

Une introduction aux techniques de stream mining

Pascal Cheung

12 janvier 2005



Qu'est-ce que le Stream Mining ?

Qu'est-ce que le Stream Mining ?

Ensemble de techniques

- permettant d'analyser un flux de données,

Qu'est-ce que le Stream Mining ?

Ensemble de techniques

- permettant d'analyser un flux de données,
- quasiment en temps réel,

Qu'est-ce que le Stream Mining ?

Ensemble de techniques

- permettant d'analyser un flux de données,
- quasiment en temps réel,
- sans stocker l'intégralité du flux de données.

Exemples de flux de données

Exemples de flux de données

- Données échangées dans un **réseau de capteurs** (sismiques, océanographiques, de télésurveillance) : mesure de température, de pression...

Exemples de flux de données

- Données échangées dans un **réseau de capteurs** (sismiques, océanographiques, de télésurveillance) : mesure de température, de pression...
- Caractéristiques détaillées des **appels dans un réseau de télécommunication mobile** : durée de l'appel, fiabilité de la communication, volume de données échangées...

Exemples de flux de données

- Données échangées dans un **réseau de capteurs** (sismiques, océanographiques, de télésurveillance) : mesure de température, de pression...
- Caractéristiques détaillées des **appels dans un réseau de télécommunication mobile** : durée de l'appel, fiabilité de la communication, volume de données échangées...
- Caractéristiques des **requêtes reçues par un serveur web** : adresse IP source, page web demandée...

Exemples de flux de données

- Données échangées dans un **réseau de capteurs** (sismiques, océanographiques, de télésurveillance) : mesure de température, de pression...
- Caractéristiques détaillées des **appels dans un réseau de télécommunication mobile** : durée de l'appel, fiabilité de la communication, volume de données échangées...
- Caractéristiques des **requêtes reçues par un serveur web** : adresse IP source, page web demandée...
- Caractéristiques des **paquets IP transitant dans un réseau IP** : adresse source, adresse destination, protocole IP...

Exemples d'analyses

Exemples d'analyses

- construire des **histogrammes** : histogramme des durées d'appel, histogramme des volumes de données transportés par les paquets IP...

Exemples d'analyses

- construire des **histogrammes** : histogramme des durées d'appel, histogramme des volumes de données transportés par les paquets IP...
- rechercher les **items les plus fréquents** : adresses IP envoyant le plus de requêtes à un serveur, adresses destination les plus fréquentes parmi les paquets IP passant par un routeur donné...

Exemples d'analyses

- construire des **histogrammes** : histogramme des durées d'appel, histogramme des volumes de données transportés par les paquets IP...
- rechercher les **items les plus fréquents** : adresses IP envoyant le plus de requêtes à un serveur, adresses destination les plus fréquentes parmi les paquets IP passant par un routeur donné...
- effectuer une **analyse harmonique** : transformée en ondelette à fenêtre glissante de la pression ou de la température transmise par un capteur...

Contraintes sur l'analyse

- quasi temps réel : analyse menée à des fins de **surveillance** ou pour lancer des **alertes**
⇒ nécessité d'effectuer l'analyse en quasi temps réel et non *a posteriori*.

Contraintes sur l'analyse

- quasi temps réel : analyse menée à des fins de **surveillance** ou pour lancer des **alertes**
⇒ nécessité d'effectuer l'analyse en quasi temps réel et non *a posteriori*.
- pas de stockage du flux : l'espace de stockage est insuffisant pour stocker tout le flux car

Contraintes sur l'analyse

- **quasi temps réel** : analyse menée à des fins de **surveillance** ou pour lancer des **alertes**
⇒ nécessité d'effectuer l'analyse en quasi temps réel et non *a posteriori*.
- **pas de stockage du flux** : l'espace de stockage est insuffisant pour stocker tout le flux car
 - la quantité de données transmise par le flux est très élevée

Contraintes sur l'analyse

- **quasi temps réel** : analyse menée à des fins de **surveillance** ou pour lancer des **alertes**
⇒ nécessité d'effectuer l'analyse en quasi temps réel et non *a posteriori*.
- **pas de stockage du flux** : l'espace de stockage est insuffisant pour stocker tout le flux car
 - la quantité de données transmise par le flux est très élevée
 - l'espace de stockage disponible est très limité : systèmes embarqués, réseau de capteurs...

Conséquences de ces contraintes

Une analyse exacte est rarement possible. En général, on se contente d'une réponse approchée :

Conséquences de ces contraintes

Une analyse exacte est rarement possible. En général, on se contente d'une **réponse approchée** :

- Réponse approchée **déterministe** : la réponse approchée \hat{X} se trouve à une distance ϵ de la valeur réelle X

$$d(\hat{X}, X) \leq \epsilon$$

Conséquences de ces contraintes

Une analyse exacte est rarement possible. En général, on se contente d'une **réponse approchée** :

- Réponse approchée **déterministe** : la réponse approchée \hat{X} se trouve à une distance ϵ de la valeur réelle X

$$d(\hat{X}, X) \leq \epsilon$$

- Réponse approchée **probabiliste** : avec une probabilité d'au moins $1 - \delta$, la réponse approchée \hat{X} se trouve à une distance ϵ de la valeur réelle X

$$P\left\{d(\hat{X}, X) \leq \epsilon\right\} \geq 1 - \delta$$

**Un exemple d'algorithme de
stream mining :
l'algorithme Count-Min Sketch**

[Cormode & Muthukrishnan 2004]

Flux de données étudié

Chaque élément du flux étudié comporte deux champs :

- un **identifiant** $i_t \in \{1, 2, \dots, N\}$ avec N « grand »,
- une **marque** $c_t \in \mathbb{R}_+$

Le flux étudié est la suite $(i_t, c_t)_{t \in \mathbb{N}}$.

But de l'algorithme CMS

But de l'algorithme CMS

Pour chaque instant $\tau \in \mathbb{N}$ et pour chaque identifiant $i \in \{1, 2, \dots, N\}$, on définit le **compte** $a_i(\tau)$ de l'identifiant i à l'instant τ par

$$a_i(\tau) \stackrel{\text{déf}}{=} \sum_{t=0}^{\tau} c_t \delta_{i, i_t}$$

avec $\delta_{i, i_t} = 1$ si $i = i_t$, et $\delta_{i, i_t} = 0$ sinon.

But de l'algorithme CMS

Pour chaque instant $\tau \in \mathbb{N}$ et pour chaque identifiant $i \in \{1, 2, \dots, N\}$, on définit le **compte** $a_i(\tau)$ de l'identifiant i à l'instant τ par

$$a_i(\tau) \stackrel{\text{déf}}{=} \sum_{t=0}^{\tau} c_t \delta_{i, i_t}$$

avec $\delta_{i, i_t} = 1$ si $i = i_t$, et $\delta_{i, i_t} = 0$ sinon.

But de l'algorithme CMS : calculer un **estimateur** $\hat{a}_i(\tau)$ du compte $a_i(\tau)$ de l'identifiant i à l'instant τ .

Un exemple :

Le flux analysé $(i_t, c_t)_{t \in \mathbb{N}}$ correspond à des paquets IP reçus par un serveur :

Un exemple :

Le flux analysé $(i_t, c_t)_{t \in \mathbb{N}}$ correspond à des paquets IP reçus par un serveur :

- l'identifiant i_t est l'adresse source du paquet numéro t . Pour une adresse IPv4, la taille de l'espace d'adressage est alors $N = 2^{32}$.

Un exemple :

Le flux analysé $(i_t, c_t)_{t \in \mathbb{N}}$ correspond à des paquets IP reçus par un serveur :

- l'identifiant i_t est l'adresse source du paquet numéro t . Pour une adresse IPv4, la taille de l'espace d'adressage est alors $N = 2^{32}$.
- la marque c_t est le nombre d'octets transportés par le paquet numéro t .

Un exemple :

Le flux analysé $(i_t, c_t)_{t \in \mathbb{N}}$ correspond à des **paquets IP reçus par un serveur** :

- l'identifiant i_t est l'**adresse source** du paquet numéro t . Pour une adresse IPv4, la taille de l'espace d'adressage est alors $N = 2^{32}$.
- la marque c_t est le **nombre d'octets** transportés par le paquet numéro t .

Le compte $a_i(\tau)$ est le **nombre total d'octets envoyés à partir de l'adresse i à l'instant τ** .

Notion de fonction de hashage aléatoire uniforme au deuxième ordre

On appelle **fonction de hashage aléatoire uniforme au deuxième ordre** de $\{1, 2, \dots, N\}$ dans $\{1, 2, \dots, w\}$ (avec $w < N$) toute fonction aléatoire h de $\{1, 2, \dots, N\}$ dans $\{1, 2, \dots, w\}$ telle que, pour tout i et j distincts appartenant à $\{1, 2, \dots, N\}$, le **couple** $(h(i), h(j))$ suit une loi uniforme.

L'algorithme CMS

L'algorithme CMS

- On se fixe une précision $\epsilon > 0$ et une probabilité d'échec $\delta > 0$.

L'algorithme CMS

- On se fixe une précision $\epsilon > 0$ et une probabilité d'échec $\delta > 0$.
- On pose $w = \lceil \frac{e}{\epsilon} \rceil$ et $d = \lceil \ln \frac{1}{\delta} \rceil$.

L'algorithme CMS

- On se fixe une précision $\epsilon > 0$ et une probabilité d'échec $\delta > 0$.
- On pose $w = \lceil \frac{e}{\epsilon} \rceil$ et $d = \lceil \ln \frac{1}{\delta} \rceil$.
- On considère d fonctions de hashage aléatoires h_1, h_2, \dots, h_d de $\{1, 2, \dots, N\}$ dans $\{1, 2, \dots, w\}$, uniformes au deuxième ordre et indépendantes.

L'algorithme CMS

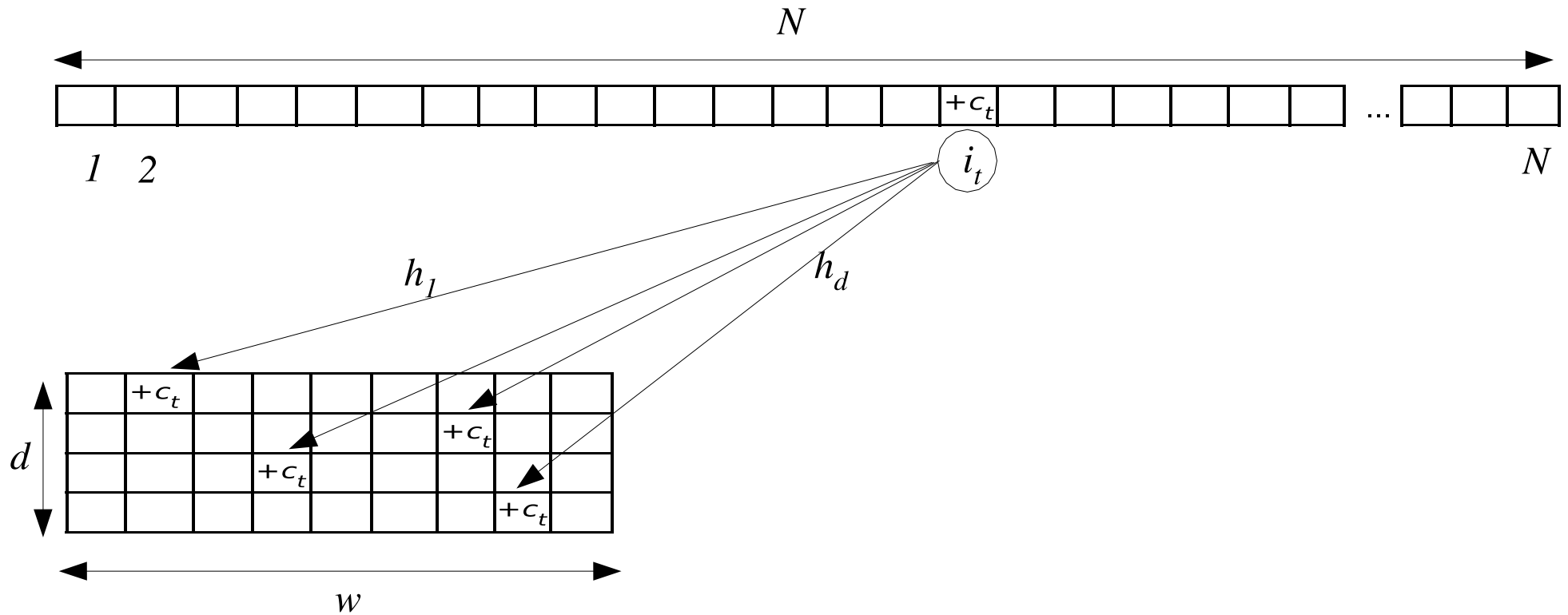
- On se fixe une précision $\epsilon > 0$ et une probabilité d'échec $\delta > 0$.
- On pose $w = \lceil \frac{e}{\epsilon} \rceil$ et $d = \lceil \ln \frac{1}{\delta} \rceil$.
- On considère d fonctions de hashage aléatoires h_1, h_2, \dots, h_d de $\{1, 2, \dots, N\}$ dans $\{1, 2, \dots, w\}$, uniformes au deuxième ordre et indépendantes.
- On alloue un tableau **Count** de taille $d \times w$, initialisé à 0.

L'algorithme CMS

- On se fixe une précision $\epsilon > 0$ et une probabilité d'échec $\delta > 0$.
- On pose $w = \lceil \frac{e}{\epsilon} \rceil$ et $d = \lceil \ln \frac{1}{\delta} \rceil$.
- On considère d fonctions de hashage aléatoires h_1, h_2, \dots, h_d de $\{1, 2, \dots, N\}$ dans $\{1, 2, \dots, w\}$, uniformes au deuxième ordre et indépendantes.
- On alloue un tableau **Count** de taille $d \times w$, initialisé à 0.
- Pour tout instant t , on met à jour le tableau **Count** ainsi :

pour tout $j \in \{1, 2, \dots, d\}$ **Count** $[j, h_j(i_t)] := \mathbf{Count}[j, h_j(i_t)] + c_t$

L'algorithme CMS



L'algorithme CMS

- A l'instant τ et pour chaque identifiant $i \in \{1, 2, \dots, N\}$, on obtient un estimateur $\hat{a}_i(\tau)$ du compte $a_i(\tau)$ ainsi

$$\hat{a}_i(\tau) = \min_j \mathbf{Count}[j, h_j(i)](\tau)$$

L'algorithme CMS

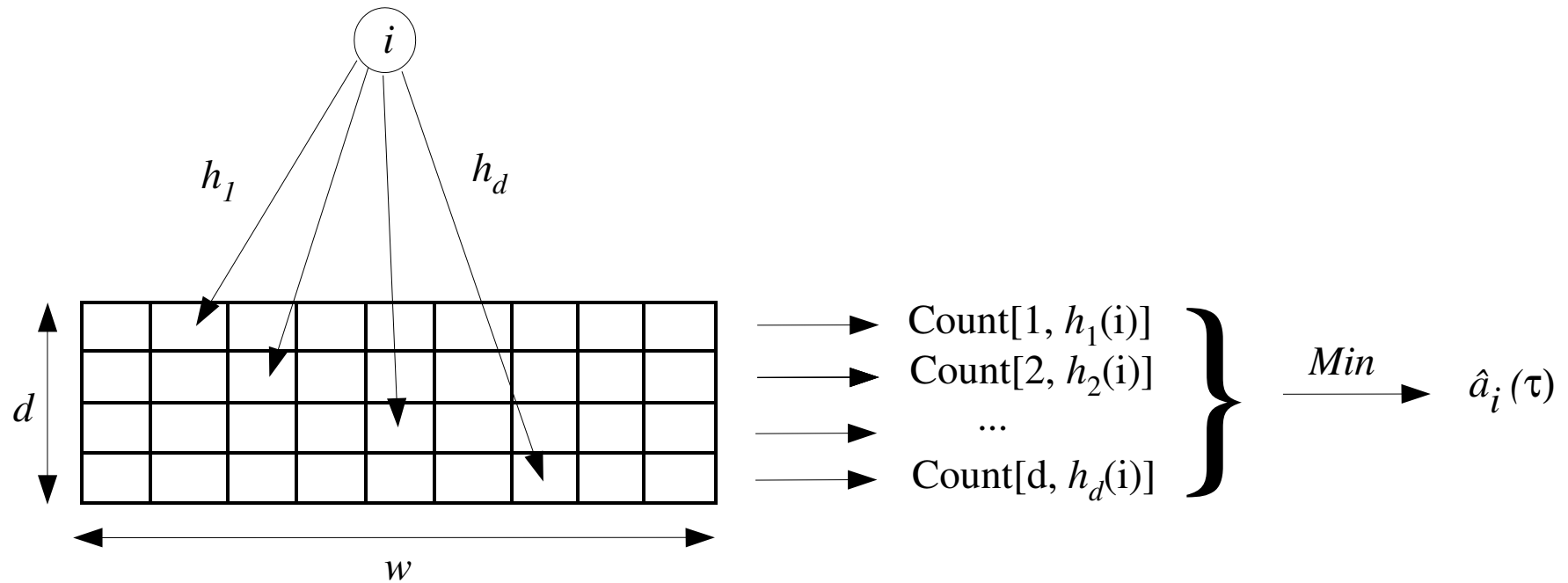


Tableau *Count* à l'instant τ

Complexité de l'algorithme CMS

Complexité de l'algorithme CMS

- Complexité en **temps** :
 - pour mettre à jour le tableau Count, le nombre d'opérations nécessaires est $O(\ln(\frac{1}{\delta}))$,
 - pour calculer \hat{a}_i , le nombre d'opérations nécessaires est aussi $O(\ln(\frac{1}{\delta}))$.

Complexité de l'algorithme CMS

- Complexité en **temps** :
 - pour mettre à jour le tableau Count, le nombre d'opérations nécessaires est $O(\ln(\frac{1}{\delta}))$,
 - pour calculer \hat{a}_i , le nombre d'opérations nécessaires est aussi $O(\ln(\frac{1}{\delta}))$.
- Complexité en **espace** :
 - pour stocker le tableau Count et les fonctions de hashage, le nombre de mots nécessaire est $O(\frac{1}{\epsilon} \ln(\frac{1}{\delta}))$.

Propriétés de l'estimateur $\hat{a}_i(\tau)$

- l'estimateur $\hat{a}_i(\tau)$ majore le compte vrai $a_i(\tau)$:

$$\hat{a}_i(\tau) \geq a_i(\tau)$$

Propriétés de l'estimateur $\hat{a}_i(\tau)$

- l'estimateur $\hat{a}_i(\tau)$ majore le compte vrai $a_i(\tau)$:

$$\hat{a}_i(\tau) \geq a_i(\tau)$$

Démonstration :

$$\begin{aligned} \forall j \quad \text{Count}[j, h_j(i)](\tau) &= a_i(\tau) + \underbrace{\sum_{t=0}^{\tau} c_t \delta_{h_j(i_t), h_j(i)} [1 - \delta_{i_t, i}]}_{\text{Collisions}} \\ &\geq a_i(\tau) \end{aligned}$$

Propriétés de l'estimateur $\hat{a}_i(\tau)$

- Avec une probabilité au moins $1 - \delta$, l'estimateur $\hat{a}_i(\tau)$ est majoré par $a_i(\tau) + \epsilon \|\vec{a}(\tau)\|_1$:

$$P \left\{ \hat{a}_i(\tau) \leq a_i(\tau) + \epsilon \|\vec{a}(\tau)\|_1 \right\} \geq 1 - \delta$$

avec

$$\vec{a}(\tau) \stackrel{\text{déf}}{=} (a_1(\tau), a_2(\tau), \dots, a_N(\tau))$$

$$\|\vec{a}(\tau)\|_1 \stackrel{\text{déf}}{=} |a_1(\tau)| + |a_2(\tau)| + \dots + |a_N(\tau)|.$$

Démonstration :

Majorons

$$P\left\{\hat{a}_i(\tau) > a_i(\tau) + \epsilon\|\vec{a}(\tau)\|_1\right\} = \prod_{j=1}^d P\left\{\text{Count}[j, h_j(i)](\tau) > a_i(\tau) + \epsilon\|\vec{a}(\tau)\|_1\right\}$$

Démonstration :

Majorons

$$P\left\{\hat{a}_i(\tau) > a_i(\tau) + \epsilon\|\vec{a}(\tau)\|_1\right\} = \prod_{j=1}^d P\left\{\text{Count}[j, h_j(i)](\tau) > a_i(\tau) + \epsilon\|\vec{a}(\tau)\|_1\right\}$$

Or, pour tout j ,

$$P\left\{\text{Count}[j, h_j(i)](\tau) > a_i(\tau) + \epsilon\|\vec{a}(\tau)\|_1\right\} = P\left\{X_{i,j}(\tau) > \epsilon\|\vec{a}(\tau)\|_1\right\}$$

où le term $X_{i,j}(\tau)$ correspond aux collisions avec l'identifiant i pour la j -ème fonction de hashage

$$X_{i,j}(\tau) \stackrel{\text{déf}}{=} \sum_{t=0}^{\tau} c_t \delta_{h_j(i_t), h_j(i)} [1 - \delta_{i_t, i}]$$

Démonstration :

Or on peut majorer l'espérance de $X_{i,j}(\tau)$ par un terme proportionnel à $\|\vec{a}(\tau)\|_1$:

$$\begin{aligned}\mathbb{E}\left\{X_{i,j}(\tau)\right\} &= \sum_{t=0}^{\tau} c_t [1 - \delta_{i_t,i}] \cdot \underbrace{\mathbb{E}\left\{\delta_{h_j(i_t),h_j(i)}\right\}}_{\text{vaut } \frac{1}{w} \text{ si } i_t \neq i} \\ &\leq \frac{1}{w} \sum_{t=0}^{\tau} c_t \\ &\leq \frac{1}{w} \|\vec{a}(\tau)\|_1\end{aligned}$$

Démonstration :

Or on peut majorer l'espérance de $X_{i,j}(\tau)$ par un terme proportionnel à $\|\vec{a}(\tau)\|_1$:

$$\begin{aligned}\mathbb{E}\left\{X_{i,j}(\tau)\right\} &= \sum_{t=0}^{\tau} c_t [1 - \delta_{i_t,i}] \cdot \underbrace{\mathbb{E}\left\{\delta_{h_j(i_t),h_j(i)}\right\}}_{\text{vaut } \frac{1}{w} \text{ si } i_t \neq i} \\ &\leq \frac{1}{w} \sum_{t=0}^{\tau} c_t \\ &\leq \frac{1}{w} \|\vec{a}(\tau)\|_1\end{aligned}$$

D'où

$$\begin{aligned}P\left\{X_{i,j}(\tau) > \epsilon \|\vec{a}(\tau)\|_1\right\} &\leq P\left\{X_{i,j}(\tau) > \epsilon w \mathbb{E}\left\{X_{i,j}(\tau)\right\}\right\} \\ &\leq P\left\{X_{i,j}(\tau) > e \mathbb{E}\left\{X_{i,j}(\tau)\right\}\right\} \quad \text{car } w = \lceil \frac{e}{\epsilon} \rceil \\ &\leq \frac{1}{e} \quad (\text{Inégalité de Markov})\end{aligned}$$

Démonstration :

On en déduit

$$\begin{aligned} P\left\{\hat{a}_i(\tau) > a_i(\tau) + \epsilon\|\vec{a}(\tau)\|_1\right\} &= \prod_{j=1}^d P\left\{X_{i,j}(\tau) > \epsilon\|\vec{a}(\tau)\|_1\right\} \\ &\leq \frac{1}{e^d} \\ &\leq \delta \quad \text{car } d = \lceil \ln \frac{1}{\delta} \rceil \end{aligned}$$

Démonstration :

On en déduit

$$\begin{aligned} P\left\{\hat{a}_i(\tau) > a_i(\tau) + \epsilon\|\vec{a}(\tau)\|_1\right\} &= \prod_{j=1}^d P\left\{X_{i,j}(\tau) > \epsilon\|\vec{a}(\tau)\|_1\right\} \\ &\leq \frac{1}{e^d} \\ &\leq \delta \quad \text{car } d = \lceil \ln \frac{1}{\delta} \rceil \end{aligned}$$

ce qui implique

$$P\left\{\hat{a}_i(\tau) \leq a_i(\tau) + \epsilon\|\vec{a}(\tau)\|_1\right\} \geq 1 - \delta \quad \square$$

Application de l'algorithme CMS : recherche d'objets massifs

Application de l'algorithme CMS : recherche d'objets massifs

A un instant τ donné et pour un seuil $\phi \in]0, 1[$ donné, on appelle **objet massif** tout identifiant i tel que

$$a_i(\tau) \geq \phi \|\vec{a}(\tau)\|_1.$$

Application de l'algorithme CMS : recherche d'objets massifs

A un instant τ donné et pour un seuil $\phi \in]0, 1[$ donné, on appelle **objet massif** tout identifiant i tel que

$$a_i(\tau) \geq \phi \|\vec{a}(\tau)\|_1.$$

On montre que le nombre d'objets massifs ne peut pas dépasser $\lfloor \frac{1}{\phi} \rfloor$.

Algorithme de recherche d'objets massifs

Initialisation

A l'instant $t = 0$, le couple (i_0, c_0) est traité ainsi :

Algorithme de recherche d'objets massifs

Initialisation

A l'instant $t = 0$, le couple (i_0, c_0) est traité ainsi :

- on calcule le compte total $\|\vec{a}(0)\|_1$:

$$\|\vec{a}(0)\|_1 = c_0$$

Algorithme de recherche d'objets massifs

Initialisation

A l'instant $t = 0$, le couple (i_0, c_0) est traité ainsi :

- on calcule le compte total $\|\vec{a}(0)\|_1$:

$$\|\vec{a}(0)\|_1 = c_0$$

- on met à jour le tableau **Count**

Algorithme de recherche d'objets massifs

Initialisation

A l'instant $t = 0$, le couple (i_0, c_0) est traité ainsi :

- on calcule le compte total $\|\vec{a}(0)\|_1$:

$$\|\vec{a}(0)\|_1 = c_0$$

- on met à jour le tableau **Count**
- on ajoute l'identifiant i_0 avec son compte estimé $\hat{a}_{i_0}(0)$ à une liste \mathcal{L} d'objets massifs potentiels.

Algorithme de recherche d'objets massifs

Itération

A l'instant t , le couple (i_t, c_t) est traité ainsi :

Algorithme de recherche d'objets massifs

Itération

A l'instant t , le couple (i_t, c_t) est traité ainsi :

- on calcule le compte total $\|\vec{a}(t)\|_1$:

$$\|\vec{a}(t)\|_1 = \|\vec{a}(t-1)\|_1 + c_t$$

Algorithme de recherche d'objets massifs

Itération

A l'instant t , le couple (i_t, c_t) est traité ainsi :

- on calcule le compte total $\|\vec{a}(t)\|_1$:

$$\|\vec{a}(t)\|_1 = \|\vec{a}(t-1)\|_1 + c_t$$

- on met à jour le tableau **Count**

Algorithme de recherche d'objets massifs

Itération

A l'instant t , le couple (i_t, c_t) est traité ainsi :

- on calcule le compte total $\|\vec{a}(t)\|_1$:

$$\|\vec{a}(t)\|_1 = \|\vec{a}(t-1)\|_1 + c_t$$

- on met à jour le tableau **Count**
- on calcule l'estimateur $\hat{a}_{i_t}(t)$ du compte de l'identifiant i_t

Algorithme de recherche d'objets massifs

Itération

A l'instant t , le couple (i_t, c_t) est traité ainsi :

- on calcule le compte total $\|\vec{a}(t)\|_1$:

$$\|\vec{a}(t)\|_1 = \|\vec{a}(t-1)\|_1 + c_t$$

- on met à jour le tableau **Count**
- on calcule l'estimateur $\hat{a}_{i_t}(t)$ du compte de l'identifiant i_t
- si $\hat{a}_{i_t}(t) \geq \phi \|\vec{a}(t)\|_1$, alors :
 - si l'identifiant i_t n'appartient à la liste \mathcal{L} :
l'identifiant i_t avec son compte estimé $\hat{a}_{i_t}(t)$ est rajouté à la liste \mathcal{L}
 - si l'identifiant i_t appartient à la liste \mathcal{L} :
l'ancien compte estimé de l'identifiant i_t stocké avec l'identifiant i_t est remplacé par le compte estimé $\hat{a}_{i_t}(t)$.

Algorithme de recherche d'objets massifs

Itération

A l'instant t , le couple (i_t, c_t) est traité ainsi :

- on calcule le compte total $\|\vec{a}(t)\|_1$:

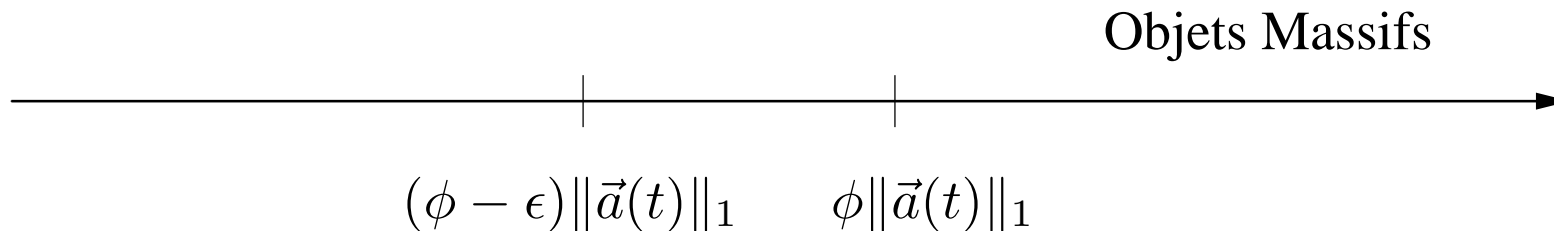
$$\|\vec{a}(t)\|_1 = \|\vec{a}(t-1)\|_1 + c_t$$

- on met à jour le tableau **Count**
- on calcule l'estimateur $\hat{a}_{i_t}(t)$ du compte de l'identifiant i_t
- si $\hat{a}_{i_t}(t) \geq \phi \|\vec{a}(t)\|_1$, alors :
 - si l'identifiant i_t n'appartient à la liste \mathcal{L} :
l'identifiant i_t avec son compte estimé $\hat{a}_{i_t}(t)$ est rajouté à la liste \mathcal{L}
 - si l'identifiant i_t appartient à la liste \mathcal{L} :
l'ancien compte estimé de l'identifiant i_t stocké avec l'identifiant i_t est remplacé par le compte estimé $\hat{a}_{i_t}(t)$.
- on élimine de la liste \mathcal{L} tous les identifiants dont le compte est strictement inférieur à $\phi \|\vec{a}(t)\|_1$.

Algorithme de recherche d'objets massifs

Propriétés de cet algorithme

- La liste \mathcal{L} contient tous les objets massifs du flux.
- La liste \mathcal{L} ne contient que des identifiants dont le compte réel est supérieur à $(\phi - \epsilon) \|\vec{a}(t)\|_1$, avec une probabilité d'au moins $1 - \delta$.



Conclusion

Conclusion

- Le stream mining permet de répondre à de nombreux problèmes dans le cas où :
 - les données doivent être traitées en une seule passe
 - l'espace mémoire est limitée
 - une réponse approchée est acceptable

Conclusion

- Le stream mining permet de répondre à de nombreux problèmes dans le cas où :
 - les données doivent être traitées en une seule passe
 - l'espace mémoire est limitée
 - une réponse approchée est acceptable
- Le stream mining est un domaine dans lequel France Télécom cherche à développer son expertise.

Questions ?