Άσκηση 3

Στατιστική Μοντελοποίηση και Αναγνώριση Προτύπων

Αναστάσιος Πατερίτσας

ΑΜ: 2016030065

Θέμα 2.

Στο θέμα 2 μας ζητήθηκε να υλοποιήσουμε τον K-means και να τον χρησιμοποιήσουμε για την συμπίεση μιας εικόνας. Αρχικά χρησιμοποιήσαμε ένα σύνολο 2D δεδομένων για να δούμε πως λειτούργει ο αλγόριθμος. O K-means μια μέθοδος αυτόματης συσσώρευσης παρόμοιων δεδομένων. Η διαίσθηση πίσω από το K-means είναι μια επαναληπτική διαδικασία που ξεκινά υποθέτοντας τα αρχικά κεντροειδή και, στη συνέχεια, βελτιώνει αυτήν την εικασία, εκχωρώντας επανειλημμένα παραδείγματα στα πλησιέστερα κεντροειδή τους και, στη συνέχεια, υπολογίζοντας εκ νέου τα κεντροειδή στις εργασίες.

Αρχικά έπρεπε να υλοποιήσουμε την findClosestCentroids η οποία παίρνει τον πίνακα δεδομένων X και τις θέσεις όλων των κεντροειδών και θα πρέπει να παράγει ένα μονοδιάστατο πίνακα idx που κρατά το index (μια τιμή σε {1, ..., K}, όπου K είναι ο συνολικός αριθμός κεντροειδών) του πλησιέστερου κέντρου σε κάθε training example. Ο τύπος για κάθε example i :

$$c^{(i)} := j \quad \text{that minimizes} \quad ||x^{(i)} - \mu_j||^2$$

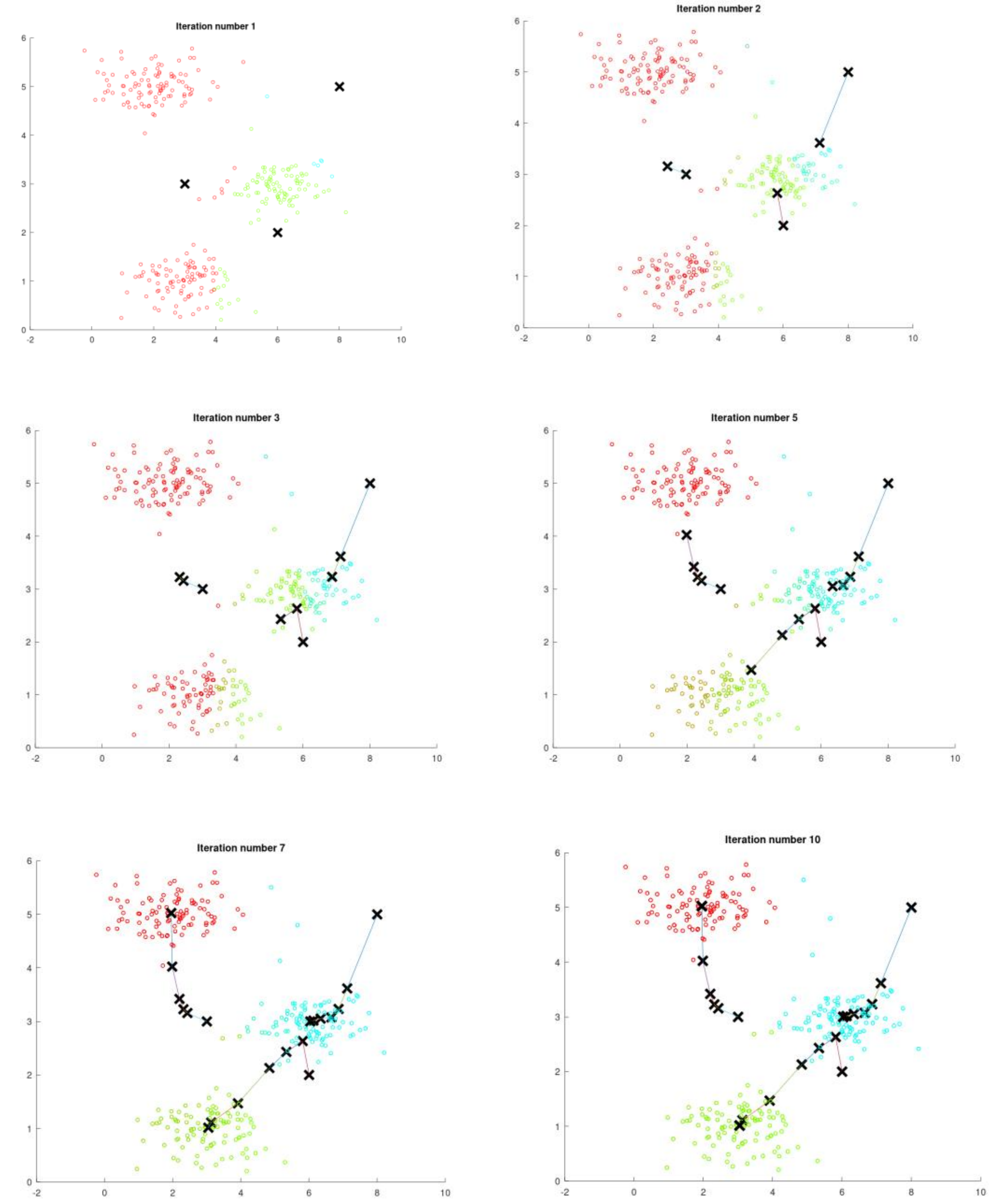
Οπου c index of the centroid that is closest to x(i).

Στην συνεχεία έπρεπε να υλοποιήσουμε την computeCentroid σύμφωνα με τον τύπο :

$$\mu_k := \frac{1}{|C_k|} \sum_{i \in C_k} x^{(i)}$$

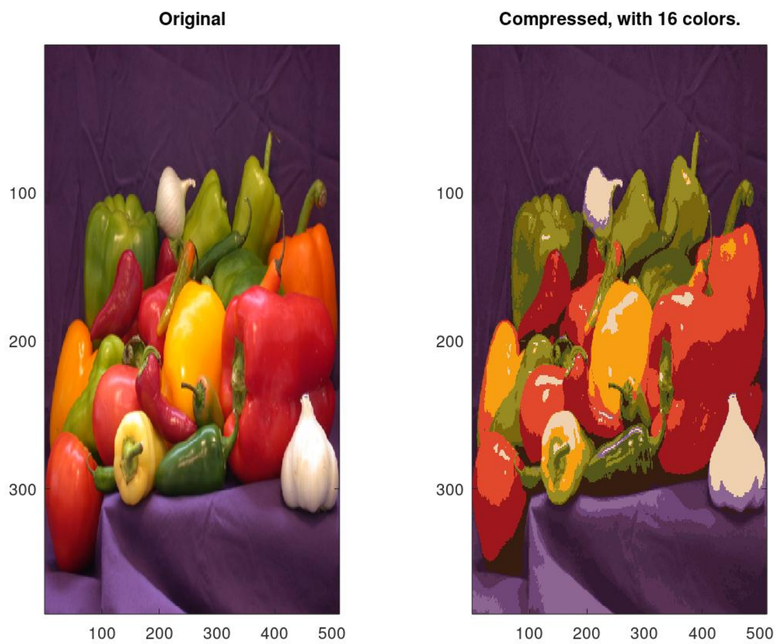
Όπου ο μ είναι είναι το mean για κάθε centroid και C_k

είναι το σύνολο των παραδειγμάτων που εκχωρούνται στο centroid k.



Παρατηρούμε ότι μετά τις 10 επαναλήψεις το clustering είναι σε πολύ καλό επίπεδο για τις 3 κλάσεις.

Ενώ στην συνεχεία εφαρμόσαμε τον ίδιο αλγόριθμο σε μια εικόνα για να συμπιέσουμε την εικόνα σε 16 pixel. Με την μόνη διαφορά είναι ότι επιλεγούμε τυχαία τα αρχικά centroids.



Θέμα 3.

Στην συγκεκριμένη άσκηση έπρεπε να υλοποιήσουμε τον αλγόριθμο Expetation Maximization για να εκπαιδεύσουμε τις παραμέτρους ενός μοντέλου GMM. Σε αντίθεση με το K-Means, με τα μοντέλα Gaussian Mixture θέλουμε να ορίσουμε μια κατανομή πιθανότητας στα δεδομένα. Για να γίνει αυτό, πρέπει να μετατρέψουμε το πρόβλημα ομαδοποίησης σε πρόβλημα συμπερασμάτων.

Ο αλγόριθμος EM είναι επαναληπτικός και αποτελείται από τα βήματα e step και m step.

Χρησιμοποιώντας τη EM, μπορούμε να ξαναγράψουμε την GMM objective function ως:

$$\sum_{i=1}^{n} lnp(x_i|\pi,\mu,\Sigma) = \sum_{i=1}^{n}\sum_{k=1}^{K} q(c_i=k) ln\frac{p(x_i,c_i=k|\pi,\mu,\Sigma)}{q(c_i=k)} + \sum_{i=1}^{n}\sum_{k=1}^{K} q(c_i=k) ln\frac{q(c_i=k)}{p(c_i=k|x_i,\pi,\mu,\Sigma)}$$

First: Set $q(c_i=k) \Leftarrow p(c_i=k|x_i,\pi,\mu,\Sigma)$ using Bayes rule:

$$p(c_i=k|x_i,\pi,\mu,\Sigma) \; \alpha \; p(c_i=k|\pi)p(x_i|c_i=k,\mu,\Sigma)$$

We can solve the posterior of c_i given π, μ and Σ :

$$q(c_i=k) = \frac{\pi_k \mathcal{N}(x_i|\mu_k,\Sigma_k)}{\sum_j \pi_j \mathcal{N}(x_i|\mu_j,\Sigma_j)} \Rightarrow \phi_i(k)$$

E-step: Take the expectation using the updated q's

$Q = \sum_{i=1}^{n} \sum_{k=1}^{K} \phi_i(k) lnp(x_i, c_i = k|\pi, \mu_k, \Sigma_k) +$ constant w.r.t π, μ, Σ

$lnp(x_i, c_i = k|\pi, \mu_k, \Sigma_k) = ln\pi_k + ln\mathcal{N}(x_i|\mu_k, \Sigma_k)$

M-step: Maximize Q with respect to π and each μ_k, Σ_k

Ο Αλγόριθμος:

Given: $x_1, \ldots, x_n$ where $x \in IR^d$

Goal*: Maximize $L = \sum_{i=1}^{n} lnp(x_i|\pi, \mu, \Sigma)$

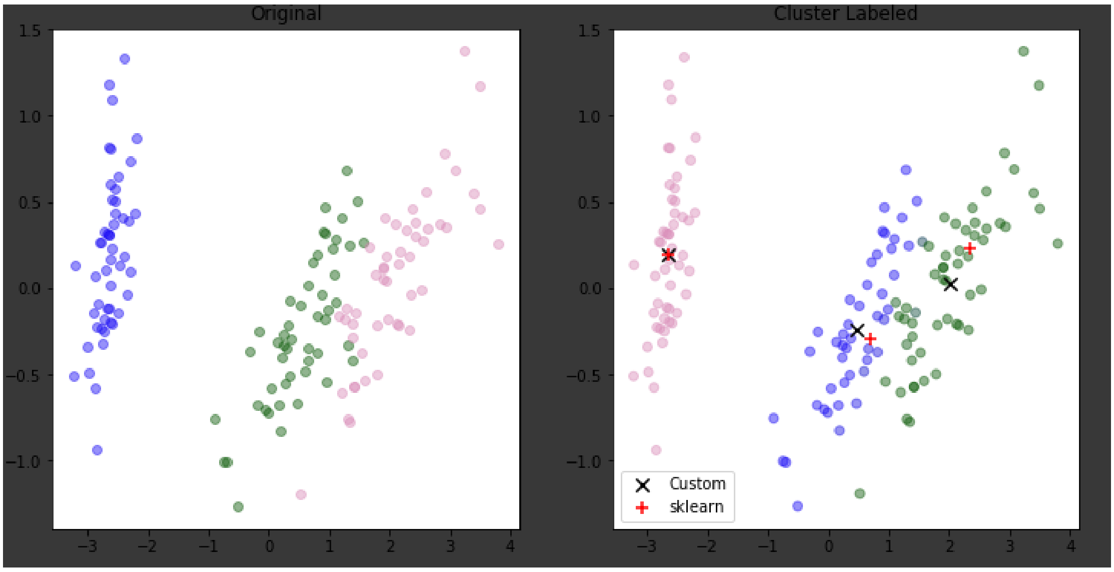
- Iterate until incremental improvement to L is "small"

1. **E-Step:** For $i = 1, \ldots, n$, set $\phi_i(k) = \frac{\pi_k \mathcal{N}(x_i|\mu_k,\Sigma_k)}{\sum_j \pi_j \mathcal{N}(x_i|\mu_j,\Sigma_j)}$, for $k = 1, \ldots, K$
2. **M-step:** For $k = 1, \ldots, K$, define $n_k = \sum_{i=1}^{n} \phi_i(k)$ and update the values:

$\pi_k = \frac{n_k}{n}$, $\mu_k = \frac{1}{n_k} \sum_{i=1}^{n} \phi_i(k) x_i$, $\Sigma_k = \frac{1}{n_k} \sum_{i=1}^{n} \phi_i(k)(x_i - \mu_k)(x_i - \mu_k)^T$

The update value for μ_k is used when updating Σ_k.

Αποτελεσματα:



Από την αριστερή πλευρά είναι τα original δεδομένα ενώ στην αριστερή βλέπουμε τα δεδομένα σύμφωνα με το πώς τα πρόβλεψε ο αλγόριθμος μας .Επιπλέον βλέπουμε και τα κέντρα σύμφωνα με το πώς τα πρόβλεψε ο GMM της sklearn για να τα συγκρίνουμε με τα δικά μας.

Για τα Kmeans χρησιμοποιήθηκε τα kmeans του sklearn.

Μετά από συνεννόηση με τον κύριο Τσιαρα μου επέτρεψε να το υλοποιήσω την άσκηση σε Python . Οπότε στο φάκελο exercise_3_3 θα βρείτε το αρχείο GMM.py οπου για να το τρέξετε θα πρέπει να τρέξετε στο terminal python GMM.py. Αν και συνιστώ να το τρέξετε στο google colab σαν cell για να μην χρειάζεται να κάνετε τα απαραιτητα install.

Θέμα 1.

Στην άσκηση 1 έπρεπε να υλοποιήσουμε έναν text classifier με KNN. Τα αρχεία που μας δόθηκαν είναι από ένα σύνολο της WebKb που έχει ιστοσελίδες από 4 πανεπιστήμια. Για να χρησιμοποιήσουμε τα δεδομένα μας δημιουργήθηκε ένας term document matrix που είναι ένας πινάκας οπου οι στήλες είναι οι λέξεις του λεξικού ενώ οι γραμμές είναι κείμενα. Τα κελία των συγκεκριμένων csv αντιστοιχούν στην συχνότητα εμφάνισης . Το πρόβλημα με αυτά τα είδη δεδομένων είναι ότι περιέχουν Αρχύτα μηδενικά. Για τον λόγο αυτό πρέπει να πάρουμε τις 300 σημαντικότερες λέξεις και χρησιμοποιούμε το μέγεθος της εντροπίας της κάθε λέξης.

Ο τύπος που δίνει την εντροπία είναι :

$$e_j = 1 + \frac{\sum_{i=1}^{n} p_{ij} \log(p_{ij})}{\log(nD)}$$

ενώ της κανονοικοποιημενης συχνότητας είναι

$$p_{ij} = f_{ij} / \sum_{i=1}^{nD} f_{ij}$$

.

Στην συνέχεια επιλεγούμε με βάσει την χαμηλότερη εντροπία τις 300 πρώτες λέξεις, ενώ χρησιμοποιούμε τον TF-IDF για να ποσοτικοποιουσουμε την σημαντικότητα των λέξεων.

Για την υλοποίηση του KNN χρησιμοποιήσαμε την τεχνική K-Flod Cross Validation με αποτέλεσμα να χωρίζουμε τα δεδομένα μας σε Folds και σε κάθε iterate κάνουμε το ένα fold test και τα άλλα train set. Σε κάθε Fold εφαρμόζουμε τον ταξινομητή για κάθε στοιχειό του test. Σαν μετρικές αποστάσεις θέτουμε την ευκλείδεια απόσταση η το cosine similarity. Σε κάθε περίπτωση επιλεγούμε την μικρότερη απόσταση για να πάρουμε τους κοντινότερους γείτονες.

Αποτελεσματα:

Ευκλείδεια Απόσταση:

```
Number of K nearest neighbors: 1

Fold: 1, Accuracy: 0.552347
Fold: 2, Accuracy: 0.527076
Fold: 3, Accuracy: 0.542254
Fold: 4, Accuracy: 0.584838
Fold: 5, Accuracy: 0.523132
K=1 -- Total Accuracy: 0.545929
Number of K nearest neighbors: 3

Fold: 1, Accuracy: 0.512635
Fold: 2, Accuracy: 0.412811
Fold: 3, Accuracy: 0.407942
Fold: 4, Accuracy: 0.468310
Fold: 5, Accuracy: 0.505415
K=3 -- Total Accuracy: 0.461423
Number of K nearest neighbors: 5

Fold: 1, Accuracy: 0.425993
Fold: 2, Accuracy: 0.411552
Fold: 3, Accuracy: 0.416370
Fold: 4, Accuracy: 0.367857
Fold: 5, Accuracy: 0.437722
K=5 -- Total Accuracy: 0.411899
Number of K nearest neighbors: 10

Fold: 1, Accuracy: 0.357401
Fold: 2, Accuracy: 0.660650
Fold: 3, Accuracy: 0.393502
```

Cosine similarity:

```
Number of K nearest neighbors: 1

Fold: 1, Accuracy: 0.782918
Fold: 2, Accuracy: 0.747292
Fold: 3, Accuracy: 0.750903
Fold: 4, Accuracy: 0.753571
Fold: 5, Accuracy: 0.718861
K=1 -- Total Accuracy: 0.750709
Number of K nearest neighbors: 3

Fold: 1, Accuracy: 0.772242
Fold: 2, Accuracy: 0.750000
Fold: 3, Accuracy: 0.772242
Fold: 4, Accuracy: 0.736462
Fold: 5, Accuracy: 0.790614
K=3 -- Total Accuracy: 0.764312
Number of K nearest neighbors: 5

Fold: 1, Accuracy: 0.735714
Fold: 2, Accuracy: 0.782918
Fold: 3, Accuracy: 0.783394
Fold: 4, Accuracy: 0.794224
Fold: 5, Accuracy: 0.779359
K=5 -- Total Accuracy: 0.775122
Number of K nearest neighbors: 10

Fold: 1, Accuracy: 0.828571
Fold: 2, Accuracy: 0.808664
Fold: 3, Accuracy: 0.750890
Fold: 4, Accuracy: 0.747292
Fold: 5, Accuracy: 0.786477
```

Όπως είναι ευκολο να παρατηρηθεί με το Cosine similiraty παιρνουνε πολύ καλυτέρα αποτελέσματα.