INF8225 – Leçon 4
Modèles probabilistes, inférence, et apprentissage

Christopher Pal

École Polytechnique de Montréal

Au menu

- **Inférence : somme-produit vs max-produit**
- Le problème avec les cycles
- Les arbres d'intersections « junction trees» et d'autres algorithmes d'inférence
- D-séparation « dependancy separation »
- Apprentissage dans un réseau bayésien quand toutes les variables sont observées
- L'algorithme EM dans un réseau bayésien - quand il y des variables cachées

Data Mining

Practical Machine Learning Tools and Techniques

Slides for Chapter 9, Probabilistic methods

of *Data Mining* by I. H. Witten, E. Frank,
M. A. Hall, and C. J. Pal

Computing marginal probabilities

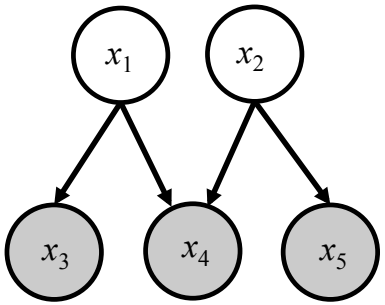
- The marginal for variable x_i is

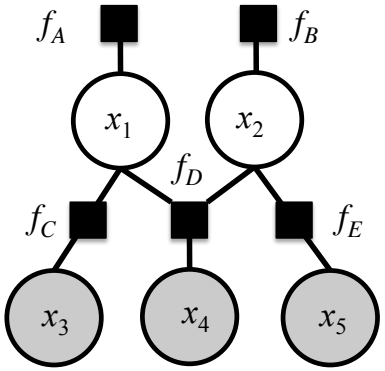
$$P(x_i) = \sum_{x_{j \neq i}} P(x_1, \ldots, x_n),$$

where the sum is over the states of all variables $x_j \neq x_i$

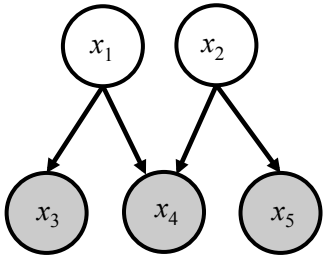
- Consider the task of computing the marginal *conditional* probability of variable x_3 given an observation for x_4 from a model where

$$P(x_1, \ldots, x_5) = P(x_1)P(x_2)$$
$$P(x_3 \mid x_1)P(x_4 \mid x_1, x_2)P(x_5 \mid x_2)$$





Marginal probabilities



- Since other variables in the graph have not been observed, they should be integrated out of the graphical model to obtain the desired result, ex.

$$P(x_3 \mid \tilde{x}_4) = \frac{P(x_3, \tilde{x}_4)}{P(\tilde{x}_4)} = \frac{P(x_3, \tilde{x}_4)}{\sum_{x_3} P(x_3, \tilde{x}_4)},$$ where the key quantity is

$$P(x_3, \tilde{x}_4) = \sum_{x_1} \sum_{x_2} \sum_{x_5} P(x_1, x_2, x_3, \tilde{x}_4, x_5)$$

$$= \sum_{x_1} \sum_{x_2} \sum_{x_5} P(x_1) P(x_2) P(x_3 \mid x_1) P(\tilde{x}_4 \mid x_1, x_2) P(x_5 \mid x_2).$$

- However, this sum involves a large data structure containing the joint probability, composed of the products over the individual probabilities

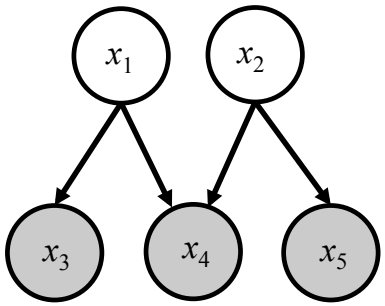
Intuition for the sum-product algorithm

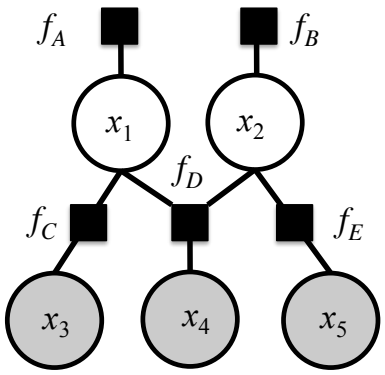
- The *sum-product algorithm* refers to a much better solution for computing marginals: simply *push the sums as far as possible to the right* before computing products of probabilities
- In our example the required marginalization can be computed by

$$P(x_3, \tilde{x}_4) = \sum_{x_1} P(x_3 \mid x_1) P(x_1) \sum_{x_2} P(\tilde{x}_4 \mid x_1, x_2) P(x_2) \sum_{x_5} P(x_5 \mid x_2)$$

$$= \sum_{x_1} P(x_3 \mid x_1) P(x_1) P(\tilde{x}_4 \mid x_1)$$

$$= \sum_{x_1} P(x_1, x_3, \tilde{x}_4).$$





The sum-product algorithm

- Computes exact marginals in tree structured factor graphs
- Begin with variable or function nodes that have only one connection (leaf nodes)
- Function nodes send the message: $\mu_{f \to x}(x) = f(x)$ to the variable connected to them
- Variable nodes send the message: $\mu_{x \to f}(x) = 1$
- Other nodes wait until they have received a message from all neighbors except the one will send a message to

Function to variable messages

- When ready, function nodes send messages of the following form to variable *x*:

$$\mu_{f\to x}(x) = \sum_{x_1,\ldots x_K} f(x, x_1, \ldots, x_K) \prod_{k\in N(f)\setminus x} \mu_{x_k\to f}(x_k),$$

- where $N(f)\setminus x$ represents the set of the function node *f*'s neighbors, excluding the recipient variable *x*
- Variables of the *K* other neighboring nodes are $x_1,\ldots,x_K$
- If a variable is observed, messages for functions involving it no longer need a sum over states of the variable, the function is evaluated with the observation
- One could think of the associated variable node as being transformed into the new modified function

Variable to function messages

- Variable nodes send messages to functions of form:

$$\mu_{x\to f}(x) = \mu_{f_1\to x}(x)\ldots\mu_{f_K\to x}(x) = \prod_{k\in N(x)\setminus f}\mu_{f_k\to x}(x),$$

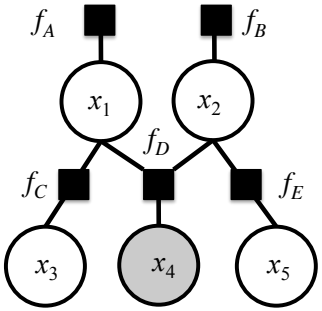
- where the product is over the (*K*) messages from all neighboring functions *N*(*x*) other than the recipient function *f, i.e.* $f_k \in N(x)\setminus f$
- The marginal for each node is obtained from the product over all *K*+1 incoming messages from all functions connected to a variable

$$P(x_i) = \mu_{f_1\to x}(x)\ldots\mu_{f_K\to x}(x)\mu_{f_{K+1}\to x}(x) = \prod_{k=1}^{K+1}\mu_{f_k\to x}(x)$$

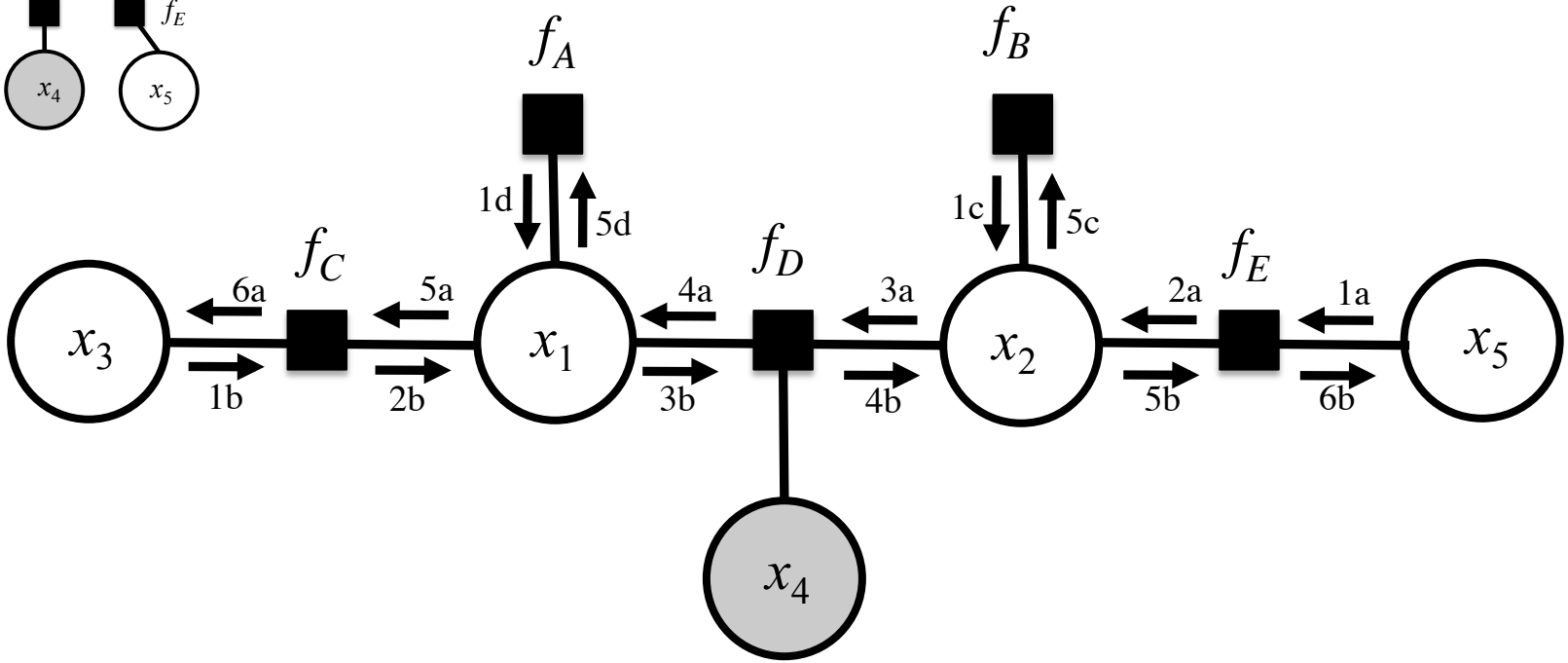
Numerical stability

- Multiplying many probabilities quickly leads to very small numbers
- The sum-product algorithm is often implemented with re-scaling
- Alternatively or additionally the computations can be performed in log space leading to computations of the form $c = \log(\exp(a) + \exp(b))$
- To help prevent loss of precision when computing the exponents, use the equivalent expression with the smaller exponent below

$$c = \log(e^a + e^b) = a + \log(1 + e^{b-a}) = b + \log(1 + e^{a-b})$$



Sum product messages



$$P(x_3, \tilde{x}_4) = \sum_{x_1} P(x_3 \mid x_1) \underbrace{P(x_1)}_{1d} \sum_{x_2} P(\tilde{x}_4 \mid x_1, x_2) \underbrace{P(x_2)}_{1c} \sum_{x_5} P(x_5 \mid x_2) \cdot \underbrace{1}_{1a}$$

$$\underbrace{\qquad\qquad}_{2a}$$

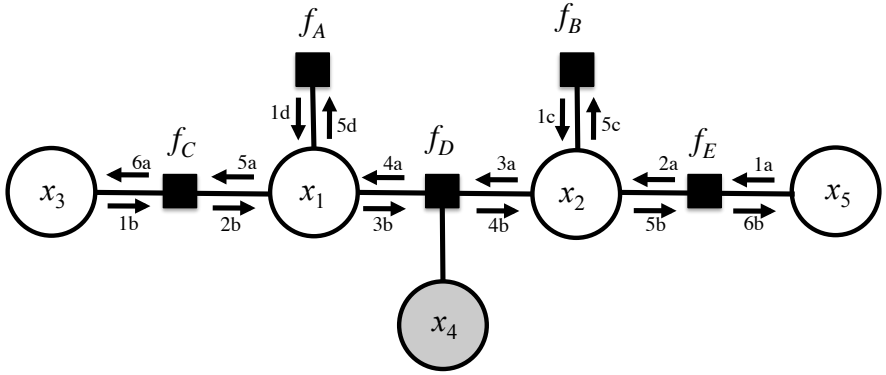
$$\underbrace{\qquad\qquad\qquad\qquad}_{3a}$$

$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad}_{4a}$$

$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{5a}$$

$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{6a}$$

The messages for our example



$$P(x_3,\tilde{x}_4)=\sum_{x_1}P(x_3\mid x_1)\underbrace{P(x_1)}_{1d}\sum_{x_2}P(\tilde{x}_4\mid x_1,x_2)\underbrace{P(x_2)}_{1c}\sum_{x_5}P(x_5\mid x_2)\cdot\underbrace{1}_{1a}$$

$2a$, $3a$, $4a$, $5a$, $6a$

$$1\text{a}:\mu_{x_5\to f_E}(x_5)=1,\ 1\text{c}:\mu_{f_B\to x_2}(x_2)=f_B(x_2),\ \ 1\text{d}:\mu_{f_A\to x_1}(x_1)=f_A(x_1)$$

$$2\text{a}:\mu_{f_E\to x_2}(x_2)=\sum_{x_5}f_E(x_5,x_2)$$

$$3\text{a}:\mu_{x_2\to f_D}(x_5)=\mu_{f_B\to x_2}(x_2)\mu_{f_E\to x_2}(x_2)$$

$$4\text{a}:\mu_{f_D\to x_1}(x_1)=\sum_{x_2}f_D(\tilde{x}_4\mid x_1,x_2)\mu_{x_2\to f_D}(x_5)$$

$$5\text{a}:\mu_{x_1\to f_C}(x_1)=\mu_{f_A\to x_1}(x_1)\mu_{f_D\to x_1}(x_1)$$

$$6\text{a}:\mu_{f_C\to x_3}(x_3)=\sum_{x_1}f_C(x_3,x_1)\mu_{x_1\to f_C}(x_1)$$

The complete algorithm can yield all single-variable marginals in the graph using the other messages shown in the diagram

Finding the most probable configuration

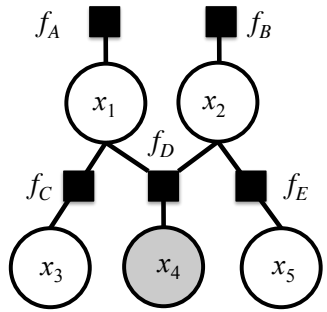
- Finding the most probable configuration of all other variables in our example given $x_4 = \tilde{x}_4$ involves searching for

$$\left\{x_1^*, x_2^*, x_3^*, x_5^*\right\} = \underset{x_1, x_2, x_3, x_5}{\operatorname{argmax}} P(x_1, x_2, x_3, x_5 \mid \tilde{x}_4),$$

for which

$$P(x_1^*, x_2^*, x_3^*, x_5^* \mid \tilde{x}_4) = \max_{x_1, x_2, x_3, x_5} P(x_1, x_2, x_3, x_5 \mid \tilde{x}_4).$$

Pushing max to the right



- Because max behaves in a similar way to sum, like in the sum-product algorithm we can push the max operations as far to the right as possible, noting that $\max(ab, ac) = a \max(b, c)$
- For our example we have

$$\max_{x_1} \max_{x_2} \max_{x_3} \max_{x_5} P(x_1, x_2, x_3, \tilde{x}_4, x_5)$$

$$= \max_{x_3} \max_{x_1} P(x_1) P(x_3 \mid x_1) \max_{x_2} P(x_2) P(\tilde{x}_4 \mid x_1, x_2) \max_{x_5} P(x_5 \mid x_2).$$

The max-sum algorithm

- A log-space version of max-product
- As in the sum-product algorithm, variables or factors that have only one connection in the graph begin by sending either :
  - a function-to-variable message $\mu_{x \to f}(x) = 0$
  - or a variable-to-function message $\mu_{f \to x}(x) = \log f(x)$
- Each function and variable node in the graph waits until it has received a message from all neighbors other than the node that will receive its message

Function to variable messages

- Each function and variable node in the graph waits until it has received a message from all neighbors other than the node that will receive its message
- Function nodes send messages of the following form to variable *x*

$$\mu_{f\to x}(x) = \max_{x_1,\ldots,x_K}\left[\log f(x,x_1,\ldots,x_K) + \sum_{k\in N(f)\setminus x}\mu_{x_k\to f}(x_k)\right],$$

where the notation $N(f)\setminus x$ is the same as for the sum-product algorithm above

Variable to function messages

- Variables send messages to functions of this form

$$\mu_{x \to f}(x) = \sum_{k \in N(x) \setminus f} \mu_{f_k \to x}(x),$$

  where the sum is over the messages from all functions other than the recipient function.

- When the algorithm terminates, the probability of the most probable configuration (MPC) can be extracted from any node using

$$p^* = \max_x \left[\sum_{k \in N(x)} \mu_{f_k \to x}(x) \right],$$

the MPC itself is:

$$x^* = \arg\max_x \left[\sum_{k \in N(x)} \mu_{f_k \to x}(x) \right].$$

Au menu

- Révision: concepts de base
- **Inférence : somme-produit vs max-produit**
- Le problème avec les cycles
- Les arbres d'intersections « junction trees» et d'autres algorithmes d'inférence
- D-séparation « dependancy separation »
- Apprentissage dans un réseau bayésien quand toutes les variables sont observées
- L'algorithme EM dans un réseau bayésien - quand il y des variables cachées

Somme-produit vs. max-produit

- Question : Quelles quantités sont produites par l'algorithme de somme-produit dans un modèle probabiliste?
- Considérez une fonction $f(x_1, x_2, x_3) = \psi_1(x_1, x_2)\psi_2(x_1, x_3)$
- On peut utiliser la propriété de distributivité de la somme et le maximum à obtenir…

Somme-produit vs. max-produit

- Question : Quelles quantités sont produites par l'algorithme de somme-produit dans un modèle probabiliste?
- Considérez une fonction $f(x_1,x_2,x_3) = \psi_1(x_1,x_2)\psi_2(x_1,x_3)$
- On peut utiliser la propriété de distributivité de la somme et le maximum à obtenir

$$\sum_{x_1,x_2,x_3} f(x_1,x_2,x_3) = \sum_{x_1}\left[\left(\sum_{x_2}\psi_1(x_1,x_2)\right)\left(\sum_{x_3}\psi_2(x_1,x_3)\right)\right],$$

$$\max_{x_1,x_2,x_3} f(x_1,x_2,x_3) = \max_{x_1}\left[\left(\max_{x_2}\psi_1(x_1,x_2)\right)\left(\max_{x_3}\psi_2(x_1,x_3)\right)\right].$$

Somme-produit vs. max-produit

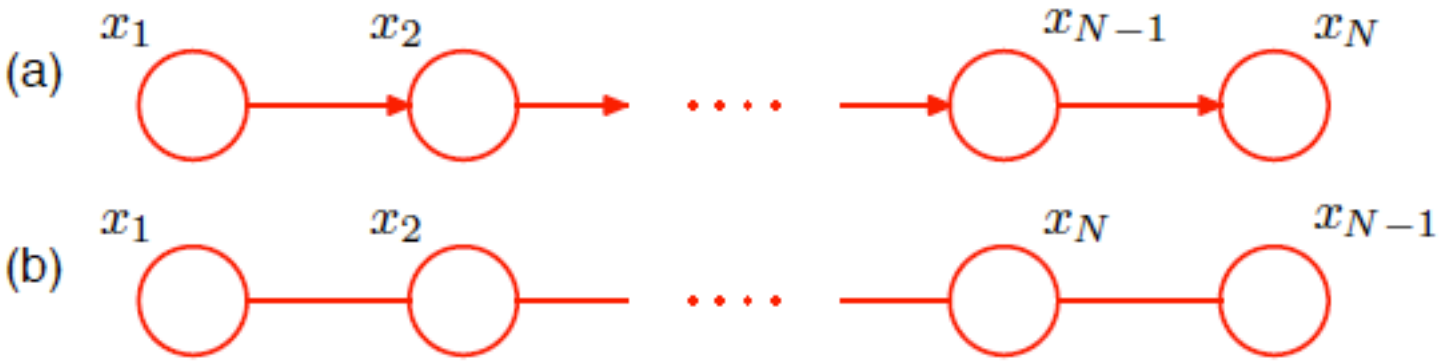
L'obtention des quantités importantes

- Avec somme-produit nous avons un stratégie efficace pour marginaliser les distributions
- Avec max-produit nous avons un stratégie efficace pour obtenir les configurations de maximum probabilité

Exemple concrète

- Considérez les deux modèles suivants

$$p(\mathbf{x}) = p(x_1)p(x_2|x_1)p(x_3|x_2)\cdots p(x_N|x_{N-1}).$$



$$p(\mathbf{x}) = \frac{1}{Z}\psi_{1,2}(x_1, x_2)\psi_{2,3}(x_2, x_3)\cdots\psi_{N-1,N}(x_{N-1}, x_N).$$

- Voici un graphe non-orienté encodant les mêmes relations d'indépendance

Exemple de Bishop

Les champs aléatoires de Markov « Markov Random Fields MRFs »

- Étant donné des cliques C des variables x_c, on peut représenter une distribution conjointe comme un produit des *fonctions potentielles* $\psi_C(C)$ sur des cliques maximales du graphe

$$p(\mathbf{x}) = \frac{1}{Z} \prod_C \psi_C(\mathbf{x}_C) \qquad Z = \sum_{\mathbf{x}} \prod_C \psi_C(\mathbf{x}_C)$$

- Où Z est appelé la « fonction de partition » ou la constante de normalisation
- Il est possible de modéliser certaines relations d'indépendance conditionnelle qui sont impossibles de modéliser avec des réseaux bayésiens

L'algorithme max-produit

- Donc, étant donné un but de trouver

$$\mathbf{x}^{\max} = \arg\max_{\mathbf{x}} p(\mathbf{x}) \quad \text{ou} \quad p(\mathbf{x}^{\max}) = \max_{\mathbf{x}} p(\mathbf{x}).$$

- En utilisant le fait que $\max(ab, ac) = a \max(b, c)$
- On peut écrire $\max_{\mathbf{x}} p(\mathbf{x}) = \max_{x_1} \dots \max_{x_M} p(\mathbf{x})$

  dans notre chaine comme:

$$\max_{\mathbf{x}} p(\mathbf{x}) = \frac{1}{Z} \max_{x_1} \cdots \max_{x_N} [\psi_{1,2}(x_1, x_2) \cdots \psi_{N-1,N}(x_{N-1}, x_N)]$$

$$= \frac{1}{Z} \max_{x_1} \left[\psi_{1,2}(x_1, x_2) \left[\cdots \max_{x_N} \psi_{N-1,N}(x_{N-1}, x_N) \right] \right].$$

Stabilité numérique

- Toutefois, en pratique les produits de plusieurs petites probabilités causent des problèmes avec soupassement « underflow »
- Le logarithme est une fonction monotone, donc si a > b, ln a > ln b, donc on peut échanger ln et max. comme

$$\ln\left(\max_{\mathbf{x}} p(\mathbf{x})\right) = \max_{\mathbf{x}} \ln p(\mathbf{x}).$$

- De plus, la propriété de distributivité est préservée, donc $\max(a+b, a+c) = a + \max(b, c).$

L'algorithme « max-sum »

- Nous avons les messages suivants pour

a) Messages entre les fonctions et variables et

$$\mu_{f\to x}(x) = \max_{x_1,\ldots,x_M}\left[\ln f(x,x_1,\ldots,x_M) + \sum_{m\in \mathrm{ne}(f_s)\setminus x} \mu_{x_m\to f}(x_m)\right]$$

b) Messages entre les variables et fonctions

$$\mu_{x\to f}(x) = \sum_{l\in \mathrm{ne}(x)\setminus f} \mu_{f_l\to x}(x).$$

- Avec les messages initiaux dans les feuilles de:

$$\mu_{x\to f}(x) = 0$$
$$\mu_{f\to x}(x) = \ln f(x)$$

L’algorithme « max-sum »

- À la fin, nous avons les calculs suivants pour obtenir la probabilité maximale et les configurations maximales :

$$p^{\max} = \max_{x} \left[\sum_{s \in \mathrm{ne}(x)} \mu_{f_s \to x}(x) \right].$$

$$x^{\max} = \arg\max_{x} \left[\sum_{s \in \mathrm{ne}(x)} \mu_{f_s \to x}(x) \right].$$

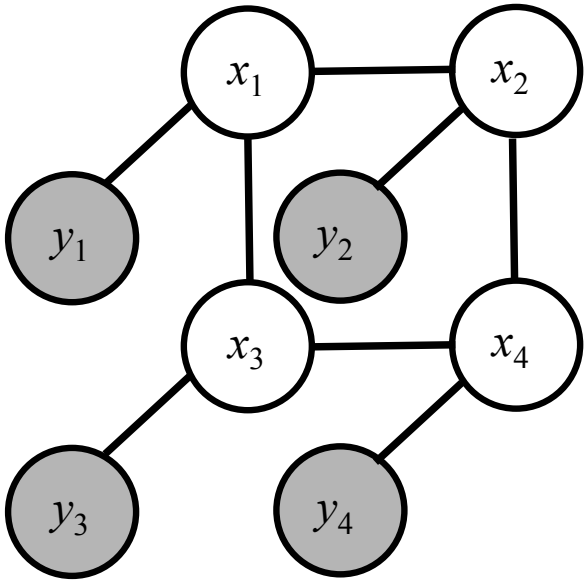
MRFs are often used for images, consider this example:

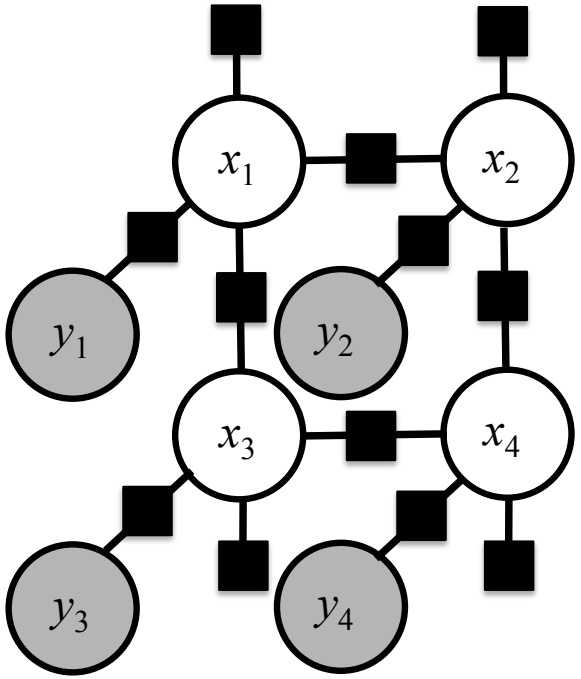
$$P(x_1, x_2, x_3, x_4) = \frac{1}{Z} \prod_{u=1}^{U} \phi_u(X_u) \prod_{v=1}^{V} \Psi_v(X_v)$$

$$= \frac{1}{Z} f_A(x_1) f_B(x_2) f_C(x_1) f_D(x_2) f_E(x_1, x_2) f_F(x_2, x_3) f_G(x_3, x_4) f_H(x_4, x_1)$$

- Note we used unary and pairwise potentials

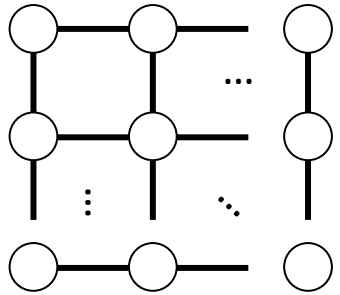
An MRF using a traditional MRF style undirected graph





An MRF as a factor graph

MRFs and energy functions



- An MRF lattice is often repeated over an image
- MRFs can be expressed in terms of an energy function $F(X)$, where

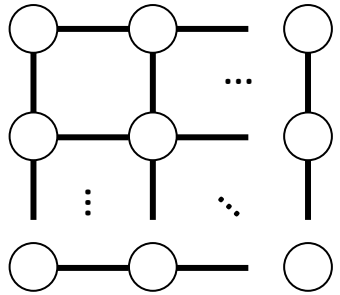
$$F(X) = \sum_{u=1}^{U} U(X_u) + \sum_{v=1}^{V} V(X_v), \qquad \text{and}$$

$$P(X) = \frac{1}{Z}\exp(-F(X)) = \frac{1}{Z}\exp\left(-\sum_{u=1}^{U} U(X_u) - \sum_{v=1}^{V} V(X_v)\right).$$

- Since Z is constant for any assignment of the variables X we have

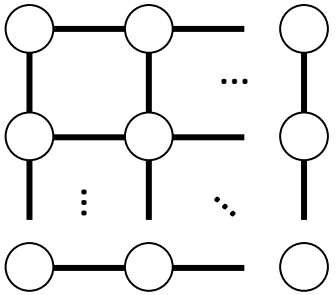
$$-\log P(x_1, x_2, x_3, x_4) \propto \sum_{u=1}^{U} U(X_u) + \sum_{v=1}^{V} V(X_v)$$

Minimizing MRF energies



- A commonly used strategy for tasks such as image segmentation and entity resolution in text documents is to minimizing an energy function of the form in the previous slide
- When such energy functions are “sub-modular”, an exact minimum can be found using algorithms based on graph-cuts; otherwise methods such as tree-reweighted message passing can be used.

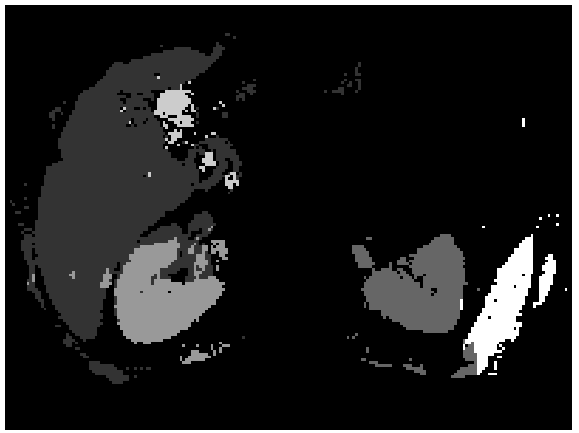
Example: Image Segmentation



- Given a labeled dataset of medical imagery, such as a computed tomography or CT image
- A well known approach is to learn a Markov Random Field model to segment that image into classes of interest, ex. anatomical structures or tumors
- Image features are combined with spatial context

Original Image

Ground truth



MRF segmentation

From Bhole et al. (2013)

Au menu

- Révision: concepts de base
- Inférence : somme-produit vs max-produit
- **Le problème avec les cycles**
- Les arbres d'intersections « junction trees» et d'autres algorithmes d'inférence
- D-séparation « dependancy separation »
- Apprentissage dans un réseau bayésien quand toutes les variables sont observées
- L'algorithme EM dans un réseau bayésien - quand il y des variables cachées

Quel est le problème avec les cycles?

- Exercice: Créer un réseau bayésien pour les relations en logique suivantes :

  Si $A=a_1$, alors $B = b_1$ et $C = c_1$

  Si $A=a_2$, alors $B = b_2$ et $C = c_1$

  Si $A=a_3$, alors $B = b_1$ et $C = c_2$

  Si $B=b_1$ et $C = c_1$, alors $D = d_1$

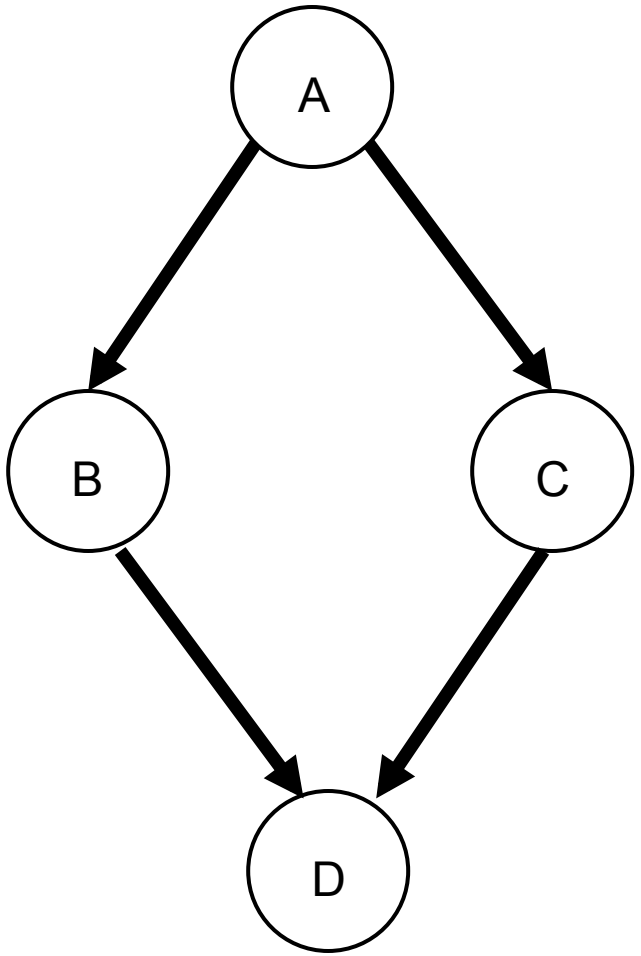
  Si $B=b_1$ et $C = c_2$, alors $D = d_2$

  Si $B=b_2$ et $C = c_1$, alors $D = d_2$

  Si $B=b_2$ et $C = c_2$, alors $D = d_1$

- Avec l'observation : $D = d_1$

Quel est le problème avec les cycles?



- En utilisant une stratégie avec des messages seulement sur P(B) et P(C), mais pas sur P(B,C) nous allons avoir un problème si notre but est d'inférer (correctement) que $A=a_1$

Quel est le problème avec les cycles?

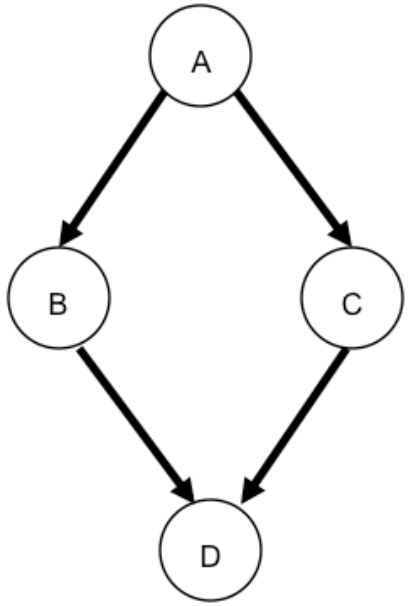
Si $A=a_1$, alors $B = b_1$ et $C = c_1$

Si $A=a_2$, alors $B = b_2$ et $C = c_1$

Si $A=a_3$, alors $B = b_1$ et $C = c_2$

Si $B=b_1$ et $C = c_1$, alors $D = d_1$

Si $B=b_2$ et $C = c_2$, alors $D = d_1$



- Avec l'observation : $D = d_1$
  En utilisant une stratégie où l'information sur b et c peuvent devenir découplés, nous allons avoir un problème.

En pratique il y a une autre représentation utilisé pour obtenir des calculs exacte

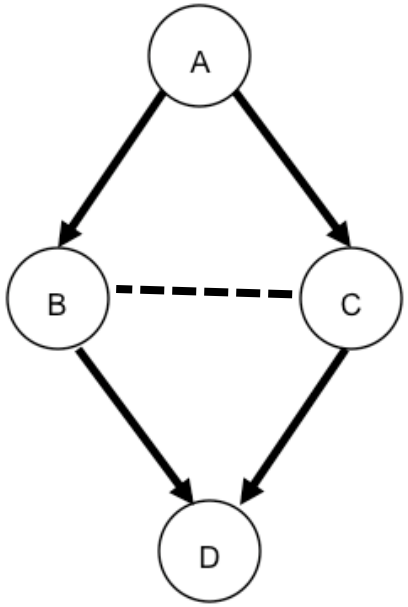
- On peut transformer notre réseau de Bayes à la représentation l'ensemble - chaîne « set-chain representation »
- Il consiste de toutes les variables X_G dans un graphe et il utilise le concept de cliques X_c et séparateurs X_s

$$P(X_G) = \frac{\prod_{c \in C} P(X_c)}{\prod_{s \in S} P(X_s)} \qquad P(X_G = x_g) = \frac{\prod_{c \in C} \phi_c(X_c = x_c)}{\prod_{s \in S} \varphi_s(X_s = x_s)}$$

- La deuxième équation est appelée une représentation potentielle « potential representation »

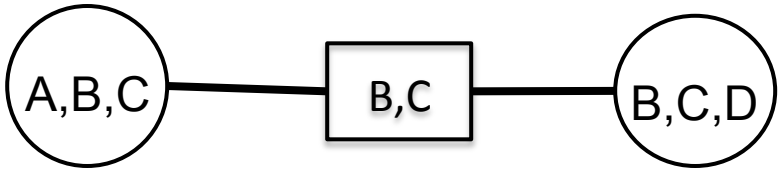
Exemple Simple

- Etant donné une observation D=d
- Calculer P(A|D=d) exactement!



$$P(X_G) = \frac{\prod_{c \in C} P(X_c)}{\prod_{s \in S} P(X_s)}$$

$$= \frac{P(A,B,C)P(B,C,D)}{P(B,C)}$$



- Étape 1: Transformer le réseau à un arbre consistant de clusters et les intersections, « a junction tree »
- Étape 2: Appliquer un autre algorithme utilisant le passage de messages (voir Huang et Darwiche, 1995)

Exact inference in general Bayes Nets

(See Huang and Darwiche, 1995 for more details)

1. Construct an undirected graph, called a moral graph, from the DAG (the Bayes Net).
2. Selectively add arcs to the moral graph to form a triangulated graph.
3. From the triangulated graph, identify select subsets of nodes, called cliques.
4. Build a join tree (junction tree), starting with the cliques as clusters: connect the clusters to form an undirected tree satisfying the join tree property, inserting the appropriate sepsets.
5. *Apply another alternative message passing algorithm in this graph (see the paper for details)*

Pour construire un arbre de jonction

1. Construire un graphe non orienté, appelé un graphe moral de la DAG -- effacer les orientations et ajouter des arêtes entre des parents
2. Ajouter sélectivement des arcs au graphe moral pour former un graphe triangulé
   – c.-à-d. chaque cycle de la longueur de quatre ou plus contiens une arête entre deux noeuds non adjacents dans le cycle

Exemple : la construction d'un graphe moral

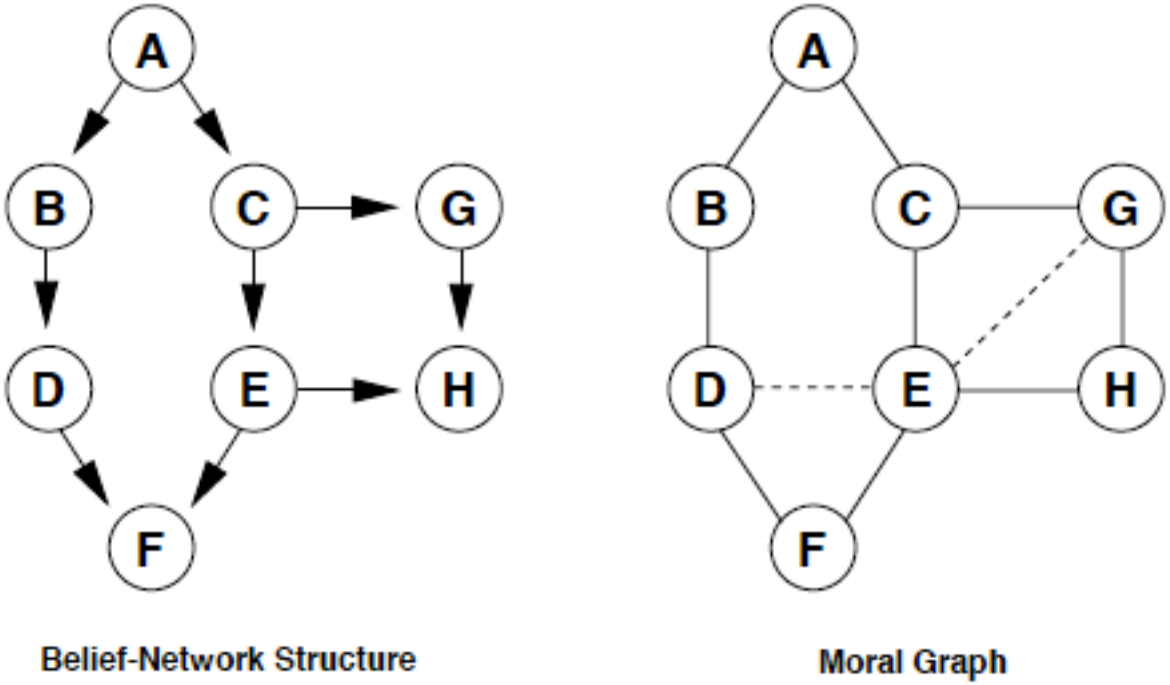


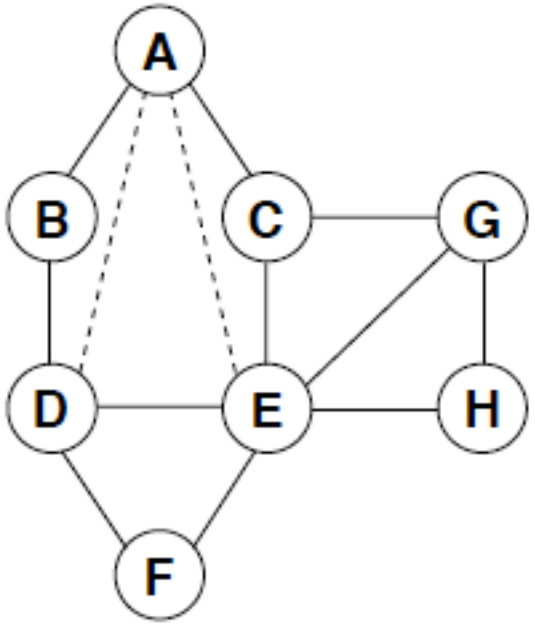
Figure 1. Constructing the moral graph.

Huang et Darwiche, 1995

La création d'un arbre de jonction

3. Dans le graphe triangulé, identifier des sous-ensembles des nœuds appelés des cliques -- c.-à-d. des sous-ensembles où il existe une arête entre chaque nœud dans l'ensemble.
4. Construire un arbre d'intersections « junction tree», en commençant par les cliques comme clusters, puis joindre les clusters pour former un arbre non orienté satisfaisant la propriété des arbres d'intersections en ajoutant des séparateurs «sepsets» appropriés (variables en commun)

La triangulation d'un graphe moral



Triangulated Graph

Eliminated Vertex	Induced Cluster	Edges Added
H	EGH	none
G	CEG	none
F	DEF	none
C	ACE	(A,E)
B	ABD	(A,D)
D	ADE	none
E	AE	none
A	A	none

Elimination Ordering

Figure 2. Triangulating the moral graph.

Article de Huang et Darwiche, 1994 p. 15

Building an Optimal Join Tree

1. Begin with a set of n trees, each consisting of a single clique, and an empty set $\mathcal{S}$.

2. For each distinct pair of cliques $\mathbf{X}$ and $\mathbf{Y}$:[9]

   (a) Create a candidate sepset, labeled $\mathbf{X} \cap \mathbf{Y}$, with backpointers to the cliques $\mathbf{X}$ and $\mathbf{Y}$. Refer to this sepset as $\mathbf{S_{XY}}$.
   (b) Insert $\mathbf{S_{XY}}$ into $\mathcal{S}$.

3. Repeat until $n - 1$ sepsets have been inserted into the forest.

   (a) Select a sepset $\mathbf{S_{XY}}$ from $\mathcal{S}$, according to the criterion specified in Section 4.4.2. Delete $\mathbf{S_{XY}}$ from $\mathcal{S}$.
   (b) Insert the sepset $\mathbf{S_{XY}}$ between the cliques $\mathbf{X}$ and $\mathbf{Y}$ *only if* $\mathbf{X}$ and $\mathbf{Y}$ are on different trees in the forest.[10] (Note that the insertion of such a sepset would merge two trees into a larger tree.)

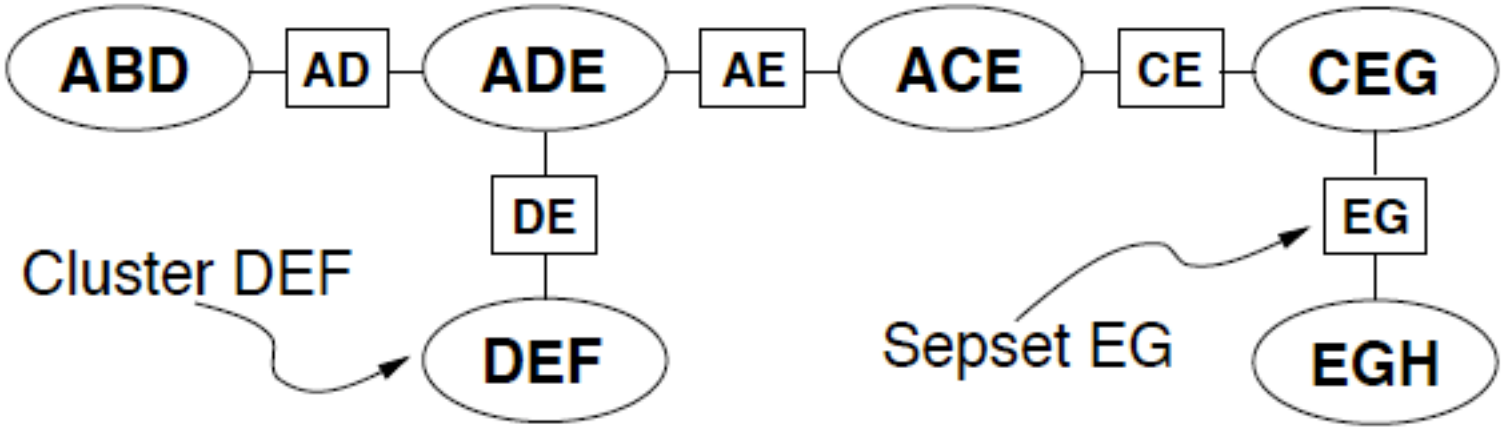
Choisir les sepsets appropriés (i.e. section 4.4.2)

- Rappel : “sepset” = le sous-ensemble de variables commun dans deux cliques ou plus
  - Afin de décrire comment choisir le prochain candidat sepset, nous définissons les notions de masse, et de coût, comme suit :
  - La masse d'un sepset, S_{XY} est le nombre de variables qu'il contient, ou le nombre de variables dans X \ Y.
  - Le coût d'un sepset S_{XY} est le poids de X, plus le poids de Y, où le poids est défini comme suit:
    * Le poids d'une variable V est le nombre de valeurs de V.
    * Le poids d'un ensemble de variables X est le produit des poids des variables dans X.

Choisir les sepsets appropriés (i.e. section 4.4.2)

- Avec ces notions établies, nous pouvons maintenant choisir le prochain candidat sepset de S :
- Pour l'arbre de cliques résultant de satisfaire la propriété d'un arbre de jonction, nous devons choisir le candidat avec le sepset de la plus grande masse.
- Lorsqu'il existe deux ou plusieurs sepsets de masse égale, nous pouvons optimiser le temps de l'inférence sur l'arbre résultant comme suit : choisir le candidat sepset au moindre coût.
- La base de cette méthode de construction d'un arbre de jonction optimal peut être trouvée dans un article de recherche de Jensen, F. V., and Jensen, F., "Optimal junction trees", UAI 1994.

Avec notre exemple nous avons



ϕ_{ABD} =

a	b	d	ϕ_{ABD}(abd)
on	on	on	.225
on	on	off	.025
on	off	on	.125
on	off	off	.125
off	on	on	.180
off	on	off	.020
off	off	on	.150
off	off	off	.150

ϕ_{AD} =

a	d	ϕ_{AD}(ad)
on	on	.35
on	off	.15
off	on	.33
off	off	.17

etc.

Data Mining

Practical Machine Learning Tools and Techniques

Slides for Chapter 9, Probabilistic methods

of *Data Mining* by I. H. Witten, E. Frank,
M. A. Hall, and C. J. Pal

Probabilistic inference

- With complex probability models – and even with some seemingly simple ones – computing quantities such as posterior distributions, marginal distributions, and the maximum probability configuration, often require specialized methods to achieve results efficiently – even approximate ones.
- This is the field of *probabilistic inference*

Sampling

- Sampling methods are popular in both statistics and machine learning
- Useful for implementing fully Bayesian methods that use distributions on parameters, or for inference in graphical models with cyclic structures
- *Gibbs sampling* is a popular special case of the more general Metropolis-Hastings algorithm
- Allows one to generate samples from a joint distribution even when the true distribution is a complex continuous function

Gibbs sampling

- Gibbs sampling is conceptually very simple.
- Assign an initial set of states to the random variables of interest.
- With *n* random variables, the initial assignments or samples can be written as $x_1 = x_1^{(0)}, \ldots, x_n = x_n^{(0)}$
- Then iteratively cycling through updates to each variable by *sampling* from its conditional distribution given the others:

$$x_1^{(i+1)} \sim p(x_1 \mid x_2 = x_2^{(i)}, \ldots, x_n = x_n^{(i)}),$$
$$\vdots$$
$$x_n^{(i+1)} \sim p(x_n \mid x_1 = x_1^{i}, \ldots, x_{n-1} = x_{n-1}^{i}).$$

Simulated annealing

- A procedure that seeks an approximate most probable explanation (MPE) or configuration
- We can obtain it by adapting Gibbs sampling to include an iteration-dependent "temperature" term t_i, that decreases (usually slowly) over time
- Starting with an initial assignment, $x_1 = x_1^{(0)}, \ldots, x_n = x_n^{(0)}$ subsequent samples take the form

$$x_1^{(i+1)} \sim p(x_1 \mid x_2 = x_2^{(i)}, \ldots, x_n = x_n^{(i)})^{\frac{1}{t_i}},$$
$$\vdots$$
$$x_n^{(i+1)} \sim p(x_n \mid x_1 = x_1^{i}, \ldots, x_{n-1} = x_{n-1}^{i})^{\frac{1}{t_i}},$$

Iterated conditional modes

- Consists of iterations of the form

$$x_1^{(i+1)} \sim \underset{x_1}{\operatorname{argmax}}\, p(x_1 \mid x_2 = x_2^{(i)}, \ldots, x_n = x_n^{(i)}),$$

$$\ldots$$

$$x_n^{(i+1)} \sim \underset{x_n}{\operatorname{argmax}}\, p(x_n \mid x_1 = x_1^{i}, \ldots, x_{n-1} = x_{n-1}^{i}).$$

- Often converges quickly to an approximate MPE, but is prone to local minima
- Zero temperature limit of a simulated annealer
- Gibbs sampling is a popular special case of the more general Metropolis Hastings algorithm
- Gibbs sampling and simulated annealing are special cases of the broader class of Markov Chain Monte Carlo (MCMC) methods

Variational methods

- Rather than sampling from a complex distribution the distribution can be approximated by a simpler, more tractable, function
- Suppose we have a probability model with a set of hidden variables *H* and observed variables *X*
- Let $p = p(H \mid \tilde{X}; \theta)$ be the exact posterior
- Define $q = q(H \mid \tilde{X}; \Phi)$ as the variational approx. having its own variational parameters, Φ
- Variational methods make q close to p, yielding principled EM algorithms with approx. posteriors

INF8225

Apprentissage automatique
avec des modèles probabilistes

Au menu

- Révision: concepts de base
- Inférence : somme-produit vs max-produit
- Le problème avec les cycles
- Les arbres d'intersections « junction trees» et d'autres algorithmes d'inférence
- D-séparation « dependancy separation »
- **Apprentissage** dans un réseau bayésien quand toutes les variables sont observées
- L'algorithme EM dans un réseau bayésien - quand il y des variables cachées

Apprentissage par MV pour des réseaux bayésiennes discrètes

1. Avec des données complètes, le problème d'apprentissage de paramètres par maximum de vraisemblance pour un réseau bayésien se décompose en problèmes séparés, un pour chaque paramètre
2. Les valeurs des paramètres pour une variable, étant donné ses parents, sont simplement les fréquences observées des valeurs de la variable pour chaque instanciation des valeurs des parents

À lire (Textbook)

- **Maximum likelihood estimation in a Bayesian Network**

Data Mining

Practical Machine Learning Tools and Techniques

Slides for Chapter 9, Probabilistic methods

of *Data Mining* by I. H. Witten, E. Frank,
M. A. Hall, and C. J. Pal

Estimating Bayesian network parameters

- The log-likelihood of a Bayesian network with *V* variables and *N* examples of complete variable assignments to the network is

$$\sum_{i=1}^{N} \log P(\{\tilde{A}_1, \tilde{A}_2, ..., \tilde{A}_V\}_i) = \sum_{i-1}^{N} \sum_{v=1}^{V} \log P(\tilde{A}_{v,i} \left| \text{Parents}(\tilde{A}_{v,i}); \Theta_v \right.)$$

  where the parameters of each conditional or unconditional distribution are given by Θ_v

- We use the $\tilde{A}_{v,i}$ notation to indicate the *i*th observation of variable *v*

Estimating probabilities in Bayesian networks

- The estimation problem *decouples* into separate estimation problems for each conditional or unconditional probability
- Unconditional probabilities can be written as

$$P(A = a) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}(\tilde{A}_i = a)$$

where $\mathbf{1}(\tilde{A}_i = a)$ is an indicator function returning 1 when the i^{th} observed value for A_i=a and 0 otherwise

Estimating conditional distributions

- Estimating conditional distributions in Bayesian networks is equally easy and amounts to simply counting configurations and dividing, ex.

$$P(B = b \mid A = a) = \frac{P(B = b, A = a)}{P(A = a)} = \frac{\sum_{i=1}^{N} \mathbf{1}(\tilde{A}_i = a, \tilde{B}_i = b)}{\sum_{i=1}^{N} \mathbf{1}(\tilde{A}_i = a)}.$$

- Zero counts cause problems and this motivates the use of Bayesian priors

Estimating network structure

- One possibility is to use cross-validation to estimate the goodness of fit on held out data (so as to avoid over fitting) another is to penalize model complexity
- Let *K* be the number of parameters, *LL* the log-likelihood, and *N* the number of instances in the data.
- Two popular measures for evaluating the quality of a network are the *Akaike Information Criterion* (AIC):

$$\text{AIC score} = -LL + K$$

and the following *MDL metric* based on the MDL principle:

$$\text{MDL score} = -LL + \frac{K}{2}\log N$$

- In both cases the log-likelihood is negated, so the aim is to minimize these scores.
- More Bayesian approach: use prior over model structures

Apprentissage (simple)

dans un réseau bayésien

(Quand chaque variable est observée)

Comment construire un réseau bayésien

Situations typiques :

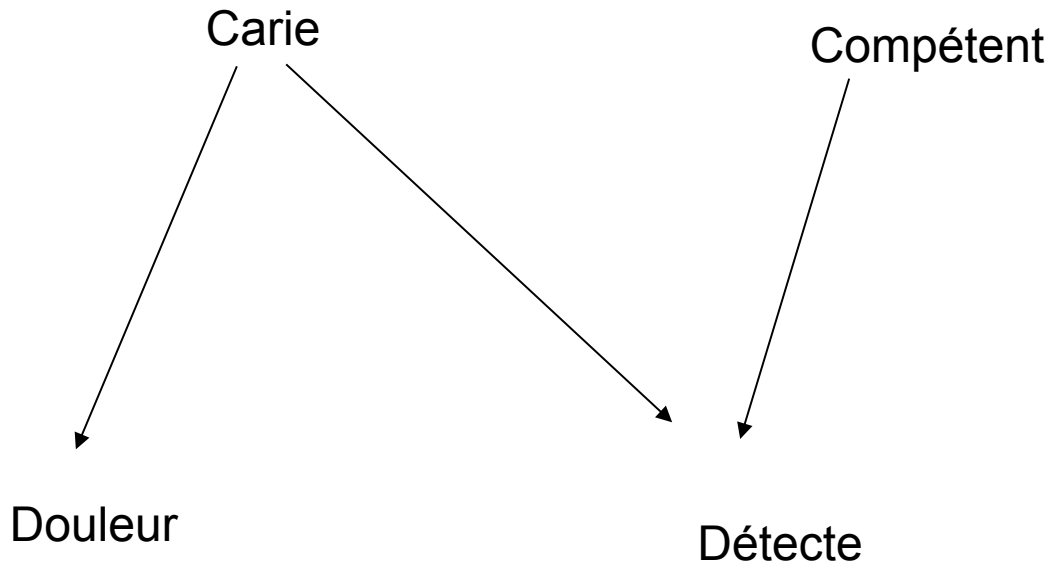
- Par les relations de causes à effet, nous savons construire le réseau: il ne manque alors que les tables de probabilité conditionnelle
  - Cas 1: Nous avons des tables de donnée complètes
  - Cas 2: Certaines valeurs sont manquantes ou cachées
- Nous ne savons pas quelle topologie de réseau est la meilleure, selon les données dont nous disposons

Premier cas: le réseau est connu

- Si nous sommes chanceux, nous avons un très grand jeu de données
- Ce jeu de données consiste en une table qui fournit les fréquences calculées pour chaque combinaison de valeurs
- Dans ce cas, nous avons tout ce qu'il nous faut pour établir les tables de probabilité conditionnelle

Premier cas: le réseau est connu (suite)

- Soit par exemple le réseau suivant:



Premier cas: le réseau est connu (suite)

Carie	Compétent	Douleur	Détecte	# occurrences
0	0	0	0	3
0	0	1	0	45
0	1	0	0	1
0	1	1	0	16
1	0	1	0	3
1	1	0	1	1
1	1	1	0	1
1	0	1	1	4
1	1	1	1	26
			TOTAL	100

Premier cas: le réseau est connu (suite)

Carie	Compétent	Douleur	Détecte	# occurrences
0	0	0	0	3
0	0	1	0	45
0	1	0	0	1
0	1	1	0	16
1	0	1	0	3
1	1	0	1	1
1	1	1	0	1
1	0	1	1	4
1	1	1	1	26
			TOTAL	100

P(Carie) = (3 + 1 + 1 + 4 + 26) / 100 = 35/100 = 0,35

Premier cas: le réseau est connu (suite)

Carie	Compétent	Douleur	Détecte	# occurrences
0	0	0	0	3
0	0	1	0	45
0	1	0	0	1
0	1	1	0	16
1	0	1	0	3
1	1	0	1	1
1	1	1	0	1
1	0	1	1	4
1	1	1	1	26
			TOTAL	100

P(Douleur|Carie) = (3 + 1 + 4 + 26) / (3 + 1 + 1 + 4 + 26) = 34/35= 0,97

Premier cas: le réseau est connu (suite)

Carie	Compétent	Douleur	Détecte	# occurrences
0	0	0	0	3
0	0	1	0	45
0	1	0	0	1
0	1	1	0	16
1	0	1	0	3
1	1	0	1	1
1	1	1	0	1
1	0	1	1	4
1	1	1	1	26
			TOTAL	100

P(Detecte|Carie, Pas Compétent) = 4 / (3 + 4) = 4/7 = 0,57

Apprentissage dans un réseau bayésien

(Quand il y a des variables inconnue)

L'algorithme EM

Espérance-maximisation

Quand le réseau est connu, mais il y a des variables inconnue

- On utilise les tables de probabilité et le réseau bayesien pour calculer les valeurs de probabilité de la variable inconnue
- Dans le cas de la valeur inconnue pour douleur, supposons que la probabilité est de 0,6
- On sépare alors l'entrée en deux entrées dont les poids sont modulés par la probabilité que nous venons de calculer

Espérance-maximisation (EM)

- Le problème est que pour faire cela, il faut utiliser les tables de probabilité conditionnelle que nous sommes justement en train de calculer!
- Solution: une processus itératif
- L'algorithme le plus utilisé dans ce cas est l'algorithme EM

Espérance-maximisation (EM)

- D'abord on fixe des valeurs arbitraires pour les tables de probabilité conditionnelles
- **Phase E :**
  - On calcule les probabilités des variables inconnues
  - On ajoute à la table les lignes supplémentaires requise en fonction des probabilités obtenues pour le valeurs inconnues
- **Phase M:**
  - On calcule les tables de probabilités du réseau, et on répète le processus jusqu'à convergence

À lire (Textbook)

- **Hidden variable models & the EM algorithm**
- **The EM algorithm for Bayesian networks**

Hidden variable models

- Given a model with observations given by $\tilde{x}_i$, a per observation hidden discrete random variable h_i and hidden continuous random variable z_i the marginal likelihood of such a model is given by

$$\mathrm{L}(\theta; \tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n) = \prod_{i=1}^{n} p(\tilde{x}_i; \theta) = \prod_{i=1}^{n} \int_{z_i} \sum_{h_i} p(\tilde{x}_i, z_i, h_i; \theta) dz_i$$

  where the sum is taken over all possible discrete values of h_i & the integral is taken over the entire domain of z_i

- To be extra clear about different quantities here we use $p(x_i = \tilde{x}_i) = p(\tilde{x}_i)$ to denote the probability of the random variable x_i associated with instance *i* taking the value represented by the observation $\tilde{x}_i$

Learning hidden variable models

- If a model has hidden variables we perform learning by integrating out the uncertainty of the hidden variables, i.e. we work with the *marginal likelihood*
- Taking the derivative of the marginal likelihood we see

$$\frac{\partial}{\partial\theta}\log p(\tilde{x}_i;\theta) = \frac{1}{p(\tilde{x}_i;\theta)}\frac{\partial}{\partial\theta}\int_{z_i}\sum_{h_i} p(\tilde{x}_i, z_i, h_i;\theta)dz_i$$

$$= \int_{z_i}\sum_{h_i}\frac{p(\tilde{x}_i, z_i, h_i;\theta)}{p(\tilde{x}_i;\theta)}\frac{\partial}{\partial\theta}\log p(\tilde{x}_i, z_i, h_i;\theta)dz_i$$

$$= \int_{z_i}\sum_{h_i} p(z_i, h_i \mid \tilde{x}_i;\theta)\frac{\partial}{\partial\theta}\log p(\tilde{x}_i, z_i, h_i;\theta)dz_i$$

$$= \mathrm{E}\left[\frac{\partial}{\partial\theta}\log p(\tilde{x}_i, z_i, h_i;\theta)\right]_{p(z_i, h_i \mid \tilde{x}_i;\theta)}$$

Learning with expected gradient descent (EGD)

- The marginal likelihood can be optimized using gradient ascent by computing and following the expected gradient of the log joint likelihood
- This can be decomposed into three steps, a

  P-step where we compute the posterior over hidden variables; then an

  E-step where we compute the expectation of the gradient given the posterior; then a

  G-step, where we use gradient-based optimization to maximize the (*marginal likelihood*) objective function with respect to the model parameters.

The Expectation Maximization (EM) Algorithm

- A more well known method, similar to EGD
- The EM algorithm consists of two steps, an **E-step** that computes the expectations used in the expected log-likelihood; and an **M-step** in which the objective is maximized – typically using a closed-form parameter update.
- The expected log joint likelihood is related to the marginal likelihood in the following way

$$\log P(\tilde{x}_i;\theta) = \mathrm{E}_{p(z_i,h_i \mid \tilde{x}_i;\theta)}\left[\log p(\tilde{x}_i, z_i, h_i;\theta)\right] + \mathrm{H}[p(z_i,h_i \mid \tilde{x}_i;\theta)],$$

The EM algorithm in more detail

- In a joint probability model with discrete hidden variables **H** the probability of the observed data can be maximized by initializing θ^{old} and repeating the steps

1. **E-step**: Compute expectations using $P(H \mid X;\theta^{old})$
2. **M-step**: Find $\theta^{new} = \arg\max_{\theta} \left[\sum_{H} P(H \mid X;\theta^{old}) \log P(X,H;\theta) \right]$
3. If the algorithm has not converged, set $\theta^{old} = \theta^{new}$ and return to step 1

- The M step corresponds to maximizing the expected log-likelihood, the overall procedure maximizes the marginal likelihood of the data
- Although discrete hidden variables are used above, the approach generalizes to continuous ones

EM for Bayesian networks

- Unconditional probability distributions are estimated in the same way in which it would be computed if the variables A_i had been observed, but with each observation replaced by its posterior marginal probability (i.e. marginalizing over the other variables)

$$\theta_{A=a}^{new} = P(A=a) = \frac{1}{N}\sum_{i=1}^{N} P(A_i = a \mid \{\tilde{X}\}_i; \theta^{current})$$

- Conditional distributions are updated using ratios of posterior marginals, ex.

$$P(B=b \mid A=a) = \frac{\sum_{i=1}^{N} P(A_i = a, B_i = b \mid \{\tilde{X}\}_i; \theta^{current})}{\sum_{i=1}^{N} P(A_i = a \mid \{\tilde{X}\}_i; \theta^{current})}.$$

EM in practice

- For many latent variable models – Gaussian mixture models, probabilistic principal component analysis, and hidden Markov models – the required posterior distributions can be computed exactly, which accounts for their popularity.
- However, for many probabilistic models it is simply not possible to compute an exact posterior distribution.
- This can easily happen with multiple hidden random variables, because the posterior needed in the E-step is the joint posterior of the hidden variables.
- There is a vast literature on the subject of how to compute approximations to the true posterior distribution over hidden variables in more complex models.
- Sampling methods and variational methods are popular in machine learning

Bayesian estimation and prediction

- In a more Bayesian modelling setting the joint distribution of data *and* parameters under a model can be written as

$$p(x_1, x_2, \ldots, x_n, \theta; \alpha) = \prod_{i=1}^{n} p(x_i \mid \theta) p(\theta; \alpha),$$

- Where α is a hyperparameter
- Bayesian predictions use a quantity known as the *posterior predictive distribution* - the probability model for a new observation marginalized over the posterior probability inferred for the parameters given the observations so far.

$$p(x_{new} \mid \tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n; \alpha) = \int_{\theta} p(x_{new} \mid \theta) p(\theta \mid \tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n; \alpha) d\theta,$$

Empirical Bayesian methods

- One approach is obtained by maximizing the marginal likelihood with respect to the model's hyperparameters, ex.

$$\alpha_{MML} = \underset{\alpha}{\operatorname{argmax}} \left[\log \int \prod_{i=1}^{n} p(x_i \mid \theta) p(\theta; \alpha) d\theta \right]$$

Apprentissage dans un réseau bayésien

(Quand il y a des variables inconnue)

L'algorithme EM

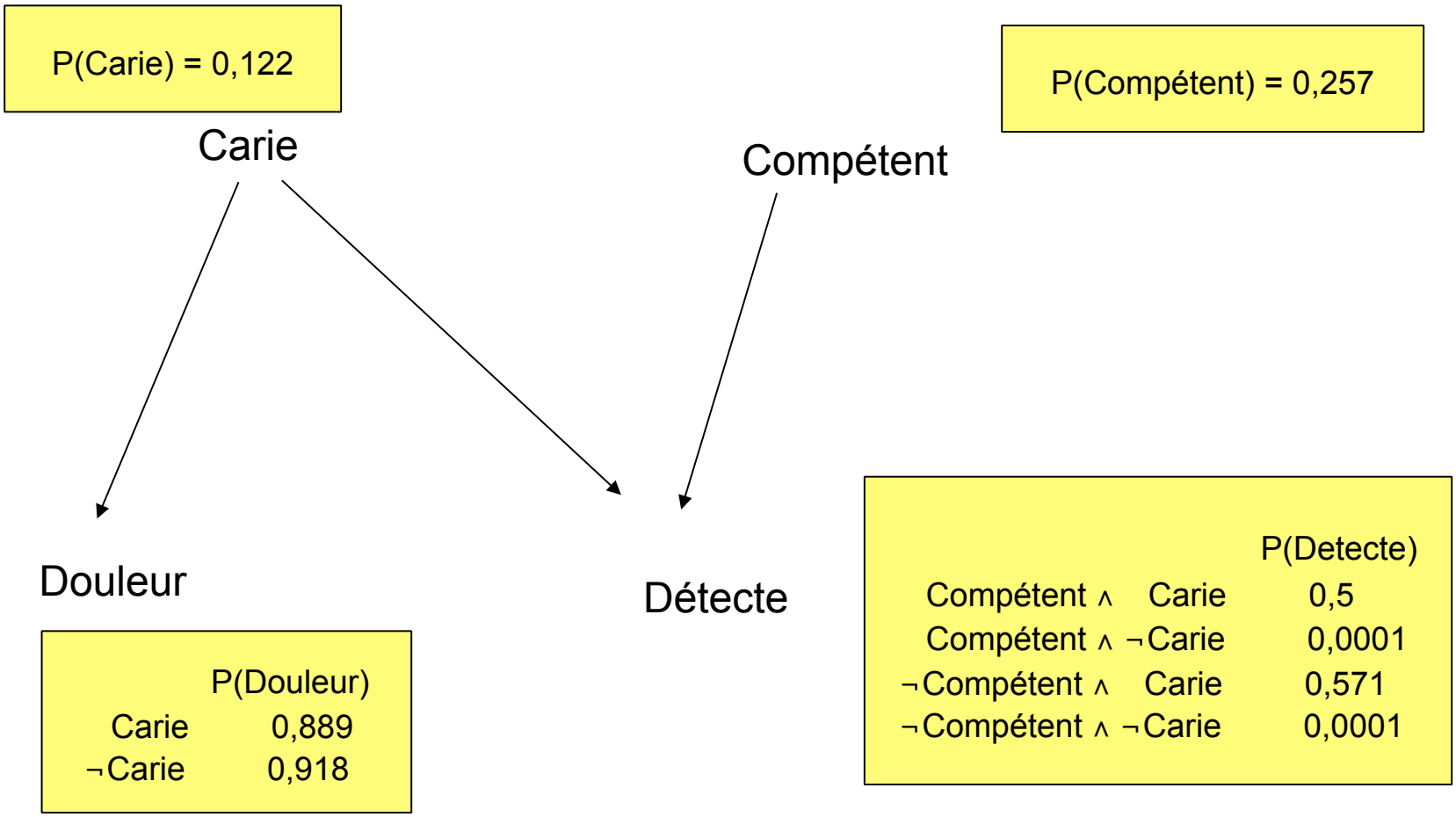
Espérance-maximisation

Algorithme EM - exemple

- Que fait-on maintenant si certaines valeurs sont inconnues:

Carie	Compétent	Douleur	Détecte	#
0	0	0	0	3
0	0	1	0	45
0	1	*	0	**17**
1	0	1	0	3
1	1	0	1	1
1	1	1	0	1
*	*	1	1	**30**
			Total	100

Valeurs initiales (fixées arbitrairement)



Algorithme EM - exemple

- Première itération:

P(Douleur = 1 | Carie = 0, Compétent = 1, Détecte = 0) = 0,918

P(Carie = 0, Compétent = 0 | Douleur = 1, Détecte = 1) = 0,001017
P(Carie = 0, Compétent = 1 || Douleur = 1, Détecte = 1) = 0,000351
P(Carie = 1, Compétent = 0 | Douleur = 1, Détecte = 1) = 0,7665
P(Carie = 1, Compétent = 1 | Douleur = 1, Détecte = 1) = 0,2322

Carie	Compétent	Douleur	Détecte	#	
0	0	0	0	3	
0	0	1	0	45	(= 0,072*__17__)
0	1	0	0	1,394	(= 0,918*__17__)
0	1	1	0	15,606	
1	0	1	0	3	
1	1	0	1	1	
1	1	1	0	1	
0	0	1	1	0,02993	(= 0,001017*__30__)
0	1	1	1	0,0104	(= 0,000351*__30__)
1	0	1	1	22,9949	(= 0,7665*__30__)
1	1	1	1	6,9648	(= 0,2322*__30__)
			Total	100	

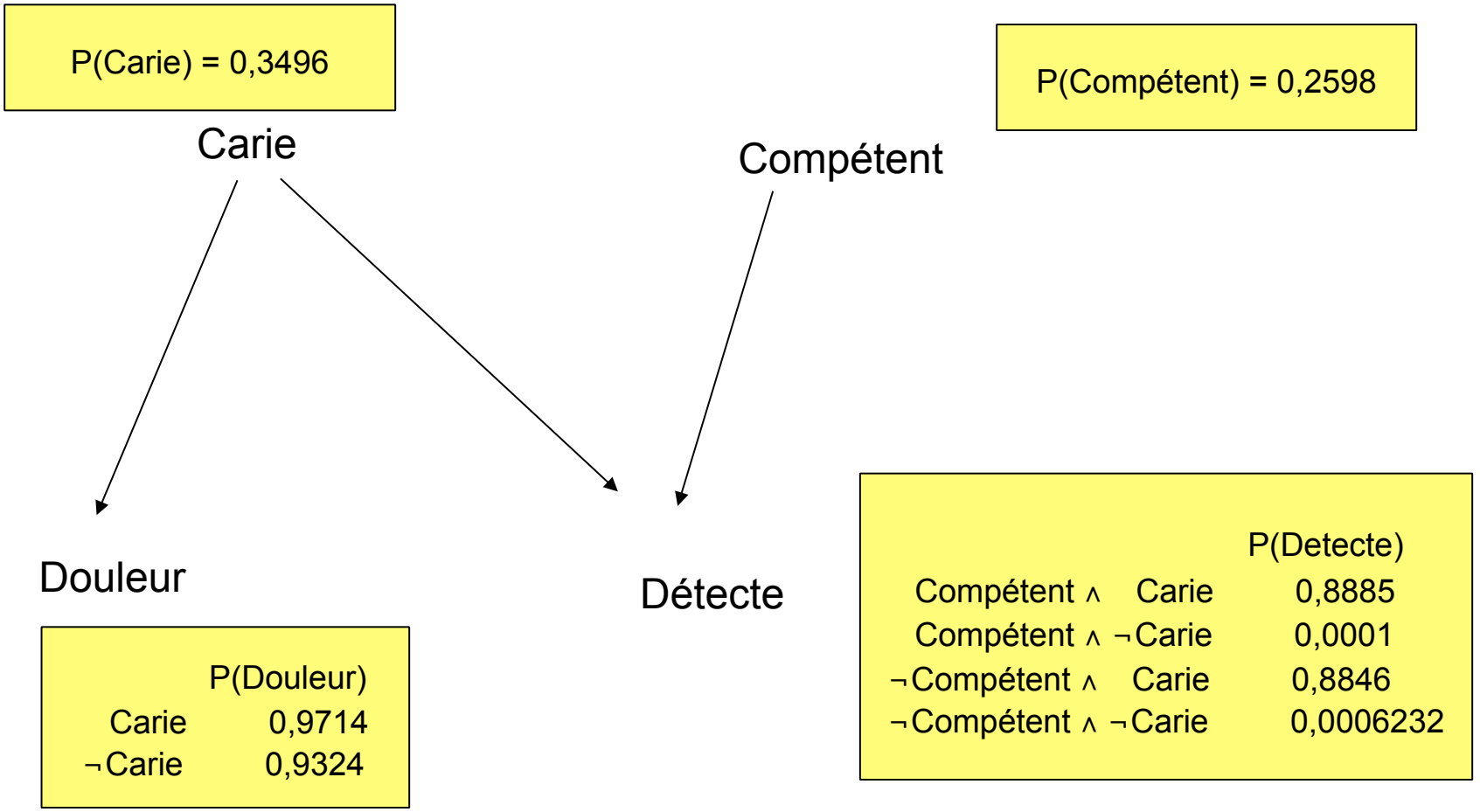
Algorithme EM - exemple

Carie	Compétent	Douleur	Détecte	N	P	N*ln(P)
0	0	0	0	3	0,0535	-8,78
0	0	1	0	45	0,5988	-23,08
0	1	*	0	17	0,2256	-25,31
1	0	1	0	3	0,0346	-10,09
1	1	0	1	1	0,0017	-6,35
1	1	1	0	1	0,0139	-4,27
*	*	1	1	30	0,0600	-84,39

Log vraisemblance = -162,28

-162,28

Algorithme EM - exemple



Algorithme EM - exemple

P(Douleur = 1 | Carie = 0, Compétent = 1, Détecte = 0) = 0,9324

P(Carie = 0, Compétent = 0 | Douleur = 1, Détecte = 1) = 0,0009293
P(Carie = 0, Compétent = 1 || Douleur = 1, Détecte = 1) = 0,0000523
P(Carie = 1, Compétent = 0 | Douleur = 1, Détecte = 1) = 0,7387
P(Carie = 1, Compétent = 1 | Douleur = 1, Détecte = 1) = 0,2603

Carie	Compétent	Douleur	Détecte	#
0	0	0	0	3
0	0	1	0	45
0	1	0	0	1,158
0	1	1	0	15,852
1	0	1	0	3
1	1	0	1	1
1	1	1	0	1
0	0	1	1	0,2788
0	1	1	1	0,00157
1	0	1	1	22.16

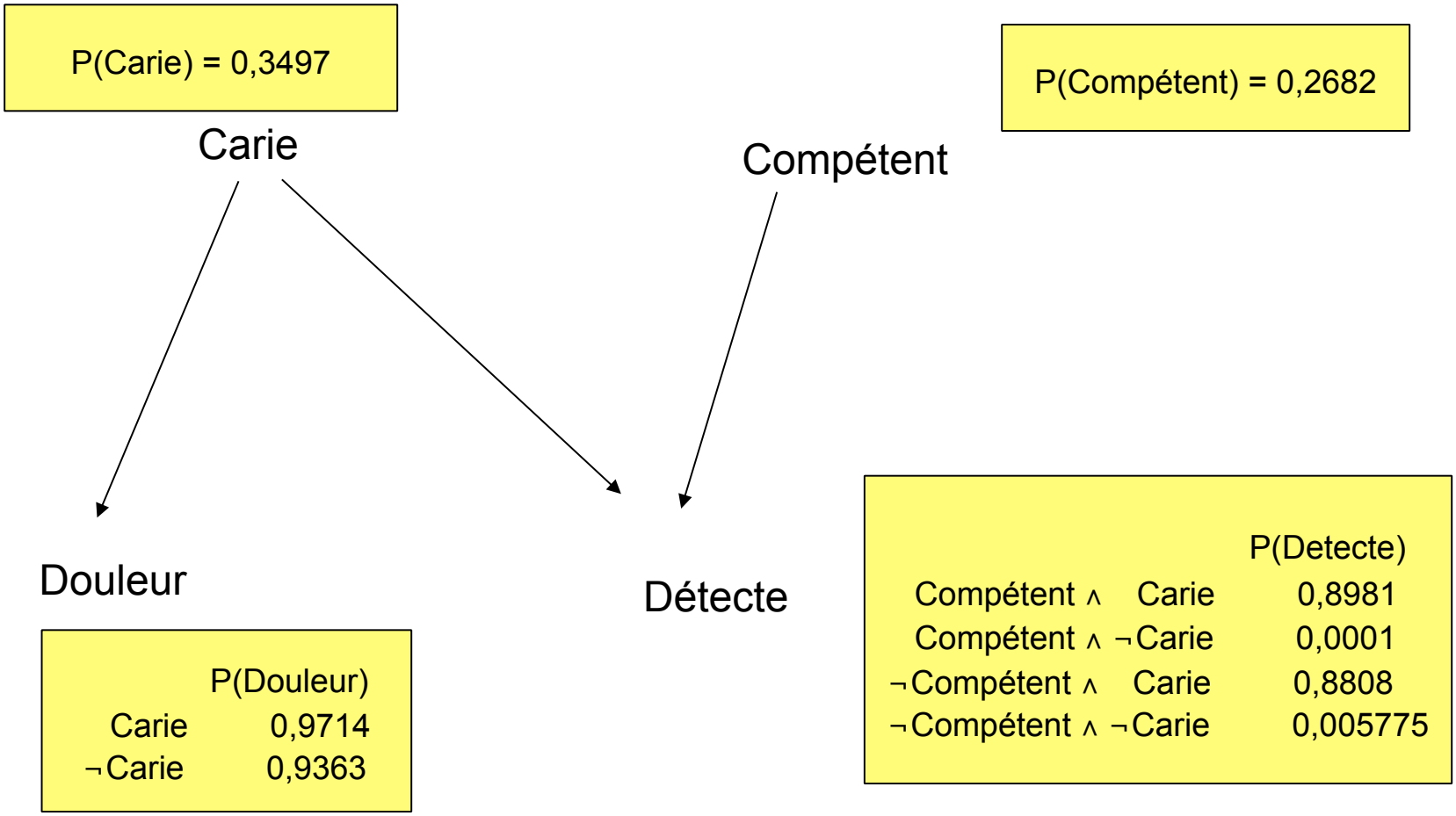
Algorithme EM - exemple

Carie	Compétent	Douleur	Détecte	N	P	N*ln(P)
0	0	0	0	3	0,0325	-10,28
0	0	1	0	45	0,4486	-36,07
0	1	*	0	17	0,1690	-30,23
1	0	1	0	3	0,0290	-10,62
1	1	0	1	1	0,0023	-6,07
1	1	1	0	1	0,0098	-4,62
*	*	1	1	30	0,3011	-36,01

Log vraisemblance = -133,91

-133,91

Algorithme EM - exemple



Algorithme EM - exemple

P(Douleur = 1 | Carie = 0, Compétent = 1, Détecte = 0) = 0,9324

P(Carie = 0, Compétent = 0 | Douleur = 1, Détecte = 1) = 0,0008265
P(Carie = 0, Compétent = 1 || Douleur = 1, Détecte = 1) = 0,0000525
P(Carie = 1, Compétent = 0 | Douleur = 1, Détecte = 1) = 0,7062
P(Carie = 1, Compétent = 1 | Douleur = 1, Détecte = 1) = 0,2855

Carie	Compétent	Douleur	Détecte	#
0	0	0	0	3
0	0	1	0	45
0	1	0	0	1,15
0	1	1	0	15,85
1	0	1	0	3
1	1	0	1	1
1	1	1	0	1
0	0	1	1	0,0269
0	1	1	1	0,00168
1	0	1	1	21,83
1	1	1	1	8,15

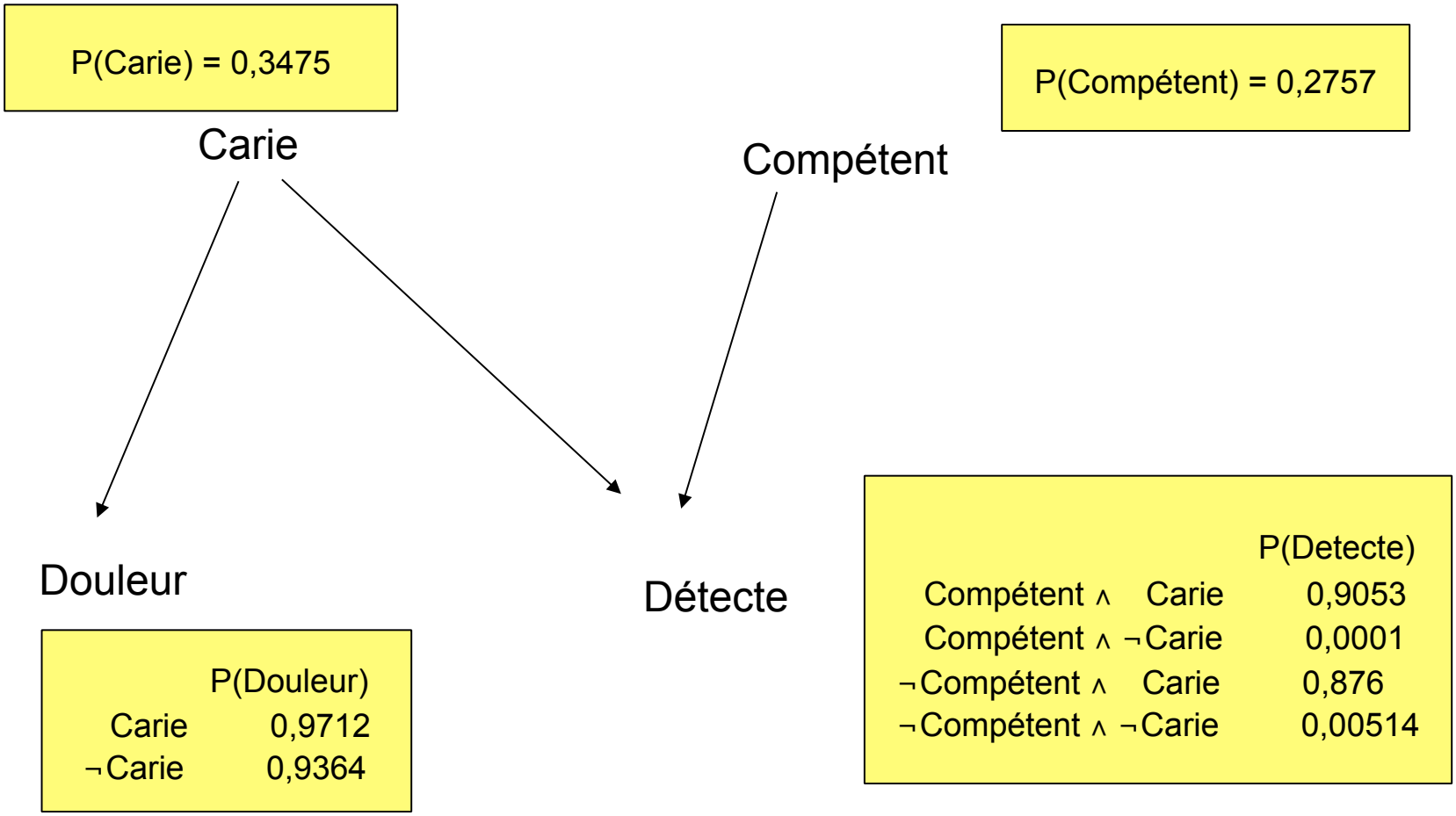
Algorithme EM - exemple

Carie	Compétent	Douleur	Détecte	N	P	N*ln(P)
0	0	0	0	3	0,0303	-10,49
0	0	1	0	45	0,4453	-36,40
0	1	*	0	17	0,1744	-29,69
1	0	1	0	3	0,0296	-10,56
1	1	0	1	1	0,0024	-6,03
1	1	1	0	1	0,0093	-4,68
*	*	1	1	30	0,3011	-36,01

Log vraisemblance = -133,86

-133,86

Algorithme EM - exemple



Algorithme EM - exemple

P(Douleur = 1 | Carie = 0, Compétent = 1, Détecte = 0) = 0,9364

P(Carie = 0, Compétent = 0 | Douleur = 1, Détecte = 1) = 0,007566
P(Carie = 0, Compétent = 1 || Douleur = 1, Détecte = 1) = 0,00005602
P(Carie = 1, Compétent = 0 | Douleur = 1, Détecte = 1) = 0,7122
P(Carie = 1, Compétent = 1 | Douleur = 1, Détecte = 1) = 0,2801

Carie	Compétent	Douleur	Détecte	#
0	0	0	0	3
0	0	1	0	45
0	1	0	0	1,08
0	1	1	0	15,92
1	0	1	0	3
1	1	0	1	1
1	1	1	0	1
0	0	1	1	0,227
0	1	1	1	0,00168
1	0	1	1	21,37
1	1	1	1	8,404

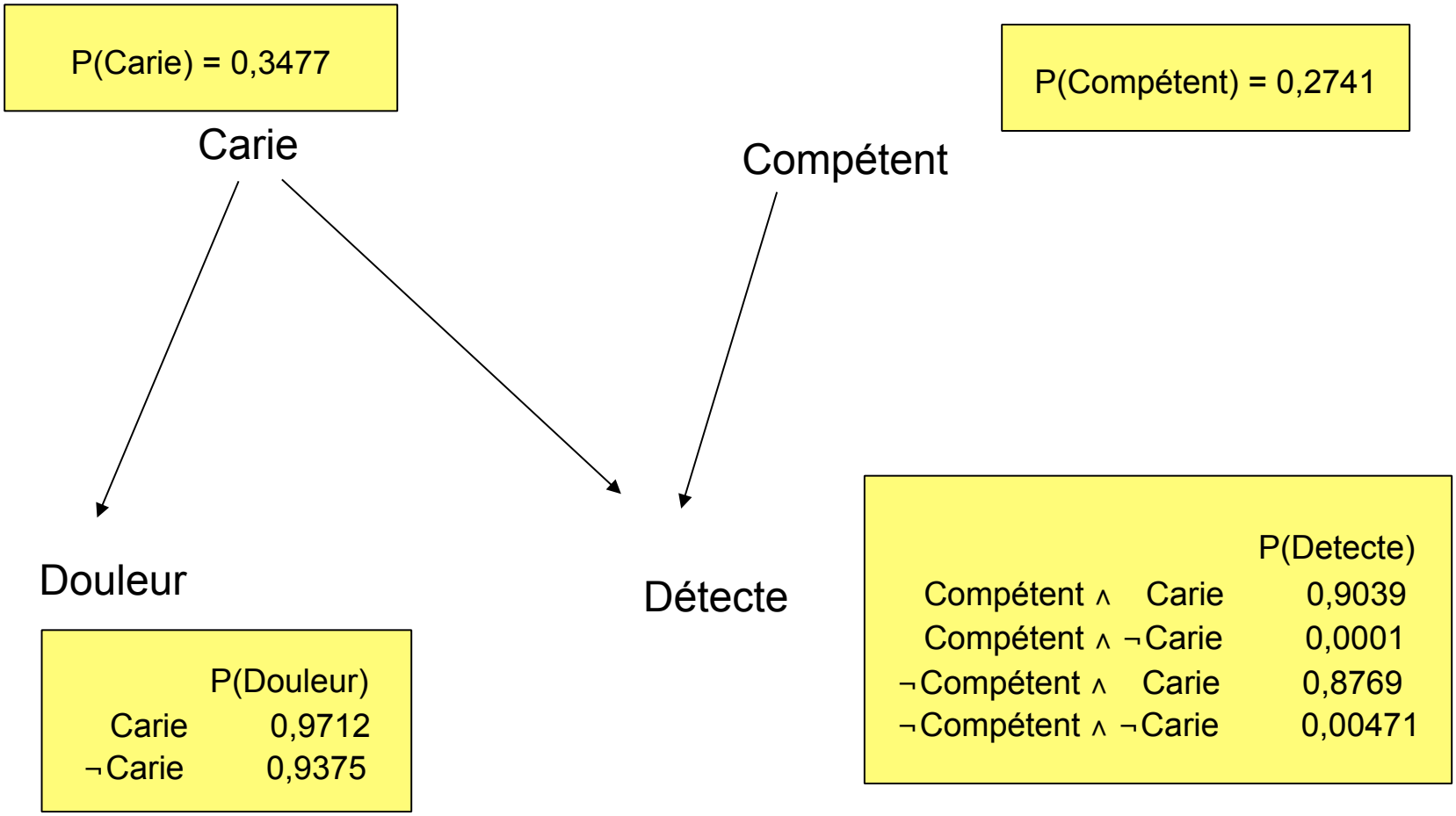
Algorithme EM - exemple

Carie	Compétent	Douleur	Détecte	N	P	N*ln(P)
0	0	0	0	3	0,0299	-10,53
0	0	1	0	45	0,4403	-36,92
0	1	*	0	17	0,1799	-29,16
1	0	1	0	3	0,0303	-10,49
1	1	0	1	1	0,0025	-5,99
1	1	1	0	1	0,0088	-4,73
*	*	1	1	30	0,3007	-36,05

Log vraisemblance = -133,87

-133,87

Algorithme EM - exemple



EM – un peu plus formellement

- But: ajuster θ pour maximiser la probabilité des données $\tilde{x}_i$

$$P(\tilde{X};\theta) = \prod_{i=1}^{N} P(\tilde{x}_i;\theta) = \prod_{i=1}^{N} \int_{z_i} \sum_{h_i} p(\tilde{x}_i, z_i, h_i;\theta) dz_i$$

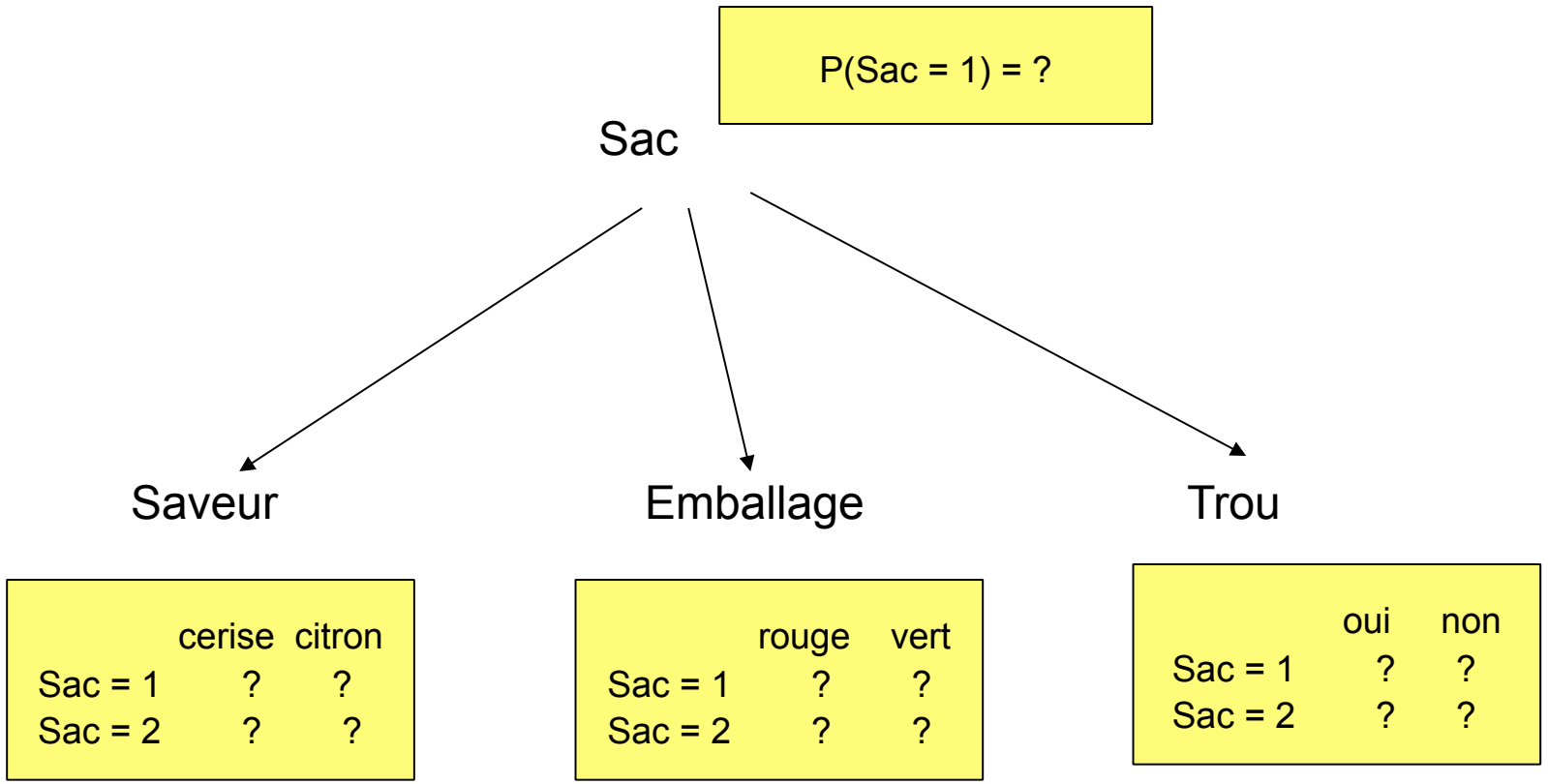
avec une marginalisation sur les variable cachées, h_i – discrète, et z_i – continue

- Cependant, il est possible de montrer que

$$\frac{\partial}{\partial\theta} \log p(\tilde{x}_i;\theta) = \sum_{i=1}^{n} \mathrm{E}\left[\frac{\partial}{\partial\theta} \log p(\tilde{x}_i, z_i, h_i;\theta)\right]_{p(z_i, h_i \mid \tilde{x}_i;\theta)},$$

ou, la dérivée du log de la probabilité marginale est égale à la dérivée du log de la probabilité jointe espérée en fonction de la distribution a posteriori de h_i et z_i

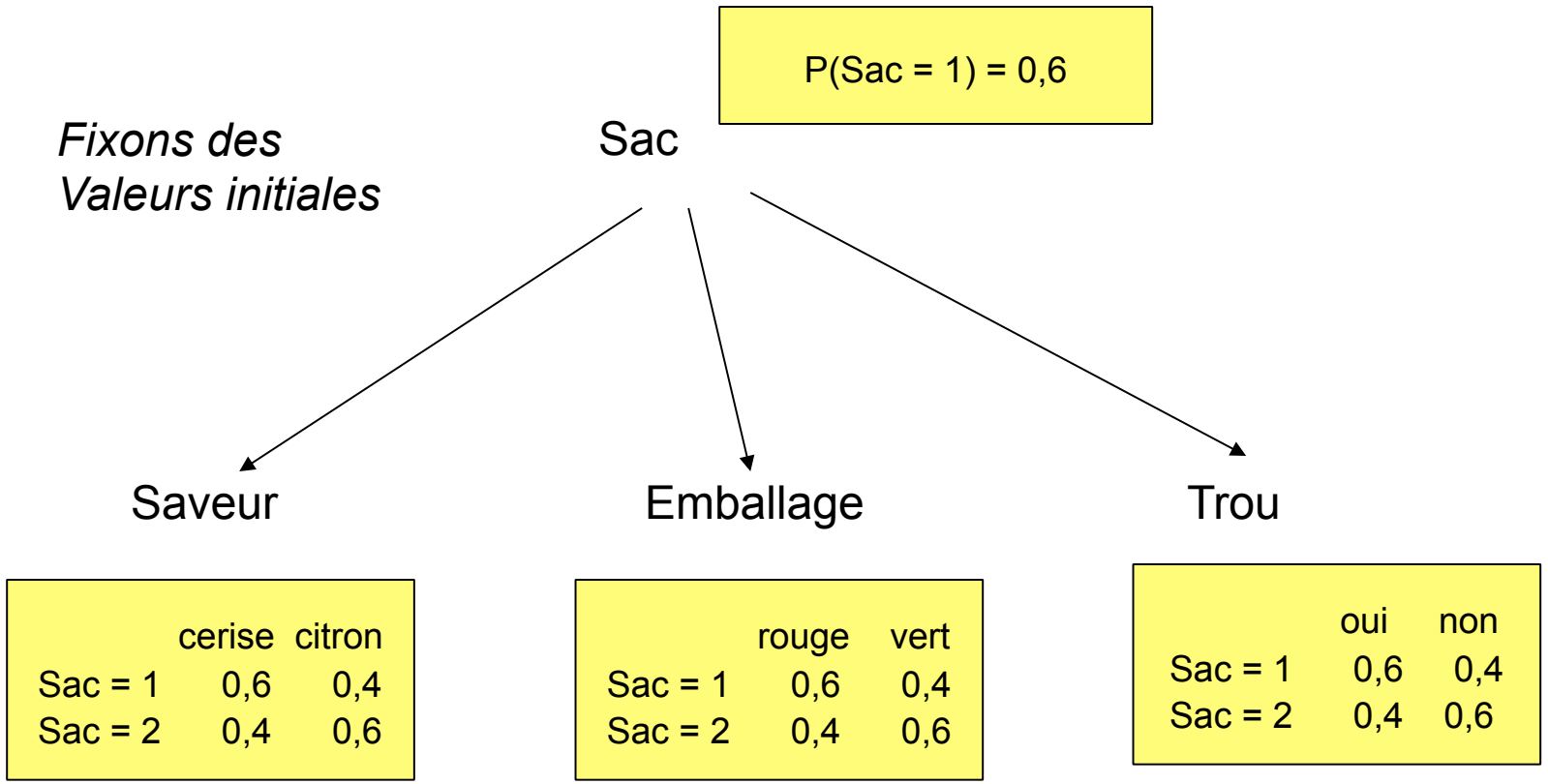
Algorithme EM – exemple avec "Naïve Bayes" et la "class" cachée (sac)



Algorithme EM – exemple des données

Sac	Saveur	Emballage	Trou	N
*	cerise	rouge	oui	273
*	cerise	rouge	non	93
*	cerise	vert	oui	104
*	cerise	vert	non	90
*	citron	rouge	oui	79
*	citron	rouge	non	100
*	citron	vert	oui	94
*	citron	vert	non	167

Algorithme EM – les paramètres préliminaire



Algorithme EM – Phase E, calculer les postérieurs

Probabilités selon le réseau bayesien avec les probabilités initiales

P(sac=1 | cerise, rouge, oui)	0,8351
P(sac=1 | cerise, rouge, non)	0,6923
P(sac=1 | cerise, vert, oui)	0,6923
P(sac=1 | cerise, vert, non)	0,5000
P(sac=1 | citron, rouge, oui)	0,6923
P(sac=1 | citron, rouge, non)	0,5000
P(sac=1 | citron, vert, oui)	0,5000
P(sac=1 | citron, vert, non)	0,3077

Algorithme EM – Phase E, calculer les statistiques équivalentes

Sac	Saveur	Emballage	Trou	N	
1	cerise	rouge	oui	227,9823	(= 0,8351 * 273)
1	cerise	rouge	non	64,3839	
1	cerise	vert	oui	71,9992	
1	cerise	vert	non	45	
1	citron	rouge	oui	54,6917	(= 0,1649 * 273)
1	citron	rouge	non	50	
1	citron	vert	oui	47	
1	citron	vert	non	51,3859	
2	cerise	rouge	oui	45,0177	
2	cerise	rouge	non	28,6161	

Algorithme EM – vérification, calculer la log vraisemblance

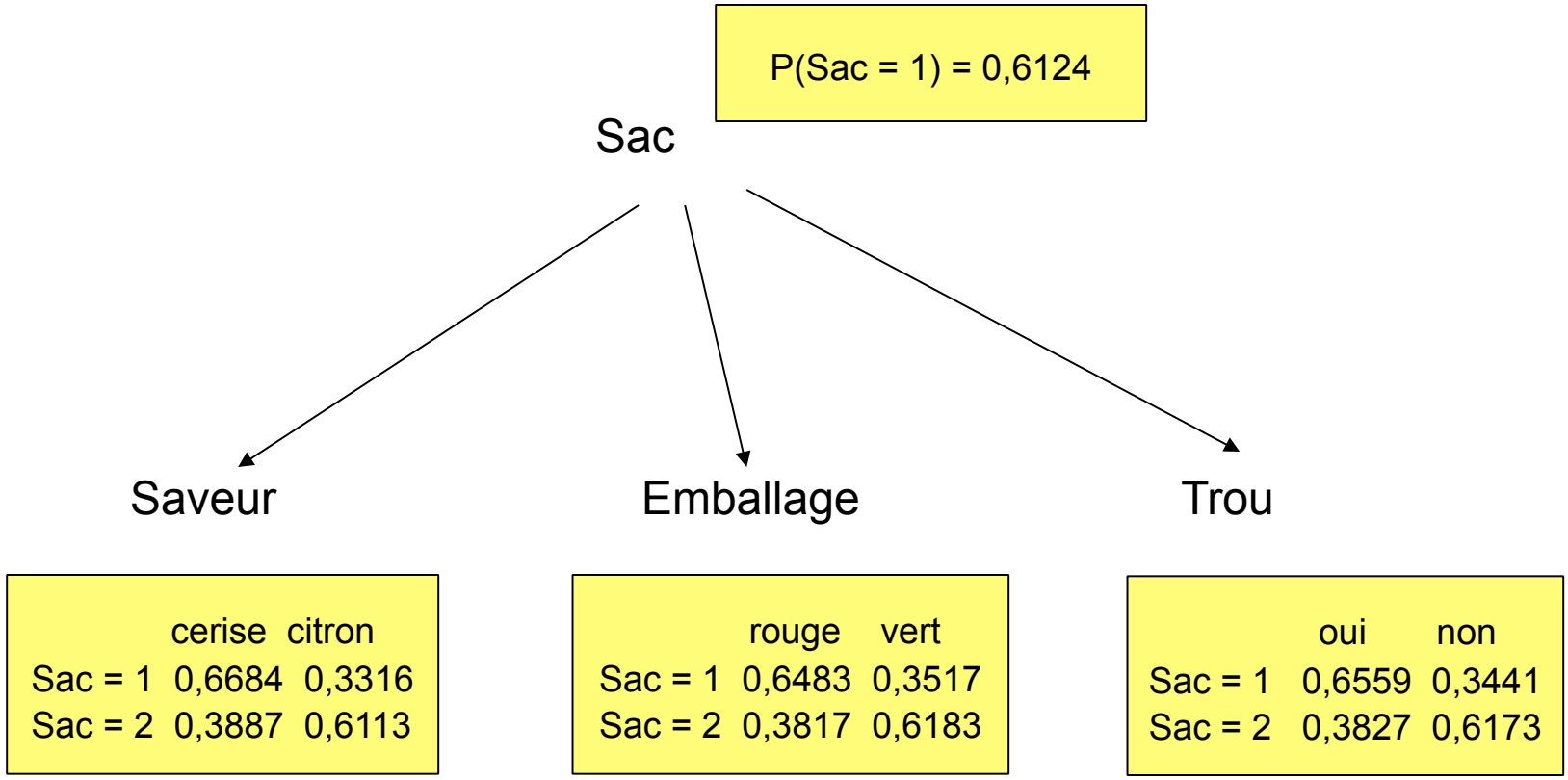
Saveur	Emballage	Trou	P(S,E,T)	N*Ln(P)
cerise	rouge	oui	0,1552	-508,61
cerise	rouge	non	0,1248	-193,54
cerise	vert	oui	0,1248	-216,43
cerise	vert	non	0,1152	-194,50
citron	rouge	oui	0,1248	-164,40
citron	rouge	non	0,1152	-216,11
citron	vert	oui	0,1152	-203,14
citron	vert	non	0,1248	-347,53

=273 * ln(0,1552)

Log vraisemblance = -2044,26

Algorithme EM – Phase M

Mettre à jour les probabilités inconditionnelles et conditionnelles en utilisant les statistiques équivalentes



Algorithme EM – Phase E

P(sac=1 | cerise, rouge, oui)	0,8878
P(sac=1 | cerise, rouge, non)	0,7201
P(sac=1 | cerise, vert, oui)	0,7259
P(sac=1 | cerise, vert, non)	0,4628
P(sac=1 | citron, rouge, oui)	0,7139
P(sac=1 | citron, rouge, non)	0,448
P(sac=1 | citron, vert, oui)	0,4552
P(sac=1 | citron, vert, non)	0,2137

Algorithme EM – Phase E

Sac	Saveur	Emballage	Trou	N	
1	cerise	rouge	oui	242,37	
1	cerise	rouge	non	66,97	
1	cerise	vert	oui	75,49	
1	cerise	vert	non	41,65	
1	citron	rouge	oui	56,40	
1	citron	rouge	non	44,80	
1	citron	vert	oui	42,79	(= 0,1122 * 273)
1	citron	vert	non	35,69	
2	cerise	rouge	oui	30,63	
2	cerise	rouge	non	26,03	
2	cerise	vert	oui	28,51	
2	cerise	vert	non	48,35	
2	citron	rouge	oui	22,60	
2	citron	rouge	non	55,20	
2	citron	vert	oui	51,21	
2	citron	vert	non	131.31	

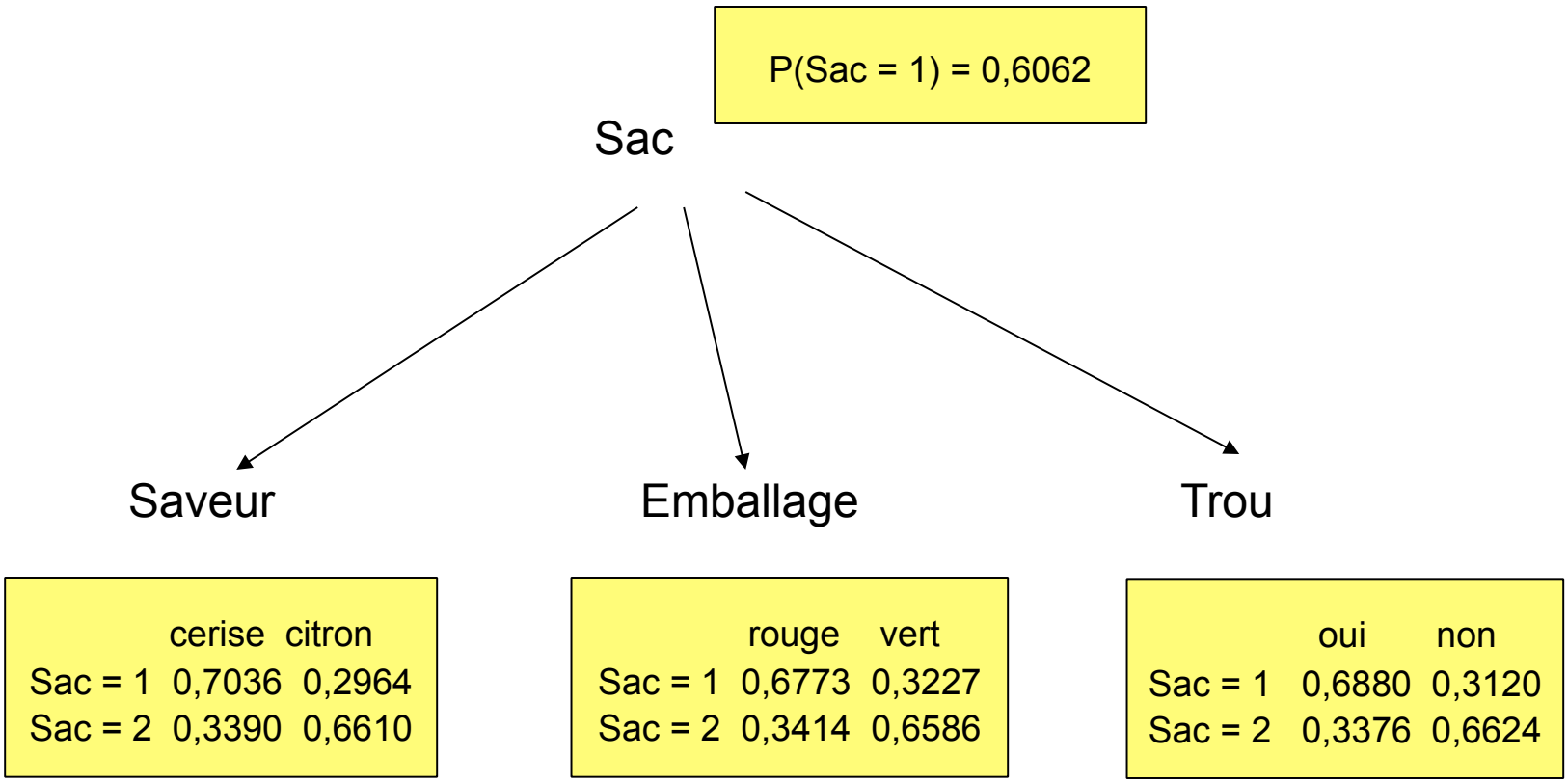
(= 0,8878 * 273)

Algorithme EM – exemple avec variable cachée

Saveur	Emballage	Trou	P	N*Ln(p)
cerise	rouge	oui	0,1961	-444,75
cerise	rouge	non	0,1268	-192,06
cerise	vert	oui	0,1301	-212,10
cerise	vert	non	0,107	-201,14
citron	rouge	oui	0,121	-166,85
citron	rouge	non	0,1011	-229,16
citron	vert	oui	0,11029	-213,76
citron	vert	non	0,115	-361,19

Log vraisemblance = -2021

Algorithme EM – Phase M



À lire (textbook)

- **APPENDIX A : VARIATIONAL METHODS AND THE EM ALGORITHM**

INF8225

Modèles probabilistes séquentiels et apprentissage automatique

Leçon 4

Christopher Pal

École Polytechnique de Montréal

Plan du cours

- Modèles temporels et séquentiels.
- Modèles de Markov cachés et réseaux bayésiens dynamiques.
- Champs aléatoires de Markov et
- Généralisations de ces modèles avec les modèles probabilistes graphiques
- Modèles génératifs vs modèles discriminatifs et
- Leurs impacts sur la conception de systèmes concrets, tels que les systèmes pour la recherche d'information et l'exploration de Données

Markov models

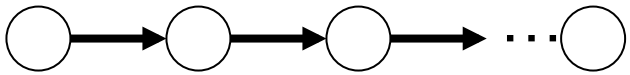
- One simple and effective probabilistic model for discrete sequential data is known as a *Markov model*
- A first-order Markov model assumes that each symbol in a sequence can be predicted using its conditional probability given the preceding symbol.
- An unconditional probability is used for the first symbol
- Given observation variables O={O_1,…,O_T}, we have

$$P(\mathrm{O}) = P(O_1)\prod_{t=1}^{T} P(O_{t+1} \mid O_t).$$

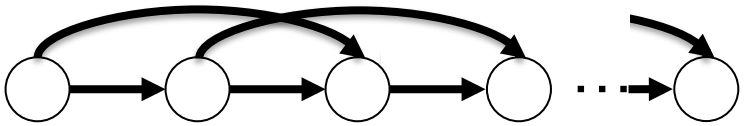
- Usually, every conditional probability used in such models is the same
- The Bayesian network is a linear chain of variables with directed edges between each successive pair

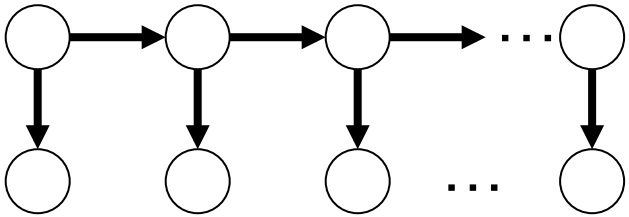
Extending Markov models

First order Markov model

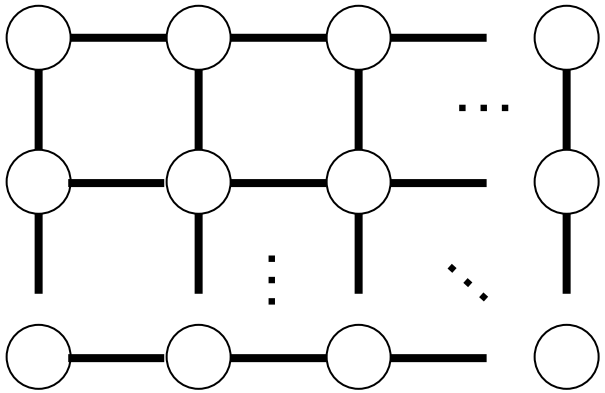


Second order Markov model





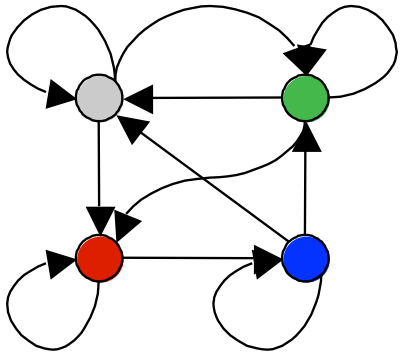
Hidden Markov model

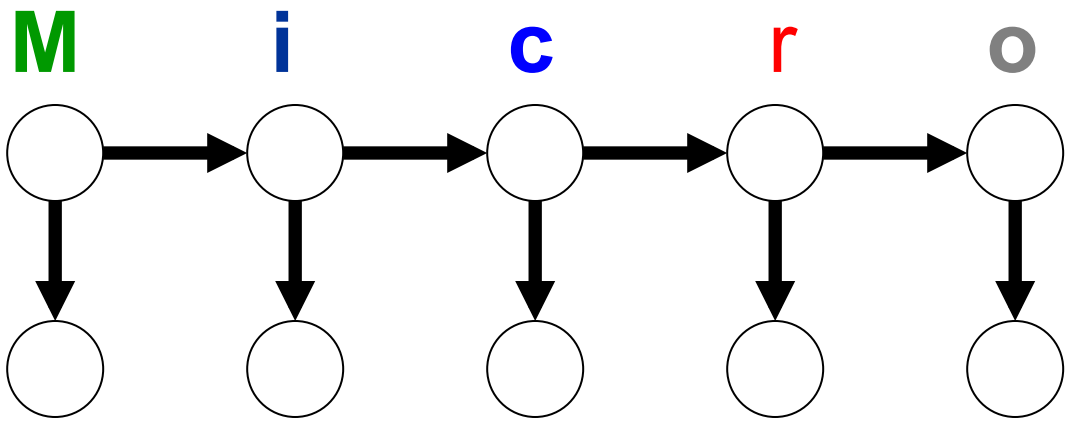


Markov random field

Modèles probabilistes séquentiels

- Les modèles de Markov cachés (Hidden Markov Models HMMs)



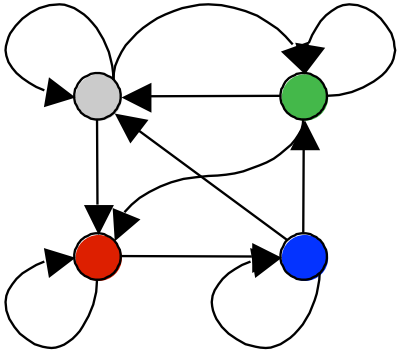


Today in Redmond, **Microsoft** Chair Bill Gates…

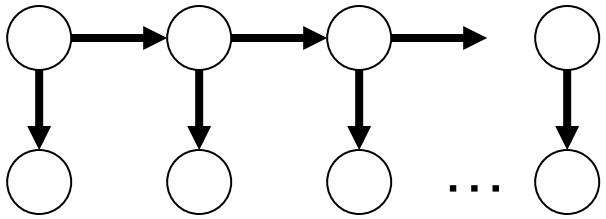
- La modélisation des séquences et des donées temporels : ex. texte, parole, la reconnaissance des mot et l'extraction dinformation

Automates stochastiques, réseaux bayésiens dynamiques, champs aléatoires de Markov et champs aléatoires conditionnels

(1) Automates finis stochastiques

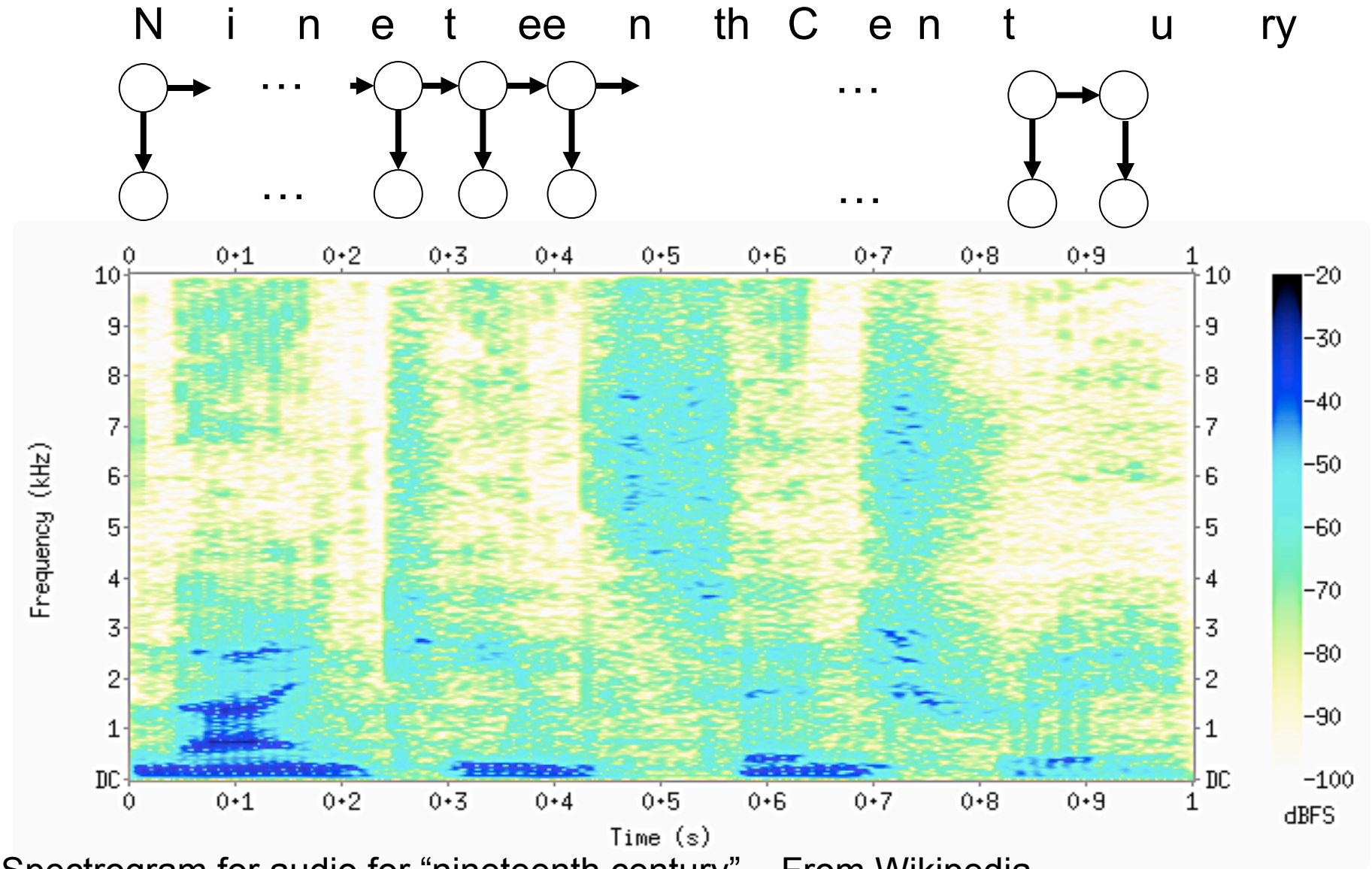


(2) Réseaux bayésiens dynamiques (DBN)



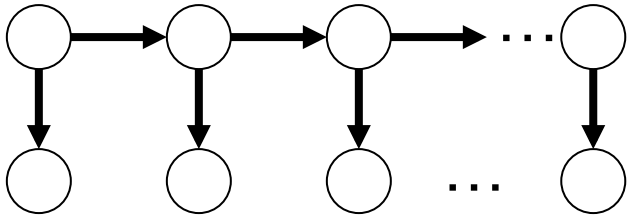
- Confrontons alors ces *modèles génératifs* aux *modèles conjoints* (a) et *discriminatifs* (b)

Example: HMMs for speech recognition



Spectrogram for audio for “nineteenth century” – From Wikipedia

Hidden Markov models



- A hidden Markov model is a joint probability model of a set of discrete observed variables O={O_1,…,O_T} and discrete hidden variables H={H_1,…,H_T} for T observations that factors the joint distribution as follows

$$P(\mathrm{O},\mathrm{H}) = P(H_1)\prod_{t=1}^{T} P(H_{t+1} \mid H_t)\prod_{t=1}^{T} P(O_t \mid H_t),$$

- Each O_t is a discrete random variable with N possible values, and each H_t is a discrete random variable with M possible values.
- The previous figure illustrates a hidden Markov model as a type of Bayesian network that is known as a "dynamic" Bayesian network where variables are replicated dynamically over the appropriate number of time steps.

Hidden Markov models (HMMs)

- Common to use "time-homogeneous" models where the transition matrix $P(H_{t+1}|H_t)$ is the same at each time step
- Define **A** to be a transition matrix whose elements encode $P(H_{t+1}= j|H_t= i)$, and **B** to be an emission matrix **B** whose elements b_{ij} correspond to $P(O_t= j|H_t= i)$
- For t=1, the initial state probability distribution is encoded in a vector π with elements $\pi_i=P(H_t=i)$
- The complete set of parameters is $\theta=\{\mathbf{A},\mathbf{B},\boldsymbol{\pi}\}$, i.e. a set containing two matrices and one vector
- We write a particular observation sequence as a set of observations

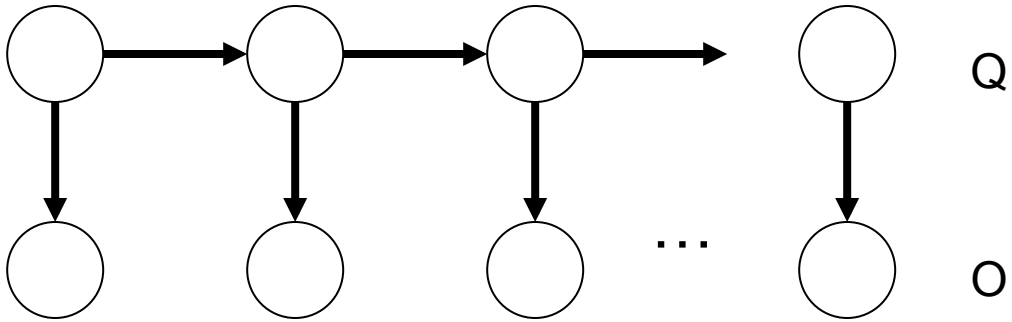
$$\tilde{\mathrm{O}}=\{O_1 = o_1,\ldots,O_T = o_T\}$$

HMMs: key problems

1. Compute $P(\tilde{O};\theta)$, the probability of a sequence under the model with parameters θ
2. Find the most probable explanation—the best sequence of states $H^* = \{H_1 = h_1, \ldots, H_T = h_T\}$ that explains an observation
3. Find the best parameters θ for the model given a data set of observed sequences

- The first problem can be solved using the sum-product algorithm, the second using the max-product algorithm, and the third using the EM algorithm in which the required expectations are computed using the sum-product algorithm

Les modèles de Markov caché (MMC) « Hidden Markov Models - HMMs »



- Un MMC est un modèle de probabilité joint des variables observées $\{O_t\}$ (discret or continu) et des variables cachée $\{Q_t\}$ (discret)

$$P(\{O\},\{Q\}) = \prod_{t=1}^{T} P(O_t \mid Q_t) \prod_{t=1}^{T} P(Q_{t+1} \mid Q_t) P(Q_t)$$

MMCs - Définitions en détail

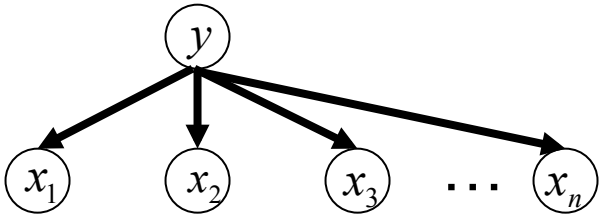
- Un séquence d'observation est donné par $O = (O_1 = o_1, \ldots, O_T = o_T)$
- Q_t est une variable aléatoire discrète avec des valeurs possibles parmi $\{1, \ldots, D\}$.
- Donc, nous pouvons définir une matrice de transition stochastique et indépendante du temps $A = \{a_{ij}\} = P(Q_{t+1} = j \mid Q_t = i)$.
- Normalement on utilise la même probabilité conditionnelle $P(Q_{t+1} \mid Q_t)$ à chaque étape t
- *Nous appelons* $P(O_t \mid Q_t)$ la matrice d'émission
- Le cas spécial du temps t=1 est décrit par la distribution d'état initial $\pi_i = P(Q_t = i)$

Les solutions de trois problèmes

- Nous décrivons l'ensemble complet de paramètres d'un MMC par λ = (A, B, π)
- Il y a trois problèmes de base liés aux MMCs

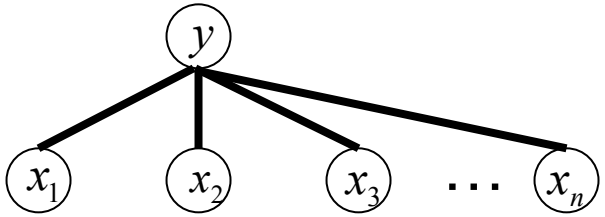
1. Trouvez p(O|λ) pour certains O = (O_1= o_1,…,O_T= o_T). Nous employons l'algorithme « **forward-backward** » ou l'algorithme somme-produit
2. Donné quelques observations et quelques λs, trouvez le meilleur ordre d'état q = (q_1,…, q_T) qui explique O L'algorithme de **Viterbi** qui résout ce problème (C'est le même chose que l'algorithme max-sum)
3. Trouvez arg max λ P(O|λ) . L'algorithme de « **Baum-Welch** » résout ce problème (C'est simplement l'approche d'**EM**)

Rappel - les modèles (simple) analogues : génératif, conjoint et conditionnel



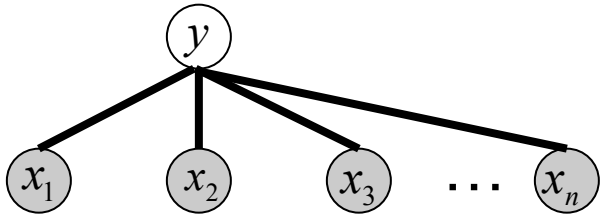
- Modèle génératif

$$P(y,x) = P(y)\prod_{i=1}^{N} P(x_i \mid y)$$



- Modèle conjoint

$$P(y,x) = \frac{1}{Z}\phi(y)\prod_{i=1}^{N} \phi(x_i, y)$$

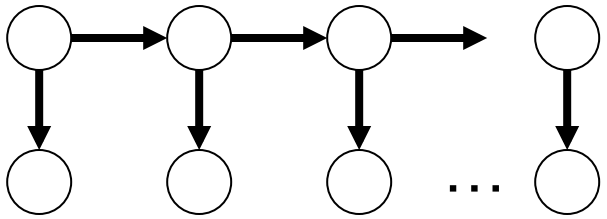


- Modèle conditionnel

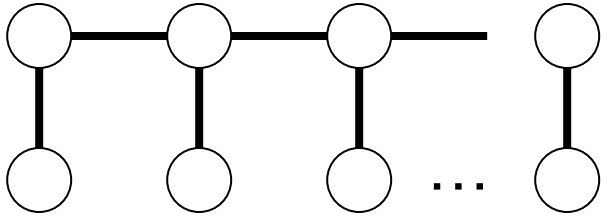
$$P(y \mid x) = \frac{1}{Z(\mathbf{x})}\phi(y)\prod_{i=1}^{N} \phi(x_i, y)$$

Modèles génératifs, conjoints et conditionnels

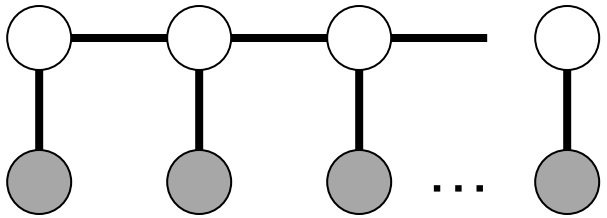
- Modèle de Markov caché (MMC)
  Hidden Markov model (HMM)



- Champ aléatoire
  « Markov Random Field » (MRF)



- Champ aléatoire conditionnel
  « conditional random field »
  (CRF)



Conditional random fields (CRFs)

- Recall, a Markov random field factorizes the *joint distribution* for X using an exponentiated energy function *F*(*X*):

$$P(X) = \frac{1}{Z}\exp(-F(X)), \qquad Z = \sum_X \exp(-F(X)),$$

- Conditional random fields condition on some observations X, yielding a *conditional distribution*

$$P(Y \mid X) = \frac{1}{Z(X)}\exp(-F(Y,X)), \qquad Z(X) = \sum_Y \exp(-F(Y,X)),$$

CRF Energy Function

- Both Markov and conditional random fields can be defined for general model structures, but the energy functions usually include just one or two variables—unary and pairwise potentials.
- Conceptually, to create a conditional random field for $P(Y|X)$ based on U unary and V pairwise functions of variables in Y, the energy function takes the form

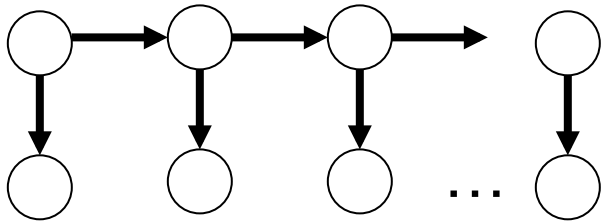
$$F(Y,\tilde{X}) = \sum_{u=1}^{U} U(Y_u,\tilde{X}) + \sum_{v=1}^{V} V(Y_v,\tilde{X})$$

$$P(\mathrm{Y}|\tilde{\mathrm{X}}) = \frac{1}{Z(\tilde{\mathrm{X}})}\exp\left[\sum_{u=1}^{U} U(\mathrm{Y}_u,\tilde{X}) + \sum_{v=1}^{V} V(\mathrm{Y}_v,\tilde{X})\right]$$

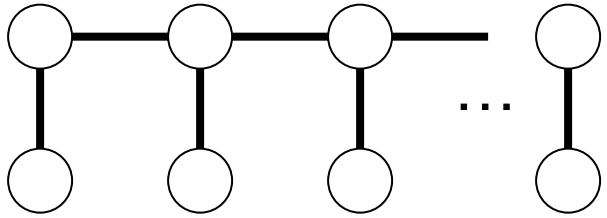
$$= \frac{1}{Z(\tilde{\mathrm{X}})}\prod_{u=1}^{U}\phi_u(\mathrm{Y}_u,\tilde{\mathrm{X}})\prod_{v=1}^{V}\Psi_v(\mathrm{Y}_v,\tilde{\mathrm{X}}).$$

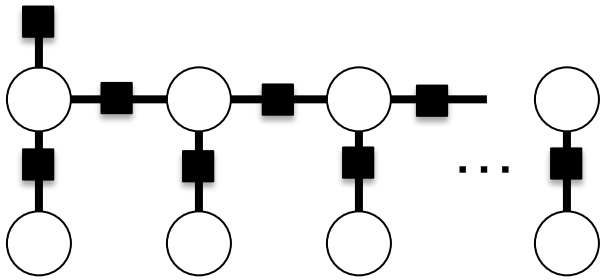
HMMs, MRFs, CRFs and Factor Graphs

Hidden Markov model

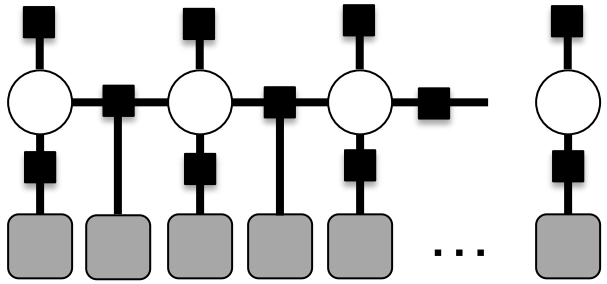


Markov random field



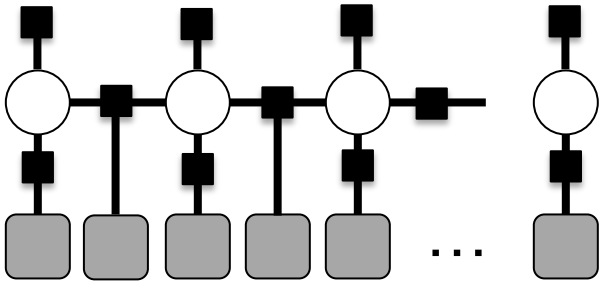


Factor graph for an HMM



Conditional random field

Linear chain CRFs



- For a given sequence of length *N* define

$$P(\mathrm{Y}|\tilde{\mathrm{X}}) = \frac{1}{Z(\tilde{\mathrm{X}})}\prod_{u=1}^{N}\phi_u(\mathrm{y}_u,\tilde{\mathrm{X}})\prod_{v=1}^{N}\Psi_v(\mathrm{y}_v,y_{v+1},\tilde{\mathrm{X}})$$

$$\phi_i[\mathrm{y}_\mathrm{i}] = \exp\left[\sum_{\mathrm{j}=1}^{\mathrm{J}}\theta_j^u u_\mathrm{j}(\mathrm{y}_\mathrm{i},\tilde{\mathrm{X}},i)\right] \quad \Psi_i[\mathrm{y}_\mathrm{i},\mathrm{y}_{\mathrm{i}+1}] = \exp\left[\sum_{\mathrm{k}=1}^{\mathrm{K}}\theta_k^v v_\mathrm{k}(\mathrm{y}_\mathrm{i},\mathrm{y}_{\mathrm{i}+1},\tilde{\mathrm{X}},i)\right]$$

- There are two types of features: a set of *J single variable* (state) features $u_j(y_\mathrm{i},\mathrm{X},i)$ that are a function of a single y_i, and which are computed for each y_i in the sequence i=1…N; and a set of *K pairwise* (transition) features $v_k(y_{\mathrm{i}-1},y_i,\mathrm{X},i)$ for i>1.
- Each type has its own associated unary weights $\theta_\mathrm{j}^\mathrm{u}$ and pairwise weights $\theta_\mathrm{k}^\mathrm{v}$

The global feature vector view for CRFs

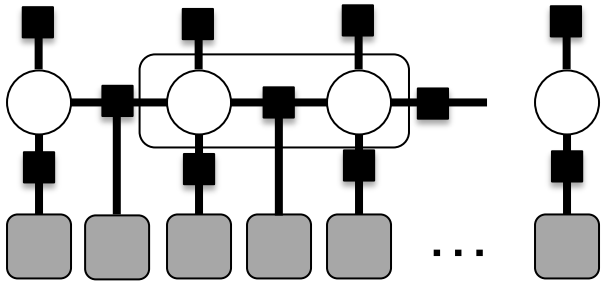
- Features can be a function of the entire observed sequence
- The global dependence on the input is a major advantage of conditional random fields over HMMs
- Can be useful to combine unary and pairwise features and their parameters to work with all features for a given position *i*
- Define $\mathbf{f}(y_i, y_{i+1}, \mathrm{X}, i)$ as a length-*L* vector containing all single-variable and pairwise features & define the global feature vector to be the sum over position dependent feature vectors so that

$$\mathbf{g}(\mathrm{Y},\mathrm{X}) = \sum_{i=1}^{N} \mathbf{f}(\mathrm{y}_i, \mathrm{y}_{i+1}, \mathrm{X}, i) \qquad P(\mathrm{Y}|\mathrm{X}) = \frac{\exp\left(\boldsymbol{\theta}^{\mathrm{T}} \mathbf{g}(\mathrm{Y},\mathrm{X})\right)}{\sum_{Y} \exp\left(\boldsymbol{\theta}^{\mathrm{T}} \mathbf{g}(\mathrm{Y},\mathrm{X})\right)}$$

- The gradient has similar form to logistic regression

$$\frac{\partial}{\partial \boldsymbol{\theta}} \log P(\{\tilde{\mathrm{Y}}_1,\ldots,\tilde{\mathrm{Y}}_{\mathrm{M}}\} \mid \{\tilde{\mathrm{X}}_1,\ldots,\tilde{\mathrm{X}}_M\}) = \sum_{m=1}^{M} \left[\mathbf{g}(\tilde{\mathrm{Y}}_m, \tilde{\mathrm{X}}_{\mathrm{m}}) - \mathrm{E}_m \left[\mathbf{g}(\mathrm{Y}_m, \tilde{\mathrm{X}}_{\mathrm{m}}) \mid P(\mathrm{Y}_m \mid \tilde{\mathrm{X}}_{\mathrm{m}}) \right] \right]$$

Computing gradients



- A linear chain CRF with L_2 regularization

$$\log P(\mathrm{B}\mid \mathrm{A}) = \sum_{m=1}^{M}\sum_{i=1}^{N_m}\sum_{l=1}^{L}\theta_l g_l(\tilde{y}_{m,i},\tilde{y}_{m,i+1},\tilde{\mathrm{X}}_{\mathrm{m}}) - \sum_{\mathrm{m}=1}^{\mathrm{M}}\log Z(\tilde{\mathrm{X}}_{\mathrm{m}}) - \sum_{l=1}^{L}\frac{\theta_l^2}{2\sigma^2}.$$

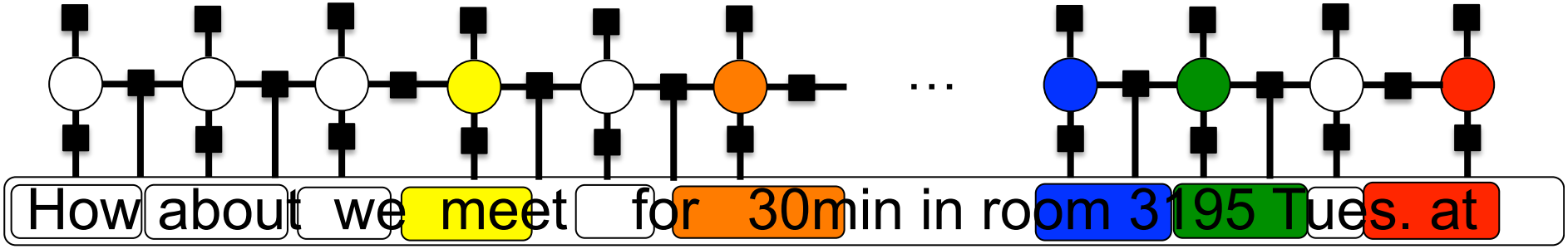
- The contribution to the gradient of each example is

$$\frac{\partial}{\partial\theta_l}\log P(\tilde{\mathrm{Y}}_{\mathrm{m}}\mid\tilde{\mathrm{X}}_{\mathrm{m}}) = \sum_{i=1}^{N_m} g_l(\tilde{y}_i,\tilde{y}_{i+1},\tilde{\mathrm{X}}) - \sum_{i=1}^{N_m}\sum_{y_i}\sum_{y_{i+1}} g_l(y_i,y_{i+1},\tilde{\mathrm{X}})P(y_i,y_{i+1}\mid\tilde{\mathrm{X}}) - \frac{\theta_l}{\sigma^2}.$$

- This is the difference between the observed occurrence of the feature and its expectation under the current prediction of the model, taken with respect to $P(y_i, y_{i+1}\mid\tilde{\mathrm{X}})$, (marginal conditionals) - minus regularization terms
- For unary functions the expectation is w.r.t. $P(y_i\mid\tilde{\mathrm{X}})$
- These are the single- and pairwise-variable marginal conditional distributions, efficiently obtained with the sum-product algorithm

Example: CRFs for information extraction

- Idea: label each word with a *named entity* so as to populate a database record with extracted information
- Consider the following CRF for the line of text

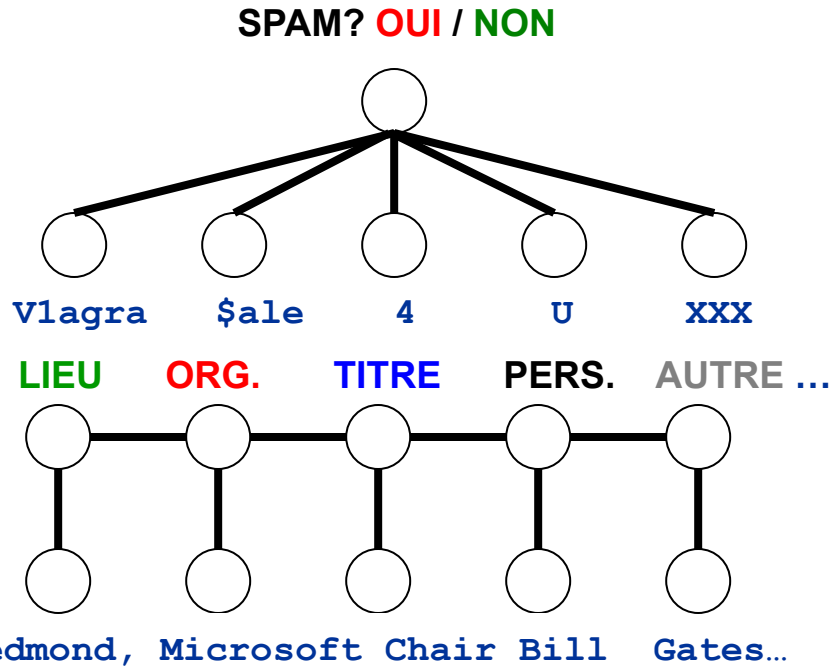


Meeting Record	
Date	Tuesday, March 1, 2016
Time	12:00pm
Location	3195
Duration	unknown

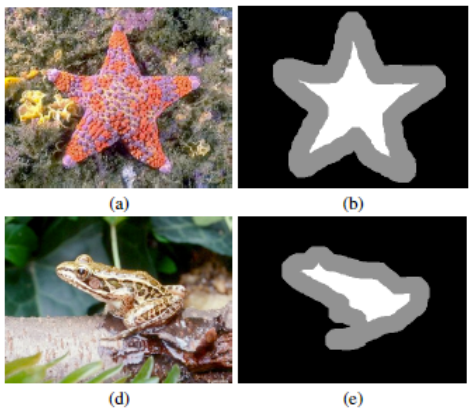
- Each word is tagged with one of the named entities using features computed locally and globally from the sequence

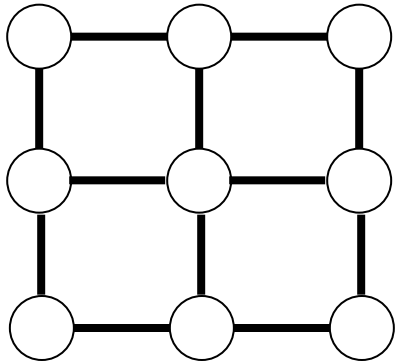
Modèles de probabilité

- Classificateurs : ex. documents et problèmes généraux

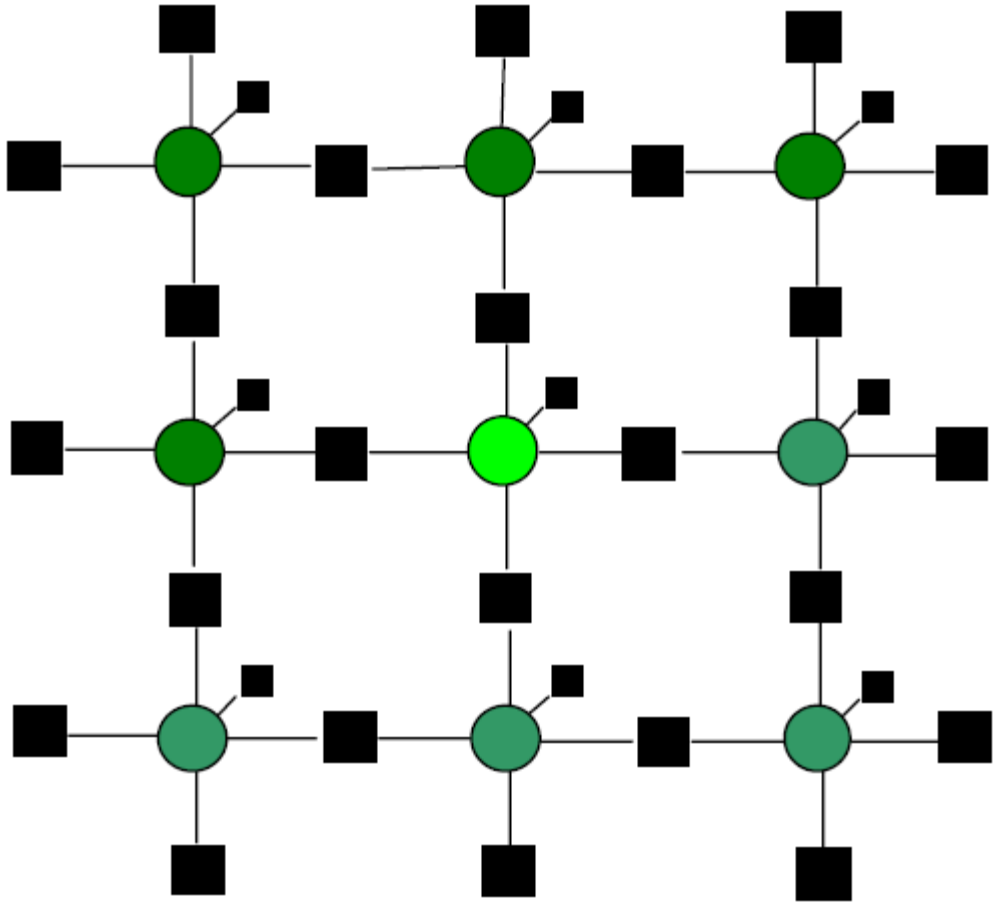


- Séquences : ex. texte, parole, extraction et reconnaissance
- Images : ex. Segmentation et l'estimation de profondeur par imagerie stéréo





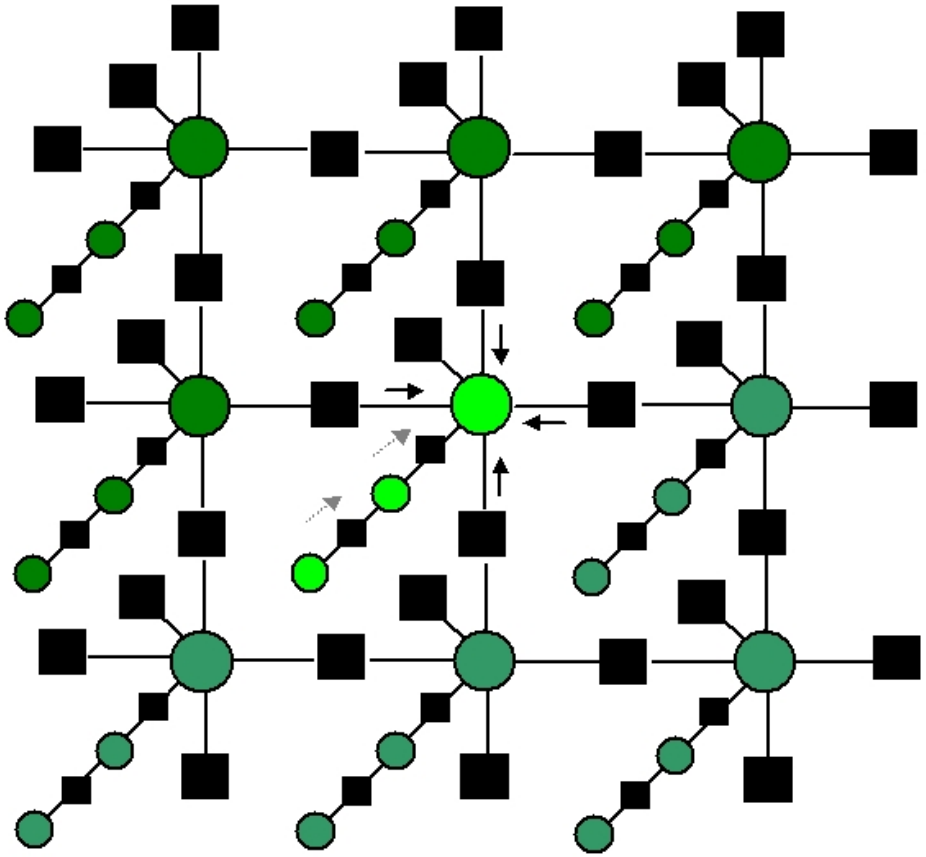
Champs aléatoires pour les images



From: Pal

- Modèles non-oriente

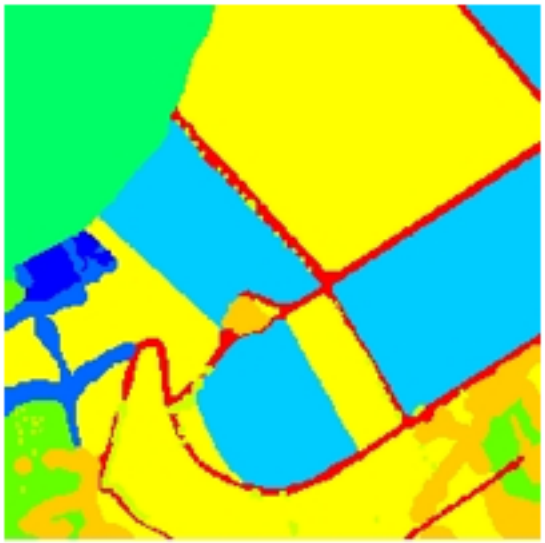
Markov Random Fields



From: Pal

- Widely applicable, ‘undirected’ graphical models

Markov Random Fields



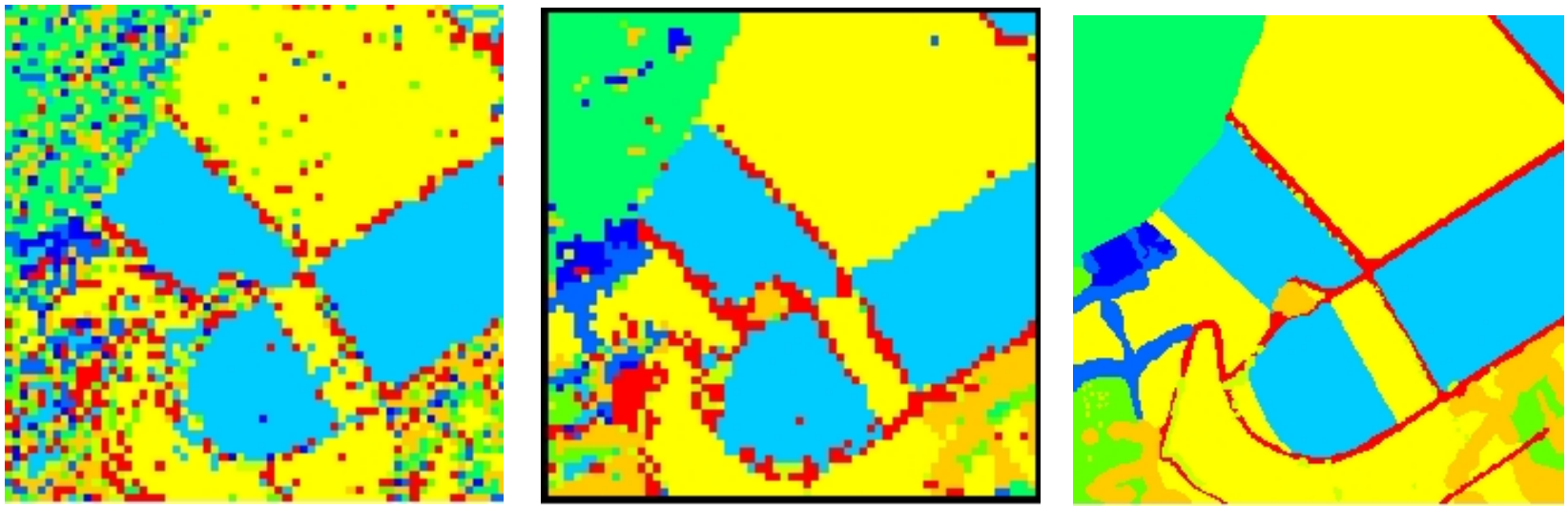
Color	Interpretation
	Field Boundaries
	Enclosed Grass Area
	Deciduous Forest
	Small Sparse Trees
	Deciduous Tree Line
	Open Field
	Dense Coniferous
	Mixed Tree line
	Crops

Figure 1. Aerial imagery and an associated coarse scale hand classification.

From: Pal

- Widely applicable, 'undirected' graphical models

Markov Random Fields



From: Pal

- Widely applicable, ‘undirected’ graphical models