# Génération de codes d'inférences probabilistes

Marvin Lavechin et Pierre Stefani

24 juillet 2015

# Table des matières

# 1 Présentation du projet

De Mars à Juillet 2014, encadré par Mr Pierre Henri Wuillemin, chercheur à l'Université Pierre et Marie Curie, nous avons travaillé sur la génération d'un code pour calculer des probabilités d'un réseau bayésien.

Les réseaux bayésiens sont des faisceaux de probabilités, ordonnés par des liens de parentés. Chacun des événements du réseau bayésien possède une table de probabilités dépendant des valeurs prises par ses parents. L'information peut ainsi être diffusé dans de tels réseaux. Le projet metaGenBayes utilise cette diffusion, appelé inférence, pour calculer une probabilité spécifique appelé target à partir d'un évènement certain(hard evidence) ou connu(soft evidence). Nous reviendrons sur ce vocabulaire plus en détail par la suite. Dans un cadre classique, ce calcul de probabilités selon des observations : $P(X|e)$ est simple. Toutefois, les modèles étant très vite complexes, l'inférence probabiliste devient un problème NP-difficile.

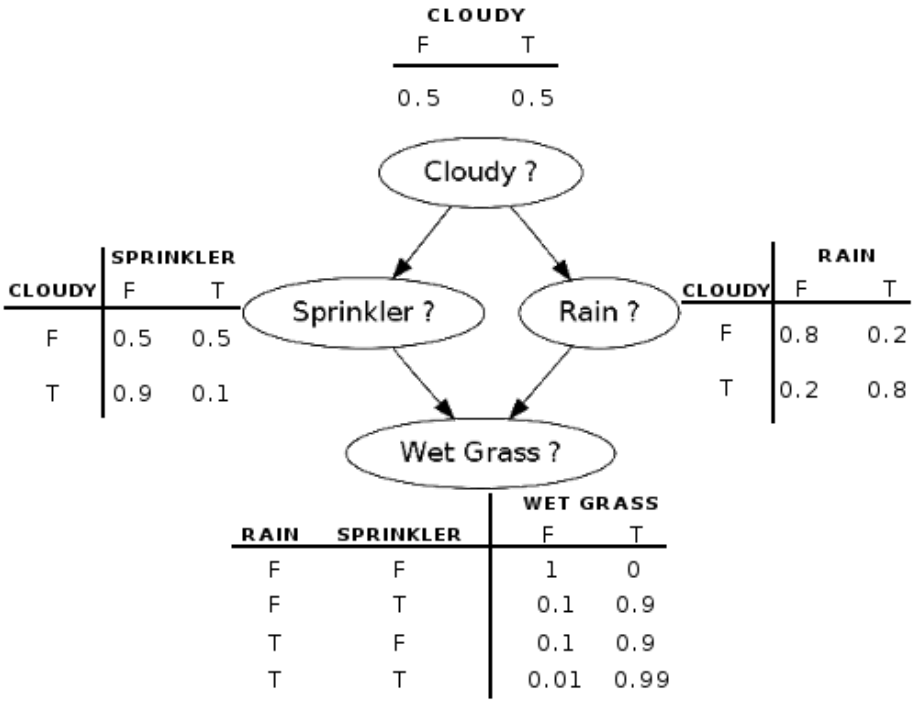


FIGURE 1 – Exemple de réseau bayésien

Le but du projet était de créer un moteur d'inférence probabiliste, qui n'allait pas faire le calcul mais générer des codes calculatoires dans différents langages cibles. Ce moteur s'appuie sur certaines bibliothèques lip6, comme par exemple pyAgrum.

# 2 Présentation du problème

## 2.1 Calcul de probabilité

### 2.1.1 Quelques formules

Comme explicité dans la courte présentation, l'inférence probabiliste s'appuie sur des probabilités conditionnelles, régies par quelques lois de calcul qu'il est important d'introduire :
Soient A et B deux événements quelconques d'un même univers. On s'intéresse à ce que devient la probabilité de A lorsqu'on apprend que B est déjà réalisé. On note $P(A|B)$ et on lit "probabilité de A sachant B". La définition mathématique de $P(A|B)$ est :

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)}, \quad P(B) \neq 0$$

(Formule de Bayes)

Cette formule s'écrit aussi : $P(A|B) \quad \alpha \quad P(A) \times P(B|A)$ [1]
Ceci nous amène à considérer deux autres formules. Soit $(A_i)_{i \in I}$ un système complet d'événements de probabilités non nulles. Alors pour tout événement $B$ on a :

$$P(B) = \sum_{i \in I} P(A_i \cap B) = \sum_{i \in I} P(A_i)P(B|A_i)$$

(Formule des probabilités totales)

Si $P(A_1 \cap \ldots \cap A_n) \neq 0$, on a :

$$P(A_1 \cap \ldots \cap A_n) = \prod_{i=1}^{n} P(A_i|A_{i+1} \cap \ldots \cap A_n)$$

(Formule des probabilités composées)

### 2.1.2 Calcul au sein d'une table de probabilité jointe

Soient $A_1, A_2, A_3$ trois variables aléatoires binaires. On peut écrire une table listant toutes les combinaisons possibles de ces 3 variables, qui sont au nombre de $2^3$.
En effet, on a :

$$\begin{aligned} &\big(A_1 = true \quad A_2 = true \quad A_3 = true\big) \\ &\big(A_1 = false \quad A_2 = true \quad A_3 = true\big) \\ &\big(A_1 = true \quad A_2 = false \quad A_3 = true\big) \\ &\qquad\qquad\qquad\qquad\qquad\qquad etc... \end{aligned}$$

Pour chacune des combinaisons, supposons que nous disposions de la probabilité jointe de cette combinaison.

1. $P(A|B)$ est la loi à posteriori, $P(A)$ la loi à priori et $P(B|A)$ la vraisemblance, tandis que P(B) sert de coefficient normalisateur

Si l'on veut la probabilité $P(A_1 \cap A_2)$, la formule des probabilités totales nous dit qu'il suffit qu'on somme les probabilités des combinaisons de la table pour laquelle $A_1$ est vrai et $A_2$ l'est aussi.
Si l'on veut la probabilité $P(A_1|A_2)$, la formule de Bayes nous dit qu'il suffit qu'on somme toutes les probabilités des combinaisons de la table pour lesquelles $A_1$ est vrai et $A_2$ est vrai en divisant par la somme des probabilités des combinaisons de la table pour lesquelles $A_2$ est vrai.
On peut donc à partir de la table des probabilités jointes déduire n'importe quelle probabilité conditionnelle. Les tailles de ces tables évoluant de manière exponentielle, il devient impossible d'utiliser simplement cette approche pour le calcul d'inférences probabilistes.

## 2.2 Arbres de Jonction

Avant de revenir en détail sur les méthodes utilisés pour calculer ces probabilités conditionnelles, nous allons décrire le principe des arbres de jonctions, transformation nécessaire des réseaux bayésiens en arbres dépourvus de toute orientation. La motivation principale de cette étape est de créer des cliques - noeuds regroupants plusieurs probabilités/évènements de notre réseau. Ces cliques seront alors liées sans aucune orientation (évitant ainsi la gestion des cycles). Ces arbres doivent également satisfaire la propriété suivante des arbres de jonctions :
*Deux cliques U et V possédant un ensemble S de probabilités communes, sont séparés, lors de leur liaison dans l'arbre de jonction, par cet ensemble S*

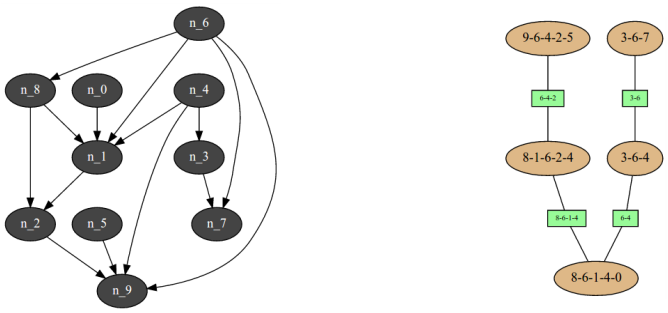


FIGURE 2 – Un réseau bayésien généré (à gauche), et un arbre de jonction possible pour ce réseau (à droite)

La première étape de création de ces arbres est une phase de *moralisation* : il s'agit de créer un lien entre deux parents d'un même élément. Dans l'exemple de la Figure 2, un lien est crée entre n2 et n5, deux des parents de n9. Une fois cette moralisation effectuée, la création de *supercliques* regroupant des élèments connectés du réseau bayésien est quasiment finie. Toute orientation est ensuite enlevée, et la dernière étape est une phase dite de 'triangulation', pour enlever toute possibilité de cycles à la nouvelle structure. Les arbres de jonctions regroupent désormais plusieurs probabilités, et l'on associera à ces ensembles leurs probabilités respectives avec des *potentiels* dans le lexique des réseaux

bayésiens, qui seront simplement des tableaux de probabilités dans nos langages plus courants.

## 2.3 Diffusion des probabilités dans les arbres de jonctions

Rappelons notre principal objectif : à partir d'une évidence donnée pour une probabilité A, nous devons calculer la nouvelle probabilité de l'évènement B : l'information de A va donc devoir circuler dans le réseau. La structure d'arbres de jonctions va grandement simplifier ce que l'on appelera la *diffusion* de l'information. La clique de départ, celle pour laquelle a été transmise l'information (ce choix arbitraire n'a d'influence que sur le temps de réponse du programme) possède un potentiel $\Phi$ contenant les probabilités des variables de notre réseau. Pour passer d'une clique A à la clique B, l'information est projetée de A à B selon les séparateurs comme le montre la Figure 3.

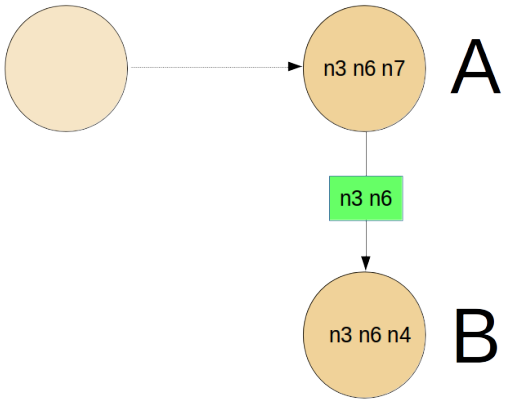


FIGURE 3 – Exemple de diffusion d'une clique A vers une clique B dans un arbre de jonction

La projection de A sur B suit d'abors la formule de projection sur le séparateur :

$$\Psi'_{AB} = \sum_{X_i \in \Phi_A, X_i \notin \Psi_{AB}} \Phi_A$$

Puis du séparateur à la clique B :

$$\Phi'_B = \Phi_B * \Psi'_{AB}$$

Dans le cas de la figure 3, cela donne :

$$\Phi'_{n3,n6,n4} = \Phi_{n3,n6,n4} * \sum_{n7} \Phi_{n3,n6,n7}$$

Ainsi, à partir des évidences fournies à certaines cliques de l'arbre de jonction, toute l'information sera diffusée vers une clique principale, appelée clique racine, dans une étape que nous avons appelée *l'absorption*. Voici l'algorithme utilisé pour définir cette clique racine, vers qui tout converge :

**Algorithm 1** Trouver la clique racine

```
VoisinsMax = −1
for I ∈ cliques do
  for X ∈ targets do
    if X ∈ variables(I) then
      if nbVoisins(I) > voisinsMax then
        VoisinsMax ← nbVoisins(I)
        CliqueRacine ← I
      end if
    end if
  end for
end for
return CliqueRacine
```

Cet algorithme permet à la fois à la clique racine de contenir au moins une target, et donc permettra un calcul rapide pour au moins un résultat, et également d'avoir le maximum de voisins, nous assurant ainsi qu'il ne s'agira pas d'une clique isolée pour laquelle de nombreux calculs inutiles seraient effectués.

Il faut pour calculer la probabilité de la target, revenir sur une clique la contenant, et appliquer une marginalisation :
Soit une target $T \in C$, une clique de notre arbre. Marginaliser revient à calculer :

$$P(T) = \sum\nolimits_{X \in C, X \neq T} \Phi_C$$

Une fois la phase d'absorption effectuée, notre arbre est dans l'état suivant : toute l'information a convergée et est contenue dans une clique racine. C'est à partir de celle-ci que la seconde étape commence, la *diffusion* vers les targets, pour y appliquer les marginalisations. Une fonction du module **Compiler** -que nous développerons plus tard, crée une liste de diffusion avec les cliques visitées contenant les targets. Admettons que B et C aient reçu l'information de la diffusion, qui doit maintenant être projetée sur A et D