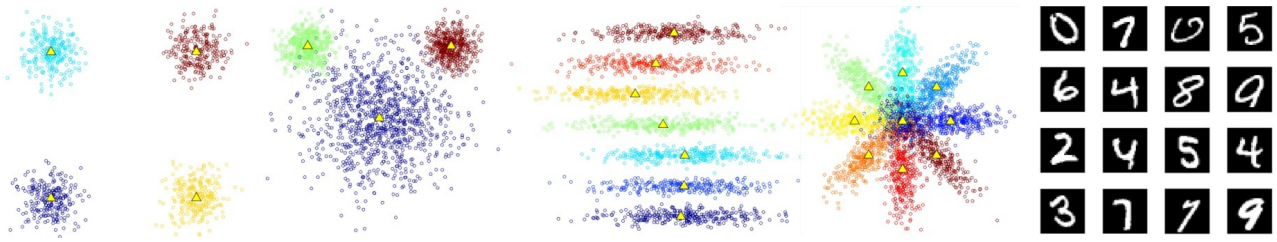
TP d'Introduction à l'Apprentissage Automatique
Exo 3: Clustering avec K-Means et GMM EM

Télécom Physique Strasbourg - antoine.deleforge@inria.fr



Téléchargez les fichiers du TP (Matlab) en suivant ce lien : `members.loria.fr/ADeleforge/files/TP_ML_Exo3_TPS.zip`. Sauf contre-indication, les scripts demandés sont à inclure dans les parties correspondantes du fichier fourni `TP_CLUSTERING.m`.

Partie I : Visualisation des Données

1) Générez un jeu de données `square(10)` en utilisant :

```
[data, true_centroids, true_labels] = dataset_square(10);
```

Visualisez le jeu de données dans une figure en utilisant :

```
fig1 = figure('Name','DATA');
clf(fig1);
movegui('northwest');
visualize_2Dclustering(fig1,data);
pause; % (Appuyer sur espace pour continuer)
```

Combien de regroupements (*clusters*) K observez-vous ? Faites varier le paramètre de la fonction `dataset_square`. À quoi correspond-il ? Visualisez ensuite les "vrais" labels et centroïdes du jeu de données en utilisant :

```
fig2 = figure('Name','GROUND TRUTH');
clf(fig2);
movegui('southwest');
visualize_2Dclustering(fig2,data,true_centroids,true_labels);
pause; % (Appuyer sur espace pour continuer)
```

2) Répondez aux même questions mais pour les jeux de données `mickeymouse(3)`, `pancakes(7)` et `flowers(9)`.

Partie II : K-Means

3) Implémentez l'algorithme k-means en complétant la fonction `kmeans.m` fournie. Visualisez et vérifiez son fonctionnement pendant `max_steps=30` itérations sur le jeu de données `dataset_square(10)` en ajoutant la ligne suivante dans la boucle principale de `TP_CLUSTERING.m` :

```
[new_centroids, new_labels] = kmeans(data, K, centroids, iter_per_step);
```

Utilisez `iter_per_step=1` pour visualiser le résultat à chaque itération.

Aide

- Pour initialiser les centroïdes, utilisez des points du jeux de données tirés au hasard grâce à la fonction Matlab `randperm(N,K)`.
- Pour visualiser le clustering avant et après la mise à jour des centroïdes, placez le code suivant dans la boucle après l'exécution de kmeans :

```
visualize_2Dclustering(fig3,data,centroids,new_labels); pause;
visualize_2Dclustering(fig3,data,new_centroids,new_labels); pause;
```

4) Ajoutez un critère d'arrêt précoce (*early stopping*) à la boucle, c'est à dire, si les labels restent identiques après une exécution de k-means, faire un break pour sortir de la boucle. Lancer k-means une dizaine de fois sur `dataset_square(10)`. Combien d'itérations sont typiquement nécessaires avant convergence ? Combien de fois k-means trouve-t-il un partitionnement (*clustering*) correcte des données ? Si k-means échoue à trouver un bon clustering, dans quelle situation et pourquoi ?

5) Lisez la fonction `kmeansplusplus_init.m` fournie pour comprendre son fonctionnement. Répétez l'exercice précédent mais cette fois-ci en initialisant les centroïdes avec cette fonction. On obtient alors l'algorithme k-means++. Quels effets observez-vous sur la durée de convergence et la qualité du clustering ?

6) Lancez k-means++ une dizaine de fois, cette fois-ci sur `dataset_square(4)`. Quel est l'impact sur la durée de convergence ? Quel est l'impact sur la qualité du clustering et sur sa stabilité ? Quels sont les points qui sont généralement incorrectement labilisés ? Mêmes questions sur `dataset_square(2)`. Déduisez-en une condition nécessaire pour qu'une estimation stable de clusters dans un jeu de données soit possible.

7) Faites tourner k-means++ sur `dataset_mickeymouse(3)` en choisissant le bon K. Qu'observez-vous ? Comment expliquez-vous ce comportement ? Quelle(s) limite(s) de k-means est(sont) ainsi mise(s) en avant ?

8) Mêmes questions que **7)** mais sur `dataset_pancakes(7)` puis sur `dataset_flowers(9)`.

Partie III : Gaussian Mixture Estimation with Expectation-Maximization

9) Lisez la fonction `gmm_em.m` fournie pour en comprendre son fonctionnement. Visualisez son exécution sur le jeu de données `mickeymouse(3)`, comme en question **3)**. Initialisez les centroïdes à l'aide de 20 itérations de l'algorithme kmeans++ et les matrices de covariances avec des matrices identité. Quels améliorations apporte GMM EM sur le jeux de données `mickeymouse(3)` par rapport à k-means ? Pourquoi ?

Aide

- Pour visualiser l'évolution des matrices de covariance, utilisez les lignes :

```
visualize_2Dclustering(fig3,data,centroids,new_labels,Sigma); pause;
visualize_2Dclustering(fig3,data,new_centroids,new_labels,new_Sigma); pause;
```

- Pensez à ajouter la ligne `Sigma = new_Sigma;`

10) Lancez maintenant GMM EM sur `dataset_flowers(9)`. Conseil : utilisez maintenant `iter_per_step=20` itérations entre chaque visualisation, car l'algorithme va mettre plus de temps à converger. Qu'observez-vous ? Ajoutez ensuite la ligne suivante à la fonction `gmm_em.m`, après la mise à jour des matrices de covariances :

```
Sigma(k,:,:) = diag(diag(squeeze(Sigma(k,:,:))));
```

puis relancez l'exécution. Cette ligne va forcer les matrices de covariance à rester diagonales. Qu'observez-vous et pourquoi ?

11) Mêmes questions que **10)** mais sur le jeu `pancakes(7)`.

Partie IV : Application au Partitionnement de MNIST

12) Téléchargez le jeu de test de MNIST à l'adresse `https://pjreddie.com/media/files/mnist_test.csv` puis chargez-le dans Matlab à l'aide de :

```
MNIST = load('mnist_test.csv');
labels = MNIST(:,1); data = MNIST(:,2:end);
```

Quelles sont la taille et la dimension de ce jeu ? Visualisez un sous-échantillon aléatoire de MNIST (par exemple 8×8 images) grâce à la fonction `visualize_MNIST.m` fournie.

13) Lancez 30 itérations de kmeans++ sur MNIST puis visualisez les centroïdes obtenus. Utilisez ensuite ces centroïdes pour initialiser GMM EM et le lancer pendant 30 itérations supplémentaires. Quels chiffres arrivez-vous à reconnaître ? Pour comparaison, visualisez la moyenne du dataset MNIST.

Comme vous pouvez le constater, exécuter k-means ou GMM EM sur MNIST est assez lourd. Pour découvrir une méthode permettant d'aller plus vite, rendez-vous au prochain TP !