Philippe Marcotte 1736842

TP1 INF2882

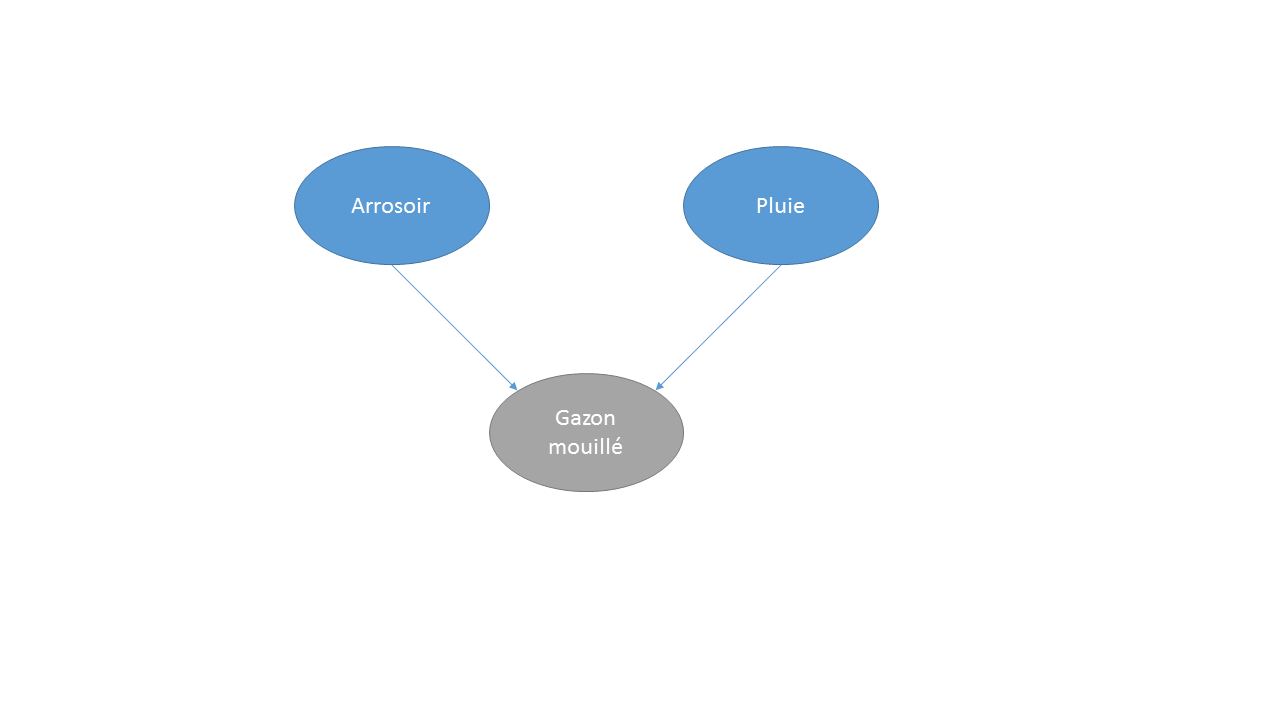
2017-02-06

Pour ce TP, python3, ainsi que, les librairies numpy et matplotlib ont été utilisées.

**Question 1** (Dossier Question 1)

Explaining away:

Ce phénomène peut être observé lorsque le fait de connaître l’état d’une variable binaire aléatoire réduit la probabilité qu’une autre variable aléatoire soit dans un certain état.

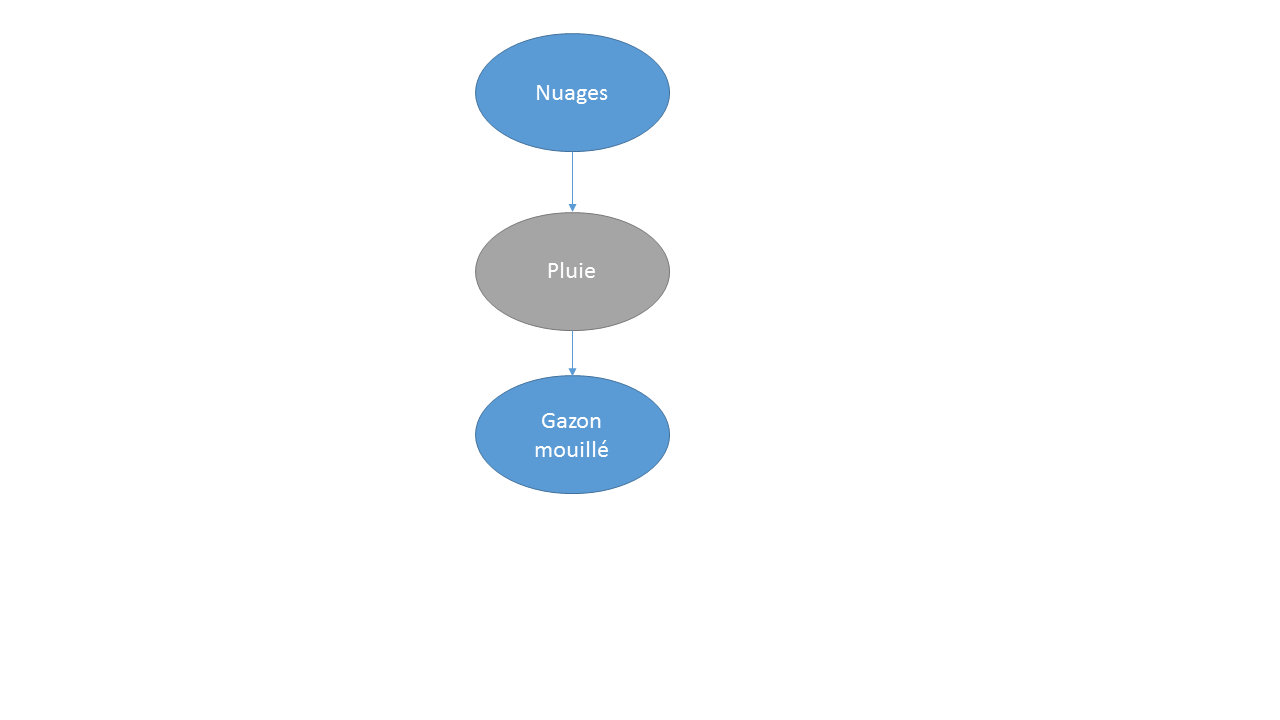
L’exemple de code fournit dans explainingAway.py utilise le réseau suivant.

Sachant que le gazon est mouillé, si l’on sait en plus que l’arrosoir a été activé, cela réduit la probabilité qu’il pleut.

Serial blocking :

Ce phénomène peut être observé lorsque le fait de connaître l’état d’une variable aléatoire ne change rien à la probabilité qu’une variable binaire aléatoire grand-parente soit dans un certain état.

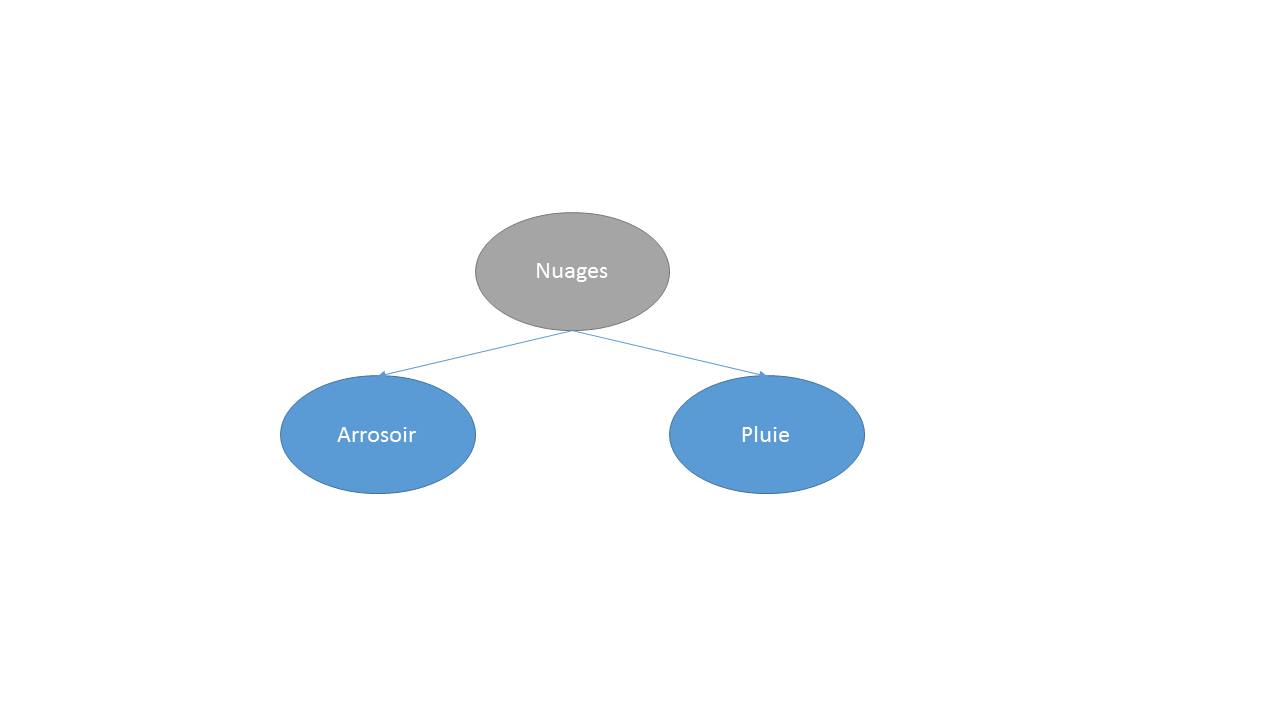
L’exemple de code fournit dans serialBlocking.py utilise le réseau suivant.



Sachant qu’il pleut, le fait de savoir que le gazon est mouillé ne change rien à la probabilité qu’il y aille des nuages.

Divergent blocking :

Ce phénomène peut être observé lorsque le fait de connaître l’était d’une variable aléatoire ne change rien à la probabilité qu’une variable aléatoire sœur soit dans un certain état.

L’exemple de code fournit dans divergentBlocking.py utilise le réseau suivant.

Sachant qu’il y a des nuages, le fait de savoir que l’arrosoir est activé ne change rien à la probabilité qu’il y aille de la pluie.

**Question 2** (Dossier Question 2)

Exécuter question2.py pour tout voir. Fermer le graphique pour voir ce qui suit la question B.

a) Voir reseau.py

b) Voir B.py

c) Voir C.py

d) Voir D.py

e) P(J) = P(A) \* P(J|A) + P(!A) \* P(J|!A)

P(A) = P(C) \* P(T) \* P(A|C,T) + P(!C) \* P(T) \* P(A|!C,T) + P(C) \* P(!T) \* P(A|C,!T) + P(!C) \* P(!T) \* P(A|!C,!T)

P(C|J) = P(C,J) / P(J)

P(C,J) = P(C,J,M) + P(C,J,!M)

P(C,J,M) = P(C,J,M,A) + P(C,J,M,!A)

P(C,J,M,A) = P(C,J,M,A,T) + P(C,J,M,A,!T)

P(C,J,M,A,T) = P(C) \* P(T) \* P(A|C,T) \* P(M|A) \* P(J|A)

P(C,J,M,A,!T) = P(C) \* P(!T) \* P(A|C,!T) \* P(M|A) \* P(J|A)

P(C,J,M,!A) = P(C,J,M,!A,T) + P(C,J,M,!A,!T)

P(C,J,M,!A,T) = P(C) \* P(T) \* P(!A|C,T) \* P(M|!A) \* P(J|!A)

P(C,J,M,!A,!T) = P(C) \* P(!T) \* P(!A|C,!T) \* P(M|!A) \* P(J|!A)

P(C,J,!M) = P(C,J,!M,A) + P(C,J,!M,!A)

P(C,J,!M,A) = P(C,J,!M,A,T) + P(C,J,!M,A,!T)

P(C,J,!M,A,T) = P(C) \* P(T) \* P(A|C,T) \* P(!M|A) \* P(J|A)

P(C,J,!M,A,!T) = P(C) \* P(!T) \* P(A|C,!T) \* P(!M|A) \* P(J|A)

P(C,J,!M,!A) = P(C,J,!M,!A,T) + P(C,J,!M,!A,!T)

P(C,J,!M,!A,T) = P(C) \* P(T) \* P(!A|C,T) \* P(!M|!A) \* P(J|!A)

P(C,J,!M,!A,!T) = P(C) \* P(!T) \* P(!A|C,!T) \* P(!M|!A) \* P(J|!A)

**Question 3** (Dossier Question 3)

Le fichier Node.py contient les classes Node, VariableNode et FactorNode. Celles-ci représentent les nœuds qu’on retrouve dans un graph de facteurs.

Le fichier Graph.py contient la classe représentant le graph de facteur. Il fait le lien entre toutes les nœuds.

Le fichier main.py répond à nouveau aux questions C) et D) en utilisant un graph de facteur cette fois-ci.