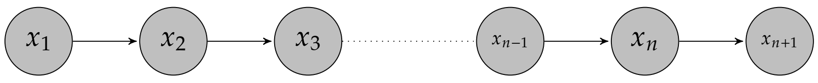
Hidden Markov Models 5

5.1 Introduction

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

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

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



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

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

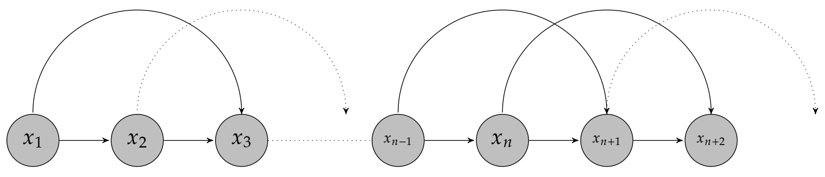
It holds that future observations are independent of all but the most recent observation:

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

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

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

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



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

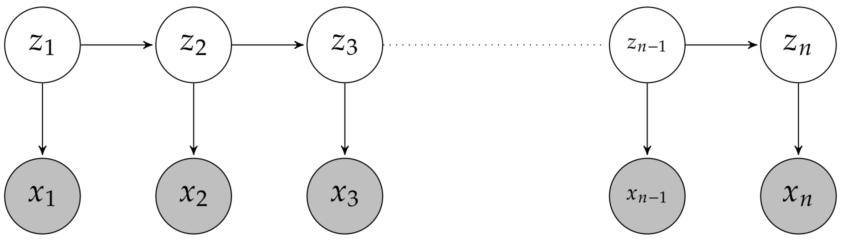
$$p(x_1, \dots, x_N) = p(x_1)p(x_2|x_1)p(x_3|x_2, x_1)\dots p(x_{n+1}|x_n, x_{n-1})\dots$$

And it holds that:

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

Remark: if x_i are discrete, we talk about *Markov chains*; if x_i are continuous and $p(x_n|\dots)$ are Gaussian, we talk about *autoregressive models* (of order k).

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



If we consider two observations x_i, x_j, then $x_i \not\perp\!\!\!\perp x_j|\underline{x}$ (with $\underline{x} = x_1, \dots, x_n$) since there isn't any observed node in the path from x_i to x_j.

In order to describe state space models like the one above we need:

- **transition probabilities** $p(z_n|z_{n-1})$, which describe the evolution of the latent variables. They can be arranged in a matrix A s.t. $A_{ij} = p(z_n = j|z_{n-1} = i)$;
- **initial distribution** $p(z_1)$, specified by a vector π s.t. $\pi_i = p(z_1 = i)$;
- **emission probability** $p(x_n|z_n)$, parametrized by ψ. The emission probability for discrete x is a categorical distribution, for continuous x is typically a Gaussian or a mixture of Gaussians.

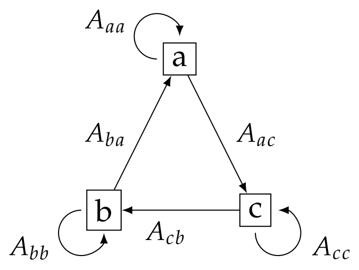
Definition 5.1.1 *The* ***Hidden Markov Model*** *is a specific instance of the state space model described before, in which the latent variables z_i are discrete. If instead the variables z_i are continuous and the transition probabilities $p(z_i|z_{i-1})$ are Gaussian, we talk about* ***Linear Dynamical System****.*

Consider the parameters $\theta = (A, \pi, \psi)$ and the variables

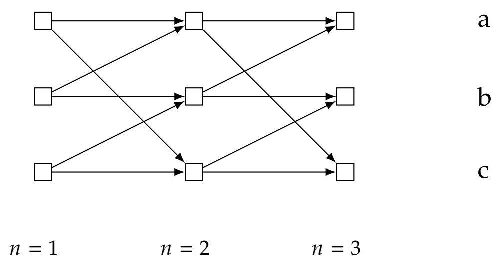
$\underline{x}, \underline{z} = (x_1, \dots, x_n, z_1, \dots, z_n)$, then:

$$p(\underline{x}, \underline{z}|\theta) = p(z_1|\pi)\left[\prod_{n=2}^{N} p(z_n|z_{n-1}, A)\right]\left[\prod_{n=1}^{N} p(x_n|z_n, \psi)\right]$$

Example. We can graphically represent the states of $\underline{z}$ and the transition matrix as follows (note that this is not a PGM):



If we unfold the above graph over time, we obtain a sort of trellis diagram of the latent states:

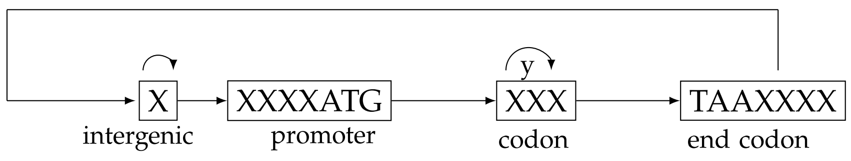


5.1.1 Application: HMM for gene finding

In the field of Computational Biology, a typical application of HMM is that of *Gene Finding*. This task consists in identifying the regions of genomic DNA that encodes genes (DNA belongs indeed to the class of sequential data).

Our goal is, given an observable sequence of bases, find the most likely sequence of internal states that generated it, where the internal state essentially describes in which part of the DNA we are.

In prokaryotic cells, the latent variable state space is roughly the following:



with $x \in \{A, C, G, T\}$, $y \in \{[A, C, G, T]\}^3$.

Without going into full detail, the function of the internal states is:

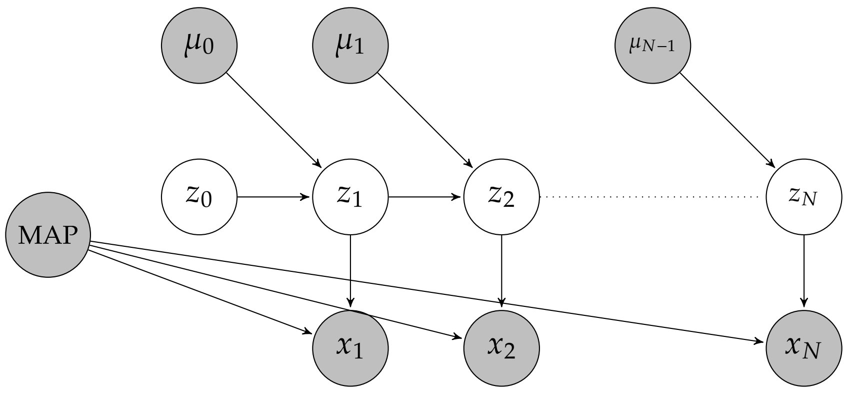
- Intergenic region: stretch of DNA sequences located between genes. It is a subset of noncoding DNA.
- Promoter: region of DNA where the transcription of gene is initiated.

- Codon: trinucleotide sequence of DNA or RNA that corresponds to a specific amino acid. It describes the relationship between the sequence of DNA bases (A, C, G and T) in a gene and the corresponding protein sequence that it encodes.
- End codon: nucleotide triplet that signals the termination of the translation process of the current protein.

5.1.2 Application: HMM for robot localization

Robot Localization is the process of determining where a mobile robot is located w.r.t. its environment. In a typical robot localization scenario, a map of the environment is available and the robot is equipped with sensors that observe the environment as well as monitor its own motion.

Graphically the situation is the following:



where μ_i are the controls we give to the robot and x_i are the readings from the sensors.

Note that this is a tree-like graphical model, so we can convert it into a factor graph and apply sum-product and max-plus algorithms to do inference.

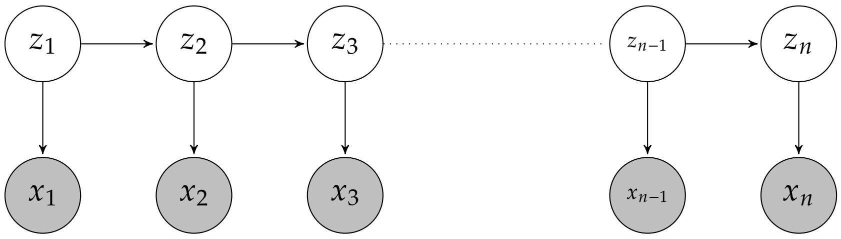
5.2 Inference in HMM

There are several class of inference problems that we may want to perform on HMMs:

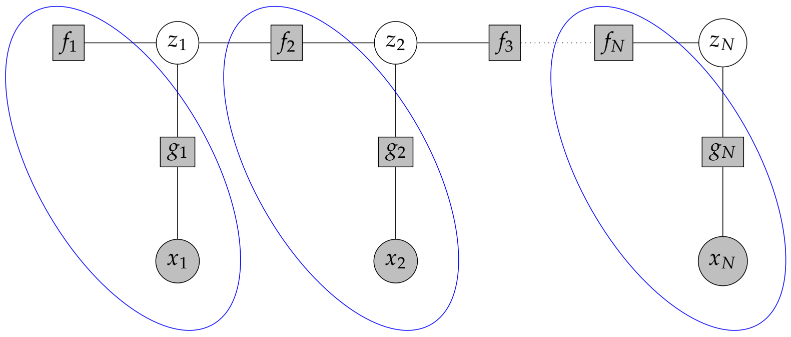
- **Filtering**: given the observations $\underline{x}$, we want to compute the distribution over hidden states of the last latent variable at the end of the sequence, i.e. compute the probability $p(z_n|\underline{x})$.
- **Smoothing**: we want to compute the distribution of a latent variable somewhere in the middle of the sequence, at a certain time k, i.e. compute the probability of $p(z_k|\underline{x})$ with $k < N$.
- **Most likely explanation**: most likely sequence of latent variables that generated a particular sequence of observations, i.e. compute $z^\star = \text{argmax}_{\underline{z}}\, p(\underline{z}|\underline{x})$ with $\underline{z} = z_1, \dots, z_n$.

Remark: we also need to take into account the *parameter learning* task, i.e. given an output sequence, find the transition and emission probabilities (usually solved via maximum likelihood).

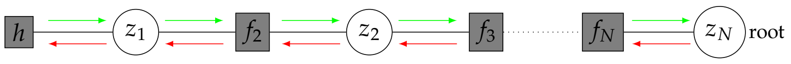
Our goal is now to recast the algorithms for exact inference in PGM (sum-product and max-plus) to the context of HMM. Since both work in Factor Graphs, we need to convert the following Bayesian Network (i.e. the representation we have used so far for HMM) into a Factor Graph:



In this case factor nodes essentially correspond to the edges of the above figure, with the addition of the factor node for the initial probability. Hence we get the following:



We can actually simplify the representation by collapsing into single factor nodes the three nodes inside each ellipsis in the diagram above (the rationale behind this is that observations, by definitions, are fixed). Hence we obtain a chain:



By construction it holds that:

$$h(z_1) = p(z_1)p(x_1|z_1) \text{ with } x_1 \text{ observed, hence clamped}$$

$$f_2(z_1, z_2) = p(z_2|z_1)p(x_2|z_2) \text{ with } x_2 \text{ observed, hence clamped}$$

until

$$f_N(z_{N-1}, z_N) = p(z_N|z_{N-1})p(x_N|z_N) \text{ with } x_N \text{ observed, hence clamped}$$

Fixing z_N as the root, the sum-product algorithm reads as:

- Forward messages (green arrows in the figure above):

$$\mu_{z_{n-1}\to f_n}(z_{n-1}) = \mu_{f_{n-1}\to z_{n-1}}(z_{n-1})$$
$$\mu_{f_n\to z_n}(z_n) = \sum_{z_{n-1}} f_n(z_{n-1}, z_n)\cdot \mu_{z_{n-1}\to f_n}(z_{n-1})$$

  We can eliminate $\mu_{z_{n-1}\to f_n}(z_{n-1})$ and obtain a recursion for the forward messages:

$$\alpha(z_n) := \mu_{f_n\to z_n}(z_n)$$
$$\alpha(z_n) = \sum_{z_{n-1}} f_n(z_{n-1}, z_n)\cdot \alpha(z_{n-1})$$

- Backward messages (red arrows in the figure above):

$$\mu_{f_{n+1}\to z_n}(z_n) = \sum_{z_{n+1}} f_{n+1}(z_n, z_{n+1})\cdot \mu_{f_{n+2}\to z_{n+1}}(z_{n+1})$$

  rewriting with the usual notation:

$$\beta(z_n) := \mu_{f_{n+1}\to z_n}(z_n)$$
$$\beta(z_n) = \sum_{z_{n+1}} f_{n+1}(z_n, z_{n+1})\cdot \beta(z_{n+1})$$

After computing all the messages, smoothing can be solved combining forward and backward messages:

$$p(z_n, \underline{x}) = \alpha(z_n)\beta(z_n)$$

while filtering:

$$p(z_N, \underline{x}) = \alpha(z_N)$$

Moreover:

$$p(z_n, z_{n+1}, \underline{x}) = \alpha(z_n) f(z_n, z_{n+1})\beta(z_{n+1})$$
$$p(z_n|\underline{x}) = \frac{p(z_n, \underline{x})}{\sum_{z_n} p(z_n, \underline{x})} = \frac{p(z_n, \underline{x})}{p(\underline{x})} = \frac{p(z_n, \underline{x})}{\sum_{z_n} \alpha(z_n)}$$

In order to find the most likely sequence, we need to plug the max-plus algorithm. This reads as:

$$\hat{\mu}_{f_n\to z_n} = \max_{z_{n-1}}\{\log f_n(z_n, z_{n-1}) + \hat{\mu}_{f_{n-1}\to z_{n-1}}(z_{n-1})\}$$
$$\Phi(z_n) = \underset{z_{n-1}}{\operatorname{argmax}}\{\log f_n(z_n, z_{n-1}) + \hat{\mu}_{f_{n-1}\to z_{n-1}}(z_{n-1})\}$$

Remark: In the context of HMM, the max-plus algorithm is known as **Viterbi algorithm**.