Sum Product Algorithm and Hidden Markov Model **2014/2015**

Cours 7 — 12th November 2014

Enseignant: Francis Bach *Scribe: Pauline Luc, Mathieu Andreux*

7.1 Sum Product Algorithm

7.1.1 Motivations

Inference, along with *estimation* and *decoding*, are the three key operations one must be able to perform efficiently in graphical models.

Given a discrete Gibbs model of the form: $p(x) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(x_C)$, where $\mathcal{C}$ is the set of cliques of the graph, inference enables:

- Computation of the marginal $p(x_i)$ or more generally, $p(x_C)$.
- Computation of the partition function Z
- Computation of the conditional marginal $p(x_i | X_j = x_j, X_k = x_k)$

And as a consequence

- Computation of the gradient in a exponential family
- Computation of the expected value of the loglikelihood of an exponential family at step E of the EM algorithm (for example for HMM)

Example 1: Ising model

Let $X = (X_i)_{i \in V}$ be a vector of random variables, taking value in $\{0,1\}^{|V|}$, of which the exponential form of the distribution is:

$$p(x) = e^{-A(\eta)} \prod_{i \in V} e^{\eta_i x_i} \prod_{(i,j) \in E} e^{\eta_{i,j} x_i x_j} \tag{7.1}$$

We then have the log-likelihood:

$$l(\eta) = \sum_{i \in V} \eta_i x_i + \sum_{(i,j) \in E} \eta_{i,j} x_i x_j - A(\eta) \tag{7.2}$$

We can therefore write the sufficient statistic:

$$\phi(x) = \begin{pmatrix} (x_i)_{i \in V} \\ (x_i x_j)_{(i,j) \in E} \end{pmatrix} \tag{7.3}$$

But we have seen that for exponential families:

$$l(\eta) = \phi(x)^T\eta - A(\eta) \tag{7.4}$$
$$\nabla_\eta \; l(\eta) = \phi(x) - \underbrace{\nabla_\eta \; A(\eta)}_{\mathbb{E}_\eta[\phi(X)]} \tag{7.5}$$

We therefore need to compute $\mathbb{E}_\eta[\phi(X)]$. In the case of the Ising model, we get:

$$\mathbb{E}_\eta[X_i] = \mathbb{P}_\eta[X_i = 1] \tag{7.6}$$
$$\mathbb{E}_\eta[X_iX_j] = \mathbb{P}_\eta[X_i = 1, X_j = 1] \tag{7.7}$$

The equation 7.6 is one of the motivations for solving the problem of inference. In order to be able to compute the gradient of the log-likelihood, we need to know the marginal laws.

Example 2: Potts model

X_i are random variables, taking value in $\{1, \dots, K_i\}$. We note Δ_{ik} the random variable such that $\Delta_{ik} = 1$ if and only if $X_i = k$. Then,

$$p(\delta) = \exp\left[\sum_{i\in V}\sum_{k=1}^{K_i}\eta_{i,k}\delta_{ik} + \sum_{(i,j)\in E}\sum_{k=1}^{K_i}\sum_{k'=1}^{K_j}\eta_{i,j,k,k'}\delta_{ik}\delta_{jk'} - A(\eta)\right] \tag{7.8}$$

and

$$\phi(\delta) = \begin{pmatrix} (\delta_{ik})_{i,k} \\ (\delta_{ik}\delta_{jk'})_{i,j,k,k'} \end{pmatrix} \tag{7.9}$$
$$\mathbb{E}_\eta[\Delta_{ik}] = \mathbb{P}_\eta[X_i = k] \tag{7.10}$$
$$\mathbb{E}_\eta[\Delta_{ik}\Delta_{jk'}] = \mathbb{P}_\eta[X_i = k, X_j = k'] \tag{7.11}$$

These examples illustrate the need to perform inference.

☡ Problem: In general, the inference problem is NP-hard.

For trees the inference problem is efficient as it is linear in n.

For "tree-like" graphs we use the *Junction Tree Algorithm* which enables us to bring the situation back to that of a tree.

In the general case we are forced to carry out approximative inference.

7.1.2 Inference on a chain

We define X_i a random variable, taking value in $\{1,\dots,K\}$, $i \in V = \{1,\dots,n\}$ with joint distribution $p(x)$ defined as:.

$$p(x) = \frac{1}{Z}\prod_{i=1}^{n}\psi_i(x_i)\prod_{i=2}^{n}\psi_{i-1,i}(x_{i-1},x_i) \tag{7.12}$$

We wish to compute $p(x_j)$ for a certain j. The naive solution would be to compute the marginal

$$p(x_j) = \sum_{x_{V\setminus\{j\}}} p(x_1,\dots,x_n) \tag{7.13}$$

Unfortunately, this type of calculation is of complexity $O(K^n)$. We therefore develop the expression

$$\begin{aligned}
p(x_j) &= \frac{1}{Z}\sum_{x_{V\setminus\{j\}}}\prod_{i=1}^{n}\psi_i(x_i)\prod_{i=2}^{n}\psi_{i-1,i}(x_{i-1},x_i) && (7.14)\\
&= \frac{1}{Z}\sum_{x_{V\setminus\{j\}}}\prod_{i=1}^{n-1}\psi_i(x_i)\prod_{i=2}^{n-1}\psi_{i-1,i}(x_{i-1},x_i)\psi_n(x_n)\psi_{n-1,n}(x_{n-1},x_n) && (7.15)\\
&= \frac{1}{Z}\sum_{x_{V\setminus\{j,n\}}}\sum_{x_n}\prod_{i=1}^{n-1}\psi_i(x_i)\prod_{i=2}^{n-1}\psi_{i-1,i}(x_{i-1},x_i)\psi_n(x_n)\psi_{n-1,n}(x_{n-1},x_n) && (7.16)
\end{aligned}$$

Which allows us to bring out the messaged passed by (n) to $(n-1)$: $\mu_{n\to n-1}(x_{n-1})$. When continuing, we obtain:

$$\begin{aligned}
p(x_j) &= \frac{1}{Z}\sum_{x_{V\setminus\{j,n\}}}\prod_{i=1}^{n-1}\psi_i(x_i)\prod_{i=2}^{n-1}\psi_{i-1,i}(x_{i-1},x_i)\underbrace{\sum_{x_n}\psi_n(x_n)\psi_{n-1,n}(x_{n-1},x_n)}_{\mu_{n\to n-1}(x_{n-1})} && (7.17)\\
&= \frac{1}{Z}\sum_{x_{V\setminus\{j,n,n-1\}}}\prod_{i=1}^{n-2}\psi_i(x_i)\prod_{i=2}^{n-2}\psi_{i-1,i}(x_{i-1},x_i)\times\\
&\quad\times\underbrace{\sum_{x_{n-1}}\psi_{n-1}(x_{n-1})\psi_{n-2,n-1}(x_{n-2},x_{n-1})\mu_{n\to n-1}(x_{n-1})}_{\mu_{n-1\to n-2}(x_{n-2})} && (7.18)\\
&= \frac{1}{Z}\sum_{x_{V\setminus\{1,j,n,n-1\}}}\mu_{1\to 2}(x_2)\ \dots\ \mu_{n-1\to n-2}(x_{n-2}) && (7.19)
\end{aligned}$$

In the above equation, we have implicitely used the following definitions for descending and ascending messages:

$$\mu_{j\to j-1}(x_{j-1}) = \sum_{x_j} \psi_j(x_j)\psi_{j-1,j}(x_{j-1}, x_j)\mu_{j+1\to j}(x_j) \tag{7.20}$$

$$\mu_{j\to j+1}(x_{j+1}) = \sum_{x_j} \psi_j(x_j)\psi_{j,j+1}(x_j, x_{j+1})\mu_{j-1\to j}(x_j) \tag{7.21}$$

Each of these messages is computed with complexity $O((n-1)K^2)$. And finally, we get

$$\boxed{p(x_j) = \frac{1}{Z}\ \mu_{j-1\to j}(x_j)\ \psi_j(x_j)\ \mu_{j+1\to j}(x_j)} \tag{7.22}$$

With only $2(n-1)$ messages, we have calculated $p(x_j)\ \forall j \in V$. Z is otained by summing

$$Z = \sum_{x_i}\ \mu_{i-1\to i}(x_i)\ \psi_i(x_i)\ \mu_{i+1\to i}(x_i) \tag{7.23}$$

7.1.3 Inference in undirected trees

We note i the vertice of which we want to compute the marginal law $p(x_i)$. We set i to be the root of our tree. $\forall\ j \in V$, we note $\mathcal{C}(j)$ the set of childen of j and $\mathcal{D}(j)$ the set of descendants of j. The joint probability is:

$$p(x) = \frac{1}{Z}\prod_{i\in V}\psi_i(x_i)\prod_{(i,j)\in E}\psi_{i,j}(x_i, x_j) \tag{7.24}$$

For a tree with at least two vertices, we define by reccurence,

$$F(x_i, x_j, x_{\mathcal{D}(j)}) \triangleq \psi_{i,j}(x_i, x_j)\psi_j(x_j)\prod_{k\in\mathcal{C}(j)} F(x_j, x_k, x_{\mathcal{D}(k)}) \tag{7.25}$$

Then by reformulating the marginal:

$$p(x_i) = \frac{1}{Z} \sum_{x_{V\setminus\{i\}}} \psi_i(x_i) \prod_{j\in\mathcal{C}(i)} F(x_i, x_j, x_{\mathcal{D}(j)}) \tag{7.26}$$

$$= \frac{1}{Z}\, \psi_i(x_i) \prod_{j\in\mathcal{C}(i)} \sum_{x_j, x_{\mathcal{D}(j)}} F(x_i, x_j, x_{\mathcal{D}(j)}) \tag{7.27}$$

$$= \frac{1}{Z}\, \psi_i(x_i) \prod_{j\in\mathcal{C}(i)} \sum_{x_j, x_{\mathcal{D}(j)}} \psi_{i,j}(x_i, x_j)\psi_j(x_j) \prod_{k\in\mathcal{C}(j)} F(x_j, x_k, x_{\mathcal{D}(k)}) \tag{7.28}$$

$$= \frac{1}{Z}\, \psi_i(x_i) \prod_{j\in\mathcal{C}(i)} \sum_{x_j} \psi_{i,j}(x_i, x_j)\psi_j(x_j) \prod_{k\in\mathcal{C}(j)} \underbrace{\sum_{x_k, x_{\mathcal{D}(k)}} F(x_j, x_k, x_{\mathcal{D}(k)})}_{\mu_{k\to j}(x_j)} \tag{7.29}$$

$$= \frac{1}{Z}\, \psi_i(x_i) \prod_{j\in\mathcal{C}(i)} \underbrace{\sum_{x_j} \psi_{i,j}(x_i, x_j)\psi_j(x_j) \prod_{k\in\mathcal{C}(j)} \mu_{k\to j}(x_j)}_{\mu_{j\to i}(x_i)} \tag{7.30}$$

Which leads us to the recurrence relation for the Sum Product Algorithm (SPA):

$$\boxed{\mu_{j\to i}(x_i) = \sum_{x_j} \psi_{i,j}(x_i, x_j)\psi_j(x_j) \prod_{k\in\mathcal{C}(j)} \mu_{k\to j}(x_j)} \tag{7.31}$$

7.1.4 Sum Product Algorithm (SPA)

Sequential SPA for a rooted tree

For a rooted tree, of root (i), the Sum Product Algorithm is written as follows:

1. All the leaves send $\mu_{n\to\pi_n}(x_{\pi_n})$

$$\mu_{n\to\pi_n}(x_{\pi_n}) = \sum_{x_n} \psi_n(x_n)\psi_{n,\pi_n}(x_n, x_{\pi_n}) \tag{7.32}$$

2. Iteratively, at each step, all the nodes (k) which have received messages from all their children send $\mu_{k\to\pi_k}(x_{\pi_k})$ to their parents.

3. At the root we have

$$p(x_i) = \frac{1}{Z}\, \psi_i(x_i) \prod_{j\in\mathcal{C}(i)} \mu_{j\to i}(x_i) \tag{7.33}$$

This algorithm only enables us to compute $p(x_i)$ at the root. To be able to compute all the marginals (as well as the conditional marginals), one must not only *collect* all the messages from the leafs to the root, but then also *distribute* them back to the leafs. In fact, the algorithm can then be written independently from the choice of a root.

SPA for an undirected tree

The case of undirected trees is slightly different:

1. All the leaves send $\mu_{n \to \pi_n}(x_{\pi_n})$

2. At each step, if a node (j) hasn't send a message to one of his neighbours, say (i) (Note: (i) here is not the root) and if it has received messages from all his other neighbours $\mathcal{N}(j)\backslash i$, it send to (i) the following message

$$\mu_{j \to i}(x_i) = \sum_{x_j} \psi_j(x_j)\psi_{j,i}(x_i, x_i) \prod_{k \in \mathcal{N}(j)\backslash\{i\}} \mu_{k \to j}(x_j) \tag{7.34}$$

Parallel SPA (flooding)

1. Initialise the messages randomly

2. At each step, each node sends a new message to each of its neighbours, using the messages received at the previous step.

Marginal laws

Once all messages have been passed, we can easily calculate all the marginal laws

$$\forall\ i \in V,\ p(x_i) = \frac{1}{Z}\ \psi_i(x_i) \prod_{k \in \mathcal{N}(i)} \mu_{k \to i}(x_i) \tag{7.35}$$

$$\forall\ (i,j) \in E,\ p(x_i, x_j) = \frac{1}{Z}\ \psi_i(x_i)\ \psi_j(x_j)\ \psi_{j,i}(x_i, x_i) \prod_{k \in \mathcal{N}(i)\backslash j} \mu_{k \to i}(x_i) \prod_{k \in \mathcal{N}(j)\backslash i} \mu_{k \to j}(x_j) \tag{7.36}$$

Conditional probabilities

We can use a clever notation to calculate the conditional probabilities. Suppose that we want to compute

$$p(x_i | x_5 = 3, x_{10} = 2) \propto p(x_i, x_5 = 3, x_{10} = 2)$$

We can set

$$\tilde{\psi_5}(x_5) = \psi_5(x_5)\ \delta(x_5, 3)$$

Generally speaking, if we observe $X_j = x_{j0}$ for $j \in J_{\text{obs}}$, we can define the modified potentials:

$$\tilde{\psi}_j(x_j) = \psi_j(x_j)\ \delta(x_j, x_{j0})$$

such that

$$p(x|X_{J_{\text{obs}}} = x_{J_{\text{obs}}0}) = \frac{1}{\tilde{Z}} \prod_{i \in V} \tilde{\psi}_i(x_i) \prod_{(i,j)\in E} \psi_{i,j}(x_i, x_j) \tag{7.37}$$

Indeed we have

$$p(x|X_{J_{\text{obs}}} = x_{J_{\text{obs}}0})\ p(X_{J_{\text{obs}}} = x_{J_{\text{obs}}0}) = p(x) \prod_{j \in J_{\text{obs}}} \delta(x_j, x_{j0}) \tag{7.38}$$

so that by dividing the equality by $p(X_{J_{\text{obs}}} = x_{J_{\text{obs}}0})$ we obtain the previous equation with $\tilde{Z} = Zp(X_{J_{\text{obs}}} = x_{J_{\text{obs}}0})$.

We then simply apply the SPA to these new potentials to compute the marginal laws $p(x_i|X_{J_{\text{obs}}} = x_{J_{\text{obs}}0})$

7.1.5 Remarks

- The SPA is also called *belief propagation* or *message passing.* On trees, it is an exact inference algorithm.
- If G is not a tree, the algorithm doesn't converge in general to the right marginal laws, but sometimes gives reasonable approximations. We then refer to "Loopy belief propagation", which is still often used in real life.
- The only property that we have used to construct the algorithm is the fact that $(\mathbb{R}, +, \times)$ is a semi-ring. It is interesting to notice that the same can therefore also be done with $(\mathbb{R}_+, \max, \times)$ and $(\mathbb{R}, \max, +)$.

  Example For $(\mathbb{R}_+, \max, \times)$ we define the Max-Product algorithm, also called "Viterbi algorithm" which enables us to solve the *decoding* problem, namely to compute the most probable configuartion of the variables, given fixed parameters, thanks to the messages

$$\mu_{j\to i}(x_i) = \max_{x_j} \left[\psi_{i,j}(x_i, x_j)\psi_j(x_j) \prod_{x_k} \mu_{k\to j}(x_j) \right] \tag{7.39}$$

  If we run the Max-Product algorithm with respect to a chosen root, the *collection* phase of the messages to the root enables us to compute the maximal probability over all configurations, and if at each calculation of a message we have also kept the argmax, we can perform a *distribution* phase, which instead of propagating the messages, will consist of recursively calculating one of the configurations which will reach the maximum.

- In practice, we may be working on such small values that the computer will return errors. For instance, for k binary variables, the joint law $p(x_1, x_2 ... x_n) = \frac{1}{2^n}$ can take infinitesimal values for a large k. The solution is to work with logarithms: if $p = \sum_i p_i$, by setting $a_i = \log(p_i)$ we have:

$$\begin{aligned}\log(p) &= \log\left[\sum_i e^{a_i}\right] \\ \log(p) &= a_i^* + \log\left[\sum_i e^{(a_i - a_i^*)}\right] \qquad (7.40)\end{aligned}$$

With $a_i^* = \max_i a_i$. Using logarithms ensures a numerical stability.

7.1.6 Proof of the algorithm

We are going to prove that the SPA is correct by recurrence. In the case of two nodes, we have:

$$p(x_1, x_2) = \frac{1}{Z}\psi_1(x_i)\psi_2(x_2)\psi_{1,2}(x_1, x_2)$$

We marginalize, and we obtain

$$p(x_1) = \frac{1}{Z}\psi_1(x_1)\underbrace{\sum_{x_2}\psi_{1,2}(x_1, x_2)\psi_2(x_2)}_{\mu_{2\to1}(x_1)}$$

We can hence deduct

$$p(x_1) = \frac{1}{Z}\psi_1(x_1)\mu_{2\to1}(x_1)$$

And

$$p(x_2) = \frac{1}{Z}\psi_2(x_2)\mu_{1\to2}(x_2)$$

We assume that the result is true for trees of size $n-1$, and we consider a tree of size n. Without loss of generality, we can assume that the nodes are numbered, so that the $n-$th be a leaf, and we will call π_n its parent (which is unique, the graph being a tree). The first message to be passed is:

$$\mu_{n\to\pi_n}(x_{\pi_n}) = \sum_{x_n}\psi_n(x_n)\psi_{n,\pi_n}(x_n, x_{\pi_n}) \qquad (7.41)$$

And the last message to be passed is:

$$\mu_{\pi_n\to n}(x_n) = \sum_{x_{\pi_n}}\psi_{\pi_n}(x_{\pi_n})\psi_{n,\pi_n}(x_n, x_{\pi_n}) \prod_{k\in\mathcal{N}(\pi_n)\setminus\{n\}}\mu_{k\to\pi_n}(x_{\pi_n}) \qquad (7.42)$$

We are going to construct a tree $\tilde{T}$ of size $n-1$, as well as a family of potentials, such that the $2(n-2)$ messages passed in T (i.e. all the messages except for the first and the last) be equal to the $2(n-2)$ messages passed in $\tilde{T}$. We define the tree and the potentials as follows:

- $\tilde{T} = (\tilde{V}, \tilde{E})$ with $\tilde{V} = \{1, \dots, n-1\}$ and $\tilde{E} = E \backslash \{n, \pi_n\}$ (i.e., it is the subtree corresponding to the $n-1$ first vertices).
- The potentials are all the same as those of T, except for the potential
$$\tilde{\psi}_{\pi_n}(x_{\pi_n}) = \psi_{\pi_n}(x_{\pi_n})\mu_{n\to\pi_n}(x_{\pi_n}) \tag{7.43}$$
- The root is unchanged, and the topological order is also kept.

We then obtain two important properties:

1) The product of the potentials of the tree of size $n-1$ is equal to:

$$\begin{aligned}
\tilde{p}(x_1, \dots, x_{n-1}) &= \frac{1}{Z}\prod_{i\neq n,\pi_n}\psi_i(x_i)\prod_{(i,j)\in E\backslash\{n,\pi_n\}}\psi_{i,j}(x_i,x_j)\tilde{\psi}_{\pi_n}(x_{\pi_n}) \\
&= \frac{1}{Z}\prod_{i\neq n,\pi_n}\psi_i(x_i)\prod_{(i,j)\in E\backslash\{n,\pi_n\}}\psi_{i,j}(x_i,x_j)\sum_{x_n}\psi_n(x_n)\psi_{\pi_n}(x_{\pi_n})\psi_{n,\pi_n}(x_n,x_{\pi_n}) \\
&= \sum_{x_n}\frac{1}{Z}\prod_{i=1}^{n}\psi_i(x_i)\prod_{(i,j)\in E}\psi_{i,j}(x_i,x_j) \\
&= \sum_{x_n}p(x_1,\dots,x_{n-1},x_n)
\end{aligned}$$

which shows that these new potentials define on $(X_1, \dots, X_{n-1})$ exactly the distribution induced by p when marginalizing X_n.

2) All of the messages passed in $\tilde{T}$ correspond to the messages passed in T (except for the first and the last).

Now, with the recurrence hypothesis that the SPA is true for trees of size $n-1$, we are going to show that it is true for trees of size n. For nodes $i \neq n, \pi_n$, the result is obvious, as all messages passed are the same:

$$\forall\ i \in V\backslash\{n,\pi_n\},\ p(x_i) = \frac{1}{Z}\ \psi_i(x_i)\prod_{k\in\mathcal{N}(i)}\mu_{k\to i}(x_i) \tag{7.44}$$

For the case $i = \pi_n$, we deduct:

$$\begin{aligned}
p(x_{\pi_n}) &= \frac{1}{Z}\tilde{\psi}_{\pi_n}(x_{\pi_n}) \prod_{k \in \tilde{\mathcal{N}}(\pi_n)} \mu_{k\to\pi_n}(x_{\pi_n}) \quad (\text{ product over the neighbours of } \pi_n \text{ in } \tilde{T}) \\
&= \frac{1}{Z}\tilde{\psi}_{\pi_n}(x_{\pi_n}) \prod_{k \in \mathcal{N}(\pi_n)\setminus\{n\}} \mu_{k\to\pi_n}(x_{\pi_n}) \\
&= \frac{1}{Z}\psi_{\pi_n}(x_{\pi_n})\mu_{n\to\pi_n}(x_{\pi_n}) \prod_{k \in \mathcal{N}(\pi_n)\setminus\{n\}} \mu_{k\to\pi_n}(x_{\pi_n}) \\
&= \frac{1}{Z}\psi_{\pi_n}(x_{\pi_n}) \prod_{k \in \mathcal{N}(\pi_n)} \mu_{k\to\pi_n}(x_{\pi_n})
\end{aligned}$$

For the case $i = n$, we have:

$$p(x_n, x_{\pi_n}) = \sum_{x_{V\setminus\{n,\pi_n\}}} p(x) = \psi_n(x_n)\psi_{\pi_n}(x_{\pi_n})\psi_{n,\pi_n}(x_n, x_{\pi_n}) \underbrace{\sum_{x_{V\setminus\{n,\pi_n\}}} \frac{p(x)}{\psi_n(x_n)\psi_{\pi_n}(x_{\pi_n})\psi_{n,\pi_n}(x_n, x_{\pi_n})}}_{\alpha(x_{\pi_n})}$$

Therefore:

$$p(x_n, x_{\pi_n}) = \psi_{\pi_n}(x_{\pi_n})\alpha(x_{\pi_n})\psi_n(x_n)\psi_{n,\pi_n}(x_n, x_{\pi_n}) \tag{7.45}$$

Consequently:

$$p(x_{\pi_n}) = \psi_{\pi_n}(x_{\pi_n})\alpha(x_{\pi_n}) \underbrace{\sum_{x_n} \psi_n(x_n)\psi_{n,\pi_n}(x_n, x_{\pi_n})}_{\mu_{n\to\pi_n}(x_{\pi_n})}$$

Hence:

$$\alpha(x_{\pi_n}) = \frac{p(x_{\pi_n})}{\psi_{\pi_n}(x_{\pi_n})\mu_{n\to\pi_n}(x_{\pi_n})} \tag{7.46}$$

By using (7.31), (7.32) and the previous result, we deduct that:

$$\begin{aligned}
p(x_n, x_{\pi_n}) &= \psi_{\pi_n}(x_{\pi_n})\psi_n(x_n)\psi_{n,\pi_n}(x_n, x_{\pi_n}) \frac{p(x_{\pi_n})}{\psi_{\pi_n}(x_{\pi_n})\mu_{n\to\pi_n}(x_{\pi_n})} \\
&= \psi_{\pi_n}(x_{\pi_n})\psi_n(x_n)\psi_{n,\pi_n}(x_n, x_{\pi_n}) \frac{\frac{1}{Z}\psi_{\pi_n}(x_{\pi_n}) \prod_{k \in \mathcal{N}(\pi_n)} \mu_{k\to\pi_n}(x_{\pi_n})}{\psi_{\pi_n}(x_{\pi_n})\mu_{n\to\pi_n}(x_{\pi_n})} \\
&= \frac{1}{Z}\psi_{\pi_n}(x_{\pi_n})\psi_n(x_n)\psi_{n,\pi_n}(x_n, x_{\pi_n}) \prod_{k \in \mathcal{N}(\pi_n)\setminus\{n\}} \mu_{k\to\pi_n}(x_{\pi_n})
\end{aligned}$$

By summing with respect to x_{π_n}, we get the result for $p(x_n)$:

$$p(x_n) = \sum_{x_{\pi_n}} p(x_n, x_{\pi_n}) = \frac{1}{Z}\psi_n(x_n)\mu_{\pi_n \to n}(x_n)$$

Proposition:

Let $p \in \mathcal{L}(G)$, for $G = (V, E)$ a tree, then we have:

$$p(x_1, \dots, x_n) = \frac{1}{Z}\prod_{i \in V} \psi(x_i) \prod_{(i,j)\in E} \frac{p(x_i, x_j)}{p(x_i)p(x_j)} \tag{7.47}$$

Proof: we prove it by reccurence. The case $n = 1$ is trivial. Then, assuming that n is a leaf, and we can write $p(x_1, \dots, x_n) = p(x_1, \dots, x_{n-1})\, p(x_n|x_{\pi_n})$. But multiplying by $p(x_n|x_{\pi_n}) = \frac{p(x_n, x_{\pi_n})}{p(x_n)p(x_{\pi_n})}\, p(x_n)$ boils down to adding the edge potential for (n, π_n) and the node potential for the leaf n. The formula is hence verified by reccurence.

7.1.7 Junction tree

Junction tree is an algorithm designed to tackle the problem of *inference on general graphs.* The idea is to look at a general graph from far away, where it can be seen as a tree. By merging nodes, one will hopefully be able to build a tree. When this is not the case, one can also think of adding some edges to the graph (i.e., cast the present distribution into a larger set) to be able to build such a graph.

The trap is that if one collapses too many nodes, the number of possibles values will explode, and as such the complexity of the whole algorithm. The *tree width* is the smallest possible clique size. For instance, for a 2D regular grid with n points, the tree width is equal to $\sqrt{n}$.

7.2 Hidden Markov Model (HMM)

The Hidden Markov Model ("Modèle de Markov caché" in French) is one of the most used Graphical Models. We note $z_0, z_1, \dots, z_T$ the states corresponding to the latent variables, and $y_0, y_1, \dots, y_T$ the states corresponding to the observed variables. We further assume that:

1. $\{z_0, \dots, z_T\}$ is a Markov chain (hence the name of the model);

2. $z_0, \dots, z_T$ take K values;

3. z_0 follows a multinomial distribution: $p(z_0 = i) = (\pi_0)_i$ with $\sum_i (\pi_0)_i = 1$;

4. The *transition* probabilities are homogeneous: $p(z_t = i|z_{t-1} = j) = A_{ij}$, where A satisfies $\sum_i A_{ij} = 1$;

5. The emission probabilities $p(y_t|z_t)$ are homogeneous, *i.e.* $p(y_t|z_t) = f(y_t, z_t)$.

6. The joint probability distribution function can be written as:

$$p(z_0, ...z_T, y_0, ..., y_T) = p(z_0) \prod_{t=0}^{T-1} p(z_{t+1}|z_t) \prod_{t=0}^{T} p(y_t|z_t).$$

There are different tasks that we want to perform on this model:

- Filtering: $p(z_t|y_1, \dots, y_{t-1})$
- Smoothing: $p(z_t|y_1, \dots, y_T)$
- Decoding: $\max_{z_0,\dots,z_T} p(z_0, \dots, z_T|y_0, \dots, y_T)$

All these tasks can be performed with a sum-product or max-product algorithm.

Sum-product

From now on, we note the observations $\overline{y} = \{\overline{y_1}, \dots, \overline{y_T}\}$. The distribution on y_t simply becomes the delta function $\delta(y_t = \overline{y_t})$. To use the sum-product algorithm, we define z_T as the root, *i.e.* we send all forward messages to z_T and go back afterwards.

Forward:

$$\begin{aligned}
\mu_{y_0 \to z_0}(z_0) &= \sum_{y_0} \delta(y_0 = \overline{y_0}) p(y_0|z_0) = p(\overline{y_0}|z_0) \\
\mu_{z_0 \to z_1}(z_1) &= \sum_{z_0} p(z_1|z_0) \mu_{y_0 \to z_0}(z_0) \\
&\vdots \\
\mu_{y_{t-1} \to z_{t-1}}(z_{t-1}) &= \sum_{y_{t-1}} \delta(y_{t-1} = \overline{y_{t-1}}) p(y_{t-1}|z_{t-1}) = p(\overline{y_{t-1}}|z_{t-1}) \\
\mu_{z_{t-1} \to z_t}(z_t) &= \sum_{z_{t-1}} p(z_t|z_{t-1}) \mu_{z_{t-2} \to z_{t-1}}(z_{t-1})\ p(z_{t-1}|y_{t-1})
\end{aligned}$$

Let us define the "alpha-message" $\alpha_t(z_t)$ as:

$$\alpha_t(z_t) = \mu_{y_t \to z_t}(z_t) \mu_{z_{t-1} \to z_t}(z_t)$$

$\alpha_0(z_0)$ is initialized with the virtual message $\mu_{z_{-1} \to z_0}(z_0) = p(z_0)$

Property 7.1

$$\alpha_t(z_t) = \mu_{y_t \to z_t}(z_t)\mu_{z_{t-1} \to z_t}(z_t) = p(z_t, \overline{y_0}, \ldots, \overline{z_t})$$

This is due to the definition of the messages: the product $\mu_{y_t \to z_t}(z_t)\mu_{z_{t-1} \to z_t}(z_t)$ represents a marginal of the distribution corresponding to the sub-HMM $\{z_0, \ldots, z_t\}$.
Moreover, the following recursion formula for $\alpha_t(z_t)$ (called "alpha-recursion") holds:

$$\alpha_{t+1}(z_{t+1}) = p(\overline{y_{t+1}}|z_{t+1}) \sum_{z_t} p(z_{t+1}|z_t)\alpha(z_t)$$

Backward:

$$\mu_{z_{t+1} \to z_t}(z_t) = \sum_{z_{t+1}} p(z_{t+1}|z_t)\mu_{z_{t+2} \to z_{t+1}}(z_{t+1})p(\overline{y_{t+1}}|z_{t+1})$$

We define the "beta-message" $\beta_t(z_t)$ as:

$$\beta_t(z_t) = \mu_{z_{t+1} \to z_t}(z_t)$$

As an initialization, we take $\beta_T(z_T) = 1$. The following recursion formula for $\beta_t(z_t)$ (called "beta-recursion") holds:

$$\beta_t(z_t) = \sum_{z_{t+1}} p(z_{t+1}|z_t)p(\overline{y_{t+1}}|z_{t+1})\beta_{t+1}(z_{t+1})$$

Property 7.2 *1.* $p(z_t, \overline{y_0}, \ldots, \overline{y_T}) = \alpha_t(z_t)\beta_t(z_t)$

2. $p(\overline{y_0}, \ldots, \overline{y_T}) = \sum\limits_{z_t} \alpha_t(z_t)\beta_t(z_t)$

3. $p(z_t|\overline{y_0}, \ldots, \overline{y_T}) = \frac{p(z_t, \overline{y_0}, \ldots, \overline{y_T})}{p(\overline{y_0}, \ldots, \overline{y_T})} = \frac{\alpha_t(z_t)\beta_t(z_t)}{\sum\limits_{z_t} \alpha_t(z_t)\beta_t(z_t)}$

4. For all $t < T$, $p(z_t, z_{t+1}|\overline{y_0}, \ldots, \overline{y_T}) = \frac{1}{p(\overline{y_0}, \ldots, \overline{y_T})}\, \alpha_t(z_t)\, \beta_{t+1}(z_{t+1})\, p(z_{t+1}|z_t)\, p(\overline{y_{t+1}}|z_{t+1})$

Remark 7.3 *1. The alpha-recursion and beta-recursion are easy to implement, but one needs to avoid errors in the indices!*

2. The sums need to be coded using logs in order to prevent numerical errors.

EM algorithm

With the previous notations and assumptions, we write the complete log-likelihood $l_c(\theta)$ with θ the parameters of the model (containing (π_0, A), but also parameters for f):

$$\begin{aligned} l_c(\theta) &= \log\Big(p(z_0)\prod_{t=0}^{T-1} p(z_{t+1}|z_t)\prod_{t=0}^{T} p(\overline{y_t}|z_t)\Big) \\ &= \log(p(z_0)) + \sum_{t=0}^{T}\log p(\overline{y_t}|z_t) + \sum_{t=0}^{T}\log p(z_{t+1}|z_t) \\ &= \sum_{i=1}^{K}\delta(z_0=i)\log((\pi_0)_i) + \sum_{t=0}^{T-1}\sum_{i,j=1}^{K}\delta(z_{t+1}=i, z_t=j)\log(A_{i,j}) + \sum_{t=0}^{T}\sum_{i=1}^{K}\delta(z_t=i)\log f(\overline{y_t}, z_t) \end{aligned}$$

When applying E-M to estimate the parameters of this HMM, we use Jensen's inequality to obtain a lower bound on the log-likelihood:

$$\log p(\overline{y_0},\dots,\overline{y_T}) \geq \mathbb{E}_q[\log p(z_0,\dots,z_T,\overline{y_0},\dots,\overline{y_T})] = \mathbb{E}_q[l_c(\theta)]$$

At the k-th expectation step, we use $q(z_0,\dots,z_T) = \mathbb{P}(z_0,\dots,z_T|\overline{y_0},\dots,\overline{y_T};\theta_{k-1})$, and this boils down to applying the following rules:

- $\mathbb{E}[\delta(z_0=i)|\overline{y}] = p(z_0=i|\overline{y};\theta^{k-1})$

- $\mathbb{E}[\delta(z_t=i)|\overline{y}] = p(z_t=i|\overline{y};\theta^{k-1})$

- $\mathbb{E}[\delta(z_{t+1}=i, z_t=j|\overline{y};\theta^{k-1}] = p(z_{t+1}=i, z_t=j|\overline{y};\theta^{k-1})$

Thus, in the former expression of the complete log-likelihood, we just have to replace $\delta(z_0=i)$ by $p(z_0=i|\overline{y};\theta^{k-1})$, and similarly for the other terms.
At the k-th maximization step, we maximize the new obtained expression with respect to the parameters θ in the usual manner to obtain a new estimator θ^k. The key is that everything will decouple, thus maximizing is simple and can be done in closed form.

7.3 Principal Component Analysis (PCA)

Framework: $x_1,\dots,x_N \in \mathbb{R}^d$
Goal: put points on a closest affine subspace

Analysis view

Find $w \in \mathbb{R}^d$ such that $\mathrm{Var}(x^T w)$ is maximal, with $||w|| = 1$

With centered data, *i.e.* $\frac{1}{N}\sum_{n=1}^{N} x_n = 0$, the empirical variance is:

$$\hat{\text{Var}}(x^T w) = \frac{1}{N}\sum_{n=1}^{N}(x_n^T w)^2 = \frac{1}{N} w^T (X^T X) w$$

where $X \in \mathbb{R}^{N \times d}$ is the design matrix. In this case: w is the eigenvector of $X^T X$ with largest eigenvalue. It is not obvious *a priori* that this is the direction we care about.
If more than one direction is required, one can use *deflation*:

1. Find w
2. Project x_n onto the orthogonal of Vect(w)
3. Start again

Synthesis view

$$\min_w \sum_{n=1}^{N} d(x_n, \{w^T x = 0\})^2 \text{ with } w \in \mathbb{R}^D,\ ||w|| = 1.$$

Advantage: if one wants more than 1 dimension, replace $\{w^T x = 0\}$ by any subspace.

Probabilistic approach: Factor Analysis

Model:

- $\Lambda = (\lambda_1, \dots, \lambda_K) \in \mathbb{R}^{d \times k}$
- $X \in \mathbb{R}^k \sim \mathcal{N}(0, I)$
- $\epsilon \sim \mathcal{N}(0, \Psi)$, $\epsilon \in \mathbb{R}^d$ independent from X with Ψ diagonal.
- $Y \in \mathbb{R}^d$: $Y = \Lambda X + \mu + \epsilon$

We have $Y|X \sim \mathcal{N}(\Lambda X + \mu, \Psi)$.
Problem: get $X|Y$.

(X, Y) is a Gaussian vector on $\mathbb{R}^{d+k}$ which satisfies:

- $\mathbb{E}[X] = 0 = \mu_X$
- $\mathbb{E}[Y] = \mathbb{E}[\Lambda X + \mu + \epsilon]\mu = \mu_Y$

- $\Sigma_{XX} = I$
- $\Sigma_{XY} = \mathrm{Cov}(X, \Lambda X + \mu + \epsilon) = \mathrm{Cov}(X, \Lambda X) = \Lambda^T$
- $\Sigma_{YY} = \mathrm{Var}(\Lambda X + \epsilon, \Lambda X + \epsilon) = \mathrm{Var}(\Lambda X, \Lambda X) + \mathrm{Var}(\epsilon, \epsilon) = \Lambda\Lambda^T + \Psi$

Thanks to the results we know on exponential families, we know how to compute $X|Y$:

$$\begin{aligned}
\mathbb{E}[X|Y=y] &= \mu_X + \Sigma_{XY}\Sigma_{YY}^{-1}(y - \mu_Y) \\
\mathrm{Cov}[X|Y=y] &= \Sigma_{XX} - \Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{YX}
\end{aligned}$$

In our case, we therefore have:

$$\begin{aligned}
\mathbb{E}[X|Y=y] &= \Lambda^T(\Lambda\Lambda^T + \Psi)^{-1}(y - \mu) \\
\mathrm{Cov}[X|Y=y] &= I - \Lambda^T(\Lambda\Lambda^T + \Psi)^{-1}\Lambda^T
\end{aligned}$$

To apply EM, one needs to write down the complete log-likelihood.

$$\log p(X,Y) \alpha - \frac{1}{2}X^TX - \frac{1}{2}(Y - \Lambda X - \mu)^T\Psi^{-1}(Y - \Lambda X - \mu) - \frac{1}{2}\log\det\Psi$$

Trap: $\mathbb{E}[XX^T|Y] \neq \mathrm{Cov}(X|Y)$
Rather, $\mathbb{E}[XX^T|Y] = \mathrm{Cov}(X|Y) + \mathbb{E}[X|Y]\mathbb{E}[X|Y]^T$

Remark 7.4

- $\mathrm{Cov}(X) = \Lambda\Lambda^T + \Psi$*: our parameters are not identifiable,* $\Lambda \leftarrow \Lambda R$ *with* R *a rotation gives the same results (in other words, a subspace has different orthonormal bases).*
- *Why do we care ?*
  1. *A probabilistic interpretation allows to model in a finer way the problem.*
  2. *It is very flexible and therefore allows to combine multiple models.*

Bibliography