Sécurité Appliquée-PVP
Apprentissage vérifiant la confidentialité différentielle avec diffprivlib

Jean-François COUCHOT
couchot[arobase]femto-st[point]fr

8 janvier 2023

1 Apprentssages bayésiens naïf

On considère à nouveau le jeu de données sur le diabète comme au TP1.

Exercice 1.1 (Apprentissage bayesien naïf avec `sklearn`).

1. *Récupérer le jeu de données.*
2. *S'inspirer du tutoriel pour classifier les personnes à l'aide d'un apprentissage bayésien naïf en exploitant la bibliothèque `sklearn`.*
3. *Vérifier que la précision de la prédiction ne change pas à chaque exécution.*
4. *Quelle serait la prédiction pour des personnes avec les caractéristiques ci-dessous ? Vérifier avec les données originales.*
   [7, 184, 84, 33, 0, 35.5, 0.355, 41]
   [6, 109, 60, 27, 0, 25.0, 0.206, 27]

Exercice 1.2 (Apprentissage bayesien naïf respectueux avec `diffprivlib`).
Maintenue par IBM, `DiffprivLib` est une bibliothèque en protection de la vie privée à base de confidentialité différentielle.

1. *Installer cette bibliothèque.*
2. *En s'inspirant du site github de diffprivlib reprendre l'exercice précédent.*
3. *Vérifier que la précision de la prédiction varie à chaque exécution même pour la valeur d'epsilon par défaut égale à 1.*
4. *Faire varier epsilon dans* $[0.001, 0.01, 0.1, 1, 10, 100]$ *et constater que les prédiction sont de plus en plus précises. Vers quelle valeur de précision cela converge-t-il ?*
5. *Quelle serait ici la prédiction pour des personnes avec les caractéristiques ci-dessous ?*
   [7, 184, 84, 33, 0, 35.5, 0.355, 41]
   [6, 109, 60, 27, 0, 25.0, 0.206, 27]
6. *Construire un graphique affichant la valeur moyenne de précision pour 10 prédictions pour* ϵ *variant dans* $[0.001, 0.01, 0.1, 1, 10, 50, 100]$.

2 K-means

Exercice 2.1 (Analyse d'un code implantant k-means.).

— *Comprendre chacune des lignes du code donné à la figure 1.*
— *Faire varier à la hausse `cluster_std` pour constater que la distance par rapport aux centroïd initiaux augmente.*
— *Décommenter la seconde partie et évaluer visuellement la perte de qualité des regroupements vérifiant la DP.*
— *Faire varier `epsilon` dans la création de l'objet `KMeans` et constater qu'en dessous d'un certain seuil, le regroupement avec cet algorithme n'est pas performant.*

```python
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import diffprivlib.models as dp

#partie 1
X, y_true = make_blobs(n_samples=400, centers=5, cluster_std=0.3, random_state=10)
plt.scatter(X[:, 0], X[:, 1], s=50);


km = KMeans(n_clusters=5)
km.fit(X)
y_km = km.predict(X)
plt.scatter(X[:, 0], X[:, 1], c=y_km, s=50, cmap='viridis')
km_centers = km.cluster_centers_
plt.scatter(km_centers[:, 0], km_centers[:, 1], c='black', s=200, alpha=0.5);
plt.show()

#partie 2
#dp_km = dp.KMeans(n_clusters=5, epsilon=25)#,random_state=0)
#dp_km.fit(X)
#y_dp_km = dp_km.predict(X)
#plt.scatter(X[:, 0], X[:, 1], c=y_dp_km, s=50, cmap='viridis')
#dp_km_centers = dp_km.cluster_centers_
#plt.scatter(dp_km_centers[:, 0], dp_km_centers[:, 1], c='black', s=200, alpha=0.5);
#plt.show()

```

FIGURE 1 – Code de k-mean sans ou avec DP