INF8225 – Lesson 2

Plan de cours

- Les concepts en probabilité essentiels de base
- Les réseaux de Bayes
- Dépendance conditionnelle, indépendance conditionnelle, la couverture de Markov
- Les graphes de facteurs, l'algorithme de sum-produit et max-produit
- Le calcul de distributions marginales versus l'explication le plus probable

Notation

- La distribution conjointe est la probabilité d'une conjonction d'affectations particulières à chaque variable comme:

$$P(V_1 = v_1, V_2 = v_2, \ldots, V_n = v_n)$$

- Souvent nous employons la notation:

a) $P(V_1, V_2, \ldots, V_n)$ et b) $P(v_1, v_2, \ldots, v_n)$

avec les variables discrètes ou binaires pour indiquer : a) les tables de probabilités (multidimensionnel) et/ou b) la probabilité unique associée à une configuration particulière

Les concepts essentiels de base

- **Règle du produit** (Règle fondamental)

$$P(A,B) = P(A \mid B)P(B)$$

- **Règle de Bayes**

$$P(B \mid A) = \frac{P(A \mid B)P(B)}{P(A)}$$

- **Règle de la somme** (Marginalisation)

$$P(X_1) = \sum_{\{X\}\backslash X_1} P(X_1, X_2, \ldots, X_N)$$

- Notez : pour obtenir une probabilité conditionnelle on peut utiliser le concept de conditionnement « **conditioning** »

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

Définition: Un réseau de Bayes

- Il représente une factorisation d'une probabilité jointe des variables V_i de la forme:

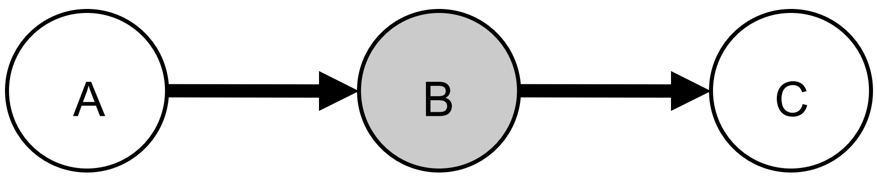
$$P(V_1, V_2, \ldots, V_n) = \prod_{i=1}^{n} P(V_i \mid \text{parents}(V_i))$$

$$\text{si} \qquad \text{parents}(V_i) = \emptyset \quad \text{(il n'a pas de parents)}$$

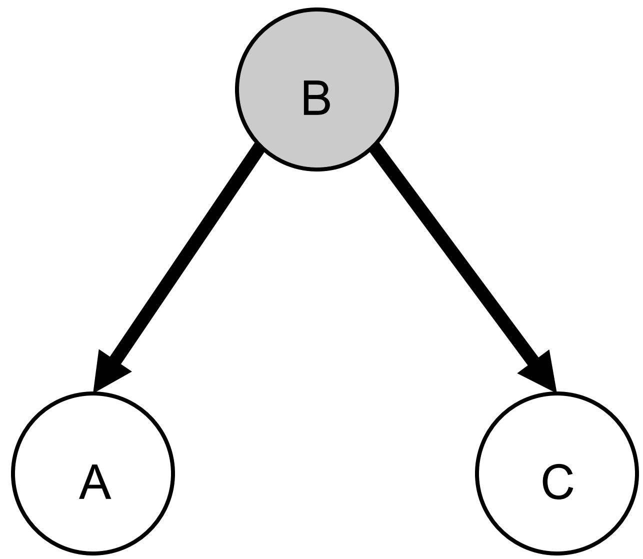
$$\text{alors,} \quad P(V_i \mid \emptyset) = P(V_i)$$

- Pour écrire le réseau: il y a un nœud (cercle) pour chaque variable et une arête orienté entre chaque variable et ses parents.
- Nœud gris implique que la variable est observée avec évidence (exact / dur, ou incertain / douce)

Exemples des modèles ou A et C sont conditionnellement indépendants sachant B

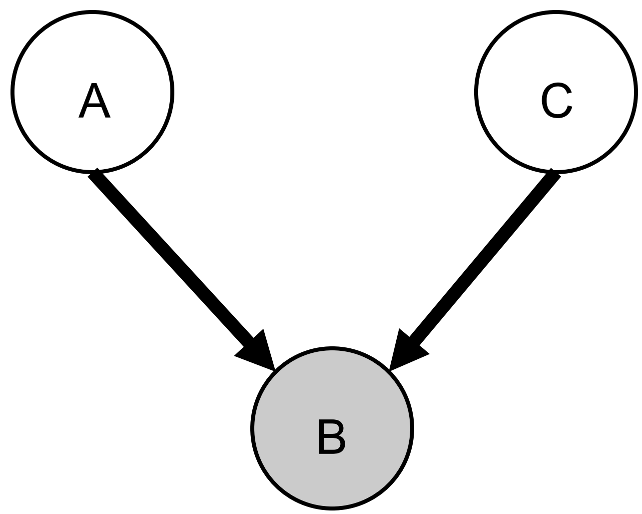


Influence pipelinée



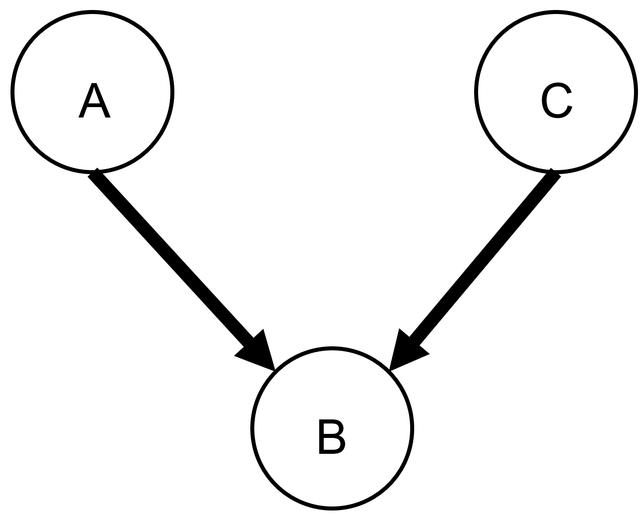
Influence divergente

Exemples d'un modèle ou A et C sont conditionnellement <u>dépendants</u> sachant B

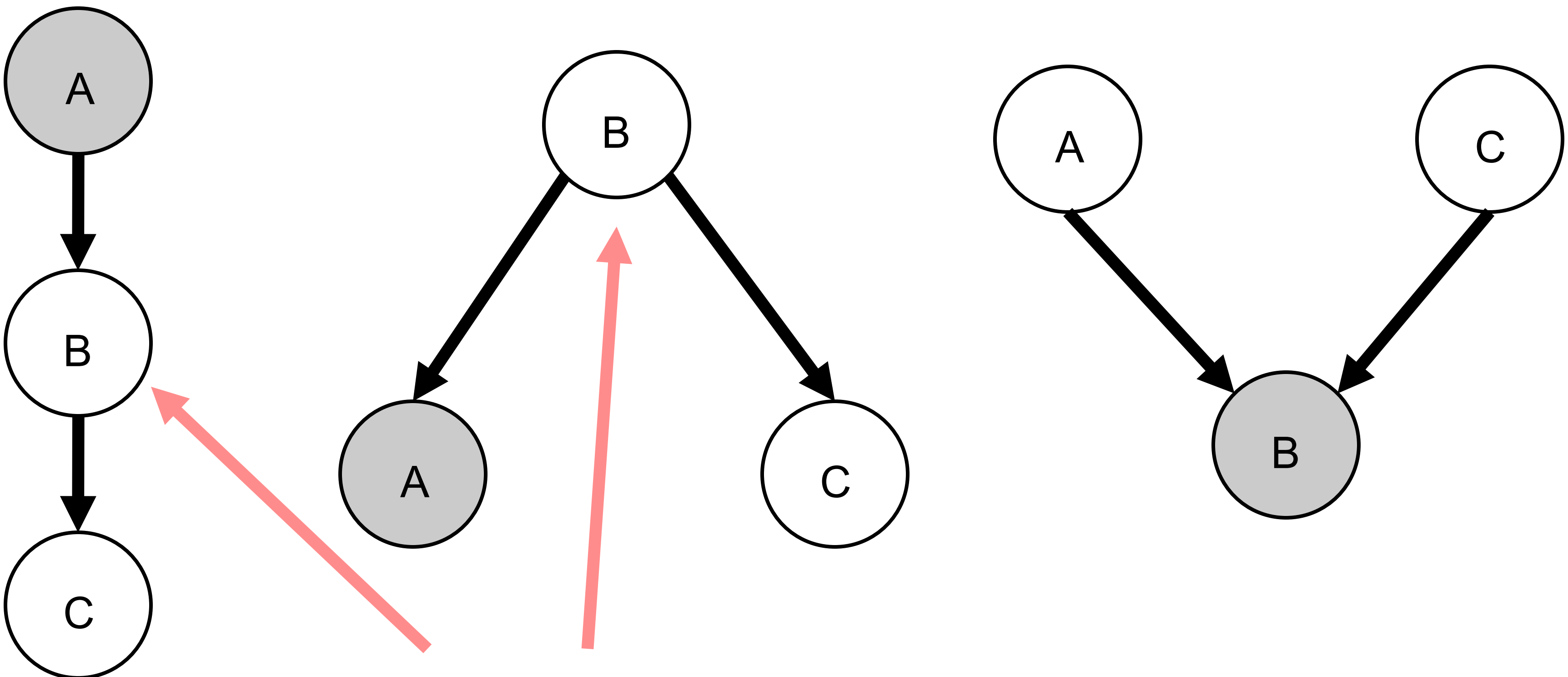


Influence convergente

Par contre, sans évidence A et C sont <u>indépendantes</u>

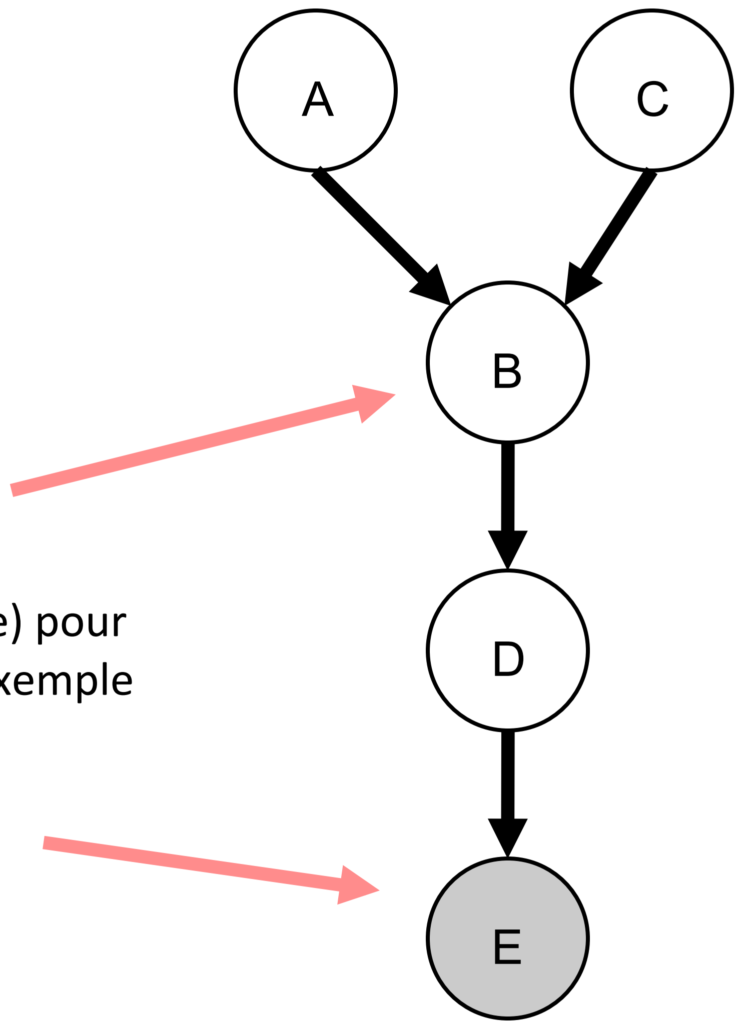


Exemples ou A et C sont dépendants



Sans une observation
pour B

Exemple ou A et C sont dépendants



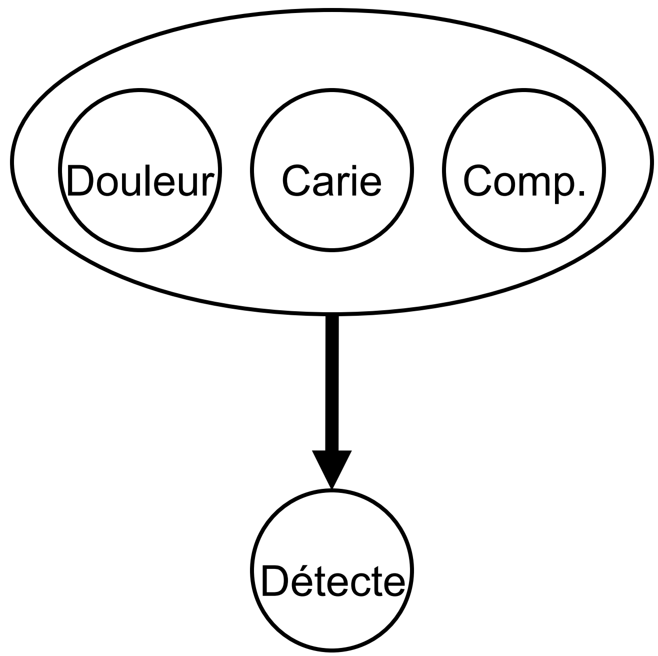
Même sans une observation (directe) pour B nous avons une exemple de dépendance conditionnelle

Exemple - Modélisation

Considérons la modélisation suivant d'un problème simple

P(Détecte,Douleur,Carie,Compétent)

= **P**(Détecte|Douleur,Carie,Compétent) × **P**(Douleur,Carie,Compétent)



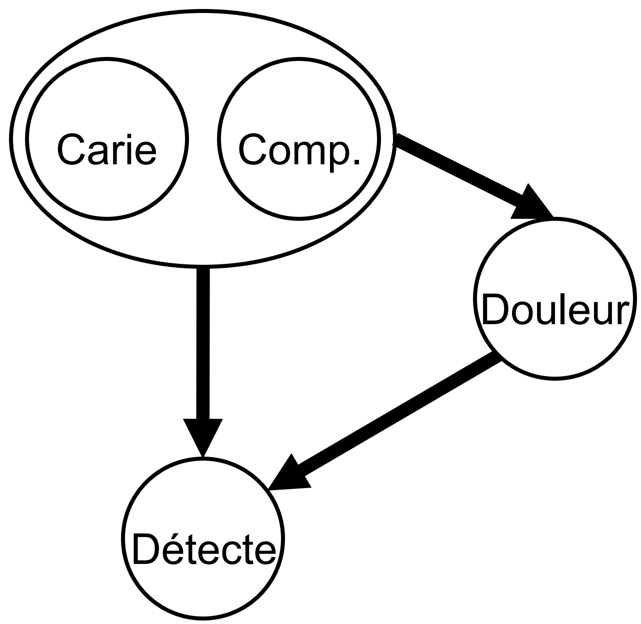
<u>Idée importante</u> : Il est possible d'appliquer nos règles simples récursivement avec les groupes de variables.

Exemple (suite) - Modélisation

P(Détecte,Douleur,Carie,Compétent)

= **P**(Détecte|Douleur,Carie,Compétent) × **P**(Douleur,Carie,Compétent)

= **P**(Détecte|Douleur,Carie,Compétent) × **P**(Douleur|Carie,Compétent) × **P**(Carie,Compétent)



Exemple (suite)

P(Détecte,Douleur,Carie,Compétent)

= **P**(Détecte|Douleur,Carie,Compétent) × **P**(Douleur,Carie,Compétent)

= **P**(Détecte|Douleur,Carie,Compétent) × **P**(Douleur|Carie,Compétent) × **P**(Carie,Compétent)

= **P**(Détecte|Douleur,Carie,Compétent) × **P**(Douleur|Carie,Compétent) × **P**(Carie|Compétent) × **P**(Compétent)

Possible, mais…

Exemple (suite)

P(Détecte,Douleur,Carie,Compétent)

= **P**(Détecte|Douleur,Carie,Compétent) × **P**(Douleur|Carie,Compétent) × **P**(Carie|Compétent) × **P**(Compétent)

= **P**(Détecte|Douleur,Carie,Compétent) × **P**(Douleur|Carie,Compétent) × **P**(Carie) × **P**(Compétent)

La présence de carie ne dépend pas de la compétence du dentiste

Exemple (suite)

$$\mathbf{P}(\text{Détecte},\text{Douleur},\text{Carie},\text{Compétent})$$

$$= \mathbf{P}(\text{Détecte}\,|\,\text{Douleur},\text{Carie},\text{Compétent}) \times \mathbf{P}(\text{Douleur}\,|\,\text{Carie},\text{Compétent}) \times \mathbf{P}(\text{Carie}\,|\,\text{Compétent}) \times \mathbf{P}(\text{Compétent})$$

$$= \mathbf{P}(\text{Détecte}\,|\,\text{Douleur},\text{Carie},\text{Compétent}) \times \mathbf{P}(\text{Douleur}\,|\,\text{Carie},\text{Compétent}) \times \mathbf{P}(\text{Carie}) \times \mathbf{P}(\text{Compétent})$$

$$= \mathbf{P}(\text{Détecte}\,|\,\text{Douleur},\text{Carie},\text{Compétent}) \times \color{red}{\mathbf{P}(\text{Douleur}\,|\,\text{Carie})} \times \mathbf{P}(\text{Carie}) \times \mathbf{P}(\text{Compétent})$$

La présence de douleur de ne dépend pas de la compétence du dentiste

Exemple (suite)

P(Détecte,Douleur,Carie,Compétent)

= **P**(Détecte|Douleur,Carie,Compétent) × **P**(Douleur|Carie,Compétent) × **P**(Carie|Compétent) × **P**(Compétent)

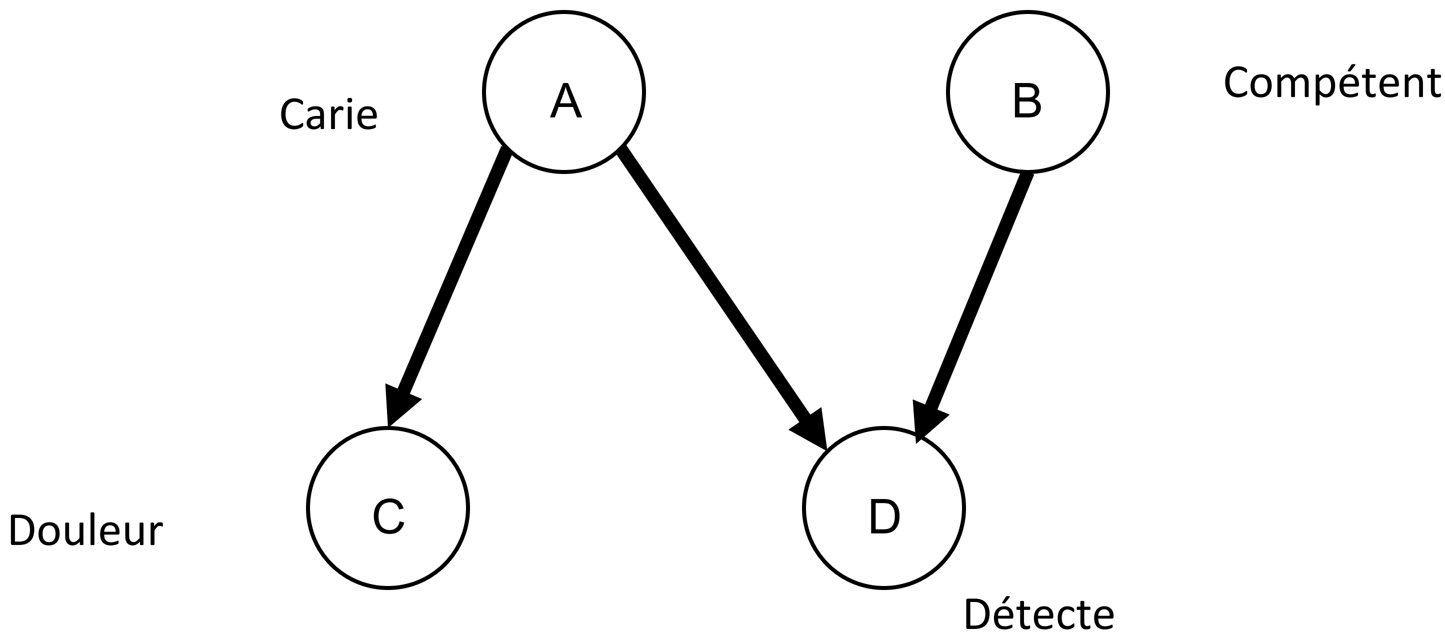
= **P**(Détecte|Douleur,Carie,Compétent) × **P**(Douleur|Carie,Compétent) × **P**(Carie) × **P**(Comp~~étent)~~

= **P(Détecte|Carie,Compétent)** × **P**(Douleur|Carie) × **P**(Carie) × **P**(Compétent)

La détection d'une anomalie ne dépend pas de la présence de douleur

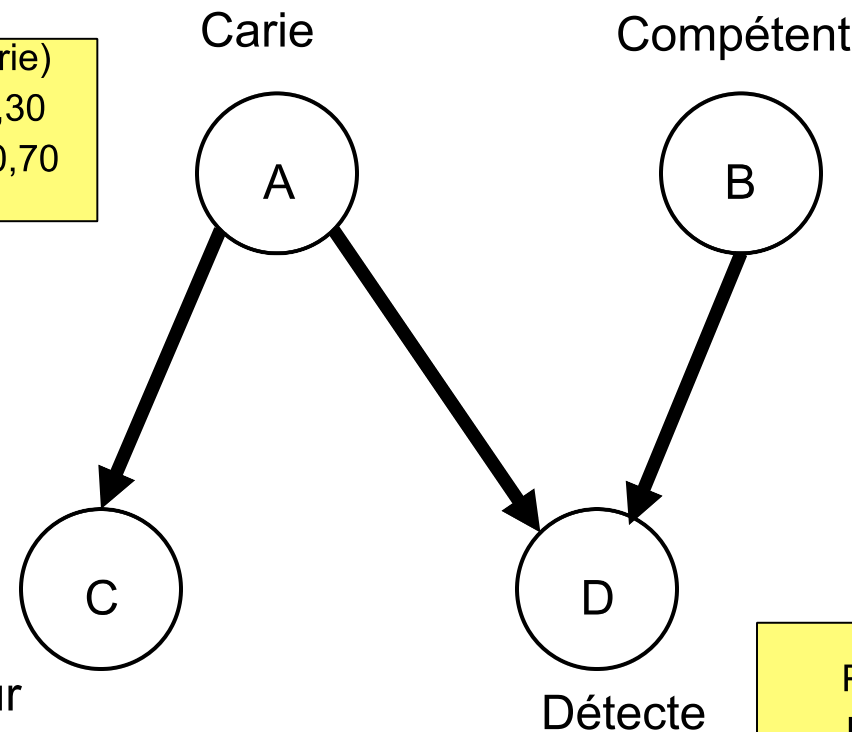
Exemple (suite)

On obtient donc un réseau de Bayes avec la structure suivante :



Exemple (suite)

P(A) = P(Carie)
P(carie) = 0,30
P(¬carie) = 0,70



P(B) = P(Compétent)
P(compétent) = 0,95
P(¬compétent) = 0,05

P(C|A) = P(Douleur | Carie)
P(douleur | carie) = 0,8
P(douleur | ¬carie) = 0,3

P(D | A, B) =
P(Détecte | Carie , Compétent)
P(détecte | compétent ∧ carie) = 0,99
P(détecte | compétent ∧ ¬carie) = 0,05
P(détecte | ¬compétent ∧ carie) = 0,50
P(détecte | ¬compétent ∧ ¬carie) = 0,15

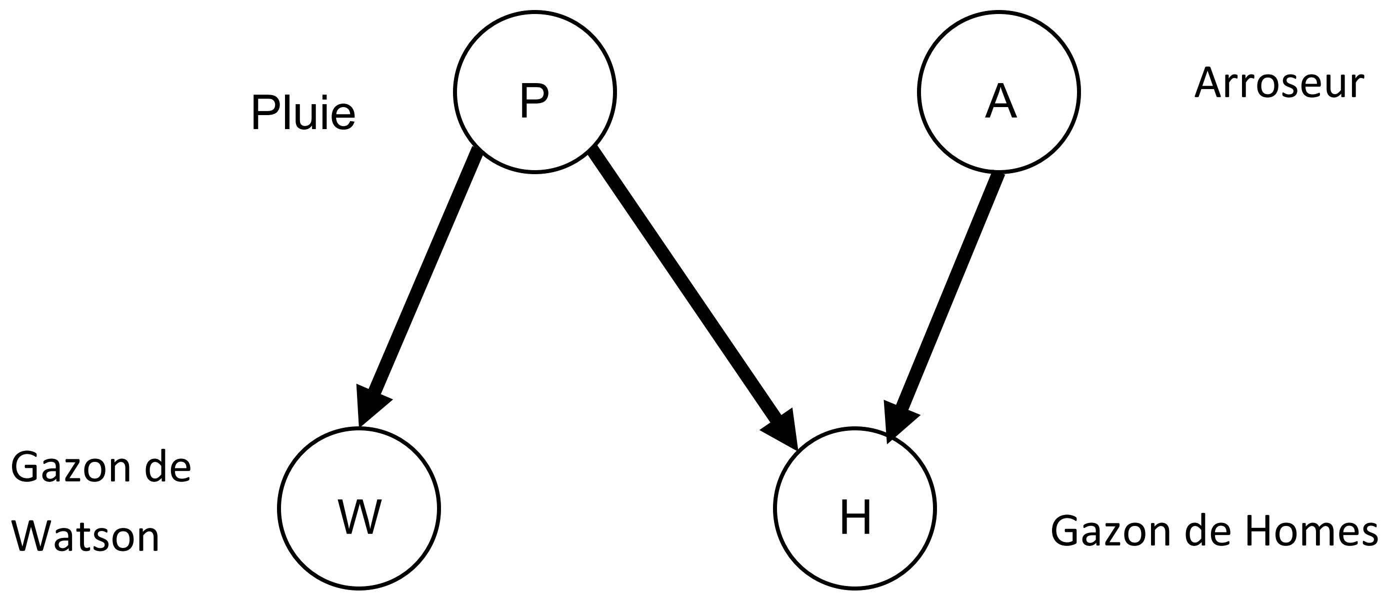
Notre exemple (suite)

- Distribution jointe complète:

[detecte===0, douleur===0, competent===0, carie===0]	0.020825
[detecte===1, douleur===0, competent===0, carie===0]	0.003675
[detecte===0, douleur===1, competent===0, carie===0]	0.008925
[detecte===1, douleur===1, competent===0, carie===0]	0.001575
[detecte===0, douleur===0, competent===1, carie===0]	0.442225
[detecte===1, douleur===0, competent===1, carie===0]	0.023275
[detecte===0, douleur===1, competent===1, carie===0]	0.189525
[detecte===1, douleur===1, competent===1, carie===0]	0.009975
[detecte===0, douleur===0, competent===0, carie===1]	0.0015
[detecte===1, douleur===0, competent===0, carie===1]	0.0015
[detecte===0, douleur===1, competent===0, carie===1]	0.006
[detecte===1, douleur===1, competent===0, carie===1]	0.006
[detecte===0, douleur===0, competent===1, carie===1]	0.00057
[detecte===1, douleur===0, competent===1, carie===1]	0.05643
[detecte===0, douleur===1, competent===1, carie===1]	0.00228
[detecte===1, douleur===1, competent===1, carie===1]	0.22572

Autre exemple avec le même structure (exemple célèbre de « wetgrass »)

On obtient donc un réseau de Bayes avec la structure suivante :



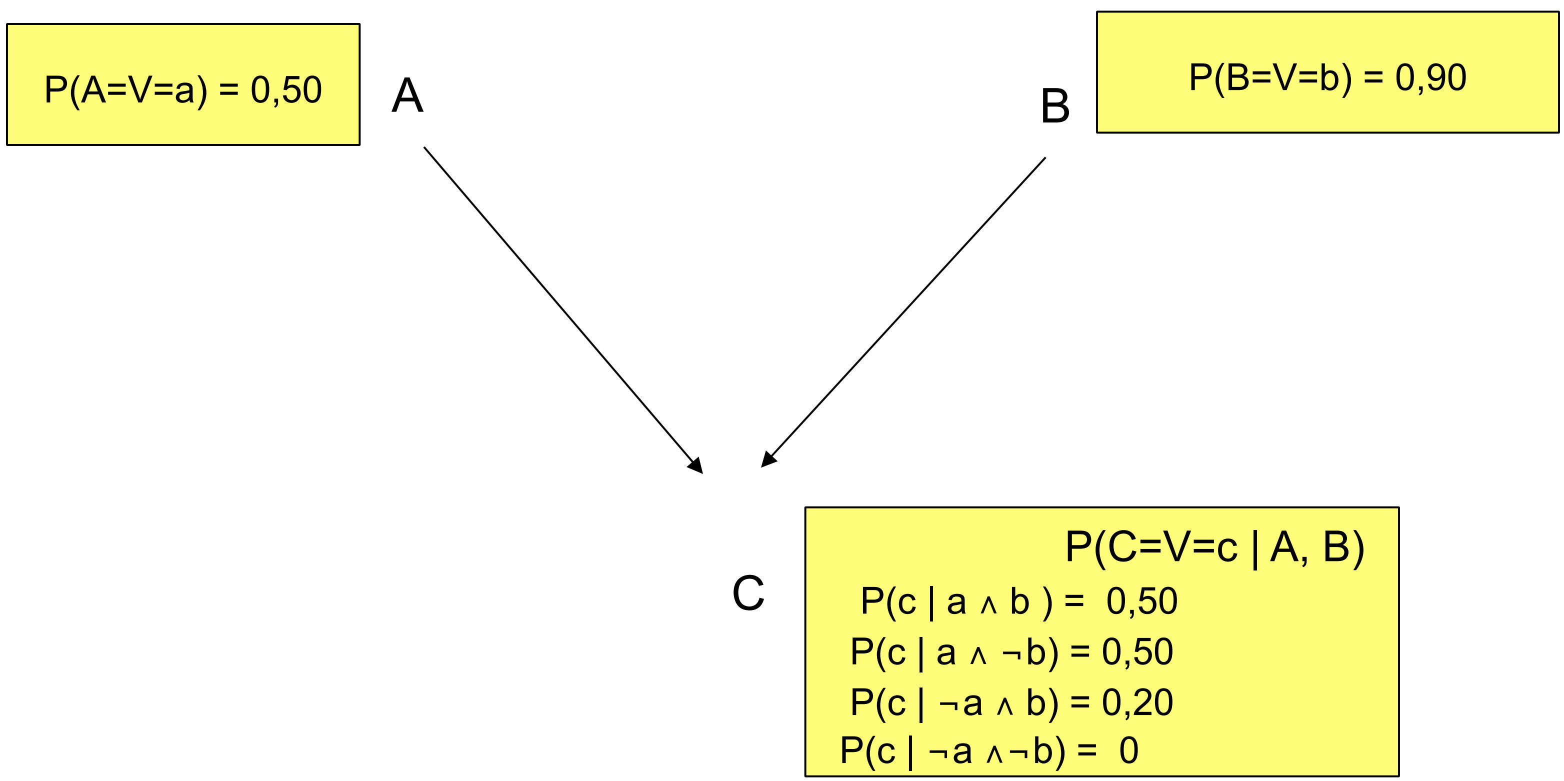
Exemple : « Explaining Away »

Pour un même jeu de données… Plusieurs réseaux possibles

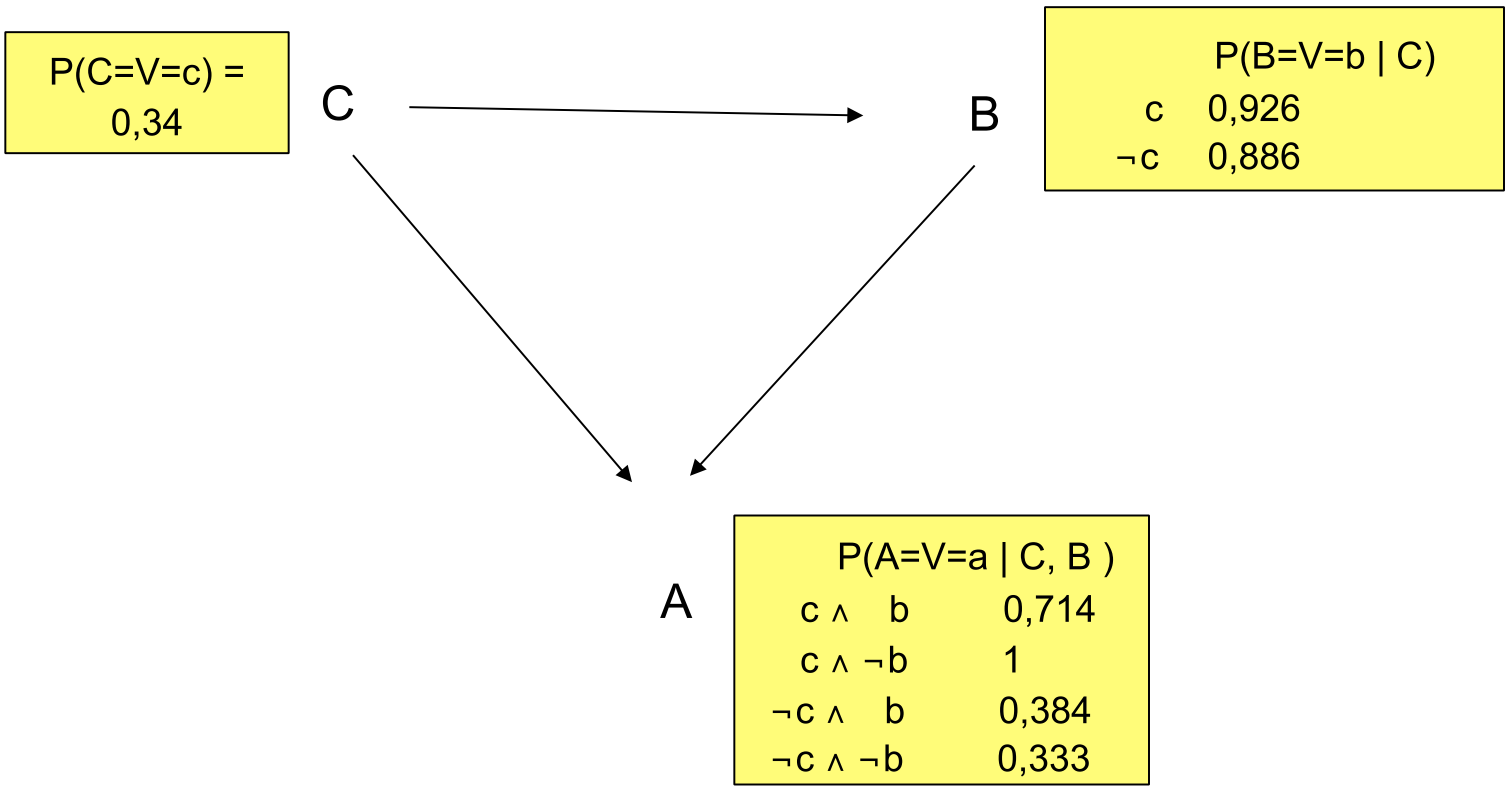
Exemple : Etant donné la distribution jointe complète pour les variables binaires A, B et C,
Rapel – notation : A = Vrai = V = a, A=Faux=F=¬a

$a \wedge b \wedge c$	0.225
$a \wedge b \wedge \neg c$	0.225
$a \wedge \neg b \wedge c$	0.025
$a \wedge \neg b \wedge \neg c$	0.025
$\neg a \wedge b \wedge c$	0.09
$\neg a \wedge b \wedge \neg c$	0.36
$\neg a \wedge \neg b \wedge c$	0
$\neg a \wedge \neg b \wedge \neg c$	0.05

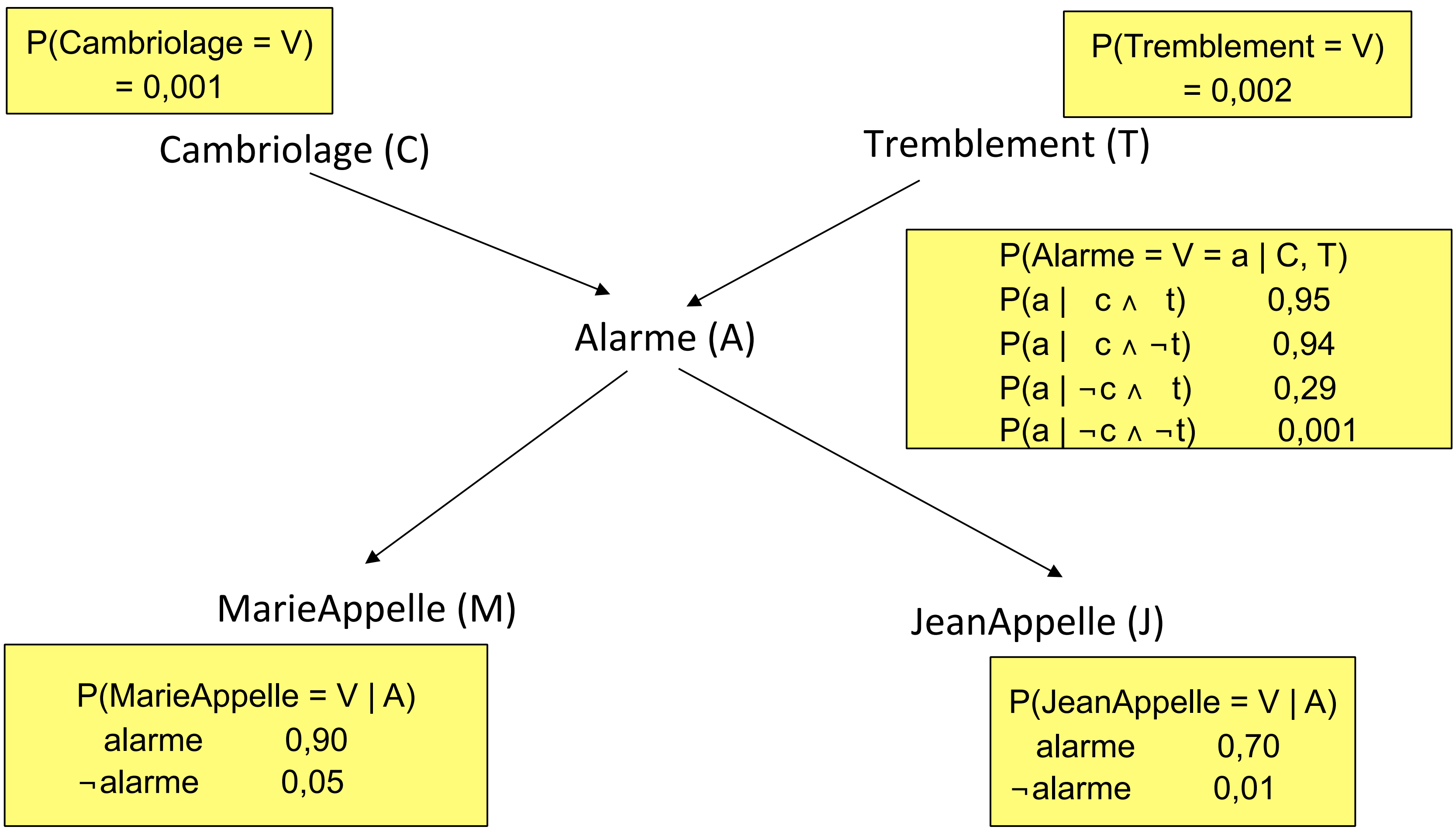
Plusieurs réseaux possibles pour un même jeu de données



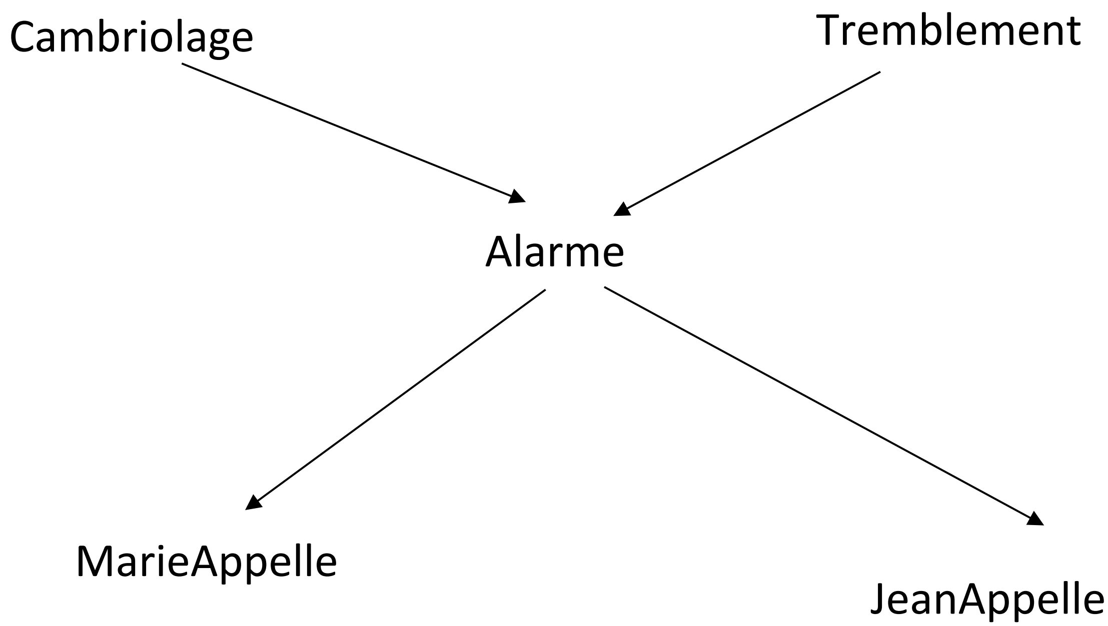
Plusieurs réseaux possibles...



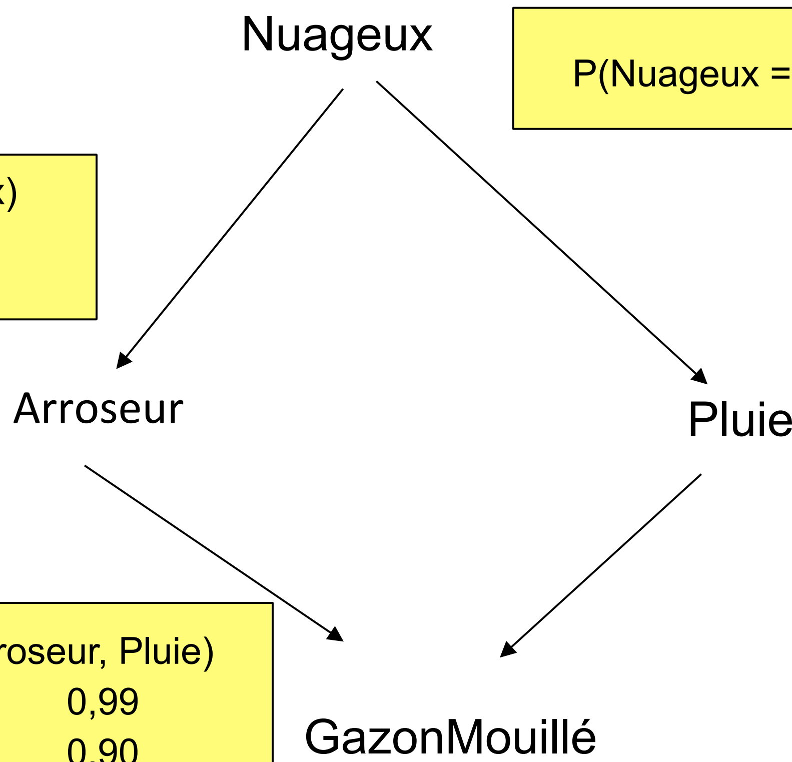
Exemple (célèbre) « earthquake »



Exemple


Cambriolage
Tremblement
Alarme
MarieAppelle
JeanAppelle

Exemple



P(Nuageux = V) = 0,5

P(Arroseur = V | Nuageux)

nuageux	0,10
¬nuageux	0,50

P(Pluie = V | Nuageux)

nuageux	0,80
¬nuageux	0,20

P(GazonMouillé = V | Arroseur, Pluie)

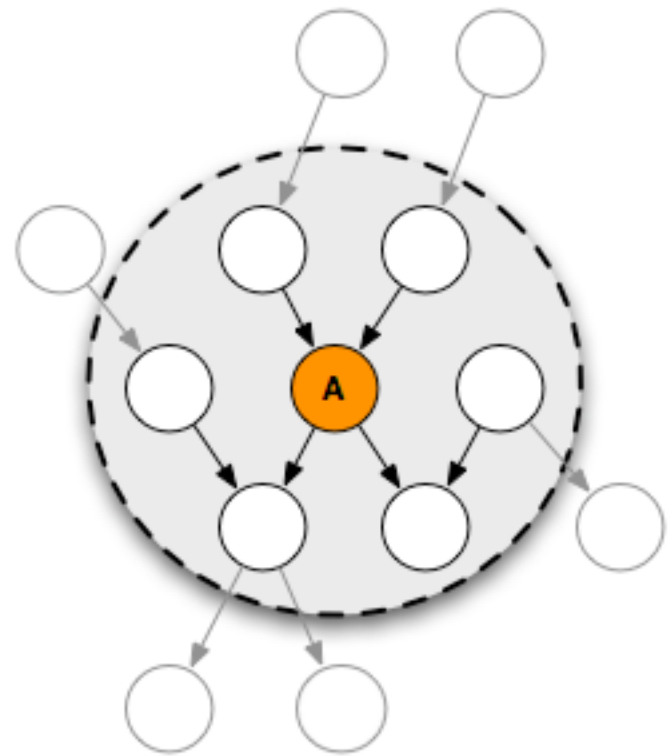
arroseur ∧ pluie	0,99
arroseur ∧ ¬pluie	0,90
¬arroseur ∧ pluie	0,90
¬arroseur ∧ ¬pluie	0,00

Indépendance conditionnelle dans les réseaux bayésiens

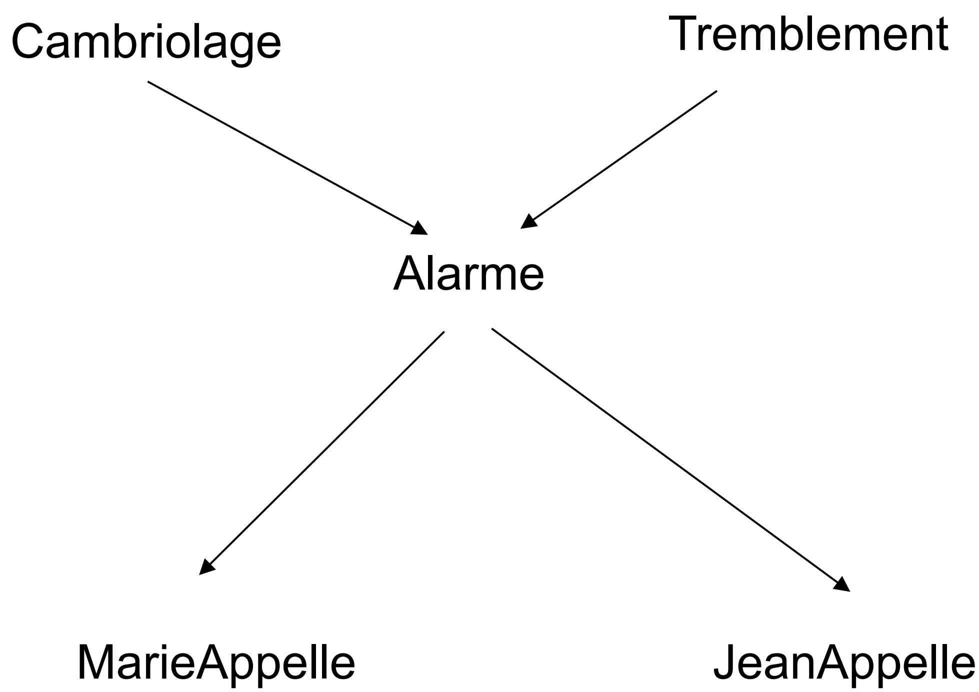
- Un noeud est conditionnellement indépendant de ses non-descendants, étant donné ses parents
- Un noeud est conditionnellement indépendant de tous les autres noeuds du réseau, étant donné sa couverture de Markov (c'est-à-dire ses parents, ses enfants et les parents de ses enfants)

Couverture de Markov

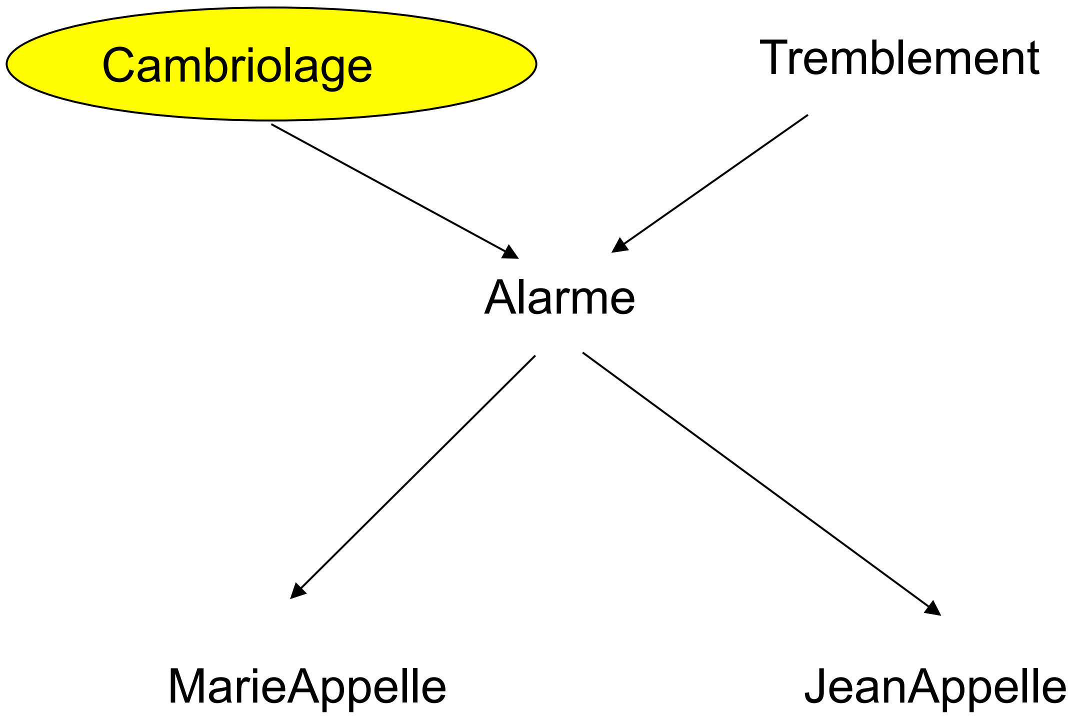
- Etant donné un nœud, la couverture de Markov consiste de ses parents, ses enfants et les autres parents de ses enfants.



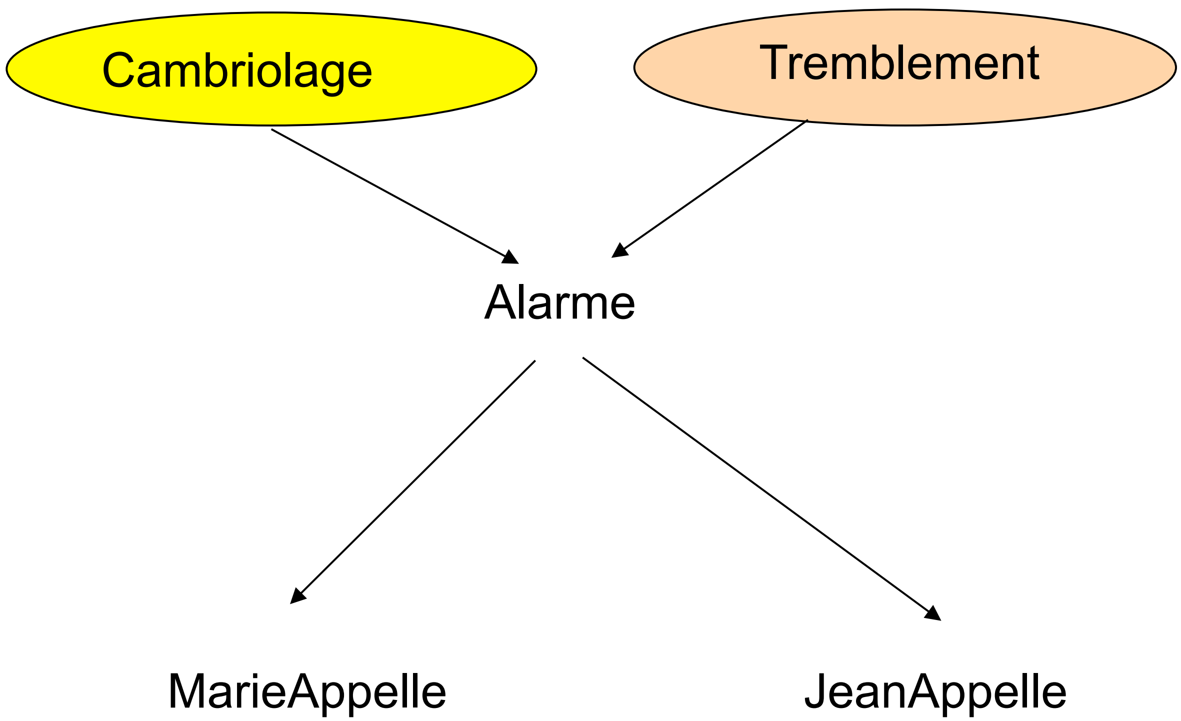
Exemple – indép. conditionnelle



Exemple – indép. conditionnelle

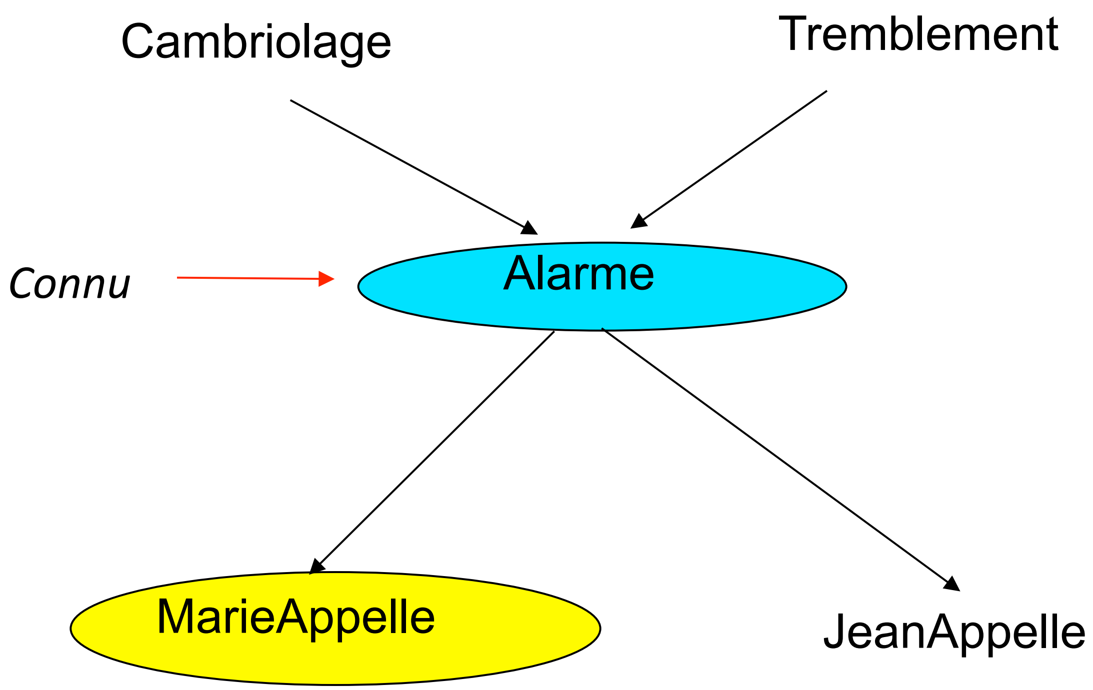


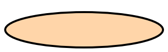
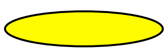
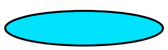
Exemple – indép. conditionnelle



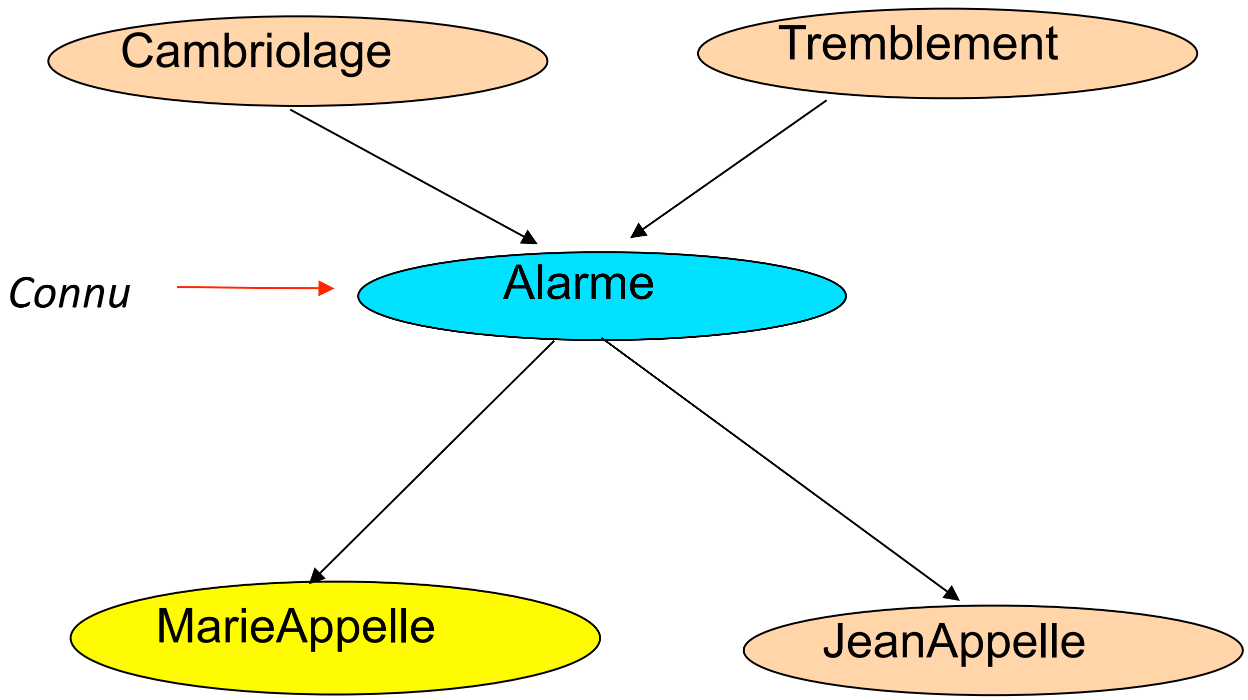
Les noeuds sont indépendants du noeud étant donné le noeud

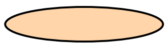
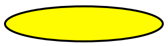
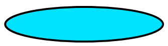
Exemple – indép. conditionnelle



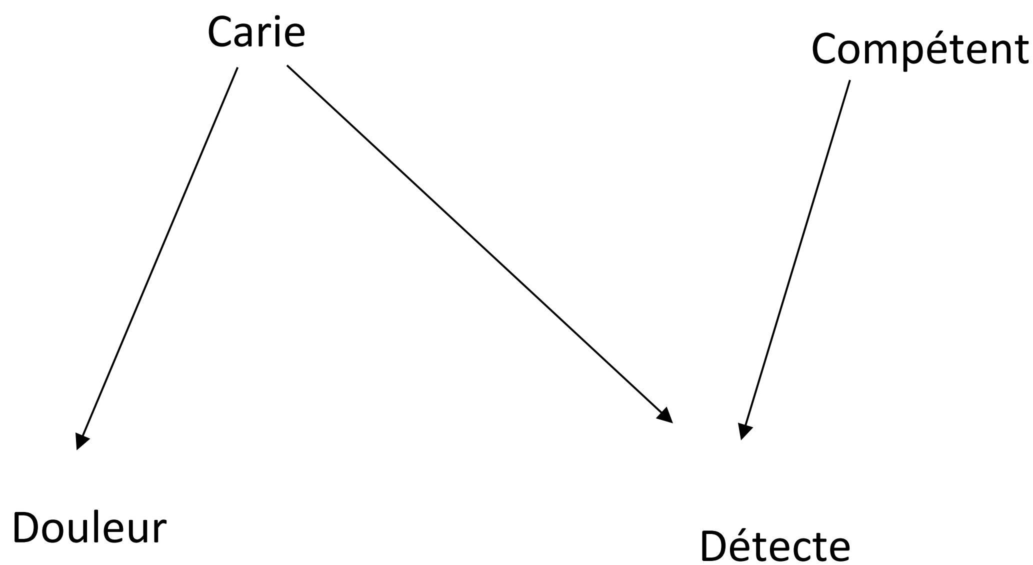
Les noeuds  *sont indépendants du noeud*  *étant donné le noeud* 

Exemple – indép. conditionnelle

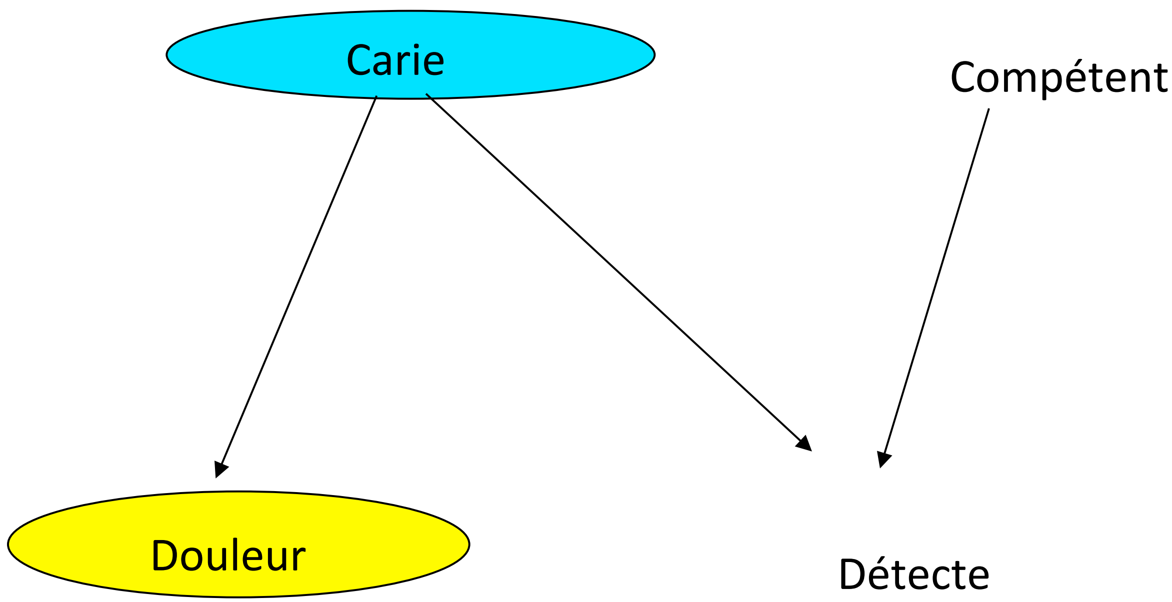


Les noeuds  *sont indépendants du noeud*  *étant donné le noeud* 

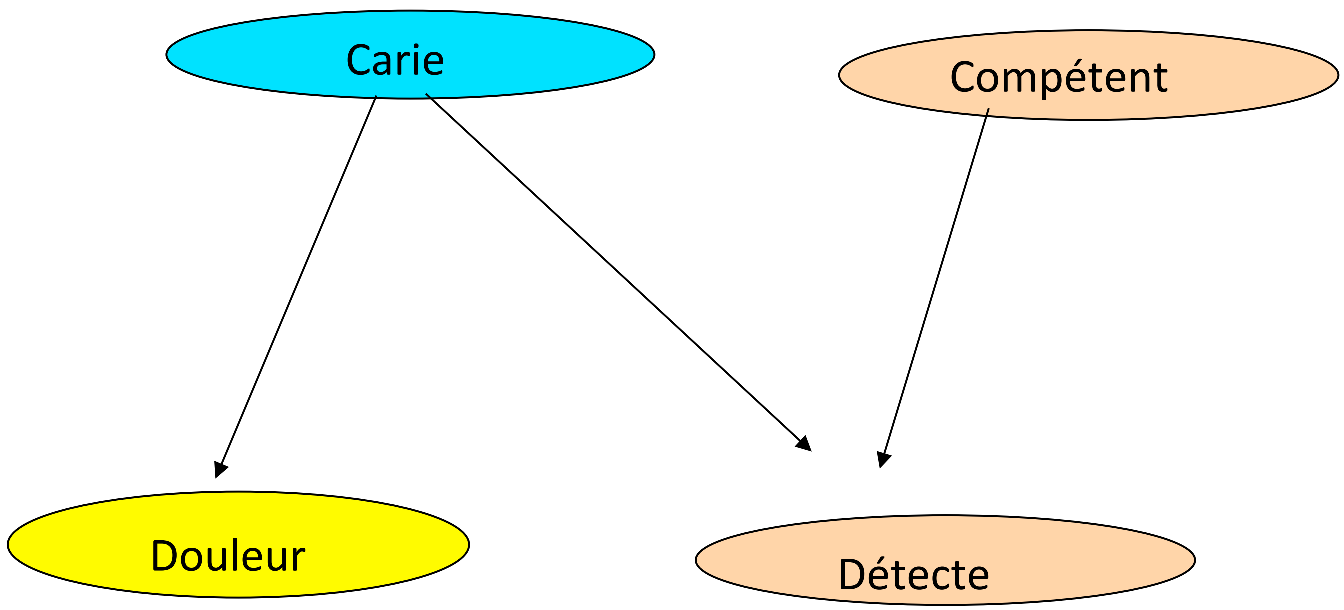
Exemple – indép. conditionnelle



Exemple – indép. conditionnelle

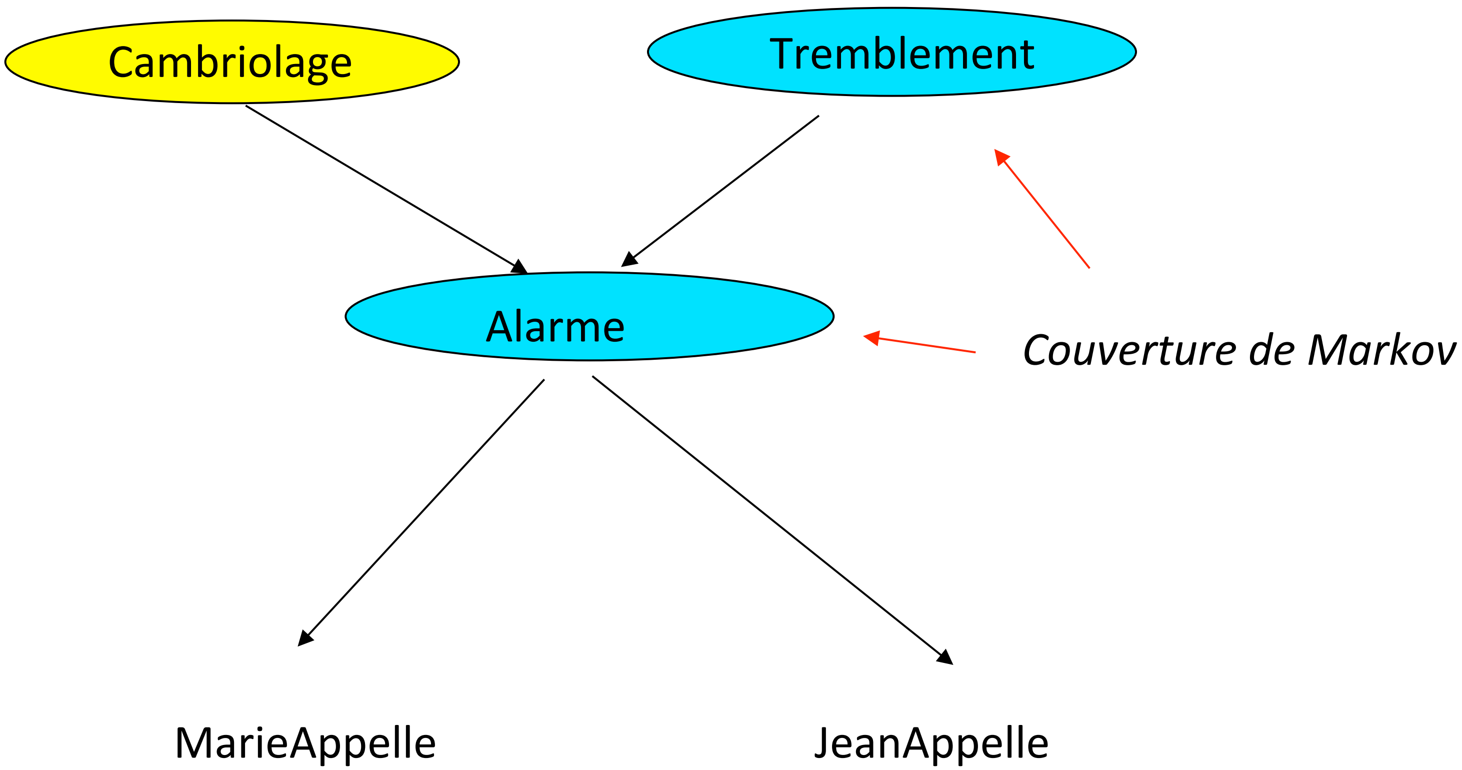


Exemple – indép. conditionnelle

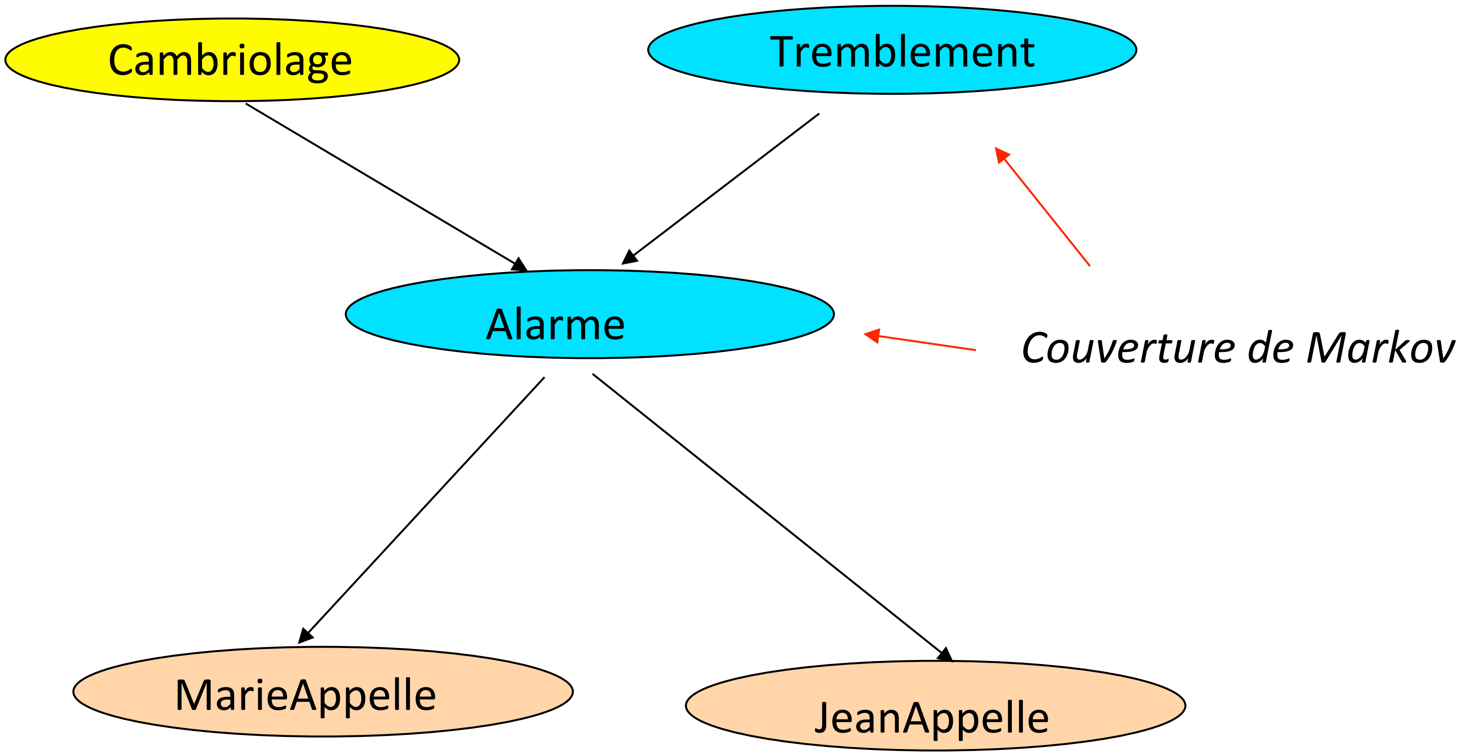


Les noeuds sont indépendants du noeud étant donné le noeud

Exemple – indép. conditionnelle



Exemple – indép. conditionnelle



Les noeuds sont indépendants du noeud étant donné le noeud

Probabilités marginales et l'algorithme de somme-produit

- Dans les graphes avec quelques observations inconnues (ou des variables cachées) nous avons besoin de distributions marginales.
- Pour apprentissage avec des variables inconnues, les algorithmes tels que la maximisation d'espérance (EM) sont typiquement utilisées pour estimer des paramètres
- Ces algorithmes exigeant de nous de calculer des distributions marginales locales
- Donc, dans beaucoup de circonstances importantes on a besoin de marginales

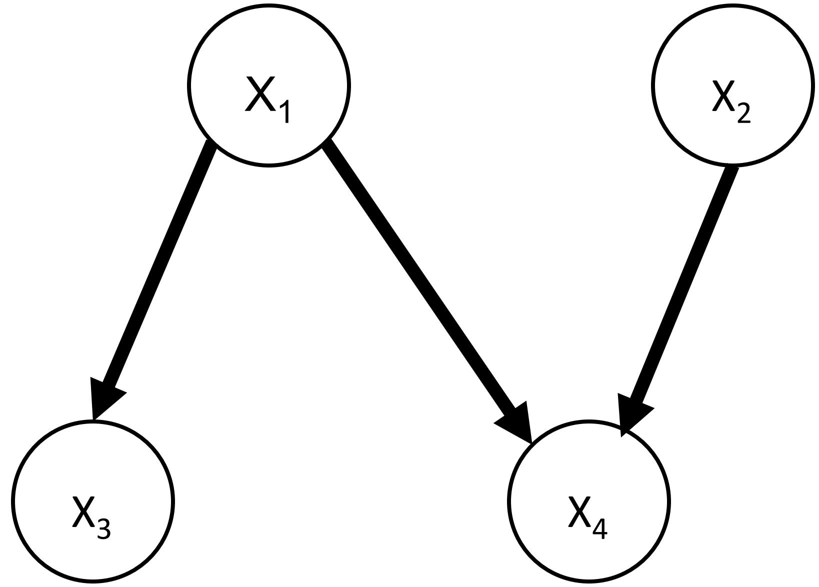
Inférence exacte vs. approximative

- Pour les graphes de facteurs à structure arborescente, nous pouvons calculer toutes les probabilités marginales sans approximation avec l'algorithme de somme-produit (défini avec des graphes de facteurs)
- On parle souvent d'inférence exacte dans un arbre
- Quand il existe un ou plusieurs cycles dans un graphe de facteurs, il est possible d'appliquer l'algorithme de somme-produit, toutefois l'inférence sera une approximation

Calculant la « meilleure configuration » et l'algorithme maximum - produit

- Un marginal est une distribution locale de probabilité dans un graphique après avoir intégré sur d'autres variables
- La meilleure configuration d'un graphe est les meilleures valeurs des variables
- Pour un modèle de probabilité, c'est l'explication la plus probable (l'EPP ou MPE en anglais)
- Pour un réseau bayésien ou un modèle avec une distribution a priori, une vraisemblance et un postérieur, nous référons souvent à la tâche d'inférence maximum a posteriori (MAP) pour : a) des variables aléatoires ou b) pour l'estimation des paramètres en utilisant une probabilité a priori

Considérons



La marginalisation nécessaire pour calculer :

$$P(X_4 = x_4)$$

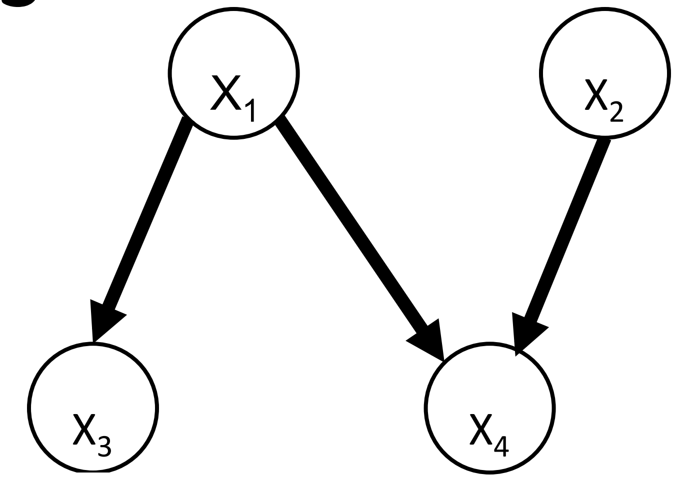
$$= \sum_{x_1} \sum_{x_2} \sum_{x_3} P(X_1 = x_1, X_2 = x_2, X_3 = x_3, X_4 = x_4)$$

$$= \sum_{x_1} \sum_{x_2} \sum_{x_3} P(X_1 = x_1)\ P(X_2 = x_2)\ P(X_3 = x_3 | X_1 = x_1)$$

$$P(X_4 = x_4 | X_1 = x_1, X_2 = x_2)$$

Exercice : Combien multiplications et additions avec des variables binaires?

Considérons



$$
\begin{aligned}
f_1(x_1) &= P(X_1 = x_1) \\
f_2(x_2) &= P(X_2 = x_2) \\
f_3(x_1, x_3) &= P(X_3 = x_3 | X_1 = x_1) \\
f_4(x_1, x_2, x_4) &= P(X_4 = x_4 | X_1 = x_1, X_2 = x_2)
\end{aligned}
$$

$$P(X_4 = x_4) =$$

$$
\begin{aligned}
&= \sum_{x_1} \sum_{x_2} \sum_{x_3} f_1(x_1)\ f_2(x_2)\ f_3(x_1, x_3)\ f_4(x_1, x_2, x_4) \\
&= \sum_{x_1} f_1(x_1) \sum_{x_3} f_3(x_1, x_3) \underbrace{\sum_{x_2} f_2(x_2)\ f_4(x_1, x_2, x_4)}_{g_2(x_1, x_4)} \\
&= \sum_{x_1} f_1(x_1) \left(\sum_{x_2} f_2(x_2)\ f_4(x_1, x_2, x_4) \right) \sum_{x_3} f_3(x_1, x_3)
\end{aligned}
$$

Exercice : Combien multiplications et additions avec des variables binaires?

Rappelez une idée fondamentale : Factor Graphs

Kschischang, Frank R.; Brendan J. Frey and Hans-Andrea Loeliger (2001), "Factor Graphs and the Sum-Product Algorithm", *IEEE Transactions on Information Theory* 47 (2): pp. 498–519.

Définition: Un graphe de facteurs

- Il est possible de créer une fonction à partir des produits de s fonctions f_i de sous-ensembles de variables $\{X\}_i$

$$F(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{s} f_i(\{X\}_i)$$

- Exemple : notre modèle recent, où nous avons

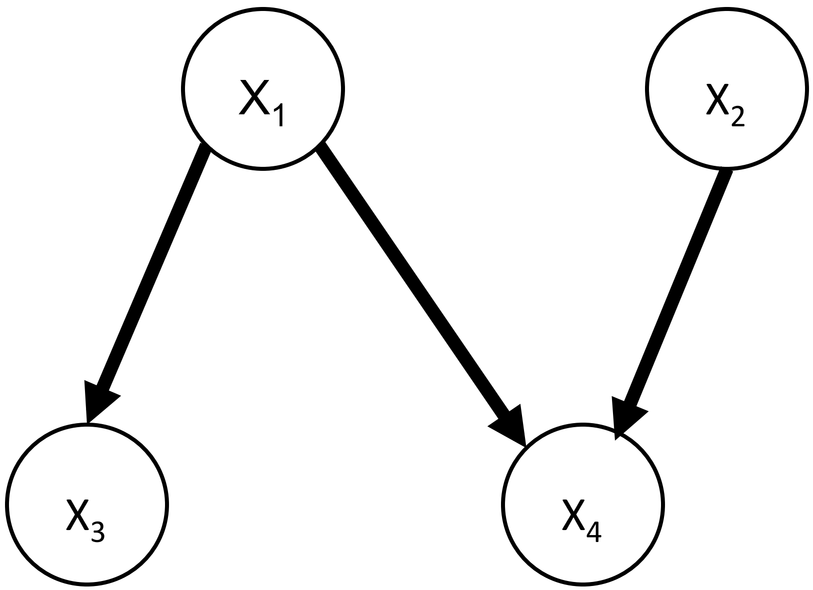
$$\begin{aligned} P(A,B,C,D) &= f_1(\{X\}_1) f_2(\{X\}_2) f_3(\{X\}_3) f_4(\{X\}_4) \\ &= P(C \mid A) P(D \mid A,B) P(A) P(B) \end{aligned}$$

Un graphe de facteurs

Il se compose de (simplement) des :

- Rectangles pour chaque fonction, et des
- Liens, sans orientation entre chaque fonction et chaque variable dans la fonction

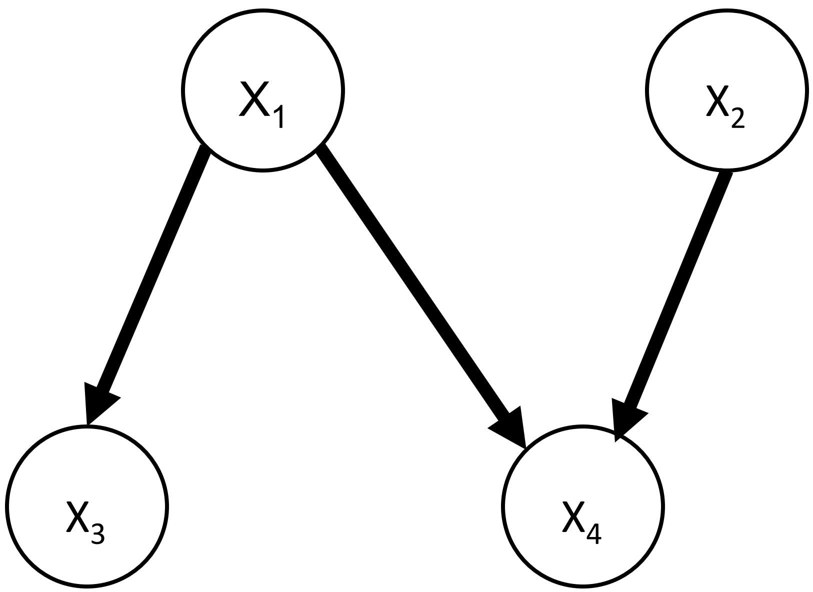
Considérons



Exercice : Ecrivez un graphe de facteurs.

Exercice :

Écrivez un graphe de facteurs.

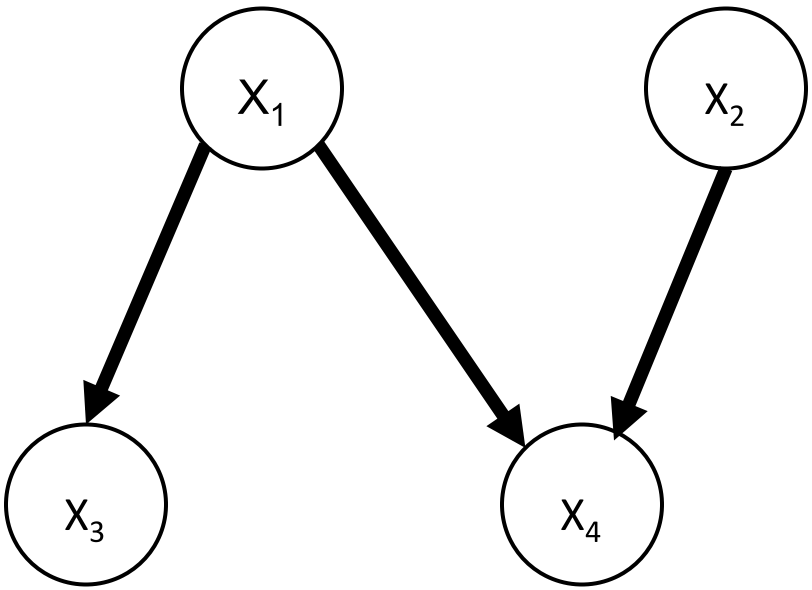


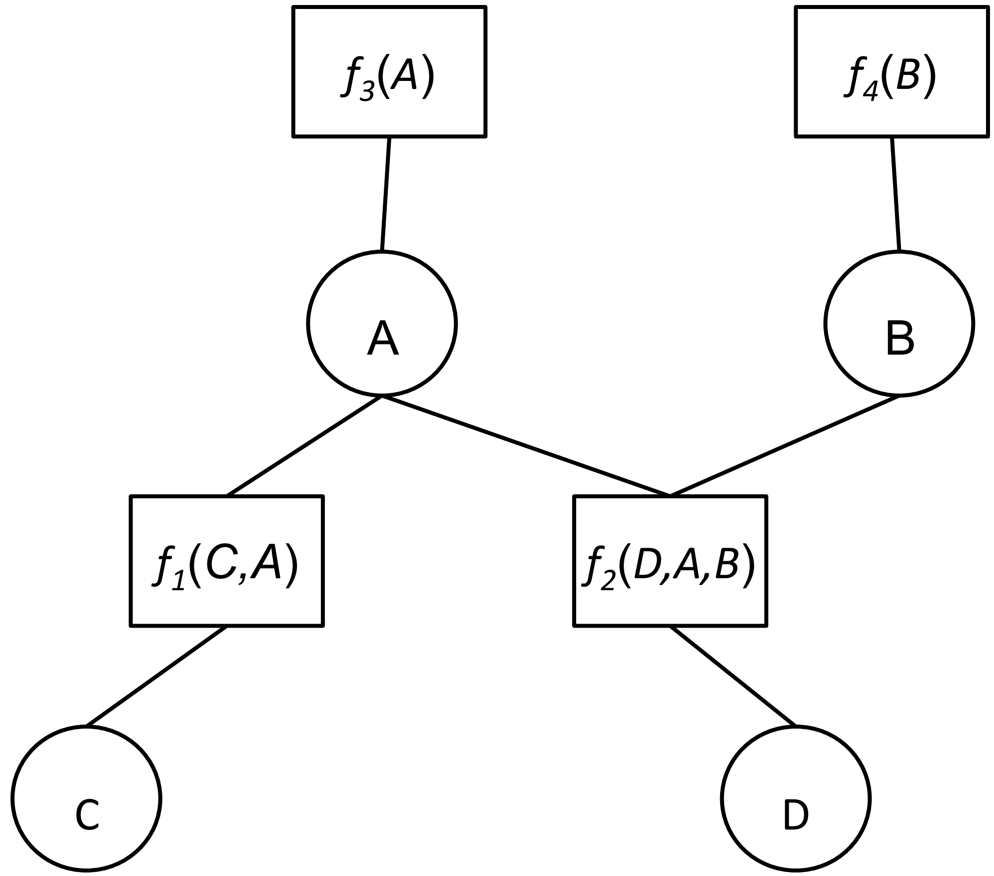
$$f_1(C,A) = P(C \mid A)$$
$$f_2(D,A,B) = P(D \mid A,B)$$
$$f_3(A) = P(A)$$
$$f_4(B) = P(B)$$

Il se compose de :

- Rectangles pour chaque fonction
- Liens, sans orientation entre la fonction et chaque variable dans la fonction

Un graphe de facteurs





Une méthode de calculer des probabilités par le passage de messages

- Les messages sont des vecteurs de nombres réels
- Les messages sont transmis de noeuds variables à des nœuds fonction, et de nœuds fonction à nœuds variables
- Un noeud peut envoyer un message à son voisin seulement si elle a reçu tous les messages de ses autres voisins.
- Étant donné un arbre, l'algorithme peut commencer par envoyant des messages de chaque des feuilles, et s'arrête une fois que chaque nœud a transmis un message à tous voisin.

Les messages fonction à variable

- Les messages entre une fonction f et une variable X sont calculé par

$$m_{f \to X}(x) = \sum_{x_1,\ldots,x_k} f(x, x_1, \ldots, x_k)\, m_{X_1 \to f}(x_1) \cdots m_{X_k \to f}(x_k)$$

- Avec une somme sur toutes les autres variables $X_1,\ldots,X_k$ (sauf X) avec un arc à f et sur le produit de la fonction et tous les messages variable-fonction à partir de ces autres variables
- Si f contient seulement X, alors (on a un nœud fonction feuille) $m_{f \to X}(x) = f(x)$.

Les messages variable-fonction

- Les messages entre une variable X et une fonction f sont calculé par

$$m_{X \to f}(x) \;=\; \begin{cases} 1 & \text{, nœud (variable) feuille} \\ m_{f_1 \to X}(x) \cdots m_{f_k \to X}(x) & \text{autrement} \end{cases}$$

- Ou simplement un produit sur toutes les messages d'autres fonctions f_1,..., f_k (sauf f) contenant X

À la fin

- Une fois que tous les messages ont été transmis,
- alors le <u>marginal</u> final pour toute variable X peut être calculé par

$$\mathrm{P}(X_i = x_i) \;=\; m_{f_1 \to X_i}(x_i) \cdots m_{f_k \to X_i}(x_i)$$

- sur toutes les fonctions $f_1,\ldots,f_k$ contenant X

Slides from:

Data Mining

Practical Machine Learning Tools and Techniques

Slides from Chapter 9 of *Data Mining*
by I. H. Witten, E. Frank, M. A. Hall and C.J. Pal

Computing Probabilities (in Tree Structured Graphs) More Efficiently with Factor Graphs

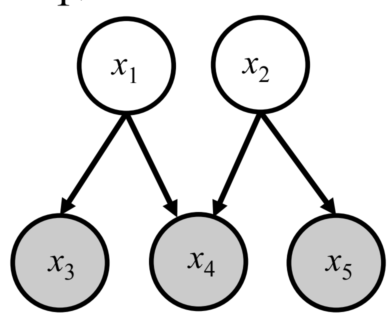
Intuition for the sum-product algorithm

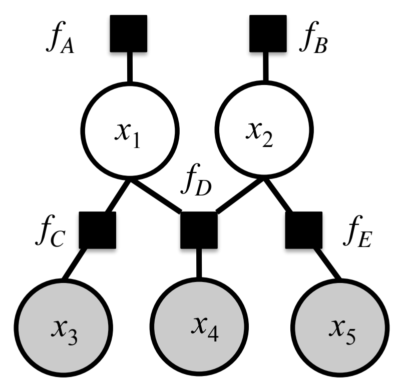
- The *sum-product algorithm* refers to a much better solution for computing marginals: simply *push the sums as far as possible to the right* before computing products of probabilities
- In our example the required marginalization can be computed by

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

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

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





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*:

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

- where *N*(*f*)*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:

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

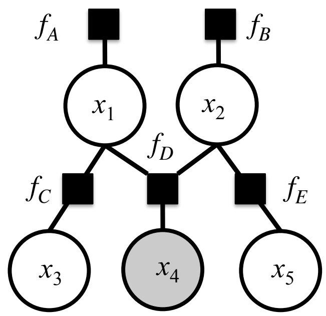
- where the product is over the (K) messages from all neighboring functions $N(x)$ other than the recipient function *f, i.e.* $f_k \in N(x) \setminus f$
- The marginal for each node is obtained from the product over all K+1 incoming messages from all functions connected to a variable

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

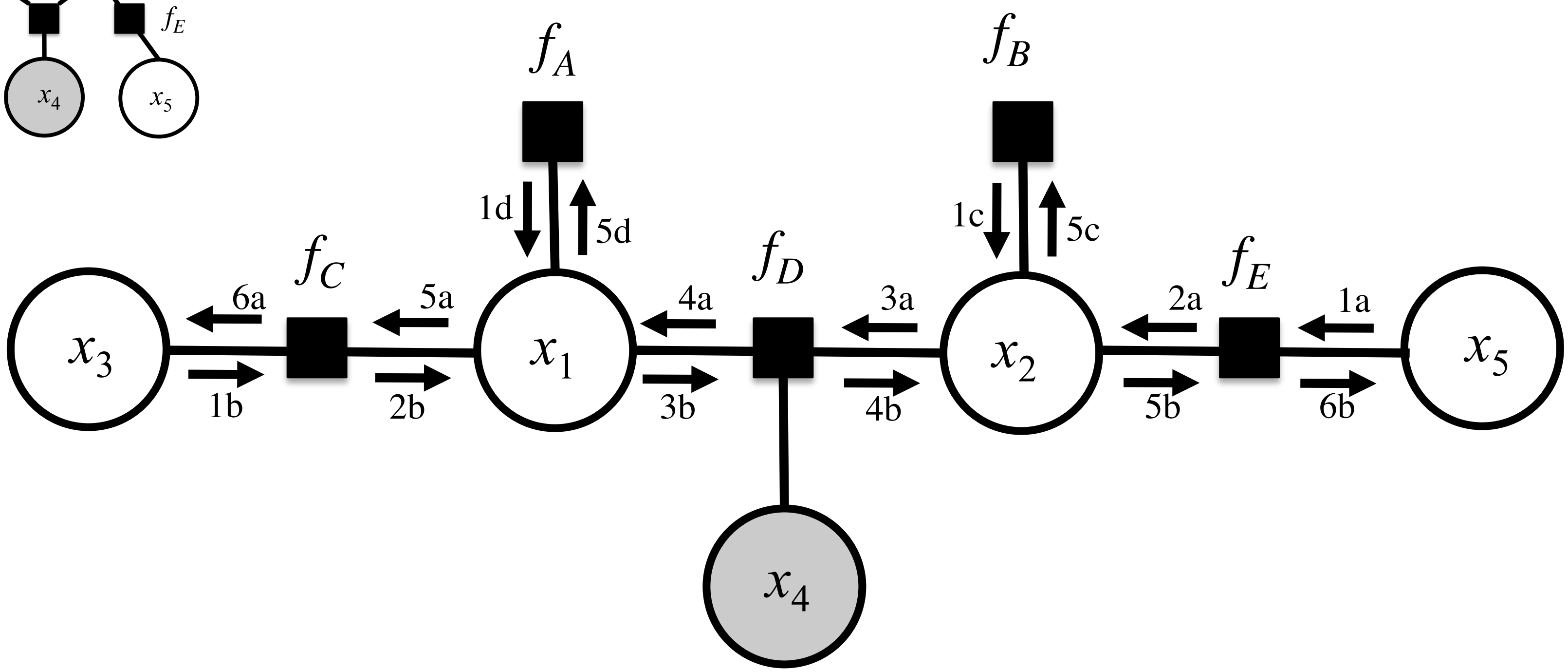
Numerical stability

- 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

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

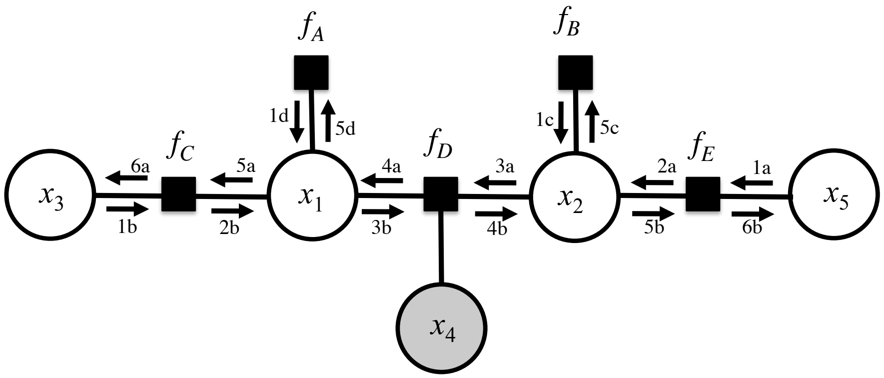


Sum product messages



$$P(x_3,\tilde{x}_4) = \underbrace{\sum_{x_1} P(x_3 \mid x_1) \underbrace{\underbrace{P(x_1)}_{1d} \underbrace{\sum_{x_2} P(\tilde{x}_4 \mid x_1, x_2) \underbrace{\underbrace{P(x_2)}_{1c} \underbrace{\sum_{x_5} P(x_5 \mid x_2)}_{2a}}_{3a}}_{4a}}_{5a}}_{6a} \cdot \underbrace{1}_{1a}$$

The messages for our example



$$P(x_3,\tilde{x}_4)=\underbrace{\sum_{x_1}P(x_3\mid x_1)\underbrace{\underbrace{P(x_1)}_{1d}\underbrace{\sum_{x_2}P(\tilde{x}_4\mid x_1,x_2)\underbrace{\underbrace{P(x_2)}_{1c}\underbrace{\sum_{x_5}P(x_5\mid x_2)\cdot\underbrace{1}_{1a}}_{2a}}_{3a}}_{4a}}_{5a}}_{6a}$$

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

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

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

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

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

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

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

Finding the Most Probable Explanation

with the max-product algorithm

Finding the most probable configuration

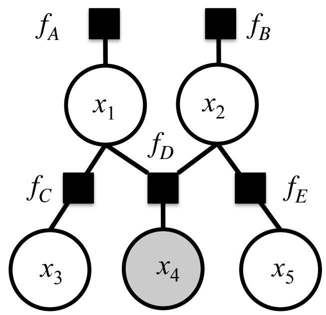
- Finding the most probable configuration of all other variables in our example given $x_4 = \tilde{x}_4$ involves searching for

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

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