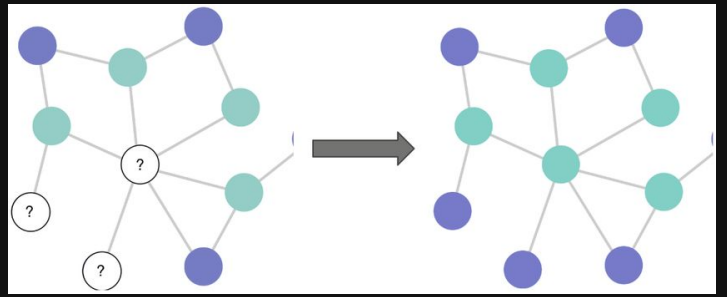
Graph Mining & network science

Classification des noeuds (méthodes traditionnelles)



Machine learning sur les graphes

- **Notre problématique :**

  étant donné un réseau avec certains noeuds labellisés, comment assigner un label à tous les autres noeuds ?

- **Exemple :**

  dans un réseau, certains noeuds sont des escrocs, et d'autres sont des personnes dignes de confiance. Comment identifier les escrocs ?

- **Classification collective :**

  assigner les labels aux autres noeuds simultanément en exploitant la structure du réseau

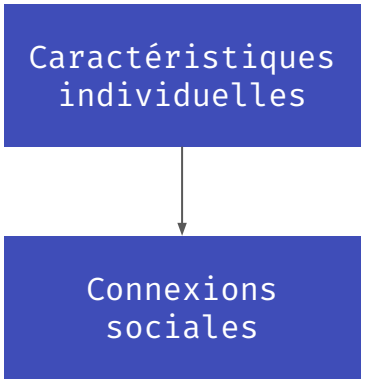
- **Pourquoi est-ce une bonne idée ?**

  Parce qu'il existe des corrélations dans les réseaux.

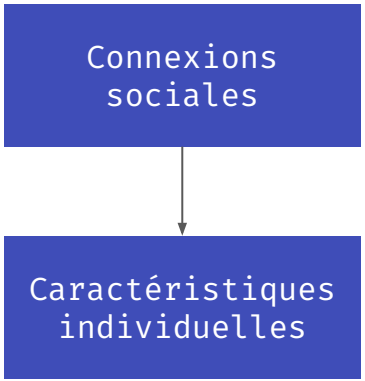
Corrélations dans les graphes

- les comportements individuels sont corrélés dans un réseau
- il existe trois grands principes qui mènent à la corrélation dans un réseau :

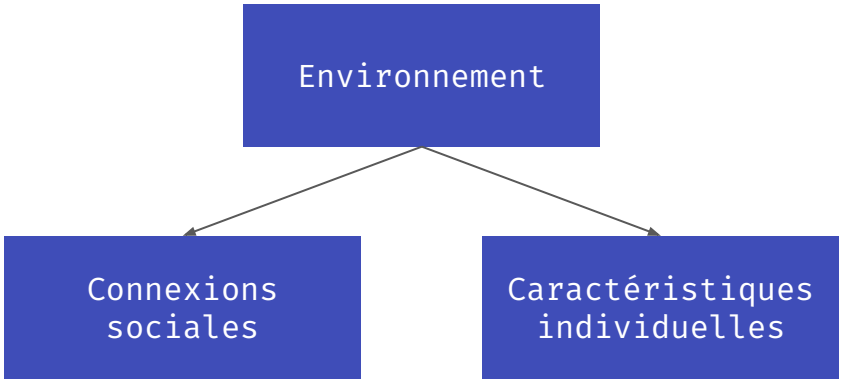
Homophilie



Influence



Confondant



Définition du problème

Des entités similaires sont typiquement proches, voire même directement connectée :

- “Coupable par association” : si je suis connecté à un noeud labellisé X, j’ai de fortes chances d’avoir également ce label
- **Exemple : les pages web frauduleuses vs. bénignes**

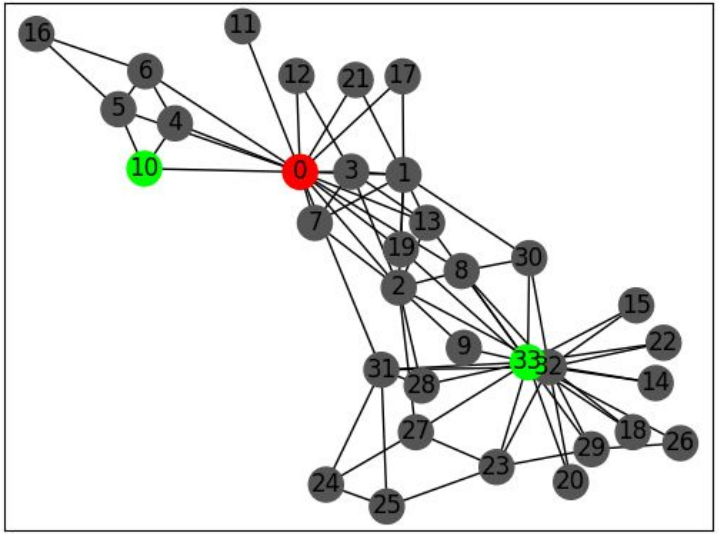
  Les pages web frauduleuses sont liées entre elles pour accroître leur visibilité, apparaître crédible et avoir un score d’importance élevé sur Google

Définition du problème

Le label d'un noeud u dans le réseau peut dépendre :

- des features de **u**
- des labels des voisins de **u**
- des features des voisins de **u**

Définition du problème



Étant donné :

- des noeuds avec des features
- quelques noeuds labellisés (apprentissage semi-supervisé)

Trouver :

- la classe (**vert**/**rouge**) pour le reste des noeuds

Hypothèse :

- homophilie du graphe

Classification collective

Intuition : classification simultanée des objets liés en exploitant des corrélations

Applications :

- classification de documents
- POS Tagging
- prédiction de lien
- OCR
- image segmentation
- sociologie / analyse des réseaux sociaux
- détection de spam ou de fraude

Classification collective

Propriété de Markov : le label Yi d'un noeud i dépend du label de ses voisins Ni

$$P(y_i|i) = P(y_i|N_i)$$

La classification collective se déroule en 3 étapes :

Classifieur local	Classifieur relationnel	Inférence collective
Initialisation des labels	Capture des corrélations entre les noeuds	Propagation des corrélations à travers le réseau

Classification collective

Classifieur local

Initialisation des labels

Classifieur relationnel

Capture des corrélations entre les noeuds

Inférence collective

Propagation des corrélations à travers le réseau

Classifieur local : utilisé pour initialiser les labels

- prédire les labels à partir des attributs et des features des noeuds
- apprentissage classique
- n'utilise pas d'information de la structure du réseau

Classifieur relationnel : capture les corrélations dans le réseau

- apprendre un classifieur à partir des labels et des features des voisins d'un noeud
- l'information de la structure du réseau est exploitée

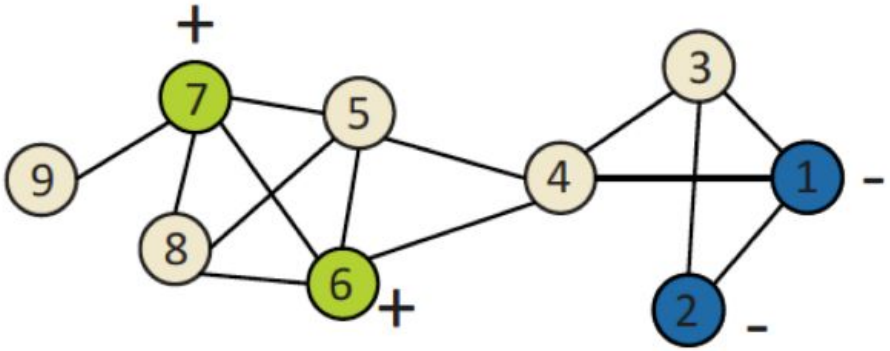
Inférence collective : fait propager les corrélations dans le réseau

- appliquer le classifieur relationnel à chaque noeud itérativement
- itérer jusqu'à stabilisation des labels
- la structure du réseau affecte considérablement la prédiction finale

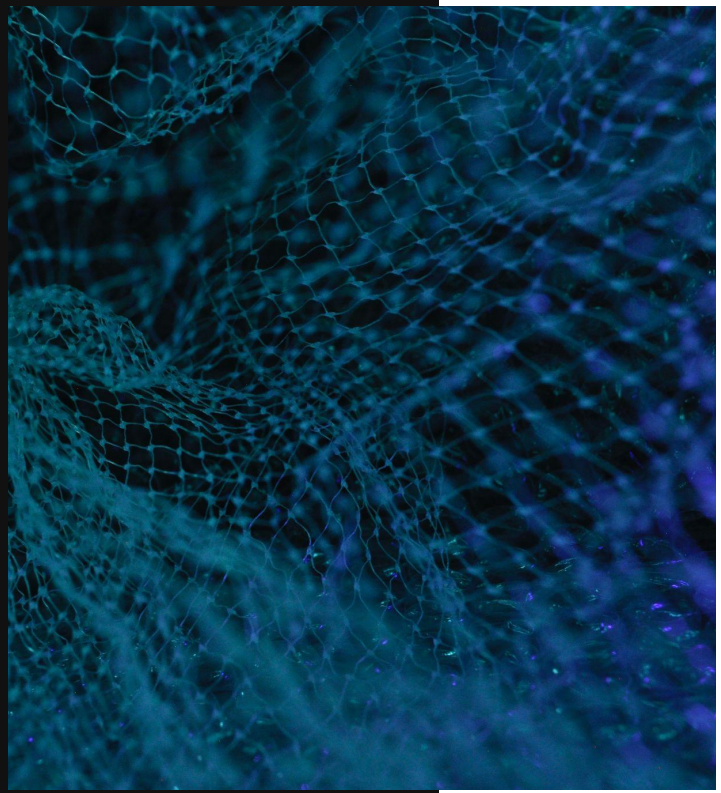
L'inférence exacte est faisable uniquement sous certaines conditions (NP-difficile)

Nous allons nous intéresser à 3 techniques :

- Classification relationnelle
- Classification itérative
- Belief propagation



Classification relationnelle



Classification relationnelle

Les classifieurs relationnels sont des classifieurs collectifs sans classifieur local

- **avantages :**
  - convergence et exécution généralement rapides
  - simple conceptuellement et à coder
  - peuvent réaliser une classification avec peu de labels
  - bons résultats lorsque les corrélations sont fortes dans le réseau
- **un problème majeur :**
  - n'exploitent pas les features individuelles des noeuds

Deux exemples d'algorithmes dans ce cours : **le classifieur relationnel probabiliste** et **le classifieur relationnel par marches aléatoires**

Classifieur probabiliste relationnel

Idée générale :

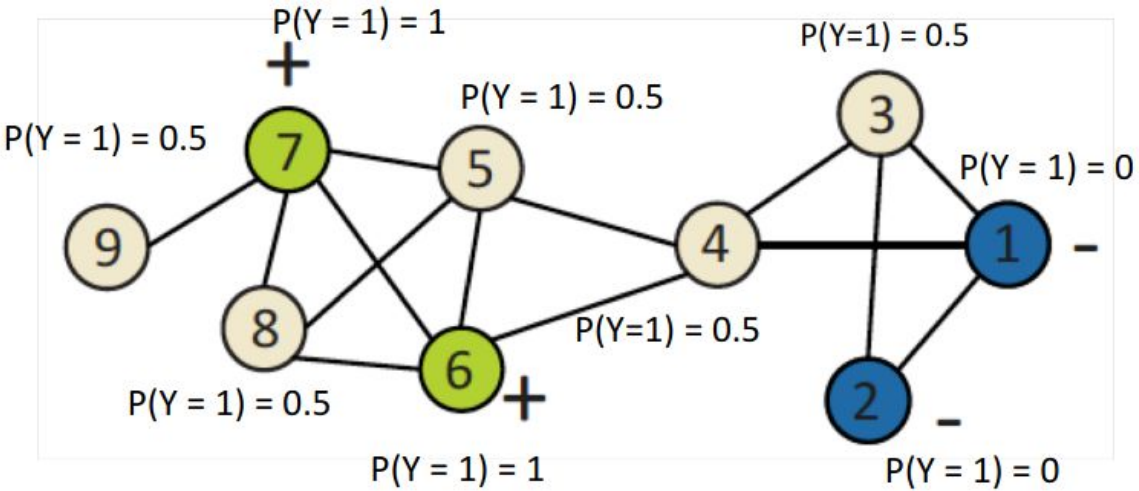
la probabilité de Y_i est un moyenne pondérée des probabilités Y_j de ses voisins.

$$P(Y_i = c) = \frac{1}{k_i} \sum_{i \to j} A_{ij} P(Y_j = c)$$

- **Initialisation** des labels Y pour les noeuds labellisés
- Pour les noeuds non-labellisés, on **initialise Y de façon uniforme** sur l'ensemble des classes
- **Mise à jour itérative** des probabilités pour tous les noeuds, jusqu'à convergence

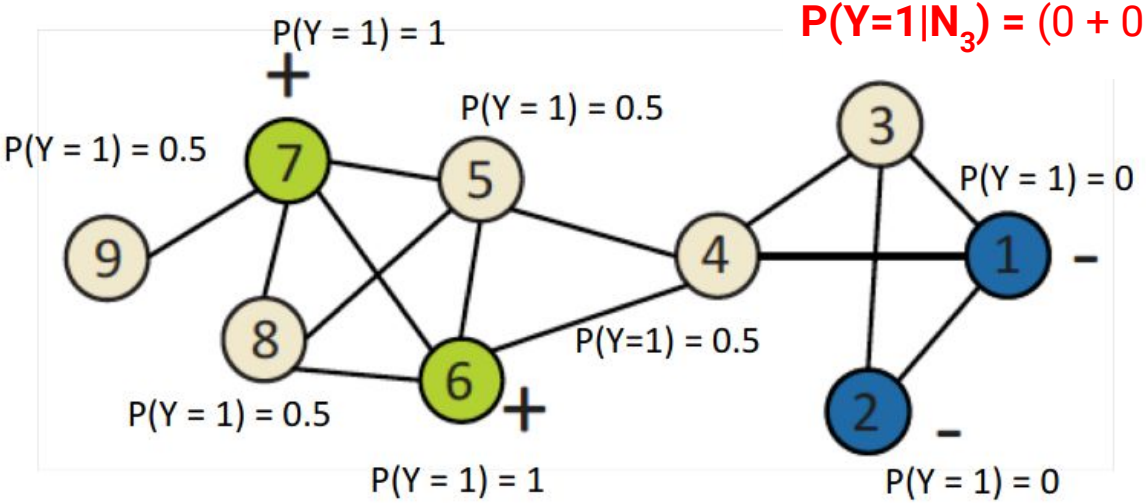
Classifieur probabiliste relationnel - initialisation

Initialisation : tous les noeuds non-labellisés sont initialisés uniformément, les noeuds du training set appartiennent à leur classe de façon certaine



Classifieur probabiliste relationnel - itération 1

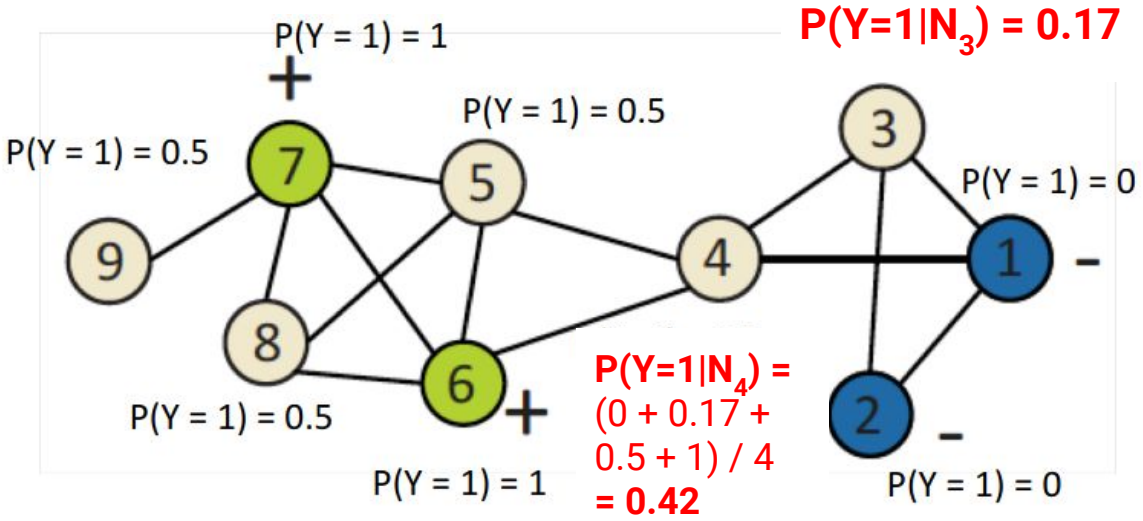
Mise à jour du noeud **3**, $N_3 = \{1, 2, 4\}$



$\mathbf{P(Y=1|N_3) = (0 + 0 + 0.5) / 3 = 0.17}$

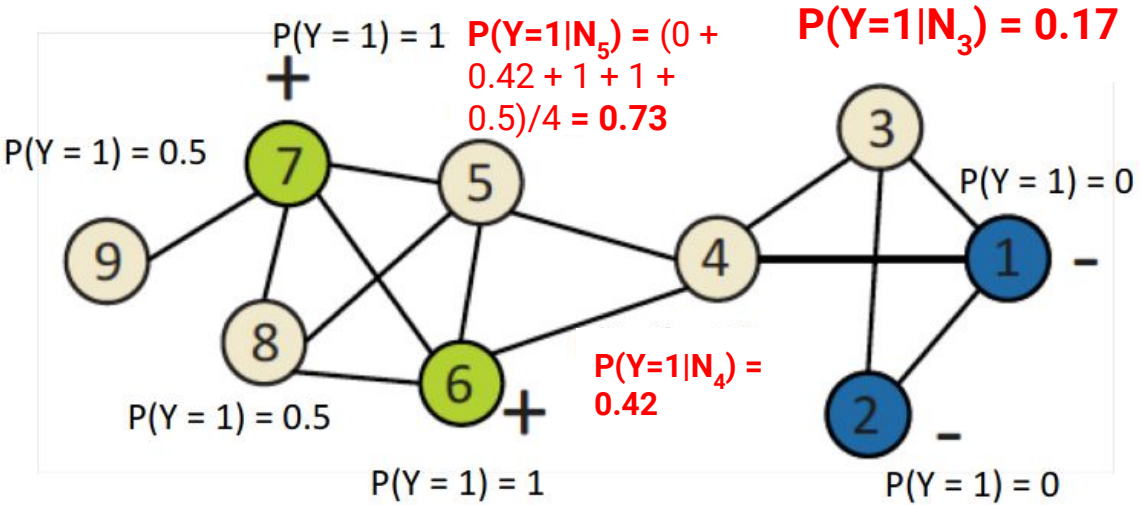
Classifieur probabiliste relationnel - itération 1

Mise à jour du noeud **4**, $N_4 = \{3, 5, 6\}$



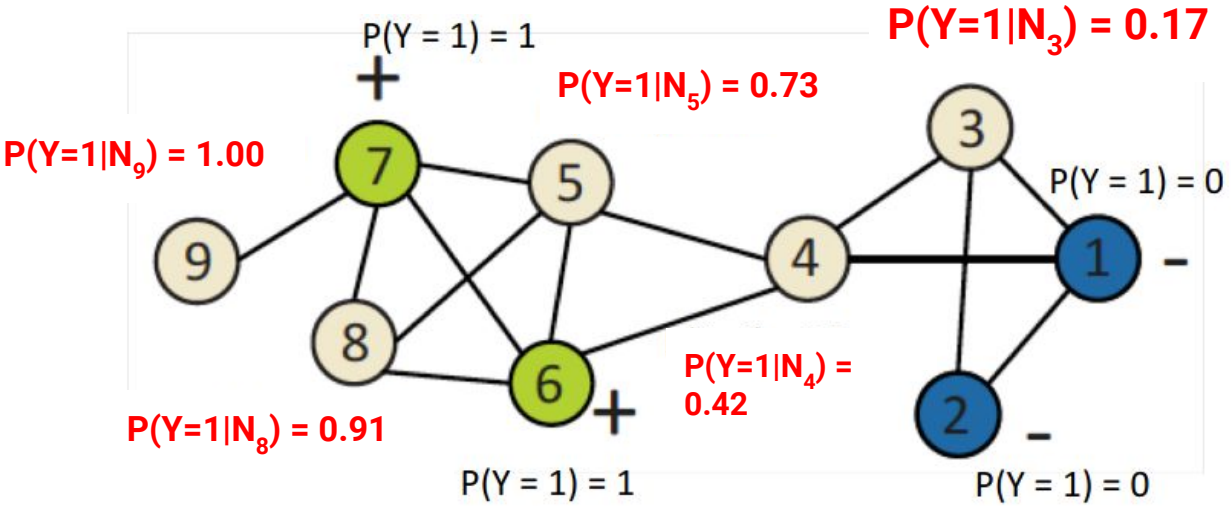
Classifieur probabiliste relationnel - itération 1

Mise à jour du noeud **5**, $N_5 = \{4, 6, 7, 8\}$

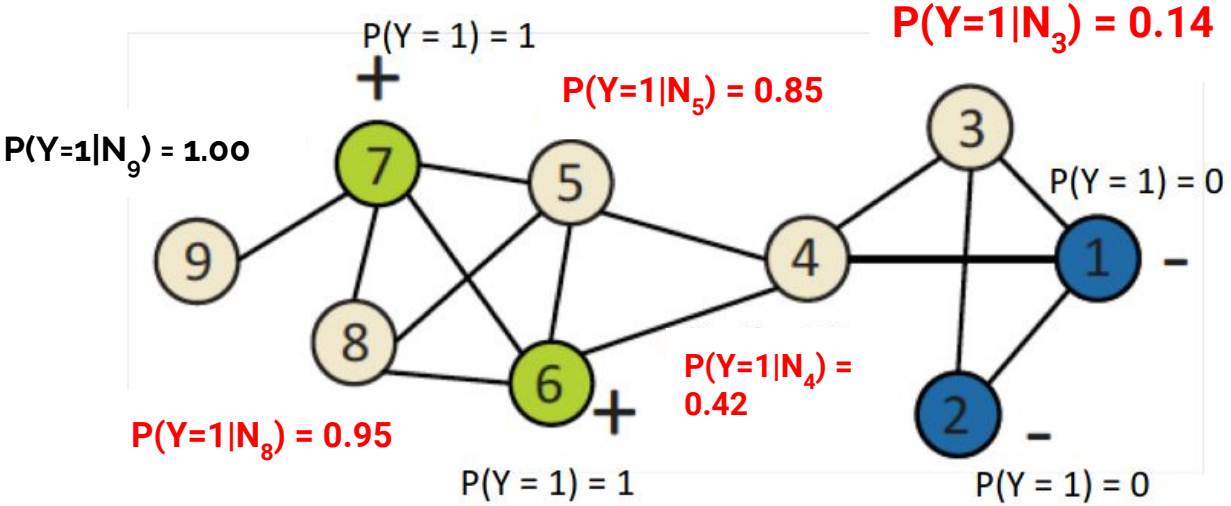


Classifieur probabiliste relationnel - itération 1

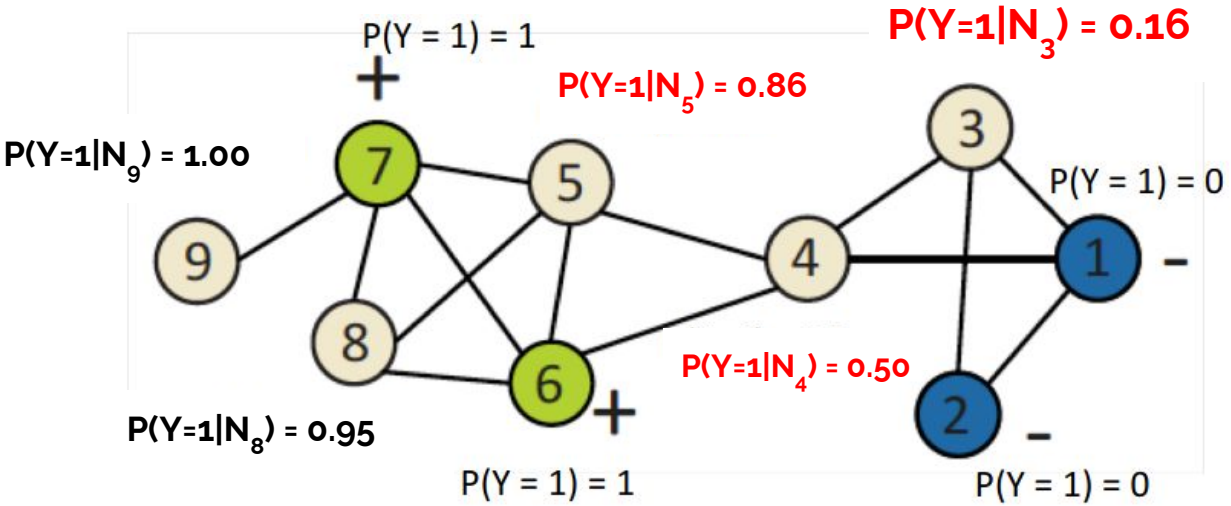
Après l'itération 1



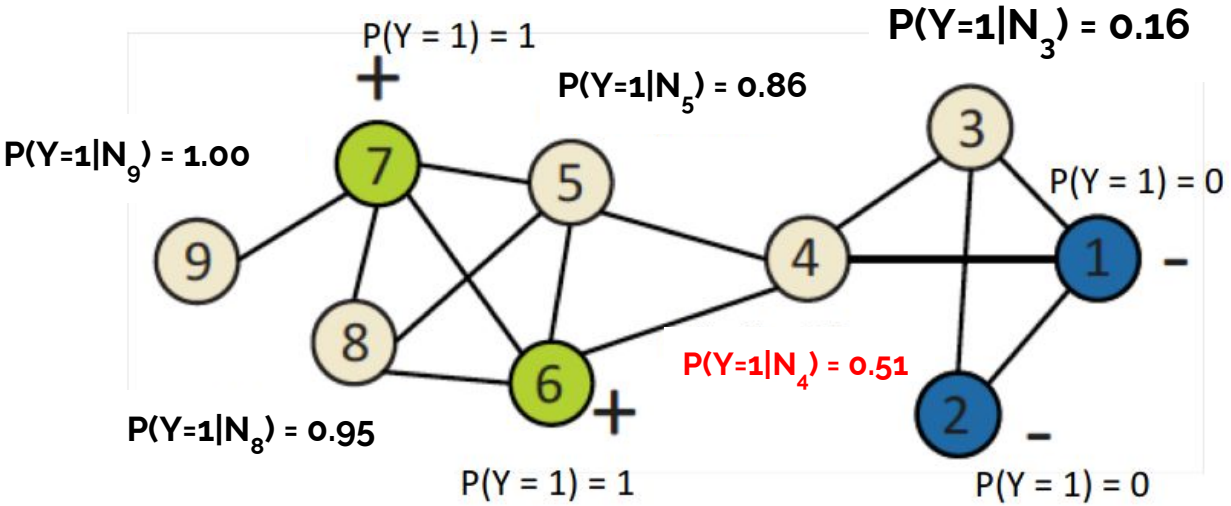
Après l'itération 2



Après l'itération 3



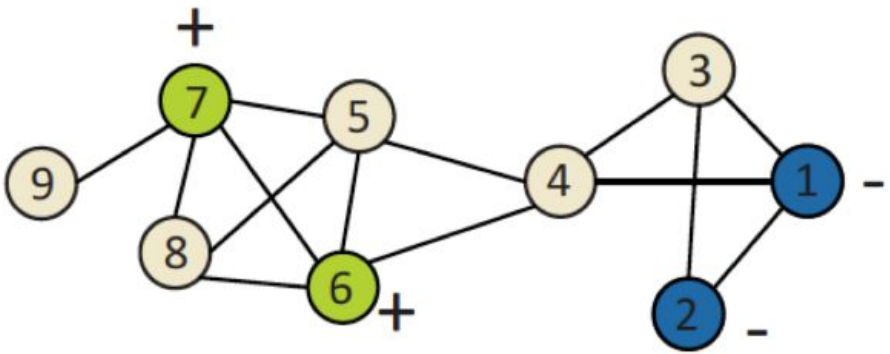
Après l'itération 4



Classifieur probabiliste relationnel - final

Tous les scores se sont stabilisés après 5 itérations :

- noeuds **5**, **8** et **9** sont **positifs** (**P(Yi =1) > 0.5**)
- noeud **3** est **négatifs** (**P(Yi =1) < 0.5**)
- noeud **4** est indéterminé : **P(Yi =1) = 0.5**



Classifieur probabiliste par marches aléatoires

Contexte

- on considère un random walk avec des “états puits” : les noeuds labellisés V_l
- **condition : chaque noeud peut atteindre un noeud labellisé**

Idée générale :

la probabilité $Y_i = c$ est la probabilité qu’un random walk partant de i termine sa course sur un noeud labélisé **c**

Classifieur probabiliste par marches aléatoires

$$\hat{y}_i(c) = \sum_{j \in V_l} p_{ij}^{\infty} y_j(c)$$

Avec $\hat{y}_i(c)$ la probabilité finale que le noeud **i** soit classé **c**

et $y_j(c)$ la probabilité initiale que le noeud **j** soit classé **c**

et p_{ij}^{∞} la probabilité d'atterrir à **j** en partant de **i**, après une infinité de timesteps

Classifieur probabiliste par marches aléatoires

Formulation matricielle :

$$\hat{Y} = P^{\infty} Y$$

où $Y = (Y_l, 0)$ et $\hat{Y} = (Y_l, \hat{Y}_u)$

et **P** est la matrice de transition : **p_{ij} = probabilité de i → j**

De manière générale, la matrice de transition est calculée par :

$P = D^{-1}A$

En **ordonnant et indexant** les **l** noeuds labellisés de **1** à **l**,

P prend la forme :

$$P = \begin{bmatrix} P_{ll} & P_{lu} \\ P_{ul} & P_{uu} \end{bmatrix} = \begin{bmatrix} I & 0 \\ P_{ul} & P_{uu} \end{bmatrix}$$

Et : $$\lim_{t\to\infty} P^t = \begin{bmatrix} I & 0 \\ (\sum_{n=0}^{\infty} P_{uu}^n) P_{ul} & P_{uu}^{\infty} \end{bmatrix} = \begin{bmatrix} I & 0 \\ (1 - P_{uu})^{-1} P_{ul} & 0 \end{bmatrix}$$

On obtient finalement :

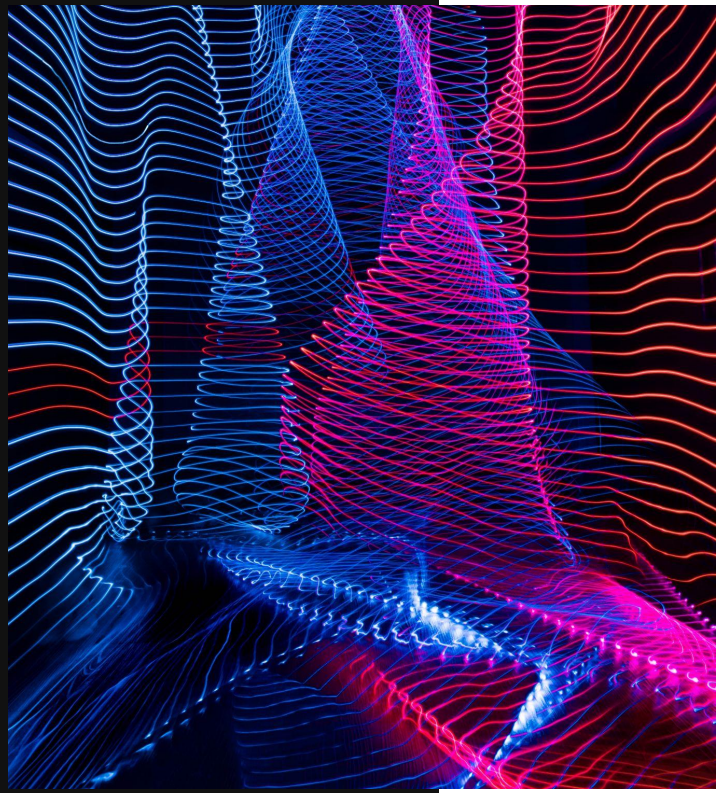
$$\begin{pmatrix} Y_l \\ \hat{Y}_u \end{pmatrix} = \begin{bmatrix} I & 0 \\ (I - P_{uu})^{-1} P_{ul} & 0 \end{bmatrix} \begin{pmatrix} Y_l \\ 0 \end{pmatrix}$$

Et donc la solution :

$$\hat{Y}_u = (I - P_{uu})^{-1} P_{ul} Y_l$$

Possible uniquement si **$(I - P_{uu})$** est non-singulière, ie. il est toujours possible d'atteindre un noeud labellisé depuis n'importe quel noeud.

Classification itérative



Classifieur itératif

- les **classifieurs relationnels** vus précédemment n'utilisent pas les features individuelles des nodes. Comment pouvons-nous les exploiter ?
- L'idée principale de la classification itérative : classer les noeuds en fonction **de leurs attributs et des labels de leurs voisins**
- **Procédure générale d'un classifieur itératif :**
  - créer un vecteur de features locales a_i pour chaque noeud **i**
  - entraîner un classifieur en utilisant les a_i
  - agréger leurs features :

    décompte, mode, proportion, moyenne, existence, etc.

Architecture d'un classifieur itératif

Bootstrap :

- on obtient de chaque noeud **i** un vecteur a_i
- on entraîne un classifieur local (ex: SVM, kNN, etc.) pour calculer la meilleure valeur de Y_i

Itération :

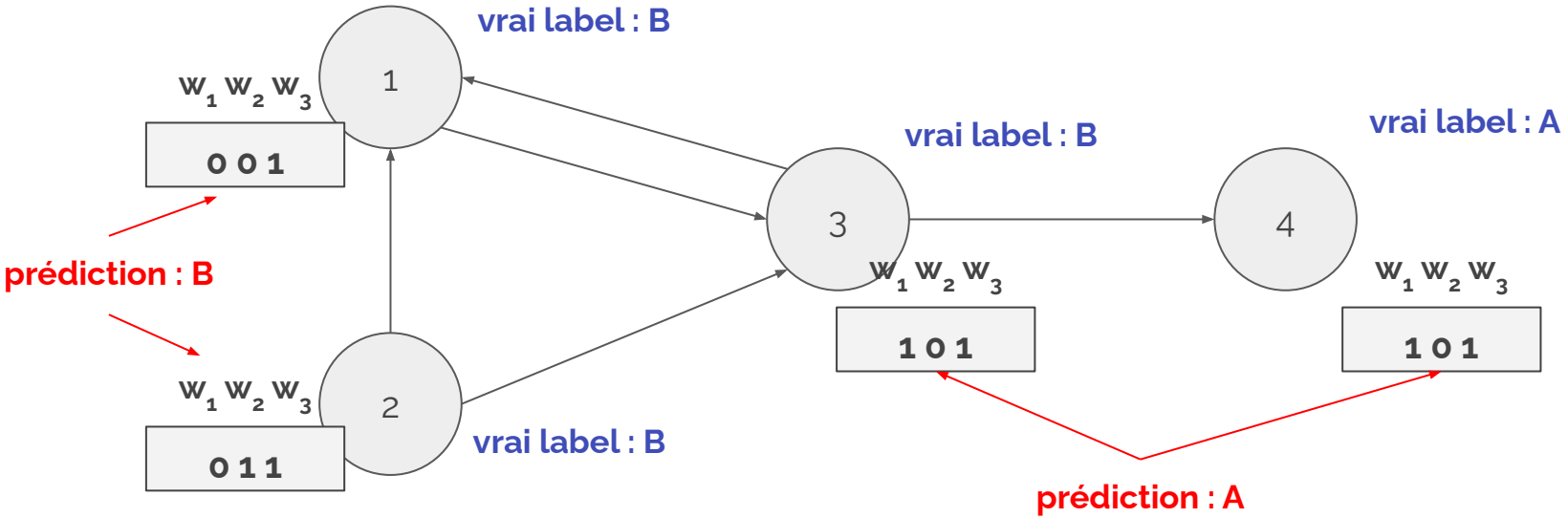
- pour chaque noeud **i** :
  - on met à jour son vecteur a_i
  - on met à jour le label Y_i via le classifieur local
- on itère jusqu'à stabilisation, nombre d'itérations ou convergence

La convergence n'est pas garantie.

Exemple : classification d'une page web

- **w_1, w_2, w_3, … :** valeurs binaires indiquant la présence d'un mot
- **Baseline :** on utilise un classifieur **kNN**

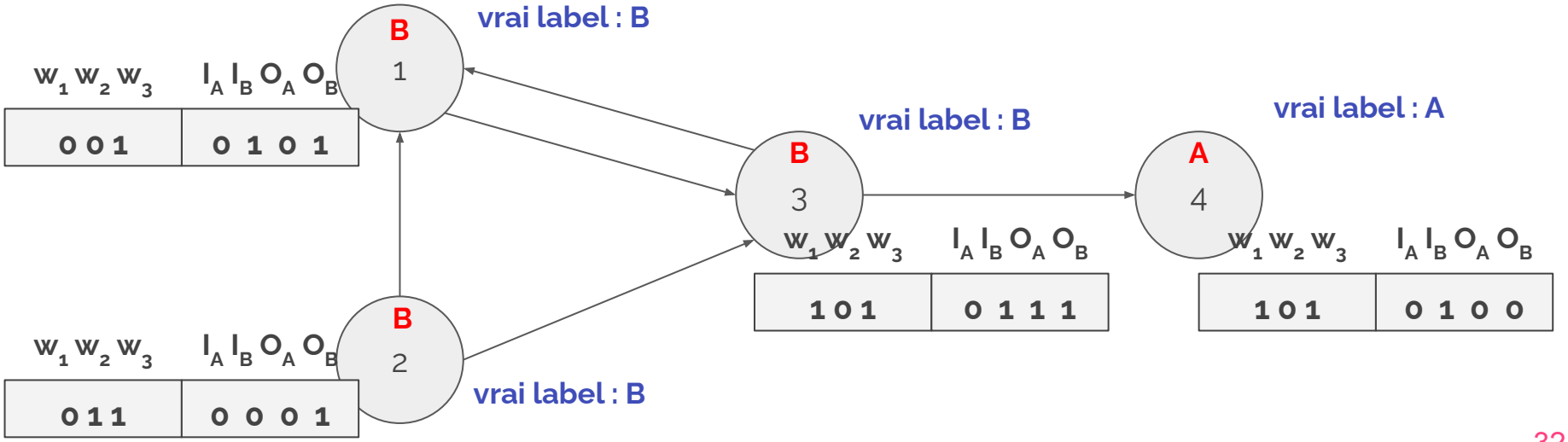
Le noeud 3 est mal classé !



Exemple : classification d'une page web

- Chaque noeud maintient un vecteur de labels de voisins :

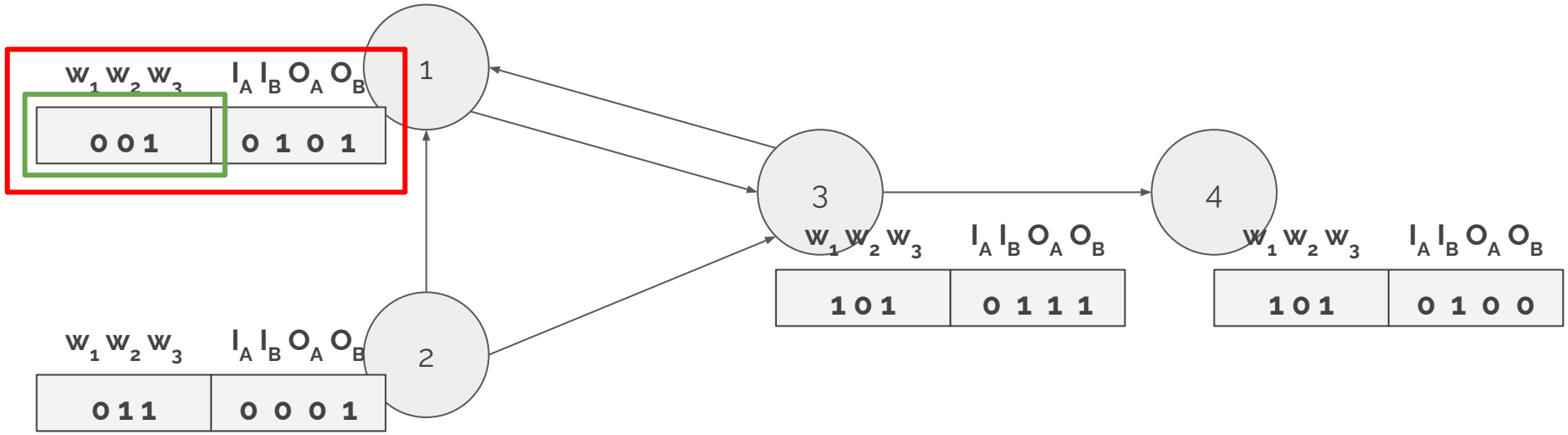
  (I_A, I_B, O_A, O_B) avec (I = In, O = Out)
- $I_A = 1$ si au moins des pages entrantes est labellisée **A**, etc.



1. Phase d'entraînement

on entraîne deux classifieurs :

- un sur les word vectors **(en vert)**
- un sur les word + link vectors **(en rouge)**



Le business des fausses reviews

Analyse comportementale :

- features individuelles,
- géographiques,
- horaires de connexion,
- durée de session, etc.

Analyse sémantique :

- usage de superlatifs,
- fautes d'orthographes,
- mots suspects

Ces deux analyses sont faciles à tromper ! ⟶

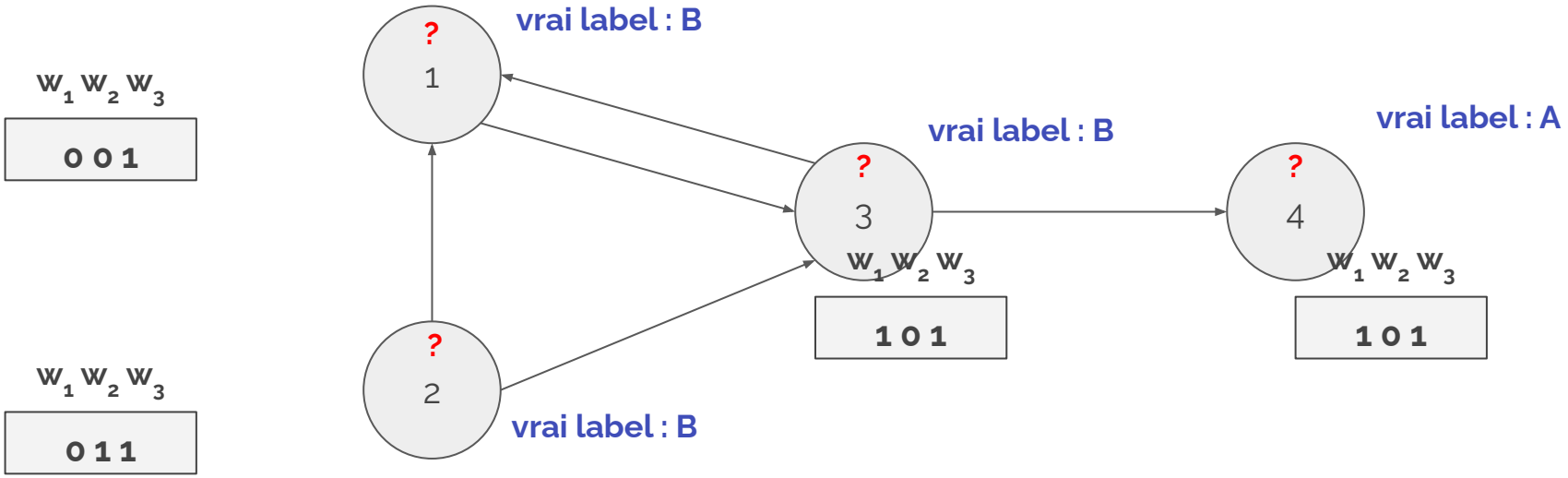
Difficilement falsifiable :

la structure du réseau

des interactions

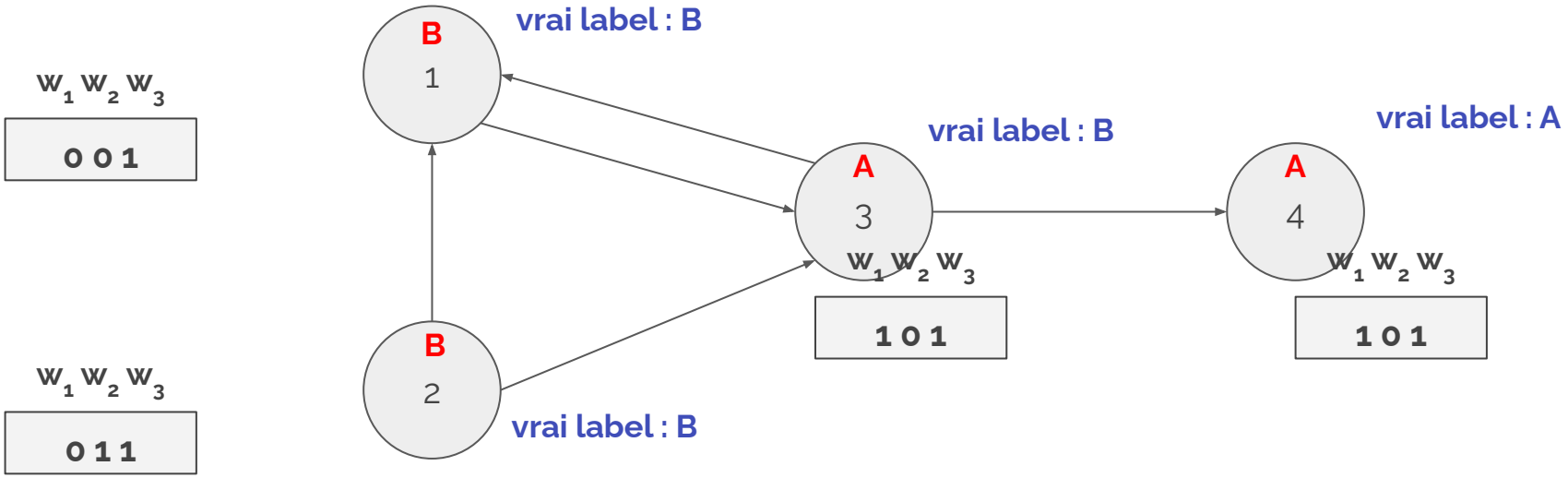
2. Bootstrap

on utilise le classifieur des word vectors pour initialiser les valeurs.



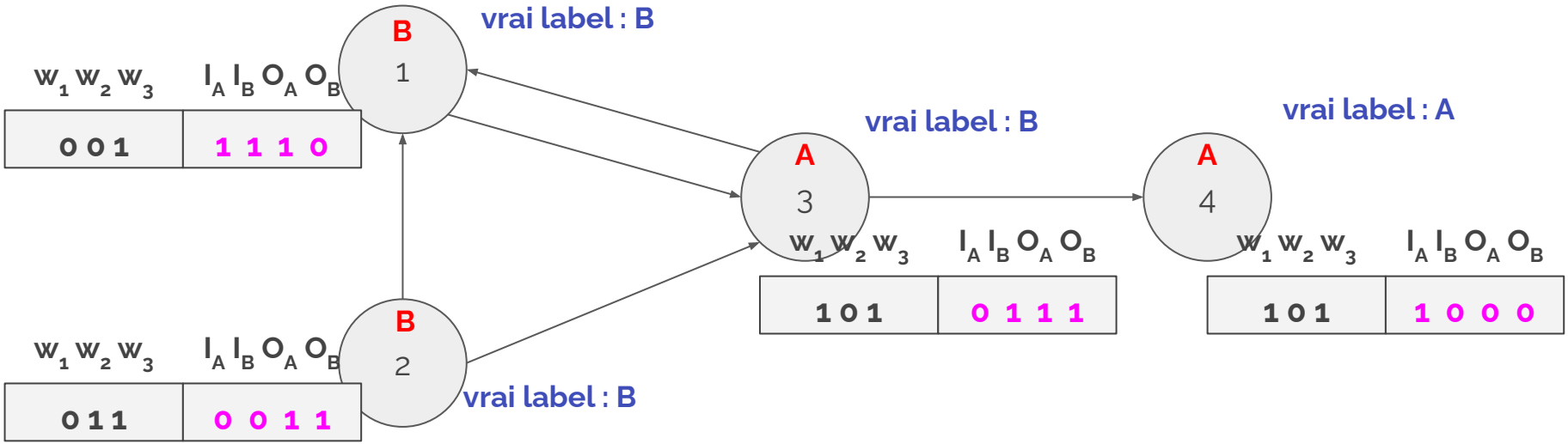
2. Bootstrap

on utilise le classifieur des word vectors pour initialiser les valeurs.



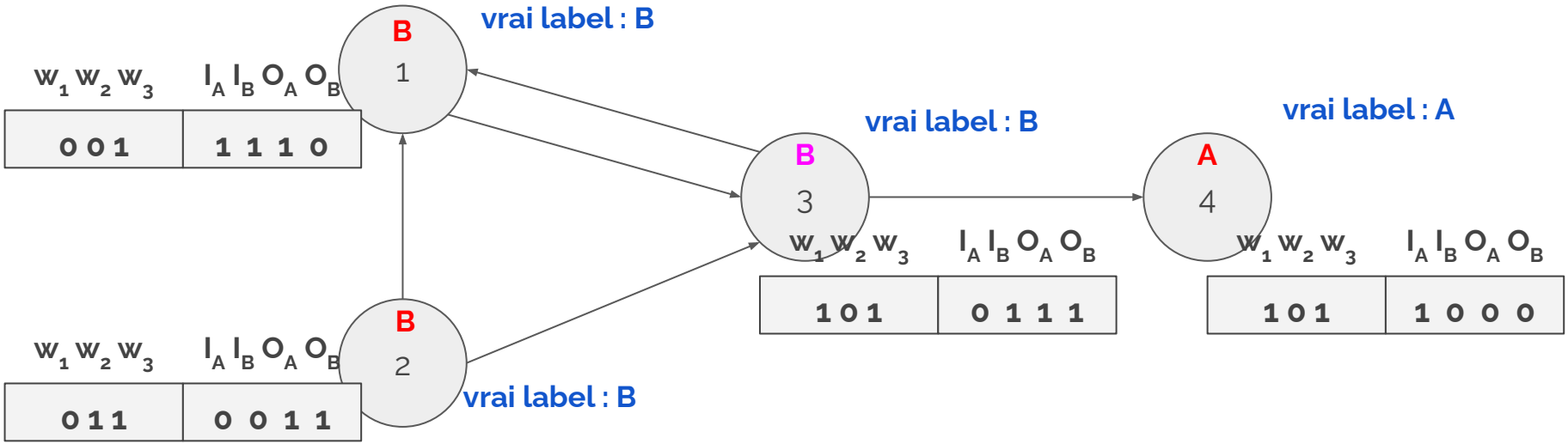
3. Itération : mise à jour des features relationnelles

on met à jour les **vecteurs de voisinage** pour chaque noeud



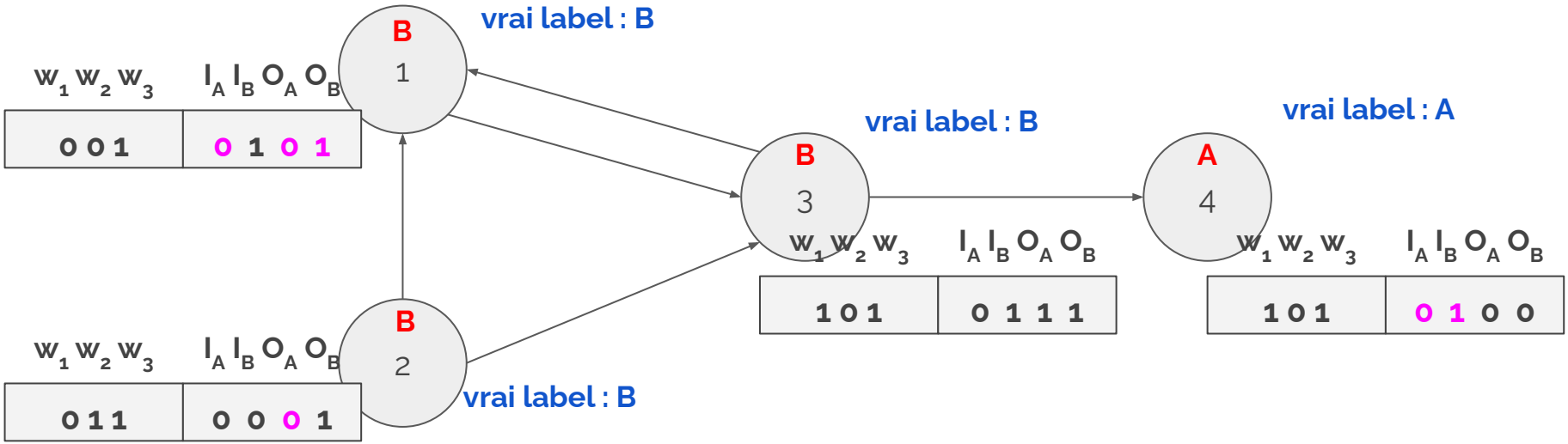
3. Itération : classification

on classifie tous les noeuds en utilisant le deuxième classifieur



3. Itération : mise à jour des features relationnelles

on continue le cycle jusqu'à convergence



REV2

un exemple de
classification itérative

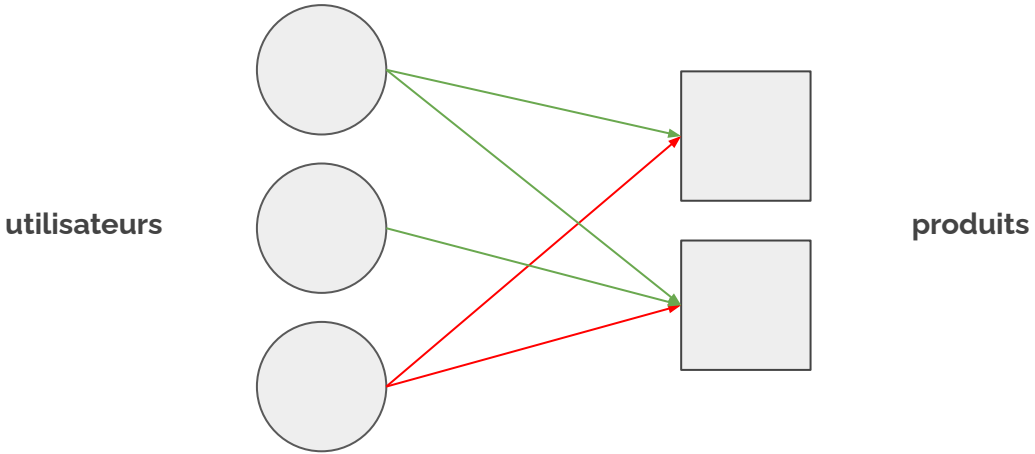
Le business des fausses reviews

- une étoile de plus sur **Google Avis** correspond à presque **10%** de CA en plus : publier des faux avis est très tentant car lucratif.
- les reviews sont généralement très négatives ou très positives
- certaines entreprises ou individus sont payés pour faire ça !

Un enjeu très important pour Google, Amazon, Yelp, TripAdvisor, etc.

Définition du problème

- **en entrée :** un graphe biparti de reviewers et de produits
  - noeuds : utilisateurs et produits
  - liens : scores entre **-1** et **1**
- **en sortie :** ensemble d'utilisateurs qui publient de fausses reviews



REV2, Kumar et al. - 2018

Idée principale :

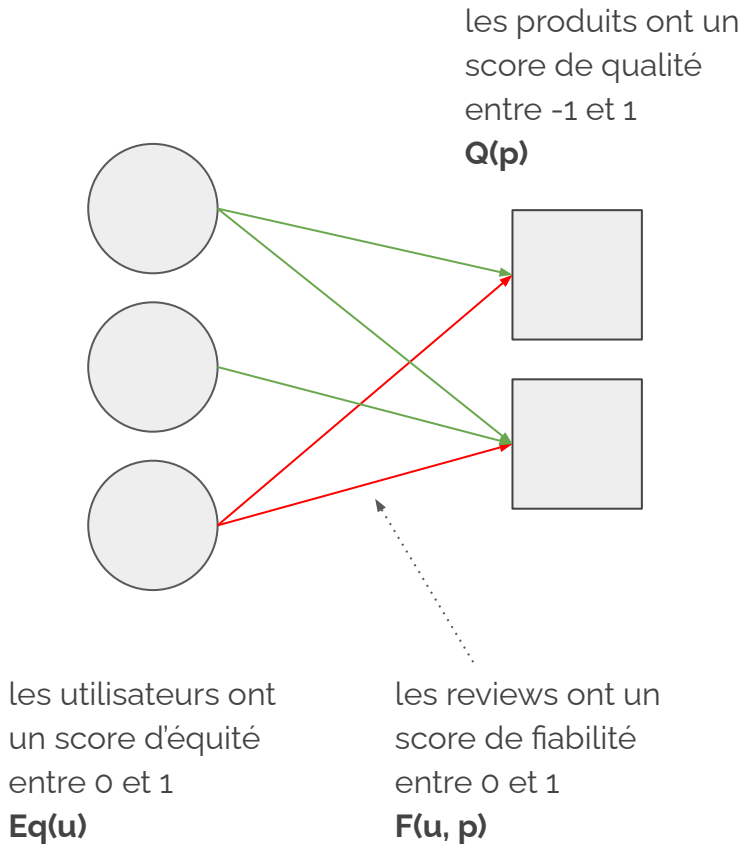
les utilisateurs, les produits et les reviews ont chacun un score intrinsèque

- les utilisateurs ont un score d'équité
- les produits ont un score de qualité
- les reviews ont un score de fiabilité

Aucune valeur n'est initialement connue.

Comment calculer la valeur des noeuds et des arêtes simultanément ?

=> Classification itérative



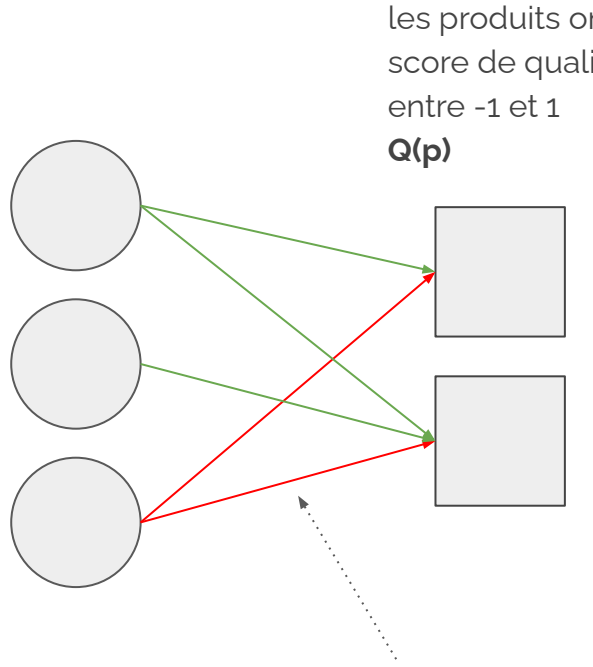
Équité des utilisateurs

Pour une fiabilité et une qualité données,

on définit l'équité d'un utilisateur comme :

$$Eq(u) = \frac{1}{d_{out}(u)} \sum_{u \to p} F(u, p)$$

Intuitivement : l'équité d'un utilisateur est la moyenne de la fiabilité de ses reviews



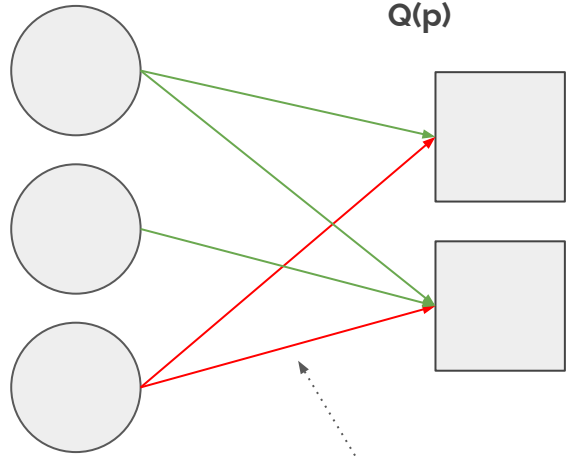
Qualité des produits

Pour une fiabilité et une équité données,

on définit la qualité d'un produit par :

$$Q(p) = \frac{1}{d_{in}(p)} \sum_{u \to p} F(u,p) \cdot score(u,p)$$

Intuitivement : la qualité d'un produit est la moyenne des scores pondérée par la fiabilité des reviews



les produits ont un score de qualité entre -1 et 1
Q(p)

les utilisateurs ont un score d'équité entre 0 et 1
Eq(u)

les reviews ont un score de fiabilité entre 0 et 1
F(u, p)

Fiabilité d'une review

Pour une qualité et une équité données,

on définit la fiabilité d'une review par :

$$F(u,p) = \frac{1}{\gamma_1 + \gamma_2}(\gamma_1 Eq(u) + \gamma_2(1 - \frac{|score(u,p) - Q(p)|}{2}))$$

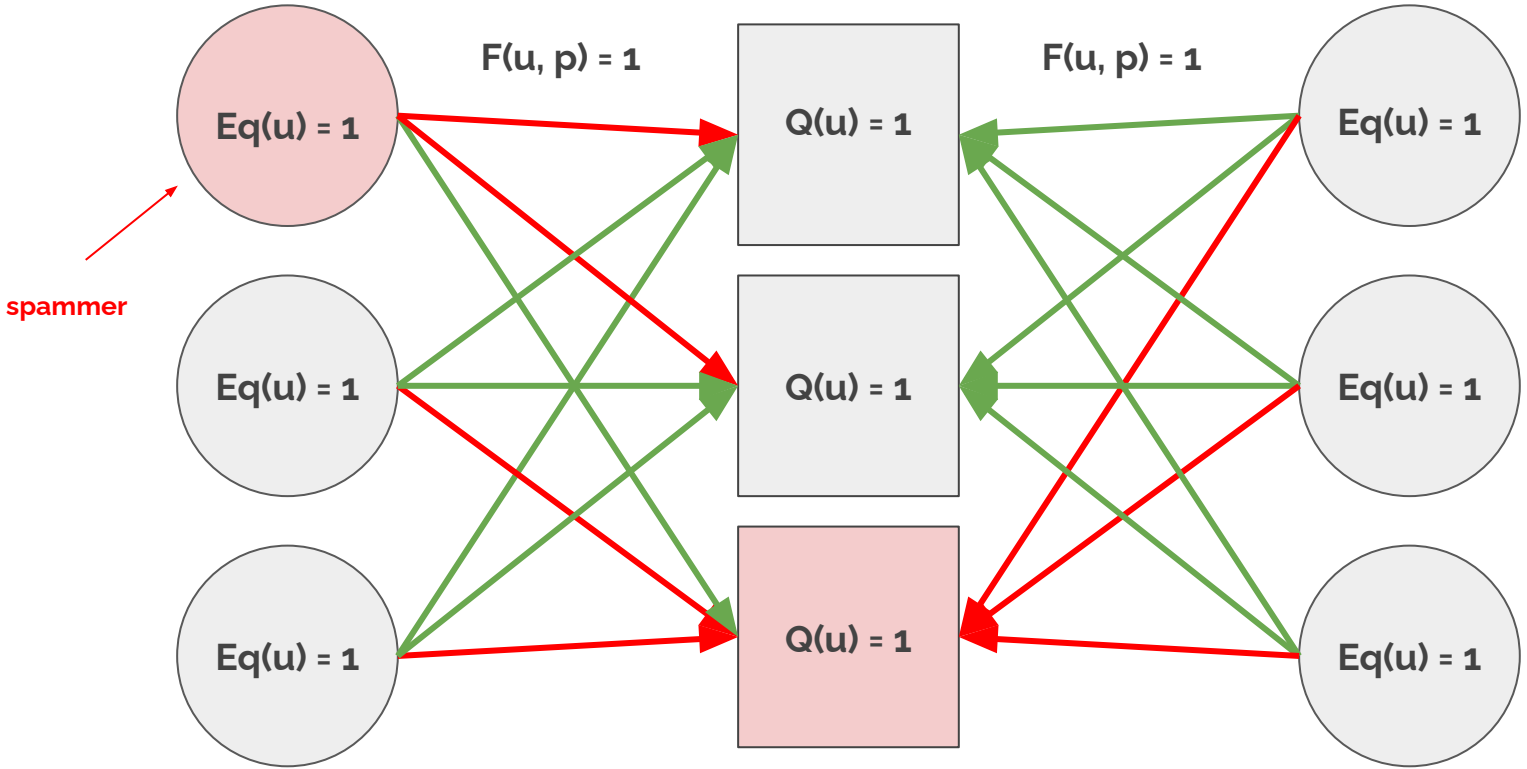
l'utilisateur est-il équitable ?

Y a-t-il une différence entre la note donnée par l'utilisateur et la qualité du produit ?

Intuitivement : plus un utilisateur est équitable et plus il note un produit "comme les autres utilisateurs", plus ses reviews sont fiables

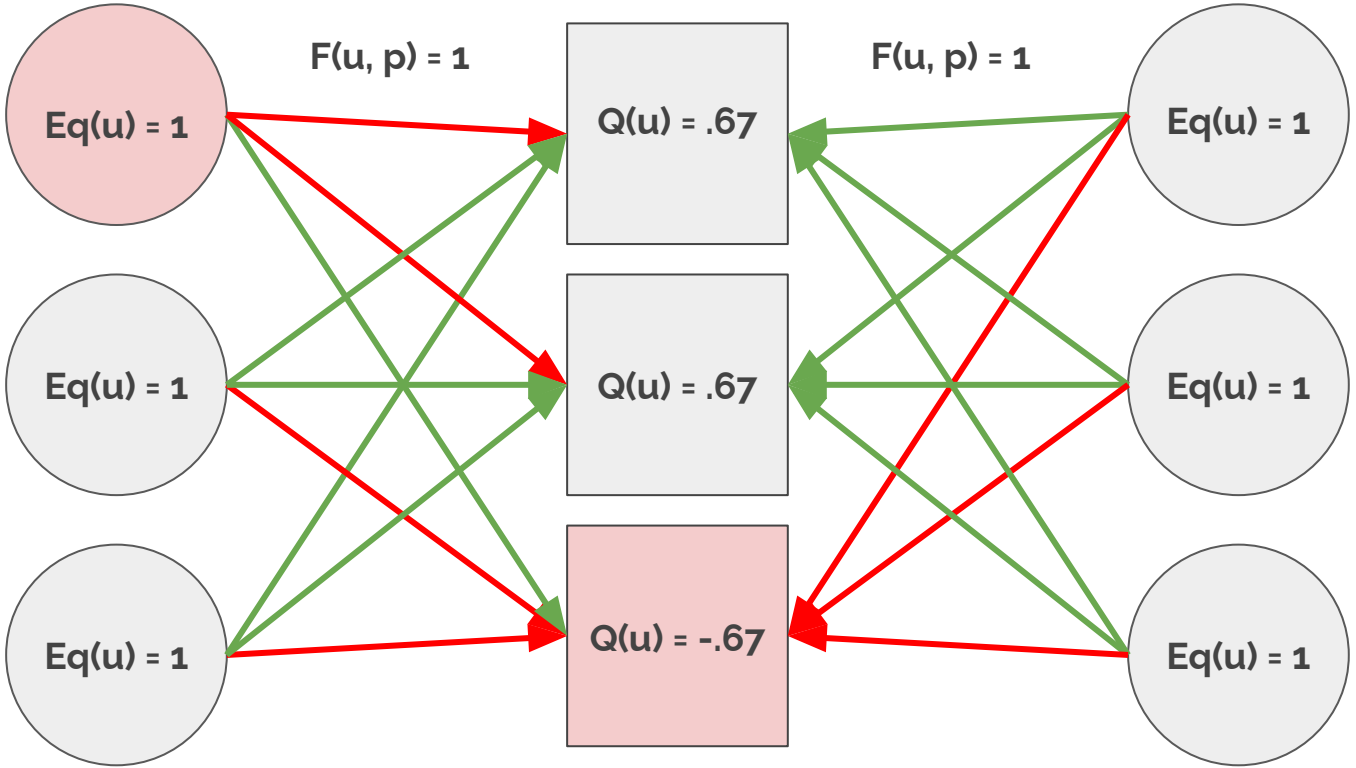
Déroulé de l'algorithme

On initialise toutes les valeurs à leur maximum



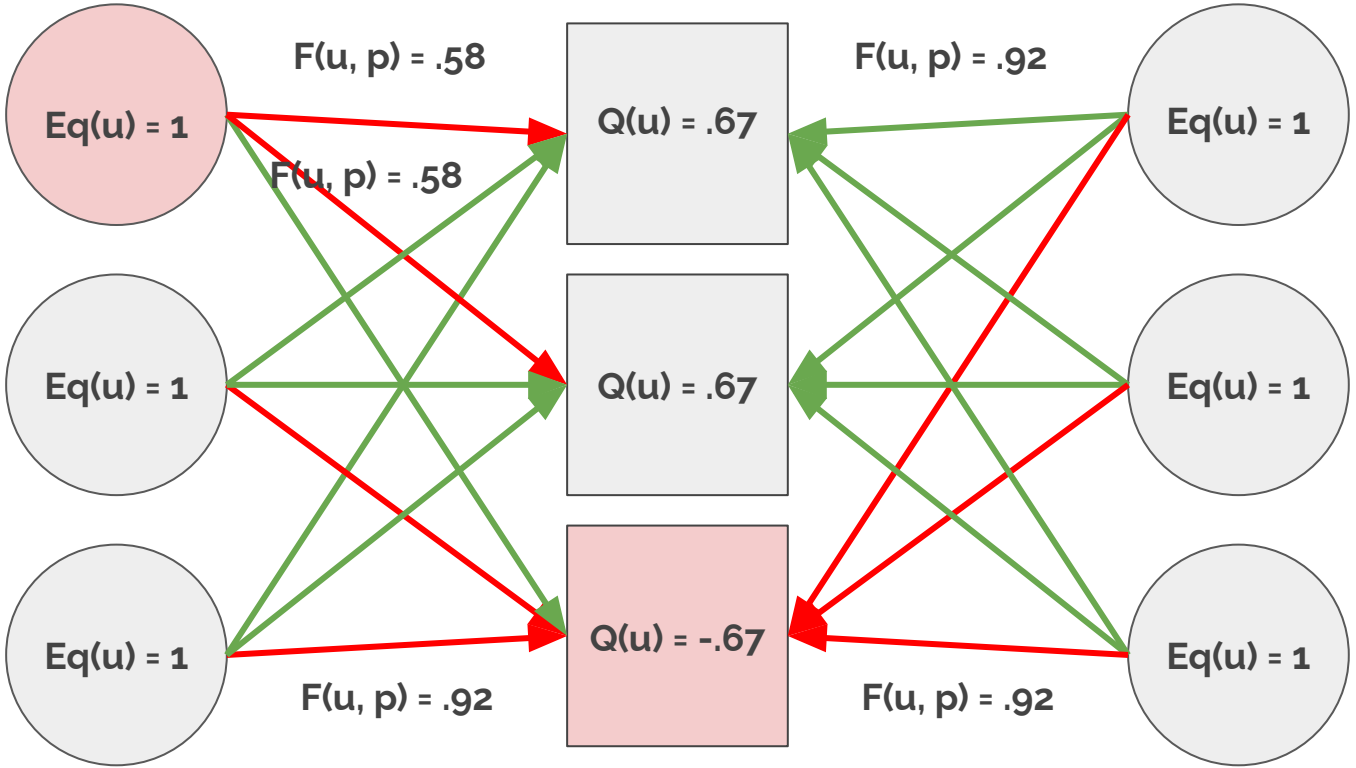
Déroulé de l'algorithme

Mise à jour de la **qualité** - itération 1



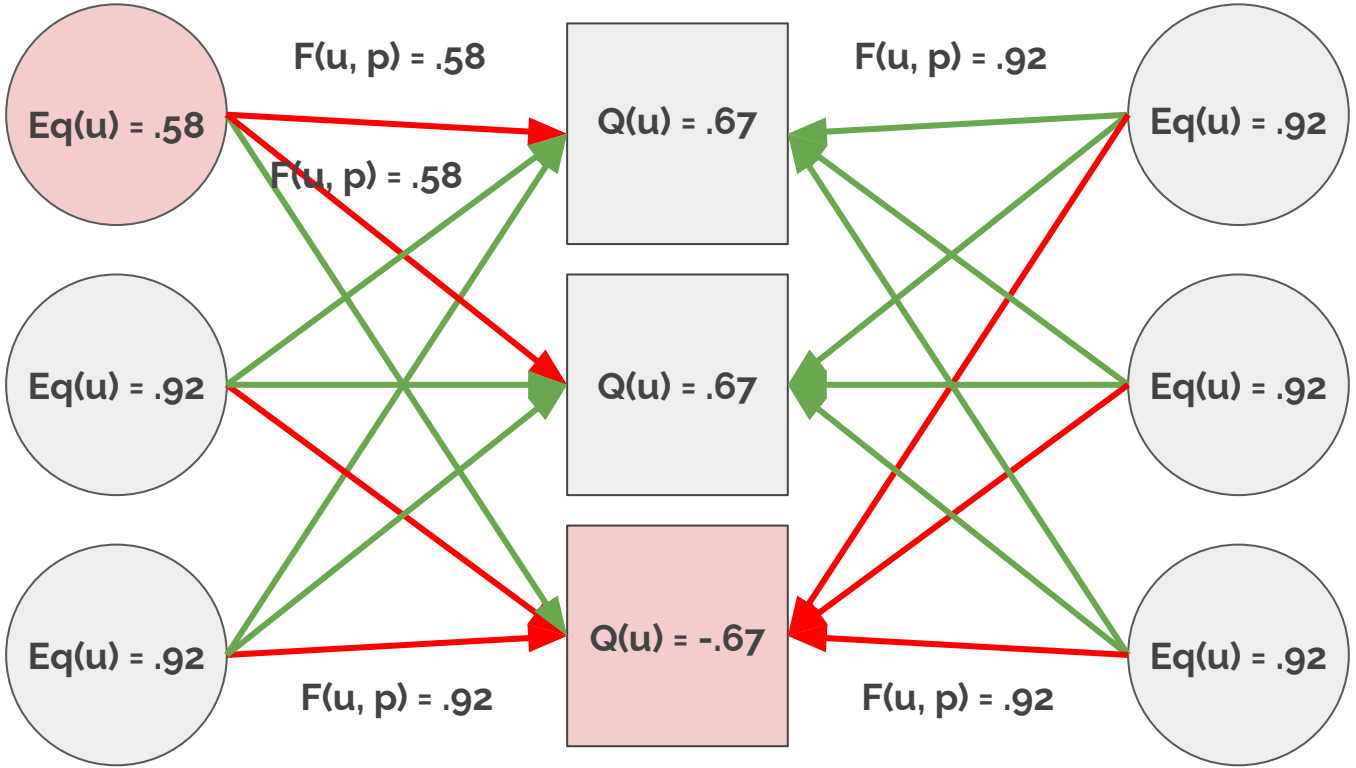
Déroulé de l'algorithme

Mise à jour de la **fiabilité** - itération 1



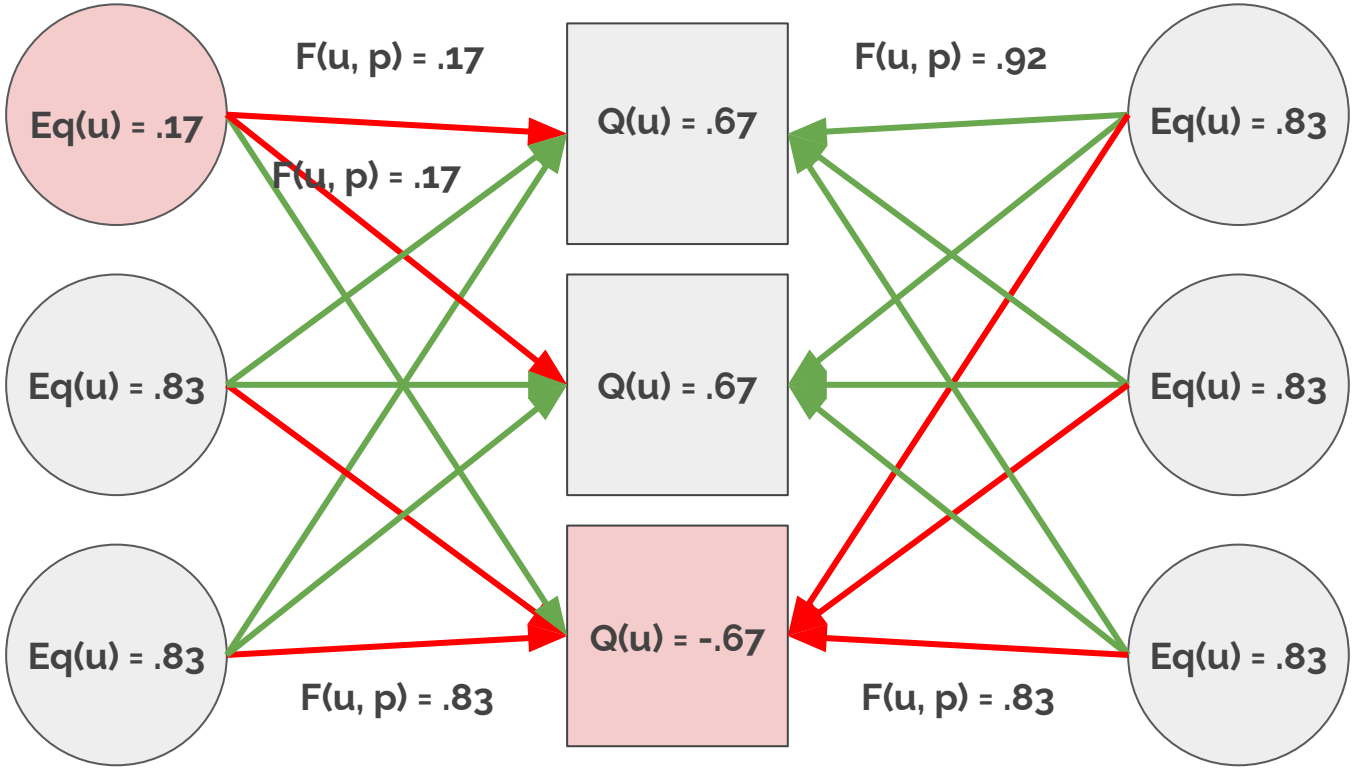
Déroulé de l'algorithme

Mise à jour de **l'équité** - itération 1



Déroulé de l'algorithme

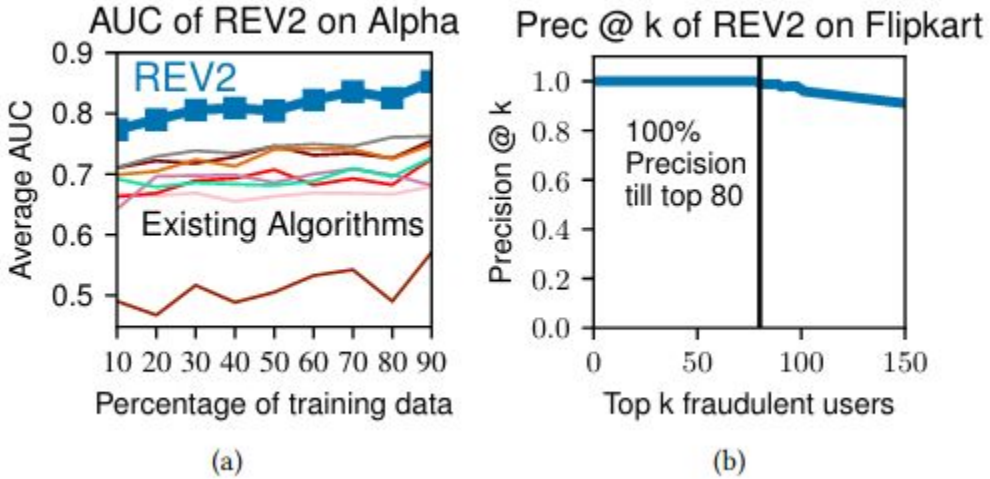
Après convergence



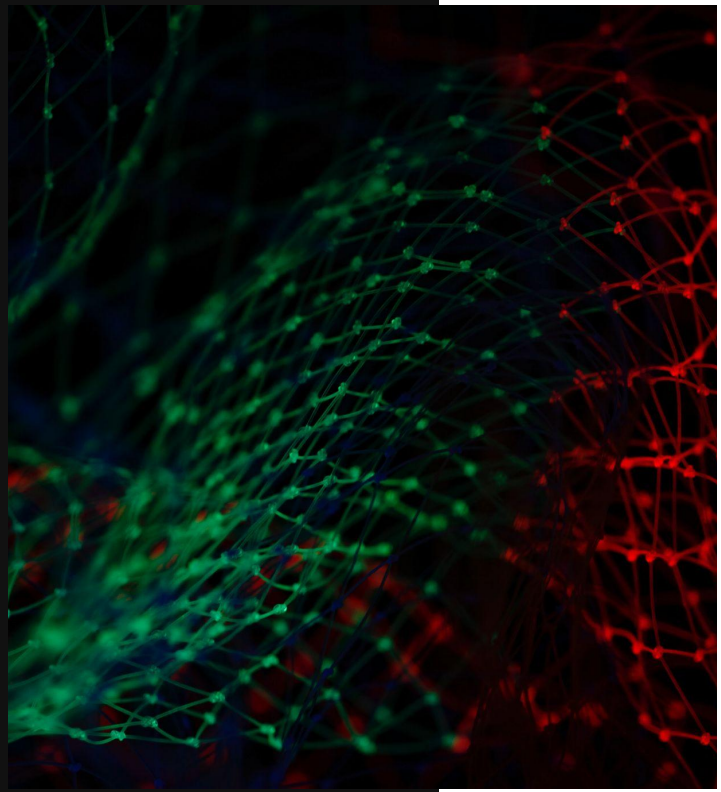
Propriétés de REV2

- convergence garantie
- le nombre d’itérations jusqu’à convergence est majorée
- complexité linéaire

Performances de REV2



Belief propagation



Loopy Belief Propagation

- utilisé pour estimer les probabilités marginales (beliefs) ou les états les plus probables des variables (noeuds)
- processus itératif dans lesquels les noeuds voisins “se passent des messages”

“Je (variable X_1) crois que toi (variable X_2) tu es dans l’état Y_2 avec telle probabilité.”

Dès qu’on aboutit à un consensus, on calcule la croyance finale

Message passing

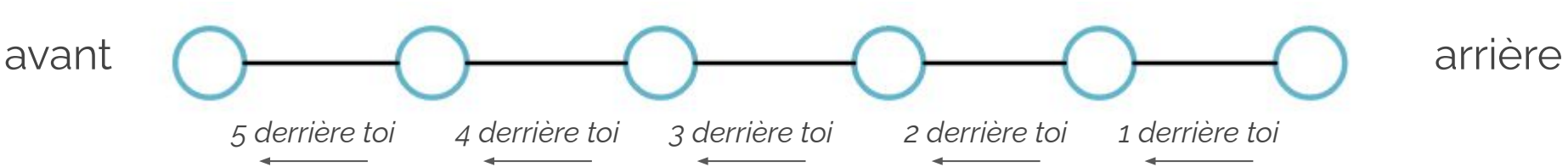
- **tâche :** compter le nombre de noeuds dans un graphe (sans cycle)
- **condition :** chaque noeud ne peut interagir qu'avec ses voisins
- **solution : chaque noeud écoute le message de ses voisins, le met à jour et le transmet.**

Message passing

- **tâche :** compter le nombre de noeuds dans un graphe (sans cycle)
- **condition :** chaque noeud ne peut interagir qu'avec ses voisins
- **solution : chaque noeud écoute le message de ses voisins, le met à jour et le transmet.**

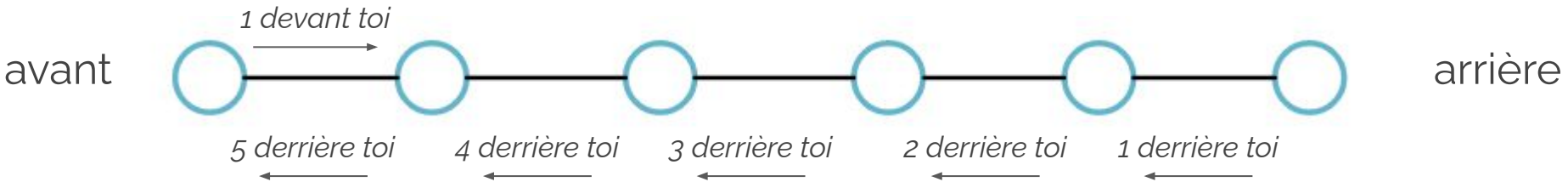
Message passing

- **tâche :** compter le nombre de noeuds dans un graphe (sans cycle)
- **condition :** chaque noeud ne peut interagir qu'avec ses voisins
- **solution : chaque noeud écoute le message de ses voisins, le met à jour et le transmet.**



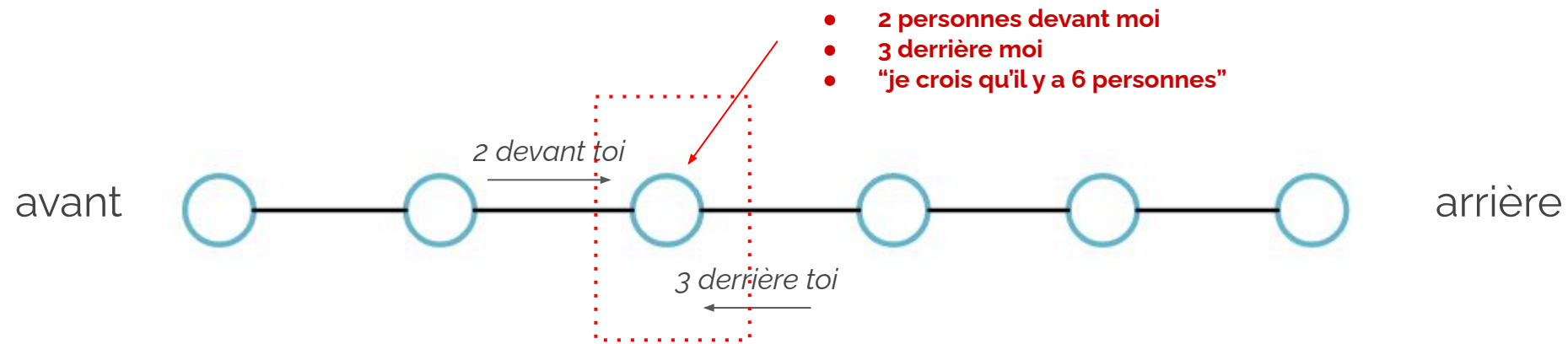
Message passing

- **tâche :** compter le nombre de noeuds dans un graphe (sans cycle)
- **condition :** chaque noeud ne peut interagir qu'avec ses voisins
- **solution : chaque noeud écoute le message de ses voisins, le met à jour et le transmet.**



Message passing

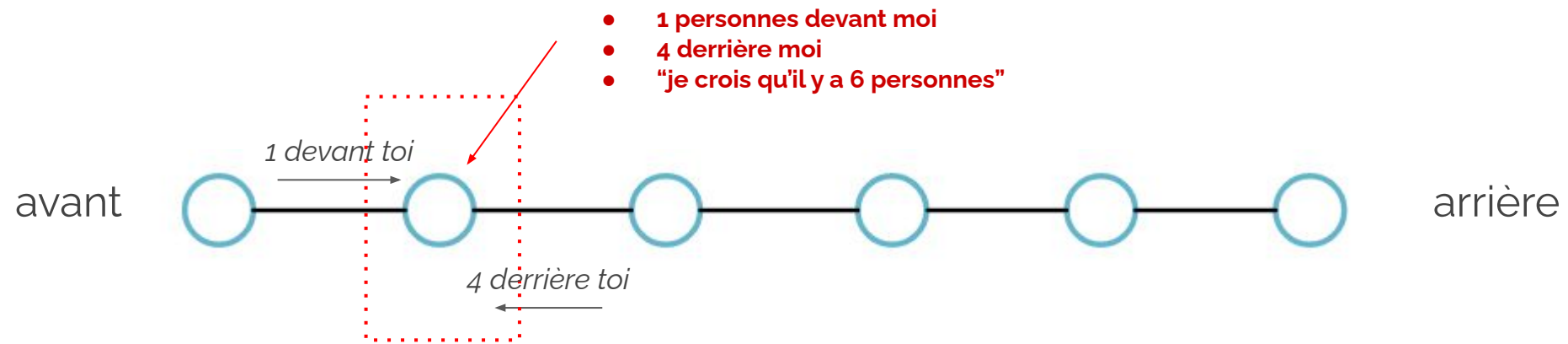
Un noeud n'a accès qu'aux messages qui lui sont transmis



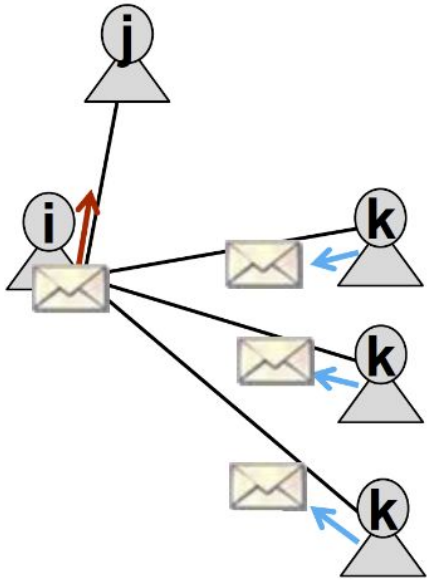
- **2 personnes devant moi**
- **3 derrière moi**
- **"je crois qu'il y a 6 personnes"**

Message passing

Un noeud n'a accès qu'aux messages qui lui sont transmis



Message passing



Quel message **i** va-t-il envoyer à **j** ?

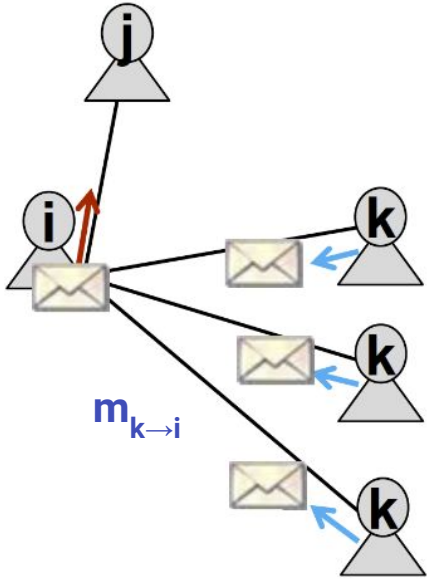
- cela dépend de ce que le noeud **i** entend de ses voisins **k**
- chaque voisin **k** passe un message à **i** : la conviction de **k** que **i** soit dans un état donné

Formalisation

- une **matrice potentielle** label-label **Ψ** : la dépendance/corrélation qui existe entre l'état des noeuds et l'état de son voisin :

  $\Psi(Y_i, Y_j)$ = probabilité que le noeud j soit dans l'état Y_j sachant qu'il a une voisin i dans l'état Y_i.
- **distribution a priori Φ** : probabilité $\Phi_i(Y_i)$ du noeud **i** d'être dans l'état Y_i
- une matrice des messages $m_{i \to j}(Y_j)$: l'estimation selon **i** de l'état Y_j de **j**
- **L** l'ensemble de tous les états possibles

L'algorithme du Loopy Belief Propagation



- Initialisation de tous les messages à 1
- On répète pour chaque noeud la mise à jour des messages :

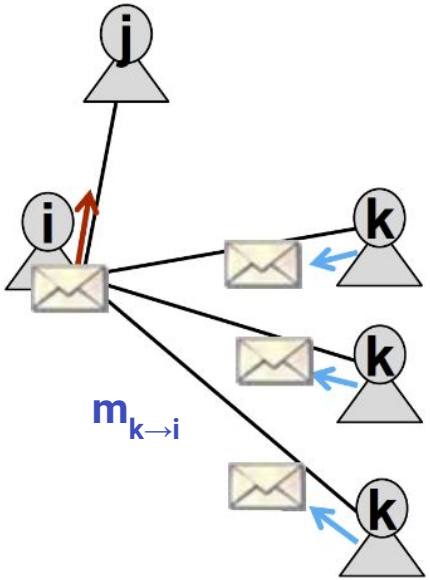
$$m_{i\to j}(Y_j) = \alpha \sum_{Y_i \in L} \Psi(Y_i, Y_j)\Phi(Y_i) \prod_{k\to i, k\neq j} m_{k\to i}(Y_i)$$

Conviction des voisins de i que i soit dans l'état Y_i

Probabilité que j soit dans l'état Y_j si i est dans l'état Y_i.

Probabilité a-priori que i soit dans l'état Y_i

L'algorithme du Loopy Belief Propagation



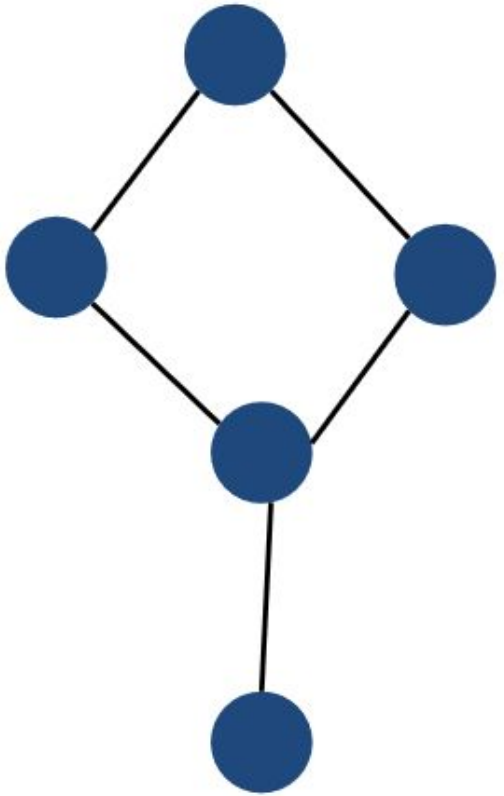
- après convergence

  $b_i(Y_i)$ = la conviction de i d'être dans l'état Y_i

$$b_i(Y_i) = \alpha\Phi(Y_i)\prod_{k\to i} m_{k\to i}(Y_i)$$

Boucles et belief propagation

- les messages des différents sous-graphes ne sont plus indépendants !
- les résultats ne seront peut-être pas corrects
- potentiels problèmes de convergence



Loopy Belief Propagation

Avantages :

- facile à coder, facile à paralléliser
- peut s'appliquer à la plupart des graphes

Inconvénients :

- convergence non garantie, surtout s'il y a de nombreuses boucles
- les fonctions potentielles (les paramètres) doivent être apprises ou estimées

Applications :

- chez Ebay, détection d'enchères frauduleuses

Machine learning sur les graphes : conclusion

Classification collective pour la classification de noeuds :

- Classification relationnelle
  - classifieur relationnel probabiliste
  - méthode des marches aléatoires
- Classification itérative
  - classifieur itératif utilisant les features des noeuds et la topologie du graphe
  - exemple : REV2
- Propagation de la conviction
  - Loopy Belief Propagation : message passing et mise à jour la conviction d'un noeud à partir des convictions de ses voisins