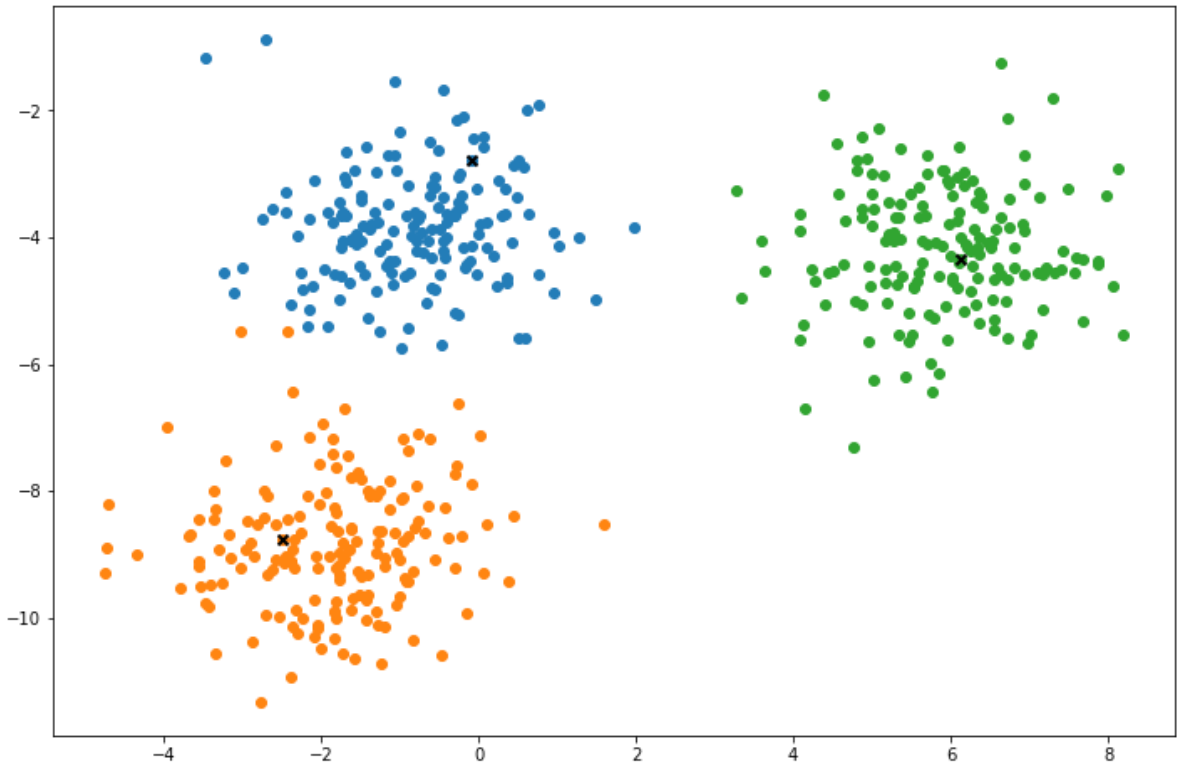
Četvrti domaći zadatak iz Tehnika i metoda analize podataka

Danilo Veljović, broj indeksa 1120

Opis skupa podataka

Za potrebe testiranja implementiranog algoritma, najjednostavnije je bilo iskoristit funkciju iz biblioteke sklearn, make_blobs() koja na osnovu prosleđenih parametara koji su: broj centara, broj uzoraka, broj atributa može da generiše klastere sa određenom količinom smetnje, tako da olakšaju testiranje algoritma za klasterizaciju. U ovom slučaju je odlučneo da se kreiraju 3 klastera, ukupno da bude 500 podataka i da postoje dva atributa u datasetu. Ovako generisani datasetovi biće svaki put različiti.

Primer jednog klastera je dat na slici 1.



Način implementacije algoritma

Prvo je potrebno implementirati funkciju rastojanja. U ovom slučaju to će biti euklidsko rastojanje i ono je dato u funkciji **euclidean_distance()**. Nakon toga implementiramo Kmeans klasu koja predstavlja apsktraciju K means clustering modela. U konstruktoru se prosleđuje broj klastera (defaultno postavljen na 5), maksimalni broj iteracija i opcioni parametar plot_steps koji ukazuje na to da li je potrebno ili ne da se grafički predstavljaju koraci prilikom određivanja klastera.
U konstruktoru takođe kreiramo prazne klastere i centroide. Klasteri su predstavljeni listom rednih brojeva uzoraka iz dataseta. Centroide su predstavljene listom vektora, gde je svaki od tih vektora, sredina nekog klastera.

Pošto je ovo tehnika nenadgledanog učenja, potrebno je samo implementirati predict() metodu. Grubi delovi predict() metode su, najpre inicijalizacija centroida, a nakon toga se radi optimizacija,

tj ažuriraju se klasteri i pomeraju se centroide. Inicijalizacija centroida se radi tako što se pomoću funkcije choice(), izabere nasumični podskup uzoraka i dodeli se nekom klasteru, ti podskupovi se ne preklapaju i elementi se ne ponavljaju.

Optimizacija algoritma se radi tako što se prvo ažurira klaster odgovarajućim vrednostima, što je definisano u create_clusters() funkciji. U ovoj funkciji se uzorci dodeljuju najbližoj centroidi da bi se kreirali klasteri. U ovoj funkciji prolazimo kroz kroz sve uzorke, tražimo indeks najbliže centroide kroz pomoćnu funkciju closest_centroid(). Kada je nađen indeks centroide, tražimo njen klaster i onda u taj klaster kačimo indeks trenutnog podataka koji obrađujemo.

Funkcija closest_centroid sadrži listu vektora, distances, u kojoj se nalaze udaljenosti svake od centroida i trenutnog uzorka. Funkcijom argmin nalazimo indeks minimalnog rastojanja, tj indeks centroide koja je minimalno udaljena od uzorka.

Nakon poziva create_clusters() funkcije, potrebno je ažurirati centroide. Čuvamo stare vrednosti centroida da bi se proverilo da li je algoritam počeo da konvergira. Nakon toga za novo kreirane klastere pribavljamo vrednosti centroida funkcijom get_centroid(). Funkcijom is_converged(), proveravamo razliku između starih centroida i novih i proveravamo da li je algoritam konvergirao ili nije. Uslov konvergencije je ovde, ako se stare i nove centroide ne razlikuju nimalo. Ako je algoritam konvergirao, funkcijom get_cluster_labels vraćamo za svaki podatak labelu klastera kojoj on pripada. Tj vraćamo indeks klastera u kom se nalazi.

Funkcija get_centroid() računa srednju vrednost svakog klastera, to pakuje u niz i vraća. Pomoćna funkcija plot() daje grafički prikaz svakog koraka, tj kako se centroida pomerala u odnosu na svaku iteraciju.

Za klaster dat na slici 1, na slici 2. je prikazana vrednost klastera i centroida nakon konvergencije.