Problem Set 3

Τρίμας Χρήστος
AM:2016030054
Στατιστική Μοντελοποιήση και Αναγνώριση Προτύπων
Πολυτεχνειο Κρητης

May 28, 2020

Ερώτηση 1

Κατηγοριοποίηση κειμένου με k-NN:

Στην 1η άσκηση της 3ης εργασίας, θα ασχοληθούμε με την κατηγοριοποιήση κειμένου, δηλαδή την κατατάξη κειμένων φυσικής γλώσσας σε ένα προκαθορισμένο αριθμό κατηγοριών - κλάσεων. Το σύνολο δεδομένων με το οποίο θα ασχοληθούμε, περιλαμβάνει λέξεις που συναντόνται σε ιστοσελίδες από 4 πανεπιστήμια.

Επειδή οι λέξεις αυτές καθαυτές, δεν μπορούν να χρησιμοποιηθούν σε ένα σύστημα μηχανικής μάθησης, το dataset με το οποίο θα δουλέψουμε στην πραγματικότητα, είναι ο term document matrix των σελίδων. Αυτός ο πίνακας $M \in \mathbb{R}^{(\mathbb{D}\times\mathbb{T})}$, όπου nD είναι το πλήθος των γραμμών, και nT, είναι το πλήθος των στηλών ή αντίστοιχα, το πλήθος των σελίδων και το πλήθος των λέξεων.

Στο πρόβλημα τώρα, θέλουμε να κάνουμε μια ταξινόμηση των κειμένων. Ένας τρόπος για να κρατήσουμε τις λέξεις με την μεγαλύτερη σημασία μέσα σε ένα κείμενο, είναι να αξιοποιήσουμε την εντροπία αυτής της λέξης. Από την εκφώνηση της άσκησης, η εντροπία ορίζεται ως:

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

Με την ποσότητα, $p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{nD} f_{ij}}$, να ορίζεται ως η κανονικοποιημένη συχνότητα της λέξης.

Σκοπός μας είναι να βρεθούν οι 300(μπορεί και κάποιος άλλος αριθμός) λέξεις με την μεγαλύτερη εντροπία, έτσι ώστε να κατασκευαστεί ένα νέο vocabulary, το οποίο θα περιλαμβάνει μόνο τις πιο σημαντικές λέξεις από τα κείμενα μας.Για να δημιουργηθεί αυτό το λεξιλόγιο, είναι απαραίτητο να υπολογιστεί και το tf-idf, το οποίο σημαίνει term frequency - inverse document frequency, και εκφράζει την κανονικοποιημένη συχνότητα ενός όρου, ενώ ταυτόχρονα ποσοτικοποιεί πόσο σημαντικός είναι αυτός ο όρος.

Αφού λοιπόν, βρεθούν οι σημαντικότερες λέξεις με την μεγαλύτερη εντροπία, και δημιουργηθεί από αυτές το νέο λεξιλόγιο, προχωράμε σε ταξινόμηση του συστήματος μας με χρήση του

k-Nearest-Neighbor αλγορίθμου.

Άλλος ένας αλγόριθμος ταξινόμησης που χρησιμοποιείται σε supervised learning, ο kNN, "υποθέτει" ότι παρόμοια αντικείμενα "βρίσκονται" το ένα κοντά στο άλλο. Άρα, για να ταξινομήσει μια λέξη, έλεγχει την απόσταση από κάθε κλάση και υποθέτει ότι αυτή με την ελάχιστη δυνατή απόσταση είναι και η κλάση ταξινόμησης του δείγματος.

Για την εύρεση της απόστασης, χρησιμοποιηθηκαν δύο απλοί αλγόριθμοι, η Ευκλείδια απόσταση, και η απόσταση όμοιου συνημιτόνου.

Υποθέτοντας ότι όλα έγιναν σωστά, πρέπει να χρησιμοποιηθεί ένα μέτρο απόδοσης του αλγορίθμου, έτσι ώστε να έχουμε μια εικόνα, κατά πόσο δουλεύει για το συγκεκριμλένο σετ δεδομένων. Η πιο απλή και εύκολα υλοποιήσιμη μέθοδος αξιολόγησης είναι η k-fold Cross Validation.

Η διαδικασία είναι αρκετά απλή:

1) Ανακάτεξε το dataset τυχαία.
2) Χώρισε το dataset σε k ομάδες.
3) Για κάθε ομάδα:
    α)χώρισε το σε train και test set.
    β)εφάρμοσε τον k-NN(στην περίπτωση μας).
4) Αξιολόγησε την μέθοδο, παρακολουθώντας την διαφορά μεταξύ των πραγματικών labels, και αυτών που έγινε πρόβλεψη.

Παρακάτω, παρουσίαζονται τα αποτελέσματα και με τις δύο μεθόδους υπολογισμού της απόστασης που αναφέρθηκαν προηγουμένως.

```
Number of words selected: 300
Distance Metric: norm2
Number of K nearest neighbors: 1
Fold: 1, Accuracy: 0.299639
Fold: 2, Accuracy: 0.281588
Fold: 3, Accuracy: 0.238434
Fold: 4, Accuracy: 0.289286
Fold: 5, Accuracy: 0.284698
K=1 -- Total Accuracy: 0.278729
Number of K nearest neighbors: 3
Fold: 1, Accuracy: 0.234657
Fold: 2, Accuracy: 0.231317
Fold: 3, Accuracy: 0.227758
Fold: 4, Accuracy: 0.242857
Fold: 5, Accuracy: 0.241877
K=3 -- Total Accuracy: 0.235693
Number of K nearest neighbors: 5
Fold: 1, Accuracy: 0.227437
Fold: 2, Accuracy: 0.227758
Fold: 3, Accuracy: 0.228571
Fold: 4, Accuracy: 0.220217
Fold: 5, Accuracy: 0.227758
K=5 -- Total Accuracy: 0.226348
Number of K nearest neighbors: 10
Fold: 1, Accuracy: 0.227437
Fold: 2, Accuracy: 0.227758
Fold: 3, Accuracy: 0.220641
Fold: 4, Accuracy: 0.221429
Fold: 5, Accuracy: 0.227437
K=10 -- Total Accuracy: 0.224940
```

Figure 1: Euclidean Distance.

```
Number of words selected: 300
Distance Metric: Cosine Similarity
Number of K nearest neighbors: 1
Fold: 1, Accuracy: 0.234875
Fold: 2, Accuracy: 0.220217
Fold: 3, Accuracy: 0.223827
Fold: 4, Accuracy: 0.232143
Fold: 5, Accuracy: 0.217082
K=1 -- Total Accuracy: 0.225629
Number of K nearest neighbors: 3
Fold: 1, Accuracy: 0.223827
Fold: 2, Accuracy: 0.223827
Fold: 3, Accuracy: 0.221429
Fold: 4, Accuracy: 0.220641
Fold: 5, Accuracy: 0.220641
K=3 -- Total Accuracy: 0.222073
Number of K nearest neighbors: 5
Fold: 1, Accuracy: 0.221429
Fold: 2, Accuracy: 0.223827
Fold: 3, Accuracy: 0.220641
Fold: 4, Accuracy: 0.220641
Fold: 5, Accuracy: 0.223827
K=5 -- Total Accuracy: 0.222073
Number of K nearest neighbors: 10
Fold: 1, Accuracy: 0.223827
Fold: 2, Accuracy: 0.223827
Fold: 3, Accuracy: 0.223827
Fold: 4, Accuracy: 0.220641
Fold: 5, Accuracy: 0.218310
K=10 -- Total Accuracy: 0.222086
```

Figure 2: Cosine Distance.

Όπως φαίνεται, η διαφορά στην απόδοση είναι πάρα πολύ μικρή, ωστόσο η ίδια η απόδοση είναι τραγικά χαμηλη, πράγμα που σημαίνει ότι ένας τέτοιος ταξινομητής δεν ταιρίαζει για το συγκεκριμένο dataset. Από πλευράς χρονικής πολυπλοκότητας, ο υπολογισμός του cosine distance είναι αρκετά πιο σύνθετος από αυτόν του euclidean και απαιτεί αρκετά περισσότερο χρόνο για τους υπολογισμούς των αποστάσενω, κάτι που είναι λογικό.

Ερώτηση 2

K-means clustering:

Ο k-means αλγόριθμος, είναι μια μέθοδος για αυτοματοποιημένο clustering όμοιων δεδομένων μαζί. Δεδομένου ενός σετ δεδομένων εκπαίδευσης, $[x^{(1)}, ..., x^{(m)}]$, θέλουμε να ομαδοποιήσουμε τα δεδομένα σε μερικούς συνεκτικούς clusters.

Η βασική αρχή του αλγορίθμου, είναι μια επαναληπτική διαδικασία με αφετηρία την πρόβλεψη των αρχικών centroids, και βελτιώνει αυτή την πρόβλεψη, αναθέτωντας δείγαμτα στον κοντινότερο centroid, και επαναυπολογίζει τους νέους centroids, με βάση τα νέα δεδομένα.

Επειδή ακριβώς η αρχική επιλογή γίνεται τυχαία, για όσο το δυνατόν καλύτερα αποτελέσματα του αλγορίθμου, εκτελείται μερικές φορές με διαφορετική αρχικοποιήση. Ένας τρόπος για επιλογή μεταξύ αυτών των τυχαίων τιμών είναι να επιλεχτεί αυτός με το μικρότερο κόστος.

Για την συγκεκριμένη άσκηση, χρείαστηκε να υλοποιηθούν τα εξής κομμάτια:
1) Εύρεση του κοντινότερου centroid.
2) Υπολογισμός του μέσου του centroid.

Για το πρώτο κομμάτι, χρειάτηκε να βρεθρί το $c^{(i)} = j$ τέτοιο ώστε να ελαχιστοποιείται ο όρος, $||x^{(i)} - \mu_j||^2$, όπου $c^{(i)}$ είναι το νούμερο του centroid(ο δείκτης) πιο κοντά στην τιμή $x^{(i)}$ και μ_j είναι η θέση του j-centroid.

Για τον υπολογισμό του μέσου κάθε centroid θέτω:

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

Όπου C_k είναι το σετ των παραδειγμάτων που έχουν ανατεθεί στον centroid.

Με την ολοκλήρωση των συναρτήσεων υπολογισμού των παραπάνω, δοκιμάζεται ο kmeans αλγόριθμος για 10 επαναλήψεις στο σετ δεδομέων που δίνεται.

Όπως φαίνεται παρακάτω, ο kmeans μετά από 7 επαναλήψεις δεν δείχνει κάποια αισθητή ένδειξη βελτίωσης.

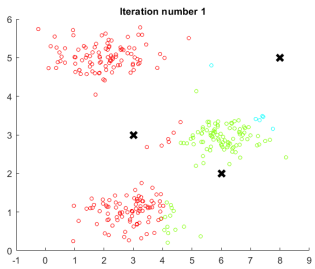

Figure 3: Iteration 1

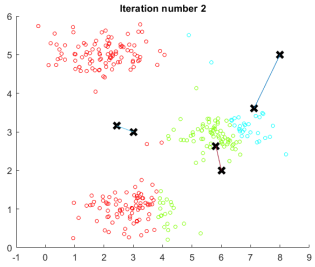

Figure 4: Iteration 2

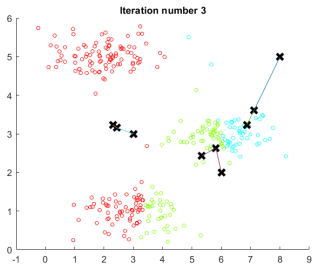

Figure 5: Iteration 3

kmeans

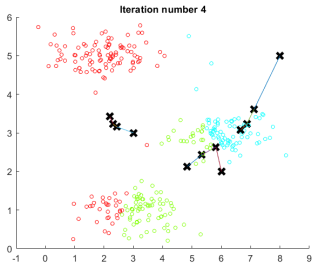

Figure 6: Iteration 4

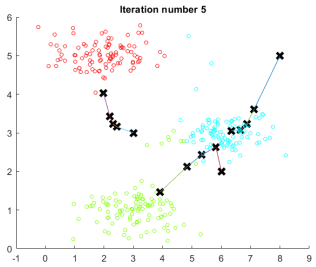

Figure 7: Iteration 5

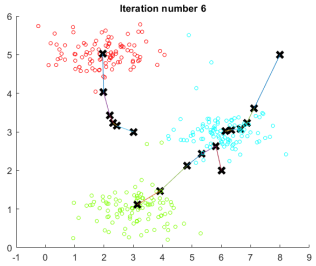

Figure 8: Iteration 6

kmeans

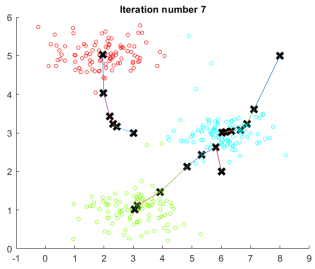

Figure 9: Iteration 7

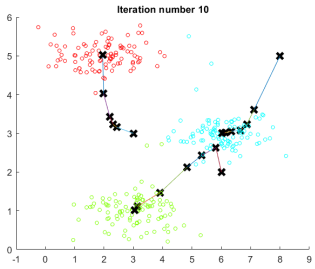

Figure 10: Iteration 8

Στη συνέχεια της άσκσης θα εφαρμοστεί ο αλγόριθμος για την συμπίεση μιας εικόνας. Παρακάτω απεικονίζονται οι εικόνες για διάφορες τιμές χρωμάτων K και για διαφορετικές τιμές επαναλήψεων του k-means αλγορίθμου.

Παρατηρείται, ότι με αύξηση του K και του αριθμού των επαναλήψεων του kmeans, η συμπιεσμένη εικόνα πλησιάζει όλο και πιο κοντά στην πραγματική εικόνα.

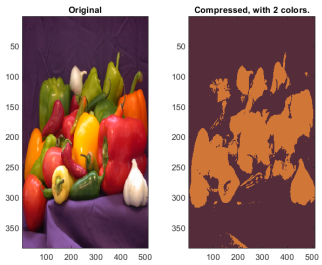

Figure 11: K=2,iter=10

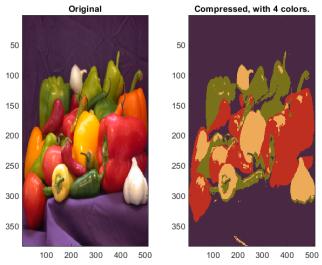

Figure 12: K=4,iter=8

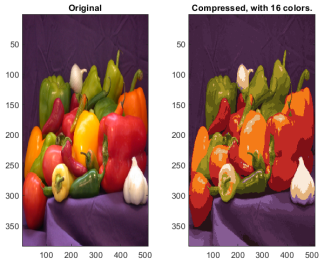

Figure 13: K=16,iter=10

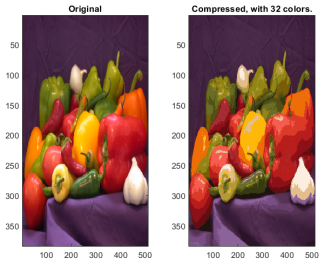

Figure 14: K=32,iter=20

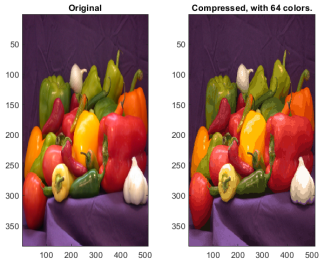

Figure 15: K=64,iter=30

Ερώτηση 3

GMMs και εκτίμηση με τον αλγόριθμο Expectation Maximization:

Στην τελευταία άσκηση του σετ, θα ασχοληθούμε άλλη μια φορά με το iris dataset, και αυτή την φορά θα συγκρίνουμε την απόδοση δυο αλγορίθμων για ταξινόμηση. Οι αλγόριθμοι μας είναι ο kmeans, που συναντήσαμε προηγουμένως, και ο Expectation-Maximization(EM) σε ένα GMM(gaussian-mixture-model). Σε ότι έχει να κάνει με τον kmeans, τα πράγματα είναι αρκετά απλά και ισχύει ότι και στην προηγούμενη άσκηση. Στην παρούσα μάλιστα άσκηση δεν μας νοιάζει η υλοποιήση του αλγορίθμου, αλλά μόνο η σύγκριση του με τον EM, άρα απλά καλείται από την έτοιμη συναρτηση matlab kmeans().

Για τον EM, τα πράγματα είναι λίγο πιο σύνθετα.Για τις συνθήκες αρχικοποιήσης, ο GMM χρησιμοποιεί τον kmeans για να αποκτήσει τις μέσες τιμές για την πρώτη επανάληψη.Στη συνέχεια με την χρήση της συνάρτησης gaussian, υπολογίζεται η posterior probability, και τελικά υπολογίζεται η πιθανοφάνεια G. Αυτό ήταν το βήμα expectation, και ακολουθεί το βήμα της μεγιστοποίησης. Πρώτα υπολογίζεται ο αριθμός των δειγμάτων κάθε cluster, και με χρήση του maximum likelihood estimator, βρίσκεται ο νέος πίνακας μέσων τιμών και ο covariance matrix.

Στη συνέχεια, θέλουμε να τεστάρουμε την απόδοση του αλγορίθμου μας. Με την δημιουργία ενός απλού bayesian classifier. Ωστόσο, ένα πρόβλημα που δημιουργήθηκε κατά την ταξινόμηση είναι ότι όποιος cluster είχε την μεγαλύτερη πιθανότητα, ο ταξινομητής "βάζει" όλα τα δείγματα ελέγχου σε αυτό τον cluster, δηλαδή αν σε μια εκτέλεση του αλγορίθμου ο cluster 2 έχει την μεγαλύτερη πιθανότητα, τότε όλα τα labels που θα επιστέψει θα έχουν την ίδια τιμή ,ίση με 2. Αποτέλεσμα αυτού είναι ο ταξινομητής να έχει σταθερά απόδοση 0,3333 και να κολλάει σε "ακρότατο".

Πρίν όμως βιαστούμε να αποφανθούμε για τον GMM classifier ότι δεν είναι ιδανικός για το πρόβλημα μας, να σημειωθεί ότι σε ένα ποσοστό κοντα στο 0,95 ο ταξινομητής συμπεριφέρεται καλύτερα(καλύτερη απόδοση δηλαδή) από ότι ο kmeans classifier. Τέλος σαν γενική παρατήρηση, ο GMM αποτελεί clustering αλγόριθμο και όχι classifier, για αυτό το λόγο γίνεται χρήση ενός bayes gaussian classifier για την ταξινόμηση, με τις πιθανότητες που επιστρέφει ο GMM για κάθε cluster.

```
accuracy_GMM =

    0.3333


accuracy_kmeans =

    0.3200
```

Figure 16: GMM vs Kmeans

```
accuracy_GMM =

    0.3333


accuracy_kmeans =

    0.0133
```

Figure 17: GMM vs Kmeans

```
accuracy_GMM =

    0.3333


accuracy_kmeans =

    0.2400
```

Figure 18: GMM vs Kmeans

```
accuracy_GMM =

    0.3333


accuracy_kmeans =

    0.5533
```

Figure 19: GMM vs Kmeans