0.2, \theta_{true} = 0.2) P(\theta_{true} = 0.2 \mid \boldsymbol{x}) \\
&\quad + U(\theta = 0.2, \theta_{true} = 0.5) P(\theta_{true} = 0.5 \mid \boldsymbol{x}) \\
&\quad + U(\theta = 0.2, \theta_{true} = 0.8) P(\theta_{true} = 0.8 \mid \boldsymbol{x}) \\
&= 10 \times 0.4979 - 20 \times 0.5059 - 20 \times 0.00024 \\
&= -5.1438, \\
U(\theta = 0.5) &= -4.9038, \\
U(\theta = 0.8) &= -20.0736.
\end{aligned}$$

Given this, if we had to make a decision for the parameter, we could choose the value with the highest utility. Other approaches like the mode or mean (continuous posterior) of the distribution are possible.

6.4 MAXIMUM A POSTERIORI ESTIMATION

Maximum a posteriori probability estimation is a Bayesian inference method of estimating the mode of the posterior distribution. In contrast to maximum likelihood estimation, it employs an augmented optimization objective which incorporates a prior distribution.

Definition 35. *Maximum A Posteriori (MAP)* refers to the value of the parameter $\boldsymbol{\theta}$ that better fits the data:

$$\boldsymbol{\theta}^{MAP} = \arg\max_{\boldsymbol{\theta}} P(\boldsymbol{x} \mid \boldsymbol{\theta}) P(\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} P(\boldsymbol{\theta} \mid \boldsymbol{x}).$$

Remark 2. Maximum likelihood estimation is a particular case of maximum a posterior estimation with a flat (constant) prior.

Remark 3. MAP estimation can be seen as a limiting case of Bayesian estimation under the $0-1$ utility function:

$$U(\boldsymbol{\theta}, \boldsymbol{\theta}_{true}) = \mathbb{I}[\boldsymbol{\theta} = \boldsymbol{\theta}_{true}],$$

using this, the expected utility of a parameter $\boldsymbol{\theta} = \boldsymbol{\theta}_0$ is

$$U(\boldsymbol{\theta} = \boldsymbol{\theta}_0) = \int_{\boldsymbol{\theta}_{true}} \mathbb{I}[\boldsymbol{\theta}_{true} = \boldsymbol{\theta}_0] P(\boldsymbol{\theta} = \boldsymbol{\theta}_{true} \mid \boldsymbol{x}) = P(\boldsymbol{\theta}_0 \mid \boldsymbol{x}).$$

This means that the maximum utility decision is to take the value $\boldsymbol{\theta}_0$ with the highest posterior value.

7

MISSING VARIABLES

When dealing with real datasets, these might not be completed and *missing information* might appear, that is, states of visible data that are missing. This differs from hidden variables in that the observation should exists but may be missing due to human errors. However, no missing information is assumed in this study. Therefore, we are just reviewing a few concepts and definitions.

There are three main types of missing data:

- **Missing completely at random (MCAR)**. If the events that lead to any particular data to be missing is independent from both the observed and the unobserved variables, and occur at random.
- **Missing at random (MAR)**. When the absence is not random but can be explained with the observed variables.
- **Missing not at random (MNAR)**. The missing data is related with the reason why it is missing. For example, skipping a question in a survey for being ashamed of the answer.

To express this mathematically, consider a single observation and split the variables $\boldsymbol{X}$ into visible $\boldsymbol{X}_{vis}$ and hidden $\boldsymbol{X}_{hid}$, let M be a variable denoting that the state of the hidden variables is known (0) or unknown (1). The difference between the three types resides on how $P(M = 1 \mid \boldsymbol{x}_{vis}, \boldsymbol{x}_{hid}, \boldsymbol{\theta})$ simplifies. This affects how the likelihood function $P(\boldsymbol{x}_{vis}, M = 1 \mid \boldsymbol{\theta})$ factorizes.

When data is *missing at random*, we assume that we can explain the missing information with the visible one, so the probability of being missing only depends on the visible data, that is

$$P(M = 1 \mid \boldsymbol{x}_{vis}, \boldsymbol{x}_{hid}, \boldsymbol{\theta}) = P(M = 1 \mid \boldsymbol{x}_{vis}).$$

Thus, the likelihood is

$$P(\boldsymbol{x}_{vis}, M = 1 \mid \boldsymbol{\theta}) = P(M = 1 \mid \boldsymbol{x}_{vis}) P(\boldsymbol{x}_{vis} \mid \boldsymbol{\theta}).$$

Assuming the data is *missing completely at random* is stronger, as we are supposing that there is no reason behind the missing data, so that it being missing is independent from the visible and hidden data:

$$P(M = 1 \mid \boldsymbol{x}_{vis}, \boldsymbol{x}_{hid}, \boldsymbol{\theta}) = P(M = 1),$$

so now the likelihood takes the form,

$$P(\boldsymbol{x}_{vis}, M = 1 \mid \boldsymbol{\theta}) = P(M = 1) P(\boldsymbol{x}_{vis} \mid \boldsymbol{\theta}).$$

In both cases we may simply use the marginal $P(\boldsymbol{x}_{vis} \mid \boldsymbol{\theta})$ to assess parameters as the likelihood does not depend on the missing variables.

In case data is *missing not at random*, no independence assumption is made over the probability of the data being unknown, meaning it depends on both the visible and the hidden information. Assuming that missing information is either MAR or MCAR could lead to a misunderstanding of the problem as in the following simple example.

Example 2. Consider a situation where data is obtained from a survey where people are asked to choose between 3 options A, B and C. Assume that no one chose option C because they are ashamed of the answer, and those answers are uniform between A, B and not answering.

Normalizing the missing information would lead to setting $P(A \mid \boldsymbol{x}) = 0.5 = P(B \mid \boldsymbol{x})$ and $P(C \mid \boldsymbol{x}) = 0$ when the reasonable result is that not answering equals to choosing C so that $P(A \mid \boldsymbol{x}) = P(B \mid \boldsymbol{x}) = P(C \mid \boldsymbol{x}) = \frac{1}{3}$

Part III

VARIATIONAL INFERENCE

Variational Bayesian methods consist of approximation techniques applied to intractable integrals that arise in inference and machine learning problems. Theese methods are commonly used in complex models containing *observed and latent variables* and *unknown parameters*. *Graphical models* might be used to describe the underlying relations of the given variables.

Variational Bayesian inference solves the inference problem by creating an equivalent *optimization problem* and approaching its solution through machine learning techniques.

8

INTRODUCTION

In Bayesian inference, both parameters are commonly clustered with *latent or hidden variables*. *Variational Bayesian methods* or simply *variational methods* are primarily used for two purposes:

1. Perform statistical inference over the unobserved variables by provide an analytical approximation to their posterior probability.
2. Derive and compute a lower bound for the observed data marginal likelihood, that is, the marginal probability of the observed data data over the unobserved variables. This is commonly used to perform model selection, where a model with a higher marginal likelihood has a greater probability for being the model that generated the data.

The considered elements of a variational Bayesian model are: a set of observed variables $\boldsymbol{X} = (X_1, \ldots, X_N)$ among with hidden variables $\boldsymbol{Z} = (Z_1, \ldots, Z_M)$ (which includes parameters). Their corresponding set of observations $\boldsymbol{x} = (x_1, \ldots, x_N)$, $\boldsymbol{z} = (z_1, \ldots, z_M)$, where the latter denotes a possible configuration for the hidden variables.

The samples are governed by the joint distribution $P(\boldsymbol{x}, \boldsymbol{z})$. The same way that *Bayesian* inference tries to learn the posterior distribution of the parameters, inference with latent variables consists of learning the posterior distribution of those, $P(\boldsymbol{z} \mid \boldsymbol{x})$, given the dataset $\boldsymbol{x}$.

In *classical inference* the conditional is calculated as

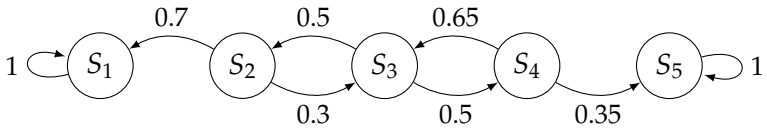
$$P(\boldsymbol{z} \mid \boldsymbol{x}) = \frac{P(\boldsymbol{x}, \boldsymbol{z})}{\int_{\boldsymbol{z}} P(\boldsymbol{x}, \boldsymbol{z})} ,$$

but for many models this integral is computationally hard to solve.

As we have already reviewed using parameters, *Bayesian inference* derives the posterior probability $P(\boldsymbol{z} \mid \boldsymbol{x})$ as a consequence of a *prior probability* $P(\boldsymbol{z})$ and a *likelihood function* $P(\boldsymbol{x} \mid \boldsymbol{z})$. Other methods as *Markov chain Monte Carlo (MCMC)* and *variational Bayesian inference* try a different approach when solving the given inference problem: on one hand, *variational inference* uses machine learning methods whose main goal is to approximate probability distributions (Jordan *et al.* (1999); Wainwright & Jordan (2008)). On the other hand, *MCMC* approximates the posterior distribution using a Markov chain. Let us briefly introduce the main idea behind this method, we need to introduce two concepts: *Markov chain* and *MCMC*.

A *Markov Chain* is formally defined as a stochastic process, i.e, a family of random variables, that satisfies the *Markov property* also known as the memoryless property: *the conditional*

probability distribution of future states of the process (conditional on both present and past values) depends only on the present state. To fully understand it, imagine a system with a number of possible states $S_1, \ldots, S_5$ and the probabilities of going from one state to another stated in the following diagram.



Consider a sequence of random variables X_t that symbolize the current state at the step t. The Markov property means that the probability of moving to the next state depends only on the present one, i.e,

$$P(X_{n+1} = x \mid X_1 = x_1 \ldots, X_n = x_n) = P(X_{n+1} = x \mid X_n = x_n).$$

We need two concepts to define the process of MCMC:

- **Ergodic Markov chain**. A Markov chain where it exists a number $N \in \mathbb{N}$ such that any state can be reached from any other state in any number of steps less or equal than N.
- **Stationary distribution**. The probability distribution to which the process converges over time.

In MCMC, an ergodic Markov chain over the latent variables $\boldsymbol{Z}$ is considered, whose stationary distribution is the posterior $P(\boldsymbol{z} \mid \boldsymbol{x})$, samples are taken from the chain to approximate the posterior with them.

In contrast, *variational inference* exchanges the inference problem with an optimization one. It fixes a family of distributions $\mathcal{Q}$ over the latent variables $\boldsymbol{Z}$ and find the element that minimizes its Kullback-Leibler divergence with the posterior $P(\boldsymbol{z} \mid \boldsymbol{x})$:

$$Q^{opt} = \arg\min_{Q \in \mathcal{Q}} KL\Big(Q(\boldsymbol{z}) \mid P(\boldsymbol{z} \mid \boldsymbol{x})\Big).$$

These Q distributions are typically referred as *variational distributions* of the optimization problem.

Compared to *Markov Chain Monte Carlo (MCMC)*, variational inference tends to be faster and scale easier to large data (Blei *et al.* (2017)), it has been applied to different problems such as computer vision, computational neuroscience and document analysis (Blei (2014)). Monte Carlo methods are usually very intense computationally and suffer from difficulties in diagnosing convergence (Winn & Bishop (2005)).

Analyzing the Kullback-Leibler divergence, it may be decomposed in the following way:

$$\begin{aligned}
KL\Big(Q(\boldsymbol{z}) \mid P(\boldsymbol{z} \mid \boldsymbol{x})\Big) &= \mathbb{E}_{Q(\boldsymbol{z})}\Big[\log Q(\boldsymbol{z})\Big] - \mathbb{E}_{Q(\boldsymbol{z})}\Big[\log P(\boldsymbol{z} \mid \boldsymbol{x})\Big] \\
&= \mathbb{E}_{Q(\boldsymbol{z})}\Big[\log Q(\boldsymbol{z})\Big] - \mathbb{E}_{Q(\boldsymbol{z})}\Big[\log P(\boldsymbol{x}, \boldsymbol{z}) - \log P(\boldsymbol{x})\Big]. \\
&= \mathbb{E}_{Q(\boldsymbol{z})}\Big[\log Q(\boldsymbol{z})\Big] - \mathbb{E}_{Q(\boldsymbol{z})}\Big[\log P(\boldsymbol{x}, \boldsymbol{z})\Big] + \mathbb{E}_{Q(\boldsymbol{z})}\Big[\log P(\boldsymbol{x})\Big]. \\
&= \mathbb{E}_{Q(\boldsymbol{z})}\Big[\log Q(\boldsymbol{z})\Big] - \mathbb{E}_{Q(\boldsymbol{z})}\Big[\log P(\boldsymbol{x}, \boldsymbol{z})\Big] + \log P(\boldsymbol{x}).
\end{aligned}$$

Although the Kullback-Leibler divergence cannot be computed ($P(\boldsymbol{z} \mid \boldsymbol{x})$ is unknown), we can optimize an equivalent objective: we can use its positiveness to set the following lower bound to the evidence, defined as *evidence lower bound* or *ELBO*.

$$\log P(\boldsymbol{x}) \geq \underbrace{-\mathbb{E}_{Q(\boldsymbol{z})}\Big[\log Q(\boldsymbol{z})\Big]}_{Entropy} + \underbrace{\mathbb{E}_{Q(\boldsymbol{z})}\Big[\log P(\boldsymbol{x}, \boldsymbol{z})\Big]}_{Energy} = \text{ELBO}(Q).^{1}$$

As $P(x)$ does not depend on Q, minimizing the Kullback-Leibler divergence is equivalent to maximize the ELBO as equality holds if and only if $Q(\boldsymbol{z}) = P(\boldsymbol{z} \mid \boldsymbol{x})$. The ELBO may be written as

$$\begin{aligned}\text{ELBO}(Q) &= \mathbb{E}_{Q(\boldsymbol{z})}\Big[\log P(\boldsymbol{z})\Big] + \mathbb{E}_{Q(\boldsymbol{z})}\Big[\log P(\boldsymbol{x} \mid \boldsymbol{z})\Big] - \mathbb{E}_{Q(\boldsymbol{z})}\Big[\log Q(\boldsymbol{z})\Big] \\ &= \mathbb{E}_{Q(\boldsymbol{z})}\Big[\log P(\boldsymbol{x} \mid \boldsymbol{z})\Big] - KL\Big(Q(\boldsymbol{z}) \mid P(\boldsymbol{z})\Big),\end{aligned}$$

where it is expressed as the sum of the log likelihood of the observations and the Kullback-Leibler divergence between the prior $P(\boldsymbol{z})$ and $Q(\boldsymbol{z})$.

The *expectation maximization algorithm* and *coordinate ascent variational inference* are two algorithms designed to optimize this lower bound in order to solve the optimization problem we are focusing.

1 Energy and Entropy terms come from a statistical physics terminology (Barber (2007))

9

EXPECTATION MAXIMIZATION

The *expectation maximization (EM)* algorithm (Dempster *et al.* (1977); McLachlan & Krishnan (2007); Bishop (2006)) is a partially non-Bayesian, likelihoodist iterative method to find maximum likelihood estimates of parameters in statistical models where the model depends on latent variables.

The algorithm consists of a two step iteration where the first optimizes the variational distribution element Q and the second optimizes the set of parameters $\boldsymbol{\theta}$. A fully Bayesian version would consider a probability distribution over the parameter, where the distinction between the two steps disappears. In that case, as many steps as latent variables (including the parameters) are needed per iteration, where each variable is optimized at a time. For *graphical models* this is easy to compute as each variable's new variational distribution depends only on its *Markov blanket*, so local *message passing* can be used for efficient inference (Chapter 20.1).

Using the same notation as in Chapter 8, EM's iterative procedure increases the ELBO for the parametric marginal $\log P(\boldsymbol{x} \mid \boldsymbol{\theta})$,

$$\log P(\boldsymbol{x} \mid \boldsymbol{\theta}) \geq \underbrace{-\mathbb{E}_{Q(\boldsymbol{z})}\Big[\log Q(\boldsymbol{z})\Big]}_{\text{Entropy}} + \underbrace{\mathbb{E}_{Q(\boldsymbol{z})}\Big[\log P(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta})\Big]}_{\text{Energy}},$$

and the marginal likelihood $P(\boldsymbol{x} \mid \boldsymbol{\theta})$ itself. Which means that aims for the value of $\boldsymbol{\theta}$ to which the dataset better fits the model, i.e, the maximum likelihood parameter.

The EM algorithm can be viewed as two alternating maximization steps, that is, as an example of *coordinate ascent* (Neal & Hinton (1998)), an optimization algorithm that successively maximizes each coordinate to find a maxima. Consider the above ELBO as a function of Q and $\boldsymbol{\theta}$:

$$ELBO(Q, \boldsymbol{\theta}) = -\mathbb{E}_{Q(\boldsymbol{z})}\Big[\log Q(\boldsymbol{z})\Big] + \mathbb{E}_{Q(\boldsymbol{z})}\Big[\log P(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta})\Big].$$

Then, the EM algorithm consists on:

- **E-step**. For fixed $\boldsymbol{\theta}$, choose Q such as:

$$Q^{new} = \arg\max_{Q} ELBO(Q, \boldsymbol{\theta}),$$

  which is equivalent to:

$$Q^{new}(\boldsymbol{z}) = P(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{\theta}).$$

Algorithm 1: Expectation Maximization Algorithm

Data: A dataset $\boldsymbol{x}$ and a distribution $P(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta})$.
Result: Approximation of the maximum likelihood parameter.
Initialize $\boldsymbol{\theta}^{(0)}$;
$t = 1$;
while *Convergence stop criteria* **do**
  $t \leftarrow t + 1$;
  $Q^{(t)} = P(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{\theta}^{(t-1)})$;
  $\boldsymbol{\theta}^{(t)} = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{Q^{(t)}(\boldsymbol{z})}\Big[\log P(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta})\Big]$;
end
return θ;

- **M-step**. For fixed Q, choose $\boldsymbol{\theta}$ such as:

$$\boldsymbol{\theta}^{new} = \arg\max_{\boldsymbol{\theta}} ELBO(Q, \boldsymbol{\theta}).$$

  As Q does not depend on the parameter, this is equivalent to maximize the energy term:

$$\boldsymbol{\theta}^{new} = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{Q(\boldsymbol{z})}\Big[\log P(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta})\Big].$$

In the specific situation where $\boldsymbol{x}$ consists on N observations of the same variable X and each observation of X is related with a single hidden variable Z, for example, the case of a mixture distribution, the following considerations might be taken:

- Observations are treated as observations of i.i.d variables $X_1, \ldots, X_N$ and $Z_1, \ldots, Z_N$.
- The variational distribution Q now factorizes over the hidden variables as it is known that they are independent from each other:

$$Q(\boldsymbol{z}) = \prod_{n=1}^{N} Q(z_n).$$

  Notice that the same letter Q is being used for each variable Z_n, this is just notational, in practice there must be a variational distribution for each of these i.i.d variables.

- The model distribution P factorizes over the variables as:

$$P(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta}) = \prod_{n=1}^{N} P(x_n, z_n \mid \boldsymbol{\theta}).$$

- The lower bound is then written as

$$\begin{aligned}\log P(x_1, \ldots, x_N \mid \boldsymbol{\theta}) &= \sum_{n=1}^{N} \log P(x_n \mid \boldsymbol{\theta}) \\ &\geq \sum_{n=1}^{N} -\mathbb{E}_{Q(z_n)}\Big[\log Q(z_n)\Big] + \mathbb{E}_{Q(z_n)}\Big[\log P(x_n, z_n \mid \boldsymbol{\theta})\Big].\end{aligned}$$

  Where equality holds if and only if $Q(z_n) = P(z_n \mid x_n, \boldsymbol{\theta})\ \forall n = 1, \ldots, N$.

The procedure to optimize the parameter consists in two steps:

- **E-step**. For fixed θ, find the distributions that maximize the above bound, i.e, choose $Q^{new}(z_n) = P(z_n \mid x_n, \theta)\ \forall n = 1, \ldots, N$.
- **M-step**. For a fixed distribution Q, find the parameter θ that maximizes the bound:

$$\theta^{new} = \arg\max_{\theta} \sum_{n=1}^{N} \mathbb{E}_{Q(z)}\Big[\log P(z_n, x_n \mid \theta)\Big].$$

Example 3. Consider a single variable X with $Dom(X) = \mathbb{R}$ and a single hidden variable Z with $dom(Z) = \{1, 2\}$. Let the conditional probability be:

$$P(x \mid z, \theta) = \frac{1}{\sqrt{\pi}} e^{-(x-\theta z)^2},$$

and $P(Z = 1) = P(Z = 2) = 0.5$. Suppose an observation $x = 2.75$ we want to optimize the parameter θ in the marginal

$$P(X = 2.75 \mid \theta) = \sum_{z \in \{1,2\}} P(X = 2.75 \mid z, \theta) P(z) = \frac{1}{2\sqrt{\pi}} \big(e^{-(2.75-\theta)^2} + e^{-(2.75-2\theta)^2}\big).$$

Then using a distribution $Q(z)$, the lower bound given to the log likelihood is

$$\log P(x \mid \theta) \geq -Q(1) \log Q(1) - Q(2) \log Q(2) - \mathbb{E}_Q\Big[(x - \theta z)^2\Big] + \text{const.}$$

The M-step can be done analytically, noticing that due to the negative sign we want to minimize $\mathbb{E}_Q\Big[(x - \theta z)^2\Big]$

$$\frac{d}{d\theta} \mathbb{E}_Q\Big[(x - \theta z)^2\Big] = \mathbb{E}_Q\Big[2xz + 2\theta z^2\Big] = 2x\mathbb{E}_Q\Big[z\Big] + 2\theta \mathbb{E}_Q\Big[z^2\Big] = 0 \iff \theta = \frac{x\mathbb{E}_Q\Big[z\Big]}{\mathbb{E}_Q\Big[z^2\Big]},$$

$$\frac{d^2}{d^2\theta} \mathbb{E}_Q\Big[(x - \theta z)^2\Big] = 2\mathbb{E}_Q\Big[z^2\Big] \geq 0,$$

so the new parameter optimal parameter is

$$\theta_{new} = x \frac{\mathbb{E}_Q\Big[z\Big]}{\mathbb{E}_Q\Big[z^2\Big]}.$$

The E-step would set $Q_{new}(z) = P(z \mid x, \theta)$, in this case

$$Q^{new}(Z = 1) = \frac{P(X = 2.75 \mid Z = 1, \theta) P(Z = 1)}{P(X = 2.75)} = \frac{e^{-(2.75-\theta)}}{e^{-(2.75-\theta)} + e^{-(2.75-2\theta)}},$$
$$Q^{new}(Z = 2) = \frac{P(X = 2.75 \mid Z = 2, \theta) P(Z = 2)}{P(X = 2.75)} = \frac{e^{-(2.75-2\theta)}}{e^{-(2.75-\theta)} + e^{-(2.75-2\theta)}}.$$

9.1 EM INCREASES THE MARGINAL LIKELIHOOD

Not only the EM algorithm does increase the lower bound in each iteration but also the marginal likelihood.

Proposition 5. *The log likelihood $P(\boldsymbol{x} \mid \boldsymbol{\theta})$ is not decreased in each iteration of the EM algorithm, i.e, if $\boldsymbol{\theta}^{old}$ and $\boldsymbol{\theta}^{new}$ are two consecutive values of the parameter, EM verifies:*

$$\log P(\boldsymbol{x} \mid \boldsymbol{\theta}^{new}) - \log P(\boldsymbol{x} \mid \boldsymbol{\theta}^{old}) \geq 0.$$

Proof. As a result of the E-step Q is to set

$$Q(\boldsymbol{z}) = P(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{\theta}^{old}),$$

the lower bound in terms of $\boldsymbol{\theta}^{old}$ and $\boldsymbol{\theta}^{new}$ is

$$ELBO(\boldsymbol{\theta}^{new} \mid \boldsymbol{\theta}^{old}) = \underbrace{-\mathbb{E}_{P(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{\theta}^{old})}\Big[\log P(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{\theta}^{old})\Big]}_{\text{Entropy}} + \underbrace{\mathbb{E}_{P(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{\theta}^{old})}\Big[\log P(\boldsymbol{z}, \boldsymbol{x} \mid \boldsymbol{\theta}^{new})\Big]}_{\text{Energy}}.$$

From the definition of the Kullback-Leibler divergence we get that

$$\log P(\boldsymbol{x} \mid \boldsymbol{\theta}^{new}) = ELBO(\boldsymbol{\theta}^{new} \mid \boldsymbol{\theta}^{old}) + KL\Big(P(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{\theta}^{old}) \mid P(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{\theta}^{new})\Big).$$

We could use $\boldsymbol{\theta}^{old}$ in the above formula getting

$$\log P(\boldsymbol{x} \mid \boldsymbol{\theta}^{old}) = ELBO(\boldsymbol{\theta}^{old} \mid \boldsymbol{\theta}^{old}) + KL\Big(P(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{\theta}^{old}) \mid P(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{\theta}^{old})\Big) = ELBO(\boldsymbol{\theta}^{old} \mid \boldsymbol{\theta}^{old}).$$

So we can compute the difference between the log likelihood between two consecutive iterations as

$$\begin{aligned}\log P(\boldsymbol{x} \mid \boldsymbol{\theta}^{new}) - \log P(\boldsymbol{x} \mid \boldsymbol{\theta}^{old}) = {} & ELBO(\boldsymbol{\theta}^{new} \mid \boldsymbol{\theta}^{old}) - ELBO(\boldsymbol{\theta}^{old} \mid \boldsymbol{\theta}^{old}) \\ & + KL\Big(P(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{\theta}^{old}) \mid P(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{\theta}^{new})\Big).\end{aligned}$$

Where we know the last term is always positive. About the difference of bounds, the M-step ensures the new parameter makes the lower bound higher or equal to the current one, so that difference is also positive:

$$\log P(\boldsymbol{x} \mid \boldsymbol{\theta}^{new}) - \log P(\boldsymbol{x} \mid \boldsymbol{\theta}^{old}) \geq 0.$$

□

9.2 EXAMPLE: BINOMIAL MIXTURE

In this section we are using a coin-flipping experiment as an example to show limitations of classical maximum likelihood inference due to the presence of hidden variables. The EM algorithm is then used to surpass these limitations. The example consists in a *mixture distribution* based on Do & Batzoglou (2008).

The experiment consist of randomly choosing (with equal probability) one of a pair of coins A and B with unknown biases, θ_A and θ_B. Let 1 denote *heads* and 0 denote *tails*. The selected

coin is tossed M times, repeating this whole procedure N times. The experiment is then governed by two parameters:

$$P(A = 1) = \theta_A \quad \text{and} \quad P(B = 1) = \theta_B.$$

If the coins were not chosen with equal probability but using an unknown parameter, it could be also learn by the algorithm.

Maximum likelihood training attempts to infer the value of $\boldsymbol{\theta} = (\theta_A, \theta_B)$ that maximizes the likelihood. Given that the maximum likelihood distribution is the empirical distribution, the optimal parameter values are:

$$\theta_A^{ML} = \frac{\text{Heads of coin } A}{\text{Total flips of coin } A} \quad \text{and} \quad \theta_B^{ML} = \frac{\text{Heads of coin } B}{\text{Total flips of coin } B}.$$

Consider now that the identity of the coin that is being flipped is unknown. Let X be the random variable modeling the number of heads in M flips and Z the coin being flipped. In this situation, using the empirical distribution is not possible as the identity of the coin is unknown. In this kind of situations, the EM algorithm performs maximum likelihood training while dealing with the hidden variable.

The conditional $x_n \mid z_n, \boldsymbol{\theta}$ follows a Binomial distribution:

$$x_n \mid z_n, \boldsymbol{\theta} \sim B(M, \theta_{z_n}) \implies P(x_n \mid z_n, \boldsymbol{\theta}) = \binom{M}{x_n} \theta_{z_n}^{x_n} (1 - \theta_{z_n})^{M - x_n} \; \forall n = 1, \ldots, N,$$

and the probability of choosing a coin is $P(z_n = A) = P(z_n = B) = 0.5 \; \forall n = 1, \ldots, N$.

The lower bound is:

$$\sum_{n=1}^{N} \log P(x_n \mid \boldsymbol{\theta}) \geq \sum_{n=1}^{N} -\mathbb{E}_{Q(z_n)} \Big[\log Q(z_n) \Big] + \mathbb{E}_{Q(z_n)} \Big[\log P(z_n, x_n \mid \boldsymbol{\theta}) \Big].$$

The **E-step** consists on setting (given a fixed $\boldsymbol{\theta}$, which is not considered a random variable),

$$\begin{aligned} Q(z_n) = P(z_n \mid x_n, \boldsymbol{\theta}) &= \frac{P(x_n \mid z_n, \boldsymbol{\theta}) P(z_n)}{P(x_n)} \\ &= \frac{0.5 \times \theta_{z_n}^{x_n} (1 - \theta_{z_n})^{M - x_n}}{\int_{z_n} P(x_n, z_n \mid \boldsymbol{\theta})} \\ &= \frac{P(z_n) \theta_{z_n}^{x_n} (1 - \theta_{z_n})^{M - x_n}}{\int_{z_n, \boldsymbol{\theta}} P(x_n \mid z_n, \boldsymbol{\theta}) P(z_n)} \\ &= \frac{\theta_{z_n}^{x_n} (1 - \theta_{z_n})^{M - x_n}}{\theta_A^{x_n} (1 - \theta_A)^{M - x_n} + \theta_B^{x_n} (1 - \theta_B)^{M - x_n}} \quad \forall n = 1, \ldots, N. \end{aligned}$$

Conversely, the **M-step** consists on setting

$$\begin{aligned} \boldsymbol{\theta}^{new} &= \arg\max_{\theta} \sum_{n=1}^{N} \mathbb{E}_{Q(z_n)} \Big[\log P(x_n, z_n \mid \boldsymbol{\theta}) \Big] \\ &= \arg\max_{\theta} \sum_{n=1}^{N} \mathbb{E}_{Q(z_n)} \Big[\log P(x_n \mid z_n, \boldsymbol{\theta}) P(z_n) \Big] \\ &= \arg\max_{\theta} \sum_{n=1}^{N} \mathbb{E}_{Q(z_n)} \Big[\log P(x_n \mid z_n, \boldsymbol{\theta}) \Big] \end{aligned}$$

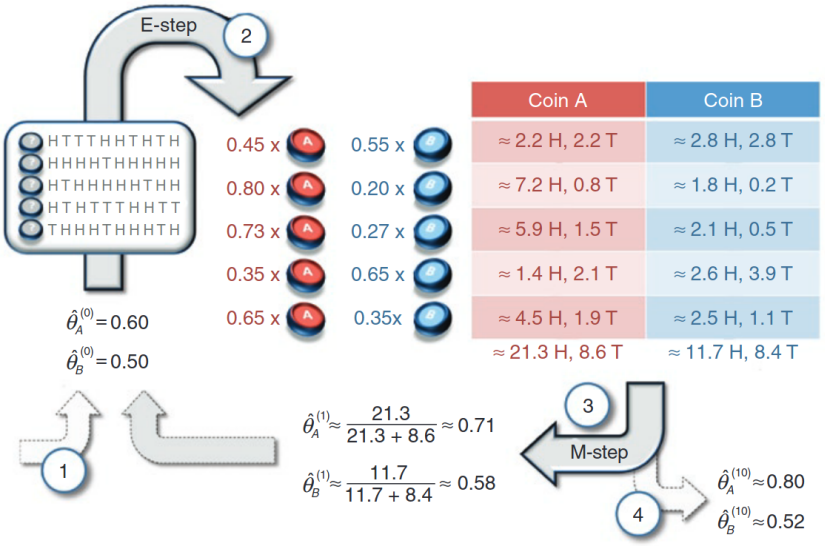


Figure 4: EM algorithm step in Mixture Do & Batzoglou (2008)

where in the last equality we used that $P(z_n) = 0.5$ is constant. The expectation is written as

$$\mathbb{E}_{Q(z_n)}\Big[\log P(x_n \mid z_n, \boldsymbol{\theta})\Big] = Q(z_n = A)\Big(x_n \log \theta_A + (M - x_n)\log(1 - \theta_A)\Big) + Q(z_n = B)\Big(x_n \log \theta_B + (M - x_n)\log(1 - \theta_B)\Big).$$

As θ_A and θ_B are separated in each term, the optimum can be found separately, the term affected by θ_A is

$$\sum_{n=1}^{N} Q(z_n = A)\Big(x_n \log \theta_A + (M - x_n)\log(1 - \theta_A)\Big),$$

deriving and setting to 0 we get the following maximum,

$$\sum_{n=1}^{N} Q(z_n = A)\Big(\frac{x_n}{\theta_A} - \frac{M - x_n}{1 - \theta_A}\Big) = 0 \iff \theta_A = \sum_{n=1}^{N} Q(z_n = A)\frac{x_n}{M}.$$

It is a maximum and not a minimum because

$$\log P(x_n \mid z_n, (0,0)) = \log P(x_n \mid z_n, (1,1)) = \log 1 = 0,$$

and the updates of A and B are independent. The same argument is valid for θ_B.

We are now using the figure 4 to set the example in a numerical context using the given data points in the diagram.

1. We are considering a prior $\boldsymbol{\theta} = (0.6, 0.5)$, a set of $N = 5$ samples and $M = 10$ flips per sample.
2. In the **E-step**, we calculate each $Q(z_n)$.

$$Q(z_1 = A) = \frac{0.6^5 0.4^5}{0.6^5 0.4^5 + 0.5^5 0.5^5} \approx 0.45 \implies Q(z_1 = B) \approx 0.55,$$

$$\vdots$$

$$Q(z_5 = A) \approx 0.65 \implies Q(z_5 = B) \approx 0.35.$$

3. The **M-step** is then:

$$\theta_A = \sum_{n=1}^{5} Q(z_n)\frac{x_n}{10} \approx \frac{21.3}{21.3+8.6} \approx 0.71,$$
$$\theta_B \approx 0.58.$$

4. Step 4 represents a possible solution after 10 iterations.

9.3 PARTIAL STEPS

Making a partial **M-step** consist on not using the optimal parameter for the energy term, but using one with just higher energy. Finding this values can be easier than finding the optimal one and convergence still follows as the only requirement to make the likelihood increase was to increase the lower bound.

When studying the increase on the likelihood, we supposed that the optimal **E-step** was being used. It cannot be guarantee that a partial step would increase the likelihood in this case, even though it does increase the lower bound.

Another important factor is, that the EM algorithm assumes that the energy term is possible to calculate, which may not be. As an approach to solve this situation, we can set a class of distributions $\mathcal{Q}$, where the term can be computed, and minimize the Kullback-Leibler divergence between $P(\boldsymbol{z} \mid \boldsymbol{x}, \theta)$ and a distribution $Q \in \mathcal{Q}$, so we pick a distribution such that

$$Q = \arg\min_{Q \in \mathcal{Q}} KL\Big(Q(\boldsymbol{z}) \;\Big|\; P(\boldsymbol{z} \mid \boldsymbol{x}, \theta)\Big).$$

An extreme case is to choose $\mathcal{Q}$ as delta functions, where the energy term is now a constant, and the optimal setting is

$$Q(z_n) = \delta(z_n, z_n^{opt}) \quad \text{and} \quad z_n^{opt} = \arg\max_z P(z, x_n \mid \theta).$$

This is called *Viterbi training* and does not guarantee that the log likelihood is being increased on each iteration.

Viterbi training justifies its approach reasoning that, given the sufficient amount of data, the likelihood function $P(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta})$ will be peaked around its optimum value. This suggest that the initial value of the parameter plays a major role in Viterbi training, compared to the classical EM. This technique is commonly used for training hidden Markov models (HMMs) (Barber (2007) Section 23.2).

10

MEAN-FIELD VARIATIONAL INFERENCE

10.1 THE MEAN-FIELD VARIATIONAL FAMILY

The *mean-field variational family* $\mathcal{Q}$ is defined as the family of distributions where the variables are mutually independent, i.e, any $Q \in \mathcal{Q}$ verifies

$$Q(\boldsymbol{z}) = \prod_{m=1}^{M} Q_m(z_m),$$

where $\boldsymbol{Z} = \{Z_1, \ldots, Z_M\}$ is the considered set of variables. The mean-field family is commonly used to model the family of distributions over the latent variables in our optimization problem. Notice that each *factor* Q_m can be different and this family does not depend on the observed data.

The mean-field family can capture any marginal of the latent variables but not correlation between them, as it assumes they are independent. For example, consider a two dimensional Gaussian distribution where a high percentage of the density is inside the blue ellipse shown in Figure 5. Any mean-field approximation would factorize as a product of two Gaussian distributions, condensing its density in a circle as shown in purple.

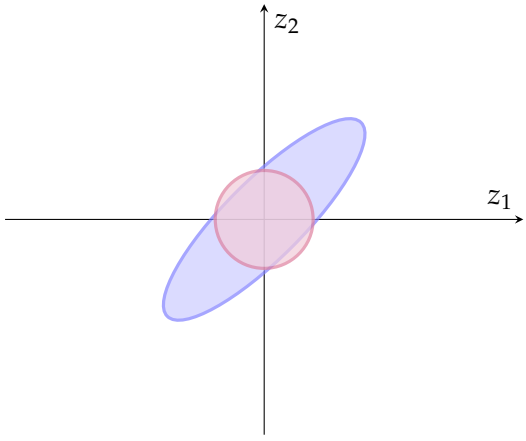


Figure 5: Mean-field family distribution (purple) approximating a Gaussian distribution (blue).

Notice the parametric form of each factor Q_m is not specified and the appropriate configuration depends on the variable. For example, a continuous variable might have a Gaussian factor and a categorical variable have a categorical factor.

Algorithm 2: Coordinate Ascent Variational Inference

Data: A distribution $P(\boldsymbol{x}, \boldsymbol{z})$ with a dataset $\boldsymbol{x}$.
Result: A distribution of the mean-field family $Q(\boldsymbol{z}) = \prod_{m=1}^{M} Q_m(z_m)$
Initialize $Q(\boldsymbol{z})$;
while *Convergence stop criteria* **do**
  for $m \in 1, \ldots, M$ **do**
    $\boldsymbol{z}_{\setminus m} = (z_1, \ldots, z_{m-1}, z_{m+1}, \ldots, z_M)$;
    Set $Q_m(z_m) \propto \exp \mathbb{E}_{Q_{\setminus m}}\Big[\log P(z_m \mid \boldsymbol{z}_{\setminus m}, \boldsymbol{x})\Big]$;
  end
  Compute $ELBO(Q)$; // Used for convergence criteria.
end
return Q;

10.2 CAVI ALGORITHM

In this section, we describe a widely used algorithm to solve the optimization problem we discussed in the previous section using the mean-field family. It is *coordinate ascent variational inference* or *CAVI* (also known as *Variational Bayes*) and its procedure is to iteratively optimize each factor of the mean-field family distribution, while fixing the others. With this, the ELBO reaches a local optimum.

The *coordinate ascent* approach is similar to the *gradient ascent* in that both attempt to reach a local optima following a iterative procedure and maximizing each step. They differ in that the former updates each *coordinate* (variable in this context) whereas the latter updates all of them using the direction of the gradient.

CAVI might be seen as a generalization of the EM algorithm, where the algorithm is extended to a full Bayesian approach, considering the parameters as latent variables. This will be reviewed in Section 10.3.

Let $\boldsymbol{x}$ be the given observations of the observed variables. CAVI iterates fixing all hidden variable but one at a time and maximizing its contribution to the ELBO. Consider the m^{th} variable Z_m, denoting by $\setminus m$ full set of indexes without the m^{th}, then $\boldsymbol{Z}_{\setminus m}$ is the full set of variables without the focused one. Let the factors $Q_n, n \neq m$ be fixed.

The contribution of Z_m to the ELBO is (summarizing other factors in the constant term):

$$
\begin{aligned}
ELBO(Q) &= \mathbb{E}_Q\Big[\log P(\boldsymbol{x}, \boldsymbol{z})\Big] - \mathbb{E}_Q\Big[\log Q(\boldsymbol{z})\Big] \\
&\overset{1}{=} \mathbb{E}_{Q_m}\Big[\mathbb{E}_{Q_{\setminus m}}\Big[\log P(\boldsymbol{x}, \boldsymbol{z})\Big]\Big] - \mathbb{E}_{Q_m}\Big[\log Q_m(z_m)\Big] + \text{ const.} \\
&\overset{2}{=} \mathbb{E}_{Q_m}\Big[\mathbb{E}_{Q_{\setminus m}}\Big[\log P(z_m \mid \boldsymbol{z}_{\setminus m}, \boldsymbol{x}) + \log P(\boldsymbol{z}_{\setminus m}, \boldsymbol{x})\Big]\Big] - \mathbb{E}_{Q_m}\Big[\log Q_m(z_m)\Big] + \text{ const.} \\
&\overset{3}{=} \mathbb{E}_{Q_m}\Big[\mathbb{E}_{Q_{\setminus m}}\Big[\log P(z_m \mid \boldsymbol{z}_{\setminus m}, \boldsymbol{x})\Big]\Big] - \mathbb{E}_{Q_m}\Big[\log Q_m(z_m)\Big] + \text{ const.} \\
&\overset{4}{=} -KL\Big(Q_m(z_m) \mid \exp \mathbb{E}_{Q_{\setminus m}}\Big[\log P(z_m \mid \boldsymbol{z}_{\setminus m}, \boldsymbol{x})\Big]\Big) + \text{ const..}
\end{aligned}
$$

1. The expectations in the ELBO formula are separated. The logarithm factorizes as $\log Q(z) = \sum_{m=1}^{M} \log Q_m(z_m)$. The constant term comes from $\mathbb{E}_{Q_{\setminus m}}\Big[\log Q_{\setminus m}(z_{\setminus m})\Big]$.
2. P is separated as $P(\boldsymbol{z}, \boldsymbol{x}) = P(z_m \mid \boldsymbol{z}_{\setminus m}, \boldsymbol{x}) P(\boldsymbol{z}_{\setminus m}, \boldsymbol{x}) \implies \log P(\boldsymbol{z}, \boldsymbol{x}) = \log P(z_m \mid \boldsymbol{z}_{\setminus m}, \boldsymbol{x}) + \log P(\boldsymbol{z}_{\setminus m}, \boldsymbol{x})$.
3. $\mathbb{E}_{Q_m}\Big[\mathbb{E}_{Q_{\setminus m}}\Big[\log P(\boldsymbol{z}_{\setminus m}, \boldsymbol{x})\Big]\Big] = \mathbb{E}_{Q_{\setminus m}}\Big[\log P(\boldsymbol{z}_{\setminus m}, \boldsymbol{x})\Big]$ is constant.
4. Applied Kullback-Leibler definition.

Maximizing the ELBO is equivalent to minimize the given Kullback-Leibler divergence and this divergence is zero when Q_m^{new} is:

$$Q_m^{new}(z_m) \propto \exp \mathbb{E}_{Q_{\setminus m}}\Big[\log P(z_m \mid \boldsymbol{z}_{\setminus m}, \boldsymbol{x})\Big].$$

Notice that the proportionality restriction is enough to fully determine the distribution as its integral is normalized. Equivalently, the distribution is proportional to

$$Q_m^{new}(z_m) \propto \exp \mathbb{E}_{Q_{\setminus m}}\Big[\log P(z_m, \boldsymbol{z}_{\setminus m}, \boldsymbol{x})\Big]. \tag{1}$$

As the ELBO is generally a non-convex function and the CAVI algorithm converges to a local optimum, the initialization values of the algorithm play an important role on its performance. The convergence criteria is usually a threshold for the ELBO or a fixed ammount of iterations.

10.3 CAVI AS AN EM GENERALIZATION

As CAVI considers parameters as hidden variables, we need to specify a variational distribution for them. We are choosing the distribution that summarizes the information in the optimal point, let $\boldsymbol{\theta}_{opt}$ be the optimal value of $\boldsymbol{\theta}$:

$$Q(\boldsymbol{\theta}) = \delta(\boldsymbol{\theta} - \boldsymbol{\theta}_{opt}).$$

The variational distribution factorizes as

$$Q(\boldsymbol{z}, \boldsymbol{\theta}) = Q(\boldsymbol{z}) Q(\boldsymbol{\theta}).$$

The lower bound takes the form

$$\begin{aligned}
\log P(\boldsymbol{x} \mid \boldsymbol{\theta}) &\geq \mathbb{E}_{Q(\boldsymbol{z},\boldsymbol{\theta})}\Big[\log P(\boldsymbol{x}, \boldsymbol{z}, \theta)\Big] - \mathbb{E}_{Q(\boldsymbol{z},\boldsymbol{\theta})}\Big[\log Q(\boldsymbol{z}, \boldsymbol{\theta})\Big] \\
&= \mathbb{E}_{Q(\boldsymbol{z})}\Big[\mathbb{E}_{Q(\boldsymbol{\theta})}\Big[\log P(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\theta})\Big]\Big] - \mathbb{E}_{Q(\boldsymbol{z})}\Big[\log Q(\boldsymbol{z})\Big] - \mathbb{E}_{Q(\boldsymbol{\theta})}\Big[\log Q(\boldsymbol{\theta})\Big] \\
&= \mathbb{E}_{Q(\boldsymbol{z})}\Big[\log P(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\theta}_{opt})\Big] - \mathbb{E}_{Q(\boldsymbol{z})}\Big[\log Q(\boldsymbol{z})\Big] + \text{ const.}
\end{aligned}$$

The CAVI update can be seen as an iterative two step process. Firstly, given a fixed $Q(\boldsymbol{z})$, as the distribution class of $Q(\boldsymbol{\theta})$ is fixed, optimizing it is equivalent to find the optimal parameter $\boldsymbol{\theta}_{opt}$.

$$\begin{aligned}
\boldsymbol{\theta}_{opt} &= \arg\max_{\boldsymbol{\theta}} \Big(\mathbb{E}_{Q(\boldsymbol{z})}\Big[\log P(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\theta})\Big]\Big) \\
&= \arg\max_{\boldsymbol{\theta}} \Big(\mathbb{E}_{Q(\boldsymbol{z})}\Big[\log P(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta}) P(\boldsymbol{\theta})\Big]\Big) \\
&= \arg\max_{\boldsymbol{\theta}} \Big(\mathbb{E}_{Q(\boldsymbol{z})}\Big[\log P(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta})\Big] + \log P(\boldsymbol{\theta})\Big).
\end{aligned}$$

If we take a flat prior, this term is equivalent to the M-step. Secondly, given a fixed parameter $\boldsymbol{\theta}$, the update is

$$Q(\boldsymbol{z}) \propto \exp \mathbb{E}_{Q(\boldsymbol{\theta})}\Big[\log P(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{\theta})\Big] = P(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{\theta}_{opt}),$$

which is the E-step of the EM algorithm.

11

EXPONENTIAL FAMILY

The exponential family (Koopman (1936)) is a parametric set of probability distributions of a certain form. This form is chosen based on some useful algebraic properties and generality, Appendix A shows some examples of common distributions in the exponential family.

Let X be a random variable and $\boldsymbol{\theta}$ a set of parameters. A family of distributions is said to belong the exponential family if its probability distribution has the form

$$P(x \mid \boldsymbol{\theta}) = h(x) \exp\Big(\sum_{i=1}^{S} \eta_i(\boldsymbol{\theta}) T_i(x) - \psi(\boldsymbol{\theta}) \Big),$$

where $h(x)$, $T_i(x)$, $\eta_i(\boldsymbol{\theta})$ and $\psi(\boldsymbol{\theta})$ are known functions such that h is called a *base measure*, $\eta_i(\boldsymbol{\theta})$ are called the *distribution parameters*, $T_i(x)$ the *test statistics* and ψ is the *log normalizer* as it ensures logarithmic normalization due to

$$\begin{aligned} 1 &= \int_x h(x) \exp\Big(\sum_{i=1}^{S} \eta_i(\boldsymbol{\theta}) T_i(x) - \psi(\boldsymbol{\theta}) \Big) \\ &= \int_x e^{-\psi(\boldsymbol{\theta})} h(x) \exp\Big(\sum_{i=1}^{S} \eta_i(\boldsymbol{\theta}) T_i(x) \Big) \\ &= e^{-\psi(\boldsymbol{\theta})} \int_x h(x) \exp\Big(\sum_{i=1}^{S} \eta_i(\boldsymbol{\theta}) T_i(x) \Big), \end{aligned}$$

so ψ verifies

$$\psi(\boldsymbol{\theta}) = \log \int_x h(x) \exp\Big(\sum_{i=1}^{S} \eta_i(\boldsymbol{\theta}) T_i(x) \Big).$$

Naming $\boldsymbol{\eta}$ and $\boldsymbol{T}$ the corresponding vector functions, the parameters can always be transformed as $\boldsymbol{\theta}^{new} = \boldsymbol{\eta}(\boldsymbol{\theta})$, in which case we say the distribution into its *canonical form* (notice ψ has changed but we do not distinguish it from the previous one since it is fully determined by the other functions):

$$P(x \mid \boldsymbol{\theta}) = h(x) e^{\boldsymbol{\theta}^T \boldsymbol{T}(x) - \psi(\boldsymbol{\theta})}.$$

In this form, it is easier to see that $\boldsymbol{T}$ is the sufficient statistic for $\boldsymbol{\theta}$. This is a consequence of *Fisher–Neyman factorization theorem* which says that $\boldsymbol{T}$ *is sufficient for* $\boldsymbol{\theta}$ *if and only if the probability distribution* P *can be factored into a product such that one factor,* h*, does not depend on* $\boldsymbol{\theta}$ *and the other factor, which does depend on* $\boldsymbol{\theta}$*, depends on* x *only through* $\boldsymbol{T}$*.*

An important property of the exponential family is that they have *conjugate priors*, this is said when the posterior distribution is in the same probability distribution family as the prior distribution, they are then called *conjugate distributions*, and the prior is called a *conjugate prior* of the likelihood distribution. Appendix B shows examples of conjugate distributions and their conjugate priors. Models with conjugate priors are usually called *conditionally conjugate models*.

Proposition 6. *Let X be a random variable and $\boldsymbol{\theta}$ a set of parameters. Suppose an exponential family likelihood:*

$$P(x \mid \boldsymbol{\theta}) = h(x)e^{\boldsymbol{\theta}^T \boldsymbol{T}(x) - \psi(\boldsymbol{\theta})}.$$

and prior with hyper-parameters $\boldsymbol{\alpha}, \gamma$:

$$P(\boldsymbol{\theta} \mid \boldsymbol{\alpha}, \gamma) \propto e^{\boldsymbol{\theta}^T \boldsymbol{\alpha} - \gamma\psi(\boldsymbol{\theta})}.$$

Then, the posterior is in the same parametric family as the prior with

$$P(\boldsymbol{\theta} \mid \boldsymbol{x}, \boldsymbol{\alpha}, \gamma) = P(\boldsymbol{\theta} \mid \boldsymbol{\alpha} + \boldsymbol{T}(x), \gamma + 1).$$

Proof.

$$P(\boldsymbol{\theta} \mid \boldsymbol{x}, \boldsymbol{\alpha}, \gamma) \propto P(x \mid \boldsymbol{\theta}) P(\boldsymbol{\theta} \mid \boldsymbol{\alpha}, \gamma) \propto \exp\Big(\boldsymbol{\theta}^T [\boldsymbol{\alpha} + \boldsymbol{T}(x)] - [\gamma + 1]\psi(\boldsymbol{\theta})\Big).$$

□

11.1 LATENT VARIABLE AND CONDITIONALLY CONJUGATE MODELS

One important case of exponential family are *latent variable models* or *LVMs*. In this models, the following assumptions are made:

1. There set of i.i.d random variables $\boldsymbol{X} = (X_1, \ldots, X_N)$ and a set of observations $\boldsymbol{x} = (x_1, \ldots, x_N)$.

2. There are both global and local latent variables. The global ones are denoted by $\boldsymbol{\theta}$ whereas the locals are denoted by $\boldsymbol{Z} = (Z_1, \ldots, Z_N)$. We refer to *global hidden variables* when they affect the whole distribution and *local hidden variables* when they affect only to a subset, in this case each Z_n affects only X_n. Given this, the joint probability is

$$P(\boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{z}) = P(\boldsymbol{\theta}) \prod_{n=1}^{N} P(z_n, x_n \mid \boldsymbol{\theta}).$$

3. The n^{th} observation x_n and the n^{th} local hidden variable z_n are conditionally independent, given the global variables $\boldsymbol{\theta}$, of all other observations and local hidden variables,

$$P(x_n, z_n \mid \boldsymbol{\theta}, \boldsymbol{x}_{\setminus n}, \boldsymbol{z}_{\setminus n}) = P(x_n, z_n \mid \boldsymbol{\theta}).$$

Remark 4. These models are widely used to discover patterns in data sets (Blei (2014)). LVMs include popular models like *Latent Dirichlet Allocation* models used to uncover the hidden topics in text corpora (Blei *et al.* (2003)), mixture of Gaussian models to discover

hidden clusters in data (Bishop (2006)) and probabilistic principal component analysis for dimensionality reduction (Tipping & Bishop (1999)). These three models are reviewed throughout this document.

One important case of LVMs and exponential family models are *conditionally conjugate models* with local and global variables (Blei *et al.* (2017)), where the following assumptions are made.

1. The prior for the global latent variable $P(\boldsymbol{\theta})$ is in the exponential family with an hyper-parameter $\alpha = [\alpha_1, \alpha_2]$, where α_1 is a vector and α_2 is a scalar, and statistics that concatenate the global latent variable $\boldsymbol{\theta}$ and its log normalizer from $P(z_n, x_n \mid \boldsymbol{\theta})$, $\psi(\boldsymbol{\theta})$[1],

$$P(\boldsymbol{\theta}) = h(\boldsymbol{\theta}) \exp\Big(\alpha^T[\boldsymbol{\theta}, -\psi(\boldsymbol{\theta})] - \psi(\alpha)\Big).$$

2. Each local term $P(z_n, x_n \mid \boldsymbol{\theta})$ is in the exponential family of the form

$$P(z_n, x_n \mid \boldsymbol{\theta}) = h(z_n, x_n) \exp\Big(\boldsymbol{\theta}^T T(z_n, x_n) - \psi(\boldsymbol{\theta})\Big).$$

3. The complete conditional of a local latent variable verifies

$$P(z_n \mid \boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{z}_{\setminus n}) = P(z_n \mid x_n, \boldsymbol{\theta})$$

   and is also in the exponential family

$$P(z_n \mid x_n, \boldsymbol{\theta}) = h(z_n) \exp\Big(\eta(\boldsymbol{\theta}, x_n)^T T(z_n) - \psi(\boldsymbol{\theta}, x_n)\Big).$$

Using Proposition 6, the posterior $P(\boldsymbol{\theta} \mid \boldsymbol{x}, \boldsymbol{z})$ is in the same family with parameter

$$\bar{\alpha} = [\alpha_1 + \sum_{n=1}^{N} T(z_i, x_i), \alpha_2 + N]^T.$$

A step by step reasoning would be:

$$\begin{aligned}
P(\boldsymbol{\theta} \mid \boldsymbol{x}, \boldsymbol{z}) = \frac{P(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta}) P(\boldsymbol{\theta})}{P(\boldsymbol{x}, \boldsymbol{z})} &\propto P(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta}) P(\boldsymbol{\theta}) = P(\boldsymbol{\theta}) \prod_{n=1}^{N} h(x_n, z_n) \exp\Big(\boldsymbol{\theta}^T T(z_n, x_n) - \psi(\boldsymbol{\theta})\Big) \\
&\propto h(\boldsymbol{\theta}) \exp\Big(\alpha^T[\boldsymbol{\theta}, -\psi(\boldsymbol{\theta})]\Big) \prod_{n=1}^{N} h(x_n, z_n) \exp\Big(\boldsymbol{\theta}^T T(z_n, x_n) - \psi(\boldsymbol{\theta})\Big) \\
&\propto h(\boldsymbol{\theta}) \exp\Big(\alpha^T[\boldsymbol{\theta}, -\psi(\boldsymbol{\theta})] + \sum_{n=1}^{N} \boldsymbol{\theta}^T T(z_n, x_n) - \psi(\boldsymbol{\theta})\Big) \\
&\propto h(\boldsymbol{\theta}) \exp\Big(\alpha_1^T \boldsymbol{\theta} - \alpha_2^T \psi(\boldsymbol{\theta}) - N\psi(\boldsymbol{\theta}) + \sum_{n=1}^{N} T(z_n, x_n)^T \boldsymbol{\theta}\Big) \\
&\propto h(\boldsymbol{\theta}) \exp\Big(\big(\alpha_1 + \sum_{n=1}^{N} T(z_n, x_n)\big)^T \boldsymbol{\theta} - (\alpha_2 + N)^T \psi(\boldsymbol{\theta})\Big).
\end{aligned}$$

1 The same symbol ψ is used for every normalizer function. The distinction must be made using their parameters.

11.2 CAVI IN CONDITIONALLY CONJUGATE MODELS

Consider the following situation where we fit a distribution $Q(\boldsymbol{z}) = \prod_{n=1}^{N} Q_n(z_n)$ in the mean-field family, using an exponential family distribution for the marginal $P(z_n \mid \boldsymbol{z}_{\setminus n}, \boldsymbol{x})$:

$$P(z_n \mid \boldsymbol{z}_{\setminus n}, \boldsymbol{x}) = h(z_n) \exp\Big(\eta_n(\boldsymbol{z}_{\setminus n}, \boldsymbol{x})^T T(z_n) - \psi(\boldsymbol{z}_{\setminus n}, \boldsymbol{x})\Big).$$

The update of the CAVI algorithm is then given by

$$\begin{aligned} Q(z_n) &\propto \exp \mathbb{E}_{Q(\boldsymbol{z}_{\setminus n})}\Big[\log P(z_n \mid \boldsymbol{z}_{\setminus n}, \boldsymbol{x})\Big] \\ &= h(z_n) \exp\Big(\mathbb{E}_{Q_{\boldsymbol{z}_{\setminus n}}}\Big[\eta_n(\boldsymbol{z}_{\setminus n}, \boldsymbol{x})\Big]^T T(z_n) - \mathbb{E}_{Q_{\boldsymbol{z}_{\setminus n}}}\Big[\psi(\boldsymbol{z}_{\setminus n}, \boldsymbol{x})\Big]\Big) \\ &\propto h(z_n) \exp\Big(\mathbb{E}_{Q_{\boldsymbol{z}_{\setminus n}}}\Big[\eta_n(\boldsymbol{z}_{\setminus n}, \boldsymbol{x})\Big]^T T(z_n)\Big) = h(z_n) e^{v_n^T T(z_n)}, \end{aligned} \tag{2}$$

where

$$v_n = \mathbb{E}_{Q_{\boldsymbol{z}_{\setminus n}}}\Big[\eta_n(\boldsymbol{z}_{\setminus n}, \boldsymbol{x})\Big].$$

Summarizing, the factor is in the exponential family with the same base measure h and updating it is equivalent to setting the distribution parameter η to the expected one of the complete conditional $\mathbb{E}\Big[\eta_n(\boldsymbol{x}, \boldsymbol{z}_{\setminus n})\Big]$. This expression facilitates deriving CAVI algorithm for many complex models.

Going back to the conditionally conjugate models, our variational distribution is in the mean-field family with $Q(\boldsymbol{\theta} \mid \lambda)$ where λ is called the *global variational parameter*, and for each local variable, the distribution is $Q(z_n \mid \gamma_n)$, where γ_n is called a *local variational parameter*:

$$Q(z_n \mid \gamma_n) \propto h(z_n) e^{\gamma_n{}^T T(z_n)},$$

$$Q(\boldsymbol{\theta} \mid \lambda) = h(\boldsymbol{\theta}) \exp\Big(\lambda^T [\boldsymbol{\theta}, -\psi(\boldsymbol{\theta})] - \psi(\lambda)\Big).$$

CAVI iteratively updates each local variational parameter and the global variational parameter.

Following the steps done in 2, the local variational parameter update is

$$\gamma_n^{new} = \mathbb{E}_{Q(\boldsymbol{\theta}|\lambda)}\Big[\eta(\boldsymbol{\theta}, x_n)\Big],$$

In this case, the local hidden variable conditional does not depend on the other local hidden variables, neither other data-points.

The global variational parameter update is calculated as

$$\begin{aligned} Q^{new}(\boldsymbol{\theta} \mid \lambda) &\propto \exp \mathbb{E}_{Q(\boldsymbol{z})}\Big[\log P(\boldsymbol{\theta} \mid \boldsymbol{z}, \boldsymbol{x})\Big] \propto h(\boldsymbol{\theta}) \exp \mathbb{E}_{Q(\boldsymbol{z})}\Big[\big[\lambda_1 + \sum_{n=1}^{N} T(x_n, z_n), \lambda_2 + N\big]^T [\boldsymbol{\theta}, \psi(\boldsymbol{\theta})]\Big] \\ &\propto h(\boldsymbol{\theta}) \exp\left(\Big[\lambda_1 + \sum_{n=1}^{N} \mathbb{E}_{Q(\boldsymbol{z})}\Big[T(x_n, z_n)\Big], \lambda_2 + N\Big]^T \Big[\boldsymbol{\theta}, -\psi(\boldsymbol{\theta})\Big]\right) \\ &= Q(\boldsymbol{\theta}, \lambda^{new}). \end{aligned}$$

Then, the variational parameter updated is

$$\lambda^{new} = \left[\lambda_1 + \sum_{n=1}^{N} \mathbb{E}_{Q(z_n|\gamma_n)}\Big[T(x_n, z_n)\Big], \lambda_2 + N\right].$$

The ELBO can be computed at each iteration up to a constant which does not depend on the variational parameters,

$$\begin{aligned} ELBO = &\left(\lambda_1 + \sum_{n=1}^{N} \mathbb{E}_{Q(z_n|\gamma_n)}\Big[T(x_n, z_n)\Big]\right)^T \mathbb{E}_{Q(\boldsymbol{\theta}|\lambda)}\Big[\boldsymbol{\theta}\Big] - (\lambda_2 + N)\mathbb{E}_{Q(\boldsymbol{\theta}|\lambda)}\Big[\psi(\boldsymbol{\theta})\Big] \\ &+ \lambda^T \mathbb{E}_{Q(\boldsymbol{\theta},\lambda)}\Big[T(\boldsymbol{\theta})\Big] - \psi(\lambda) + \sum_{n=1}^{N} \gamma_n^T \mathbb{E}_{Q(z_n,\gamma_n)}\Big[z_n\Big] - \psi(\gamma_n) + \text{ const.} \end{aligned}$$

The calculations are the following:

$$\begin{aligned} ELBO(Q(\boldsymbol{\theta}, \boldsymbol{z})) &= \mathbb{E}_{Q(\boldsymbol{\theta},\boldsymbol{z})}\Big[\log P(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{x})\Big] - \mathbb{E}_{Q(\boldsymbol{\theta},\boldsymbol{z})}\Big[\log Q(\boldsymbol{\theta}, \boldsymbol{z})\Big] \\ &= \mathbb{E}_{Q(\boldsymbol{z}|\boldsymbol{\gamma})}\Big[\mathbb{E}_{Q(\boldsymbol{\theta}|\lambda)}\Big[\log P(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{x})\Big]\Big] - \mathbb{E}_{Q(\boldsymbol{\theta},\boldsymbol{z})}\Big[\log Q(\boldsymbol{\theta}, \boldsymbol{z})\Big] \\ &= \mathbb{E}_{Q(\boldsymbol{\theta}|\lambda)}\Big[\log P(\boldsymbol{\theta})\Big] + \mathbb{E}_{Q(\boldsymbol{z}|\boldsymbol{\gamma})}\Big[\mathbb{E}_{Q(\boldsymbol{\theta}|\lambda)}\Big[\sum_{n=1}^{N} \log P(z_n, x_n \mid \boldsymbol{\theta})\Big]\Big] \\ &\quad - \mathbb{E}_{Q(\boldsymbol{\theta},\boldsymbol{z})}\Big[\log Q(\boldsymbol{\theta}, \boldsymbol{z})\Big] \end{aligned}$$

The first term is

$$\begin{aligned} \mathbb{E}_{Q(\boldsymbol{\theta}|\lambda)}\Big[\log P(\boldsymbol{\theta})\Big] &= \mathbb{E}_{Q(\boldsymbol{\theta}|\lambda)}\Big[\lambda_1 \boldsymbol{\theta} - \lambda_2 \psi(\boldsymbol{\theta}) - \psi(\lambda)\Big] \\ &= \lambda_1 \mathbb{E}_{Q(\boldsymbol{\theta}|\lambda)}\Big[\boldsymbol{\theta}\Big] - \lambda_2 \mathbb{E}_{Q(\boldsymbol{\theta}|\lambda)}\Big[\psi(\boldsymbol{\theta})\Big] - \psi(\lambda). \end{aligned}$$

The middle term is

$$\begin{aligned} &\mathbb{E}_{Q(\boldsymbol{z}|\boldsymbol{\gamma})}\Big[\mathbb{E}_{Q(\boldsymbol{\theta}|\lambda)}\Big[\log P(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{x})\Big]\Big] = \mathbb{E}_{Q(\boldsymbol{z}|\boldsymbol{\gamma})}\Big[\mathbb{E}_{Q(\boldsymbol{\theta}|\lambda)}\Big[\log P(\boldsymbol{\theta}) + \sum_{n=1}^{N} \log P(z_n, x_n \mid \boldsymbol{\theta})\Big]\Big] \\ &= \mathbb{E}_{Q(\boldsymbol{z}|\boldsymbol{\gamma})}\Big[\mathbb{E}_{Q(\boldsymbol{\theta}|\lambda)}\Big[\log P(\boldsymbol{\theta})\Big]\Big] + \mathbb{E}_{Q(\boldsymbol{z}|\boldsymbol{\gamma})}\Big[\mathbb{E}_{Q(\boldsymbol{\theta}|\lambda)}\Big[\sum_{n=1}^{N} \log P(z_n, x_n \mid \boldsymbol{\theta})\Big]\Big] \\ &= \left(\lambda_1 + \sum_{n=1}^{N} \mathbb{E}_{Q(z_n|\gamma_n)}\Big[T(x_n, z_n)\Big]\right)^T \mathbb{E}_{Q(\boldsymbol{\theta}|\lambda)}\Big[\boldsymbol{\theta}\Big] - (\lambda_2 + N)\mathbb{E}_{Q(\boldsymbol{\theta}|\lambda)}\Big[\eta(\boldsymbol{\theta})\Big]. \end{aligned}$$

The last term is

$$\begin{aligned} \mathbb{E}_{Q(\boldsymbol{\theta},\boldsymbol{z})}\Big[\log Q(\boldsymbol{\theta}, \boldsymbol{z})\Big] &= \mathbb{E}_{Q(\boldsymbol{\theta})}\Big[\log Q(\boldsymbol{\theta})\Big] + \mathbb{E}_{Q(\boldsymbol{z})}\Big[\sum_{n=1}^{N} \log Q(z_n)\Big] \\ &= \mathbb{E}_{Q(\boldsymbol{\theta})}\Big[\lambda^T T(\boldsymbol{\theta}) - \psi(\lambda)\Big] + \sum_{n=1}^{N} \mathbb{E}_{Q(z_n)}\Big[\gamma_n^T z_n - \psi(\gamma_n)\Big] \\ &= \lambda^T \mathbb{E}_{Q(\boldsymbol{\theta},\lambda)}\Big[T(\boldsymbol{\theta})\Big] - \psi(\lambda) + \sum_{n=1}^{N} \gamma_n^T \mathbb{E}_{Q(z_n,\gamma_n)}\Big[z_n\Big] - \psi(\gamma_n). \end{aligned}$$

12

EXAMPLE: GAUSSIAN MIXTURE

12.1 MODEL STATEMENT

A *Gaussian mixture model* is a latent variable model where the data is assumed to be generated from a finite number of Gaussian distributions with unknown parameters.

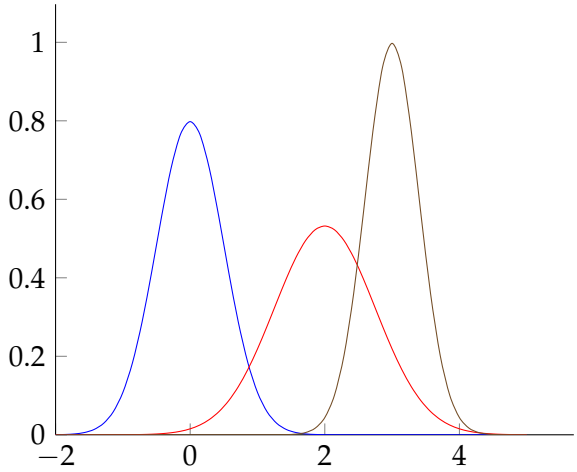


Figure 6: One-dimensional Gaussian mixture with three clusters. Those are $\mathcal{N}(0, 0.5)$, $\mathcal{N}(2, 0.75)$ and $\mathcal{N}(3, 0.4)$.

The following elements are being considered (using Bishop (2006) Chapter 10.2 notation):

- K mixture components and N observations.
- A set of i.i.d $\mathbb{R}^D$-valued random variables $\boldsymbol{X} = (X_1, \ldots, X_N)$ and a corresponding set of observations $\boldsymbol{x} = (x_1, \ldots, x_n)$.
- The cluster assignment i.i.d latent variables $\boldsymbol{Z} = (Z_1, \ldots, Z_N)$, where each z_n is a binary vector where only one component is non zero, indicating the cluster to which x_n belongs.
- We choose a Dirichlet distribution over the mixing coefficients $\boldsymbol{\pi}$

$$\boldsymbol{\pi} \sim \text{Symmetric-Dirichlet}(\alpha_0) \implies P(\boldsymbol{\pi}) \propto \prod_{k=1}^{K} \pi_k^{\alpha_0 - 1}.$$

The hyper-parameter α_0 is the effective prior of each mixture component. Then $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_K)$ are the mixture weights, i.e, prior probability of a particular component k.

$$P(\boldsymbol{z} \mid \boldsymbol{\pi}) = \prod_{n=1}^{N} \prod_{k=1}^{K} \pi_k^{z_{n,k}} \implies Z_n \mid \boldsymbol{\pi} \sim \text{Categorical}(\boldsymbol{\pi}).$$

- The distribution mean and precision of each Gaussian distribution, $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_K)$ and $\boldsymbol{\Lambda} = (\Lambda_1, \ldots, \Lambda_K)$,

$$P(\boldsymbol{x} \mid \boldsymbol{z}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = \prod_{n=1}^{N} \prod_{k=1}^{K} \mathcal{N}(\mu_k, \Lambda_k^{-1})^{z_{n,k}}.$$ [1]

- The prior governing $\boldsymbol{\mu}$ and $\boldsymbol{\Lambda}$ is an independent Gaussian-Wishart distribution with hyper parameters m_0, β_0, W_0 and ν_0:

$$P(\boldsymbol{\mu}, \boldsymbol{\Lambda}) = P(\boldsymbol{\mu} \mid \boldsymbol{\Lambda}) P(\boldsymbol{\Lambda}) = \prod_{k=1}^{K} P(\mu_k \mid \Lambda_k) P(\Lambda_k) = \prod_{k=1}^{K} \mathcal{N}(m_0, (\beta_0 \Lambda_k)^{-1}) \mathcal{W}(W_0, \nu_0).$$

The joint probability factorizes as

$$P(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = P(\boldsymbol{x} \mid \boldsymbol{z}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) P(\boldsymbol{z} \mid \boldsymbol{\pi}) P(\boldsymbol{\pi}) P(\boldsymbol{\mu} \mid \boldsymbol{\Lambda}) P(\boldsymbol{\Lambda}).$$

The model can be represented with a Bayesian network as done in Chapter 21. The needed steps to give the explicit CAVI update are now summarized following Bishop (2006).

12.2 VARIATIONAL DISTRIBUTION AND CAVI UPDATE

We consider a variational distribution which factorizes as

$$Q(\boldsymbol{z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = Q(\boldsymbol{z}) Q(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda}).$$

Let us consider the update for $Q(\boldsymbol{z})$, the update might be written as the following (using the explicit update given in 1):

$$Q^{new}(\boldsymbol{z}) \propto \exp \mathbb{E}_{Q(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda})} \Big[\log P(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) \Big].$$

Which implies:

$$\begin{aligned}
\log Q^{new}(\boldsymbol{z}) &= \mathbb{E}_{Q(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda})} \Big[\log P(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) \Big] + \text{const.} \\
&= \mathbb{E}_{Q(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda})} \Big[\log \Big(P(\boldsymbol{x} \mid \boldsymbol{z}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) P(\boldsymbol{z} \mid \boldsymbol{\pi}) P(\boldsymbol{\pi}) P(\boldsymbol{\mu} \mid \boldsymbol{\Lambda}) P(\boldsymbol{\Lambda}) \Big) \Big] + \text{const.} \\
&= \mathbb{E}_{Q(\boldsymbol{\pi})} \Big[\log P(\boldsymbol{z} \mid \boldsymbol{\pi}) \Big] + \mathbb{E}_{Q(\boldsymbol{\mu}, \boldsymbol{\Lambda})} \Big[\log P(\boldsymbol{x} \mid \boldsymbol{z}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) \Big] + \text{const.}
\end{aligned}$$

The first term is,

$$\begin{aligned}
\mathbb{E}_{Q(\boldsymbol{\pi})} \Big[\log P(\boldsymbol{z} \mid \boldsymbol{\pi}) \Big] &= \mathbb{E}_{Q(\boldsymbol{\pi})} \Big[\log \prod_{n=1}^{N} \prod_{k=1}^{K} \pi_k^{z_{n,k}} \Big] = \mathbb{E}_{Q(\boldsymbol{\pi})} \Big[\sum_{n=1}^{N} \sum_{k=1}^{K} z_{n,k} \log \pi_k \Big] \\
&= \sum_{n=1}^{N} \sum_{k=1}^{K} z_{n,k} \mathbb{E}_{Q(\boldsymbol{\pi})} \Big[\log \pi_k \Big].
\end{aligned}$$

1 This notation symbolizes the density function of the given Gaussian distribution.

Finally, the last term is,

$$\begin{aligned}
\mathbb{E}_{Q(\boldsymbol{\mu},\boldsymbol{\Lambda})}\Big[\log P(\boldsymbol{x} \mid \boldsymbol{z},\boldsymbol{\mu},\boldsymbol{\Lambda})\Big] &= \\
&= \mathbb{E}_{Q(\boldsymbol{\mu},\boldsymbol{\Lambda})}\Big[\log \prod_{n=1}^{N}\prod_{k=1}^{K} \frac{1}{\sqrt{(2\pi)^D|\Lambda_k^{-1}|}} \exp\Big(-\frac{1}{2}(x_n-\mu_k)^T\Lambda_k(x_n-\mu_k)\Big)\Big] \\
&= \mathbb{E}_{Q(\boldsymbol{\mu},\boldsymbol{\Lambda})}\Big[\sum_{n=1}^{N}\sum_{k=1}^{K} -\frac{D}{2}\log(2\pi) + \frac{1}{2}\log|\Lambda_k| + \Big(-\frac{1}{2}(x_n-\mu_k)^T\Lambda_k(x_n-\mu_k)\Big)\Big] \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K} -\frac{D}{2}\log(2\pi) + \frac{1}{2}\mathbb{E}_{Q(\boldsymbol{\Lambda})}\Big[\log|\Lambda_k|\Big] - \frac{1}{2}\mathbb{E}_{Q(\mu_k,\Lambda_k)}\Big[(x_n-\mu_k)^T\Lambda_k(x_n-\mu_k)\Big].
\end{aligned}$$

Given this, the new variational distribution $Q^{new}(\boldsymbol{z})$ follows a Categorical distribution with parameters $r_{n,k}$, which are the normalization of $\rho_{n,k}$, defined as,

$$\log\rho_{n,k} = \mathbb{E}_{Q(\boldsymbol{\pi})}\Big[\pi_k\Big] + \frac{1}{2}\mathbb{E}_{\boldsymbol{\Lambda}}\Big[\log|\Lambda_k|\Big] - \frac{D}{2}\log(2\pi) - \frac{1}{2}\mathbb{E}_{Q(\mu_k,\Lambda_k)}\Big[(x_n-\mu_k)^T\Lambda_k(x_n-\mu_k)\Big],$$

$$r_{n,k} = \frac{\rho_{n,k}}{\sum_{k=1}^{K}\rho_{n,k}}.$$

Notice that, as $\rho_{n,k}$ is defined as the exponential of a real value, it is positive. Therefore, $r_{n,k}$ are non-negative and sum one as required. The distribution is then

$$Q^{new}(\boldsymbol{z}) = \prod_{n=1}^{N}\prod_{k=1}^{K} r_{n,k}^{z_{n,k}}.$$

Using a similar reasoning to the factor $Q(\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda})$, the update is:

$$\begin{aligned}
\log Q^{new}(\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda}) &= \mathbb{E}_{Q(\boldsymbol{z})}\Big[\log P(\boldsymbol{x},\boldsymbol{z},\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda})\Big] + \text{const.} \\
&= \mathbb{E}_{Q(\boldsymbol{z})}\Big[\log\big(P(\boldsymbol{x} \mid \boldsymbol{z},\boldsymbol{\mu},\boldsymbol{\Lambda})P(\boldsymbol{z} \mid \boldsymbol{\pi})P(\boldsymbol{\pi})P(\boldsymbol{\mu} \mid \boldsymbol{\Lambda})P(\boldsymbol{\Lambda})\big)\Big] + \text{const.} \\
&= \log P(\boldsymbol{\pi}) + \log P(\boldsymbol{\mu} \mid \boldsymbol{\Lambda}) + \log P(\boldsymbol{\Lambda}) + \mathbb{E}_{Q(\boldsymbol{z})}\Big[\log P(\boldsymbol{z} \mid \boldsymbol{\pi})\Big] \\
&\quad + \mathbb{E}_{Q(\boldsymbol{z})}\Big[\log P(\boldsymbol{x} \mid \boldsymbol{z},\boldsymbol{\mu},\boldsymbol{\Lambda})\Big] + \text{const.} \\
&= \log P(\boldsymbol{\pi}) + \sum_{k=1}^{K}\log P(\mu_k,\Lambda_k) + \mathbb{E}_{Q(\boldsymbol{z})}\Big[\log P(\boldsymbol{z} \mid \boldsymbol{\pi})\Big] \\
&\quad + \sum_{n=1}^{N}\sum_{k=1}^{K} r_{n,k}\log\mathcal{N}(\mu_k,\Lambda_k^{-1}) + \text{const.}
\end{aligned}$$

Where $\mathbb{E}_{\boldsymbol{z}}\Big[z_{n,k}\Big] = r_{n,k}$ is used. The right term can be decomposed in terms involving only $\boldsymbol{\pi}$ and each μ_k,Λ_k. Therefore, the variational distribution factorizes as

$$Q(\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Lambda}) = Q(\boldsymbol{\pi})\prod_{k=1}^{K} Q(\mu_k,\Lambda_k).$$

Where

$$\log Q^{new}(\boldsymbol{\pi}) = \log P(\boldsymbol{\pi}) + \mathbb{E}_{Q(\boldsymbol{z})}\Big[\log P(\boldsymbol{z} \mid \boldsymbol{\pi})\Big] + \text{const.}$$

$$\log Q^{new}(\mu_k,\Lambda_k) = \log P(\mu_k,\Lambda_k) + \sum_{n=1}^{N} r_{n,k}\log\mathcal{N}(\mu_k,\Lambda_k^{-1}) + \text{const.}$$

Inspecting the term that depends on $\boldsymbol{\pi}$ one get that,

$$\log Q^{new}(\boldsymbol{\pi}) = (\alpha_0 - 1)\sum_{k=1}^{K}\log \pi_k + \sum_{n=1}^{N}\sum_{k=1}^{K} r_{n,k}\log \pi_k + \text{const.}$$

Naming $N_k = \sum_{n=1}^{N} r_{n,k}\ \forall k = 1,\dots,K$, the update is

$$Q^{new}(\boldsymbol{\pi}) \propto \prod_{k=1}^{K} \pi_k^{\alpha_0 - 1 + N_k}.$$

Defining $\boldsymbol{\alpha}$, with components:

$$\alpha_k = \alpha_0 + N_k, \quad \forall k = 1,\dots,K,$$

the new variational distribution follows a Dirichlet distribution

$$Q^{new}(\boldsymbol{\pi}) \equiv \text{Dirichlet}(\boldsymbol{\alpha}).$$

Lastly, the variational distribution $Q(\mu_k, \Lambda_k)$ does not factorize but the update is given by

$$\begin{aligned}
\log Q^{new}(\mu_k,\Lambda_k) &= \log P(\mu_k,\Lambda_k) + \sum_{n=1}^{N} r_{n,k}\log \mathcal{N}(\mu_k,\Lambda_k^{-1}) + \text{const.} \\
&= \log \mathcal{N}(m_0,(\beta_0\Lambda_k)^{-1}) + \log \mathcal{W}(W_0,\nu_0) + \sum_{n=1}^{N} r_{n,k}\log \mathcal{N}(\mu_k,\Lambda_k^{-1}) + \text{const.} \\
&= \frac{\beta_0}{2}(\mu_k - m_0)^T\Lambda_k(\mu_k - m_0) + \frac{1}{2}\log|\Lambda_k| - \frac{1}{2}Tr(\Lambda_k W_0^{-1}) \\
&\quad + \frac{\nu_0 - D - 1}{2}\log|\Lambda_k| - \frac{1}{2}\sum_{n=1}^{N} r_{n,k}(x_n - \mu_k)^T\Lambda_k(x_n - \mu_k) \\
&\quad + \frac{1}{2}N_k\log|\Lambda_k| + \text{const.}
\end{aligned}$$

Using Appendix B, the above formula is similar the posterior distribution of a Gaussian likelihood and Gaussian-Wishart prior. Using the procedure used in the appendix, the posterior follows a Gaussian-Wishart distribution with parameters:

$$Q^{new}(\mu_k,\Lambda_k) \equiv \mathcal{N}(m_k,(\beta_k\Lambda_k)^{-1})\mathcal{W}(W_k,\nu_k).$$

Where,

$$\begin{aligned}
\beta_k &= \beta_0 + N_k, \\
m_k &= \frac{1}{\beta_k}\Big(\beta_0 m_0 + \sum_{n=1}^{N} r_{n,k}x_n\Big), \\
\bar{x}_k &= \frac{1}{N_k}\sum_{n=1}^{N} r_{n,k}x_n, \\
S_k &= \frac{1}{N_k}\sum_{n=1}^{N} r_{n,k}(x_n - \bar{x}_k)(x_n - \bar{x}_k)^T, \\
W_k^{-1} &= W_0^{-1} + N_k S_k + \frac{\beta_0 N_k}{\beta_0 + N_k}(\bar{x}_k - m_0)(\bar{x} - m_0)^T, \\
\nu_k &= \nu_0 + N_k.
\end{aligned}$$

Part IV

GRAPHICAL MODELS

A *graphical model* is a statistical model for which the dependence structure between random variables is expressed by a graph.

Commonly, they provide a graph-based representation for encoding a multi-dimensional distribution representing a set of independences that hold in the specific distribution. The most commonly used are *Bayesian networks* and *Markov random fields*, these two differ in the set of independences they are able to encode and the factorization of the distribution they include.

13

INTRODUCTION

Probabilistic graphical models are diagrammatic representations of probability distributions that represent the dependence/independence relation between the considered variables. Their use presents the following advantages (Bishop (2006)):

1. They allow a simple visualization of probabilistic models.
2. Insights into the model's properties can be obtained form the graph.
3. Complex computations, required in inference task, are simplified using the graph structure.

In probabilistic graphical models, each node represents a random variable and links represents relations between them. The graph encodes how the joint probability distribution of the considered variables factorizes. The different classes of graphical models differ in how they represent these relations and how the distribution is factorized.

We begin discussing *Bayesian networks*, also known as *directed graphical models*, where links between variables are indicated by a directed arrow. The other major class of graphical models are *Markov random fields* or *undirected graphical models* in which links have no directional meaning. The former are useful to express casual relations between the variables whereas the latter express soft constraints between the variables.

Consider a set of variables $\boldsymbol{X} = (X_1, \ldots, X_N)$, the possible ways these variables can interact is extremely large, for binary variables, the joint distribution table would take $O(2^N)$ space, which is unpractical when the amount of variables scales. When dealing with this such large distributions it is a common practice to factorize the joint probability in a graphical model, reducing the needed resources to deal with the inference problem.

14

BAYESIAN NETWORKS

Given a set of variables $\boldsymbol{X} = (X_1, \ldots, X_N)$, *Bayesian networks* might be defined either as a probability distribution of a certain form or a DAG whose nodes represent these variables and links an independence constraint. Both ideas are present in the following definition.

Definition 36. A *belief network or Bayesian network* is a pair (G, P) formed by a DAG G and joint probability distribution P such that, there is a correspondence between variables and nodes such as:

$$P(x_1, \ldots, x_N) = \prod_{n=1}^{N} P(x_n \mid pa(x_n)).$$

Remark 5. A Bayesian network might be given as a distribution from which the DAG can be constructed or a DAG which represents the distribution. For example in Figure 7, given the DAG one could easily define the joint distribution and conversely.

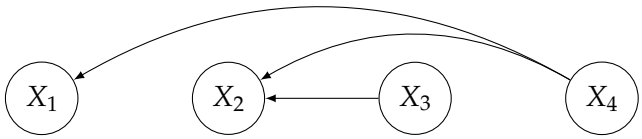


Figure 7: Bayesian Network factorizing $P(x_1, x_2, x_3, x_4) = P(x_1 \mid x_4)P(x_2 \mid x_3, x_4)P(x_3)P(x_4)$.

Any probability distribution can be written as a Bayesian network, even though it may end up being a fully-connected "cascade"[1] DAG, which means that each variable X_n is a parent of any X_m with $m > n$. This is because any distribution satisfies:

$$P(x_1, \ldots, x_N) = P(x_1) \prod_{n=2}^{N} P(x_n \mid x_1, \ldots, x_{n-1})$$

Bayesian Networks are good for encoding *conditional independence* over the variables, but are not for encoding dependence. For example, with the following network

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

represented as $x \to y$ in a DAG, it may appear to encode dependence between both variables but the conditional $P(y \mid x)$ could happen to equal $P(y)$, giving $P(x, y) = P(x)P(y)$.

1 This term comes from the visual structure of the graph.

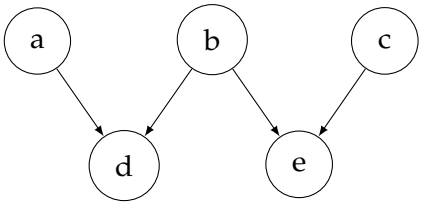


Figure 8: D-separation example

How could we check if two variables are conditionally independent given a Bayesian network? For example in Figure 3, $X_1 \perp\!\!\!\perp X_2 \mid X_4$ as

$$\begin{aligned} P(x_2 \mid x_4) &= \frac{1}{P(x_4)} \int_{x_1,x_3} P(x_1, x_2, x_3, x_4) = \frac{1}{P(x_4)} \int_{x_1,x_3} P(x_1 \mid x_4) P(x_2 \mid x_3, x_4) P(x_3) P(x_4) \\ &= \int_{x_3} P(x_2 \mid x_3, x_4) P(x_3), \end{aligned}$$

$$\begin{aligned} P(x_1, x_2 \mid x_4) &= \frac{1}{P(x_4)} \int_{x_3} P(x_1, x_2, x_3, x_4) = \frac{1}{P(x_4)} \int_{x_3} P(x_1 \mid x_4) P(x_2 \mid x_3, x_4) P(x_3) P(x_4) \\ &= P(x_1 \mid x_4) \int_{x_3} P(x_2 \mid x_3, x_4) P(x_3) = P(x_1 \mid x_4) P(x_2 \mid x_4). \end{aligned}$$

14.1 D-SEPARATION AND D-CONNECTION

Now we are going to define two central concepts to determine conditional independence in any Bayesian network, these are *d-connection* and *d-separation*.

Definition 37. Let G be a DAG where $\boldsymbol{X}, \boldsymbol{Y}$ and $\boldsymbol{Z}$ are disjoint sets of vertices. We say that $\boldsymbol{X}$ and $\boldsymbol{Y}$ are *d-connected* by $\boldsymbol{Z}$ if and only if there exists an undirected path U from any vertex in $\boldsymbol{X}$ to any vertex in $\boldsymbol{Y}$ such that:

- For any collider C, itself or any it's descendants is in $\boldsymbol{Z}$.
- No non-collider on U is on $\boldsymbol{Z}$.

That is, there exists a path where $\boldsymbol{Z}$ contains all of its colliders and their descendants, and no other node from the path.

Definition 38. Let G be a DAG where $\boldsymbol{X}, \boldsymbol{Y}$ and $\boldsymbol{Z}$ are disjoint sets of vertices. $\boldsymbol{X}$ and $\boldsymbol{Y}$ are *d-separated* by $\boldsymbol{Z}$ if and only if they are not d-connected by $\boldsymbol{Z}$ in G. That is, for any undirected path from $\boldsymbol{X}$ to $\boldsymbol{Y}$, either there is a collider or a descendant of it not in $\boldsymbol{Z}$ or there is a non-collider in $\boldsymbol{Z}$.

For example, in Figure 8, d d-separates a and c (e is a collider in the path that is not in $\{d\}$), and $\{d, e\}$ d-connect them.

Theorem 2 (Spirtes *et al.* (2000) Theorem 3.3)**.** *Let G be a DAG where $\boldsymbol{X}, \boldsymbol{Y}$ and $\boldsymbol{Z}$ are disjoint sets of vertices. $\boldsymbol{X}$ and $\boldsymbol{Y}$ are d-separated by $\boldsymbol{Z}$ if and only if they are independent conditional on $\boldsymbol{Z}$ in all probability distributions that G may represent.*

The Bayes Ball algorithm (Shachter (2013)) provides a linear time complexity algorithm that computes conditional independent using this theorem.

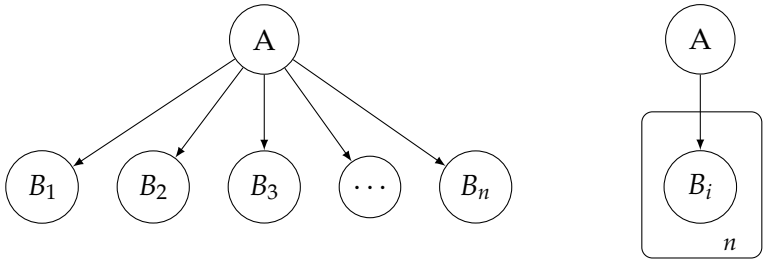


Figure 9: Plate notation example. Standard notation on the left and plate on the right.

In cases where the Bayesian networks contains i.i.d nodes that are essentially the same but repeated a number of times, the *plate notation* is commonly used to represent this nodes in a compacted manner (Figure 9).

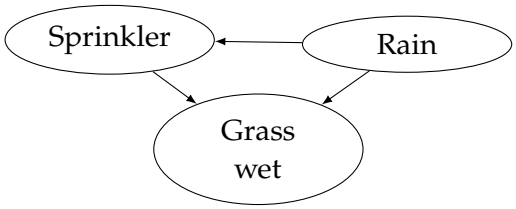
Example 4. In this example we are modeling three discrete random variables: Sprinkler (S), Rain (R) and Grass wet (G).

The joint probability function is:

$$P(s, r, g) = P(s|r)P(g|s, r)P(r)$$

The following DAG illustrates the Bayesian network among with the probability tables we are using.

	Sprinkler	
Rain	T	F
F	0.4	0.6
T	0.01	0.99



Rain	
T	F
0.2	0.8

		Grass wet	
Sprinkler	Rain	T	F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

This model can answer questions about the presence of a cause given the presence of an effect. For example, What is the probability that it has being raining given the grass is wet?

$$P(R = T|G = T) = \frac{P(G = T, R = T)}{P(G = T)} = \frac{\sum_s P(G = T, R = T, s)}{\sum_{r,s} P(G = T, r, s)}.$$

Algorithm 3: Sum-product Variable Elimination

Data: Set of factors ϕ in which the joint distribution factorizes, set of variables to be eliminates $\boldsymbol{X}$ and an ordering on $\boldsymbol{X} = (X_1, \dots, X_N)$ such that the set is ordered $X_i < X_j \iff i < j$.

Result: A factor ψ^* with the variables eliminated.

for $n = 1, \dots, N$; // Eliminate variable X_n

do

Let ϕ' be the set of factors that do depend on X_n;

$\psi = \prod_{\phi \in \phi'} \phi$; // Define the product of factors ϕ

$\tau = \sum_{X_n} \phi$; // Define the sum-product factor τ

$\phi = \phi' \cup \tau$; // Append the new factor

end

$\phi^* = \prod_{\phi \in \phi} \phi$;

return ϕ^*;

Using the expression of the joint probability among with the tables we can compute every term. For example:

$$\begin{aligned} P(G = T, R = T, S = T) &= P(S = T|R = T)P(G = T|R = T, S = T)P(R = T) \\ &= 0.01 * 0.99 * 0.2 = 0.00198. \end{aligned}$$

14.2 EXACT INFERENCE IN BAYESIAN NETWORKS

In this section, exact inference in graphical models is reviewed. The Bayesian network structure makes it possible to perform inference efficiently in complex models.

Let $\boldsymbol{X}$ and $\boldsymbol{Y}$ be two set of variables with $\boldsymbol{X} \cap \boldsymbol{Y} = \emptyset$ and $\boldsymbol{X} \cup \boldsymbol{Y} \cup \boldsymbol{W}$ being the full set of variables in the network. The objective is to efficiently compute conditional distributions $\boldsymbol{X} \mid \boldsymbol{y}$, that is, computing $P(\boldsymbol{x} \mid \boldsymbol{y})\ \forall \boldsymbol{x}$.

This conditional can be computed as

$$P(\boldsymbol{x} \mid \boldsymbol{y}) = \frac{P(\boldsymbol{x}, \boldsymbol{y})}{P(\boldsymbol{y})} \quad \text{where} \quad P(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\boldsymbol{w}} P(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}) \quad \text{and} \quad P(\boldsymbol{y}) = \sum_{\boldsymbol{x}} P(\boldsymbol{x}, \boldsymbol{y}).$$

One may notice that, the numerator $P(\boldsymbol{x}, \boldsymbol{y})$ can be used to compute the denonimator. Given this, the procedure would be to compute $P(\boldsymbol{x}, \boldsymbol{y})\ \forall \boldsymbol{x}$ using $P(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w})$, which are the entries of the joint distribution, and then using this values to efficiently calculate $P(\boldsymbol{y})$.

The inference problem in graphical models turns to be $\mathcal{NP}-$hard, which, assuming the Exponential Time Hypothesis (Impagliazzo & Paturi (1999)), means that the worst case requires exponential time. Much worse is the fact that *approximate inference* is still $\mathcal{NP}-$hard (Koller & Friedman (2009) Section 9.1).

Variable elimination algorithm (3) uses *dynamic programming* techniques to perform inference in a reasonable time. Let $\boldsymbol{X} = (X_1, \dots, X_N)$ be a set of variables, $\{\boldsymbol{Y}, \boldsymbol{Z}\}$ a partition of it, and $P(\boldsymbol{x})$ factorize as the product of the Bayesian network factors $\phi = \{P(x_n \mid pa(x_n))\}_{n=1,\dots,N}$,

then, for any ordering of $\boldsymbol{Z}$, the variable elimination algorithm on ϕ and $\boldsymbol{Z}$ returns a factor $\phi^*(\boldsymbol{y})$ such that

$$\phi^*(\boldsymbol{y}) = \sum_{\boldsymbol{z}} \prod_{\phi \in \phi} \phi = \sum_{\boldsymbol{z}} P(\boldsymbol{y}, \boldsymbol{z}) = P(\boldsymbol{y}).$$

Computing a conditional probability $P(\boldsymbol{y}_1 \mid \boldsymbol{y}_2)$ where $\boldsymbol{Y}_1, \boldsymbol{Y}_2 \subset \boldsymbol{Y}$, reduces to using the algorithm to compute $P(\boldsymbol{y}_1, \boldsymbol{y}_2)$ and then use it to compute $P(\boldsymbol{y}_2)$.

Remark 6. This algorithm can be applied to Markov random fields, having to normalize the factors to compute the real distribution.

Example 5. Let A, B, C and D be the four variables of the Bayesian network (Figure 10). Suppose the goal is to compute $P(d)$ for each d, possible value of D. Given that the joint distribution factorizes as

$$P(a, b, c, d) = P(a)P(b \mid a)P(c \mid b)P(d \mid c),$$

the calculations would be

$$P(d) = \sum_{a,b,c} P(a, b, c, d) = \sum_{c} P(d \mid c) \sum_{b} P(c \mid b) \sum_{a} P(a)P(b \mid a),$$

where terms appear several times, for example, if the variables were binary, the term $P(a = 1)P(b = 1 \mid a = 1)$ would appear twice, one for each value of C. Which means that it would appear four times (two when $d = 0$ and two when $d = 1$). Storing these terms during the calculations would decrease the total number of needed operations.

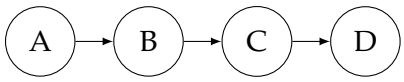


Figure 10: 4-node Bayesian network for variable elimination introduction.

The procedure followed by the algorithm is to define variables τ that one by one eliminate each variable form the above formula. Firstly, the variable A is eliminated, let $\psi_{A,B}(a, b) = P(b \mid a)P(a)$ and $\tau_B(b) = \sum_a \psi_{A,B}(a, b)$, then

$$\tau_B(b) = \sum_{a} P(a)P(b \mid a).$$

Given $\tau_B(b)$,

$$P(d) = \sum_{b,c} P(a, b, c, d) = \sum_{c} P(d \mid c) \sum_{b} P(c \mid b)\tau_B(b).$$

Using the same reasoning with B and D,

$$\psi_{B,C}(b, c) = \sum_{b} \sum_{a} P(c \mid b)P(b \mid a)P(a) = \tau_B(b)P(c \mid b),$$
$$\tau_C(c) = \sum_{b} \psi_{B,C}(b, c).$$

Where $P(d)$ is finally computed as

$$P(d) = \sum_{c} \tau_C(c)P(d \mid c)$$

for each value d of D.

15

MARKOV RANDOM FIELDS

Whereas Bayesian networks are represented by an acyclic directed graph, *Markov random fields* are undirected graphs that may be cyclic. Thus, this kind of graphical models are able to represent certain dependencies which Bayesian networks cannot, like cyclic ones. *Markov random fields* are commonly used to model tasks in computer vision and image processing (Li (2009)).

Definition 39. Given an undirected graph $G = (V, E)$, a set of random variables $\boldsymbol{X} = (X_1, \ldots, X_N)$ *form a Markov random field over* G if they satisfy the, so called, Markov properties (Barber (2007)):

- **Pairwise Markov property**. Any two non-adjacent variables are conditionally independent given all other variables:

$$X_n \perp\!\!\!\perp X_m \mid \boldsymbol{X}_{\setminus n,m} \quad \forall n, m \in \{1, \ldots, N\} \quad n \neq m.$$

- **Local Markov property**. Any variable is conditionally independent over all other variables given its neighbors:

$$X_n \perp\!\!\!\perp \boldsymbol{X}_{\setminus ne(X_n)} \mid \boldsymbol{X}_{ne(X_n)} \quad \forall n \in \{1, \ldots, N\}.$$

- **Global Markov property**. Any two subsets of variables are conditionally independent given a separating subset (any path from one set to the other passes trough this one). Let A and B be two subset of indexes and S a separating subset:

$$\boldsymbol{X}_A \perp\!\!\!\perp \boldsymbol{X}_B \mid \boldsymbol{X}_S.$$

The Global Markov property is stronger than the Local Markov property, which is stronger than the Pairwise one. However, these properties are equivalent for positive distributions ($P(x) > 0$), this is Theorem 4.4 in Koller & Friedman (2009).

As these Markov properties can be difficult to establish, a commonly used class of Markov random fields are those who factorize as product of potentials over the graph's cliques.

Definition 40. A *potential* ϕ is a non-negative function. It is worth to mention that a probability distribution is a special case of a potential.

Let $\boldsymbol{X}$ be a set of random variables, G an undirected graph whose nodes are $\boldsymbol{X}$ and $\boldsymbol{X}_c, c \in \{1, \ldots, C\}$ be the maximal cliques of G. This class of Markov random fields is characterized by the fact that the joint probability distribution P can be factorized as:

$$P(x_1, \ldots, x_n) = \frac{1}{Z} \prod_{c=1}^{C} \phi_c(\boldsymbol{X}_c).$$

where Z is a constant that ensures normalization. Figure 11 shown an example in this class of Markov random fields.

Markov random fields factorize if any of the following conditions is fulfilled:

- The distribution is positive, this is shown in Hammersley–Clifford theorem (Grimmett (1973)).
- The graph is *chordal*, which means that any cycle of four of more vertices has a chord, an edge that is not part of the cycle but connects two of its vertices.

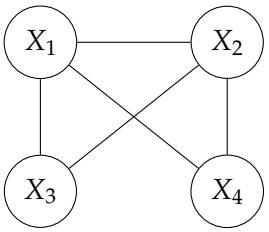


Figure 11: Markov Network $P(x_1, x_2, x_3, x_4) = \phi(x_1, x_2, x_3)\phi(x_2, x_3, x_4)/Z$

Part V

BAYESIAN NETWORKS LEARNING

Some inference methods get optimized when applied among with a graphical model. In this part we focus in how methods we have already reviewed use the Bayesian network structure to get simplified and optimized.

The *PC algorithm* is also reviewed as a procedure to generate a Bayesian network from a given dataset.

16

STRUCTURE LEARNING

The Bayesian network structure is not always given and has to be learned from the data, to achieve this, there are some issues that need to be considered. These are:

- The number of Bayesian networks is exponential over the number of variables so a brute force algorithm would not be viable.
- Testing dependencies requires a large amount of data, therefore, a threshold needs to be established to measure when a dependence is significant.
- The presence of hidden variables might not be learned from the data.

16.1 PC ALGORITHM

The *PC algorithm* (Spirtes *et al.* (2000) Chapter 5.4.2) learns the skeleton structure given a complete graph $G = (V, E)$ (constructed from the considered set of variables) and orients its edges using variable independence in the empirical distribution. The following theorem provides the motivation for the algorithm's procedure.

Theorem 3 (Spirtes *et al.* (2000) Theorem 3.4)**.** *A distribution P is faithful to a directed acyclic graph $G = (V, E)$ if and only if*

- *two vertices X and Y are adjacent if and only if are dependent conditionally on every set of vertices that does not contain either of them, i.e,*

$$X \in ne(Y) \iff X \not\perp\!\!\!\perp Y \mid S \quad \forall S \subset V \setminus \{X, Y\},$$

  and
- *for any three vertices such that $X \in ne(Y)$, $Y \in ne(Z)$ and $X \notin ne(Z)$, $X \to Y \leftarrow Z$ is in G if and only if $X \not\perp\!\!\!\perp Z \mid S \cup \{Y\}$ $\forall S \subset V \setminus \{X, Z\}$.*

The *PC algorithm* is split in two parts, firstly, the skeleton structure is learned, that is, given the complete graph, as many edges as possible are removed. This part (Algorithm 4) iterates over subsets of neighborhoods, from smaller to bigger ones. It chooses a linked pair of variables $(X, Y) \in E$ and a subset $S_{XY} \subset ne(X)$, verifying $Y \notin S_{XY}$. If $X \perp\!\!\!\perp Y \mid S_{XY}$, then the link is removed and S_{XY} is stored as their separation set.

The followed reasoning is: as $X \perp\!\!\!\perp Y \mid S_{XY}$ and $S_{XY} \subset V \setminus \{X, Y\}$, the first part of the theorem ensures that $X \notin ne(Y)$.

Algorithm 4: PC Algorithm for skeleton learning

Data: Complete undirected graph G, with vertices V
Result: G with removed links
$i = 0$;
while *all nodes have* $\leq i$ *neighbors* **do**
  for $X \in V$ **do**
    for $Y \in ne(x)$ **do**
      if $\exists S \subset ne(X) \backslash Y$ *such that* $\#S = i$ *and* $X \perp\!\!\!\perp Y \mid S$ **then**
        Remove $X - Y$ from G;
        $S_{XY} = S$;
      end
    end
  end
  $i = i + 1$;
end

This procedure results in the skeleton of the Bayesian network, no more edges will be removed or added. The directed graph may be constructed following two rules:

1. For any undirected link $X - Y - Z$, if $Y \notin S_{XZ}$ then set $X \to Y \leftarrow Z$ (we are creating a collider for that path).
2. The rest of links may oriented arbitrarily not creating cycles or colliders.

The reasoning behind is the d-separation Theorem 2, if $Y \notin S_{XZ}$ then $X \not\perp\!\!\!\perp Z \mid Y$. Otherwise $(X \perp\!\!\!\perp Z \mid Y)$ $\{Y\}$ would be the d-separation set $S_{XZ} = \{Y\}$ as they are checked from smaller to bigger ones, which we know is not the case. Therefore, using Theorem 2, Y must d-connect X and Y, to achieve this, Y must be set as a collider of the path. This is because, using the d-connection definition, the only known undirected path is $U = X - Y - Z$ and no non-collider on U must belong to $\{Y\}$.

Conversely, if $Y \in S_{XZ}$ and $X \perp\!\!\!\perp Z \mid S_{XZ}$ then S_{XZ} should d-separate them, that is, using any configuration that does not create a collider in S_{XZ}.

16.2 INDEPENDENCE LEARNING

The PC algorithms assumes there exists a procedure of testing conditional independence of variables, that is, given three variables X, Y and Z, measure $X \perp\!\!\!\perp Y \mid Z$. One approach is to measure the empirical *conditional mutual information* of the variables.

Definition 41. Given two random variables X, Y, we define their *mutual information* as the Kullback-Leibler divergence of their joint distribution and the product of their marginals,

$$MI(X;Y) = KL\Big(P_{X,Y} \mid P_X P_Y\Big).$$

Definition 42. Given three random variables X, Y, Z we define the *conditional mutual information* of X and Y over Z as

$$MI(X;Y \mid Z) = \mathbb{E}_Z\Big[KL\Big(P_{X,Y|Z} \mid P_{X|Z}P_{Y|Z}\Big)\Big].$$

Where $MI(X;Y \mid Z) \geq 0$ and

$$MI(X;Y \mid Z) = 0 \iff P_{X,Y|Z} = P_{X|Z}P_{Y|Z} \iff X \perp\!\!\!\perp Y \mid Z.$$

These values might be estimated using their empirical distributions, however, this *empirical* mutual information will be typically greater than 0 even when $X \perp\!\!\!\perp Y \mid Z$. Thus, a threshold must be established, This is commonly done by using a statistical test, taking into consideration that the empirical mutual information follows a chi-squared distribution under independence (Kullback (1997)).

A Bayesian approach would consist on comparing the model likelihood under independence and dependence hypothesis, let $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \{(x_n, y_n, z_n)\}_{n=1,\dots,N}$ be the known data of each variable:

$$P_{indep}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \int_{\boldsymbol{\theta}} P(\boldsymbol{\theta}) \prod_{n=1}^{N} P(x_n \mid z_n, \boldsymbol{\theta}) P(y_n \mid z_n, \boldsymbol{\theta}) P(z_n \mid \boldsymbol{\theta}),$$

$$P_{dep}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \int_{\boldsymbol{\theta}} P(\boldsymbol{\theta}) \prod_{n=1}^{N} P(x_n, y_n, z_n \mid \boldsymbol{\theta}).$$

Which means checking which assumptions is most probable to have generated the data.

17

MAXIMUM LIKELIHOOD TRAINING

Consider a set of i.i.d variables $\boldsymbol{X} = (X_1, \ldots, X_N)$, when reviewing maximum likelihood training, we concluded that, in case $P(x_1, \ldots, x_N \mid \theta)$ is unconstrained, the maximum likelihood distribution corresponds to the empirical distribution.

However, Bayesian networks are constraint by:

$$P(x_1, \ldots, x_N \mid \theta) = \prod_{n=1}^{N} P(x_n \mid pa(x_n), \theta).$$

This restriction requires to re-study the Kullback-Leibler divergence between the empirical distribution $Q(x_1, \ldots, x_N)$ and $P(x_1, \ldots, x_N \mid \theta)$ in order to extract the maximum likelihood value. The divergence might be written as,

$$\begin{aligned} KL\Big(Q \mid P\Big) &= -\mathbb{E}_Q\Big[\sum_{n=1}^{N} \log P(x_n \mid pa(x_n), \theta)\Big] + \mathbb{E}_Q\Big[\sum_{n=1}^{N} \log Q(x_n \mid pa(x_n))\Big] \\ &= -\sum_{n=1}^{N} \mathbb{E}_Q\Big[\log P(x_n \mid pa(x_n), \theta)\Big] + \sum_{n=1}^{N} \mathbb{E}_Q\Big[\log Q(x_n \mid pa(x_n))\Big]. \end{aligned}$$

Proposition 1 might be used over each term of the summation, which says that, as each term of the summation depends only on $(x_n, pa(x_n))$, the expectation distribution can be marginalized over them, resulting:

$$\begin{aligned} KL\Big(Q \mid P\Big) &= \sum_{n=1}^{N} \mathbb{E}_{Q(x_n, pa(x_n))}\Big[\log Q(x_n \mid pa(x_n))\Big] - \mathbb{E}_{Q(x_n, pa(x_n))}\Big[\log P(x_n \mid pa(x_n), \theta)\Big] \\ &= \sum_{n=1}^{N} \mathbb{E}_{Q(x_n, pa(x_n))}\Big[KL\Big(Q(x_n \mid pa(x_n)) \mid P(x_n \mid pa(x_n), \theta)\Big)\Big]. \end{aligned}$$

Thus, the optimal setting is

$$P(x_n \mid pa(x_n), \theta) = Q(x_n \mid pa(x_n)).$$

18

BAYESIAN TRAINING

In this section Bayesian inference is used to solve an inference problem governed by a Bayesian network, the problem statement is the following:

A disease D and two habits A and B are being studied. Consider the following i.i.d variables $\{A_1, \ldots, A_N\}$, $\{B_1, \ldots, B_N\}$ and $\{D_1, \ldots, D_N\}$ governed by the parameters θ_A, θ_B and $\boldsymbol{\theta}_D$ as shown in figure 12. Let $N = 7$ be the number of observations of the variables as shown in Table 1 and $\boldsymbol{x} = \{(a_n, b_n, d_n),\ n = 1, \ldots, N\}$ the set of observations.

All the variables are binary satisfying

$$P(A_n = 1 \mid \theta_A) = \theta_A, \quad P(B_n = 1 \mid \theta_B) = \theta_B \quad \forall n = 1, \ldots, N,$$

$$P(D_n = 1 \mid A_n = 0, B_n = 1, \boldsymbol{\theta}_D) = \theta_1, \quad \forall n = 1, \ldots, N,$$

$$\boldsymbol{\theta}_D = (\theta_0, \theta_1, \theta_2, \theta_3).$$

Where a binary to decimal transformation between the states of A and B and the sub-index of θ is being used.

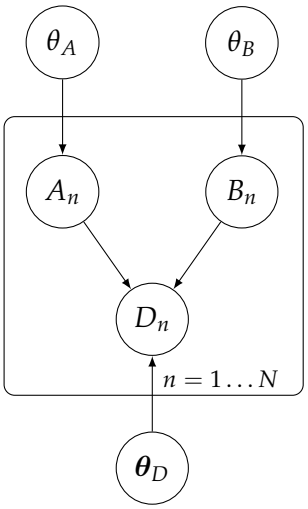


Figure 12: Bayesian parameter model for the relation between A, B and D.

A	B	D
1	1	1
1	0	0
0	1	1
0	1	0
1	1	1
0	0	0
1	0	1

Table 1: Set of observations, where 1 means true and 0 means false.

The graph gives the following joint probability distribution the variables:

$$P(a_n, b_n, d_n, \theta_A, \theta_B, \theta_D) = P(d_n \mid a_n, b_n, \theta_D)P(a_n \mid \theta_A)P(b_n \mid \theta_A).$$

A prior distribution must be specified and since dealing with multidimensional continuous distributions is computationally problematic, it is usual to use uni-variate distributions.

18.1 GLOBAL AND LOCAL PARAMETER INDEPENDENCE

A convenient assumption is that the prior factorizes, this is usually called *global parameter independence*:

$$P(\theta_A, \theta_B, \boldsymbol{\theta}_D) = P(\theta_A)P(\theta_B)P(\boldsymbol{\theta}_D).$$

Under this assumption, the joint probability factorizes as

$$P(\boldsymbol{x}, \theta_A, \theta_B, \boldsymbol{\theta}_D) = P(\theta_A)P(\theta_B)P(\boldsymbol{\theta}_D)\prod_n P(a_n \mid \theta_A)P(b_n \mid \theta_B)P(d_n \mid a_n, b_n, \boldsymbol{\theta}_D).$$

The posterior distribution is then

$$\begin{aligned} P(\theta_A, \theta_B, \boldsymbol{\theta}_D \mid \boldsymbol{x}) &= \frac{P(\boldsymbol{x} \mid \theta_a, \theta_B, \boldsymbol{\theta}_D)P(\theta_A, \theta_B, \boldsymbol{\theta}_D)}{P(\boldsymbol{x})} = \frac{P(\boldsymbol{x} \mid \theta_A, \theta_B, \boldsymbol{\theta}_D)P(\theta_A)P(\theta_B)P(\boldsymbol{\theta}_D)}{P(\boldsymbol{x})} \\ &= \frac{1}{P(\boldsymbol{x})}P(\theta_A)P(\theta_B)P(\boldsymbol{\theta}_D)\prod_{n=1}^{N} P(a_n \mid \theta_A)P(b_n \mid \theta_B)P(d_n \mid a_n, b_n, \boldsymbol{\theta}_D) \\ &= P(\theta_A \mid \boldsymbol{x}_A)P(\theta_B \mid \boldsymbol{x}_B)P(\boldsymbol{\theta}_D \mid \boldsymbol{x}). \end{aligned}$$

Where $\boldsymbol{x}_A$ is the subset of observations related to the variable A. If we further assume that $P(\boldsymbol{\theta}_D)$ factorizes as $P(\boldsymbol{\theta}_D) = P(\theta_0)P(\theta_1)P(\theta_2)P(\theta_3)$, which is called *local parameter independence*, $P(\boldsymbol{\theta}_D \mid \boldsymbol{x})$ factorizes as

$$P(\boldsymbol{\theta}_D \mid \boldsymbol{x}) = P(\theta_0 \mid \boldsymbol{x})P(\theta_1 \mid \boldsymbol{x})P(\theta_2 \mid \boldsymbol{x})P(\theta_3 \mid \boldsymbol{x}).$$

18.2 LEARNING BINARY VARIABLES

The simplest cases to continue are $P(\theta_A \mid \boldsymbol{x}_A)$ and $P(\theta_B \mid \boldsymbol{x}_B)$ since they require only a uni-variate prior distribution $P(\theta_A)$ or $P(\theta_B)$. The procedure is shown using θ_A and it is analogous when using θ_B.

The posterior is

$$P(\theta_A \mid \boldsymbol{x}_A) = \frac{1}{P(\boldsymbol{x}_A)}P(\theta_A)\theta_A^{\#(A=1)}(1-\theta_A)^{\#(A=0)}.$$

The most convenient choice for the prior is a Beta distribution as conjugacy will hold:

$$\theta_A \sim \text{Beta}(\alpha_A, \beta_A) \implies P(\theta_A) = \frac{1}{B(\alpha_A, \beta_A)}\theta_A^{\alpha_A - 1}(1-\theta_A)^{\beta_A - 1}.$$

Therefore, it follows that

$$\theta_A \mid \boldsymbol{x}_A \sim \text{Beta}(\alpha_A + \#(A=1), \beta_A + \#(A=0)).$$

The predictive marginal is then

$$\begin{aligned} P(A=1 \mid \boldsymbol{x}_A) &= \frac{P(A=1,\boldsymbol{x}_A)}{P(\boldsymbol{x}_A)} = \int_{\theta_A} \frac{P(A=1,\boldsymbol{x}_A,\theta_A)}{P(\boldsymbol{x}_A)} = \int_{\theta_A} \frac{P(A=1 \mid \boldsymbol{x}_A,\theta_A)P(\boldsymbol{x}_A,\theta_A)}{P(\boldsymbol{x}_A)} \\ &= \int_{\theta_A} \frac{P(A=1 \mid \boldsymbol{x}_A,\theta_A)P(\theta_A \mid \boldsymbol{x}_A)P(\boldsymbol{x}_A)}{P(\boldsymbol{x}_A)} \\ &= \int_{\theta_A} P(\theta_A \mid \boldsymbol{x}_A)\theta_A = \mathbb{E}\Big[\theta_A \mid \boldsymbol{x}_A\Big] \\ &= \frac{\alpha_A + \#(A=1)}{\alpha_A + \#(A=1) + \beta_A + \#(A=0)}. \end{aligned}$$

Where the last equality is given by the expected value of a Beta distribution.

For $P(d \mid a, b)$ the situation is more complex, the simplest approach is to specify a Beta prior for each of the components of $\boldsymbol{\theta}_D$. Focus on θ_2, notice the parameters α and β we used before now do depend on A and B, these are called *hyperparameters*.

$$\theta_2 \sim Beta\Big(\alpha_D(1,0) + \#(D=1,A=1,B=0),\ \beta_D(1,0) + \#(D=0,A=1,B=0)\Big).$$

Repeating the procedure we used with A we get that

$$P(D=1 \mid A=1,B=0,\boldsymbol{x}) = \frac{\alpha_D(1,0) + \#(D=1,A=1,B=0)}{\alpha_D(1,0) + \beta_D(1,0) + \#(A=1,B=0)}.$$

All hyperparameters could be set to the same value, where a complete ignorance prior would correspond to set them to 1.

There are two limit possibilities depending on the amount of data available.

- **No data**. The marginal probability corresponds to the prior, which in the last case is

  $$P(D=1 \mid A=1,B=0,\boldsymbol{x}) = \frac{\alpha_D(1,0)}{\alpha_D(1,0) + \beta_D(1,0)}.$$

  Note that equal hyperparameters would give a result of 0.5.

- **Infinite data**. When infinite data is available, the marginal is generally dominated by it, this corresponds to the Maximum Likelihood solution.

  $$P(D=1 \mid A=1,B=0,\boldsymbol{x}) = \frac{\#(D=1,A=1,B=0)}{\#(A=1,B=0)}.$$

  This happens unless the prior has a pathologically strong effect.

Consider the data given in the table in figure 12, and equal parameters and hyperparameters 1 and a prior belief that any setting is equally probable, i.e, $P(A=1) = 0.5$.

We may illustrate the different results that are obtained using using Bayesian inference and Maximum likelihood training. The former is

$$P(A=1 \mid \boldsymbol{x}) = \frac{1 + \#(A=1)}{2+N} = \frac{5}{9} \approx 0.556.$$

and the latter is $4/7 = 0.571$. In conclusion, the Bayesian result is more prudent than this one, which fits in with our prior belief.

18.3 LEARNING DISCRETE VARIABLES

The natural generalization to more than two states is using a Dirichlet distribution as prior, assuming i.i.d data, local and global prior independence. Two different scenarios are being considered, firstly one where the variable has no parents, as the case for A and B in the previous example. And secondly a variable with a non-void set of parents.

Let $\boldsymbol{x}$ denote the given dataset as observations of a random variable X, and $\boldsymbol{\theta}$ the set of parameters modeling the experiment.

No parents

Consider a variable $X \sim Categorical(\boldsymbol{\theta})$ with $Dom(X) = \{1, \dots, I\}$ and $\boldsymbol{\theta} = (\theta_1, \dots, \theta_I)$ so that

$$P(x) = \prod_{i=1}^{I} \theta_i^{\mathbb{I}[x=i]} \text{ with } \sum_{i=1}^{I} \theta_i = 1.$$

So that the posterior is

$$P(\boldsymbol{\theta} \mid \boldsymbol{x}) = \frac{1}{P(\boldsymbol{x})} P(\boldsymbol{\theta}) \prod_{n=1}^{N} \prod_{i=1}^{I} \theta_i^{\mathbb{I}[x_n=i]} = \frac{1}{P(\boldsymbol{x})} P(\boldsymbol{\theta}) \prod_{i=1}^{I} \theta_i^{\sum_n \mathbb{I}[x_n=i]}.$$

Assuming a Dirichlet prior $\boldsymbol{\theta} \sim Dirichlet(\boldsymbol{u})$ with $\boldsymbol{u} = (u_1, \dots, u_I)$ and using the fact that the Dirichlet distribution is the conjugate prior of the Categorical distribution (Appendix B), the posterior follows a Dirichlet distribution

$$\boldsymbol{\theta} \mid \boldsymbol{x} \sim Dirichlet(\boldsymbol{u} + \boldsymbol{c}).$$

Where

$$\boldsymbol{c} = \Big(\sum_{n=1}^{N} \mathbb{I}[x_n = 1], \dots, \sum_{n=1}^{N} \mathbb{I}[x_n = I] \Big).$$

The predictive marginal is then given by

$$\begin{aligned} P(X = i \mid \boldsymbol{x}) &= \int_{\theta} P(X = i \mid \theta) P(\theta \mid \boldsymbol{x}) = \int_{\theta} \theta_i P(\theta \mid \boldsymbol{x}) \\ &= \int_{\theta_i} \theta_i P(\theta_i \mid \boldsymbol{x}) = \mathbb{E}\Big[\theta_i \mid \boldsymbol{x}\Big]. \end{aligned}$$

Where we used that

$$\int_{\theta_{j \neq i}} \theta_i P(\theta \mid \boldsymbol{x}) = \theta_i \prod_{k \neq j} P(\theta_k \mid \boldsymbol{x}) \int_{\theta_j} P(\theta_j \mid \boldsymbol{x}) = \theta_i \prod_{k \neq j} P(\theta_k \mid \boldsymbol{x}).$$

From Proposition 2, the uni-variate marginal of a Dirichlet distribution is a Beta distribution of parameters

$$\theta_i \mid \boldsymbol{x} \sim \text{Beta}(u_i + c_i, \sum_{j \neq i} u_j + c_j).$$

Using the expectation of a Beta distribution we get that

$$P(X = i \mid \boldsymbol{x}) = \frac{u_i + c_i}{\sum_j u_j + c_j}.$$

Parents

Consider now that X has a set of parent variables $pa(X)$, in this case, we want to compute the marginal given a state of its parents and the data:

$$P(X = i \mid pa(X) = \boldsymbol{j}, \boldsymbol{x}).$$

We are using the following notation for the parameters

$$P(X = i \mid pa(X) = \boldsymbol{j}, \theta) = \theta_{i,\boldsymbol{j}}, \quad \boldsymbol{\theta_j} = (\theta_{1,\boldsymbol{j}}, \ldots, \theta_{I,\boldsymbol{j}}).$$

Local independence means that

$$P(\boldsymbol{\theta}) = \prod_j P(\boldsymbol{\theta_j}).$$

As we did before, we consider a Dirichlet prior

$$\boldsymbol{\theta_j} \sim Dirichlet(\boldsymbol{u_j}),$$

the posterior is then

$$\begin{aligned}
P(\boldsymbol{\theta} \mid \boldsymbol{x}) &= \frac{P(\boldsymbol{\theta})P(\boldsymbol{x} \mid \boldsymbol{\theta})}{P(\boldsymbol{x})} = \frac{1}{P(\boldsymbol{x})}\Big(\prod_{\boldsymbol{j}} P(\boldsymbol{\theta_j})\Big)P(\boldsymbol{x} \mid \boldsymbol{\theta}) \\
&= \frac{1}{P(\boldsymbol{x})}\Big(\prod_{\boldsymbol{j}} \frac{1}{B(\boldsymbol{u_j})}\prod_i \theta_{i,\boldsymbol{j}}^{u_{i,\boldsymbol{j}}-1}\Big)P(\boldsymbol{x} \mid \boldsymbol{\theta}) \\
&= \frac{1}{P(\boldsymbol{x})}\Big(\prod_{\boldsymbol{j}} \frac{1}{B(\boldsymbol{u_j})}\prod_i \theta_{i,\boldsymbol{j}}^{u_{i,\boldsymbol{j}}-1}\Big)\Big(\prod_n \prod_{\boldsymbol{j}} \prod_i \theta_{i,\boldsymbol{j}}^{\mathbb{I}[x_n=i, pa(x_n)=\boldsymbol{j}]}\Big) \\
&= \frac{1}{P(\boldsymbol{x})}\prod_{\boldsymbol{j}} \frac{1}{B(\boldsymbol{u_j})}\prod_i \theta_{i,\boldsymbol{j}}^{u_{i,\boldsymbol{j}}-1+\#(X=i, pa(X)=\boldsymbol{j})}.
\end{aligned}$$

Naming $\boldsymbol{v_j} = \boldsymbol{u_j} + \#(X = i, pa(X) = \boldsymbol{j})$ and using the same argument as we did in the *no parents* case with the normalization constants, the posterior is

$$(\boldsymbol{\theta} \mid \boldsymbol{x}) \sim \prod_j \text{Dirichlet}(\boldsymbol{v_j}).$$

Denoting $v_{i,j}$ the components of $\boldsymbol{v_j}$, the marginal is

$$P(X = i, pa(X) = \boldsymbol{j}, \boldsymbol{x}) = \frac{v_{i,j}}{\sum_i v_{i,j}}.$$

Notice all the above has been done using a fixed variable X, so that all the parameters depend on that variable.

19

EXPECTATION-MAXIMIZATION ALGORITHM

As both hidden variables and Bayesian networks are used, the following notation from Barber (2007) is used: $\boldsymbol{X} = (\boldsymbol{V}, \boldsymbol{H}) = (X_1, \ldots, X_M)$ is the set of variables partitioned in visible and hidden. Let $\{\boldsymbol{v}^1, \ldots, \boldsymbol{v}^N\}$ be the set of observations of the visible variables. For each data-point $\boldsymbol{x}^n = (\boldsymbol{v}^n, \boldsymbol{h}^n)$ decomposes into visible and hidden parts.

Remember the ELBO is:

$$\text{ELBO}(Q) = \underbrace{\sum_{n=1}^{N} \mathbb{E}_{Q(\boldsymbol{h}^n)}\Big[\log Q(\boldsymbol{h}^n)\Big]}_{Entropy} + \underbrace{\sum_{n=1}^{N} \mathbb{E}_{Q(\boldsymbol{h}^n)}\Big[\log P(\boldsymbol{v}^n, \boldsymbol{h}^n \mid \boldsymbol{\theta})\Big]}_{Energy}.$$

Where the steps of the EM algorithm are:

- **E-step**: $Q^{new}(\boldsymbol{h}^n) = P(\boldsymbol{h}^n \mid \boldsymbol{v}^n, \boldsymbol{\theta}) \quad \forall n = 1, \ldots, N.$
- **M-step**: $\boldsymbol{\theta}^{new} = \arg\max_{\boldsymbol{\theta}} \text{ELBO}(Q).$

Whereas the structure of the Bayesian network gives no advantage over the E-step, it does on the M-step. Let the variational distribution be fixed as Q_{old}, the *energy term* in a Bayesian network is

$$\sum_{n=1}^{N} \mathbb{E}_{Q_{old}(\boldsymbol{h}^n)}\Big[\log P(\boldsymbol{x}^n \mid \boldsymbol{\theta})\Big] = \sum_{n=1}^{N} \sum_{m=1}^{M} \mathbb{E}_{Q_{old}(\boldsymbol{h}^n)}\Big[\log P(x_m^n \mid pa(x_m^n), \boldsymbol{\theta})\Big].$$

It is useful to use the following notation that defines a conditional distribution of the hidden variable when the visible one equals $\boldsymbol{v}^n$.

$$Q^n(\boldsymbol{x}) = Q^n(\boldsymbol{v}, \boldsymbol{h}) = Q_{old}(\boldsymbol{h}^n)\mathbb{I}(\boldsymbol{v} = \boldsymbol{v}^n).$$

A mixture distribution might be defined as

$$Q(\boldsymbol{x}) = \frac{1}{N}\sum_{n=1}^{N} Q^n(\boldsymbol{x}).$$

One may show that the energy term equals

$$\sum_{n=1}^{N} \mathbb{E}_{Q_{old}(\boldsymbol{h}^n)}\Big[\log P(\boldsymbol{x}^n \mid \boldsymbol{\theta})\Big] = N\, \mathbb{E}_{Q(\boldsymbol{x})}\Big[\log P(\boldsymbol{x} \mid \boldsymbol{\theta})\Big],$$

Algorithm 5: Expectation Maximization Algorithm for Bayesian networks

Data: A dataset $\boldsymbol{x}$, a distribution $P(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\theta})$ that factorizes in a Bayesian network.
Result: The maximum likelihood estimates for $P(x_m \mid pa(x_m)), m = m, \ldots, M$,
Initialize $P(x_m \mid pa(x_i)), m = 1, \ldots, M$;
$t \leftarrow 0$;
while *Convergence stop criteria* **do**
  for $n = 1$ *to* N **do**
    $Q_t^n(\boldsymbol{x}) = P_t(\boldsymbol{h}^n \mid \boldsymbol{v}^n)\delta(\boldsymbol{v}, \boldsymbol{v}^n)$; // E-step
  end
  for $m = 1$ *to* M **do**
    $P_t(x_m \mid pa(x_m)) = \frac{\sum_{n=1}^{N} Q_t^n(x_m, pa(x_m))}{\sum_{n=1}^{N} Q_t^n(pa(x_m))}$; // M-step
  end
  $t \leftarrow t + 1$;
end
return $P_t(x_m \mid pa(x_m)), \quad m = 1, \ldots, M$;

as

$$\begin{aligned}
\mathbb{E}_{Q(\boldsymbol{x})}\Big[\log P(\boldsymbol{x} \mid \theta)\Big] &= \int_{\boldsymbol{x}} Q(\boldsymbol{x}) \log P(\boldsymbol{x} \mid \boldsymbol{\theta}) = \int_{\boldsymbol{x}} \frac{1}{N} \sum_{n=1}^{N} Q^n(\boldsymbol{x}) \log P(\boldsymbol{x} \mid \boldsymbol{\theta}) \\
&= \frac{1}{N} \int_{\boldsymbol{x}} \sum_{n=1}^{N} Q_{old}(\boldsymbol{h}^n) \mathbb{I}[\boldsymbol{v} = \boldsymbol{v}^n] \log P(\boldsymbol{x} \mid \boldsymbol{\theta}) \\
&= \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}_{Q_{old}(\boldsymbol{h}^n)}\Big[\log P(\boldsymbol{h}^n, \boldsymbol{v}^n \mid \boldsymbol{\theta})\Big] \\
&= \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}_{Q_{old}(\boldsymbol{h}^n)}\Big[\log P(\boldsymbol{x}^n \mid \boldsymbol{\theta})\Big],
\end{aligned}$$

In the other hand, using the Bayesian network structure and Proposition 1:

$$\begin{aligned}
\mathbb{E}_{Q(x)}\Big[\log P(\boldsymbol{x} \mid \boldsymbol{\theta})\Big] &= \sum_{m=1}^{M} \mathbb{E}_{Q(\boldsymbol{x})}\Big[\log P(x_m \mid pa(x_m), \boldsymbol{\theta})\Big] \\
&= \sum_{m=1}^{M} \mathbb{E}_{Q(x_m, pa(x_m))}\Big[\log P(x_m \mid pa(x_m, \boldsymbol{\theta}))\Big] \\
&= \sum_{m=1}^{M} \mathbb{E}_{Q(pa(x_m))}\Big[\mathbb{E}_{Q(x_m \mid pa(x_m))}\Big[\log P(x_m \mid pa(x_m), \boldsymbol{\theta})\Big]\Big].
\end{aligned}$$

Adding a constant as

$$\mathbb{E}_{Q(pa(x_m))}\Big[\mathbb{E}_{Q(x_m \mid pa(x_m))}\Big[\log Q(x_m \mid pa(x_m))\Big]\Big]$$

to the last term results in a Kullback-Leibler Divergence:

$$\sum_{m=1}^{M} E_{Q(pa(x_m))}\Big[KL\Big(Q(x_m \mid pa(x_m)) \mid P(x_m \mid pa(x_m), \boldsymbol{\theta})\Big)\Big]$$

$$= \sum_{m=1}^{M} \mathbb{E}_{Q(pa(x_m))} \Big[\mathbb{E}_{Q(x_m|pa(x_m))} \Big[\log Q(x_m \mid pa(x_m)) \Big] - \mathbb{E}_{Q(x_m|pa(x_m))} \Big[\log P(x_m \mid pa(x_m), \boldsymbol{\theta}) \Big] \Big].$$

So maximizing the energy term is equivalent to minimize the above formula, that is, setting

$$P^{new}(x_m \mid pa(x_m), \boldsymbol{\theta}) = Q(x_m \mid pa(x_m)).$$

So the first observation is that $\boldsymbol{\theta}$ is not needed in order to maximize the energy term due to the Belief network structure. Algorithm 5 shows the full procedure of the algorithm in Bayesian networks.

20

VARIATIONAL MESSAGE PASSING ALGORITHM

In this section, the *Variational Message Passing* or *VMP* algorithm (Winn & Bishop (2005); Bishop *et al.* (2003)) is reviewed as a variational Bayes application to Bayesian networks. The exponential family is considered using a conditionally conjugate model where global hidden variables are simple hidden variables. The algorithm consists of a message passing procedure between the nodes of the given graphical model.

VMP automatically applies variational inference to a wide class of Bayesian networks. Its main advantage is that no application-specific updates equations need to be derived.

The full set of variables is $\boldsymbol{X} = (X_1 \ldots, X_N)$, where both visible and hidden variables $\boldsymbol{Z} = (Z_1, \ldots, Z_J) \subset \boldsymbol{X}$ are being considered with a variational distribution Q in the mean-field family

$$Q(\boldsymbol{Z}) = \prod_{j=1}^{J} Q(z_j).$$

The optimized factor for a fixed term $Q(z)$ is (as shown in the CAVI update 1) given by

$$\log Q^{new}(z) = \mathbb{E}_{Q_{\setminus Z}}\Big[\log P(\boldsymbol{x})\Big] + \text{const.} \tag{3}$$

Using the Bayesian network structure, the update is factorized as

$$\log Q^{new}(z) = \mathbb{E}_{Q_{\setminus Z}}\Big[\sum_{n=1}^{N} \log P(x_n \mid pa(x_n))\Big] + \text{const.}$$

The contribution of the latent variable Z to the update lies on the terms $P(z \mid pa(z))$ and the conditionals of all its children. Let $cp(Z, X)$ denote the co-parents of Z with X,

$$cp(Z, X) = pa(X) \setminus \{Z\},$$

then, arranging all other terms to the constant term, the variational update is given by

$$\log Q^{new}(z) = \mathbb{E}_{Q_{\setminus Z}}\Big[\log P(z \mid pa(z))\Big] + \sum_{X \in ch(Z)} \mathbb{E}_{Q_{\setminus Z}}\Big[\log P(x \mid z, cp(z, x))\Big] + \text{const.} \tag{4}$$

This shows how the update of a hidden variable only depends on its Markov blanket. The optimization of Q is therefore computed as the sum of a term involving Z and its parent nodes, along with a term for each children. These terms are seen as "messages" between nodes.

The exact messages depends on the conditional distributions of the model and their functional form. The variational update equations present important simplifications when the conditional distribution of a node given its parents is in the exponential family. Sub-indexes will be used to denote parents, children and co-parents in order to simplify the notation.

Consider a latent variable Z and $X \in ch_Z$. Then suppose that both distributions belong to the exponential family as

$$P(z \mid pa_z) = h_Z(z) \exp\Big(\boldsymbol{\eta}_Z(pa_z)^T \boldsymbol{T}_Z(z) - \psi_Z(pa_z)\Big).$$

$$P(x \mid z, cp_{z,x}) = h_X(x) \exp\Big(\boldsymbol{\eta}_X(z, cp_{z,x})^T \boldsymbol{T}_X(x) - \psi_X(z, cp_{z,x})\Big).$$

The distribution $P(z \mid pa_z)$ can be though as a prior belief over Z, whereas $P(x \mid pa_x) = P(x \mid z, cp_{z,x})$ might be a contribution to the likelihood of Z.

In order to achieve conjugacy, the product of these distributions must be in the same form as the prior, to do so they must have the same form with respect to Z. In short, $P(x \mid z, cp_{z,x})$ needs to be written in terms of Z's natural statistic $\boldsymbol{T}_Z(z)$. This is done by defining functions $\boldsymbol{\eta}_{X,Y}$ and λ such that

$$\log P(x \mid z, cp_{z,x}) = \boldsymbol{\eta}_{X,Z}(x, cp_{z,x})^T \boldsymbol{T}_Z(z) + \lambda(x, cp_{z,x}). \tag{5}$$

Example 6. If X is Gaussian distributed with mean μ (Gaussian distributed) and precision τ (Gamma distributed), the log conditional is

$$\log P(x \mid \mu, \tau) = \begin{pmatrix} \mu\tau & -\tau/2 \end{pmatrix} \begin{pmatrix} x \\ x^2 \end{pmatrix} - \frac{\mu^2\tau}{2} + \frac{1}{2}(\log\tau - \log 2\pi).$$

We may rewrite it the conditional as

$$\log P(x \mid \mu, \tau) = \begin{pmatrix} \tau x & -\tau/2 \end{pmatrix} \begin{pmatrix} \mu \\ \mu^2 \end{pmatrix} + \frac{1}{2}(\log\tau - \tau x^2 - \log 2\pi),$$

where

$$\boldsymbol{\eta}_{X,\mu}(x, \tau) = \begin{pmatrix} \tau x \\ -\tau/2 \end{pmatrix}^T \quad \text{and} \quad \boldsymbol{T}_\mu(\mu) = \begin{pmatrix} \mu \\ \mu^2 \end{pmatrix}.$$

As a result, the variational update of a variable (Equation 3) might be written in terms of $\boldsymbol{T}_Z(z)$.

$$\begin{aligned}
\log Q^{new}(z) &= \mathbb{E}_{Q_{\setminus Z}}\Big[\log(h_Z(pa_z)) + \boldsymbol{\eta}_Z(pa_z)^T\boldsymbol{T}_Z(z) + \psi_Z(pa_z)\Big] \\
&\quad + \sum_{X\in ch(Z)} \mathbb{E}_{Q_{\setminus Z}}\Big[\boldsymbol{\eta}_{X,Z}(x, cp_{z,x})^T\boldsymbol{T}_Z(z) + \lambda_X(x, cp_{z,x})\Big] + \text{const.} \\
&= \left[\mathbb{E}_{Q_{\setminus Z}}\Big[\boldsymbol{\eta}_Z(pa_Z)^T\Big] + \sum_{X\in ch(Z)} \mathbb{E}_{Q_{\setminus Z}}\Big[\boldsymbol{\eta}_{X,Z}(x, cp_{z,x})^T\Big]\right]^T \boldsymbol{T}_Z(z) \\
&\quad + \log h_Z(pa_z) + \text{const.} \\
&= \left[\mathbb{E}_{Q}\Big[\boldsymbol{\eta}_Z(pa_Z)^T\Big] + \sum_{X\in ch(Z)} \mathbb{E}_{Q}\Big[\boldsymbol{\eta}_{X,Z}(x, cp_{z,x})^T\Big]\right]^T \boldsymbol{T}_Z(z) \\
&\quad + \log h_Z(pa_z) + \text{const.}
\end{aligned}$$

Where the last equality comes from the fact that Z does not appear inside the expectation. Which means that $Q^{new}(z)$ is in the exponential family with the form of $P(z \mid pa_z)$ with the following parameter function

$$\boldsymbol{\eta}_Z^{new} = \mathbb{E}_Q\Big[\boldsymbol{\eta}_Z(pa_Z)^T\Big] + \sum_{X\in ch(Z)} \mathbb{E}_Q\Big[\boldsymbol{\eta}_{X,Z}(x, cp_{z,x})^T\Big]. \tag{6}$$

From Equation 5, it can be seen that $\log P(x \mid z, cp_{z,x})$ is linear in $\boldsymbol{T}_X(x)$ and $\boldsymbol{T}_Z(z)$, and, by the same reasoning, linear in any sufficient statistic of any parent of X. This means that the variational update given in Equation 3 is a multi-linear[1] function of the expectations of the natural statistics $\mathbb{E}_Q\Big[\boldsymbol{T}\Big]$ of each node in Z's Markov blanket. As a result, the right hand side of Equation 6 is multi-linear on the expectations of the statistic functions of their corresponding variables. And so are the expectations of $\boldsymbol{\eta}_Z$ and $\boldsymbol{\eta}_{X,Z}$. Therefore, it is possible to reparameterize these functions in terms of these expectations

$$\bar{\boldsymbol{\eta}}_Z\left(\Big\{\mathbb{E}_Q\Big[\boldsymbol{T}_x(x)\Big]\Big\}_{x\in pa_Z}\right) = \mathbb{E}_Q\Big[\boldsymbol{\eta}_Z(pa_Z)\Big],$$

$$\bar{\boldsymbol{\eta}}_{X,Z}\left(\mathbb{E}_Q\Big[\boldsymbol{T}_X(x)\Big], \Big\{\mathbb{E}_Q\Big[\boldsymbol{T}_Y(y)\Big]\Big\}_{Y\in cp_{Z,X}}\right) = \mathbb{E}_Q\Big[\boldsymbol{\eta}_{X,Z}(x, cp_{z,x})\Big].$$

Example 7. The log conditional in the previous example $\log P(x \mid \mu, \tau)$ is multi-linear in

$$\boldsymbol{T}(x) = \begin{pmatrix} x \\ x^2 \end{pmatrix}, \quad \boldsymbol{T}(\mu) = \begin{pmatrix} \mu \\ \mu^2 \end{pmatrix} \quad \text{and} \quad \boldsymbol{T}(\tau) = \begin{pmatrix} \tau \\ \log\tau \end{pmatrix}.$$

The multi-linear function $\boldsymbol{\eta}_{X,\mu}(x,\tau) = \begin{pmatrix} \tau x \\ -\tau/2 \end{pmatrix}$ is defined. It can be reparameterized as

$$\bar{\boldsymbol{\eta}}_{X,\mu}\Big(\mathbb{E}\Big[\boldsymbol{T}(x)\Big], \mathbb{E}\Big[\boldsymbol{T}(\tau)\Big]\Big) = \begin{pmatrix} \mathbb{E}\Big[\boldsymbol{T}(\tau)\Big]_0 \mathbb{E}\Big[\boldsymbol{T}(x)\Big]_0 \\ -\frac{1}{2}\mathbb{E}\Big[\boldsymbol{T}(\tau)\Big]_0 \end{pmatrix}$$

1 A function is multi-linear on a set of variables if it varies linearly with respect to each one.

Where the sub-indexes refer to the component of each vector. So that $\mathbb{E}\Big[\boldsymbol{T}(x)\Big]_0 = \mathbb{E}\Big[x\Big]$ and $\mathbb{E}\Big[\boldsymbol{T}(\tau)\Big]_0 = \mathbb{E}\Big[\tau\Big]$.

To sum up, to compute the variational update, only the expectations of the sufficient statistics are needed, which might be computed using the following proposition.

Proposition 7. *Given a variable X in the exponential family, with known natural parameter vector $\boldsymbol{\eta}$, the expectation of the statistic function with respect to the distribution might be calculated as:*

$$\mathbb{E}_P\Big[\boldsymbol{T}(x)\Big] = -\frac{d\bar{\psi}(\boldsymbol{\eta}(\theta))}{d\boldsymbol{\eta}}.$$

Proof. Defining $\bar{\phi}$ as a reparameterisation of ϕ in terms of $\boldsymbol{\eta}$:

$$\begin{aligned} P(x \mid \theta) &= h(x) \exp\Big(\boldsymbol{\eta}(\theta)^T \boldsymbol{T}(x) + \psi(\theta)\Big) \\ &= h(x) \exp\Big(\boldsymbol{\eta}(\theta)^T \boldsymbol{T}(x) + \bar{\psi}(\boldsymbol{\eta}(\theta))\Big), \end{aligned}$$

then, integrating with respect to X:

$$1 = \int_x h(x) \exp\Big(\boldsymbol{\eta}(\theta)^T \boldsymbol{T}(x) + \bar{\psi}(\boldsymbol{\eta}(\theta))\Big).$$

Differentiating with respect to $\boldsymbol{\eta}$:

$$\frac{d}{d\theta}1 = 0 = \int_x \frac{d}{d\boldsymbol{\eta}} h(x) \exp\Big(\boldsymbol{\eta}(\theta)^T \boldsymbol{T}(x) + \bar{\psi}(\boldsymbol{\eta}(\theta))\Big) = \int_x P(x \mid \theta)\Big[\boldsymbol{T}(x) + \frac{d\bar{\psi}(\boldsymbol{\eta}(\theta))}{d\boldsymbol{\eta}}\Big].$$

What implies that the expectation of the statistic function is

$$\mathbb{E}_P\Big[\boldsymbol{T}(x)\Big] = -\frac{d\bar{\psi}(\boldsymbol{\eta}(\theta))}{d\boldsymbol{\eta}}.$$

□

The messages between each node are therefore defined as the following (the full algorithm is shown in Algorithm 6):

- The message from a parent node X to a child node Z is the expectation under Q of its natural statistic:
$$\boldsymbol{m}_{X \to Z} = \mathbb{E}_Q\Big[\boldsymbol{T}_X(x)\Big].$$
- The message from a child node X to a parent node Z is:
$$\boldsymbol{m}_{X \to Z} = \bar{\boldsymbol{\eta}}_{X,Z}\Big(\mathbb{E}_Q\Big[\boldsymbol{T}_X(x)\Big], \{\boldsymbol{m}_{Y \to X}\}_{Y \in cp_{Z,X}}\Big),$$
which relied on having received all messages from all the co-parents. If a node X is observed, its messages are defined as $\boldsymbol{T}_X(x)$ instead of $\mathbb{E}_Q\Big[\boldsymbol{T}_X(x)\Big]$.

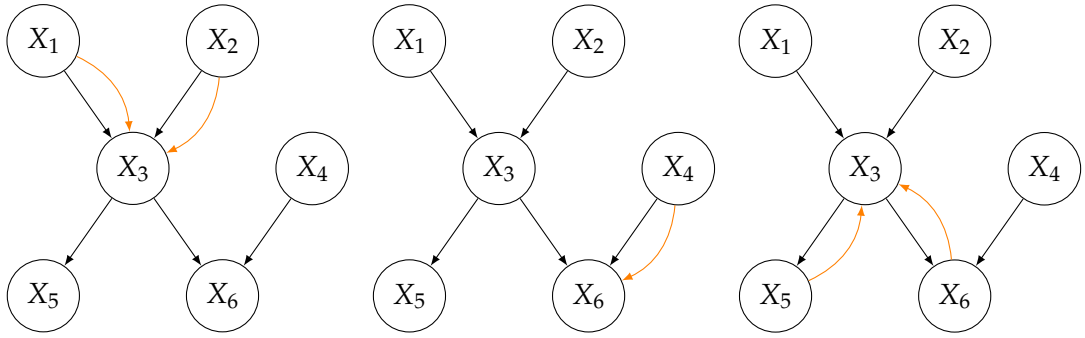


Figure 13: Six node example of one variational message passing algorithm updating node X_3 . **1.** All parent nodes pass their messages. **2.** Send all messages from co-parents to its children. **3.** Send messages from children to X_3.

When a node Z has received all messages from its parents and children, we can compute the updated parameter $\boldsymbol{\eta}_Z^{new}$ as

$$\begin{aligned}\boldsymbol{\eta}_Z^{new} &= \mathbb{E}_Q\Big[\boldsymbol{\eta}_Z(pa_Z)\Big] + \sum_{X \in ch(Z)} \mathbb{E}_Q\Big[\boldsymbol{\eta}_{X,Z}(x, cp_{z,x})\Big] \\ &= \bar{\boldsymbol{\eta}}_Z(\{\boldsymbol{m}_{X\to Z}\}_{X\in pa_Z}) + \sum_{X\in ch_Z} \boldsymbol{m}_{X\to Z}.\end{aligned}$$

Remark 7. In case Z has no parent nodes, the first term of the update equals its natural parameter function $\boldsymbol{\eta}_Z$ evaluated on its hyper-parameters. The update of μ in Section 20.1 is an example.

Example 8. If X is Gaussian distributed and μ, τ are its parents, the messages from X to its parents are:

$$\boldsymbol{m}_{X\to\mu} = \begin{pmatrix}\mathbb{E}\Big[\tau\Big]\mathbb{E}\Big[X\Big] \\ -\mathbb{E}\Big[\tau\Big]/2\end{pmatrix}, \quad \boldsymbol{m}_{X\to\tau} = \begin{pmatrix}-\frac{1}{2}\left(\mathbb{E}\Big[x^2\Big] - 2\mathbb{E}\Big[x\Big]\mathbb{E}\Big[\mu\Big] + \mathbb{E}\Big[\mu^2\Big]\right) \\ \frac{1}{2}\end{pmatrix}.$$

And the messages from X to any child node Y (in case it had any) is

$$\boldsymbol{m}_{X\to Y}\begin{pmatrix}\mathbb{E}\Big[\tau\Big]\mathbb{E}\Big[x\Big] \\ -\mathbb{E}\Big[x^2\Big]\end{pmatrix}.$$

20.1 EXAMPLE: UNI-VARIATE GAUSSIAN MODEL

In order to illustrate the procedure that VMP follows, the following uni-variate Gaussian model is used. Let X be a Gaussian distributed random variable with unknown mean μ and precision τ. Let $\boldsymbol{x} = (x_1, \ldots, X_N)$ be a set of i.i.d observations of the variable, such as

$$P(\boldsymbol{x} \mid \mu, \tau^{-1}) \equiv \prod_{n=1}^{N} \mathcal{N}(\mu, \tau^{-1}).$$

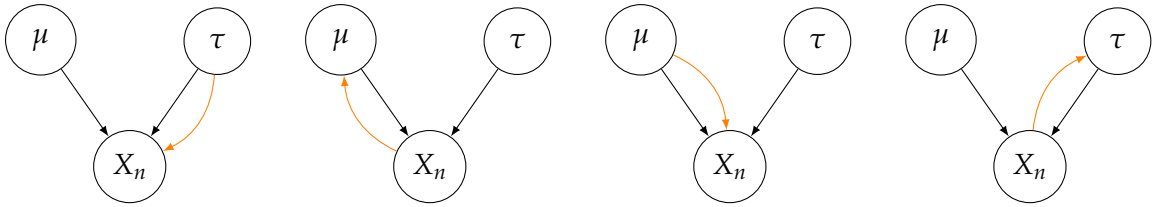


Figure 14: Message passing procedure of the uni-variate Gaussian model.

The parameters are assumed to follow a Gaussian and Gamma distribution respectively:

$$\log P(\mu) = \begin{pmatrix} \alpha\beta \\ -\beta/2 \end{pmatrix}^T \begin{pmatrix} \mu \\ \mu^2 \end{pmatrix} + \frac{1}{2}(\log\beta + \beta\alpha^2 - \log 2\pi)$$
$$\log P(\tau) = \begin{pmatrix} -b \\ a-1 \end{pmatrix}^T \begin{pmatrix} \tau \\ \log\tau \end{pmatrix} + a\log b - \log\Gamma(a)$$

Given this, their sufficient statistics are:

$$\boldsymbol{T}(\mu) = \begin{pmatrix} \mu \\ \mu^2 \end{pmatrix} \quad \boldsymbol{T}(\tau) = \begin{pmatrix} \tau \\ \log\tau \end{pmatrix}.$$

The log conditional of a given observation x_n takes the form

$$\log P(x_n \mid \mu, \tau^{-1}) = \begin{pmatrix} \tau\mu & -\tau/2 \end{pmatrix} \begin{pmatrix} x_n \\ x_n^2 \end{pmatrix} + \frac{1}{2}(\log\tau + \tau\mu^2 + \log 2\pi).$$

The reparameterizations of this conditional are

$$\begin{aligned} \log P(x_n \mid \mu, \tau^{-1}) &= \begin{pmatrix} \tau x_n & -\tau/2 \end{pmatrix} \begin{pmatrix} \mu \\ \mu^2 \end{pmatrix} + \frac{1}{2}(\log\tau + \tau x_n^2 + \log 2\pi) \\ &= \begin{pmatrix} -\frac{1}{2}(x_n-\mu)^2 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} \tau \\ \log\tau \end{pmatrix} - \log 2\pi \end{aligned}$$

The first step of the algorithm is initializing the variational distribution

$$Q(\mu, \tau) = Q(\mu)Q(\tau),$$

and the initial values of $\mathbb{E}_Q\left[\boldsymbol{T}(\mu)\right]$ and $\mathbb{E}_Q\left[\boldsymbol{T}(\tau)\right]$. The update procedure is now described as seen in Figure 14.

1. Let us start with μ, as the variable has no parent nodes, the only messages have to be retrieved from the observed variables, to do so, each X_n needs to retrieve its message from τ.
$$\boldsymbol{m}_{\tau\to x_n} = \mathbb{E}_Q\left[\boldsymbol{T}(\tau)\right] = \begin{pmatrix} \mathbb{E}_Q\left[\tau\right] & \mathbb{E}_Q\left[\log\tau\right] \end{pmatrix}^T.$$

2. The message from X_n to μ can now be sent. This is the expectation of $\boldsymbol{\eta}_{\mu,x_n}(x_n,\tau) = \begin{pmatrix}\tau x_n & -\tau/2\end{pmatrix}$

$$\boldsymbol{m}_{x_n\to\mu} = \mathbb{E}_Q\Big[\boldsymbol{\eta}_{\mu,x_n}(x_n,\tau)\Big] = \begin{pmatrix}\mathbb{E}_Q\Big[\tau\Big]x_n \\ -\frac{\mathbb{E}_Q\Big[\tau\Big]}{2}\end{pmatrix}.$$

3. The variational distribution of μ is updated as

$$\boldsymbol{\eta}_\mu^{new} = \mathbb{E}_Q\Big[\boldsymbol{\eta}(\alpha,\beta)\Big] + \sum_{x_n}\boldsymbol{m}_{x_n\to\mu} = \begin{pmatrix}\alpha\beta \\ -\beta/2\end{pmatrix} + \sum_{x_n}\begin{pmatrix}\mathbb{E}_Q\Big[\tau\Big]x_n \\ -\frac{\mathbb{E}_Q\Big[\tau\Big]}{2}\end{pmatrix}.$$

The new expectation of $\boldsymbol{T}(\mu)$ is computed and send to each X_n as

$$\boldsymbol{m}_{\mu\to X_n} = \begin{pmatrix}\mathbb{E}_Q\Big[\mu\Big] \\ \mathbb{E}_Q\Big[\mu^2\Big]\end{pmatrix}.$$

4. Each X_n sends a message to τ:

$$\boldsymbol{m}_{X_n\to\tau} = \begin{pmatrix}-\frac{1}{2}\Big(x_n^2 - 2x_n\mathbb{E}_Q\Big[\mu\Big] + \mathbb{E}_Q\Big[\mu^2\Big]\Big) \\ \frac{1}{2}\end{pmatrix}.$$

Which allows τ to update its distribution

$$\boldsymbol{\eta}_\tau^{new} = \begin{pmatrix}-b \\ a-1\end{pmatrix} + \sum_{x_n}\boldsymbol{m}_{x_n\to\tau}.$$

Algorithm 6: Variational Message Passing Algorithm

Data: Bayesian network structure with latent variables $\boldsymbol{Z}$ and a dataset. Each variable must belong to the exponential family.

Result: Optimized parameters for the variational distributions of each variable.

for $Z \in \boldsymbol{Z}$ **do**

  Initialize its sufficient statistic expectation $\mathbb{E}_Q\left[\boldsymbol{T}(z)\right]$.

end

while *convergence stop criteria* **do**

  for $Z \in \boldsymbol{Z}$ **do**

    for $Y \in pa(Z)$ **do**

      // Retrieve messages from its parent nodes.

      if $Y \in \boldsymbol{V}$ **then**

        $\boldsymbol{m}_{Y \to Z} = \boldsymbol{T}_Y(y)$

      end

      else

        $\boldsymbol{m}_{Y \to Z} = \mathbb{E}_Q\left[\boldsymbol{T}_Y(y)\right]$

      end

    end

    for $Y \in ch(Z)$ **do**

      // Retrieve messages from the co-parent nodes.

      for $Z \in cp(Z, Y)$ **do**

        // Send their message to the child node.

        if $Z \in \boldsymbol{V}$ **then**

          $\boldsymbol{m}_{Z \to Y} = \boldsymbol{T}_Z(z)$

        end

        else

          $\boldsymbol{m}_{Z \to Y} = \mathbb{E}_Q\left[\boldsymbol{T}_Z(z)\right]$

        end

      end

      if $Y \in \boldsymbol{V}$ **then**

        $\boldsymbol{m}_{Y \to Z} = \bar{\boldsymbol{\eta}}_{Y,Z}\left(\boldsymbol{T}_Y(y), \{\boldsymbol{m}_{Z \to Y}\}_{Z \in cp_{Z,Y}}\right)$

      end

      else

        $\boldsymbol{m}_{Y \to Z} = \bar{\boldsymbol{\eta}}_{Y,Z}\left(\mathbb{E}_Q\left[\boldsymbol{T}_Y(y)\right], \{\boldsymbol{m}_{Z \to Y}\}_{Z \in cp_{Z,Y}}\right)$

      end

    end

    // Update parameter vector.

    $\boldsymbol{\eta}_Z^{new} = \bar{\boldsymbol{\eta}}_Z\left(\{\boldsymbol{m}_{Y \to Z}\}_{Y \in pa(Z)}\right) + \sum_{Y \in ch(Z)} \boldsymbol{m}_{Y \to Z}$.;

    // Update the expected value of the sufficient statistic.

    $\mathbb{E}_Q\left[\boldsymbol{T}(z)\right] = -\frac{d\bar{\psi}(\boldsymbol{\eta})(\theta)}{d\boldsymbol{\eta}}$.

  end

end

return $\boldsymbol{\eta}_Z^{new}$ $\forall Z \in \boldsymbol{Z}$;

Part VI

COMMONLY STUDIED LATENT VARIABLE MODELS

In this part, three common latent variable models are reviewed. These are *Gaussian mixture*, *latent Dirichlet allocation* and *principal components analysis*.

In *principal components analysis*, the extension of regular *latent variable models* with non-linear functions in the form of *neural networks* is reviewed.

21

GAUSSIAN MIXTURE

One of the most studied models is the Gaussian Mixture we reviewed in Chapter 12, its elements were

- A corresponding set of observations $\boldsymbol{x} = \{x_1, \ldots, x_n\}$.
- The cluster assignment latent variables $\boldsymbol{Z} = \{Z_1, \ldots, Z_N\}$.
- The mixture weights $\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_K\}$, i.e, prior probability of a particular component k.
- Each normal distribution $\mathcal{N}(\mu_k, \Lambda_k)$.

The joint probability factorizes as

$$P(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = P(\boldsymbol{x} \mid \boldsymbol{z}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) P(\boldsymbol{z} \mid \boldsymbol{\pi}) P(\boldsymbol{\pi}) P(\boldsymbol{\mu} \mid \boldsymbol{\Lambda}) P(\boldsymbol{\Lambda}).$$

We are now in situation to give the explicit Bayesian network for this model:

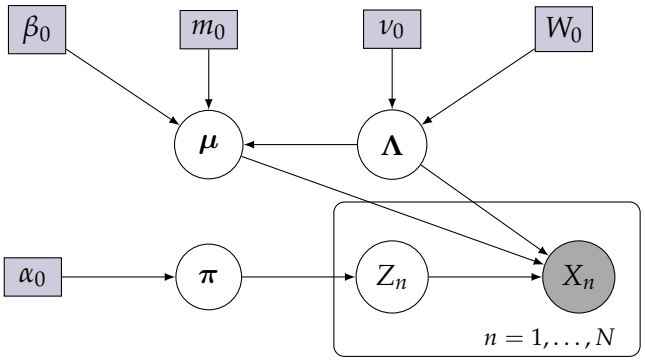


Figure 15: Gaussian mixture model. Squares represent hyper-parameters.

22

LATENT DIRICHLET ALLOCATION

Latent Dirichlet allocation or *LDA* is a conditionally conjugate model (figure 16) in natural language processing that allows set of observations to be explained by unobserved groups that explain why some parts of the data are similar.

For example, observations may be words in a document, which is a mixture of a small number of topics and each word's presence is attributable to one of the document's topics. Learning corresponds to extract information as the set of topics, their associated word probabilities, the topic of each word, and the particular topic mixture of each document.

The considered elements are (using Hoffman *et al.* (2013) and Blei *et al.* (2003) notation):

- K number of topics, V number of words in the vocabulary, M number of documents, N_d number of words in document d and N total number of words.
- $\boldsymbol{\beta} = \{\beta_1, \dots, \beta_K\}$, where β_k is the distribution of words in topic k. Each component $\beta_{k,n}$ is the probability of the n^{th} word in topic k.
- Each document d is associated with a vector of topic proportions θ_d, which is a $K-1$ simplex. Then each component $\theta_{d,k}$ is the probability of topic k in document d. Denote $\boldsymbol{\theta} = \{\theta_1, \dots, \theta_K\}$.
- Each word in each document is assumed to be related with a single topic. The variable $Z_{d,n}$ indexes the topic of the n^{th} word in the d^{th} document.

LDA model assumes that each document is generated with the following generative process:

1. Draw topics from a Dirichlet distribution, for each $k = 1, \dots, K$:

$$P(\beta_k) = \frac{1}{B(\eta)} \prod_{v=1}^{V} \beta_{k,v}^{\eta-1} \implies \beta_k \sim \text{Symmetric-Dirichlet}_V(\eta)$$

2. For each document $d = 1, \dots, D$:
   a) Draw topic proportions,

$$P(\theta_d) = \frac{1}{B(\alpha)} \prod_{k=1}^{K} \theta_{d,k}^{\alpha-1} \implies \theta_d \sim \text{Symmetric-Dirichlet}_K(\alpha)$$

   b) For each word in the document $n = 1, \dots, N_d$:

i. Draw a topic,

$$P(z_{d,n} \mid \theta_d) = \prod_{k=1}^{K} \theta_{d,k}^{\mathbb{I}[z_{d,n}=k]} \implies (Z_{d,n} \mid \theta_d) \sim \text{Categorical}(\theta_d).$$

ii. Draw a word form the topic

$$P(w_{d,n} \mid z_{d,n}, \boldsymbol{\beta}) = \prod_{v=1}^{V} \beta_{z_{d,n},v}^{\mathbb{I}[w_{d,n}=v]} \implies (W_{d,n} \mid Z_{d,n}, \boldsymbol{\beta}) \sim \text{Categorical}(\beta_{Z_{d,n}})$$

The joint probability distribution is then

$$\begin{aligned} P(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{w}, \boldsymbol{\beta}) &= P(\boldsymbol{\beta}) \prod_{d=1}^{D} P(\theta_d \mid \alpha) \prod_{n=1}^{N_d} P(z_{d,n} \mid \theta_d) P(w_{d,n} \mid z_{d,n}, \boldsymbol{\beta}) \\ &= \Big(\prod_{k=1}^{K} P(\beta_k) \Big) \prod_{d=1}^{D} P(\theta_d) \prod_{n=1}^{N_d} P(z_{d,n} \mid \theta_d) P(w_{d,n} \mid z_{d,n}, \boldsymbol{\beta}) \end{aligned}$$

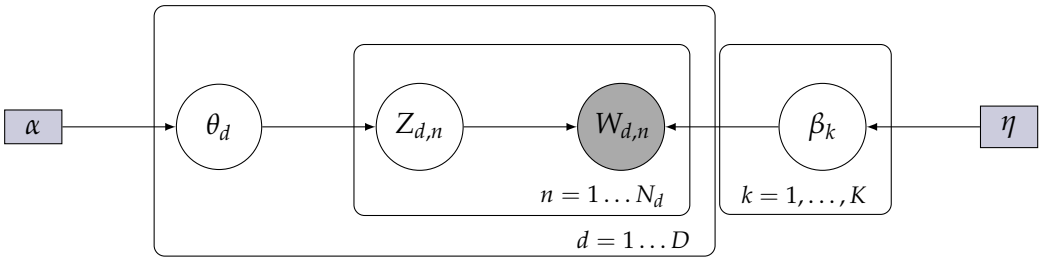


Figure 16: Latent Dirichlet Allocation model. Squares represent hyper-parameters

We can compute the posterior distributions for this model, starting with the complete conditional distribution of the local hidden variables. They only depend on other variables in the local context (same document) and the global variables

$$\begin{aligned} P(z_{d,n} \mid w_{d,n}, \theta_d, \boldsymbol{\beta}) &= \frac{P(z_{d,n}, w_{d,n}, \theta_d, \boldsymbol{\beta})}{\int_{z_{n,d}} P(z_{d,n}, w_{d,n}, \theta_d, \boldsymbol{\beta})} = \frac{P(z_{d,n} \mid \theta_d) P(w_{d,n} \mid z_{d,n}, \boldsymbol{\beta})}{\int_{z_{d,n}} P(z_{d,n} \mid \theta_d) P(w_{d,n} \mid z_{d,n}, \boldsymbol{\beta})} \\ &= \frac{\prod_{k=1}^{K} \theta_{d,k}^{\mathbb{I}[z_{d,n}=k]} \prod_{v=1}^{V} \beta_{z_{d,n},v}^{\mathbb{I}[w_{d,n}=v]}}{\int_{z_{d,n}} \prod_{k=1}^{K} \theta_{d,k}^{\mathbb{I}[z_{d,n}=k]} \prod_{v=1}^{V} \beta_{z_{d,n},v}^{\mathbb{I}[w_{d,n}=v]}} = \frac{\theta_{d,z_{d,n}} \beta_{z_{d,n},w_{d,n}}}{\sum_{k=1}^{K} \theta_{d,k} \beta_{k,w_{d,n}}} \end{aligned}$$

Naming

$$\gamma_{d,n} = \frac{\theta_{d,z_{d,n}} \beta_{z_{d,n} w_{d,n}}}{\sum_{k=1}^{K} \theta_{d,k} \beta_{k,w_{d,n}}}$$

and $\boldsymbol{\gamma} = \{\gamma_{d,n}\}_{d=1,\ldots,D \; n=1,\ldots,N_d}$, we get

$$(Z_{d,n} \mid w_{d,n}, \theta_d, \boldsymbol{\beta}) \sim \mathit{Categorical}(\boldsymbol{\gamma}).$$

The complete conditional of the topic proportions θ_d is only affected by the topic appearances, since $z_{d,n}$ is an indicator vector, the k^{th} element of the parameter to this Dirichlet is the

summation of the hyper-parameter α and the number of words assigned to topic k in document d:

$$P(\theta_d \mid \boldsymbol{z_d}, \boldsymbol{w_d}, \boldsymbol{\beta}) = \frac{P(\theta_d, \boldsymbol{z_d}, \boldsymbol{w_d}, \boldsymbol{\beta})}{\int_{\theta_d} P(\theta_d, \boldsymbol{z_d}, \boldsymbol{w_d}, \boldsymbol{\beta})} = \frac{P(\theta_d) \prod_{n=1}^{N_d} P(z_{d,n} \mid \theta_d)}{\int_{\theta_d} P(\theta_d) \prod_{n=1}^{N_d} P(z_{d,n} \mid \theta_d)}$$
$$\propto \prod_{k=1}^{K} \theta_{d,k}^{\alpha-1} \prod_{n=1}^{N_d} \theta_{d,k}^{\mathbb{I}[z_{d,n}=k]} = \prod_{k=1}^{K} \theta_{d,k}^{\alpha-1+\sum_{n=1}^{N_d} \mathbb{I}[z_{d,n}=k]}.$$

The complete conditional depends only on the topic assignments

$$(\theta_d \mid z_d) \sim \text{Dirichlet}_K\Big(\alpha + \sum_{n=1}^{N_d} \mathbb{I}[z_{d,n} = 1], \ldots, \alpha + \sum_{n=1}^{N_d} \mathbb{I}[z_{d,n} = K]\Big).$$

In words, the probability of a topic in document d is updated with the number of times that topic appears in the document. The complete conditional of a topic depends on the words and topics assignments of the entire collection

Using a similar reasoning, the words distribution in a topic k, β_k, is updated with the number of appearances in all documents of the given topic.

$$(\beta_k \mid \boldsymbol{w}, \boldsymbol{z}) \sim \text{Dirichlet}_V\Big(\eta + \sum_{d=1}^{D} \sum_{n=1}^{N_d} \mathbb{I}[z_{d,n} = k]\mathbb{I}[w_{d,n} = 1], \ldots, \eta + \sum_{d=1}^{D} \sum_{n=1}^{N_d} \mathbb{I}[z_{d,n} = k]\mathbb{I}[w_{d,n} = V]\Big).$$

23

PROBABILISTIC PRINCIPAL COMPONENTS ANALYSIS

LVMs are usually applied to the exponential family because inference is feasible in this case. Neural network have recent enabled LVMs to be extended outside the exponential family, for example, *variational auto-encoders* or *VAEs* (Kingma & Welling (2013)) are the most influential models combining both concepts. They extent the classical *principal components analysis* (PCA Pearson (1901)) technique for reduction. Suppose we have a D-dimensional representation of a data point x and z is its latent K-dimensional representation ($K < D$). PCA computes an affine transformation $\boldsymbol{W}$, represented by a $K \times D$ matrix.

Whereas the PCA reduction is usually computed using either the singular value decomposition or the eigenvalue decomposition of the covariance matrix of the original data, a probabilistic view of PCA can be modeled with an LVM (Tipping & Bishop (1999)). The following elements are considered:

- $\boldsymbol{X} = \{X_1, \dots, X_N\}$ i.i.d $\mathbb{R}^D$-valued random variables and the corresponding observations $\boldsymbol{x} = \{x_1, \dots, x_N\}$.
- $\boldsymbol{Z} = \{Z_1, \dots, Z_N\}$ i.i.d latent $\mathbb{R}^K$-valued random variables, where Z_n models the K-dimensional representation of x_n.
- A global latent $K \times D$-dimensional random variable $\boldsymbol{W}$, which models the transformation between the spaces.
- A noise hyper-parameter σ^2.

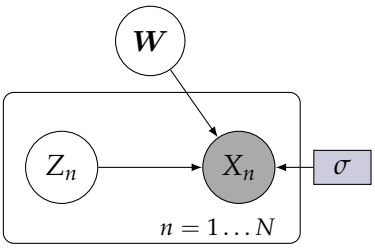


Figure 17: Probabilistic PCA model

We assume the priors are normally distributed:

$$Z_n \sim \mathcal{N}_K(0, I) \quad \forall n = 1, \dots, N \quad \text{and} \quad \boldsymbol{W} \sim \mathcal{N}_{K \times D}(0, I).$$

The data points are considered to be generated via a projection to the higher dimensional space,

$$X_n \mid z_n, \boldsymbol{w} \sim \mathcal{N}_D(\boldsymbol{w}^T z_n, \sigma^2 I) \quad \forall n = 1, \dots, N.$$

The probabilistic model extends the classical one in the way that the latter assumes the noise is infinitesimally small, i.e, $\sigma^2 \to 0$. The *expectation-maximization algorithm* (Chapter 9) is commonly used to solve this variational inference problem.

There exists other approaches for the PCA model, for example, another variable δ might be considered in order to produce non-centered points, that is,

$$\delta \sim \mathcal{N}_D(0, I) \quad \text{and} \quad X_n \mid z_n, \boldsymbol{w}, \delta \sim \mathcal{N}(\boldsymbol{w}^T z_n + \delta, \sigma^2 I) \quad \forall n = 1, \ldots, N.$$

23.1 ARTIFICIAL NEURAL NETWORKS

An *artificial neural network* or *ANN* with L hidden layers can be defined as a deterministic non-linear function f parameterized by a set of matrices $\boldsymbol{W} = \{\boldsymbol{W}_0, \ldots, \boldsymbol{W}_L\}$ and non-linear activation functions $\{r_0, \ldots, r_L\}$. Given an input x the output y of the network is calculated has

$$h_0 = r_0(\boldsymbol{W}_0^T x), \quad \ldots \quad h_l = r_l(\boldsymbol{W}_l^T h_{l-1}), \quad \ldots \quad y = r_L(\boldsymbol{W}_L^T h_{L-1}).$$

Deep neural networks or *DNNs* are ANNs where the number of hidden layers is considered high. Commonly, any neural network with more that 2 hidden layers is said deep. Given a dataset (set of inputs and their corresponding outputs) $\{(x_1, y_1), \ldots, (x_N, y_N)\}$ and a loss function $l(y, y^*)$ that defines how well the output $y^* = f_{\boldsymbol{W}}(x)$ returned by the network matches the real output y, learning reduces to the optimization problem

$$\boldsymbol{W}^{opt} = \arg\min_{\boldsymbol{W}} \sum_{n=1}^{N} l(y_n, f_{\boldsymbol{W}}(x_n)).$$

This problem is usually solved by applying a variant of the *stochastic gradient descent method* (Kiefer *et al.* (1952)), which is an *stochastic approximation* (Robbins (2007)) of the *gradient descend* (Cauchy (1847)) technique. *Gradient descent* involves the computation of the loss function's gradient with respect to the network's parameter. Let $\boldsymbol{W}_t$ be the set of parameters in the iteration t^{th} and f the loss function, seen as a function of the parameters. Then, the parameter update is defined as

$$\boldsymbol{W}_{t+1} = \boldsymbol{W}_t + \gamma \nabla f(\boldsymbol{W}_t).$$

Where $\gamma \in \mathbb{R}^+$ is a small value. This technique uses the fact that $-\nabla f(\boldsymbol{W})$ points to the local minimum nearest to$\boldsymbol{W}$. This iterative method guarantees convergence to this local minimum, and, in case f is convex, it converges to the global minimum.

As the loss function is typically unknown, the gradient estimated using the given dataset. In comparison, *stochastic gradient descent* computes an approximation of the gradient from a randomly selected subset of the data.

The algorithm for computing this gradient is known as *back-propagation* (Goodfellow *et al.* (2016)), which is based on a recursive application of the chain-rule of derivatives. This can be implemented using the computational graph on the network.

The main idea of a computational graph is to express a deterministic function, as is the case of a neural network, using an acyclic directed graph. It is composed of input, output and operation nodes, where model data and parameters are shown as input nodes.

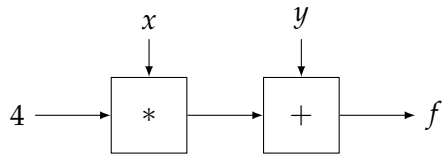


Figure 18: Computational graph example of function $f(x, y) = 4x + y$

The input nodes are typically tensors (n-dimensional arrays), therefore, the operations are defined over tensors. The main point of using computational graphs is that they enable *automatic differentiation* (Griewank *et al.* (1989)), a technique for computing all partial derivatives of the function.

23.2 NON-LINEAR PCA

Non-linear PCA or NLPCA, extends the classical PCA where the relation between the low dimensional space and the observed data is governed by a DNN instead of a linear transformation. It can be seen as a non-linear probabilistic PCA model.

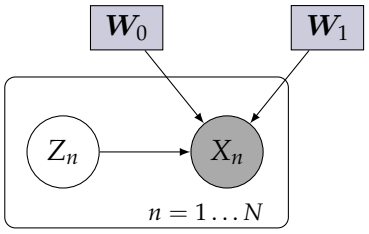


Figure 19: Non-linear PCA model. $\boldsymbol{W}_0$ and $\boldsymbol{W}_1$ together with an activation function r represent an ANN.

The model is quite similar to the one presented for the PCA, the difference comes from the conditional distribution of X, that depends on Z through a fully-connected ANN with a single hidden layer.

As in the case of the PCA, the prior of the latent variable follows a centered Gaussian

$$Z_n \sim \mathcal{N}_K(0, I) \quad \forall n \in 1, \dots, N.$$

Let D be the dimension of the data X and K the dimension of the hidden variable Z. Let f a single hidden layer ANN with input dimension Z and output dimension D. Where the output is the mean value of the normal distribution under X.

As we are considering a single hidden layer, let $\boldsymbol{W}_0$ and $\boldsymbol{W}_1$ be the matrices governing that ANN and r the activation function, the ANN f is

$$f(z_n) = \boldsymbol{W_1}(r(\boldsymbol{W}_0(z_n)))),$$

and the data-points are then generated as

$$X_n \mid Z_n \sim \mathcal{N}_D(f(Z_n), I) \quad \forall n \in 1, \dots, N.$$

Where no noise is being considered this time.

23.3 VARIATIONAL AUTO-ENCODER

Variational auto-encoders or VAEs are a dimensionality reduction model, like PCA and NLPCA. In contrast with this two, this auto-encoders contain two neural networks, one in the P model (decoder) and another one in the variational model Q (encoder).

On one hand, the P model has the same structure as the non-linear PCA, and, on the other hand, the distribution Q is defined with a reverse ANN, with input dimension D and output dimension K.

The neural networks can be defined to give an output of double the dimension, that is, the encoder would give a point in with $2K$ components, so the first K are used for the mean and the last K for the variance of the normal distribution.

In conclusion, the parametric model assumes the data is generated as:

1. Get a sample from a standard K-dimensional Gaussian distribution.

$$Z \sim \mathcal{N}_K(0, I).$$

2. The data-point is generated from a Gaussian distribution with mean the image of the neural network on the previous sample. Let f denote the neural network:

$$X \mid z \sim \mathcal{N}_D(f(z), I).$$

Conversely, the variational model supposes the hidden values are generated from the observed ones as:

1. Let x be a sample from a standard Gaussian distribution.

$$X \sim \mathcal{N}_D(0, I).$$

2. Let g denote the neural network from the D dimensional space to a $2K$ dimensional one. Let μ be the first K components of $g(x)$ and σ the last ones.

$$g(x) = (\mu, \sigma).$$

3. The hidden representation of x is generated as

$$Z \mid x \sim \mathcal{N}_K(\mu, \texttt{softplus}(\sigma)).$$

Where, as the standard deviation must have a positive value, we use a `softplus` function to smoothly approximate a rectifier function ($\text{ReLU}(x) = \max(x, 0)$), this function is defined as

$$\texttt{softplus}(x) = \log(1 + e^x).$$

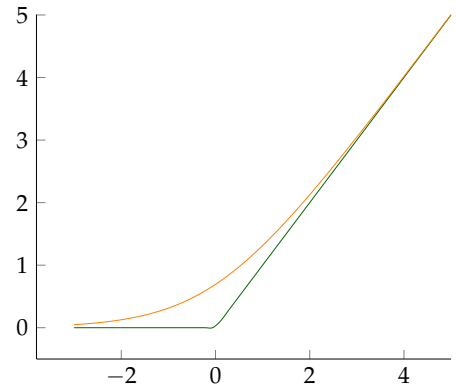


Figure 20: ReLU (green) and softplus (orange) comparison in $[-3, 5]$.

Figure 21 shows a reduction from a 6-dimensional space to a 2-dimensional space. There, each x_n on the left symbolizes the value on each dimension, a_i symbolize weights of the used neural networks (despite using the same symbols, networks are different) and the activation functions are not represented.

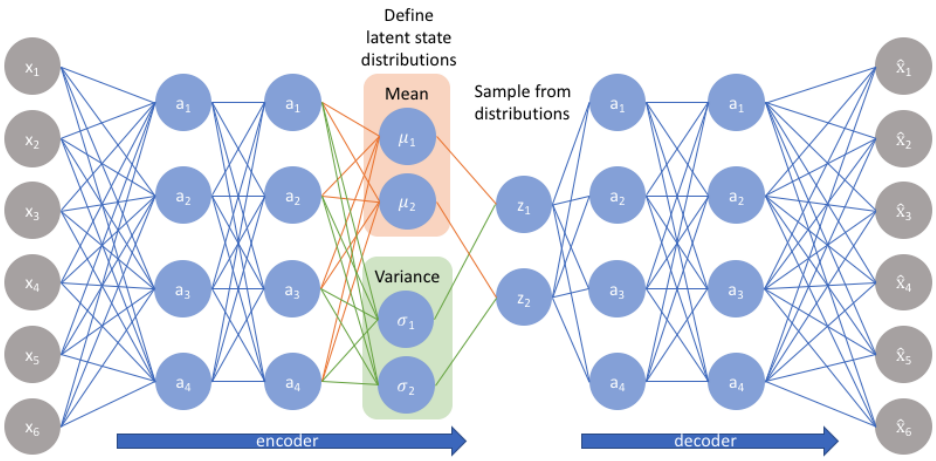


Figure 21: Variational auto-encoder structure. Image taken from Jordan (2018).

Part VII

CASE STUDY

Three different `python` frameworks are reviewed: `InferPy`, `BayesPy` and `Scikit-Learn`. The main objective is to test each framework's features and limitations, in order to do so, both Gaussian mixtures and dimensionality reduction models are used.

24

USED FRAMEWORKS

Three different frameworks are used during this study, which are `InferPy`, `BayesPy` and `Scikit-Learn`. The first two are specifically designed to probabilistic modeling whereas the last one is a general machine learning library. For this reason, we are briefly describing the usage of the first two and only explaining the functions we are using on the last, but, before briefly describing their usage, the study's file structure and used database are described.

Each study is made in a different file, available in both as a `Jupyter Notebook (.ipynb)` and `Python script (.py)` using the name format `[model]_[framework]_[database]`. Apart from these, there are three main auxiliary files:

- `packages`: This file contains a list of the installed packages. These packages might be installed using `pip install -r packages`.
- `models.py`: In this `Python` file, all parametric and variational models from `InferPy` are defined. This includes PCA, NLPCA, VAE and Gaussian mixture.
- `functions.py`: In this file, all auxiliary functions are defined. The aim of these functions is either to make graphic representations or summarize the inference results.

Two different databases are used:

- `Mnist`: a largely used dataset on a set of handwritten digits.
- `Breast Cancer Wisconsin`: characteristics of the cell nuclei present in breast mass.

Mnist

`Mnist` database (LeCun & Cortes (2010)) can be directly obtained through one of the used frameworks, `InferPy.data`. It consists of a set of 70.000 handwritten digits, codified as 28×28 matrices where each position is the gray-scale value of the digit, in $[0, 255]$ (Figure 22 shows an example of digits within the database). Given this, the observed data belongs to $[0, 255]^{784}$. Each sample is labeled with its corresponding digit.

Due to the size of the database, 1000 samples of digits 1, 4 and 7 are used. These are loaded form `InferPy` directly:

```
mnist.load_data(num_instances=1000, digits=[1,4,7])
```

Function `print_class_pie_diagram` draws a class pie diagram of the given label set, in this case, their proportions are almost equal (Figure 23).

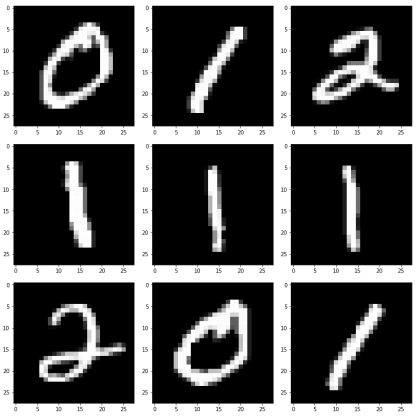


Figure 22: Mnist dataset example.

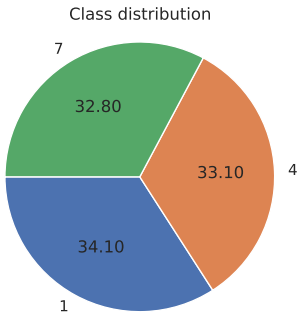


Figure 23: Mnist class proportion between numbers 1, 4 and 7.

Breast cancer Wisconsin

`Breast cancer` database can be obtained from its `UCI` (Dua & Graff (2017)) repository link. The database consists of 569 instances of 30 features of the cell nuclei present in breast mass. Among with each instance, an ID number and the diagnosis (M = malign and B = benign) are given. The given attributes are mean, standard error and worst (largest) value of all cells of the following features:

- Radius: mean distance from the center to the perimeter.
- Texture: standard deviation of gray-scale values.
- Perimeter.
- Area.
- Smoothness: local variation in radius length.
- Compactness: $perimeter^2/area - 1$.
- Concavity: severity of concave portions of the contour.
- Concave points: number of concave portions of the contour.
- Symmetry.
- Fractal dimension: a ratio comparing how detail in a fractal pattern changes with the scale of measurement.

All these features are coded with four significant digits.

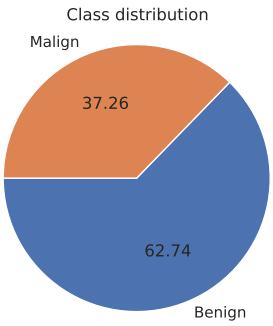


Figure 24: Breast cancer Wisconsin proportion of malign and benign labels.

In this database the class proportion is not balanced at all, Figure 24 shows a dominance of benign entries over the malign ones. Either way, this should not affect the obtained results.

24.1 INFERPY

`InferPy` (Cózar *et al.* (2019)) is a high-level API written in `Python` inspired by `Keras` and run on top of `Tensorflow` and `Edward` for probabilistic modeling. It is focused on enabling probabilistic modeling, flexible data processing and scalable inference.

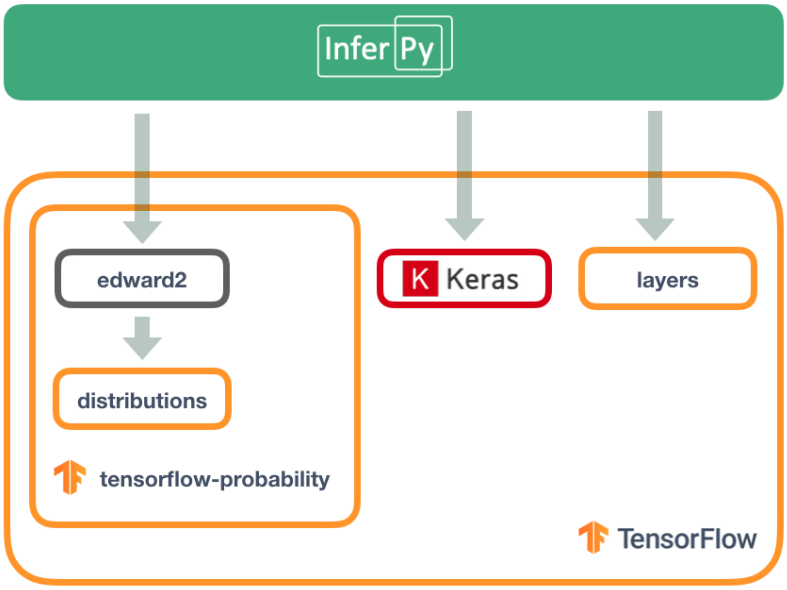


Figure 25: InferPy architecture (Cózar *et al.* (2019)).

`InferPy`'s main features are:

- Allows to define probabilistic models whether they contain or not neural networks in a simple way.
- All models that can be defined using `Edward2` can also be defined in `InferPy`, whose probability distributions are mainly inherited from `tensorflow-probability`.

- All the models parameters can be defined using the standard `Python` types (`Numpy` and `TensorFlow` compatibility).
- `InferPy` relies on top of `Edward`'s inference engine, therefore, it includes all the inference algorithms available in that package.
- As it uses `Edward` and `TensorFlow` as its inference engine, all the parallelization details are hidden to the user.
- The same code will run either in *CPUs* or *GPUs*.

Installation

`InferPy` has the following package requirements:

- `Python` $\geq$ 3.5 and $<$ 3.8.
- `Tensorflow` $\geq$ 1.12.1 and $<$ 2.0.
- `Tensorflow-probability` 0.7.0.
- `NetworkX` $\geq$ 2.2.0 and $<$ 3.0.

`InferPy` is available at `Pip` and can be installed with the following command:

```
pip install inferpy
```

Usage guide

In all the following examples `InferPy` is imported as `inf`.

`Inferpy` requires that both the parametric P and the variational Q distribution are defined. In order to do so, models must be defined as `Python` functions inside a `@inf.probmodel` macro.

```
@inf.probmodel
def model():
```

Inside the model definition, both visible and hidden variables must be defined using the following syntax:

```
x =  inf.Normal(loc=tf.zeros([2]), scale=1, name="x")
```

Distributions are imported from the package itself, as the case for `inf.Normal`. Distribution parameters must be passed during the definition, in this case, the mean (`loc`) and the standard deviation (`scale`) are used.

All variables must be given a name, variables with the same name in both the parametric and the variational model are considered the same variable. This is how both models communicate during the learning task. In the variational model, there must exist a parameter for each variable's parameter, in the case of the previous variable:

```
loc = inf.Parameter(tf.zeros([2]), name="x_loc")
scale = tf.math.softplus(inf.Parameter(tf.ones([2]), name="x_scale"))
x =  inf.Normal(loc, scale, name="x")
```

Where a softplus function is used to avoid non-positive values.

If either the mean or the standard deviation are given as an array, a constant value on the other parameter would be interpreted as a constant array of the corresponding size. This is the case of the last definition where

$$X \sim \mathcal{N}_2(0, I).$$

`InferPy` gives an explicit syntax for declaring a set of i.i.d random variables. Variables defined inside `with inf.datamodel(size)` are replicated the indicated number of times. This size can be omitted in the case of observations as it is calculated from the dataset at learning time.

```
with inf.datamodel():
    x = inf.Normal(tf.zeros([2]),1, name="x")
```

When both models are defined and instantiated in a variable, we need to create an inference object, using a variational model as argument:

```
VI = inf.inference.VI(variational_model)
```

By default, the inference is made using the machine learning technique *gradient descent* which uses $-$ELBO as loss function (source code), notice that the ELBO takes values in $(-\infty, 1]$, and uses `AdamOptimizer` (Kingma & Ba (2014)) from `TensorFlow`. A number of `epochs` must be establish, which set the stop criteria to a number of iterations to be done.

Given a training dataset `X`, the model is trained indicating which set of observations corresponds to each observed variable using the following syntax:

```
model.fit({"x": X}, VI)
```

Once the model is trained, a variable posterior can be taken as

```
model.posterior("z").parameters()
```

24.2 BAYESPY

`BayesPy` (Luttinen (2014)) is a Bayesian inference `Python` package for modeling Bayesian networks and make inference. Currently, it only supports conjugate models in the exponential family as it only implements *variational message passing algorithm* in contrast with `InferPy` that uses *gradient descent*.

Instalation

The package can be installed using `pip install bayespy` and has the following requirements:

- `NumPy` $\geq$ 1.10.
- `SciPy` $\geq$ 0.13.0.
- `matplotlib` $\geq$ 1.2.
- `h5py`.

Usage guide

Three steps are needed to make Bayesian inference in `BayesPy`, firstly construct the model, secondly observe the data and lastly run the inference method.

Distributions are available at `BayesPy.nodes` and these nodes must be defined on `Python` objects. For example the following syntax

```
mu = nodes.Gaussian(np.zeros(2), np.identity(2), plates=(10))
```

creates 10 i.i.d variables $X_1, \ldots, X_{10}$ such that

$$X_n \sim \mathcal{N}_2(0, I) \quad \forall n = 1, \ldots, 10.$$

Notice that the plate notation is done using a single parameter `plates`, in contrast with `InferPy` that used the syntax `with inf.datamodel()`.

When all nodes are created, the data must be observed using each node's `.observe()` method. Using the node's observations as argument.

The only inference method (VMP) is available at `bayespy.inference.VB` whose arguments must be all probabilistic nodes in the model, for example,

```
Q = VB(x, z, mu, Lambda, pi)
```

creates the inference object of the Gaussian mixture model we studied. The inference method initializes the nodes variational prior automatically, either way, the user can decide how to initialize these values calling any of the following methods on the node itself:

- `initialize_from_prior`: Uses the parent nodes to initialize the node. This is by default.
- `initialize_from_parameters`: Use the parameters given in the argument to initialize.
- `initialize_from_value`: Use the value given in the argument to initialize.
- `initialize_from_random`: Takes a random value for the variable.

The inference task is started using the method `.update()` from the `VB` object. By default the nodes are updated using the same order in they where passed when creating the object, to change that, a new order can be given as a argument.

```
Q.update(z, mu, Lambda, pi).
```

Another possible arguments are the number of iterations (`repeat`) and the relative tolerance, i.e, distance of the ELBO between iterations (`tol`). After each iteration the value of the ELBO (`loglike`) is displayed.

24.3 SCIKIT-LEARN

`Scikit-Learn` (Pedregosa *et al.* (2011)) is an open source library focused on providing machine learning tools, which include model fitting, selection and evaluation. From this library, we are using the `BayesianGaussianMixture` class which provides variational estimation of a Gaussian mixture model using the EM algorithm. In the implementation of the algorithm, the variational distribution family is fixed as in Chapter 12, for this reason, the **E-step**, which optimizes the variational distribution, only computes auxiliary values. Meanwhile, the

M-step does optimize the parameters (hyper-parameters in this case), that is, the parameters of $\boldsymbol{\mu}, \boldsymbol{\Lambda}, \boldsymbol{z}$ and $\boldsymbol{\pi}$. This updates can be found in the class source code, and are equal to the ones we computed for the CAVI algorithm in Chapter 12.

As this package is not specifically designed to make Bayesian inference, no nodes need to be defined as the model is internally handled. This being said, `BayesianGaussianMixture` models all decisions via its parameters:

- `n_components`: total amount of mixture components, default value is 1.
- `covariance_type`: describes the type of covariance to use in the model. `full` means that each component has its own covariance matrix, `tied` means that all components have the same matrix, `diag` means that each component has its own covariance matrix but it must be diagonal and `spherical` means that each component has its own single variance value, i.e, diagonal matrix with the same value. The default value is `full`.
- `tol`: Tolerance threshold, 0.001 by default.
- `max_iter`: Number of iterations to make. By default 100.
- `init_params`: Handles the weights initialization, where `kmeans` and `random` are possible. By default K-means is used.
- `weight_concentration_prior_type`: Describes how the weights prior is modeled. `dirichlet_process` or `dirichlet_distribution`. Where a Dirichlet process is an infinite-dimensional generalization of the Dirichlet distribution, used for infinite Gaussian Mixtures. This is used by default.
- `weight_concentration_prior`: Real value corresponding to each component weight value. By default equals $1/n\_components$.
- `mean_precision_prior`: The precision prior of the mean distribution, by default 1.
- `mean_prior`: The mean prior of the mean distribution, by default the mean of the dataset.
- `degrees_of_freedom_prior`: Degrees of freedom of the Wishart distribution, by default the ammount of features of the data.
- `covariance_prior`: Prior matrix of the Wishart distribution equals. By default it is initialized to the empirical covariance matrix.

The inference task is as simple as creating the Gaussian mixture object:

```
gm = BayesianGaussianMixture()
```

Use the dataset to fit the model:

```
gm.fit(X)
```

Once done, the posterior parameters can be inspected using the attributes `weights_`, `means_` and `precisions_`. As many models in this package, the model can be used to predict a new input using `gm.predict(X)`. The probability of belonging to each component can be also computed using `gm.predict_proba(X)`.

25

DIMENSIONALITY REDUCTION

In this section, the results obtained from using `InferPy` to achieve dimensionality reduction of the two given databases to a two-dimensional and a three-dimensional space are reviewed.

25.1 DEFINED MODELS

The PCA model is defined with the following elements:

- The set of i.i.d observed variables $X_1, \ldots, X_N$.
- The hidden representation of each variable $Z_1, \ldots, Z_N$.
- A global hidden variable $\boldsymbol{W}$ the linear transformation from one space to the other.
- Another global variable δ that will allow the model to generate non-centered points.

The latent variables prior is a centered Gaussian and the data is supposed to be generated as

$$X_n \mid z_n, \boldsymbol{w}, \delta \sim \mathcal{N}(\boldsymbol{w}^T z_n + \delta, I)$$

The considered noise value is therefore $\sigma^2 = 1$.

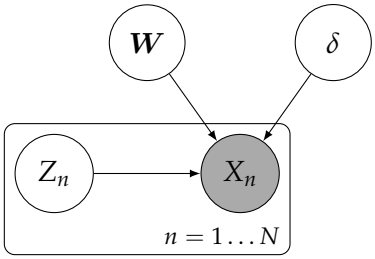
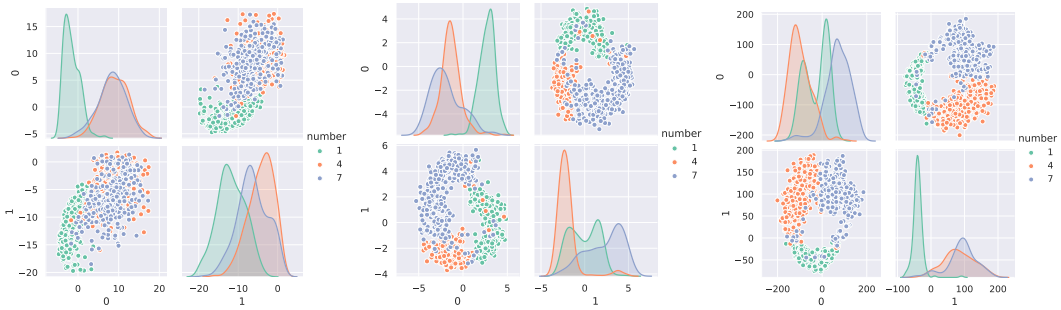


Figure 26: Non-centered Probabilistic PCA model. No noise considered.

This model is defined in `inferPy` as:

```
@inf.probmodel
def pca(hidden_dim, observed_dim):
    w = inf.Normal(loc=tf.zeros([hidden_dim, observed_dim]), scale=1, name="w")
    delta = inf.Normal(loc=tf.zeros([observed_dim]), scale=1, name="delta")
    with inf.datamodel():
        z = inf.Normal(tf.zeros([hidden_dim]), 1, name="z")
        x = inf.Normal(z @ w + w0, 1, name="x")
```



(a) (b) (c)

Figure 27: PCA, NLPCA and VAE posterior samples (from left to right) from Mnist 2D reduction.

The non-linear PCA model is defined using a two-layer neural network, to achieve this, four variables are defined $\alpha_0, \alpha_1, \beta_0$ and β_1. All of them following a Gaussian distribution $\mathcal{N}(0, I)$. The networks output is then computed as

```
h = tf.nn.relu(z @ beta0 + alpha0)
return h @ beta1 + alpha1
```

This is done to show how `InferPy` allows networks to be modeled using variables. In the other hand, the variational auto-encoder model is defined using `Keras` integration with `InferPy`, so that both networks are defined as an `inf.layers.Sequential` object. In the case of the generative model, the hidden data is supposed to follow a standard Gaussian distribution $\mathcal{N}(0, I)$ and the observed data is generated using the output of a 2-totally connected layers network. The medium dimension is set to 100.

```
inf.layers.Sequential(
   [tf.keras.layers.Dense(100, activation=tf.nn.relu),
   tf.keras.layers.Dense(observed_dim),]
)
```

Only one activation function is used because the output works as the mean value of the observed data distribution. This network is called the "decoder" as it transforms the hidden representation into the observed one. The "encoder" network (modeled inside the variational distribution), uses a double dimension output, as described in Section 23.3.

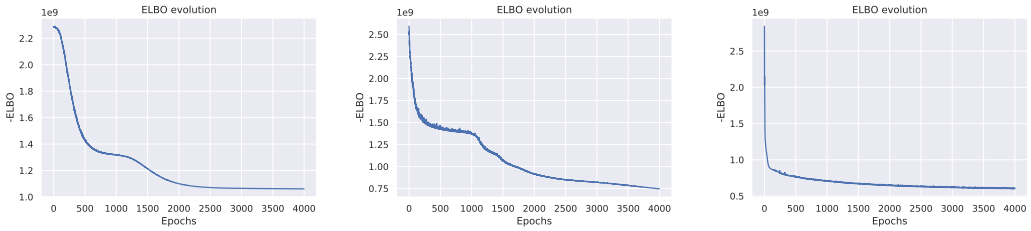
25.2 RESULTS

Mnist

Samples can be generated from the posterior of a latent variable Z using the input data:

```
sample = model.posterior("z", data={"x": X}).sample()
```

The function `print_posterior` might be used to draw 2D projections of a posterior sample. Results from the 2D reduction are shown in Figure 27. In each of the graphics in that figure, projections to each of the learned latent variables are shown, leaving the diagonals to show the density of data-points in the corresponding variable.



(a) (b) (c)

Figure 28: PCA, NLPCA and VAE loss function evolution (from left to right) from Mnist 2D reduction.

Model	2D reduction	3D reduction
PCA	0.737	0.916
NLPCA	0.947	0.969
VAE	0.948	0.965
Original data	0.998	

Figure 29: SVM score.

For example in the first image, shown Figure 27 (a), one may see that the first variable (0) strongly separates samples corresponding to 1 from the other samples, whereas the second variable (1) does not make such distinction.

The generated representation of the NLPCA and VAE result in a similar ring form, where data samples seem separated between classes. One difference between both results is that the ring scale in the VAE case is bigger (200 compared to 5).

In the 3 cases the loss function (-ELBO) does decrease strongly during the first iteration and slowly during the last ones. This is accentuated in the VAE case (last plot).

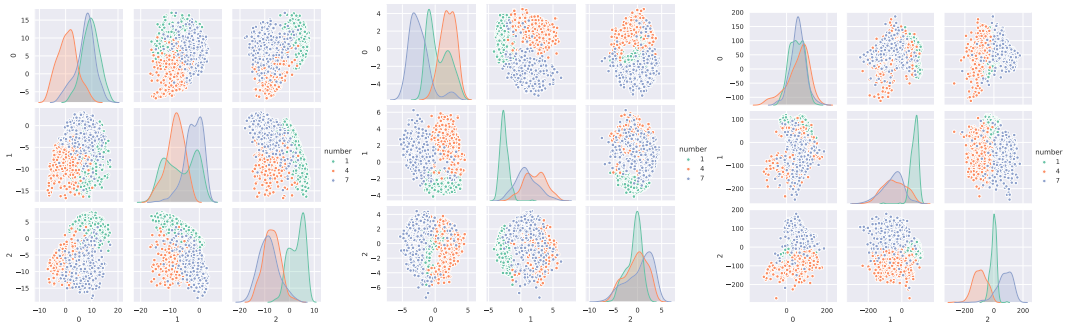
In order to test the quality of the reduction, the fact that a good reduction must preserve class separability is used. To test this, the function `test_separability` trains a `Support Vector Machine` from `Scikit-learn` over both the observed and the reduced space, displaying the score obtained in each one of these. Table 29 shows the obtained results.

The three dimensional reduction results are shown in Figure 30.

Breast Cancer Wisconsin

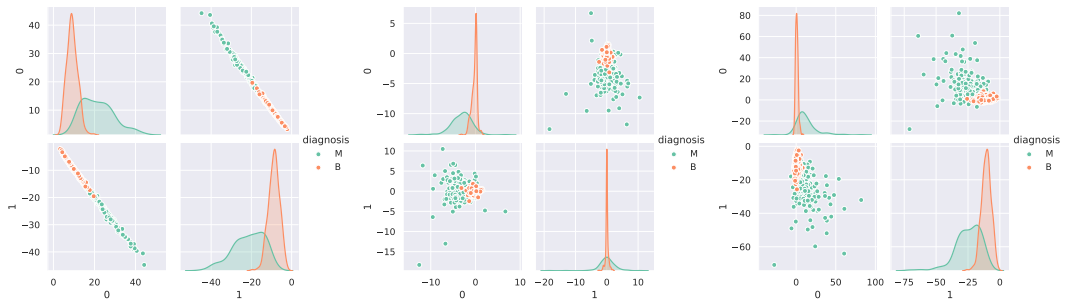
Firstly, as the given identifier (`id`) is a non-predictive variable, it must be dropped from the dataset. Due to the relatively small size to the database, all entries are used in this case.

The obtained results from the two-dimensional (Figure 31) and the three-dimensional (Figure 33) are quite similar, where the score table (Table 32) shows very similar results in all cases. However, there are some details worth mentioning that make a difference with the obtained results in `Mnist`.



(a) (b) (c)

Figure 30: PCA, NLPCA and VAE posterior samples (from left to right) from Mnist 2D reduction.

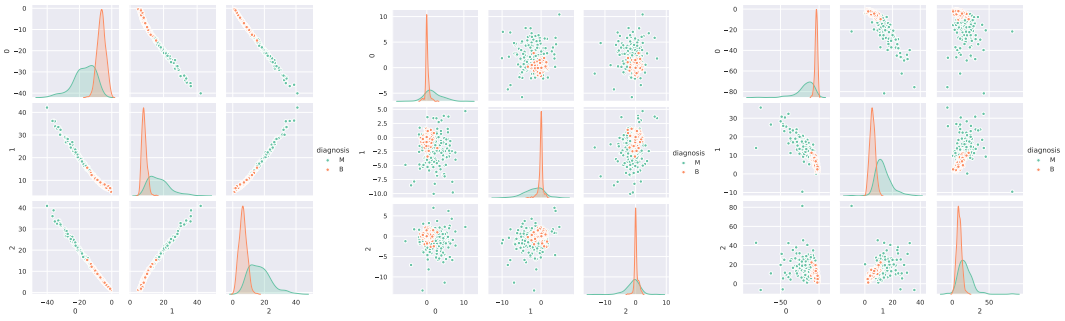


(a) (b) (c)

Figure 31: PCA, NLPCA and VAE posterior samples (from left to right) from Breast Cancer 2D reduction.

Model	2D reduction	3D reduction
PCA	0.9068	0.9103
NLPCA	0.9121	0.9121
VAE	0.9349	0.9209
Original data	0.9226	

Figure 32: SVM score.



(a) (b) (c)

Figure 33: PCA, NLPCA and VAE posterior samples (from left to right) from Breast Cancer 3D reduction.

First of all, the results obtained from all models in this database show a highly sharp distribution on the benign entries and a more extended one on the malign one. Secondly, in both reductions, the probabilistic PCA model made a reduction where all the data-points resulted aligned, as a result, these two or three variables are highly correlated, which is not a desired outcome.

In this database, the obtained loss functions present greater oscillations compared to `Mnist`, which means that the learning task has encountered more local minimum compared to the other database.

It is worth mentioning that the separability has being increased (from 0.9226 to 0.9349) after reducing to a two-dimensional space using a variational auto-encoder. This ensures that the given reduction does preserve the properties of the database.

There is no big difference between using a two or three dimension in this dataset.

26

GAUSSIAN MIXTURE

In this section, the obtained results from a Gaussian mixture training are reviewed. Two out of three frameworks are succesfully tested in this model.

In both frameworks `Breast Cancer` database is used with the main goal of learning a Gaussian mixture model, the original database and the one obtained via a dimensionality reduction. The latter is done using a variational auto-encoder with `InferPy`, showing that frameworks can be used together.

26.1 INFERPY

Firstly, the Gaussian mixture cannot be directly modeled using `InferPy` due to the fact that `TensorFlow -Probability` name categorical variables as `NOT_REPARAMETERIZED`, which, citing its documentation means that "samples from the distribution are not fully reparameterized, and straight-through gradients are either partially unsupported or are not supported at all". Unsuccessful attempts have being made using `MixtureGaussian` distribution available in `InferPy`, which encapsulates the mixture without needed explicit categorical variables. In this attempts, parameters that must remain positive reach negatives values even using a `softplus` filter, which makes inference not possible.

Adding a constant term to those parameters, such as, 0.1 in an attempt to keep them positive resulted in no learning errors but also no real results, where even the component weight parameter remained nearby its prior value on Breast Cancer database (where it should tend to $[0.63,\ 0.37]$).

26.2 SCIKIT-LEARN

Because of the high number of parameters, models are defined with `covariance_type = "diag"`, which means that all covariance matrix are diagonal. Using `"full"` covariance matrix would mean that two 30×30 matrices might be learned from only 569 datapoints. Doing this reduction, the amount of covariance parameters is reduced to $2 \times 30 = 60$.

One of the parameters that can be easily studied when dealing with 30 parameters is the weight of each component:

```
print(gm.weights_)
```

Which resulted in [0.31003524 0.68996476] after the training and is more accurate than the prior value (0.5 each).

`Scikit-Learn` allows to compute the posterior probability of belonging to each component by the usage of `gm.predict_proba(points)`. This might be applied to each class, showing how it is distributed over the components, an ideal result would be that each of the classes is totally governed by each component. Function `plot_mixture_distplot` computes the posterior probabilities for each class, showing a density function fitting this results. Figure 34 shows this density functions over the original dataset.

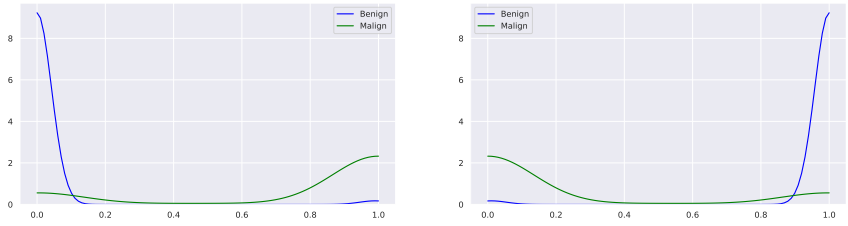


Figure 34: Component belonging density.

One may see that benign cases are almost completely modeled by the second component whereas the malign ones are more distributed but still modeled by the first component.

The dataset is now reduced to a three-dimensional representation using a variational auto-encoder, the dimensionality of the middle layer is 100 and 4000 epochs are used. The obtained reduction is shown in Figure 36. The obtained component belonging densities are shown in Figure 35 on the reduced space, showing a sharper density on the benign class compared to the original data, meaning that this class is better represented by the component in the reduced space.

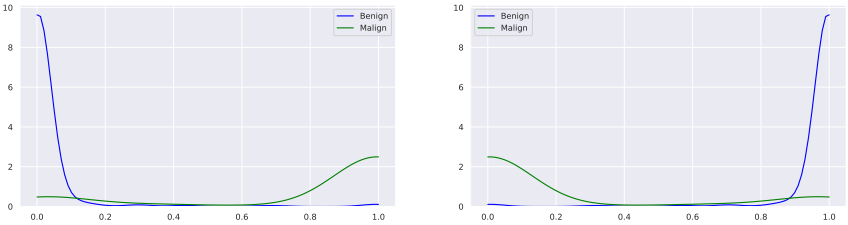


Figure 35: Component belonging density in reduced data.

As the representation is three-dimensional it is easy to compare other parameters as the mean value of each component, these are [16.15134999, −3.88342216, −18.26021824] and [7.25244497, 1.90782194, −5.72329033]. Using this values and Figure 36, one may notice that the first component seeks to model the malign points and the second the benign ones.

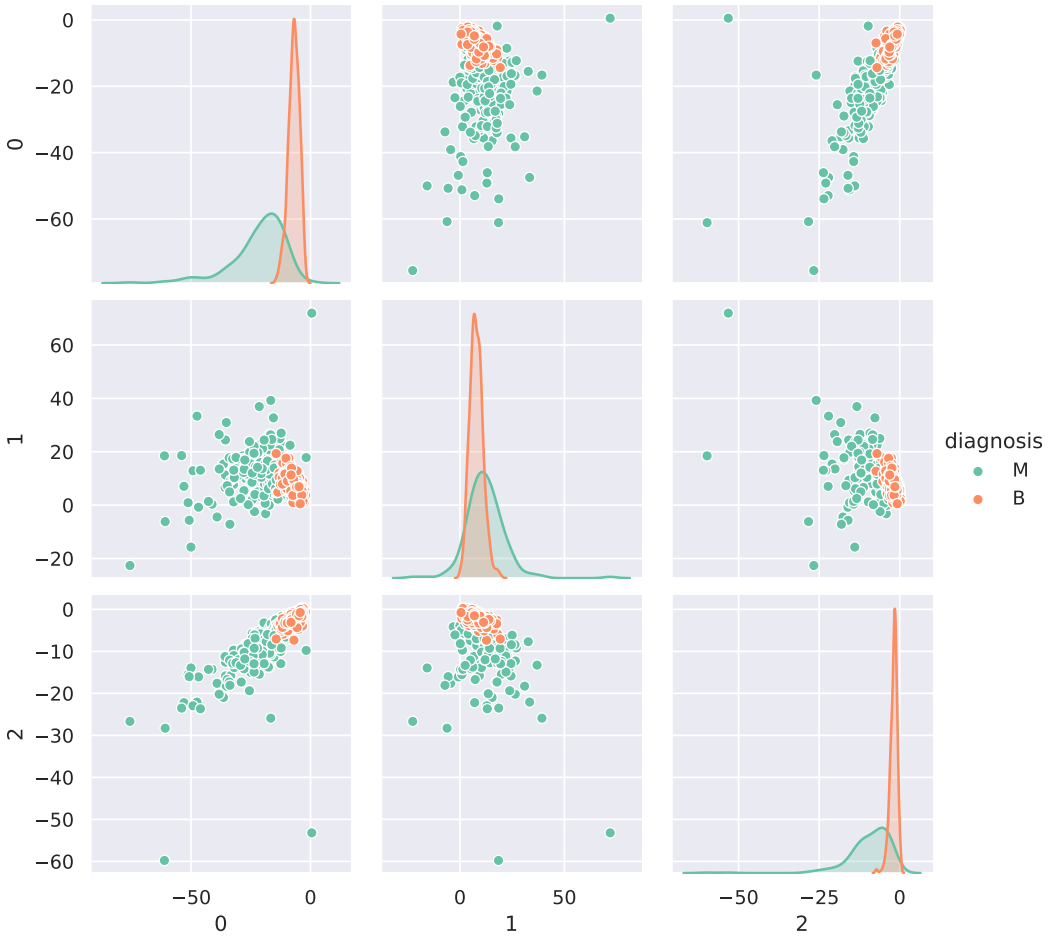


Figure 36: Breast Cancer two-dimensional reduction representation. Reduction made trough a variational auto-encoder.

26.3 BAYESPY

The `BayesPy` model can be easily created using the syntax explained before, the model parameters are initialized as

- The means variable μ follows a centered normal distribution:

$$\mu_k \sim \mathcal{N}_{data\_dim}(0, I).$$

- The precision variables Λ follow a Wishart distribution with parameters:

$$\Lambda_k \sim \mathcal{W}_{data\_dim}(data\_dim, I).$$

- The weights concentration variable follows a Dirichlet with parameter:

$$\pi \sim \text{Symmetric-Dirichlet}\Big(\frac{1}{n\_classes}\Big).$$

Which is coded as the following.

```
Lambda = nodes.Wishart(dim, np.identity(dim), plates=(n_components,))
mu = nodes.Gaussian(np.zeros(dim), np.identity(dim), plates=(n_components,))
pi = nodes.Dirichlet(np.ones(n_components)/n_components)
z = nodes.Categorical(pi, plates=(n_samples,))
x = nodes.Mixture(z, nodes.Gaussian, mu, Lambda)
```

Due to the high number of attributes (784) in `Mnist` database, `BayesPy` is not able to handle the VMP algorithm with a great number of samples (it attempts to locate a matrix with shape $(n\_samples, n\_components, 784, 784)$). For this reason, `Breast Cancer` is used.

The model is created with the same number of components as different classes (2) in an attempt to learn one Gaussian distribution for each class. After the inference task, `BayesPy` provides a way to inspect each variable posterior by simply printing the desired variable.

Variable moments are also accessible via each variable `u` parameter, given this, we may learn each data-point component belonging posterior, using `Z's` first moment.

The obtained results show a model that does not learn the two classes, despite that, almost all data-points belong to one of the components with high probability. The exact posterior for π is $[6.5/570\ 563.5/570]$. Furthermore, the component assignment variable posterior sets the benign class in the second component with probability 1, with a 0 variance between datapoints. Given this, no component belonging density function can be computed as we did with `Scikit-Learn`.

We may reduce the space to a two-dimensional one using a variational auto-encoder. In this case, the posterior for π is $[0.61\ 0.39]$, which is similar to the real one ($[0.62\ 0.38]$). This gives the intuition that the first component models the more common class (benign). This is confirmed using the component belonging densities (Figure 37). These are similar to the ones obtained using `Scikit-Learn`.

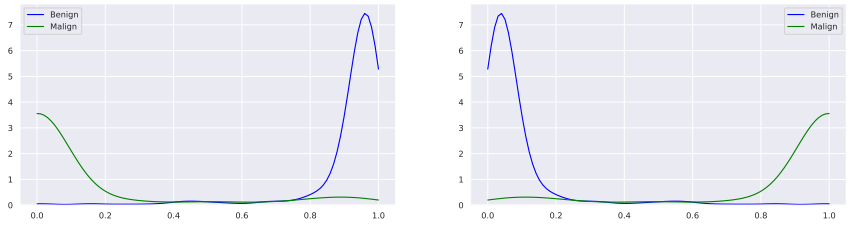


Figure 37: Component belonging density in reduced data.

CONCLUSIONS

In this document I have presented an overview of variational inference and statistical graphical models, explaining how the latter affects the former. Variational methods have increased the possibilities of defining models where parameter Bayesian estimation can be performed.

In order to build a Bayesian network structure from a given set of observations, the concepts of D-separation and D-connection turned to be of great use in order to develop the PC algorithm, which, from a fully connected graph, uses these concepts to erase links between nodes. Later, links are directed using each node's empirical mutual information.

The Bayesian network structure has shown to highly simplify variational methods, for example, in the case of the EM algorithm, the M-step is reduced to assigning each factor of the parametric distribution to the variational one, leading to a simpler procedure than computing the parameter that maximizes the ELBO, which might not be possible.

Conversely, the usage of conditionally conjugate models in the exponential family, transforms the iterative procedure of the CAVI algorithm into a parameter optimization strategy, where each parameter is updated using its distribution parameter function's expectation.

These two concepts, conditionally conjugate models in the exponential family and Bayesian networks, are combined in the variational message passing algorithm, which simplifies each CAVI iteration to a message procedure where each node retrieves messages from its Markov blanket in order to update its own hyper-parameters. This leads to a fully automatized algorithm to perform Bayesian inference with hidden variables.

The combination of artificial neural networks with variational methods, allows to consider non-linear generative models, generalizing existing models such as principal component analysis to more complex ones such as variational auto-encoders, which have shown better results in the studied cases compared to the classical approach.

The success of variational inference has been boosted by the development of software tools, the so called probabilistic programming languages, that have automatized the construction and learning of complex probabilistic models. In this work, we have studied and used some of them, specifically, three different `Python` frameworks have been used. Each of these have shown advantages and disadvantages over the others, not only they perform inference using different techniques but also are limited in different ways.

INFERPY

`InferPy` has shown to be highly limited by the presence of categorical variables in the models. This is a consequence of using `TensorFlow-probability` inference engine to perform gradient descent, as these variables cannot be optimized using this method.

Conversely, InferPy presents high flexibility with model definition, allowing the user to build its own models with explicit relation between the variables. It is the only framework that allows Keras integration, which enables to use models as non-linear principal components analysis and variational auto-encoders.

This framework does also provide a convenient way to inspect each variable posterior and generate samples from it.

BAYESPY

BayesPy is the only of the three frameworks that performs variational message passing as it inference algorithm. This has made it impossible to test on larger databases, as it internally stores much of the information to optimize the procedure, leading to high memory requisites.

This framework has the same flexibility as InferPy providing a more comfortable syntax as it does not request the variational model to be defined.

BayesPy does not provide a default function to access each component probability for a given point, and it must be retrieved from the variable posterior moments. The in-built print function for the Gaussian mixture model works perfectly on "testing" databases (constructed manually from a set of Gaussian distributions) but fails to represent all the components in Breast Cancer reduction.

SCIKIT-LEARN

Scikit-Learn's BayesianGaussianMixture does correctly perform EM on the given model, providing the user a totally functional API from where one may get any posterior information. A negative aspect is the lack of flexibility available compared to the other frameworks. It does allows the user to choose between several model parameters but not make any substantial change to the model.

As being part of a machine learning framework, it provides the necessary functions to use the model in a classification problem, such as score() and predict().

APPENDICES

A

DISTRIBUTIONS IN THE EXPONENTIAL FAMILY

Distribution	Parameter set $\boldsymbol{\theta}$	Base measure h	Parameter function $\boldsymbol{\eta}(\boldsymbol{\theta})$	Statistics $\boldsymbol{T}(x)$
Bernoulli	p	1	$\log \frac{p}{1-p}$	x
Binomial	p	$\binom{n}{x}$	$\log \frac{p}{1-p}$	x
Categorical	$p_1, \dots, p_K$	1	$\begin{pmatrix} \log p_1 \\ \vdots \\ \log p_K \end{pmatrix}$	$\begin{pmatrix} [x=1] \\ \vdots \\ [x=k] \end{pmatrix}$
Gaussian	μ, σ^2	$\frac{1}{\sqrt{2\pi}}$	$\begin{pmatrix} \frac{\mu}{\sigma^2} \\ -\frac{1}{2\sigma^2} \end{pmatrix}$	$\begin{pmatrix} x \\ x^2 \end{pmatrix}$
Multivariate Gaussian	$\boldsymbol{\mu}, \boldsymbol{\Sigma}$	$\sqrt{2\pi}^{-k}$[1]	$\begin{pmatrix} \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \\ -\frac{1}{2}\boldsymbol{\Sigma}^{-1} \end{pmatrix}$	$\begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{x}\boldsymbol{x}^T \end{pmatrix}$
Beta	α, β	$\frac{1}{x(1-x)}$	$\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$	$\begin{pmatrix} \log x \\ \log(1-x) \end{pmatrix}$
Dirichlet	$\alpha_1, \dots, \alpha_K$	1	$\begin{pmatrix} \alpha_1 - 1 \\ \vdots \\ \alpha_K - 1 \end{pmatrix}$	$\begin{pmatrix} \log x_1 \\ \vdots \\ \log x_K \end{pmatrix}$

1 k is the dimensionality of the variable.

Gamma	α, β	1	$\begin{pmatrix} -\beta \\ \alpha - 1 \end{pmatrix}$	$\begin{pmatrix} x \\ \log x \end{pmatrix}$
Wishart	$\nu, \boldsymbol{V}$	1	$\begin{pmatrix} \frac{\nu - p - 1}{2} \\ -\frac{1}{2}\boldsymbol{V}^{-1} \end{pmatrix}$ [2]	$\begin{pmatrix} \boldsymbol{x} \\ \log \lvert\boldsymbol{x}\rvert \end{pmatrix}$

2 $\boldsymbol{V}$ is $p \times p$ dimensional.

B

CONJUGATE DISTRIBUTIONS

In this appendix, examples of conjugate distributions and their conjugate priors are shown. In the following sections X will denote the given random variable and $\boldsymbol{x} = (X_1, \dots, X_N)$ a given set of observations.

CATEGORICAL AND DIRICHLET DISTRIBUTIONS

The Dirichlet distribution is the conjugate prior of the Categorical distribution.

Let $X \sim Categorical(\boldsymbol{\theta})$, where X takes values in $\{1, \dots, I\}$ and $\boldsymbol{\theta} = (\theta_1, \dots, \theta_I)$. The parameter follows a Dirichlet distribution $\boldsymbol{\theta} \sim Dirichlet(\boldsymbol{u})$,

$$P(\boldsymbol{x} \mid \boldsymbol{\theta}) = \prod_{i=1}^{I} \theta_i^{\sum_n \mathbb{I}[x_n = i]} \quad \text{and} \quad P(\boldsymbol{\theta}) = \frac{1}{B(\boldsymbol{u})} \prod_{i=1}^{I} \theta_i^{u_i - 1}.$$

The posterior is therefore,

$$P(\boldsymbol{\theta} \mid \boldsymbol{x}) = \frac{P(\boldsymbol{\theta}) P(\boldsymbol{x} \mid \boldsymbol{\theta})}{P(x)} = \frac{1}{P(\boldsymbol{x}) B(\boldsymbol{u})} \prod_{i=1}^{I} \theta_i^{u_i - 1 + \sum_n \mathbb{I}[x_n = i]}$$

Naming $\boldsymbol{c} = \big(\sum_n \mathbb{I}[x_n = 1], \dots, \sum_n \mathbb{I}[x_n = N]\big)$, and given

$$\int_{\boldsymbol{\theta}} P(\boldsymbol{\theta} \mid \boldsymbol{x}) = 1 \implies \int_{\boldsymbol{\theta}} \prod_{i=1}^{I} \theta_i^{u_i + c_i - 1} = B(\boldsymbol{u} + \boldsymbol{c}) = B(\boldsymbol{u}) P(\boldsymbol{x}).$$

The posterior distribution follows a Dirichlet distribution with

$$\boldsymbol{\theta} \mid \boldsymbol{x} \sim Dirichlet(\boldsymbol{c} + \boldsymbol{u}).$$

A particular case would be using a Binomial distribution for the variable and a Beta distribution for the parameter. In short, *the Beta distribution is the conjugate prior of the Binomial distribution.*

GAUSSIAN AND WISHART DISTRIBUTIONS

The Wishart distribution is the conjugate prior of the Gaussian distribution with known mean and unknown precision.

Let $X \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Lambda})$, where $\boldsymbol{\mu}$ is the mean vector and $\boldsymbol{\Lambda}$ the $p \times p$ precision matrix. The parameter follows a Wishart distribution as

$$\boldsymbol{\Lambda} \sim \mathcal{W}(\nu, \boldsymbol{V}).$$

Given N observations of the variable and $\boldsymbol{x} = (x_1, \dots, x_N)$. The density functions are

$$P(\boldsymbol{x} \mid \boldsymbol{\Lambda}) = \frac{|\boldsymbol{\Lambda}|^{N/2}}{(2\pi^k)^{N/2}} \exp\Big(-\frac{1}{2} \sum_{n=1}^{N} (x_n - \boldsymbol{\mu})^T \boldsymbol{\Lambda} (x_n - \boldsymbol{\mu}) \Big)$$

$$P(\boldsymbol{\Lambda}) = \frac{1}{2^{\nu p/2} |\boldsymbol{V}|^{\nu/2} \Gamma_p(\nu/2)} |\boldsymbol{\Lambda}|^{(\nu-p-1)/2} e^{-(1/2) tr(\boldsymbol{V}^{-1}\boldsymbol{\Lambda})},$$

The prior distribution can be seen as proportional to

$$P(\boldsymbol{\Lambda}) \propto |\boldsymbol{\Lambda}|^{\frac{\nu-p-1}{2}} \exp\Big(-\frac{1}{2} tr(\boldsymbol{V}^{-1}\boldsymbol{\Lambda}) \Big),$$

where the rest comes form normalization. The posterior is then

$$\begin{aligned}
P(\boldsymbol{\Lambda} \mid x) &= \frac{1}{(2\pi^k)^{N/2}} |\boldsymbol{\Lambda}|^{\frac{\nu+N-p-1}{2}} \exp\Big(-\frac{1}{2} \big(tr(\boldsymbol{V}^{-1}\boldsymbol{\Lambda}) + \sum_{n=1}^{N} (x_n - \boldsymbol{\mu})^T \boldsymbol{\Lambda} (x_n - \boldsymbol{\mu}) \big) \Big) \\
&= \frac{1}{(2\pi^k)^{N/2}} |\boldsymbol{\Lambda}|^{\frac{\nu+N-p-1}{2}} \exp\Big(-\frac{1}{2} \big(tr(\boldsymbol{V}^{-1}\boldsymbol{\Lambda}) + \sum_{n=1}^{N} tr((x_n - \boldsymbol{\mu})^T \boldsymbol{\Lambda} (x_n - \boldsymbol{\mu})) \big) \Big) \\
&= \frac{1}{(2\pi^k)^{N/2}} |\boldsymbol{\Lambda}|^{\frac{\nu+N-p-1}{2}} \exp\Big(-\frac{1}{2} \big(tr(\boldsymbol{V}^{-1}\boldsymbol{\Lambda}) + \sum_{n=1}^{N} tr((x_n - \boldsymbol{\mu})(x_n - \boldsymbol{\mu})^T \boldsymbol{\Lambda}) \big) \Big) \\
&= \frac{1}{(2\pi^k)^{N/2}} |\boldsymbol{\Lambda}|^{\frac{\nu+N-p-1}{2}} \exp\Big(-\frac{1}{2} \big(tr(\boldsymbol{V}^{-1} + \sum_{n=1}^{N} (x_n - \boldsymbol{\mu})(x_n - \boldsymbol{\mu})^T) \boldsymbol{\Lambda} \big) \Big).
\end{aligned}$$

Where the property $tr(ABC) = tr(BCA) = tr(CAB)$ is used. In conclusion, the posterior follows a Wishart distribution

$$\boldsymbol{\Lambda} \mid \boldsymbol{x} \sim \mathcal{W}\Big(\nu + N, \Big(\boldsymbol{V}^{-1} + \sum_{n=1}^{N} (x_n - \boldsymbol{\mu})(x_n - \boldsymbol{\mu})^T \Big)^{-1} \Big).$$

GAUSSIAN AND GAUSSIAN-WISHART DISTRIBUTIONS

The Gaussian-Wishart distribution is the conjugate prior of the Gaussian distribution with unknown mean and precision.

Let $X \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Lambda})$, where $\boldsymbol{\mu}$ is the mean vector and $\boldsymbol{\Lambda}$ the $p \times p$ precision matrix. These parameters follow a Gaussian-Wishart distribution as

$$(\boldsymbol{\mu}, \boldsymbol{\Lambda}) \sim \mathcal{NW}(\boldsymbol{\mu}_0, (\beta_0 \boldsymbol{\Lambda})^{-1}, \nu, \boldsymbol{V}).$$

Given N observations of the variable and $\boldsymbol{x} = (x_1, \dots, x_N)$. The density functions are proportional to

$$P(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Lambda}) \propto |\boldsymbol{\Lambda}|^{\frac{N}{2}} \exp\Big(-\frac{1}{2} \sum_{n=1}^{N} (x_n - \boldsymbol{\mu})^T \boldsymbol{\Lambda} (x_n - \boldsymbol{\mu}) \Big),$$

$$P(\boldsymbol{\Lambda}) \propto |\boldsymbol{\Lambda}|^{\frac{\nu-p-1}{2}} \exp\Big(-\frac{1}{2} tr(\boldsymbol{V}^{-1}\boldsymbol{\Lambda}) \Big),$$

$$P(\boldsymbol{\mu} \mid \boldsymbol{\Lambda}) \propto |\boldsymbol{\Lambda}|^{\frac{1}{2}} \exp\Big(-\frac{1}{2} (\boldsymbol{\mu} - \boldsymbol{\mu}_0)^T (\beta_0 \boldsymbol{\Lambda}) (\boldsymbol{\mu} - \boldsymbol{\mu}_0) \Big).$$

The joint prior is proportional to

$$P(\boldsymbol{\mu}, \boldsymbol{\Lambda}) \propto |\boldsymbol{\Lambda}|^{\frac{1}{2}} |\boldsymbol{\Lambda}|^{\frac{\nu-p-1}{2}} \exp\Big(-\frac{1}{2} tr(\boldsymbol{V}^{-1}\boldsymbol{\Lambda}) - \frac{1}{2}(\boldsymbol{\mu}-\boldsymbol{\mu}_0)^T(\beta_0\boldsymbol{\Lambda})(\boldsymbol{\mu}-\boldsymbol{\mu}_0)\Big).$$

The posterior is then

$$P(\boldsymbol{\mu}, \boldsymbol{\Lambda} \mid \boldsymbol{x}) \propto P(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Lambda}) P(\boldsymbol{\mu}, \boldsymbol{\Lambda}) = P(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Lambda}) P(\boldsymbol{\mu} \mid \boldsymbol{\Lambda}) P(\boldsymbol{\Lambda}).$$

$$P(\boldsymbol{\mu}, \boldsymbol{\Lambda} \mid \boldsymbol{x}) \propto |\boldsymbol{\Lambda}|^{\frac{\nu+N-p}{2}} \exp -\frac{1}{2}\Big(tr(\boldsymbol{V}^{-1}\boldsymbol{\Lambda}) + (\boldsymbol{\mu}-\boldsymbol{\mu}_0)^T(\beta_0\boldsymbol{\Lambda})(\boldsymbol{\mu}-\boldsymbol{\mu}_0) + \sum_{n=1}^{N}(x_n-\boldsymbol{\mu})^T\boldsymbol{\Lambda}(x_n-\boldsymbol{\mu})\Big)$$

As the Gaussian-Wishart distribution has $\boldsymbol{\Lambda}^{\frac{1}{2}}$ as a constant factor, the posterior's degrees of freedom ν' is calculated using:

$$|\boldsymbol{\Lambda}|^{\frac{\nu-p+N}{2}} = |\boldsymbol{\Lambda}|^{\frac{\nu-p+N-1}{2}} |\boldsymbol{\Lambda}|^{\frac{1}{2}} \implies \nu' = \nu + N.$$

Using the factors that surround $\boldsymbol{\Lambda}$ inside the exponential term, the new scale parameter β_0' is:

$$\boldsymbol{\mu}^T\beta_0\boldsymbol{\Lambda}\boldsymbol{\mu} + N\boldsymbol{\mu}^T\boldsymbol{\Lambda}\boldsymbol{\mu} = \boldsymbol{\mu}^T((\beta_0+N)\boldsymbol{\Lambda})\boldsymbol{\mu} \implies \beta_0' = \beta_0 + N.$$

The mean parameter comes from the updated term from $2\boldsymbol{\mu}^T\beta_0\boldsymbol{\Lambda}\boldsymbol{\mu}_0$, which is

$$2\boldsymbol{\mu}^T\beta_0'\boldsymbol{\Lambda}\boldsymbol{\mu}_0' = 2\boldsymbol{\mu}^T(\boldsymbol{\Lambda}N\bar{\boldsymbol{x}} + \beta_0\boldsymbol{\Lambda}\boldsymbol{\mu}_0) \implies \boldsymbol{\mu}_0' = \frac{1}{\beta_0'}(N\bar{\boldsymbol{x}} + \beta_0\boldsymbol{\mu}_0).$$

Where

$$\bar{\boldsymbol{x}} = \frac{1}{N}\sum_{n=1}^{N} x_n.$$

The updated scale matrix $\boldsymbol{V}'$ comes from computing the new trace value, adding to it any one dimensional term multiplied by $\boldsymbol{\Lambda}$.

$$\begin{aligned} tr(\boldsymbol{V}'^{-1}\boldsymbol{\Lambda}) = tr(\boldsymbol{V}^{-1}\boldsymbol{\Lambda} + \sum_{n=1}^{N}(x_n-\bar{\boldsymbol{x}})(x_n-\bar{\boldsymbol{x}})^T\boldsymbol{\Lambda} + \beta_0(\bar{\boldsymbol{x}}-\boldsymbol{\mu}_0)(\bar{\boldsymbol{x}}-\boldsymbol{\mu}_0)^T\boldsymbol{\Lambda} \\ + (N-\beta_0)\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}^T\boldsymbol{\Lambda} + 2\beta_0\bar{\boldsymbol{x}}\mu_0^T\boldsymbol{\Lambda}). \end{aligned}$$

Then,

$$\boldsymbol{V}'^{-1} = \Big(\boldsymbol{V}^{-1} + \sum_{n=1}^{N}(x_n-\bar{\boldsymbol{x}})(x_n-\bar{\boldsymbol{x}})^T + \frac{\beta_0 N}{\beta_0+N}(\bar{\boldsymbol{x}}-\boldsymbol{\mu}_0)(\bar{\boldsymbol{x}}-\boldsymbol{\mu}_0)^T\Big).$$

As a result, the posterior follows a Gaussian-Wishart distribution

$$(\boldsymbol{\mu}, \boldsymbol{\Lambda} \mid \boldsymbol{x}) \sim \mathcal{NW}(\boldsymbol{\mu}_0', (\beta_0'\boldsymbol{\Lambda})^{-1}, \nu', \boldsymbol{V}').$$

NOTATION

X	Observed random variable.
$\boldsymbol{X}$	Bold symbols indicate sets of variables $(X_1, \ldots, X_N)$.
$\boldsymbol{x}$	Set of observations of the set of variables $\boldsymbol{X}$.
x	Specific value of the random variable X.
$P(x)$	Either probability mass function or density function of X evaluated on x.
P	Probability distribution.
Q	Variational probability distribution used in variational inference methods.
Z	Latent random variable.
$\boldsymbol{\theta}$	Set of parameters of a parametric model $P(\boldsymbol{x} \mid \boldsymbol{\theta})$.
θ	Single parameter of a model $P(\boldsymbol{x} \mid \theta)$.
$ch(X)$	Children nodes of X in a graph.
$cp(X, Y)$	Co-parents of Y with X, that is, $pa(Y) \backslash \{X\}$.
$ne(X)$	Neighbor nodes of X in a graph.
$pa(X)$	The set of parents of node X in a graph. The notation pa_X is also used.
$pa(x)$	Specific value of the random variables conforming the set of parents of node X in a graph. The notation pa_x is also used. Using lowercase in the random variable is also used in other graph nodes subsets, such as children and co-parents.
$tr()$	Trace of the given matrix.

BIBLIOGRAPHY

Anderson, James R, & Peterson, Carsten. 1987. A mean field theory learning algorithm for neural networks. *Complex Systems*, **1**, 995–1019.

Bandyopadhyay, Prasanta S, & Forster, Malcolm R. 2011. Philosophy of statistics: An introduction. *Pages 1–50 of: Philosophy of statistics*. Elsevier.

Barber, David. 2007. *Bayesian Reasoning and Machine Learning*. Cambridge University Press.

Bishop, Christopher M. 2006. *Pattern recognition and machine learning*. springer.

Bishop, Christopher M, Spiegelhalter, David, & Winn, John. 2003. VIBES: A variational inference engine for Bayesian networks. *Pages 793–800 of: Advances in neural information processing systems*.

Blei, David M. 2012. Probabilistic topic models. *Communications of the ACM*, **55**(4), 77–84.

Blei, David M. 2014. Build, compute, critique, repeat: Data analysis with latent variable models. *Annual Review of Statistics and Its Application*, **1**, 203–232.

Blei, David M, Ng, Andrew Y, & Jordan, Michael I. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, **3**(Jan), 993–1022.

Blei, David M, Kucukelbir, Alp, & McAuliffe, Jon D. 2017. Variational inference: A review for statisticians. *Journal of the American statistical Association*.

Buitinck, Lars, Louppe, Gilles, Blondel, Mathieu, Pedregosa, Fabian, Mueller, Andreas, Grisel, Olivier, Niculae, Vlad, Prettenhofer, Peter, Gramfort, Alexandre, Grobler, Jaques, Layton, Robert, VanderPlas, Jake, Joly, Arnaud, Holt, Brian, & Varoquaux, Gaël. 2013. API design for machine learning software: experiences from the scikit-learn project. *Pages 108–122 of: ECML PKDD Workshop: Languages for Data Mining and Machine Learning*.

Cauchy, Augustin. 1847. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, **25**(1847), 536–538.

Cox, David Roxbee. 2006. *Principles of statistical inference*. Cambridge university press.

Cózar, Javier, Cabañas, Rafael, Salmerón, Antonio, & Masegosa, Andrés R. 2019. InferPy: Probabilistic Modeling with Deep Neural Networks Made Easy. *arXiv preprint arXiv:1908.11161*.

Dempster, Arthur P, Laird, Nan M, & Rubin, Donald B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, **39**(1), 1–22.

Do, Chuong B, & Batzoglou, Serafim. 2008. What is the expectation maximization algorithm? *Nature biotechnology*, **26**(8), 897–899.

Dua, Dheeru, & Graff, Casey. 2017. *UCI Machine Learning Repository*. http://archive.ics.uci.edu/ml.

Farrow, Malcolm. 2008. MAS3301 Bayesian Statistics.

Goodfellow, Ian, Bengio, Yoshua, & Courville, Aaron. 2016. *Deep learning*. MIT press.

Griewank, Andreas, *et al.* 1989. On automatic differentiation. *Mathematical Programming: recent developments and applications*, **6**(6), 83–107.

Grimmett, Geoffrey R. 1973. A theorem about random fields. *Bulletin of the London Mathematical Society*, **5**(1), 81–84.

Hinton, Geoffrey E, & Van Camp, Drew. 1993. Keeping the neural networks simple by minimizing the description length of the weights. *Pages 5–13 of: Proceedings of the sixth annual conference on Computational learning theory*.

Hoffman, Matthew D, Blei, David M, Wang, Chong, & Paisley, John. 2013. Stochastic variational inference. *The Journal of Machine Learning Research*, **14**(1), 1303–1347.

Impagliazzo, Russell, & Paturi, Ramamohan. 1999. Complexity of fc-SAT. *Page 237 of: Proceedings*, vol. 14. IEEE Computer Society Press.

Jordan, Jeremy. 2018. *Variational autoencoders [Online]*. https://www.jeremyjordan.me/variational-autoencoders/ Accesed on 2020-08-10.

Jordan, Michael I, Ghahramani, Zoubin, Jaakkola, Tommi S, & Saul, Lawrence K. 1999. An introduction to variational methods for graphical models. *Machine learning*.

Kiefer, Jack, Wolfowitz, Jacob, *et al.* 1952. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, **23**(3), 462–466.

Kingma, Diederik P., & Ba, Jimmy. 2014. *Adam: A Method for Stochastic Optimization*.

Kingma, Diederik P, & Welling, Max. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Koller, Daphne, & Friedman, Nir. 2009. *Probabilistic Graphical Models, Principles and Techniques*. The MIT Press.

Koopman, Bernard Osgood. 1936. On distributions admitting a sufficient statistic. *Transactions of the American Mathematical society*, **39**(3), 399–409.

Kullback, Solomon. 1997. *Information theory and statistics*. Courier Corporation.

Le, Thuc Duy, Hoang, Tao, Li, Jiuyong, Liu, Lin, Liu, Huawen, & Hu, Shu. 2019. A Fast PC Algorithm for High Dimensional Causal Discovery with Multi-Core PCs. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, **16**(5), 1483–1495.

LeCun, Yann, & Cortes, Corinna. 2010. MNIST handwritten digit database.

LeCun, Yann, Cortes, Corinna, & Burges, CJ. 2010. MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, **2**.

Li, Stan Z. 2009. *Markov random field modeling in image analysis*. Springer Science & Business Media.

Luttinen, Jaakko. 2014. BayesPy: Variational Bayesian Inference in Python. **17**(10).

Masegosa, Andrés R., Cabañas, Rafael, Langseth, Helge, Nielsen, Thomas D., & Salmerón, Antonio. 2019. *Probabilistic Models with Deep Neural Networks*.

McLachlan, Geoffrey J, & Krishnan, Thriyambakam. 2007. *The EM algorithm and extensions*. Vol. 382. John Wiley & Sons.

Neal, Radford M, & Hinton, Geoffrey E. 1998. A view of the EM algorithm that justifies incremental, sparse, and other variants. *Pages 355–368 of: Learning in graphical models*. Springer.

Pearl, Judea, & Dechter, Rina. 2013. Identifying Independences in Casual Graphs with Feedback.

Pearson, Karl. 1901. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, **2**(11), 559–572.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, **12**, 2825–2830.

Robbins, H. 2007. A Stochastic Approximation Method. *Annals of Mathematical Statistics*, **22**, 400–407.

Rossi, Richard J. 2018. *Mathematical statistics: an introduction to likelihood based inference*. John Wiley & Sons.

Shachter, Ross D. 2013. Bayes-Ball: The Rational Pastime.

Spirtes, Peter, Glymour, Clark N, Scheines, Richard, & Heckerman, David. 2000. *Causation, prediction, and search*. MIT press.

Svensén, Markus. 2007. Pattern Recognition and Machine Learning Solutions to the Exercises : Web-Edition.

Tipping, Michael E, & Bishop, Christopher M. 1999. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **61**(3), 611–622.

Upton, Graham, & Cook, Ian. 2014. *A dictionary of statistics 3e*. Oxford university press.

Wainwright, Martin J., & Jordan, Michael I. 2008. *Graphical Models, Exponential Families and Variational Inference*. Now Publishers Inc.

Winn, John, & Bishop, Christopher M. 2005. Variational message passing. *Journal of Machine Learning Research*, **6**(Apr), 661–694.