Εργασία στο μάθημα

ΕΞΟΡΥΞΗ ΔΕΔΟΜΕΝΩΝ

Αλγόριθμοι συσταδοποίησης – ποιότητα συσταδοποίησης

Μέρος 1ο: Διαμεριστική συσταδοποίηση με k-means

Ο k-means χρησιμοποιεί έναν επαναληπτικό τρόπο ώστε να ελαχιστοποιηθεί το άθροισμα των αποστάσεων μεταξύ των σημείων από το κέντρο βάρους της κάθε συστάδας.

Τα βασικά βήματα του αλγορίθμου είναι τα εξής:

1. Επιλογή του αριθμού των συστάδων k.
2. Τυχαία δημιουργία k ομάδων και ορισμός των κεντροειδών των συστάδων.
3. Μεταβίβαση του κάθε σημείου στο κεντροειδές της κοντινότερης συστάδας.
4. Υπολογισμός της μέσης τιμής των συστάδων για ορισμό των νέων κεντροειδών.
5. Επανάληψη μέχρι να συγκλίνει ο αλγόριθμος σε κάποιο κριτήριο.

Ο αλγόριθμος ξεκινά διαχωρίζοντας τα αρχικά σημεία σε k αρχικά σύνολα συνήθως σε τυχαία δεδομένα. Στη συνέχεια, υπολογίζει το μεσαίο ή το κεντροειδές του κάθε συνόλου, υλοποιεί νέο διαχωρισμό ώστε το κάθε σημείο να σχετίζεται με το κοντινότερο κεντροειδές. Έπειτα, τα κεντροειδή υπολογίζονται ξανά για τις νέες συστάδες, ο αλγόριθμος επαναλαμβάνει τα δυο βήματα, ωσότου τα σημεία δεν μπορούν να αλλάξουν συστάδες (ή εναλλακτικά τα κεντροειδή παραμένουν αμετάβλητα).

Παράμετροι k-means (Sklearn)

n_clusters	Αριθμός συστάδων

Init	Μέθοδος αρχικοποίησης. (default='k-means++') 'k-means++' : Επιλέγει τα αρχικά κέντρα των συστάδων για k-mean clustering με έξυπνο τρόπο ώστε να επιταχύνει τη σύγκλιση. 'random': Επιλέγει k παρατηρήσεις (σειρές) τυχαία από τα δεδομένα για τα αρχικά κεντροειδή. Αν το όρισμα είναι ένας πίνακας ndarray τότε θα πρέπει να έχει τη μορφή (n_clusters, n_features) και να δίνει τα αρχικά κέντρα.
n_init	Πλήθος φορών που ο αλγόριθμος k-means θα τρέξει με διαφορετικά κεντροειδή. Τα τελικά αποτελέσματα θα είναι η καλύτερη απόδοση των διαδοχικών εκτελέσεων n_init. (default=10)
max_iter	Μέγιστος αριθμός επαναλήψεων του αλγορίθμου k-means για μία μόνο εκτέλεση. (default=300)
Tol	Σχετική ανοχή όσον αφορά την αδράνεια να δηλώνεται η σύγκλιση. (default=1e-4)
precompute_distances	Προϋπολογισμός αποστάσεων. 'auto': Δεν προϋπολογίζει τις αποστάσεις αν ισχύει n_samples * n_clusters > 12 εκατομμύρια. True: πάντα να προϋπολογίζει τις αποστάσεις. False: να μην προϋπολογίζει τις αποστάσεις καθόλου. (default='auto')
verbose	Κατάσταση λειτουργίας εμφάνισης λεπτομερειών. (default=0)
random_state	Καθορίζει την παραγωγή τυχαίων αριθμών για την αρχικοποίηση των κεντροειδών. (default=None)
copy_x	Όταν προϋπολογίζονται οι αποστάσεις, είναι αριθμητικά ακριβέστερο να έχουν κεντραριστεί τα δεδομένα προηγουμένως. (default=True)

	True: Τα αρχικά δεδομένα δεν τροποποιούνται. False: Τα αρχικά δεδομένα τροποποιούνται
n_jobs	Ο αριθμός των εργασιών που πρέπει να χρησιμοποιηθούν για τον υπολογισμό. Λειτουργεί με τον υπολογισμό κάθε παράλληλης εκτέλεσης. n_init. (default=None)

Γνωρίσματα k-means (Sklearn)

cluster_centers_	Συντεταγμένες των κέντρων των συστάδων.
labels_	Ετικέτες-αναγνωριστικά κάθε σημείου.
inertia_	Το άθροισμα των τετραγωνικών αποστάσεων των δειγμάτων στο πλησιέστερο κέντρο.
n_iter_	Αριθμός επαναλήψεων εκτέλεσης.

Μέθοδοι k-means (Sklearn)

fit(self, X [, y, sample_weight])	Συσταδοποίηση k-means.
fit_predict(self, X [, y, sample_weight])	Υπολογίζει τα κέντρα των συστάδων και προβλέπει τον δείκτη συστοιχίας για κάθε δείγμα.
fit_transform(self, X [, y, sample_weight])	Υπολογίζει τη συσταδοποίηση και μετασχηματίζει το X σε χώρο απόστασης συστάδων.
get_params(self[, deep])	Προσάρτηση των παραμέτρων του εκτιμητή.
predict(self, X [, sample_weight])	Προβλέπει την πλησιέστερη συστάδα κάθε δείγματος το οποίο ανήκει στο X.
score(self, X [, y, sample_weight])	Αντίθετο της τιμής του X του αντικειμένου k-means.
set_params(self, **params)	Ορισμός των παραμέτρων του εκτιμητή.

transform(self, X)	Μετασχηματίζει το X σε χώρο απόστασης συστάδων.

1.1. Εφαρμογή στο σύνολο δεδομένων iris

Φορτώστε το σύνολο δεδομένων IRIS που υπάρχει στο άρθρωμα **sklearn.datasets**, το οποίο περιέχει τυχαία δείγματα των λουλουδιών, που ανήκουν σε τρία είδη της ίριδας (μικρούς κρίνους): α) το setosa, β) το versicolor και γ) το virginica. Για καθένα από τα τρία διαφορετικά είδη υπάρχουν 50 παρατηρήσεις που αφορούν το μήκος σεπάλου, το πλάτος σεπάλου, το μήκος πετάλου και το πλάτος πετάλου.

Εφαρμόζοντας τη μέθοδο διαμεριστικής συσταδοποίησης k-means μπορείτε να βρείτε τρεις συστάδες στις οποίες να ομαδοποιούνται οι παρατηρήσεις αυτού του συνόλου δεδομένων;

Βήμα 1. Φόρτωση του συνόλου παρατηρήσεων Iris.

```
from sklearn.datasets import load_iris
meas = load_iris().data
```

Βήμα 2. Εκτέλεση του αλγορίθμου k-means με την Ευκλείδεια απόσταση.

```
from sklearn.cluster import KMeans
# Χρησιμοποιούνται οι 2 τελευταίες διαστάσεις του πίνακα
X = meas[:, [2, 3]]
k = 3  # Ορίζεται ότι τα δεδομένα θα οργανωθούν σε 3 συστάδες
kmeans = KMeans(n_clusters=k).fit(X)  # Εφαρμογή του k-means
IDX = kmeans.labels_
C = kmeans.cluster_centers_
```

Σημείωση: Εξ ορισμού, ο αλγόριθμος συσταδοποίησης k-means βασίζεται στην ευκλείδεια απόσταση, συνεπώς η Python δεν επιτρέπει τη χρήση άλλων αποστάσεων.

Βήμα 3. Παρουσιάστε τα δεδομένα σε γράφημα.

```
import matplotlib.pyplot as plt
plt.figure(1)
# Παρουσιάζεται η κλάση που ανήκει η κάθε παρατήρηση
```

```
plt.plot(IDX[:],'o') plt.show()
plt.plot(X[IDX==0][:,0], X[IDX==0][:,1], 'limegreen', marker='o', linewidth=0, label='C1')
plt.plot(X[IDX==1][:,0], X[IDX==1][:,1], 'yellow', marker='o', linewidth=0, label='C2')
plt.plot(X[IDX==2][:,0], X[IDX==2][:,1], 'c.', marker='o', label='C3')
plt.scatter(C[:,0], C[:,1], marker='x', color='black', s=150 , linewidth=3, label="Centroids", zorder=10)
plt.legend()
plt.show()
```

Βήμα 4. Χρησιμοποιώντας ως κριτήρια αποτίμησης το SSE και τον συντελεστή περιγράμματος, μελετήστε την επίδραση του k στη συσταδοποίηση. Για τον καλύτερο συνδυασμό χαρακτηριστικών, να απεικονιστεί η σχέση k – SSE σε γραφική παράσταση.

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html

1.2. Εφαρμογή στο σύνολο δεδομένων xV.mat

Βήμα 1. Φορτώστε το σύνολο δεδομένων xV που υπάρχει στο αρχείο δεδομένων **'xV.mat'**, το οποίο περιέχει 600 δείγματα με 469 χαρακτηριστικά το κάθε ένα.

```
import scipy.io
import numpy as np
mat_file = scipy.io.loadmat('xV.mat')
xV = np.array(mat_file['xV'])
```

Βήμα 2. Εκτελέστε συσταδοποίηση με τον αλγόριθμο k-means με Ευκλείδεια απόσταση για 3 συστάδες, χρησιμοποιώντας τα δύο πρώτα χαρακτηριστικά του πίνακα xV. Υπολογίστε το SSE.

```
X = xV[:,[0,1]]
k = 3
kmeans = KMeans(n_clusters=k).fit(X)
IDX = kmeans.labels_
C = kmeans.cluster_centers_
plt.plot(X[IDX==0][:,0], X[IDX==0][:,1], 'limegreen', marker='o', linewidth=0, label='C1')
plt.plot(X[IDX==1][:,0], X[IDX==1][:,1], 'yellow', marker='o', linewidth=0, label='C2')
plt.plot(X[IDX==2][:,0], X[IDX==2][:,1], 'c.', marker='o', label='C3')
plt.scatter(C[:,0], C[:,1], marker='x', color='black', s=150 , linewidth=3, label="Centroids", zorder=10)
```

```
plt.legend()
plt.show()
```

Βήμα 3. Εκτελέστε συσταδοποίηση με τον αλγόριθμο k-means με Ευκλείδεια απόσταση για 3 συστάδες χρησιμοποιώντας τα χαρακτηριστικά [296, 305] του πίνακα xV. Υπολογίστε το SSE.

```
X = xV[:,[296, 305]]
k = 3
kmeans = KMeans(n_clusters=k).fit(X)
IDX = kmeans.labels_
C = kmeans.cluster_centers_
plt.plot(X[IDX==0][:,0], X[IDX==0][:,1], 'limegreen', marker='o', linewidth=0, label='C1')
plt.plot(X[IDX==1][:,0], X[IDX==1][:,1], 'yellow', marker='o', linewidth=0, label='C2')
plt.plot(X[IDX==2][:,0], X[IDX==2][:,1], 'c.', marker='o', label='C3')
plt.scatter(C[:,0], C[:,1], marker='x', color='black', s=150 , linewidth=3, label="Centroids", zorder=10)
plt.legend()
plt.show()
```

Βήμα 4. Επαναλάβετε το **Βήμα 3** χρησιμοποιώντας τα δύο τελευταία χαρακτηριστικά του πίνακα xV.

Βήμα 5. Επαναλάβετε το **Βήμα 3** για 3 συστάδες χρησιμοποιώντας τα χαρακτηριστικά [205, 175] του πίνακα xV. Υπολογίστε το SSE.

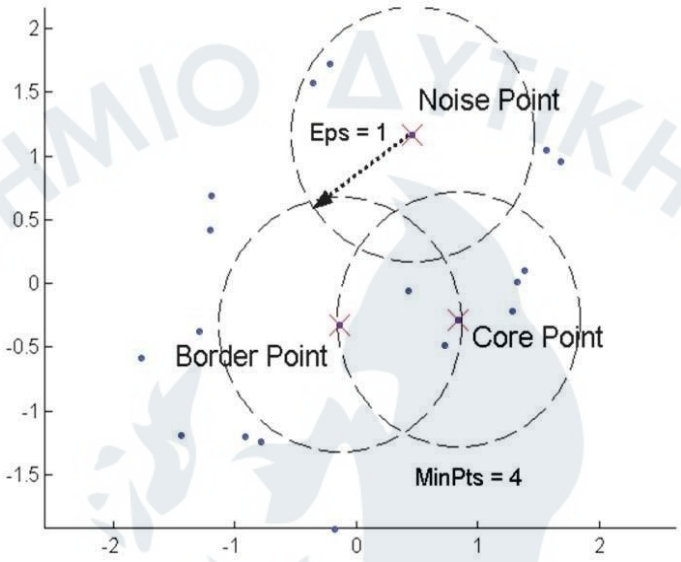
Βήμα 6. Συγκρίνετε τα αποτελέσματα των βημάτων 2, 4 και 5.

Μέρος 2ο: Συσταδοποίηση βάσει πυκνότητας με DBSCAN

Ο **DBSCAN** (**Density-based spatial clustering of applications with noise**) είναι ένας αλγόριθμος βασισμένος στην πυκνότητα [Πυκνότητα = αριθμός σημείων (**MinPts**)] μέσα σε προκαθορισμένη ακτίνα (**Eps**). Οι παράμετροι αυτοί μαζί με το σύνολο δεδομένων αποτελούν τις παραμέτρους εισόδου του αλγορίθμου. Ο DBSCAN έχει πολυπλοκότητα υπολογισμών O(*n*) όπου n το πλήθος των σημείων. Μετά την εφαρμογή του αλγορίθμου τα σημεία διαχωρίζονται σε:

- **Βασικά** (core): ένα σημείο για το οποίο υπάρχουν περισσότερα από ένα προκαθορισμένο αριθμό (MinPts) σημεία σε ακτίνα Eps Αυτά είναι τα σημεία που βρίσκονται στο εσωτερικό μιας συστάδας.

- **Οριακά** (border): ένα σημείο για το οποίο υπάρχουν λιγότερα από ένα προκαθορισμένο αριθμό (MinPts) σημεία σε ακτίνα Eps, αλλά είναι στη γειτονιά ενός βασικού σημείου.
- **Θορύβου** (noise): ένα σημείο που δεν είναι ούτε βασικό ούτε οριακό.



2.1. Εφαρμογή στο σύνολο δεδομένων mydata

Βήμα 1. Φορτώστε το σύνολο δεδομένων 'mydata' που υπάρχει στο αρχείο δεδομένων **'mydata.mat'**.

```
import scipy.io
import numpy as np
mat_file = scipy.io.loadmat( 'mydata.mat')
X = np.array(mat_file['X'])
```

Βήμα 2. Εκτελέστε τη μέθοδο DBSCAN για τις δύο πρώτες διαστάσεις του πίνακα X, δίνοντας τιμές στις παραμέτρους του αλγορίθμου ε=0.5 και MinPts=15 .

```
from sklearn.cluster import DBSCAN
dbscan = DBSCAN(eps=0.5, min_sampes=15).fit(X)
IDX = dbscan.labels_
```

Βήμα 3. Παρουσιάστε σε γράφημα διασποράς τις τιμές των δύο διαστάσεων του πίνακα X.

```
import matplotlib.pyplot as plt
plt.figure(1)
```

```
plt.scatter(X[:,0],X[:,1])
plt.show()
```

Βήμα 4. Παρουσιάστε σε δεύτερο γράφημα τις συστάδες στις οποίες χώρισε τα δεδομένα η μέθοδος DBSCAN, καθώς και τον θόρυβο.

```
import matplotlib.pyplot as plt
plt.figure(1)
plt.scatter(X[:,0],X[:,1], c=IDX)
plt.show()
```

2.2. Εφαρμογή στο σύνολο δεδομένων iris

Βήμα 1. Φορτώστε το σύνολο δεδομένων Iris που υπάρχει στο αρχείο δεδομένων και χρησιμοποιείστε μόνο τις διαστάσεις [2,3].

```
from sklearn.datasets import load_iris
meas = load_iris().data
X = meas[:, [2, 3]]
```

Βήμα 2. Εκτελέστε τη μέθοδο DBSCAN για τις δύο πρώτες διαστάσεις του πίνακα X δίνοντας τιμές στις παραμέτρους του αλγορίθμου ε=0.1 και MinPts=5 .

```
from sklearn.cluster import DBSCAN
dbscan = DBSCAN(eps=0.1, min_samples=5).fit(X)
IDX = dbscan.labels_
```

Βήμα 3. Παρουσιάστε σε γράφημα διασποράς τις τιμές των δύο διαστάσεων του πίνακα X.

```
import matplotlib.pyplot as plt
plt.figure(1)
plt.scatter(X[:,0],X[:,1])
plt.show()
```

Βήμα 4. Παρουσιάστε σε δεύτερο γράφημα τις συστάδες στις οποίες χώρισε τα δεδομένα η μέθοδος DBSCAN, καθώς και τον θόρυβο.

```
import matplotlib.pyplot as plt
plt.figure(1)
```

```
plt.scatter(X[:,0],X[:,1], c=IDX)
plt.show()
```

Βήμα 5. Επαναλάβετε την παραπάνω διαδικασία αφού πρώτα κανονικοποιήσετε τα δεδομένα του πίνακα X με τη μέθοδο zscore. Σχολιάστε το αποτέλεσμα.

```
import numpy as np
from scipy.stats import zscore
xV1 = zscore(X[:,0])
xV2 = zscore(X[:,1])
Xnew = np.transpose(np.array([xV1,xV2]))
dbscan = DBSCAN(eps=0.1, min_samples=5).fit(Xnew)
IDX = dbscan.labels_
plt.figure(1)
plt.scatter(Xnew[:,0],Xnew[:,1], c=IDX)
plt.show()
```

Πέραν των ανωτέρω τμημάτων κώδικα, συστήνεται να γίνει χρήση των προγραμμάτων Python που έχουν αναρτηθεί στην πλατφόρμα eclass.

Η εργασία θα παραδοθεί έως τις 30/12/2024, 14:00 μέσω της πλατφόρμας ηλεκτρονικής μάθησης eclass.uniwa.gr. Παραδοτέα θα είναι συμπιεσμένο αρχείο *AM.rar* με όνομα τον αριθμό μητρώου (π.χ. 21*390100.rar*), που θα περιλαμβάνει τον κώδικα (αρχεία .py με επαρκή σχολιασμό) και αρχείο κειμένου σε μορφότυπο pdf με την αξιολόγηση που ζητείται.

Ο βαθμός της εργασίας θα συνεισφέρει κατά 25% στον τελικό βαθμό.