femto-st SCIENCES & TECHNOLOGIES

Sécurité Appliquée-PVP Confidentialité différentielle locale ϵ-LDP

Jean-François COUCHOT
Université de Franche-Comté, UFR-ST

26 novembre 2023

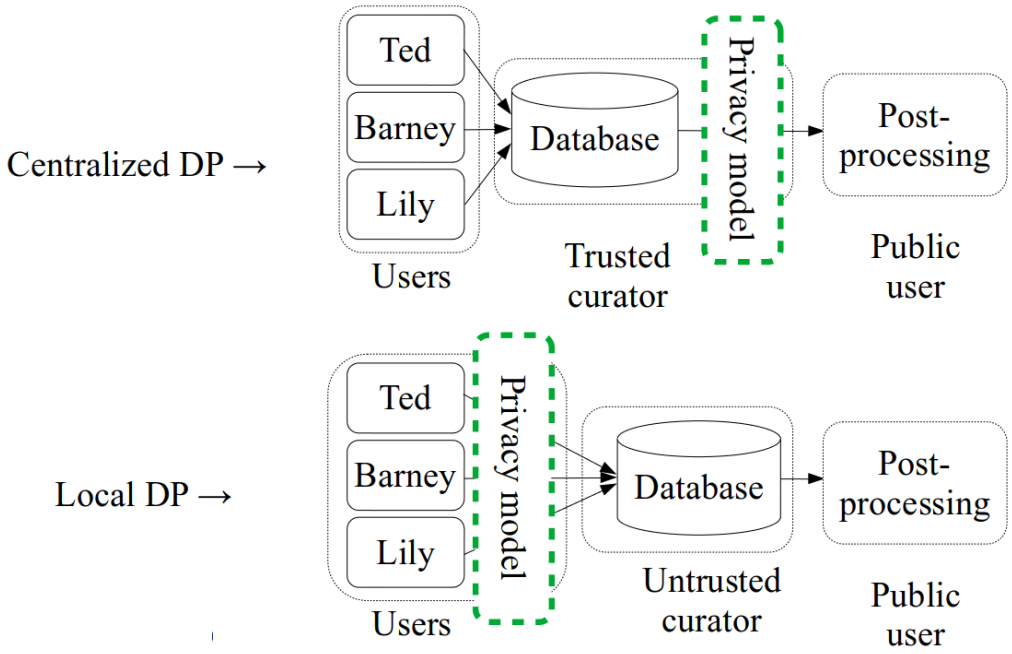
cnrs UBFC UNIVERSITÉ BOURGOGNE-FRANCHE-COMTÉ UNIVERSITÉ DE FRANCHE-COMTÉ SUP MICRO TECH ENSMM utbm université de technologie Belfort-Montbéliard

Plan

Plan

Motivations

En images



DP vs LDP

- Confiance nécessaire envers le SGBD
- Bruit optimal par requête
- Bruit individuel pour tous les post-traitements (ML p.ex)
- Confiance superflue envers le SGBD

Définition [1] et propriétés

Définition de ϵ-confidentialité différentielle locale (ϵ-LDP)

- $\mathcal{X}$: ensemble des valeurs possibles en entrée
- $\epsilon \in \mathbb{R}^+$: budget de fuite
- $\mathcal{M}$: algorithme probabiliste non déterministe qui respecte la ϵ-confidentialité différentielle locale si

$\forall x_1, x_2 \in \mathcal{X}$ (x_1 et x_2 deux données d'entrée)
$\forall y$ t.q. $y \in \mathcal{M}(\mathcal{X})$, (pour tte image y de l'algo.)
$\Pr[\mathcal{M}(x_1) = y] \leq e^{\epsilon} \Pr[\mathcal{M}(x_2) = y]$

Propriétés similaires à celles de la DP

- Robustesse au post-traitement
- Combinaison de deux mécanismes LDP : est LDP

1. Duchi, J. C., Jordan, M. I., & Wainwright, M. J. (2013, October). Local privacy and statistical minimax rates. In 2013 IEEE 54th Annual Symposium on Foundations of Computer Science (pp. 429-438). IEEE.

Motivation : données embarrassante à nettoyer [3]

Table avec 1 seul attribut binaire : Q_1="avez-vous triché au moins une fois ?"

- Embarras : tentation pour un·e étudiant·e de ne pas répondre honnêtement.

Bruiter selon Warner [2]

- Chaque étudiant·e lance 2 fois une pièce de monnaie $\{\text{Pile}, \text{Face}\}$ sans montrer les 2 résultat successifs t_1 et t_2.
- Ajout de la question Q_2 : « Est-ce que t_2 est égal à Pile ? ».
  - Si t_1 vaut Pile, l'étudiant·e répond honnêtement à la question Q_1.
  - Sinon ($t_1 = \text{Face}$), l'étudiant·e répond honnêtement à la question Q_2.

Analyse de l'extension

- Réponse partiellement aléatoire : on ne sait pas si une réponse OUI d'un·e étudiant·e provient d'une tricherie ou d'un Pile au second tirage.
- Honnêteté de l'étudiant·e renforcée : c'est lui·elle qui modifie ses données.

---

2. Warner, S. L. (1965). Randomized response : A survey technique for eliminating evasive answer bias. Journal of the American Statistical Association, 60(309), 63-69.

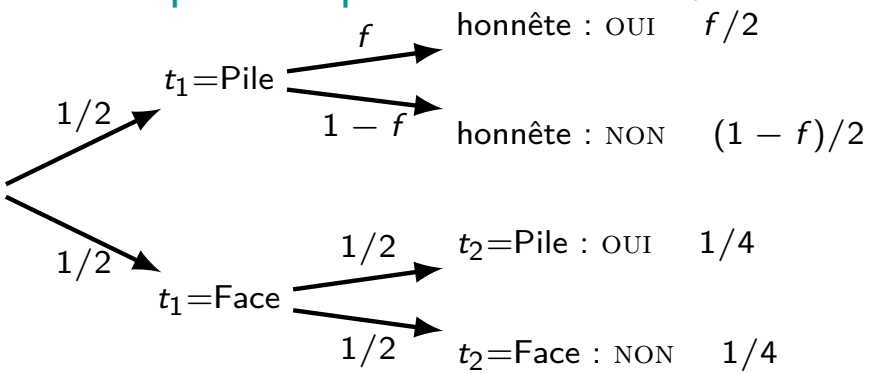
3. `https://fr.coursera.org/lecture/stanford-statistics/warners-randomized-response-model-ck65q`

Motivation : estimation du pourcentage de tricheurs

Point clef

- Un OUI *individuel* : on ne connaît pas exactement son origine.
- Après calcul du pourcentage *global* des OUI : on doit pouvoir estimer le pourcentage des étudiants ayant triché au moins une fois.

Estimateur pour N personnes et $f < N$ tricheurs



- Fréquence observée de OUI : $r \approx 1/4 + f/2$
- Estimation $\hat{f}$ du nbre. original de OUI : $\hat{f} = 2r - 1/2$

x \ y	OUI	NON
OUI	3/4	1/4
NON	1/4	3/4

- $\frac{\Pr[\mathcal{M}(x_1)=y]}{\Pr[\mathcal{M}(x_2)=y]} \leq \frac{\Pr[\mathcal{M}(\text{OUI})=\text{OUI}]}{\Pr[\mathcal{M}(\text{OUI})=\text{OUI}]} \leq 3$
- $\rightsquigarrow$ Mécanisme ln(3)-LDP

Questions en suspens

- L'estimateur est-il biaisé ? On peut espérer que non !
- Quid de la variance ?
- Et si on utilisait d'autres probabilités que 1/2, 1/2 ?

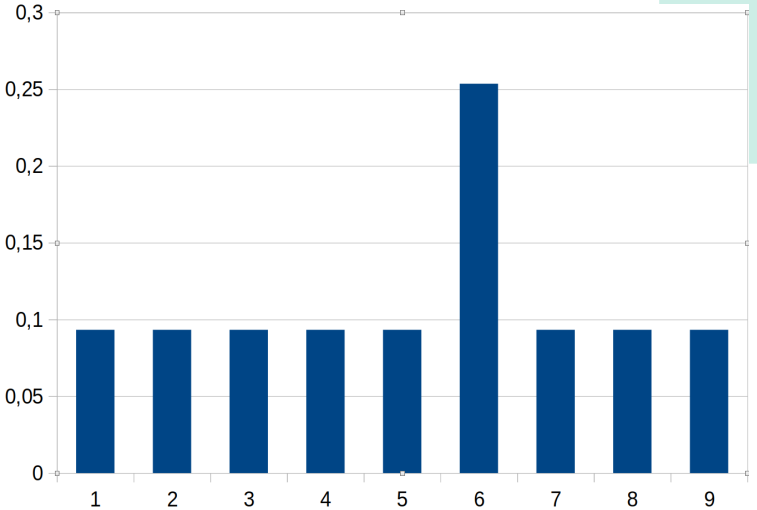
Plan

Données catégorielles : $\mathcal{M}_{GRR}$

Réponse Randomisée Généralisée [4] (GRR) dans un domaine de k éléments

Pour une entrée x :

- réponse aléatoire selon une probabilité uniforme c parmi les $k-1$ autres propositions.
- réponse avec une probabilité de $c.e^{\epsilon}$ de préserver l'entrée.



$\epsilon = 1,\ k = 9,\ x = 6$

Définition de $\mathcal{M}_{GRR}$

- $\Pr[\mathcal{M}_{GRR}(x) = r] = \begin{cases} c.e^{\epsilon} & \text{pour } r = x \\ c & \text{sinon} \end{cases}$ avec $c = \dfrac{1}{k - 1 + e^{\epsilon}}$
- Preuve d'ϵ-LDP : évidente ! Pourquoi ?

4. Kairouz, P., Bonawitz, K., & Ramage, D. (2016). Discrete distribution estimation under local privacy. arXiv preprint arXiv :1602.07387.

Données continues, encore Laplace

Intervalle continu d'amplitude Δ : mécanisme Laplacien borné $\mathcal{M}_{Lb}$

- $\mathcal{M}_{Lb}(x) = x + v$ t.q. $v \sim Lap(\frac{\Delta}{\epsilon})$
- si $x + v$ pas dans l'intervalle, réappliquer $\mathcal{M}_{Lb}$
- Preuve d'ϵ-LDP : similaire, à celle d'ϵ-DP
- Δ : ici pas la sensibilité (insensé !) mais l'amplitude de l'intervalle

Nettoyage naïf avec Laplace borné et GRR -1

```python
import pandas as pd
import numpy as np
import math as m

from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score

df = pd.read_csv(path+'diabetes.csv')

# Replace Zeroes with the median value of the column
for column in ['Glucose', 'BloodPressure', 'SkinThickness', 'BMI', 'Insulin']:
   df[column].replace(0, np.NaN,inplace=True)
   df[column].fillna(df[column].median(),inplace=True)

# Replace Outcome
label_encoder = LabelEncoder()
df['Outcome']= label_encoder.fit_transform(df["Outcome"])

# working with attributes
description = df.describe()
integerAttributes = ['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','Age']
floatAttributes = ['BMI', 'DiabetesPedigreeFunction']
discreteAttributes = ['Outcome']
discVals ={k : df[k].unique() for k in discreteAttributes}
```

Nettoyage naïf avec Laplace borné et GRR -2

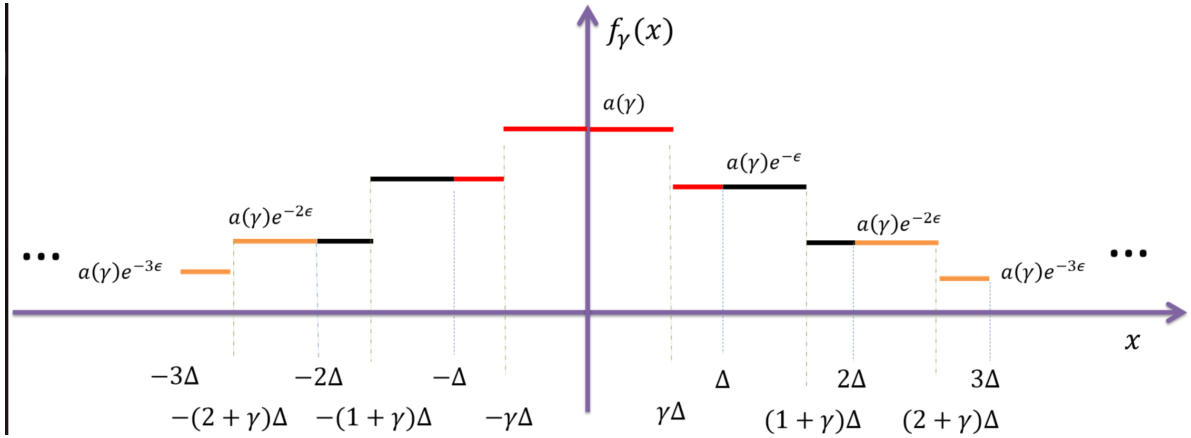
```python
def addBoundedLaplaceNoise(val,mi,mx,eps=1,isInteger=False):
  Delta = mx-mi
  insideBounds = False
  while (not insideBounds):
    r = val + np.random.laplace(0,Delta/eps)
    insideBounds  = (r <= mx) and (r>= mi)
    return int(round(r)) if isInteger else r

def generalizedRandomResponse(val,eps=1,othervals=[],k=2):
  p = (m.exp(eps)/(k-1+m.exp(eps)))
  return val if np.random.rand() < p else np.random.choice(othervals)

def LDP(eps,description,integerAttributes,floatAttributes,discreteAttributes,discVals):
    dfp = pd.DataFrame()
    for att in integerAttributes:
      mi, mx = description[att]['min'], description[att]['max']
      dfp[att] = df[att].map(lambda a: addBoundedLaplaceNoise(a,mi,mx,eps,True))

    for att in floatAttributes:
      mi, mx = description[att]['min'], description[att]['max']
      dfp[att] = df[att].map(lambda a: addBoundedLaplaceNoise(a,mi,mx,eps,False))

    for att in discreteAttributes:
      dfp[att] = df[att].map(
          lambda a: generalizedRandomResponse(a,eps,
                                              list(set(discVals[att])-set([a])),
                                              len(discVals[att])))
    return dfp
```

Nettoyage naïf avec Laplace borné et GRR -3

```
1  dfp = LDP(1,description,integerAttributes,
2            floatAttributes,discreteAttributes,discVals) #valeur du eps global?
3
4  X = dfp.iloc[:, 0:8]
5  y = dfp.iloc[:, 8]
6  X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0, test_size=0.20)
7
8  gnb = GaussianNB()
9  gnb.fit(X_train.values, y_train)
10 y_pred = gnb.predict(X_test.values)
11
12 print(accuracy_score(y_test, y_pred))
13 print(f1_score(y_test, y_pred))
14
15 # 0.6038961038961039 vs 0.7857142857142857
16 # 0.11594202898550726 vs 0.6373626373626374
```

Données continues, par paliers

Des paliers[5] pour uniformiser localement le bruit



Formalisation

$\mathcal{M}_{\text{Stair}}(x) = x + v$, v de pdf $\begin{cases} \alpha(\gamma) & \text{si} |z| < \gamma\Delta \\ \alpha(\gamma) b^j & \text{si} |z| \in [(j+\gamma)\Delta, (j+1+\gamma)\Delta[, j \in \mathbb{N} \end{cases}$

- $\alpha(\gamma) = \dfrac{1-b}{2\Delta\left(\gamma + b(1-\gamma)\right)}$, $\gamma = -\dfrac{b}{1-b} + \dfrac{\left(b - 2b^2 + 2b^4 - b^5\right)^{1/3}}{2^{1/3}(1-b)^2}$, $b = e^{-\epsilon}$.
- Preuvee d'ϵ-LDP : admise.

5. Geng, Q., Kairouz, P., Oh, S., & Viswanath, P. (2015). The staircase mechanism in differential privacy. IEEE Journal of Selected Topics in Signal Processing, 9(7), 1176-1184.

Nettoyage avec stairCase et GRR

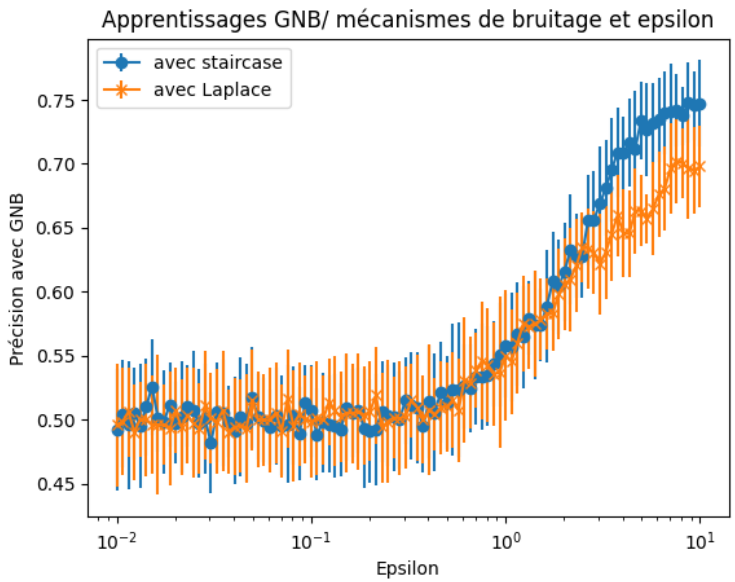
```
1  from diffprivlib import mechanisms
2  ...
3  def addBoundedStaircaseNoise(stairc,val,mi,mx,isInteger=False):
4    insideBounds = False
5    while (not insideBounds):
6      r = stairc.randomise(val)
7      insideBounds  = (r <= mx) and (r>= mi)
8    return int(round(r)) if isInteger else r
9
10 for att in integerAttributes:
11   mi, mx = description[att]['min'], description[att]['max']
12   stairc =  mechanisms.Staircase(epsilon=1,sensitivity=mx-mi)
13   df[att] = df[att].map(lambda a: addBoundedStaircaseNoise(stairc,a,mi,mx,True))
14
15 for att in floatAttributes:
16   mi, mx = description[att]['min'], description[att]['max']
17   stairc =  mechanisms.Staircase(epsilon=1,sensitivity=mx-mi)
18   df[att] = df[att].map(lambda a: addBoundedStaircaseNoise(stairc,a,mi,mx,False))
19
```

Utilité de mécanismes v.-à-v. du GNB

Contexte expérimental

- Sur le jeu de données du diabete.
- Outcome appris par GNB
- Mécanismes en concurrence : $\mathcal{M}_{\text{Stair}}$ et $\mathcal{M}_{Lb}$
- Pour chaque $\epsilon \in [10^{-2}, 10]$ individuel : $30\times$ nettoyages puis apprentissage
- Affichés : précision moyenne de l'apprentissage et écart-type

Résultats expérimentaux



- $\epsilon \in [10^{-2}, 1[$: réponses $\approx$ aléatoire similaires
- $\epsilon \in [1, 10]$: $\mathcal{M}_{\text{Stair}}$ plus utile que $\mathcal{M}_{Lb}$