Bayesian Filtering
Sigma 203b - Telecom ParisTech

A. Sabourin[1]

March 6, 2017

[1]LTCI, Telecom ParisTech, univeristé Paris-Saclay.

Contents

Organisation du cours

- organisation générale: cours et TP en alternance, les 2 dernieres séances sont consacrées au miniprojet.
- Langage de programmation: matlab ou octave
- prise en main d'octave: voir en annexe.
- Evaluation: Mini projet
  - Pour la réalisation du mini-projet, il vous faudra réutiliser certains algorithmes implémentés pendant les séances de TP. Une partie du mini-projet comportera des questions théoriques.
  - Vous présenterez vos résultats lors d'un oral d'ue vingtaine de minute lors pendant la derniere séance (19/04)
  - Vous devrez produire
    * Un rapport écrit à rendre avant la soutenance, contenant les réponses aux questions théoriques et vos résultats **commentés**, en pdf ou en format papier.
    * L'ensemble du code matlab (ou octave) implémenté. Ce code devra être exécutable directement par le correcteur, sans nécessiter de modification.

Chapter 1

Introduction: filtering problems

Material for this course:

- Särkkä (2013): a good introduction, which explains nicely the main principles without insisting much on the mathematical details
- Cappé et al. (2009): more details, gives the full mathematical story. Focus displaced towards inference.

1.1 Filtering problems in signal processing

General Goal : Recover the 'true' signal $(X_n)_{n\geq 0}$, based on noisy observations $(Y_n)_{n\geq 0}$. (X_n), (Y_n) are times series with 'simple' temporal structures, n is the time index

- Probabilistic framework: $(\Omega, \mathcal{A}, \mathbb{P})$ a probability space, X_n, Y_n: random vectors.
- True signal $X_n : \Omega \to \mathbb{R}^p$: a random vector, $(X_n)_{n\in\mathbb{N}}$ is a stochastic process with discrete time, also called the 'hidden process'.
- Noisy signal $Y_n : \Omega \to \mathbb{R}^q$, $(Y_n)_{n\geq 1}$ is the observed process (= the data).

Main challenge: The data $(Y_n)_{n\geq 1}$ are observed in a sequential manner. One wants to use the information at time n, combined with Y_{n+1}, to 'deduce something' about the new state X_{n+1} (and maybe also the whole past $X_0, \ldots, X_n$. In the machine learning community, this task is usually called **online learning** . 'Deduce something' means (in this course) compute the posterior distribution. In order to save time, one does not want to compute the latter using all the previous data $(Y_0, \ldots, Y_{n+1})$ at each time step. The idea is rather to combine the posterior distribution at time n with the single new information Y_{n+1} to obtain the posterior distribution at time $n+1$. In other words, we are looking for recurrence formulae concerning the posterior distributions.

1.1.1 Examples

Filtering (in a Bayesian way or not) is useful in many domains: telecommunications, machine learning (tracking/tagging). Below are a few examples.

1. Geolocalisation.
   X_n: true localization of a captor C. The respective distances from C to three satellites are measured $\rightarrow Y_n = (Y_n^1, Y_n^2, Y_n^3)_{n\geq 0}$.
   Goal: Recover $(X_n)_{n\geq 1}$ using $(Y_n)_{n\geq 1}$ and some information about the measurement errors.

2. Target tracking.
   Several sensors $(S_1, \ldots, S_p)$ record the position of a target (T). The signal $(Y_n^1, \ldots, Y_n^p)$ is a noisy version of the true position X_n. If one knows about a sensible model describing the trajectory of T, observing $(Y_0, \ldots, Y_n)$ gives more information about X_n than Y_n alone.

3. Multiple target tracking: same problem, except that now several targets $(T_1, \ldots, T_k)$ are considered. An additional question arises: how to assign each measurement to a given target?

4. Medical imaging (electroencephalogram): the goal is to reconstruct the 'source' (*i.e.,* the magnetic field in the brain) given the recorded field on the skull from sensors.

1.1.2 Classical models

Inputs We consider in this course that the following are given (expert advice, preliminary estimation step, ...):

1. a prior distribution for $X_{0:T}$ $(T \geq 1)$: $p(x_{0:T})$
2. an observation model : $p\left(y_{1:T} \mid x_{0:T}\right)$

This applies to the two standard models defined below.

Definition 1.1.1 (Linear Gaussian model)**.**

$$\begin{cases} X_0 & \sim \mathcal{N}(m_0, P_0) \\ X_k & = A_{k-1}X_{k-1} + \epsilon_{k-1} \\ Y_k & = B_k X_k + \eta_k \end{cases},$$

where

- $P_0 \in \mathbb{R}^{p\times p}$ *is a positive definite matrix,* $m_0 \in \mathbb{R}^p$
- $(A_k)_{k\geq 0} \in \mathbb{R}^{p\times p}$, $B_k \in \mathbb{R}^{q\times p}$ *are known and deterministic (constants, not random variables)*
- $\epsilon_k,\ \eta_k$ *are Gaussian noises:* $\epsilon_k \sim \mathcal{N}\left(0, Q_k\right)$ *and* $\eta_k \sim \mathcal{N}\left(0, R_k\right)$
- $\left(X_0,\ (\epsilon_k)_{k\geq 0},\ (\eta_k)_{k\geq 1}\right)$ *are independent*

The following is just a generalization of the Gaussian linear model

Definition 1.1.2 (Hidden Markov model (HMM))**.**

$$\begin{cases} X_0 & \sim \mathrm{P}_{X_0} \qquad (\textit{ any initial distribution}) \\ X_k & = f_{k-1}\left(X_{k-1}, \epsilon_{k-1}\right) \\ Y_k & = g_k\left(X_k, \eta_k\right) \end{cases},$$

where f_k, g_k *are known functions and* $\left(X_0, (\epsilon_k)_{k\geq 0},\ (\eta_k)_{k\geq 1}\right)$ *are independent.*

(An alternative definition of HMM's and some useful properties are given later in Section 1.2)

1.1.3 Three main tasks with HMM's: filtering, smoothing, predicting.

Notations, conventions : For $0 \leq k \leq r$, we denote $x_{k:r} = (x_k, x_{k+1}, \ldots, x_r)$. In the sequel, for notational convenience, we assume that all the considered distributions have a density with respect to the Lebesgue measure or to a counting measure in the discrete case. All these densities will be denoted by p, and we denote *e.g.,* by $p(x)$ the density of $\mathcal{L}(X)$ and by $p(x|y)$ the density of $\mathcal{L}(X|Y=y)$.

Three main tasks in a classical framework

1. **Filtering :** estimating the (hidden) current value x_k using the observed values $y_{1:k} = (y_1, \ldots, y_k)$ *i.e.,* constructing an estimator $\hat{x}_k = f_k\left(y_{1:k}\right)$
2. **Smoothing :** estimating $x_{0:k}$ from $y_{1:k}$, *i.e.,* constructing an estimator $\widehat{x_{0:k}} = f_k\left(y_{1:k}\right)$
3. **Prediction :** estimating a 'likely' next hidden value x_{k+1}, given $y_{1:k}$, *i.e.,* constructing an estimator $\hat{x}_{k+1} = f_k\left(y_{1:k}\right)$.

Then how to choose a 'good' estimator? It depends on the chosen framework (classical/Bayesian). One option is to construct an estimator which minimizes the quadratic risk, defined below.

Definition 1.1.3 (Minimum squared error estimators)**.**

1. *Filtering :* $\hat{x}_n = \arg\min_{a=f(Y_{1:k}), f\in\mathcal{F}} \mathbb{E}\left[(X_k - a)^2 | Y_{1:k} = y_{1:k}\right] = \mathbb{E}\left(\, X_k \mid y_{1:k}\,\right)$
2. *Smoothing :* $\hat{x}_n = \arg\min_{a=f(Y_{1:k}), f\in\mathcal{F}} \mathbb{E}\left[(X_{0:k} - a)^2 | Y_{1:k} = y_{1:k}\right] = \mathbb{E}\left(\, X_{0:k} \mid y_{1:k}\,\right)$
3. *Prediction :* $\hat{x}_n = \arg\min_{a=f(Y_{1:k}), f\in\mathcal{F}} \mathbb{E}\left[(X_{k+1} - a)^2 | Y_{1:k} = y_{1:k}\right] = \mathbb{E}\left[X_{k+1} \mid y_{1:k}\right]$

with $\mathcal{F}$ *a class of function (sufficiently basic for the optimization problem to be well posed).*

Remark 1.1.4. *In practice* $\mathcal{F}$ *is often the family of linear estimators.*

Three main tasks in a Bayesian framework

The idea is to compute the conditional distributions $\mathcal{L}(X \mid Y = y)$ rather than point estimators. Then point estimators can always be deduced, such as the conditional expectancy. The advantage is that the conditional distribution gives some information about the uncertainty of the estimator (through *e.g.,* its quantiles). The connection with Bayesian statistics is the following:

- $(X_n)_{n\geq 0}$ plays the role of the parameter (usually called θ in statistics).
- The observation (the data) are $(Y_n)_{n\geq 1}$ (not the X_n's !)
- The prior distribution is $\mathcal{L}(X_n)_{n\geq 0}$
- The observation model is $\mathcal{L}(Y|X) := \mathcal{L}\left((Y_n)_{n\geq 1} \,|\, (X_n)_{n\geq 0}\right)$
- the posterior distribution is the conditional distribution $\mathcal{L}(X|Y)$.

The three main learning tasks are then defined as follows:

1. **Bayesian filtering :** computation of the posterior distributions $p(x_k \mid y_{1:k})$
2. **Bayesian smoothing :** computation of the posterior distributions $p(x_{0:k} \mid y_{1:k})$
3. **Bayesian prediction :** computation of the posterior distributions $p(x_{k+1} \mid y_{1:k})$

Remark 1.1.5. *The knowledge of* $p(x_k \mid y_{1\ :k})$ *gives access to* $\hat{x}_n = \mathbb{E}[X_k|y_{1:k}]$

1.1.4 Conditioning, conditional distribution: probabilistic background

Definition 1.1.6 (Conditional distribution)**.** *Let* $W : \Omega \to \mathbb{R}^p$ *and* $Z : \Omega \to \mathbb{R}^q$ *be random vectors defined on* $(\Omega, ,\mathbb{P})$*. It is assumed that the distribution of the pair* (W, P) *has a density* w.r.t. *the Lebesgue measure/ a discrete counting measure on* $\mathbb{R}^{p+q}$*. Then the conditional distribution* $\mathcal{L}(Z \mid W = w)$*, for* w *such that* $p(w) > 0$*, is given by its density*

$$p(z \,|\, w) = \frac{p(z, w)}{p(w)} = \frac{p(w \,|\, z)p(z)}{p(w)}.$$

The latter equality is called the Bayes formula*. For any sets* $A \subset \mathbb{R}^q, B \subset \mathbb{R}^p$*, it holds that*

$$\mathbb{P}(Z \in A, W \in B) = \int_{w\in B} \Big(\int_{z\in A} p(z\,|w)\,\mathrm{d}z \Big) p(w)\,\mathrm{d}w$$

also for any measurable, positive function f*:*

$$\mathbb{E}\,(\,f(Z, W)\,) = \int_{w\in\mathbb{R}^p} \Big(\int_{z\in\mathbb{R}^q} f(z, w)p(z \,|\, w)\,\mathrm{d}z \Big) p(w)\,\mathrm{d}w$$

N.B.*: we denote* $\mathrm{d}w = \mathrm{d}w_1, \ldots, \mathrm{d}w_p$*, similar meaning for* $\mathrm{d}z$*.*

1.2 Hidden Markov models (HMM)

HMM's have been defined in 1.1.2. An alternative definition of HMM's is given in this section, based on the following central notion:

1.2.1 Conditional independence

Definition 1.2.1 (Conditional independence: general case)**.** *Let (W, Y, Z) be three random vectors, with values respectively in Euclidean spaces $\mathcal{W}, \mathcal{Y}, \mathcal{Z}$. Then Z is independent from X given Y, which we denote '$Z \perp\!\!\!\perp_Y X$' if $\forall g : \mathcal{Z} \to \mathbb{R}^+$ (or $\mathcal{Z} \to \mathbb{R}$ bounded),*

$$\mathbb{E}\left(\, g(Z) \mid (Y, X)\,\right) = \mathbb{E}\left(\, g(Z) | Y\,\right) \qquad \textit{almost surely } (a.s.)$$

In the continuous case, an convenient equivalent definition is available:

Proposition 1.2.2 (Equivalent definition (I), continuous case)
In the case where $\mathcal{L}(X, Y, Z)$ has a density p,

$$Z \perp\!\!\!\perp_Y X \iff p\left(z|(x, y)\right) = p\left(z|y\right)$$

Proposition 1.2.3 (Equivalent definition (II), continuous case)
In the case where $\mathcal{L}(X, Y, Z)$ has a density p,

$$Z \perp\!\!\!\perp_Y X \iff p\left(z, x|y\right) = p\left(z|y\right) p(x|y)$$

1.2.2 Hidden Markov Models: alternative definitions

Definition 1.2.4 (Markov chain)**.** *$(X_n)_{n\geq 0}$ is called a Markov chain if $\forall k \geq 1$ $X_k \perp\!\!\!\perp_{X_{k-1}} X_{0:k-2}$ i.e $p\left(x_k|x_{0:k-1}\right) = p\left(x_k|x_{k-1}\right)$.*

The following definition can be shown to be equivalent to Definition 1.1.2 from the last section.

Definition 1.2.5 (HMM)**.** *A HMM is a pair of processes $\Big((X_n)_{n\geq 0},\ (Y_n)_{n\geq 1}\Big)$ such that*

1. *(X_n) is a Markov chain,*
2. *$\forall\ k \geq 1,\ \forall T \geq k\ \ Y_k \perp\!\!\!\perp_{X_k} \{(X_{0:T}, Y_{1:T})\ \setminus\ (X_k, Y_k)\}$*

1.2.3 Graphical representation : Directed Acyclic Graph

The dependencies among random variables $(Z_1, \ldots, Z_n)$ may be summarized by a Directed acyclic graph (DAG), *i.e.,* a graph with oriented edges (arrows) and no cycle. The vertices are the random variables themselves, and the oriented edges are defined such that

1. $Z_i \perp\!\!\!\perp Z_j \iff$ there is no path connecting Z_i and Z_j
2. $Z_j \perp\!\!\!\perp_{Z_k} Z_i \iff$ when node Z_k and its edges are deleted from the graph, there is no more path between Z_j and Z_i.

Such a DAG is called the 'Bayesian network' associated with $(Z_1, \ldots, Z_n)$ Given the joint distribution $p(z_1, \ldots, z_n)$, a DAG satisfying 1. and 2. as above may always be constructed as follows:

- Start with a graph with vertices $(Z_i)_{i=1:n}$, and an empty set of edges.

- Decompose the joint distribution as a product

$$p(z_1, \ldots, z_n) = p(z_1)p(z_2|z_1)p(z_3|z_1, z_2), \ldots, p(z_n|z_1, \ldots, z_{n-1}).$$

- In the above decomposition, due to conditional independence relations, some conditioning variables may be removed, *e.g.,* one may have $p(z_3|z_1, z_2) = p(z_3|z_1)$. Define for $k = 1, \ldots, n$, the sets of 'parents' as the minimal set of indices $I(k)$ such that

$$p(z_k|z_1, \ldots, z_{k-1}) = p(z_k|\ (z_i, i \in I(k))$$

- Add the edge $Z_i \to Z_k$ to the set of edges if and only if $i \in I(k)$.

Following this principle, consider the particular case of a HMM: the joint distribution factorizes as

$$p(x_{0:k}, y_{1:k}) = p(x_0) \prod_{k=1}^{n} p(x_k|x_{k-1})p(y_k|x_k). \tag{1.1}$$

Thus a HMM can be represented by the the following DAG:

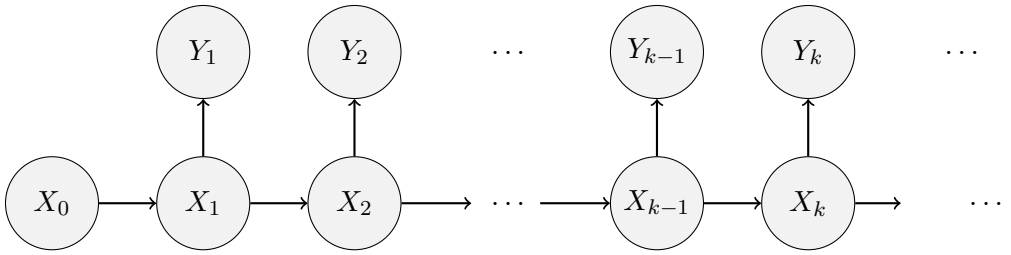

Figure 1.1: DAG representation of a HMM

The advantage of this graphical representation is to provide a straightforward proof of the following result:

Proposition 1.2.6 (conditional independence properties of a HMM)

1. $p\left(x_k|x_{0:k-1}, y_{1:k-1}\right) = p\left(x_k|x_{k-1}\right)$
   (conditional independence between present and past, given the last hidden state)

2. $p\left(x_{k-1}|x_{k:T}, y_{k:T}\right) = p\left(x_{k-1}|x_k\right)$
   (conditional independence between present and future, given the next hidden state)

3. $p\left(y_k|x_{0:k}, y_{1:k-1}\right) = p\left(y_k|x_k\right)$
   (conditional independence between current observation and the past, given the current hidden state)

The particular factorization (1.1) for the joint distribution of a HMM allows to specify the whole distribution of $(X_{1:n}, Y_{1:n})$ using more basic quantities, as stated below.

Proposition 1.2.7 (specification of a HMM)
*A HMM is entirely determined (*i.e., *the joint distribution $p(x_{1:n}, y_{1:n})$) is entirely determined by*

1. *The transition model for the hidden states* $p(x_{k+1}|x_k)$
2. *The observation model* $p\left(y_k|x_k\right)$
3. *The prior on the initial state* $p\left(x_0\right)$

1.2.4 Sequential formula for filtering distributions:

As mentioned at the beginning of this chapter, the goal is to establish sequential formulas, in order to avoid the heavy computation of the posterior distribution $p(x_k|y_1,\ldots,y_k)$ (involving all past observations). The following proposition is a starting point.

Proposition 1.2.8
In a HMM, it holds that

$$p\left(x_{k+1}|y_{1:k+1}\right) = \frac{1}{Z} \underbrace{p\left(y_{k+1}|x_{k+1}\right)}_{\textit{observation model}} \underbrace{p(x_{k+1}|y_{1:k})}_{\textit{predictive distribution}} \tag{1.2}$$

where $Z = p\left(y_{k+1}|y_{1:k}\right)$ *is constant with respect to* x_k *and where the predictive distribution is obtained by integration,*

$$p\left(x_{k+1}|y_{1:k}\right) = \int \underbrace{p\left(x_{k+1}|x_k\right)}_{\textit{transition model}} \underbrace{p\left(x_k|y_{1:k}\right)}_{\textit{filtering distribution at time } k} dx_k \tag{1.3}$$

Proof.

$$\begin{aligned}
p\left(x_{k+1}|y_{1:k+1}\right) &= \frac{p\left(x_{k+1}, y_{1:k+1}\right)}{p(y_{1:k+1})} \\
&= \frac{p\left(x_{k+1}, y_{k+1}|y_{1:k}\right) p\left(y_{1:k}\right)}{p(y_{1:k}, y_{k+1})} \\
&= \frac{p\left(x_{k+1}, y_{k+1}|y_{1:k}\right)}{p(y_{k+1}|y_{1:k})} \\
&= \frac{1}{Z} p\left(x_{k+1}, y_{k+1}|y_{1:k}\right) \\
&= \frac{1}{Z} \frac{p\left(x_{k+1}, y_{k+1}, y_{1:k}\right)}{p(y_{1:k})} \\
&= \frac{1}{Z} \frac{p\left(y_{k+1}|x_{k+1}, y_{1:k}\right) p\left(x_{k+1}, y_{1:k}\right)}{p(y_{1:k})} \\
&= \frac{1}{Z} p\left(y_{k+1}|x_{k+1}, y_{1:k}\right) p\left(x_{k+1}|y_{1:k}\right) \\
&= \frac{1}{Z} p\left(y_{k+1}|x_{k+1}\right) p\left(x_{k+1}|y_{1:k}\right) \text{ using proposition 1.2.6-3.}
\end{aligned}$$

The formula for the predictive distribution comes from the fact that

$$\begin{aligned}
p\left(x_{k+1}|y_{1:k}\right) &= \int p(x_{k+1}, x_k|y_{1:k}) dx_k \\
&= \int p(x_{k+1}|x_k, y_{1:k}) p(x_k|y_{1:k}) dx_k \\
&= \int p(x_{k+1}|x_k) p(x_k|y_{1:k}) dx_k
\end{aligned}$$

■

Algorithm 1.1
Inputs:
Initial state distribution $p(x_0)$*, transition distributions* $p(x_{k+1}|x_k)$*, observation model* $p(y_k|x_k)$*.*

- **Initialize:** $p(x_0 \,|\, y_{1:0}) = p_0\,(x_0)$
- **for** $k \geq 0$**:**
  - **Prediction step:**
    compute the predictive $p\,(x_{k+1}|y_{1:k})$ *using* (1.3)
  - **Update step** *(using the new observation* y_{k+1}*):*
    Compute the filtering distribution $p(x_{k=1}|y_{1:k+1})$ *using* (1.2)*.*

1.3 Problems

Exercise 1.1 (factorization of a HMM)**:**
Using definition 1.2.5 of a HMM (conditional independence), prove that the joint density for $(X_{0:n}, Y_{1:n})$ factorizes according to equation (1.1).

Exercise 1.2 (conditional independence)**:**
Prove Proposition 1.2.3 using Proposition 1.2.2.

Exercise 1.3 (conditional independence properties in a HMM)**:**
Prove the conditional independence properties of a HMM stated in Proposition 1.2.6, using definition 1.2.5.

Exercise 1.4 (One armed bandit)**:**
A slot machine has two states $\{0, 1\}$. Let X_k denote the state of the machine at time $k, k = 0, \ldots, n$. $x_k = 0$ correspond to a regular regime, whereas when $x_k = 1$, something is wrong in the mechanism. Let $Y_k \in \{0, 10\}$ denote the player's reward at time k. The variables $(X_{0:n}, Y_{1:n})$ are supposed to form a HMM. If $x_k = 0$, then $Y_k = 10$ with probability 0.01. If $x_k = 1$, the probability that $Y_k = 10$ is 0.5. The initial distribution for X_0 is $(p(x_0 = 0) = 0.5,\ p(x_0 = 1) = 0.5$. The transition matrix for the Markov chain $(X_k)_{k\geq 0}$ is $P = \begin{pmatrix} 0.8 & 0.2 \\ 0.5 & 0.5 \end{pmatrix}$.

1. Identify the observation model, the state transition model and the prior distribution on the initial state.
2. We denote by $\pi_k = \Big((p(x_k = 0|y_{1:k}), p(x_k = 1|y_{1:k})\Big)$ the filtering distribution at time k. Write the predictive distribution $\pi^-_{k+1} = \Big(p(x_{k+1} = 0|y_{1;\ldots,k}), p(x_{k+1} = 1|y_{1:k})\Big)$ as a function of π_k and the transition matrix P.
3. Compute explicitly the recurrence formula for the filtering distribution π_k.

Chapter 2

Filtering Linear Gaussian Models: Kalman filter

2.1 Kalman Filter

2.1.1 Gaussian Vectors

Proposition 2.1.1 (Properties of Gaussian vectors)

1. *If* $Z = (Z_1, \cdots, Z_d) \sim \mathcal{N}(\mu, \Sigma)$, *then* $\forall A \in \mathbb{R}^{n\times d}$, $\forall b \in \mathbb{R}^n$,
$$AZ + b \sim \mathcal{N}\left(b + A\mu, A\Sigma A^T\right).$$

2. ***Marginal distributions :*** *If* $\begin{bmatrix} X \\ Y \end{bmatrix} \sim \mathcal{N}(\mu, \Sigma)$ *with*
$$X \in \mathbb{R}^p, \quad Y \in \mathbb{R}^q, \quad \mu = \begin{bmatrix} m_X \\ m_Y \end{bmatrix}, \quad \Sigma = \begin{bmatrix} \Sigma_X & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_Y \end{bmatrix},$$
then $X \sim \mathcal{N}(m_X, \Sigma_X)$ *and* $Y \sim \mathcal{N}(m_Y, \Sigma_Y)$

3. ***Conditional distributions:*** *Let* $\begin{bmatrix} X \\ Y \end{bmatrix} \sim \mathcal{N}(\mu, \Sigma)$ *as in* 2. *Then*
$$\mathcal{L}(X|Y = y) = \mathcal{N}\left(m_X + \Sigma_{XY}\Sigma_Y^{-1}(y - m_Y), \Sigma_X - \Sigma_{XY}\Sigma_Y^{-1}\Sigma_{YX}\right)$$

4. ***Augmenting:*** *Let* $Z \in \mathbb{R}^p$ *and* $U \in \mathbb{R}^q$ *be two Gaussian vectors such that* $Z \perp\!\!\!\perp U$. *Then* $\forall A \in \mathbb{R}^{p\times q}$
$$\begin{bmatrix} Z \\ AZ + U \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m_Z \\ Am_Z + m_U \end{bmatrix}, \begin{bmatrix} \Sigma_Z & \Sigma_Z A^T \\ A\Sigma_Z & A\Sigma_Z A^T + \Sigma_U \end{bmatrix}\right)$$

Proof.

1. Consider the characteristic function of $AZ + b$. Let $t \in \mathbb{R}^n$. Then

$$\phi_{AX+b}(t) = E\left[e^{i\langle AZ+b,t\rangle}\right] = e^{i\langle b,t\rangle} E\left[e^{i\langle AZ,t\rangle}\right] = e^{i\langle b,t\rangle} E\left[e^{i\langle Z,A^T t\rangle}\right] = e^{i\langle b,t\rangle}\phi_Z(A^T t).$$

   But Z is a Gaussian vector, so that its characteristic function is $\phi_Z(t) = e^{i\langle t,\mu\rangle - \frac{1}{2}t^T\Sigma t}$, whence

$$\phi_{AX+b}(t) = e^{i(\langle b,t\rangle + \langle A^T t,\mu\rangle - \frac{1}{2}(A^T t)^T\Sigma(A^T t)} = e^{i\langle t,A\mu+b\rangle - \frac{1}{2}t^T(A\Sigma A^T)t},$$

   which is the characteristic function of another Gaussian vector with mean and covariance as in the statement.

2. Write $X = [I_p, 0_q]\begin{bmatrix} X \\ Y \end{bmatrix}$ and $Y = [0_p, I_q]\begin{bmatrix} X \\ Y \end{bmatrix}$, then use 1..

3. See Exercises sheet 1, ex. 1.

4. See Exercises sheet 1, ex. 2.

■

2.1.2 Kalman filtering equations

The goal is to determine recursively the conditional distributions $\mathcal{L}(X_k|y_{1:k})$, in the Gaussian linear model introduced in Chapter 1,

$$\begin{cases} X_0 & \sim \mathcal{N}(m_0, P_0) \\ X_k & = A_{k-1}X_{k-1} + \epsilon_{k-1} \\ Y_k & = B_k X_k + \eta_k \end{cases}, \tag{2.1}$$

with

- $P_0 \in \mathbb{R}^{p\times p}$ is a positive definite matrix, $m_0 \in \mathbb{R}^p$
- $(A_k)_{k\geq 0} \in \mathbb{R}^{p\times p}$, $B_k \in \mathbb{R}^{q\times p}$ are known and deterministic (constants, not random variables)
- $\epsilon_k,\ \eta_k$ are Gaussian noises: $\epsilon_k \sim \mathcal{N}(0, Q_k)$ and $\eta_k \sim \mathcal{N}(0, R_k)$
- $\left(X_0,\ (\epsilon_k)_{k\geq 0},\ (\eta_k)_{k\geq 1}\right)$ are independent

First, notice that $\forall T \geq 1$, $(X_0, \cdots, X_T, Y_1, \cdots, Y_T)$ is a Gaussian vector, since it is a linear transform of the Gaussian vector $(X_0, (\epsilon_k)_{k\geq 0}, (\eta_k)_{k\geq 1})$. A a consequence, the filtering distributions $\mathcal{L}(X_k|y_{1:k})$, the predictive distributions $\mathcal{L}(X_{k+1}|y_{1:k})$ and the observation distributions $\mathcal{L}(Y_{k+1}|x_{k+1})$ are again Gaussian, in view of Proposition 2.1.1.

We denote respectively (m_k, P_k) and (m_k^-, P_k^-) the expectancy and covariance matrix for the filtering distribution $\mathcal{L}(X_k|y_{1:k})$ and the predictive distribution $\mathcal{L}(X_k|y_{1:k-1})$. Thus

$$\mathcal{L}(X_k|y_{1:k}) = \mathcal{N}(m_k, P_k)\ , \qquad \mathcal{L}(X_k|y_{1:k-1}) = \mathcal{N}(m_k^-, P_k^-) \tag{2.2}$$

The 'Kalman Filter' is and algorithm implementing the recurrence equations between (m_k, P_k) and (m_k^-, P_k^-), as summarized by the following theorem.

Theorem 2.1.2
In the GLM (2.1), the filtering and predictive parameters (m_k, P_k) and (m_k^-, P_k^-) satisfy the following relations:

1. ***(Prediction step):***
   - $m_k^- = A_{k-1} m_{k-1}$
   - $P_k^- = A_{k-1} P_{k-1} A_{k-1}^T + Q_{k-1}$
2. ***(Update step):***
   - $m_k = m_k^- + K_k(y_k - B_k m_k^-)$
   - $P_k = (I_p - K_k B_k) P_k^-$

   where $K_k = P_k^- B_k^T S_k^{-1}$ *and* $S_k = B_k P_k^- B_k^T + R_k$

The proof is left as a guided exercise (Worksheet 1). The following general result is used.

Lemma 2.1.3
Let $\mathcal{X}, \mathcal{U}, \mathcal{Z}$ be Euclidean spaces and $(X, U, Z) : \Omega \to \mathcal{X} \times \mathcal{U} \times \mathcal{Z}$ be a random variable such that $U \perp\!\!\!\perp (X, Z)$. Let $\mathcal{L}((X,U)|z)_{z\in\mathcal{Z}}$ (resp. $\mathcal{L}(X|z)_{z\in\mathcal{Z}}$) denote the conditional distribution of (X, U) (resp. X) given Z. For $z \in \mathcal{Z}$, let $\tilde{X}_z$ be a random variable such that $\tilde{X}_z \perp\!\!\!\perp U$ and $\tilde{X}_z \sim \mathcal{L}(X|z)$. Then P_Z-almost surely,

$$\mathcal{L}((X,U)|z) = \mathcal{L}(\tilde{X}_z, U) \tag{2.3}$$

Proof. It is enough to show that for continuous, bounded functions f, h,

$$\mathbb{E}(f(X,U)h(Z)) = \int_{\mathcal{Z}} \left(\int_{\mathcal{X}\times\mathcal{U}} f(x,u) \quad \mathrm{d}\Big[P_{X|z} \otimes P_U\Big](x,u) \right) h(z)\, \mathrm{d}P_Z(z). \tag{2.4}$$

By independence between U and (X, Z), and an application of Fubini, we have

$$\begin{aligned}
\mathbb{E}(f(X,U)h(Z)) &= \int_u \left(\int_{x,z} f(x,u)h(z)\, \mathrm{d}P_{X,Z}(x,z) \right) \mathrm{d}P_U(u) \\
&= \int_u \left(\int_z \underbrace{\int_x f(x,u)\ \mathrm{d}P_{X|z}(x)\ h(z)}_{=\Phi(z,u)}\ \mathrm{d}P_Z(z) \right) \mathrm{d}P_U(u) \quad \text{(def. conditional distr.)} \\
&= \int_z \int_u \Phi(z,u)\, \mathrm{d}P_U(u)\, \mathrm{d}P_Z(z) \quad \text{(Fubini)} \\
&= \int_z \int_u \int_x f(x,u)\ \mathrm{d}P_{X|z}(x)\ h(z)\, \mathrm{d}P_U(u)\, \mathrm{d}P_Z(z) \\
&= \int_z \left(\int_u \int_x f(x,u)\ \mathrm{d}P_{X|z}(x)\ \mathrm{d}P_U(u) \right) h(z)\, \mathrm{d}P_Z(z)
\end{aligned}$$

Which shows (2.4) and completes the proof. ■

Corollary 2.1.4
Under the hypotheses of Lemma 2.1.3, For all bounded, continuous function Φ, P_Z-almost surely,

$$\mathcal{L}\left(\Phi(X,U)|z\right) \overset{p.s.}{=} \mathcal{L}(\Phi(\tilde{X}_z, U)).$$

2.2 Extended Kalman filter

In this section, we extend the Kalman filter algorithm to non linear models of the kind

$$\begin{cases} X_k &= f(X_{k-1}) + \epsilon_{k-1} \\ Y_k &= g(X_k) + \eta_k \end{cases} \tag{2.5}$$

with initial distribution P_{X_0} and noises ϵ_k, η_k, Gaussian, independent.

Example 2.1 (Car tracking, angular captors)**:**
Let $Z_k = (Z_{k,1}, Z_{k,2})$ denotes the position of a car at time k, in a $2D$ Cartesian coordinate system. Let $(V_{k,1}, V_{k,2})$ denote its velocity. The (hidden) state is $X_k = (Z_{k,1}, Z_{k,2}, V_{k,1}, V_{k,2})$. The transition model is linear,

$$\begin{aligned} Z_{k+1,i} &= Z_{k,i} + hV_{k,i} + \sigma^2\zeta_{k,i} \\ V_{k+1,i} &= V_{k,i} + \sigma^2\omega_{k,i}, \qquad i = 1, 2. \end{aligned} \tag{2.6}$$

where $\epsilon_{k,i} = (\zeta_{k,i}, \omega_{k,i}) \sim \mathcal{N}\left(0, \begin{pmatrix} h^3/3 & h^2/2 \\ h^2/2 & h \end{pmatrix}\right)$, $\epsilon_{k,1} \perp\!\!\!\perp \epsilon_{k,2}$, $k \geq 0$.

The position is tracked via two angular captors recording the angles θ_1, θ_2 from which the car is seen, from their respective position $(0, 0)$ and $(0, L)$. For simplicity it is assumed that the car remains in the right half-plane $\{z_1 > 0\}$. An observation noise must be accounted for, so that the observation model for $Y_k = (Y_{k,1}, Y_{k,2})$ is non linear,

$$\begin{cases} Y_{k,1} &= \theta_{k,1} + \eta_{k,1} &= \mathsf{atan}(Z_{k,2}/Z_{k,1}) + \eta_{k,1} \\ Y_{k,2} &= \theta_{k,2} + \eta_{k,2} &= \mathsf{atan}((Z_{k,2} - L)/Z_{k,1}) + \eta_{k,2} \end{cases} \tag{2.7}$$

In other terms, $Y_k = g(X_k) + \eta_k$, with g a non linear function.

2.2.1 Gaussian approximations by linearisation

Let $X \sim \mathcal{N}(m, P)$ be a p-dimensional Gaussian vector and let g be a non linear function $\mathbb{R}^p \mapsto \mathbb{R}^q$. We assume that g is differentiable, *i.e.,* for all $x \in \mathbb{R}^p$, $\exists$ a linear function $dg_x : \mathbb{R}^p \mapsto \mathbb{R}^q$, such that

$$g(x + h) = g(x) + dg_x(h) + o(\|h\|),$$

as $\|h\| \to 0$.

Let $\eta \sim \mathcal{N}(0, R)$ be a noise, $\eta \perp\!\!\!\perp X$. Put

$$Y = g(X) + \eta.$$

Then Y is not Gaussian but may be approximated by a Gaussian variable as follows.

Write $\delta X = X - m$ and $B = dg(m)$. Then

$$g(X) = g(m) + B\delta X + o(\|\delta X\|)$$

Provided δX is small (i.e. its variance is small), we may use the approximation $Y \approx \tilde{Y} = g(m) + B\delta X$. Then

$$\begin{bmatrix} X \\ \tilde{Y} \end{bmatrix} = \begin{bmatrix} m + \delta X \\ g(m) + B\ \delta X \end{bmatrix}$$

Then $(X, \tilde{Y})$ is Gaussian, as a linear transform of the Gaussian vector δX. The vector $(X, \tilde{Y})$ is called a *Gaussian approximation* of (X, Y). The general properties of Gaussian vectors easily give the joint distribution of $(X, \tilde{Y})$, namely

$$\begin{bmatrix} X \\ \tilde{Y} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m \\ g(m) \end{bmatrix}, \begin{bmatrix} P & PB^T \\ BP & BPB^T + R \end{bmatrix} \right) \tag{2.8}$$

2.2.2 Extended Kalman filtering equation

Consider model (2.5). The idea of the extended Kalman is to linearise the transition and observation functions f and g during respectively the prediction and updating step. The arguments leading to the Kalman filtering equations, combined with equation 2.8 will result in analogous recurrence relations between linear approximations of the filtering distribution.

Prediction step Assume $\mathcal{L}(X_{k-1}|y_{1:k-1}) = \mathcal{N}(m_{k-1}, P_{k-1})$. Introduce a phantom random variable $W_k \sim \mathcal{N}(m_{k-1}, P_{k-1})$, such that $W_k \perp\!\!\!\perp \epsilon_{k-1}$. Then, using Lemma 2.1.3 and the fact that $\epsilon_k \perp\!\!\!\perp (X_k, Y_{1:k-1})$,

$$\mathcal{L}\left(\begin{matrix} X_{k-1} \\ X_k \end{matrix} \middle| y_{1:k-1} \right) = \mathcal{L}\left(\begin{matrix} X_{k-1} \\ f(X_{k-1}) + \epsilon_{k-1} \end{matrix} \middle| y_{1:k-1} \right) = \mathcal{L}\left(\begin{matrix} W_{k-1} \\ f(W_{k-1}) + \epsilon_{k-1} \end{matrix} \right)$$

According to (2.8), and denoting $A = df_{m_{k-1}}$, the Gaussian approximation of the r.v. $\Big(W_{k-1}, f(W_{k-1}) + \epsilon_{k-1}\Big)$ on the right-hand side is

$$\begin{pmatrix} m_{k-1} + \delta W_k \\ f(m_{k-1}) + A\delta W_{k-1} + \epsilon_{k-1} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} m_{k-1} \\ m_k^- \end{pmatrix}, \begin{pmatrix} P_{k-1} & P_{k-1}A^\top \\ AP_{k-1} & P_k^- \end{pmatrix} \right)$$

with

$$m_k^- = f(m_{k-1})\,; \quad P_k^- = AP_{k-1}A^T + Q_{k-1} \tag{2.9}$$

Marginalizing and getting back to the variable of interest X_k, we obtain the approximation

$$\mathcal{L}(X_k|y_{1:k-1}) \approx \mathcal{N}(m_k^-, P_k^-).$$

Update step : The line of reasoning is similar: write

$$\mathcal{L}\left(\begin{matrix} X_k \\ Y_k \end{matrix} \middle| y_{1:k-1} \right) = \mathcal{L}\left(\begin{matrix} X_k \\ g(X_k) + \eta_k \end{matrix} \middle| y_{1:k-1} \right) = \mathcal{L}\left(\begin{matrix} W_k \\ g(W_k) + \eta_k \end{matrix} \right)$$

where $W_k \sim \mathcal{N}(m_k^-, P_k^-)$. Writing $B = dg_{m_k^-}$, the Gaussian approximation of $(W_k, g(W_k) + \eta_k)$ is

$$\begin{pmatrix} m_k^- + \delta W_k \\ g(m_k^-) + B\delta W_k + \eta_k \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} m_k^- \\ g(m_k^-) \end{pmatrix}, \begin{pmatrix} P_k^- & P_k^- B_k^\top \\ B_k P_k^- & S \end{pmatrix} \right)$$

with

$$S = BP_k^- B^T + R_k \tag{2.10}$$

Conditioning the first variable upon the second and getting back to the original variable X_k gives us

$$\mathcal{L}(X_k|y_{1/k}) \approx \mathcal{N}(m_k, P_k) \tag{2.11}$$

with

$$\begin{cases} m_k &= m_k^- + K(y_k - m_k^-) \\ P_k &= (\mathbf{I} - KB)P_k^- \\ K &= P_k^- B^\top S^{-1} \\ S &\quad \text{as in (2.10)} \end{cases} \tag{2.12}$$

The Extended Kalman filter algorithm is thus the same as the regular Kalman filter, up to replacing the matrices A_k, B_k from the linear model with differentials $df_{m_{k-1}}$ and $\mathrm{d}g_{m_k^-}$.

Chapter 3

Particle methods

3.1 Introduction: Monte-Carlo methods

Goal approximate a 'target' probability distribution P over a space $\mathcal{X}$ with a discrete distribution

$$\mathrm{P}_N = \sum_{i=1}^{N} W_i \delta_{X_i} \tag{3.1}$$

where δ_a is the Dirac mass at point a, $RW_i \geq 0$ is a rnadom (or deterministic) weight, $(X_i)_{i=1:N}$ are random variables $\Omega \to \mathcal{X}$.

'Approximating' a probability distribution means considering that for any continuous, bounded function $f : \mathcal{X} \to \mathbb{R}$,

$$\int f(x)\,\mathrm{d}\mathrm{P}_N(x) \;=\; \sum_{i=1}^{N} w_i f(X_i) \;\approx\; \int f(x)\,\mathrm{d}\mathrm{P}(x) \;=\; \mathbb{E}f(X) \tag{3.2}$$

where, according to the context '$\approx$' stands for almost sure convergence, or convergence in probability as $N \to \infty$.

An Monte-Carlo method is thus characterized by a set of random points and weights,

$$\Big[X_i, W_i, i = 1, \ldots, N\Big].$$

Example 3.1 ('naïve' Monte-Carlo)**:**
Suppose one is able to draw an *i.i.d.* sample $X_i \sim \mathrm{P}$. Set $P_N = \frac{1}{N}\sum_1^N \delta_{X_i}$, *i.e.*, $w_i = 1/N$ in (3.1). Then the law of large numbers ensures that $\sum_{i=1}^N w_i f(X_i) \xrightarrow[n\to\infty]{a.s.} \int f(x)\,\mathrm{d}\mathrm{P}(x)$.

Notice that this approach will fail if one cannot easily obtain an *i.i.d.* sample for P or if the support of f lies within region of low probability P.

Example 3.2 (Sampling the tail of the Gaussian distribution)**:**
Consider the special case where $f(x) = \mathbf{1}\{x \geq u\}$ where $u \gg 1$ is a large threshold and $\mathrm{P} = \mathcal{N}(0,1)$. In other words the quantity of interest is $\tau = \mathbb{E}f(X) = \mathrm{P}[x,\infty) = \mathbb{P}(X > u)$, where $X \sim \mathcal{N}(0,1)$. Using the naïve Monte-Carlo method boils down to drawing $X_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1)$ and setting

$$\hat{\tau} = \sum_{1}^{N} \mathbf{1}\{X_i \geq u\}.$$

Doing so, only a proportion of order $\tau \ll 1$ of X_i will be above u, so that very few terms in the above sum will be non-zero and the convergence of the estimator will be very slow.

3.2 Importance Sampling (IS)

3.2.1 Principle

The inputs are a function of interest f and a target probabilty distribution P. For convenience we assume that P has a density p *w.r.t.* Lebesque. We need to estimate the unknown quantity $I(f) = \int f(x)p(x)\,\mathrm{d}x$.

The main idea is to use a 'change of measure' in order to circumvent the issue raised in Example 3.2. This means using an approximation of type (3.1), with X_i's following not P but a more convenient distribution Π with density π for the problem at hand. One needs to use appropriate weights (not $1/N$) in order to ensure that the approximation is consistent (unbiased). This 'convenient' distribution Π is called the 'proposal distribution'. It should satistfy

$$\{x : p(x)f(x) > 0\} \subset \{x : \pi(x) > 0\}, \tag{3.3}$$

up to a P- negligible set. Consider the Importance sampling scheme :

$$\Big[(X_i)_{i=1}^n \overset{\text{i.i.d.}}{\sim} \pi \ ; \ W_i = \frac{p(X_i)}{N\pi(X_i)}\Big] \tag{3.4}$$

Then $\mathbb{P}(W_i) < \infty = 1$ and for any positive or bounded function f,

$$\begin{aligned}
\mathbb{E}\,(\,NW_1 f(X_1)\,) &= \int_{\{x:\pi(x)>0\}} \frac{p(x)}{\pi(x)} f(x)\pi(x)\,\mathrm{d}x \\
&= \int_{\{x:\pi(x)>0\}} p(x)f(x)\,\mathrm{d}x \\
&= \int_{\mathcal{X}} p(x)f(x)\,\mathrm{d}x \qquad = I(f).
\end{aligned}$$

where the latter equality comes from (3.3). Thus by the law of large numbers

$$\sum_{i=1}^N W_i f(X_i) = \frac{1}{N}\sum_{i=1}^N NW_i f(X_i) \xrightarrow[n\to\infty]{p.s.} I(f).$$

We summarize, in the general case where P has a density with respect to a reference measure μ, not necessaly Lebesgue (could be the counting measure):

Definition 3.2.1 (Importance sampling estimator of an integral)**.** *Let $f : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X} \subset \mathbb{R}^d$. Let P be a distribution on $\mathcal{X}$ with density $p(x)$* w.r.t. *a reference measure μ. We assume that*

$$I(f) := \mathbb{E}_P f(X) = \int |f(x)|\, p(x)\ \mathrm{d}\mu(x) < \infty.$$

Let Π be a probability measure on $\mathcal{X}$ with density π w.r.t. *μ and let $N \in \mathbb{N}^*$. The importance sampling estimator of $I(f)$ is*

$$I_N(f) = \sum_{i=1}^N W_i f(X_i),$$

where

$$X_i \overset{i.i.d.}{\sim} \pi\ , \quad W_i = \frac{p(X_i)}{N\pi(X_i)}.$$

Proposition 3.2.2 (consistency)
In the setting of Definition 3.2.1, if π is chosen in such a way that $\{x : p(x)f(x) > 0\} \subset \{x : \pi(x) > 0\}$, then $I_N(f)$ is a strongly consistent estimator of $I(f)$, i.e.,

$$I_N(f) \xrightarrow[n\to\infty]{a.s.} I(f).$$

3.2.2 Self-normalisation

3.2.3 Re-sampling

3.3 Particle filtering

(Sequential importance sampling)

3.3.1 Principle

3.3.2 Recurrence relation on the smoothing distributions

3.3.3 Sequential sampling of the particles

3.3.4 Recurrece relation on the importance weights

3.3.5 Algorithm: SIS (Sequential Importance Sampling)

3.3.6 SIR: Sequential Importance Re-sampling

Appendix A

Introduction to Octave

MATLAB et Octave sont des environnements intégrés pour le calcul scientifique et la visualisation. Ils sont écrits principalement en langage C et C++. MATLAB est distribué par la société The MathWorks (voir le site www.mathworks.com). Son nom vient de MATrix LABoratory, car il a été initialement développé pour le calcul matriciel. Octave, aussi connu sous le nom de GNU Octave (voir le site www.octave.org), est un logiciel libre et gratuit.

Les deux langages sont en général interchangeables, cependant, il existe quelques différences mineures entre MATLAB et Octave, au niveau des environnements, des langages de programmation ou des toolboxes (collections de fonctions dédiées à un usage spécifique).

Attention: lorsque vous devrez rendre un code pour un tp noté, le correcteur l'exécutera avec octave. Ainsi, même si vous préférez travailler avec matlab, **testez votre code en l'exécutant avec octave.**

Il existe plusieurs manières de travailler avec Octave: dans des interfaces graphiques ou en ligne de commande. Une manière simple de procéder est la suivante, en ligne de commande:

- Créez un répertoire pour le projet en cours, disons TD1
- Ouvrir un terminal et se placer dans TD1. Lancer Octave:

```
anne@bibi:~/TELECOM/ENSEIGNEMENT/calculScient/2015/TD_TP/TD1$ octave

GNU Octave, version 3.8.1
Copyright (C) 2014 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE. For details, type 'warranty'.

Octave was configured for "x86_64-pc-linux-gnu".

Additional information about Octave is available at http://www.octave.org.

Please contribute if you find this software useful.
For more information, visit http://www.octave.org/get-involved.html

Read http://www.octave.org/bugs.html to learn how to submit bug reports.
For information about changes from previous versions, type 'news'.
```

```
octave:1>
```

- Créez un fichier texte `*.m` contenant toutes vos instructions, par exemple `script1.m`. Voici un exemple:

```
disp('Ceci est un premier script qui affiche le resultat de 1.5 + 3.5 .')
a = 1.5 + 3.5;
printf('le resultat est %d, bravo ! ', a)
```

- Depuis `Octave`, exécutez votre script comme ceci

```
octave:1> script1
Ceci est un premier scipt qui affiche le resulat de 1.5 + 3.5 .
le resultat est 5, bravo !
```

- Depuis l'invite de commande d'`Octave`, vous pouvez inspectez les variables définies par le script (après l'voir exécuté). Par exemple

```
octave:2> a
a =  5
```

- Si vous devez écrire des fonctions, vous devez pour chaque nouvelle fonction créer un nouveau fichier `*.m` du nom de la fonction, dans le répertoire `TD1`, contenant uniquement la définition de la fonction. Par exemple, pour définir une fonction `mafonction`, prenant comme arguments `(x,y)` et renvoyant `z`, où $z = x + y$, créer un fichier nommé `mafonction.m`, commençant par

```
function z = mafonction(x,y)
# cette fonction calcule la somme de ses deux arguments
  z = x+y ;
end
```

  Vous pouvez ensuite faire appel à `mafonction` dans votre script ou depuis le terminal `Octave`, par exemple

```
octave:3> mafonction(3.5,1.5)
ans =  5
```

- La commande **la plus utile** : `help` :

```
octave:4> help mafonction
```

```
octave:5> help plot
```

- Pour quitter: `Ctrl+d` ou `exit`

Quelques références pour démarrer avec `Octave`:

- https://www.gnu.org/software/octave/doc/interpreter/Introduction.html#Introduction : l'introduction du manuel de référence.
- http://fr.wikibooks.org/wiki/Programmation_Octave/Introduction : le wikilivre
- http://www.tutorialspoint.com/matlab/matlab_gnu_octave.htm : Un tutoriel

Bibliography

Cappé, O., Moulines, E., and Rydén, T. (2009). Inference in hidden markov models. In *Proceedings of EUSFLAT Conference*, pages 14–16.

Särkkä, S. (2013). *Bayesian filtering and smoothing*, volume 3. Cambridge University Press.