SB1.2/SM2 Computational Statistics
Lecture notes: Hidden Markov Models

François Caron

University of Oxford, Hilary Term 2019

Version of February 5, 2019

This document builds on the following references:

- D. Barber. Bayesian Reasoning and Machine Learning, Cambridge University Press, 2012.
- K.P. Murphy. Machine Learning. A probabilistic perspective. The MIT Press, 2012

More advanced references are

- R. van Handel. Hidden Markov models. Lecture notes, University of Princeton, 2008.
- O. Cappé, E. Moulines, T. Ryden. Inference in Hidden Markov Models. Springer, 2007.

The course requires the following notions:

- Discrete Markov chains [Part A Probability]
- Bayesian methods: prior, posterior, maximum a posteriori [Part A Statistics]

Please report typos to caron@stats.ox.ac.uk.

Contents

1 Motivating example

Consider that we have a sequence of observations $y_{1:T} = (y_1, y_2, \ldots, y_T)$, $T \geq 1$ where there is some natural order of the data. The index t in the sequence $(y_t)_{t=1,\ldots,T}$ may refer to time, the index of a site on a chromosome or a piece of DNA or the position of a word in a sentence. For each index $t = 1, \ldots, T$, we are interested in inferring some non-observed/hidden quantity of interest $x_t \in \mathcal{X}$ where $\mathcal{X}$ is a finite set. For illustration, consider the following problem in natural language processing, known as Part-of-Speech tagging.

Part-of-speech tagging (POST) refers to the task of labelling a word in a text corpus as a particular part of speech, such as noun, verb, adjective or adverb. An illustration is given in Figure 1.

PRON VB ADV ADJ PREP DET ADJ NOUN PREP DET ADJ COORD ADJ NOUN

Nothing is so painful to the human mind as a great and sudden change.

Figure 1: Example of Part-Of-Speech tagging. Observations $(y_1, \ldots, y_T)$ are the T words in a document, where y_t refers to the t's word in the document. One is interested in inferring the tags $(x_1, \ldots, x_T)$ where x_t is the tag associated to the t's word y_t in the sentence.

POST may be challenging, as some words such as *change* or *mind* may correspond to different parts of speech (noun/verb) depending on the context. One possibility is to treat the unknown tags x_t as fixed parameters. However, one usually has quite a lot of prior information on the hidden sequence of tags. For example, in the POST example, we know that some POS have a higher frequency of appearance than others. We also know that a sentence has some structure: a pronoun is often followed by a verb, an adjective by a noun, etc. and we may want to probabilistically encode this information in order to get better estimates. This can be done in a Bayesian framework, by assuming that the hidden tags of interest $X_1, \ldots, X_T$ are also random variables, and by considering a joint probability mass function (pmf) over the hidden and observed variables:

$$\begin{aligned} p(x_{1:T}, y_{1:T}) : &= \mathbb{P}(X_{1:T} = x_{1:T}, Y_{1:t} = y_{1:t}) \\ &= \underbrace{\mathbb{P}(Y_{1:T} = y_{1:T} | X_{1:T} = x_{1:T})}_{\text{Likelihood}} \underbrace{\mathbb{P}(X_{1:T})}_{\text{Prior}}. \end{aligned}$$

This joint probability mass function defines our **statistical model** and can capture complex dependencies between the hidden states and the observations. Given some observation sequence $(y_1, \ldots, y_T)$ the information about the hidden parameter of interest is encapsulated in the posterior probability mass function

$$\begin{aligned} p(x_{1:T} | y_{1:T}) &:= \mathbb{P}(X_{1:T} = x_{1:T} | Y_{1:T} = y_{1:T}) \\ &= \frac{\mathbb{P}(Y_{1:T} = y_{1:T} | X_{1:T} = x_{1:T}) \mathbb{P}(X_{1:T} = x_{1:T})}{\mathbb{P}(Y_{1:T} = y_{1:T})} \end{aligned}$$

From this, we can calculate a point estimate, for example the posterior mode or maximum a posteriori (MAP) estimate. This boils down to solving the following combinatorial optimization problem

$$\hat{x}_{1:T} = \underset{x_{1:T} \in \mathcal{X}^T}{\arg\max}\, p(x_{1:T} | y_{1:T}).$$

However, the combinatorial search space has $|\mathcal{X}|^T$ elements and grows exponentially fast with T. Calculating exactly the MAP estimate quickly becomes impossible even for reasonably small values of T. For example, for a document with $T = 100$ words and $|\mathcal{X}| = 20$ tags, exhaustive search requires to evaluate the $20^{100} \simeq 10^{130}$ possible sequences.

Maybe one should make some simplifying assumptions on $p(x_{1:T}, y_{1:T})$. An obvious simplification would be to assume independence between the pairs (X_t, Y_t), (X_τ, Y_τ) for any $t \neq \tau$, thus ignoring the sequential structure. In this case, the posterior pmf factorizes over t, and MAP estimation reduces to solving independently

$$\hat{x}_t = \underset{x_t \in \mathcal{X}}{\arg\max}\, \mathbb{P}(Y_t = y_t | X_t = x_t) \mathbb{P}(X_t = x_t), \quad t = 1, \ldots, T,$$

which has a linear complexity $T|\mathcal{X}|$ in both T and $|\mathcal{X}|$, hence a combinatorial search space of size 2000 in the previous example. We now have a statistical model for which we can compute the MAP estimate. But the statistical model, although it can incorporate prior information about the frequency of each tag, appears to be too simplistic for the POST task. By considering each word independently, the estimated tag will be the same for multiple occurrences of the same word, which is clearly inappropriate for words like *mind* or *change*.

In conclusion, considering a full model $p(x_{1:T}, y_{1:T})$ may give a realistic probabilistic representation of the data and hidden variables, but is practically useless as the estimate cannot be calculated. Using a much simpler statistical model which assumes independence across time allows to compute the estimate, but is too simplistic to address the task. **Hidden Markov Models** is a class of models for sequential data that offers a very attractive trade-off between the model's ability to capture dependencies and the tractability of the estimation algorithms. It is important to keep in mind that HMMs, like other statistical models, are in general not meant to reproduce the true data generating process. They are an interpretation and approximation of the real world, targeted to the problem at hand. As George Box famously wrote in his 1987 book

> *Remember that all models are wrong; the practical question is how wrong do they have to be to not be useful.[...]*
>
> *Essentially, all models are wrong, but some are useful.*

For many problems involving sequential data, Hidden Markov Models are indeed very useful, if not realistic, statistical models!

2 Discrete-state Hidden Markov models

2.1 Recap: Discrete Markov chain

Let $X_{0:T} = (X_0, X_1, \dots, X_T)$ be a sequence of random variables (random process) taking values in some finite set $\mathcal{X}$ called the state-space. The process is called a **Markov chain** if for any $t \geq 0$ and any $x_0, \dots, x_{t+1} \in \mathcal{X}$,

$$\mathbb{P}(X_{t+1} = x_{t+1} | X_t = x_t, \dots, X_0 = x_0) = \mathbb{P}(X_{t+1} = x_{t+1} | X_t = x_t) \tag{1}$$

The Markov chain is said to be homogeneous if $\mathbb{P}(X_{t+1} = j | X_t = i)$ does not depend on t. In that case, we write

$$A_{i,j} := \mathbb{P}(X_{t+1} = j | X_t = i) \quad i, j \in \mathcal{X}$$

For simplicity of exposure, we will only consider homogeneous Markov chains, but the algorithms can also be derived in the non-homogenous case as well. For $x_0 \in \mathcal{X}$, let $\mu_{x_0} = \mathbb{P}(X_0 = x_0)$ be the pmf of the initial state X_0. The joint pmf of $X_{1:T}$ is parameterized by $(A_{i,j})_{i,j \in \mathcal{X}}$ and $(\mu_i)_{i \in \mathcal{X}}$

$$\begin{aligned} p(x_{0:T}) &:= \mathbb{P}(X_0 = x_0, \dots, X_T = x_t) \\ &= \mathbb{P}(X_0 = x_0) \prod_{t=1}^{T} \mathbb{P}(X_t = x_t | X_{t-1} = x_{t-1}) \\ &= \mu_{x_0} \prod_{t=1}^{T} A_{x_{t-1}, x_t} \end{aligned}$$

2.2 Hidden Markov model

Let $X_{0:T} = (X_0, X_1, \dots, X_T)$ be a homogeneous Markov chain taking values in $\mathcal{X}$ with transition matrix (A_{ij}). Consider another sequence of random variables $Y_{1:T} = (Y_1, \dots, Y_T)$ taking values in some set $\mathcal{Y}$ called the **observation space**. The random variables Y_t may be continuous or discrete. We assume that the random variables $(Y_1, \dots, Y_T)$ are **independent** conditional on the state sequence $(X_0, X_1, \dots, X_T)$. For discrete random variables Y_t

$$\mathbb{P}(Y_1 = y_1, \dots, Y_T = y_T | X_0 = x_0, \dots, X_T = x_T) = \prod_{t=1}^{T} \mathbb{P}(Y_t = y_t | X_t = x_t)$$

If the conditional probability $\mathbb{P}(Y_t = y_t | X_t = x_t)$ does not depend on t, then the HMM is said to be homogeneous. We write, for $x \in \mathcal{X}$ and $y \in \mathcal{Y}$

$$g_x(y) := \mathbb{P}(Y_t = y | X_t = x)$$

where $g_x(y)$ is called the **emission** probability mass function.

For continuous random variables Y_t, we use the same notation $g_x(y)$ for the probability density function of $Y_t | X_t$ defined as

$$\Pr(Y_t \leq y | X_t = x) = \int_{-\infty}^{y} g_x(\widetilde{y}) d\widetilde{y}.$$

To simplify exposure, we will only consider discrete observations in the rest of Section 2. Using the Markov and conditional independence properties of the HMM, the joint probability of the hidden states and observations factorizes as

$$\mathbb{P}(X_{0:T} = x_{0:T}, Y_{0:T} = y_{0:T}) = \mu_{x_0} \prod_{t=1}^{T} g_{x_t}(y_t) A_{x_{t-1},x_t}$$

A graphical representation of the HMM is given in Figure 2.

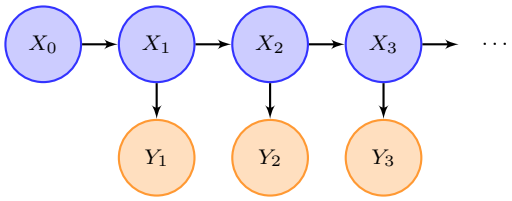

Figure 2: Graphical representation of a hidden Markov model. Hidden states are represented with blue circles, and observations with orange circles. An arrow from node A to node B indicates that A is a parent of B. For example parents$(Y_2) = X_2$. The figure encapsulates conditional independence relations in the sense that $p(x_{0:T}, y_{1:T}) = p(x_0|\text{parents}(x_0)) \prod_{t=1}^{T} p(x_t|\text{parents}(x_t))p(y_t|\text{parents}(y_t))$. For more details on graphical models, see the Part C/MSc course on graphical models.

2.3 Some examples

Part-of-Speech Tagging. In natural language processing, part-of-speech tagging (POST) refers to the task of labelling a word in a text corpus as a particular part of speech, such as noun, verb, adjective or adverb. POST may be challenging, as some words may correspond to different parts of speech depending on the context. Hidden Markov models have been used for POST as they allow to take into account the structure of the language.

The observation space $\mathcal{Y}$ is the set of words (vocabulary) and the state-space $\mathcal{X}$ is the set of tags. Y_t refers to the observed word at position t in the sentence, and X_t its unknown tag.

Robot localisation. Consider a robot equipped with a map of his environment and some sensors (e.g. sound sensors) which enable it to detect obstacles. The objective of the robot is to self localize himself in the map, based on noisy measurements and map of the environment. The state space $\mathcal{X}$ is the position of the robot on a grid. The observation state is $\mathcal{Y} = \{0, 1\}$ to indicate if it has detected an obstacle or not. X_t refers to the position of the robot at time t on the grid, and Y_t the detection/non-detection of an obstacle. The objective is to calculate over time the probability $\mathbb{P}(X_t = x_t|Y_{1:T} = y_{1:T})$ that the robot is in a given cell at time t given the measurements up to time t.

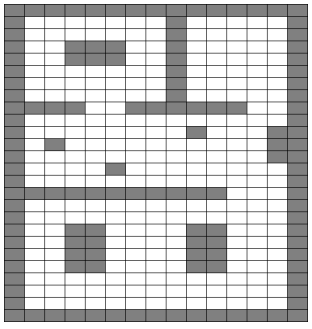

Figure 3: Robot localization: A robot needs to infer its position X_t on a grid-based map based on noisy measurements.

Gene finding. The genetic material of an organism is encoded in DNA, a long polymer which consists of a sequence of base pairs made of four chemical bases: adenine (A), guanine (G), cytosine (C) and thymine (T). The genetic sequence is made of coding and non-coding sub-sequences. Coding sub-sequences encode proteins and the task of separating coding and non-coding sequences of DNA is known as gene finding and is an important problem in computational biology. The state space is $\mathcal{X} = \{0, 1\}$ where 0 indicates a coding

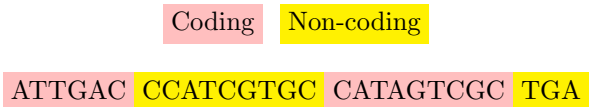

Figure 4: Illustration of gene finding.

region and 1 a non-coding region. Observations are the type of the base pair, encoded with the four-letters state-space $\mathcal{Y} = \{A, C, G, T\}$. Observations Y_t are the base pair at the location t in the genome, and $X_t \in \{0, 1\}$ in the hidden state (coding/non-coding). Based on the DNA sequence $Y_{0:T}$, we aim at inferring the most likely sequence $X_{0:T}$.

2.4 Inference in HMM

For simplicity of exposure, we will only consider discrete-valued observations Y_t, but the algorithms apply similarly with continuous observations. We will use the following notations

$$\begin{aligned} p(x_{t+1}|x_t) &= \mathbb{P}(X_{t+1} = x_{t+1}|X_t = x_t) \\ p(y_t|x_t) &= \mathbb{P}(Y_t = y_t|X_t = x_t) \\ p(x_t|y_{1:t}) &= \mathbb{P}(X_t = x_t|Y_1 = y_1, \ldots, Y_t = y_t) \\ p(y_{1:t}) &= \mathbb{P}(Y_1 = y_1, \ldots, Y_t = y_t) \end{aligned}$$

etc., where the subscripts indicate which random variables we are referring to.

Assume that we have a sequence of observations $(y_1, \ldots, y_T)$. The classical inference problems are the following:

- Filtering
$$p(x_t|y_{1:t})$$
- Prediction
$$p(x_t|y_{1:s}), \quad s < t$$
- Smoothing
$$p(x_t|y_{1:s}), \quad s > t$$
- Likelihood
$$p(y_{1:T})$$
- Most likely state path
$$\arg\max_{x_{0:T}} p(x_{0:T}|y_{1:T})$$

2.4.1 Forward filtering

We are interested in the conditional probability mass function $p(x_t|y_{1:t})$ of the state X_t given the data observed up to time t. Note that, by Bayes rule, $p(x_t|y_{1:t})$ can be obtained by normalizing $p(x_t, y_{1:t})$

$$p(x_t|y_{1:t}) = \frac{p(x_t, y_{1:t})}{\sum_{x'_t \in \mathcal{X}} p(x'_t, y_{1:t})}$$

We will derive a recursion for $p(x_t, y_{1:t})$.

$$\begin{aligned} p(x_t, y_{1:t}) &= \sum_{x_{t-1} \in \mathcal{X}} p(x_t, x_{t-1}, y_t, y_{1:t-1}) \\ &= \sum_{x_{t-1} \in \mathcal{X}} p(y_t|x_t, x_{t-1}, y_{1:t-1}) p(x_t|x_{t-1}, y_{1:t-1}) p(x_{t-1}, y_{1:t-1}) \\ &= p(y_t|x_t) \sum_{x_{t-1} \in \mathcal{X}} p(x_t|x_{t-1}) p(x_{t-1}, y_{1:t-1}) \end{aligned}$$

Define $\alpha_t(x_t) = p(x_t, y_{1:t})$. The above equation defines the **α-recursion**. For $t = 1, \ldots, T$, $x_t \in \mathcal{X}$

$$\alpha_t(x_t) = p(y_t|x_t) \sum_{x_{t-1} \in \mathcal{X}} p(x_t|x_{t-1}) \alpha_{t-1}(x_{t-1})$$

with $\alpha_0(x_0) = p(x_0)$. The forward recursion is given in Algorithm 1 for $\mathcal{X} = \{1, \ldots, K\}$. The filtering pmf is obtained by normalizing $\alpha_t(x_t)$ as

$$p(x_t|y_{1:t}) = \frac{p(x_t, y_{1:t})}{p(y_{1:t})} = \frac{\alpha_t(x_t)}{\sum_{x\in\mathcal{X}} \alpha_t(x)}.$$

The likelihood term $p(y_{1:T})$ can be computed from the α-recursion

$$p(y_{1:T}) = \sum_{x\in\mathcal{X}} \alpha_T(x)$$

Algorithm 1 Forward α-recursion

- For $i = 1, \ldots, K$, set $\alpha_0(i) = \mu_i$
- For $t = 1, \ldots, T$
  - For $j = 1, \ldots, K$, set

$$\alpha_t(j) = g_j(y_t) \sum_{i=1}^{K} A_{i,j}\alpha_{t-1}(i)$$

The computation cost of the whole forward recursion is $O(T|\mathcal{X}|^2)$. Note that the proposed recursion may suffer from numerical underflow/overflow, as α_t may become very small or very large for large t. To avoid this, we can normalize α_t, or propagate the filtering pmf $p(x_t|y_{1:t})$ instead of α_t, using the following two-step predict-update recursion

$$p(x_t|y_{1:t-1}) = \sum_{x_{t-1}\in\mathcal{X}} p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1}) \qquad \text{Predict}$$

$$p(x_t|y_{1:t}) = \frac{g_{x_t}(y_t)p(x_t|y_{1:t-1})}{\sum_{x'_t\in\mathcal{X}} g_{x'_t}(y_t)p(x'_t|y_{1:t-1})} \qquad \text{Update}$$

2.4.2 Forward-backward Smoothing

We are now interested in the conditional probability mass function

$$p(x_t|y_{1:T})$$

of the state X_t given all the data from time 1 to $T \geq t$. First note that

$$\begin{aligned} p(x_t|y_{1:T}) &= \frac{p(x_t, y_{1:T})}{p(y_{1:T})} \\ &= \frac{p(x_t, y_{1:t})p(y_{t+1:T}|x_t)}{p(y_{1:T})} \end{aligned}$$

hence $p(x_t|y_{1:T})$ can be obtained by normalizing $p(x_t, y_{1:t})p(y_{t+1:T}|x_t)$. The first term is $\alpha_t(x_t)$ which can be obtained by a forward recursion. The second term $\beta_t(x_t) = p(y_{t+1:T}|x_t)$ can be obtained by a backward recursion.

$$\begin{aligned} p(y_{t:T}|x_{t-1}) &= \sum_{x_t\in\mathcal{X}} p(y_{t:T}, x_t|x_{t-1}) \\ &= \sum_{x_t\in\mathcal{X}} p(y_t|y_{t+1:T}, x_t, x_{t-1})p(y_{t+1:T}, x_t|x_{t-1}) \\ &= \sum_{x_t\in\mathcal{X}} p(y_t|x_t)p(y_{t+1:T}|x_t, x_{t-1})p(x_t|x_{t-1}) \\ &= \sum_{x_t\in\mathcal{X}} p(y_t|x_t)p(y_{t+1:T}|x_t)p(x_t|x_{t-1}) \end{aligned}$$

Hence β_t follows the following backward recursion for $t = T, \ldots, 2$

$$\beta_{t-1}(x_{t-1}) = \sum_{x_t\in\mathcal{X}} p(y_t|x_t)p(x_t|x_{t-1})\beta_t(x_t)$$

with $\beta_T(x_T) = 1$. The backward recursion is given in Algorithm 2 when $\mathcal{X} = \{1, \ldots, K\}$.

The forward and backward recursions can be run independently. The smoothing pmf is finally obtained by normalization

$$p(x_t|y_{1:T}) = \frac{p(x_t, y_{1:T})}{p(y_{1:T})} = \frac{\alpha_t(x_t)\beta_t(x_t)}{\sum_{x\in\mathcal{X}} \alpha_t(x)\beta_t(x)}$$

The overall computational cost of the forward-backward algorithm is $O(T|\,\mathcal{X}\,|^2)$.

Algorithm 2 Backward β-recursion

- For $i = 1, \ldots, K$, set $\beta_T(i) = 1$
- For $t = 1, \ldots, T$
  - For $i = 1, \ldots, K$, set

$$\beta_{t-1}(i) = \sum_{j=1}^{K} g_j(y_t) A_{i,j} \beta_t(j)$$

2.4.3 Maximum a posterior estimation

We are interested in the Maximum a posterior estimate

$$\hat{x}_{0:T} = \arg\max_{x_{0:T}} p(x_{0:T}|y_{1:T})$$

or equivalently, for fixed $y_{1:t}$

$$\hat{x}_{0:T} = \arg\max_{x_{0:T}} p(x_{0:T}, y_{1:T})$$

Note that direct optimization would quickly become unfeasible as the number of different state paths is $|\,\mathcal{X}\,|^{T+1}$. The MAP estimate can be calculated efficiently using the **Viterbi** algorithm, which uses a backward-forward (or forward-backward) recursion and is a special case of the so-called max-product algorithm. The algorithm first performs a backward path which computes messages m_t, $t = T, \ldots, 0$. Then it performs a forward path to return the estimates $\widehat{x}_t$, for $t = 0, \ldots, T$.

$$\begin{array}{lccccccccc}
\text{Backward} & m_0(x_0) & \longleftarrow & m_1(x_1) & \longleftarrow & \ldots & \longleftarrow & m_{T-1}(x_{T-1}) & \longleftarrow & m_T(x_T) \\
& \downarrow & & \downarrow & & & & \downarrow & & \downarrow \\
\text{Forward} & \widehat{x}_0 & \longrightarrow & \widehat{x}_1 & \longrightarrow & \ldots & \longrightarrow & \widehat{x}_{T-1} & \longrightarrow & \widehat{x}_T
\end{array}$$

First note that

$$p(x_{0:T}, y_{1:T}) = p(x_0) \prod_{t=1}^{T} p(x_t|x_{t-1}) p(y_t|x_t)$$

and

$$\begin{aligned}
\max_{x_{0:T}} p(x_0) \prod_{t=1}^{T} p(x_t|x_{t-1}) p(y_t|x_t) &= \max_{x_{0:T-1}} \left\{ \left[p(x_0) \prod_{t=1}^{T-1} p(x_t|x_{t-1}) p(y_t|x_t) \right] \max_{x_T} p(x_T|x_{T-1}) p(y_T|x_T) \right\} \\
&= \max_{x_{0:T-1}} \left\{ \left[p(x_0) \prod_{t=1}^{T-1} p(x_t|x_{t-1}) p(y_t|x_t) \right] m_{T-1}(x_{T-1}) \right\}
\end{aligned}$$

where $m_{T-1}(x_{T-1}) = \max_{x_T} p(x_T|x_{T-1}) p(y_T|x_T)$ is the message from the end of the chain to the penultimate timestep.

We can continue in this manner. For $t = T-1, \ldots, 1$, let

$$m_{t-1}(x_{t-1}) = \max_{x_{t:T}} \left\{ \prod_{k=t}^{T} p(x_k|x_{k-1}) p(y_k|x_k) \right\}$$

and $m_T(x_T) = 1$. $m_{t-1}(x_{t-1})$ satisfies the following backward recursion for $t = T-1, \ldots, 1$

$$m_{t-1}(x_{t-1}) = \max_{x_t} p(y_t|x_t) p(x_t|x_{t-1}) m_t(x_t).$$

Note that

$$p(x_0)m_0(x_0) = \max_{x_{1:T}} p(x_{0:T}, y_{1:T})$$

Hence

$$\begin{aligned}\hat{x}_0 &= \arg\max_{x_0} \left(\max_{x_{1:T}} p(x_0, x_{1:T}, y_{1:T}) \right) \\ &= \arg\max_{x_0} m_0(x_0)p(x_0)\end{aligned}$$

Similarly,

$$\begin{aligned}\hat{x}_t &= \arg\max_{x_t} \left(\max_{x_{t+1:T}} p(\hat{x}_{0:t-1}, x_t, x_{t+1:T}, y_{1:T}) \right) \\ &= \arg\max_{x_t} \left(\max_{x_{t+1:T}} p(\hat{x}_{t-1}, x_t, x_{t+1:T}, y_{t:T}) \right) \\ &= \arg\max_{x_t} \left(m_t(x_t)p(y_t|x_t)p(x_t|\hat{x}_{t-1}) \right).\end{aligned}$$

The overall Viterbi algorithm is given in Algorithm 3. The computational complexity of the Viterbi algorithm is $O(T|\mathcal{X}|^2)$, the same as the forward-backward recursion. For numerical stability, logarithms are computed in practice.

Algorithm 3 Viterbi algorithm for maximum a posteriori estimation

- For $i = 1, \ldots, K$, set $m_T(i) = 1$.
- For $t = T, \ldots, 1$
  - For $i = 1, \ldots, K$, let
$$m_{t-1}(i) = \max_{j=1,\ldots,K} g_j(y_t)A_{i,j}m_t(j)$$
- Set $\hat{x}_0 = \arg\max_{i=1\ldots,K} \; m_0(i)\mu(i)$
- For $t = 1, \ldots, T$
  - Set
$$\hat{x}_t = \arg\max_{i=1\ldots,K} \; m_t(i)g_i(y_t)A_{\hat{x}_{t-1},i}.$$

2.4.4 Illustration

We consider the following illustrative example. Consider a frog on a ladder with K levels, and let X_t be the level at which the frog is at time t. We consider the following transition matrix

$$\begin{aligned}A_{i,i+1} &= \frac{1-p}{2} \quad \text{for } i = 1, \ldots, K-1 \\ A_{i,i} &= p \quad \text{for } i = 1, \ldots, K \\ A_{i,i-1} &= \frac{1-p}{2} \quad \text{for } i = 2, \ldots, K\end{aligned}$$

and $A_{1,2} = 1-p$ and $A_{K,1} = \frac{1-p}{2}$, $p = 0.4$. The frog's position is not observed, but a frog's detector is installed at the lowest level of the ladder, which sends a signal $Y_t \in \{1, 2\}$ at each time t where 2 indicates detection and 1 non-detection. The probability of detection is as follows

$$B_{k,2} := \mathbb{P}(Y_t = 2|X_t = k) = \begin{cases} 0.9 & \text{if } k = 1 \\ 0.5 & \text{if } k = 2 \\ 0.1 & \text{if } k = 3 \\ 0 & \text{Otherwise} \end{cases}$$

Assume that we observe the following sequence $y_{1:14} = (1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 2, 2, 1, 2)$ and want to infer the filtering and smoothing pmfs of the frog's position at each type t as well as the MAP estimate.

The code for the *alpha* recursion and filtering pmf is as follows.

```
alpha_recursion = function(y, mu, A, B)
{
  K = length(mu)
  T = length(y)
  alpha = matrix(0, nrow=T,ncol=K)
  for (j in 1:K) alpha[1,j] = B[j,y[1]] *sum(A[,j]* mu)
  for (t in 2:T) for (j in 1:K) alpha[t,j] = B[j,y[t]] *sum(A[,j]* alpha[t-1,])
  return(alpha)
}
K = 6
p = .4
# Transition matrix
A = diag(x=p,K,K)
A[1,2] = 1-p
for (i in 2:K){
  A[i, (i %% K)+1] = (1-p)/2
  A[i, (i-1)] = (1-p)/2 }

# Observation matrix
B = matrix(data=NA, nrow=K, ncol=2)
B[1, 2] = .9; B[2, 2] = .5; B[3,2] = .2; B[4:K,2] = 0; B[,1] = 1-B[,2]
mu = 1/K*rep(1,K)

y = c(1,1,1,1,2,2,1,1,1,1,2,2, 1, 2)
T = length(y)

# compute alpha recursion
alpha = alpha_recursion(y, mu, A, B)
# Compute filtering pmf
filtering = matrix(data=NA, nrow=T,ncol=K)
for (t in 1:T) filtering[t,] = alpha[t,]/sum(alpha[t,])

# Plot results
time = matrix(rep(c(1:T),K), T, K)
x = matrix(rep(c(1:K),each=T), T, K)
plot(time, x, cex=5*filtering,bg='lightblue2',pch=21,xlim=c(0.6, T+.5),ylim=c(.5, K+.6))
points(time[,1],rep(.4, T), cex=(y-1)*2, pch=22, bg='red')
```

Now for the smoothing pmf.

```
beta_recursion = function(y, mu, A, B)
{
  K = length(mu)
  T = length(y)
  beta = matrix(0, nrow=T,ncol=K)
  for (j in 1:K) beta[T,j] = 1
  for (t in T:2) for (i in 1:K) beta[t-1,i] =  sum(B[,y[t]]*A[i,]* beta[t,])
  return(beta)
}
beta = beta_recursion(y, mu, A, B)
smoothing = matrix(data=NA, nrow=T,ncol=K)
for (t in 1:T)  smoothing[t,]=alpha[t,]*beta[t,]/sum(alpha[t,]*beta[t,])
plot(time, x, cex=5*smoothing,bg='lightblue2',pch=21,xlim=c(0.6, T+.5),ylim=c(.5, K+.6))
points(time[,1],rep(.4, T), cex=(y-1)*2, pch=22, bg='red')
```

And the Viterbi algorithm to obtain a MAP estimate.

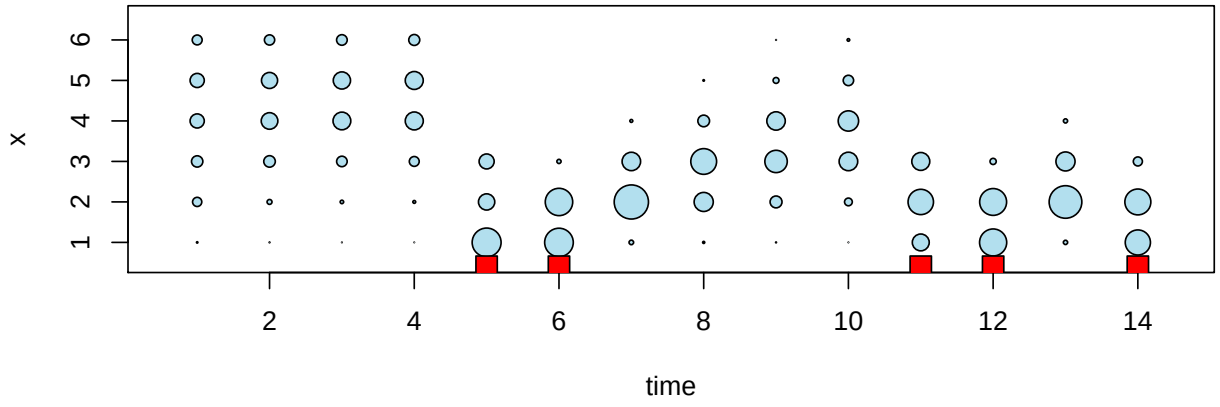

Figure 5: Filtering pmf over time t. The size of each circle at location (t, x) is proportional to $P(X_t = x|y_{1:t})$. Red squares indicate times at which detection occurs.

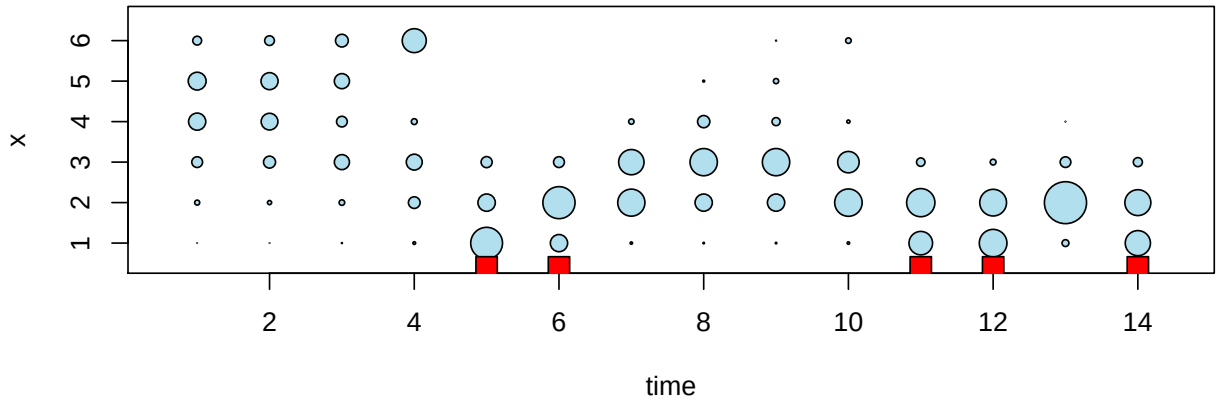

Figure 6: Smoothing pmf over time t. The size of each circle at location (t, x) is proportional to $P(X_t = x|y_{1:14})$. Red squares indicate times at which detection occurs.

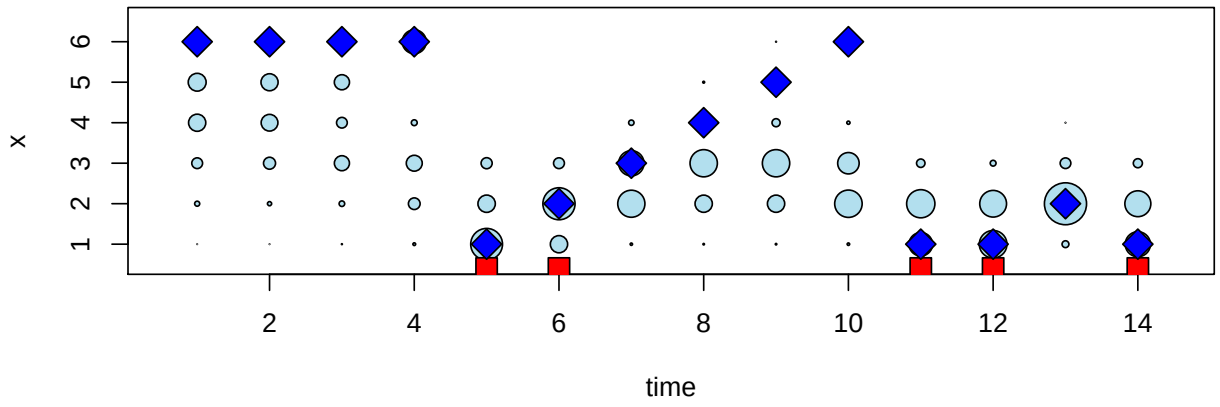

Figure 7: Smoothing pmf over time t and MAP estimate. The size of each circle at location (t, x) is proportional to $P(X_t = x|y_{1:14})$. Dark blue diamonds indicate the MAP estimate. Red squares indicate times at which detection occurs.

```
viterbi = function(y, mu, A, B)
{
  K = length(mu)
  T = length(y)
  m = matrix(0, nrow=T,ncol=K)
  m0 = matrix(0, nrow=1,ncol=K)
  x.map = rep(0, T)

  # Backward
  for (i in 1:K) m[T,i] = 1
  for (t in T:2) for (i in 1:K) m[t-1,i] =  max(B[,y[t]]*A[i,]* m[t,])
  for (i in 1:K) m0[i] =  max(B[,y[1]]*A[i,]* m[1,])

  #Forward
  x0.map = which.max(m0 * mu)
  x.map[1] = which.max(m[1,]*B[,y[1]]*A[x0.map,])
  for (t in 2:T) x.map[t] = which.max(m[t,]*B[,y[t]]*A[x.map[t-1],])
  return(x.map)
}
x.map=viterbi(y, mu, A, B)

plot(time, x, cex=5*smoothing,bg='lightblue2',pch=21,xlim=c(0.6, T+.5),ylim=c(.5, K+.6))
points(time[,1],rep(.4, T), cex=(y-1)*2, pch=22, bg='red')
points(time[,1],x.map, cex=2, pch=23, bg='blue')
```

2.5 Learning in HMM

So far we have assumed that the parameters A, μ and g of the HMMs were known. This is not the case in general. We can differentiate two cases

- The fully observed case: we have a dataset where the hidden states $(x_0, x_1, \ldots, x_T)$ are known
- The unsupervised case: all we have is the data $(y_1, \ldots, y_T)$ and the hidden variables are not observed

For simplicity, we only consider estimation of the transition matrix A.

2.5.1 Fully observed case

If the hidden states $(x_0, x_1, \ldots, x_T)$ are known, the parameter A can be fitted using maximum likelihood. Let $n_{i,j} = \sum_{t=1}^{T} I(x_t = j, x_{t-1} = i)$ be the number of transitions between state i and state j. The MLE of $A_{i,j}$ is

$$\widehat{A}_{i,j} = \frac{n_{i,j}}{\sum_{\ell \in \mathcal{X}} n_{i,\ell}}.$$

2.5.2 Unsupervised case

If the hidden states are not observed, finding the MLE is much more challenging as we want to optimize

$$\widehat{A} = \arg\max_{A} \ \log p(y_{1:T}; A)$$

It is possible to derive an iterative algorithm, known as the Baum-Welch algorithm to find the MLE. The Baum-Welch algorithm is a special case of the Expectation-Maximization algorithm, applied to HMMs. It is beyond the scope of this course to give a general description of the EM algorithm (see the module on Advanced Topics in Statistical Machine Learning). The EM algorithm is an iterative algorithm which proceeds in two steps. At iteration k

- E step
$$Q(A; A^{(k-1)}) = \mathbb{E}[\log p(X_{0:T}, y_{1:T}; A) | y_{1:T}, A^{(k-1)}]$$
- M step
$$A^{(k)} = \arg\max_{A} \ Q(A; A^{(k-1)})$$

Each iteration increases the value of the log-likelihood

$$\log p(y_{1:T}; A^{(k)}) \geq \log p(y_{1:T}; A^{(k-1)})$$

and the algorithm thus converges to a local maximum of the log-likelihood.

We now show how to calculate the Q function. The prior pmf can be expressed as

$$p(x_{0:T}; A) = \mu_{x_0} \prod_{i,j \in \mathcal{X}} A_{i,j}^{n_{i,j}}$$

The log joint pmf can be expressed as

$$\log p(x_{0:T}, y_{1:T}; A) = \log \mu_{x_0} + \sum_{i,j \in \mathcal{X}} n_{i,j} \log A_{i,j} + \sum_{t=1}^{T} \log g_{y_t}(x_t)$$

The Q function of the EM is thus expressed as

$$\begin{aligned} Q(A; A^*) &= \mathbb{E}[\log p(X_{0:T}, y_{1:T}; A) | y_{1:T}, A^*] \\ &= \sum_{i,j \in \mathcal{X}} \mathbb{E}[N_{i,j} | y_{1:T}, A^*] \log A_{i,j} + C \end{aligned}$$

where C is a constant independent of A and

$$N_{i,j} = \sum_{t=1}^{T} I(X_t = j, X_{t-1} = i).$$

The expected counts can be expressed as

$$\mathbb{E}[N_{i,j} | y_{1:T}, A^*] = \sum_{t=1}^{T} \mathbb{P}(X_t = j, X_{t-1} = i | y_{1:T}; A^*)$$

The terms $p(x_t, x_{t-1} | y_{1:T}; A^*)$ can be obtained from the forward-backward recursion, as [check!]

$$p(x_t, x_{t-1} | y_{1:T}; A^*) \propto \alpha_{t-1}(x_{t-1}) p(y_t | x_t) p(x_t | x_{t-1}) \beta_t(x_t)$$

The M step gives

$$\begin{aligned} A_{i,j}^{(k)} &= \arg\max_{A_{i,j}} \ \mathbb{E}[N_{i,j} | y_{1:T}, A^{(k-1)}] \log A_{i,j} \\ &= \frac{\mathbb{E}[N_{i,j} | y_{1:T}, A^{(k-1)}]}{\sum_{\ell} \mathbb{E}[N_{i,\ell} | y_{1:T}, A^{(k-1)}]}. \end{aligned}$$

3 Continuous-state Hidden Markov models

In many problems, the hidden parameter of interest is continuous, and we consider continuous-state hidden Markov models, also known as state-space models, or dynamical systems. We focus here on a particular subclass called linear Gaussian state space model.

3.1 Recap: Linear Gaussian system

We recall in this section some basic results on the manipulation of multivariate Gaussian random variables.

Definition 1. *The probability density function of a multivariate Gaussian random variable $X \in \mathbb{R}^{d_x}$ with mean μ and covariance matrix Σ is given as*

$$\mathcal{N}(x; \mu, \Sigma) := \frac{1}{(2\pi)^{d_x/2}\sqrt{|\Sigma|}} \exp\left\{-\frac{1}{2}(x-\mu)^\mathsf{T}\Sigma^{-1}(x-\mu)\right\}.$$

Remark 2 (Notations)**.** *For a Gaussian random variable X, we write $p_X(x)$ its probability density function. Similarly, for jointly Gaussian random variables X and Y, we write $p_{X,Y}(x,y)$ for the joint pdf of X and Y and $p_{X|Y}(x|y)$ for the conditional pdf of X given $Y = y$. Wherever this does not lead to confusion, we will drop subscripts and use the shorter notations $p(x)$, $p(x,y)$ and $p(x|y)$.*

Proposition 3. *Let (X,Y), $X \in \mathbb{R}^{d_x}$ and $Y \in \mathbb{R}^{d_y}$, be a jointly Gaussian vector with mean and covariance matrix*

$$\mu = \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix}, \Sigma = \begin{pmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{pmatrix}.$$

Then the marginals are given by

$$X \sim \mathcal{N}(\mu_x, \Sigma_{xx})$$
$$Y \sim \mathcal{N}(\mu_y, \Sigma_{yy})$$

and the conditionals

$$X|Y = y \sim \mathcal{N}(\mu_{x|y}, \Sigma_{x|y})$$

where

$$\Sigma_{x|y} = \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx}$$
$$\mu_{x|y} = \mu_x + \Sigma_{xy}\Sigma_{yy}^{-1}(y - \mu_y)$$

Corollary 4. *Consider Gaussian random variables $X \in \mathbb{R}^{d_x}$ and $Y \in \mathbb{R}^{d_y}$ with*

$$X \sim \mathcal{N}(\mu_x, \Sigma_{xx})$$
$$Y|X = x \sim \mathcal{N}(Ax + b, \Sigma_{y|x})$$

where $\mu_x \in \mathbb{R}^{d_x}$, Σ_{xx} is a $d_x \times d_x$ covariance matrix, A is a $d_y \times d_x$ matrix, b is a d_y vector and $\Sigma_{y|x}$ is a $d_y \times d_y$ covariance matrix. Then

$$(X|Y = y) \sim \mathcal{N}\left(\mu_{x|y}, \Sigma_{x|y}\right)$$
$$Y \sim \mathcal{N}\left(\mu_y, \Sigma_{yy}\right)$$

where

$$\mu_y = A\mu_x + b$$
$$\Sigma_{yy} = \Sigma_{y|x} + A\Sigma_{xx}A^\mathsf{T}$$
$$\Sigma_{x|y} = \left(\Sigma_{xx}^{-1} + A^\mathsf{T}\Sigma_{y|x}^{-1}A\right)^{-1}$$
$$\mu_{x|y} = \Sigma_{x|y}\left(\Sigma_{xx}^{-1}\mu_x + A^\mathsf{T}\Sigma_{y|x}^{-1}(y-b)\right)$$

3.2 Dynamic Linear Gaussian state-space models

Let $(X_0, \ldots, X_T)$ be a sequence of continuous random variables taking values in $\mathbb{R}^{d_x}$, corresponding to the hidden state of interest, and $(Y_1, \ldots, Y_T)$ be a sequence of continuous random variables taking values in $\mathbb{R}^{d_y}$ (observations). The linear Gaussian state-space model is defined as, for $t = 1, \ldots, T$

$$\begin{aligned} X_t &= F_t X_{t-1} + G_t V_t && \text{State model} \\ Y_t &= H_t X_t + W_t && \text{Observation model} \end{aligned}$$

where the random variables $(X_0, V_1, V_2, \ldots, V_T, W_1, W_2, \ldots, W_T)$ are independent with $X_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$ and for $t = 1, \ldots, T$,

$$\begin{aligned} V_t &\sim \mathcal{N}(0, Q_t) \\ W_t &\sim \mathcal{N}(0, R_t) \end{aligned}$$

with

- X_t is the hidden state at time t
- Y_t is the observation at time t
- V_t is the state noise at time t
- W_t is the observation noise at time t
- F_t is the $d_x \times d_x$ state transition matrix
- G_t is the $d_x \times d_v$ noise transfer matrix
- H_t is the $d_y \times d_x$ observation matrix

Under the above assumptions, the sequence $(X_0, X_1, Y_1, \ldots, X_T, Y_T)$ is a (continuous-state) hidden Markov model, which can be represented graphically as in Figure 2. If $G_t Q_t G_t^\mathsf{T}$ has full rank, the joint pdf $p(x_{0:T}, y_{1:T})$ factorizes as

$$p(x_{0:T}, y_{1:T}) = p(x_0) \prod_{t=1}^{T} p(y_t|x_t) p(x_t|x_{t-1})$$

where

$$\begin{aligned} p(x_t|x_{t-1}) &= \mathcal{N}(x_t; F_t x_{t-1}, G_t Q_t G_t^\mathsf{T}) \\ p(y_t|x_t) &= \mathcal{N}(y_t; H_t x_t, R_t) \end{aligned}$$

Example 5 (Object Tracking)**.** *Let $X_t = (P_t^x, P_t^y, P_t^z, V_t^x, V_t^y, V_t^z)^\mathsf{T}$ denote the position and velocity of an object at time index $t = 0, 1, \ldots$. The position and velocity are not directly observed, but a GPS delivers noisy observations of the position*

$$Y_t = (P_t^x, P_t^y, P_t^z)^\mathsf{T} + W_t$$

where the GPS error W_t is supposed (as a first approximation) to be Gaussian with zero mean and covariance matrix R. As an approximation to the dynamics of the mobile object, we consider the white noise acceleration model

$$\begin{aligned} P_t^x &= P_{t-1}^x + \delta V_{t-1}^x + \frac{\delta^2}{2} A_{t-1}^x \\ V_t^x &= V_{t-1}^x + \delta A_{t-1}^x \end{aligned}$$

where $\delta = 1$ here, A_{t-1}^x is the unknown acceleration at time t, assumed to be Gaussian with zero mean and variance Q^x, and similarly for the other coordinates. We therefore have a dynamic linear Gaussian model with

$$F = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, G = \begin{pmatrix} 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1/2 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, H = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

and $V_t = (A_{t-1}^x, A_{t-1}^y, A_{t-1}^z)^\mathsf{T}$. The objective is to calculate $p(x_t|y_{1:t})$ to obtain the position and velocity of each object at each time t.

Example 6 (Linear regression with time-varying regression coefficients)**.** *Let (z_t, Y_t), $t = 1, \ldots, T$ where $z_t \in \mathbb{R}^p$ are covariates and $Y_t \in \mathbb{R}$ are response variables. We assume that there is a linear relation between the response and the covariate. However the regression coefficients are not assumed to be fixed but evolve over time, according to a random walk. We consider the following model*

$$\begin{aligned}\beta_t &= \beta_{t-1} + V_t \\ Y_t &= z_t \beta_t + W_t\end{aligned}$$

where $\beta_t \in \mathbb{R}^p$ is the regressor at time t, and V_t is a vector of independent Gaussian random variables with variance σ_V^2. This parameter tunes how quickly the parameter β_t evolves over time.

3.3 Inference in dynamic linear Gaussian SSMs

3.3.1 The Kalman filter

Assume that we are interested in the pdf $p(x_t|y_{1:t})$ of the hidden state X_t given observations $y_{1:t}$ up to time t. Let

$$\begin{aligned}\mu_{t|t-1} &:= \mathbb{E}[X_t|Y_{1:t-1} = y_{1:t-1}] \\ \Sigma_{t|t-1} &:= \mathbb{E}[(X_t - \mu_{t|t-1})(X_t - \mu_{t|t-1})^\mathsf{T}|Y_{1:t-1} = y_{1:t-1}] \\ \mu_{t|t} &:= \mathbb{E}[X_t|Y_{1:t} = y_{1:t}] \\ \Sigma_{t|t} &:= \mathbb{E}[(X_t - \mu_{t|t})(X_t - \mu_{t|t})^\mathsf{T}|Y_{1:t} = y_{1:t}]\end{aligned}$$

The Kalman filter computes sequentially the above means and covariance matrices using two steps: the prediction step and the update step.

$$(\mu_0, \Sigma_0) \overset{\text{Prediction}}{\longrightarrow} (\mu_{1|0}, \Sigma_{1|0}) \longrightarrow \ldots \longrightarrow (\mu_{t-1|t-1}, \Sigma_{t-1|t-1}) \overset{\text{Prediction}}{\longrightarrow} (\mu_{t|t-1}, \Sigma_{t|t-1}) \overset{\text{Update}}{\longrightarrow} (\mu_{t|t}, \Sigma_{t|t}) \overset{\text{Prediction}}{\longrightarrow} \ldots$$

Proposition 7. *Let $p(x_t|y_{1:t})$ and $p(x_t|y_{1:t-1})$ be the filtering and one-step predictive pdfs at time t. Then*

$$\begin{aligned}p(x_t|y_{1:t-1}) &= \mathcal{N}(x_t; \mu_{t|t-1}, \Sigma_{t|t-1}) \\ p(x_t|y_{1:t}) &= \mathcal{N}(x_t; \mu_{t|t}, \Sigma_{t|t})\end{aligned}$$

where $(\mu_{t|t-1}, \Sigma_{t|t-1})$ and $(\mu_{t|t}, \Sigma_{t|t})$ follow the recursion

- *Prediction step*

$$\begin{aligned}\mu_{t|t-1} &= F_t \mu_{t-1|t-1} \\ \Sigma_{t|t-1} &= F_t \Sigma_{t-1|t-1} F_t^\mathsf{T} + G_t Q_t G_t^\mathsf{T}\end{aligned}$$

- *Update/correction step*

$$\begin{aligned}\mu_{t|t} &= \mu_{t|t-1} + K_t \nu_t \\ \Sigma_{t|t} &= (I - K_t H_t)\Sigma_{t|t-1}\end{aligned}$$

where ν_t is the residual or innovation, given by the difference between the observation and the predicted observation

$$\begin{aligned}\nu_t &:= y_t - \hat{y}_{t|t} \\ \hat{y}_{t|t} &:= \mathbb{E}[Y_t|Y_{1:t-1} = y_{1:t-1}] = H_t \mu_{t|t-1}\end{aligned}$$

and K_t is the ***Kalman gain***

$$K_t = \Sigma_{t|t-1} H_t^\mathsf{T} S_t^{-1}$$

with

$$\begin{aligned}S_t &:= \mathbb{E}[(Y_t - \hat{y}_{t|t})(Y_t - \hat{y}_{t|t})^\mathsf{T}|Y_{1:t-1} = y_{1:t-1}] \\ &= H_t \Sigma_{t|t-1} H_t^\mathsf{T} + R_t.\end{aligned}$$

Proof. (not examinable) We have $X_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$. Assume that $X_{t-1}|Y_{1:t-1} \sim \mathcal{N}(\mu_{t-1|t-1}, \Sigma_{t-1|t-1})$. Consider the state model,

$$X_t = F_t X_{t-1} + G_t V_t.$$

As V_t is Gaussian, $X_t|Y_{1:t-1}$ is a linear combination of Gaussian random variables, and is therefore Gaussian. Y_t can be written as a function of the random variables $(X_0, V_0, \ldots, V_{t-1}, W_0, \ldots, W_{t-1})$. V_t is independent of $(X_0, V_0, \ldots, V_{t-1}, W_1, \ldots, W_{t-1})$, hence it is independent of $Y_{1:t}$. We therefore have

$$\begin{aligned}\mathbb{E}[X_t|y_{1:t-1}] &= F_t\mathbb{E}[X_{t-1}|y_{1:t-1}] + G_t\mathbb{E}[V_t|y_{1:t-1}] = F_t\mu_{t-1|t-1}\\ cov[X_t|y_{1:t-1}] &= cov[F_tX_{t-1}|y_{1:t-1}] + cov[G_tV_t|y_{1:t-1}]\\ &= F_tcov[X_{t-1}|y_{1:t-1}]F_t^\mathsf{T} + G_tcov[V_t]G_t^\mathsf{T} = F_t\Sigma_{t-1|t-1}F_t^\mathsf{T} + G_tQ_tG_t^\mathsf{T}\end{aligned}$$

Assume now that

$$X_t|Y_{1:t-1} \sim \mathcal{N}(\mu_{t|t-1}, \Sigma_{t|t-1})$$

Conditional on X_t, Y_t is independent of $Y_{1:t-1}$, as the observation noise W_t is independent of $Y_{1:t-1}$. Hence, using the observation model

$$Y_t|X_t = x_t, Y_{1:t-1} \sim \mathcal{N}(H_tx_t, R_t)$$

and Corollary 4, we obtain

$$\begin{aligned}\Sigma_{t|t} &= \left(\Sigma_{t|t-1}^{-1} + H_t^\mathsf{T}R_t^{-1}H_t\right)^{-1}\\ \mu_{t|t} &= \Sigma_{t|t}\left(\Sigma_{t|t-1}^{-1}\mu_{t|t-1} + H_t^\mathsf{T}R_t^{-1}y_t\right)\end{aligned}$$

We rearrange this using the Woodbury matrix identity

$$(A+UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}.$$

This gives

$$\begin{aligned}\Sigma_{t|t} &= \Sigma_{t|t-1} - \Sigma_{t|t-1}H_t^\mathsf{T}(R_t + H_t\Sigma_{t|t-1}H_t^\mathsf{T})^{-1}H_t\Sigma_{t|t-1}\\ &= (I - K_tH_t)\Sigma_{t|t-1}\end{aligned}$$

and

$$\begin{aligned}\mu_{t|t} &= (I-K_tH_t)\Sigma_{t|t-1}\left(\Sigma_{t|t-1}^{-1}\mu_{t|t-1} + H_t^\mathsf{T}R_t^{-1}y_t\right)\\ &= (I-K_tH_t)\mu_{t|t-1} + (I-K_tH_t)\Sigma_{t|t-1}H_t^\mathsf{T}R_t^{-1}y_t\\ &= (I-K_tH_t)\mu_{t|t-1} + (I-K_tH_t)K_tS_tR_t^{-1}y_t\\ &= (I-K_tH_t)\mu_{t|t-1} + K_t(I-H_tK_t)S_tR_t^{-1}y_t\\ &= (I-K_tH_t)\mu_{t|t-1} + K_ty_t\end{aligned}$$

as $S_t - H_tK_tS_t = R_t$. □

3.3.2 The Kalman smoother

We are now interested in the pdfs $p(x_t|y_{1:T})$ of the hidden state X_t given all the observations $y_{1:T}$. Let

$$\begin{aligned}\mu_{t|T} &:= \mathbb{E}[X_t|Y_{1:T} = y_{1:T}]\\ \Sigma_{t|T} &:= \mathbb{E}[(X_t - \mu_{t|T})(X_t - \mu_{t|T})^\mathsf{T}|Y_{1:T} = y_{1:T}]\end{aligned}$$

We can obtain the smoothing pdfs by first running the forward recursion of the Kalman filter, in order to obtain $(\mu_{t|t}, \Sigma_{t|t})$ for $t = 1, \ldots, T$, and then run a backward recursion.

Proposition 8. *Let $p(x_t|y_{1:T})$ be the smoothing pdf at time t. Then*

$$p(x_t|y_{1:T}) = \mathcal{N}(x_t; \mu_{t|T}, \Sigma_{t|T})$$

where $(\mu_{t|T}, \Sigma_{t|T})$ follow the recursion

$$\begin{aligned}\mu_{t|T} &= \mu_{t|t} + J_t(\mu_{t+1|T} - \mu_{t+1|t})\\ \Sigma_{t|T} &= \Sigma_{t|t} + J_t(\Sigma_{t+1|T} - \Sigma_{t+1|t})J_t^\mathsf{T}\end{aligned}$$

where J_t is the ***backward Kalman gain***

$$J_t = \Sigma_{t|t}F_{t+1}^\mathsf{T}\Sigma_{t+1|t}^{-1}.$$

3.4 Example

Consider the following simple scalar example of a random walk observed in noise

$$X_t = X_{t-1} + V_t$$
$$Y_t = X_t + W_t$$

where $X_0 \sim \mathcal{N}(0,1)$, $V_t \sim \mathcal{N}(0,Q)$, $W_t \sim \mathcal{N}(0,R)$ where $Q = 0.02$ and $R = 0.2$. In this case, we have

- Prediction

$$\mu_{t|t-1} = \mu_{t-1|t-1}$$
$$\Sigma_{t|t-1} = \Sigma_{t-1|t-1} + Q$$

- Update

$$K_t = \frac{\Sigma_{t|t-1}}{\Sigma_{t|t-1} + R}$$

and

$$\Sigma_{t|t} = (1 - K_t)\Sigma_{t|t-1}$$
$$\mu_{t|t} = (1 - K_t)\mu_{t|t-1} + K_t y_t$$

Assume that we observe $y_1 = 1.6$. The first iteration of the Kalman filter gives

$$\mu_{1|0} = 0$$
$$\Sigma_{1|0} = 1 + 0.02 = 1.02$$
$$K_1 = \frac{1.02}{1.02 + 0.2} = 0.83$$
$$\Sigma_{1|1} = 0.17$$
$$\mu_{1|1} = 1.34$$

Results are presented in Figures 8 and 9 for data simulated from the model with $T = 50$. The R code is in Appendix 3.4.

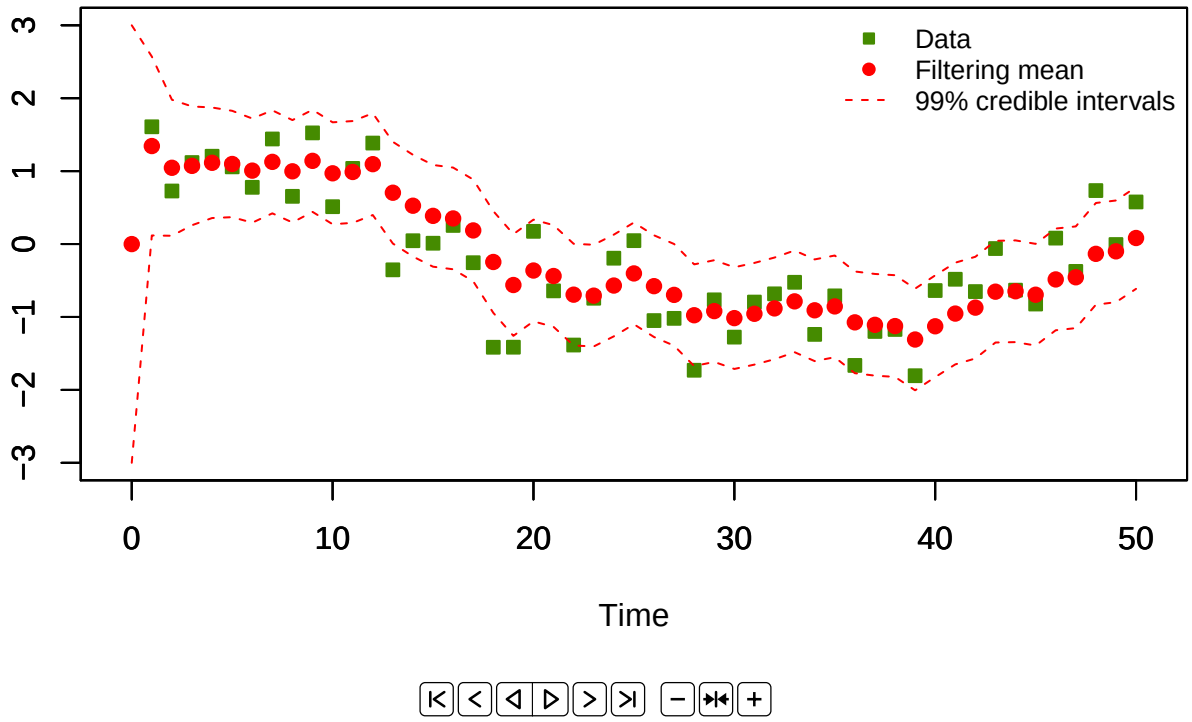

Figure 8: Filtering mean (red dot) and 99% credible intervals (red dotted line) over time. Observations are green squares.

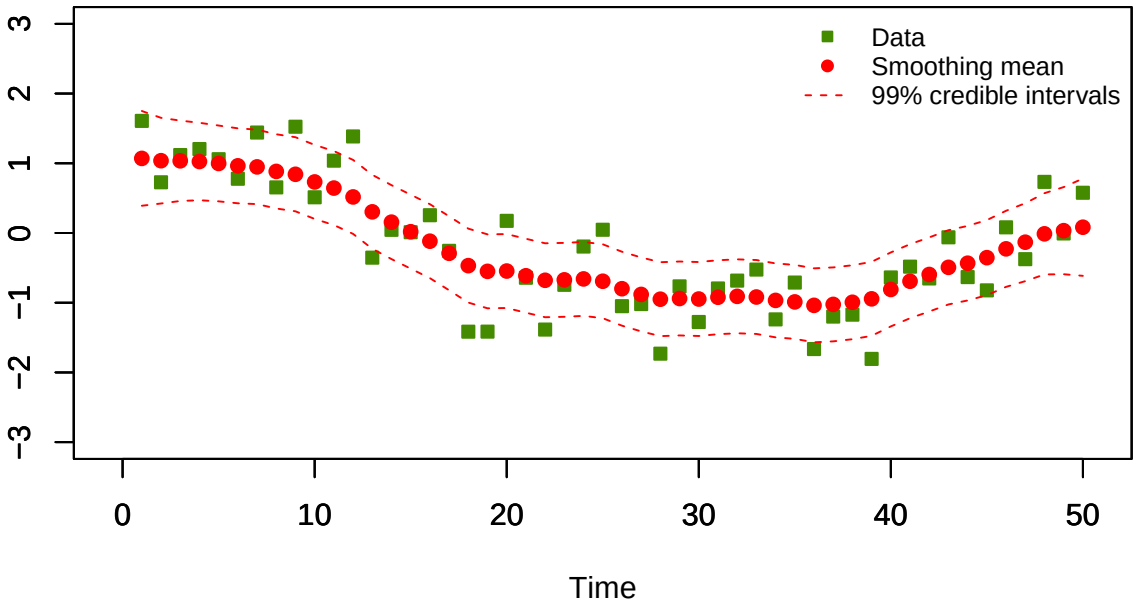

Figure 9: Smoothing mean (red dot) and 99% credible intervals (red dotted line) over time. Observations are green squares.

R code for the Kalman filter and smoother

```
kalman = function(y, F, G, Q, H, R, mu0, Sigma0) {
    dy = nrow(y)
    T = ncol(y)
    dx = length(mu0)
    I = diag(dx)

    ## INITIALIZATION ##
    mu.p = matrix(0, nrow = dx, ncol = T)
    Sigma.p = array(0, c(dx, dx, T))
    mu.f = matrix(0, nrow = dx, ncol = T)
    Sigma.f = array(0, c(dx, dx, T))
    mu.s = matrix(0, nrow = dx, ncol = T)
    Sigma.s = array(0, c(dx, dx, T))

    ## FORWARD RECURSION ## Time 1
    mu.p[, 1] = F %*% mu0
    Sigma.p[, , 1] = F %*% Sigma0 %*% t(F) + G %*% Q %*% t(G)

    nu = y[, 1] - H %*% mu.p[, 1]
    S = H %*% Sigma.p[, , 1] %*% t(H) + R
    K = Sigma.p[, , 1] %*% t(H) %*% solve(S)
    mu.f[, 1] = mu.p[, 1] + K %*% nu
    Sigma.f[, , 1] = (I - K %*% H) %*% Sigma.p[, , 1]

    # Time 2:T
    for (t in (2:T)) {
        # Prediction
        mu.p[, t] = F %*% mu.f[, t - 1]
        Sigma.p[, , t] = F %*% Sigma.f[, , t - 1] %*% t(F) + G %*% Q %*% t(G)

        # Update
        nu = y[, t] - H %*% mu.p[, t]
        S = H %*% Sigma.p[, , t] %*% t(H) + R
        K = Sigma.p[, , t] %*% t(H) %*% solve(S)
        mu.f[, t] = mu.p[, t] + K %*% nu
        Sigma.f[, , t] = (I - K %*% H) %*% Sigma.p[, , t]
    }

    ## BACKWARD RECURSION ##
    mu.s[, T] = mu.f[, T]
    Sigma.s[, , T] = Sigma.f[, , T]
    for (t in (T - 1):1) {
        J = Sigma.f[, , t] %*% t(F) %*% solve(Sigma.p[, , t + 1])
        mu.s[, t] = mu.f[, t] + J %*% (mu.s[, t + 1] - mu.p[, t + 1])
        Sigma.s[, , t] = Sigma.f[, , t] + J %*% (Sigma.s[, , t + 1] - Sigma.p[,
            , t + 1]) %*% t(J)
    }

    return(list(mu.f = mu.f, Sigma.f = Sigma.f, mu.p = mu.p, Sigma.p = Sigma.p,
        mu.s = mu.s, Sigma.s = Sigma.s))
}
```

```
T = 50
x = matrix(cos(c(1:T)/10), 1, T)
R = 0.2
mu0 = 0
```

```
Sigma0 = 1
G = 1
Q = 0.02
H = 1
F = 1
y = matrix(x + rnorm(T, sd = sqrt(R)), nrow = 1, ncol = T)

results.KF = kalman(y, F, G, Q, H, R, mu0, Sigma0)
mu.f = results.KF$mu.f
Sigma.f = results.KF$Sigma.f
```