K-Means

Geschreven door: Matthijs Koelewijn

Uitleg code

Call tree

main

……..Data__init__

……..Data.assign_labels

……..data_min_max

……..Data.normalise

……..best_clustering

……..create_random_centroids

……..KMeans

……..numpy.array_equal

……..KMeans

……..validate_clusters

……..plot_results

……..matplotlib.pyplot.plot

Totaal-uitleg

Het K-Means-programma begint in de **main** en roept vervolgens verschillende andere functies aan om de data te initialiseren, te normaliseren en te analyseren.

In eerste instantie worden drie **Data** objecten geïnitialiseerd. Deze objecten bevatten elk één dataset: "dataset1.csv", "validation1.csv" en "days.csv". Het initialisatieproces resulteert in drie objecten genaamd training_data, validation_data en unclassified_data.

Om normalisatie op deze **Data** objecten toe te kunnen passen, willen we alle minimale en maximale datapunten van alle datasets bij elkaar hebben. Daarom wordt de functie **data_min_max** uitgevoerd. Binnen deze functie worden alle datasets aan elkaar geplakt in een array via de **numpy.concatenate** functie.

Nu we alle minimale en maximale datapunten hebben, kunnen we normalisatie toepassen op alle drie de **Data** objecten. We roepen de functie **Data.normalise** aan en geven deze minimale en maximale datapunten mee.

Het volgende wat wordt gedaan is de K-means clustering analyses uitvoeren op de genormaliseerde training dataset voor verschillende k-waarden, variërend van 1 tot en met `num_iterations` (in dit geval 100). De functie **best_clustering** wordt gebruikt om de beste clustering te bepalen voor elke kk-waarde voor `frequency=10` op te geven, wat betekent dat de clustering tien keer wordt uitgevoerd en het gemiddelde van de uitvoeringen wordt genomen.

Binnen de functie **best_clustering** worden herhaaldelijk drie functies aangeroepen **create_random_centroids, Kmeans** en **validate_clusters.**

De **create_random_centroids** functie maakt een lijst van willekeurige centroids aan. Deze lijst wordt gebruikt in de functie **KMeans**.

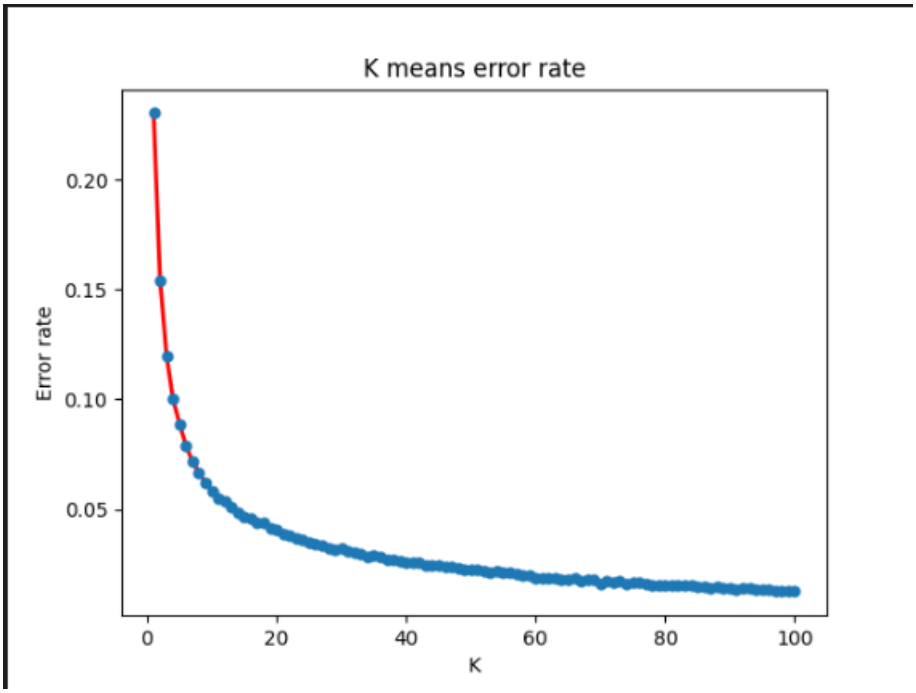
De functie **KMeans** voert het eigenlijke KMeans-algoritme uit, waarbij de datapunten worden toegewezen aan de dichtstbijzijnde centroid en de nieuwe centroids worden berekend op basis van de gemiddelde waarden van de datapunten in elk cluster. De functie blijft de clustering herhalen totdat de centroids niet meer veranderen.

De functie **validate_clusters** neemt als input een lijst van clusters, een lijst van centroids en de totale hoeveelheid data. Het berekent de totale afstand tussen de data en de centroids, verdeelt deze door de hoeveelheid data en geeft het gemiddelde als output terug.

De functie **best_clustering** geeft de best_error_rate terug die wordt toegevoegd aan de lijst best_clusters.

Ten slotte wordt de functie **plot_results** aangeroepen om de error rate van de K-means clustering analyse weer te geven voor elk aantal clusters(k) dat is getest. De plot wordt geplot door middel van matplotlib.pyplot. De plot toont het verband tussen het aantal clusters en de error rate van de clustering analyse.

Resultaten



Ik heb de error rate geplot voor verschillende waarden van K en ik heb vastgesteld dat de error rate hoog blijft wanneer K kleiner is dan 15. Dit komt waarschijnlijk doordat de clusters nog niet zo goed gedefinieerd zijn, en de datapunten dus nog steeds relatief ver van hun centroids liggen.

Maar naarmate ik K verhoog, worden de clusters meer fijnmazig en worden de datapunten beter toegewezen aan hun respectievelijke clusters, waardoor de error rate afneemt. Het lijkt erop dat de error rate verder afneemt naarmate K nog verder toeneemt, wat ook te verwachten is, omdat er meer centroids beschikbaar zijn om de datapunten in kleinere groepen te verdelen.

Het is belangrijk om te onthouden dat er geen "juiste" waarde is voor K, omdat het afhangt van de aard van de gegevens en de gewenste clusterresolutie. Ik moet dus verschillende waarden van K onderzoeken en evalueren welke het beste werken voor mijn specifieke gegevens en doelen.

Antwoorden

Om de vraag te beantwoorden “How many clusters can you (reliably) detect?” kijken we naar de scree plot waarin de error rate wordt uitgezet tegen het aantal clusters(k). Het punt waarop de error rate sterk begint af te nemen en vervolgens minder snel afneemt, beschouwen we als het optimale aantal clusters.

Als we naar de scree plot kijken bij resultaten, lijkt het punt waar de error rate sterk begint af te nemen rond 5. Dit sugereert dat er ongeveer 5 betrouwbare clusters zijn in de dataset.