Algorithme Somme Produit et modèle de Markov Caché **2011/2012**

Cours 7 — 9 novembre 2011

Enseignant: Guillaume Obozinski *Scribe: Piotr Bojanowski, Yin Chen*

7.1 Algorithme somme-produit

7.1.1 Motivations

L'*inférence* est avec l'*estimation* et le *décodage*, l'une des trois opérations qu'il est essentiel de pouvoir réaliser efficacement dans les modèles graphiques. Etant donné un modèle de Gibbs discret de la forme $p(x) = \frac{1}{Z}\prod_{C\in\mathcal{C}} \psi_C(x_C)$, avec $\mathcal{C}$ l'ensemble des cliques d'un graphe, l'inférence permets

- De calculer une marginale $p(x_i)$ ou bien plus généralement $p(x_C)$.
- De calculer la fonction de partition Z
- De calculer une marginale conditionnelle $p(x_i|X_j = x_j, X_k = x_k)$

Et comme conséquence des précédents:

- De calculer le gradient dans une famille exponentielle
- De calculer l'espérance de la log-vraisemblance d'une famille exponentielle à l'étape E de l'algorithme EM (comme par exemple pour les HMM)

Exemple 1: le modèle d'Ising

Soit un vecteur $X = (X_i)_{i\in V}$ de variables aléatoire à valeurs dans $\{0,1\}^{|V|}$, dont la distribution à la forme exponentielle

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

Nous avons donc la log-vraisemblance :

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

ainsi nous pouvons écrire la statistique exhaustive :

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

or nous avons vu pour les familles exponentielles que

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

Donc on souhaite calculer $\mathbb{E}_\eta[\phi(X)]$ ce qui dans le cas du modèle d'Ising donne :

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

L'équation 7.6 est l'une des motivations pour résoudre le problème l'inférence. Afin de pouvoir calculer le gradient de la log-vraisemblance, nous devons connaitre les lois marginales.

Exemple 2: le modèle de Potts

Les X_i sont des variables aléatoires à valeurs dans $\{1, \dots, K_i\}$. On note Δ_{ik} la variable aléatoire telle que $\Delta_{ik} = 1$ si et seulement si $X_i = k$. Alors

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

et

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

Tout ceci motive le besoin de savoir faire de l'inférence.

Problème: Le problème d'inférence est un problème NP-dur dans le cas général.

Dans le cas des arbres le problème d'inférence est efficace car linéaire en n.

Pour les graphes "presque arbres" on utilise l'algorithme de l'*arbre de jonction* (*Junction Tree Algorithm*) qui permet de se ramener à la situation d'un arbre.

Dans le cas général on est obligé de faire de l'inférence approchée.

7.1.2 Inférence sur une chaine

On note X_i une variable aléatoire a valeurs dans $\{1, \dots, K\}$, $i \in V = \{1, \dots, n\}$.

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

On souhaite calculer $p(x_j)$ pour un certain j. La solution naïve serait de calculer la marginale

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

Malheureusement ce genre de calculs présente une complexité de $O(K^n)$. On développe donc l'expression

$$p(x_j) = \frac{1}{Z} \sum_{x_{V \setminus \{j\}}} \prod_{i=1}^{n} \psi_i(x_i) \prod_{i=2}^{n} \psi_{i-1,i}(x_{i-1}, x_i) \tag{7.14}$$

$$= \frac{1}{Z} \sum_{x_{V \setminus \{j\}}} \prod_{i=1}^{n-1} \psi_i(x_i) \prod_{i=2}^{n-1} \psi_{i-1,i}(x_{i-1}, x_i) \psi_n(x_n) \psi_{n-1,n}(x_{n-1}, x_n) \tag{7.15}$$

$$= \frac{1}{Z} \sum_{x_{V \setminus \{j,n\}}} \sum_{x_n} \prod_{i=1}^{n-1} \psi_i(x_i) \prod_{i=2}^{n-1} \psi_{i-1,i}(x_{i-1}, x_i) \psi_n(x_n) \psi_{n-1,n}(x_{n-1}, x_n) \tag{7.16}$$

Ce qui nous permet de faire apparaitre le message envoyé par (n) a $(n-1)$: $\mu_{n \to n-1}(x_{n-1})$. En continuant, on obtient :

$$p(x_j) = \frac{1}{Z} \sum_{x_{V \setminus \{j,n\}}} \prod_{i=1}^{n-1} \psi_i(x_i) \prod_{i=2}^{n-1} \psi_{i-1,i}(x_{i-1}, x_i) \underbrace{\sum_{x_n} \psi_n(x_n) \psi_{n-1,n}(x_{n-1}, x_n)}_{\mu_{n \to n-1}(x_{n-1})} \tag{7.17}$$

$$= \frac{1}{Z} \sum_{x_{V \setminus \{j,n,n-1\}}} \prod_{i=1}^{n-2} \psi_i(x_i) \prod_{i=2}^{n-2} \psi_{i-1,i}(x_{i-1}, x_i) \times$$
$$\times \underbrace{\sum_{x_{n-1}} \psi_{n-1}(x_{n-1}) \psi_{n-2,n-1}(x_{n-2}, x_{n-1}) \mu_{n \to n-1}(x_{n-1})}_{\mu_{n-1 \to n-2}(x_{n-2})} \tag{7.18}$$

$$= \frac{1}{Z} \sum_{x_{V \setminus \{1,j,n,n-1\}}} \mu_{1 \to 2}(x_{n-2}) \ \dots \ \mu_{n-1 \to n-2}(x_{n-2}) \tag{7.19}$$

Et finalement on obtient :

$$\boxed{p(x_j) = \frac{1}{Z}\ \mu_{j-1\to j}(x_j)\ \psi_j(x_j)\ \mu_{j+1\to j}(x_j)} \tag{7.20}$$

Avec uniquement $2(n-1)$ messages on a calculé $p(x_j)\ \forall j \in V$. Z se calcule en sommant :

$$Z = \sum_{x_i}\ \mu_{i-1\to i}(x_i)\ \psi_i(x_i)\ \mu_{i+1\to i}(x_i) \tag{7.21}$$

7.1.3 Inférence dans les arbres non orientés

On note i le noeud dont on veut calculer la loi marginale $p(x_i)$. On décide également que i est la racine de notre arbre. $\forall\ j \in V$, on note $\mathcal{C}(j)$ l'ensemble des enfants de j et $\mathcal{D}(j)$ l'ensemble des descendants de j. Nous avons la loi jointe :

$$p(x) = \frac{1}{Z}\prod_{i\in V}\psi_i(x_i)\prod_{(i,j)\in E}\psi_{i,j}(x_i,x_j) \tag{7.22}$$

Pour un arbre avec au moins 2 noeuds on définit par récurrence:

$$F(x_i, x_j, x_{\mathcal{D}(j)}) \triangleq \psi_{i,j}(x_i,x_j)\psi_j(x_j)\prod_{k\in\mathcal{C}(j)} F(x_j,x_k,x_{\mathcal{D}(k)}) \tag{7.23}$$

Puis en réécrivant la loi marginale

$$\begin{aligned}
p(x_i) &= \frac{1}{Z}\sum_{x_{V\setminus\{i\}}}\psi_i(x_i)\prod_{j\in\mathcal{C}(i)}F(x_i,x_j,x_{\mathcal{D}(j)}) && (7.24)\\
&= \frac{1}{Z}\ \psi_i(x_i)\prod_{j\in\mathcal{C}(i)}\sum_{x_j,x_{\mathcal{D}(j)}}F(x_i,x_j,x_{\mathcal{D}(j)}) && (7.25)\\
&= \frac{1}{Z}\ \psi_i(x_i)\prod_{j\in\mathcal{C}(i)}\sum_{x_j,x_{\mathcal{D}(j)}}\psi_{i,j}(x_i,x_j)\psi_j(x_j)\prod_{k\in\mathcal{C}(j)}F(x_j,x_k,x_{\mathcal{D}(k)}) && (7.26)\\
&= \frac{1}{Z}\ \psi_i(x_i)\prod_{j\in\mathcal{C}(i)}\sum_{x_j}\psi_{i,j}(x_i,x_j)\psi_j(x_j)\prod_{k\in\mathcal{C}(j)}\underbrace{\sum_{x_k,x_{\mathcal{D}(k)}}F(x_j,x_k,x_{\mathcal{D}(k)})}_{\mu_{k\to j}(x_j)} && (7.27)\\
&= \frac{1}{Z}\ \psi_i(x_i)\prod_{j\in\mathcal{C}(i)}\underbrace{\sum_{x_j}\psi_{i,j}(x_i,x_j)\psi_j(x_j)\prod_{k\in\mathcal{C}(j)}\mu_{k\to j}(x_j)}_{\mu_{j\to i}(x_i)} && (7.28)
\end{aligned}$$

Ce qui nous donne la formule de récurrence pour l'Algorithme Somme-Produit (ASP) :

$$\boxed{\mu_{j\to i}(x_i) = \sum_{x_j} \psi_{i,j}(x_i, x_j)\psi_j(x_j) \prod_{k\in\mathcal{C}(j)} \mu_{k\to j}(x_j)} \tag{7.29}$$

7.1.4 Algorithme Somme-Produit (ASP)

ASP séquentiel pour un arbre enraciné

Pour un arbre enraciné, de racine (i), l'algorithme Somme-Produit s'écrit de la manière suivante :

1. Toutes les feuilles calculent $\mu_{n\to\pi_n}(x_{\pi_n})$

$$\mu_{n\to\pi_n}(x_{\pi_n}) = \sum_{x_n} \psi_n(x_n)\psi_{n,\pi_n}(x_n, x_{\pi_n}) \tag{7.30}$$

2. Itérativement, à chaque pas de temps, tous les noeuds (k) qui ont reçu des messages de tous leurs enfants envoient $\mu_{k\to\pi_k}(x_{\pi_k})$ a leur parent

3. A la racine, on a

$$p(x_i) = \frac{1}{Z}\ \psi_i(x_i) \prod_{j\in\mathcal{C}(i)} \mu_{j\to i}(x_i) \tag{7.31}$$

Cette algorithme permet seulement de calculer la marginale $p(x_i)$ à la racine. Pour pouvoir calculer toutes les marginales (ainsi que les marginales pour les arêtes), il faut non seulement *collecter* les messages depuis les feuilles jusqu'à la racine, mais ensuite les redistribuer jusqu'au feuilles. En fait, l'algorithme peut alors être écrit indépendamment du choix d'une racine:

ASP pour un arbre non orienté

Le cas des arbres non orientés est légèrement différent :

1. Toutes les feuilles envoient $\mu_{n\to\pi_n}(x_{\pi_n})$

2. À chaque pas de temps, si un noeud (j) n'a pas encore envoyé de message à l'un de ses voisins, disons (i) (Attention, i ne désigne pas ici la racine!), et s'il a reçu les messages de tous les autres voisins $\mathcal{N}(j)\backslash i$, il envoie à (i) le message

$$\mu_{j\to i}(x_i) = \sum_{x_j} \psi_j(x_j)\psi_{j,i}(x_i, x_i) \prod_{k\in\mathcal{N}(j)\backslash\{i\}} \mu_{k\to j}(x_j) \tag{7.32}$$

ASP parallèle (flooding)

1. Initialiser les messages de façon quelconque
2. À chaque pas de temps, chaque noeud envoie un nouveau message à chacun de ses voisins en utilisant les messages reçu à l'itération précédente:

Marginales

Une fois tous les messages passés, nous pouvons aisément calculer l'ensemble des lois marginales

$$\forall\ i \in V,\ p(x_i) = \frac{1}{Z}\ \psi_i(x_i) \prod_{k \in \mathcal{N}(i)} \mu_{k \to i}(x_i) \tag{7.33}$$

$$\forall\ (i,j) \in E,\ p(x_i, x_j) = \frac{1}{Z}\ \psi_i(x_i)\ \psi_j(x_j)\ \psi_{j,i}(x_i, x_i) \prod_{k \in \mathcal{N}(i) \setminus j} \mu_{k \to i}(x_i) \prod_{k \in \mathcal{N}(j) \setminus i} \mu_{k \to j}(x_j) \tag{7.34}$$

Probabilités conditionnelles

On peut utiliser une notation astucieuse pour calculer les probabilités conditionnelles. Supposons que l'on s'intéresse à

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

On peut poser

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

Soit de manière générale, si on observe $X_j = x_{j0}$ pour $j \in J_{\text{obs}}$, on peut définir des potentiels modifiés:

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

de telle sorte que

$$p(x | X_{J_{\text{obs}}} = x_{J_{\text{obs}}0}) = \frac{1}{\tilde{Z}} \prod_{i \in V} \tilde{\psi_i}(x_i) \prod_{(i,j) \in E} \psi_{i,j}(x_i, x_j) \tag{7.35}$$

En effet, on a

$$p(x | X_{J_{\text{obs}}} = x_{J_{\text{obs}}0})\ p(X_{J_{\text{obs}}} = x_{J_{\text{obs}}0}) = p(x) \prod_{j \in J_{\text{obs}}} \delta(x_j, x_{j0}) \tag{7.36}$$

de telle sorte qu'en divisant l'égalité par $p(X_{J_{\text{obs}}} = x_{J_{\text{obs}}0})$ on obtient l'équation précédente avec $\tilde{Z} = Zp(X_{J_{\text{obs}}} = x_{J_{\text{obs}}0})$.

Il suffit donc d'appliquer l'algorithme somme produit à ces nouveaux potentiels pour calculer les marginales $p(x_i | X_{J_{\text{obs}}} = x_{J_{\text{obs}}0})$

7.1.5 Remarques

- L'algorithme ASP est également appelé *belief propagation* ou "Algorithme de Propagation de Croyance". C'est un algorithme exact d'inférence sur les arbres.

- Si G n'est pas un arbre, l'algorithme ne converge pas en général vers les bonnes marginales, mais donne parfois des approximations raisonables. On parle alors de "Loopy belief propagation" et cet algorithme est néanmoins beaucoup utilisé en pratique.

- L'unique propriété que nous avons utilisée pour construire l'algorithme est le fait que $(\mathbb{R}, +, \times)$ est un semi-anneau. Il peut être intéressant de remarquer qu'il en est de même pour $(\mathbb{R}_+, \max, \times)$ et $(\mathbb{R}, \max, +)$.

  exemple Pour $(\mathbb{R}_+, \max, \times)$ on définit l'algorithme Max-Produit, encore appelé "algorithme de Viterbi" qui permet de résoudre le problème du *décodage*, c'est à dire de calculer la configuration des variables la plus probable étant donnés des paramètres fixés, par les messages

$$\mu_{j\to i}(x_i) = \max_{x_j}\left[\psi_{i,j}(x_i, x_j)\psi_j(x_j)\prod_{x_k}\mu_{k\to j}(x_j)\right] \tag{7.37}$$

  Si on effectue l'algorithme max-produit par rapport à une racine que l'on choisira, la phase de *collecte* des messages vers la racine permet de calculer la probabilité maximale sur toutes les configurations, et, si à chaque calcul de message on a également enregistré les argmax, on peut effectuer une phase de *distribution* qui au lieu de propager des messages, permet récursivement de calculer l'une des configurations qui atteint le maximum.

- Dans la pratique, on est susceptible de travailler avec des valeurs tellement faibles que l'ordinateur risque de renvoyer des erreurs. A titre d'exemple, pour k variables binaires, la loi jointe $p(x_1, x_2...x_n) = \dfrac{1}{2^n}$ peut prendre des valeurs négligeables pour k grand. La solution est de travailler en logarithme : si $p = \sum_i p_i$, en posant $a_i = \log(p_i)$ on a :

$$\log(p) = \log\left[\sum_i e^{a_i}\right]$$
$$\log(p) = a_i^* + \log\left[\sum_i e^{(a_i - a_i^*)}\right] \tag{7.38}$$

  Avec $a_i^* = \max_i a_i$. Passer au logarithme assure une stabilité numérique.

7.1.6 Preuve de l'algorithme

On va montrer que l'algorithme ASP est correct par récurrence. Pour le cas de 2 noeuds, on a:

$$p(x_1, x_2) = \frac{1}{Z}\psi_1(x_i)\psi_2(x_2)\psi_{12}(x_1, x_2)$$

En marginalisant, on obtient :

$$p(x_1) = \frac{1}{Z}\psi_1(x_1)\underbrace{\sum_{x_2}\psi_{12}(x_1, x_2)\psi_2(x_2)}_{\mu_{2\to 1}(x_1)}$$

Donc, on en déduit:

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

Et

$$p(x_2) = \frac{1}{Z}\psi_2(x_2)\mu_{1\to 2}(x_2)$$

On suppose que le résultat est vrai pour les arbres de taille $n-1$, et on considère un arbre de taille n. Sans nuire à la généralité on peut supposer que les noeuds sont numérotés de telle sorte que le $n-$ième soit une feuille, et on appellera π_n son parent (π_n est unique car le graphe est un arbre). Le premier message passé est :

$$\mu_{n\to\pi_n}(x_{\pi_n}) = \sum_{x_n}\psi_n(x_n)\psi_{n\pi_n}(x_n, x_{\pi_n}) \tag{7.39}$$

Et le dernier message passé est:

$$\mu_{\pi_n\to n}(x_n) = \sum_{x_{\pi_n}}\psi_{\pi_n}(x_{\pi_n})\psi_{n\pi_n}(x_n, x_{\pi_n})\prod_{k\in\mathcal{N}(\pi_n)\setminus\{n\}}\mu_{k\to\pi_n}(x_{\pi_n}) \tag{7.40}$$

Nous allons construire un arbre $\tilde{T}$ de taille $n-1$, ainsi qu'une famille de potentiels, de telle sorte que les $2(n-2)$ messages passés dans T (i.e., tous les messages exceptés le premier et le dernier) soient égaux aux $2(n-2)$ messages passés dans $\tilde{T}$. On définit l'arbre et les potentiels comme suit:

- $\tilde{T} = (\tilde{V}, \tilde{E})$ avec $\tilde{V} = \{1, \ldots, n-1\}$ et $\tilde{E} = E\setminus\{n, \pi_n\}$ (i.e., c'est le sous-arbre correspondant aux $n-1$ premiers sommets).
- Les potentiels sont tous les mêmes que ceux de T, excepté le potentiel

$$\tilde{\psi}_{\pi_n}(x_{\pi_n}) = \psi_{\pi_n}(x_{\pi_n})\mu_{n\to\pi_n}(x_{\pi_n}) \tag{7.41}$$

On peut obtenir deux propriétés importantes:

1) Le produit des potentiels de l'arbre de taille $n-1$, est égal à:

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

ce qui montre que ces nouveaux potentiels définissent sur $(X_1,\ldots,X_{n-1})$ exactement la distribution induite par p en marginalisant X_n.

2) Tous les message passés dans $\tilde{T}$ correspondent aux message passés dans T (hormis le premier et le dernier).

Maintenant, en faisant l'hypothèse de récurrence que l'algorithme somme produit est correct pour les arbres de taille $n-1$, nous allons montrer qu'il est correct pour les arbres de taille n. Pour les noeuds $i \neq n, \pi_n$, le résultat est évident, puisque les messages passés sont les mêmes:

$$\forall\ i \in V\setminus\{n,\pi_n\},\ p(x_i) = \frac{1}{Z}\ \psi_i(x_i)\prod_{k\in\mathcal{N}(i)}\mu_{k\to i}(x_i) \tag{7.42}$$

Pour le cas $i=\pi_n$, on déduit:

$$\begin{aligned}
p(x_{\pi_n}) &= \frac{1}{Z}\tilde{\psi}_{\pi_n}(x_{\pi_n})\prod_{k\in\tilde{\mathcal{N}}(\pi_n)}\mu_{k\to\pi_n}(x_{\pi_n}) \quad \text{(produit sur les voisins de } \pi_n \text{ dans } \tilde{T}) \\
&= \frac{1}{Z}\tilde{\psi}_{\pi_n}(x_{\pi_n})\prod_{k\in\mathcal{N}(\pi_n)\setminus\{n\}}\mu_{k\to\pi_n}(x_{\pi_n}) \\
&= \frac{1}{Z}\psi_{\pi_n}(x_{\pi_n})\mu_{n\to\pi_n}(x_{\pi_n})\prod_{k\in\mathcal{N}(\pi_n)\setminus\{n\}}\mu_{k\to\pi_n}(x_{\pi_n}) \\
&= \frac{1}{Z}\psi_{\pi_n}(x_{\pi_n})\prod_{k\in\mathcal{N}(\pi_n)}\mu_{k\to\pi_n}(x_{\pi_n})
\end{aligned}$$

Pour le cas $i=n$, on a:

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

Donc :

$$p(x_n, x_{\pi_n}) = \psi_{\pi_n}(x_{\pi_n})\alpha(x_{\pi_n})\psi_n(x_n)\psi_{n,\pi_n}(x_n, x_{\pi_n}) \tag{7.43}$$

Par conséquent:

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

Donc:

$$\alpha(x_{\pi_n}) = \frac{p(x_{\pi_n})}{\psi_{\pi_n}(x_{\pi_n})\mu_{n\to\pi_n}(x_{\pi_n})} \tag{7.44}$$

En utilisant (7.31), (7.32) et le résultat précédent, on en déduit:

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

En sommant par rapport à x_{π_n}, on obtient le résultat pour $p(x_n)$:

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

Proposition:

Soit $p \in \mathcal{L}(G)$, pour $G = (V, E)$ un arbre, alors on a:

$$p(x_1, \ldots, x_n) = \frac{1}{Z}\prod_{i\in V} \psi(x_i) \prod_{(i,j)\in E} \frac{p(x_i, x_j)}{p(x_i)p(x_j)} \tag{7.45}$$

Preuve: on le montre par récurrence. Le cas $n = 1$ est trivial. Ensuite, en supposant que n est une feuille, on peut écrire $p(x_1, \ldots, x_n) = p(x_1, \ldots, x_{n-1})\, p(x_n|x_{\pi_n})$. Mais multiplier par $p(x_n|x_{\pi_n}) = \frac{p(x_n, x_{\pi_n})}{p(x_n)p(x_{\pi_n})}\, p(x_n)$ correspond exactement à rajouter le potentiel d'arête pour (n, π_n) et le potentiel de noeud pour la feuille n. La formule est donc vérifiée par récurrence.

7.2 Chaines de Markov Cachées (HMM)

Dans le modèle de Markov Caché (HMM: Hidden Markov Model), nous notons z_0,z_1,...,z_T les états discrets correspondant aux variables latentes, et y_0, y_1,...,y_T les observations discrètes ou continues. Supposons:

1. z_0 suit une loi multinominale de paramètre π, $\pi \in \Delta_K$, $\forall k \in \{1, 2, \ldots, K\}$, on a: $\pi_k = P(z_{0k} = 1)$
2. $\{z_0, \ldots, z_T\}$ est une chaîne de Markov avec $P(z_{t,k} = 1|z_{t-1,k'} = 1) = A_{k,k'}$, $A = (A_{k,k'})_{1 \leqslant k,k' \leqslant K}$ matrice de transition.
3. On a:

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

Les tâches à accomplir sont:

- Le filtrage : $p(z_{t+1}|y_1, ..., y_t)$
- Le lissage : $p(z_t|y_1, ..., y_T)$
- Maximiser : $\max_{z_1, ...z_T} p(z_1, \ldots, z_T)$

On utilise l'algorithme de somme-produit. Tout d'abord on calcule les messages à faire passer. Comme on conditionne sur les variables $y_0, \ldots, y_T$ on devrait considérer les potentiels $\tilde{\psi}(z_t, y_t) = p(y_t|z_t)\delta(y_t, [y_t]_{\text{obs}})$ avec $[y_t]_{\text{obs}}$ la valeur de Y_t effectivement observée. En fait, par souci de simplicité, on identifiera y_t et $[y_t]_{\text{obs}}$ dans le reste du calcul de façons à ne pas devoir écrire toutes les fonctions de Dirac, en gardant à l'esprit que les variables y_t sont des valeurs fixées numériquement alors que z_t est une variable prenant toutes les valeurs dans $\{1, \ldots, K\}$ et qu'il faudra marginaliser.

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

Propriété:

Les messages arrivant en t vérifient:

$$\mu_{z_{t-1}\to z_t}(z_t)\mu_{y_t\to z_t}(z_t) = p(z_t, y_0, \ldots, y_t) \tag{7.46}$$

Cette propriété découle immédiatement par récurrence de la formule de calcul des messages.

Maintenant, on définit deux coefficients:

$$\begin{aligned} \text{Forward: } \alpha_t(z_t) &= \mu_{z_{t-1}\to z_t}(z_t)\mu_{y_t\to z_t}(z_t) = p(z_t, y_0, \ldots, y_t) \\ \text{Backward: } \beta_t(z_t) &= \mu_{z_{t+1}\to z_t}(z_t) \end{aligned}$$

Par récurrence, il est facile d'obtenir les résultats suivants, par la formule de récurrence des messages.

$$\begin{aligned} \alpha_t(z_t) &= p(y_t|z_t)\sum_{z_{t-1}} p(z_t|z_{t-1})\alpha_{t-1}(z_{t-1}) \\ \beta_t(z_t) &= \sum_{z_{t+1}} p(z_{t+1}|z_t)p(y_{t+1}|z_{t+1})\beta_{t+1}(z_{t+1}) \end{aligned}$$

Avec les initialisations:

$$\begin{aligned} \alpha_0(z_0) &= p(y_0|z_0)p(z_0) \\ \beta_T(z_T) &= 1 \end{aligned}$$

Proprieté 7.1

1. $p(z_t, y_0, \ldots, y_T) = \alpha_t(z_t)\beta_t(z_t)$
2. $p(y_0, \ldots, y_T) = \sum_{z_t} \alpha_t(z_t)\beta_t(z_t)$
3. $p(z_t|y_0, \ldots, y_T) = \frac{p(z_t, y_0, \ldots, y_T)}{p(y_0, \ldots, y_T)} = \frac{\alpha_t(z_t)\beta_t(z_t)}{\sum_{z_t}\alpha_t(z_t)\beta_t(z_t)}$
4. *pour* $t < T$, $p(z_t, z_{t+1}|y_0, \ldots, y_T) = \frac{1}{p(y_0, \ldots, y_T)}\, \alpha_t(z_t)\, \beta_{t+1}(z_{t+1})\, p(z_{t+1}|z_t)\, p(y_{t+1}|z_{t+1})$

Algorithme de EM

Avec les notations précédentes, on écrit la vraisemblance complète:

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

Quand on applique l'algorithme EM à l'estimation des paramètres d'un HMM on utilise l'inégalité de Jensen pour obtenir une borne inférieure de la vraisemblance

$$\log p(y_0,\dots,y_T) \geq \mathbb{E}_q[\log p(Z_0,\dots,Z_T,y_0,\dots,y_T)] = \mathbb{E}_q[l_c(\theta)]$$

A la $k$ième itération de l'algorithme, pour l'étape E, on calcule la borne inférieure pour

$$q(z_0,\dots,z_T) = \mathbb{P}(Z_0=z_0,\dots,Z_T=z_T|Y_0=y_0,\dots,Y_T=y_t;\theta^{k-1}) = \mathbb{P}(Z=z|Y=y;\theta^{k-1})$$

pour $(y_0,\dots,y_T) = ([y_0]_{\text{obs}},\dots,[y_T]_{\text{obs}})$ les valeurs observées.

- $\mathbb{E}[\delta(Z_0=i)|y] = \mathbb{P}(Z_0=i|y;\theta^{k-1})$
- $\mathbb{E}[\delta(Z_t=i)|y] = \mathbb{P}(Z_t=i|y;\theta^{k-1})$
- $\mathbb{E}[\delta(Z_{t+1}=i,Z_t=j|y;\theta^{k-1}] = \mathbb{P}(Z_{t+1}=i,Z_t=j|y;\theta^{k-1})$

Le calcul à l'étape E revient donc à remplacer les variables cachées $\delta(z_0=i)$,$\delta(z_t=i)$ et $\delta(z_{t+1}=i,z_t=j)$ par les quantités ci-dessus dans l'expression de la log-vraisemblance. Ensuite à l'étape M, de la k-ième itération, on maximise la nouvelle log-vraisemblance de la manière habituelle en θ, c'est à dire en π, A) et par rapport aux paramètres de la loi conditionelle de y_t sachant z_t, pour obtenir les estimateurs des paramètres θ^k.