**Slovenská technická univerzita v Bratislave**

**Fakulta informatiky a informačných technológii**

MNIST klasifikátor

**Zadanie č. 3a**

**Popis zadania**

Úlohou bolo vytvoríte neurónovú sieť na klasifikáciu ručne písaných číslic z dátového súboru MNIST. Zobrať súbor údajov zo 60 000 obrázkov na trénovanie a 10 000 obrázkov na testovania, veľkosť obrázkov je 28 x 28 a reprezentuje jednu číslicu od 0 do 9.

Použiť doprednú neurónovú sieť (viacvrstvový perceptrón) a natrénujte ju pomocou algoritmov SGD, SGD s momentom a ADAM, použiť knižnicu PyTorch. Okrem trénovacej a testovacej chyb, odmerať aj presnosť modelu (t. j. koľkokrát model predikoval správnu triedu na testovacej množine).

**Reprezentácia datasetov**

**Postup riešenia.**

Na zatáčku programu bolo importovane knižnica pre náhodne generovanie, matematická knižnica, knižnica pre načertenie štatistických grafov, knižnica pre kontrolu času a pre výpočet vzdialenosti.

* import math
* import copy
* import random
* import time
* import matplotlib
* import matplotlib.colors as mcolors
* matplotlib.use('TkAgg')
* import matplotlib.pyplot as plt
* import numpy as np
* from scipy.spatial.distance import cdist, pdist, squareform

Tretiu knižnicu je potrebne nainštalovať z príkazom “pip install matplotlib“.

Taktiež je potrebná nainštalovať scipy.

Po implementácii knižníc bolo zadefinovaných niekoľko najdôležitejších funkcií, takých ako:

Ďalej som načítal súbor, v ktorom mal zapísanú konfiguráciu k algoritmu v formáte:

* [centroid-1, medoid-2]: \_\_\_
* Count\_of\_points: \_\_\_
* Count\_of\_first\_points: \_\_\_
* Logout: off
* Text output: off
* Linkage[1 - single, 2 - average]: 1

\_\_\_ - číselné parametre

A pridelil som tieto parametre k zodpovedajúcim poliam. V nasledujúcemu kroku boli vygenerovane začiatočne body.

Po vygenerovaní 20 náhodných bodov sme vygenerovali ďalších 20000 bodov, tieto body neboli generované úplne náhodne, ale nasledovným spôsobom:

1. Náhodne vyberáme jeden zo **všetkých doteraz vytvorených** bodov v 2D priestore. (nie len z prvých 20)  
   Ak je bod príliš blízko okraju, tak zredukujeme príslušný interval, uvedený v nasledujúcich dvoch krokoch.
2. Vygenerujem náhodné číslo X\_offset v intervale od -100 do +100
3. Vygenerujem náhodné číslo Y\_offset v intervale od -100 do +100
4. Pridajte nový bod do 2D priestoru, ktorý bude mať súradnice ako náhodne vybraný bod v kroku 1, pričom tieto súradnice budú posunuté o X\_offset a Y\_offset

Aby nespočítavať vzdialenosti každý raz urobil som maticu, kde sa uložil hodnoty vzdialenostnej. Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

***Príklad pre 8 klastrov***

Pre reprezentáciu klastrov bola vytvorená trieda Cluster, aký vyzerá nasledovne:

* class Cluster:  
   def \_\_init\_\_(self, points):  
   self.points = np.array(points)  
    
   if mode == 2:  
   self.metoid = self.calculate\_metoid()  
   else:  
   self.centroid = self.calculate\_centroid()  
    
   def calculate\_centroid(self) -> np.ndarray:  
   return np.mean(self.points, axis=0)  
    
   def calculate\_metoid(self) -> np.ndarray:  
   distances = cdist(self.points, [self.calculate\_centroid()]) # [self.centroid]  
   closest = distances.argmin()  
   return self.points[closest]  
    
   def add\_points(self, points: np.ndarray):  
   self.points = np.vstack((self.points, points))  
    
   if mode == 2:  
   self.metoid = self.calculate\_metoid()  
   else:  
   self.centroid = self.calculate\_centroid()

V tejto triede máme metódy na nájdenie centroida alebo metoida, metóda na pridávanie bodov do klastra.

Pre generácií metoida používam príznak mode, ktorý slúži na indikáciu, čo spočítavam, zobrazím na obrazovke(centroid alebo metoid).

Zadefinovane funkcie:

* def calculate\_distance\_between\_centroid(cluster1, cluster2):
* def create\_distance\_table(clusters):
* def create\_min\_distances\_table(clusters, distances\_table):
* def find\_min\_distance\_in\_min\_dist\_table(min\_distances\_table):
* def merge\_closest\_pairs():
* def calculate\_min\_distance\_between\_centroids():
* def agglomerative\_clustering():
* def calculate\_average\_center(clusters: list[Cluster]) -> np.ndarray:
* def visualize\_clusters(clusters: list[Cluster]):

Opis metód:

* calculate\_distance\_between\_centroid – na výpočet vzdialenosti medzi centroidmi alebo metoidmi
* create\_distance\_table – vytvorenie matice v ktorej vzdialenosti v dolnom rohu.
* create\_min\_distances\_table – vytvorenie zoznamu klastrov a ich najbližších susedov.
* find\_min\_distance\_in\_min\_dist\_table – najedenie najbližšej pary v zozname klastrov a ich najbližších susedov.
* merge\_closest\_pairs – spojenie najbližších par klastrov a aktualizácia matice a zoznamu najbližších susedov.
* calculate\_distance\_between\_clusters – ohodnotenie najbližšej vzdialenosti medzi centroidmi alebo metoidmi.
* agglomerative\_clustering – hlavný cyklus algoritmus pre spojenie najbližších par a overovania podmienky ukončenia kastrovania.
* calculate\_average\_center – stred medzi všetkými centroidmi alebo metoidmi.

Aglomeratívne zhlukovanie prebieha dovtedy, kým počet klastrov nedosiahne nastavenú hodnotu alebo priemerná vzdialenosť bodov v klastroch nie je väčšie ako 500. Počas cyklu zhlukujeme najbližšie body v metóde merge\_closest\_pairs , takým spôsobom, že nájdeme zo zoznamu najbližších klastrov najbližšiu paru urobíme aktualizáciu pre prvý index v matice vzdialenostnej a odstránime druhy riadok a stĺpec. Pre tom to kroku môžeme použiť jednu z dvoch zvolených metrík pre spočítanie hodnoty vzdialenosti nového klastra. Pridáme bod do prvého klastra a odstránim klaster zo zoznamu klastrov. Potom ideme aktualizovať zoznam najbližších par. Vyberiem z neho druhy klaster, aký sme odstránili. Potrebujeme vedieť najbližšieho suseda ku novému klastru, preto idem a hľadáme najbližšieho suseda, taktiež zároveň pozeráme, či nemali ku sebe najbližšie index druhého klastra, ak áno ,musíme spraviť aktualizáciu. Aktualizujeme aj ak hodnota nového klastra vyhovuje viac pre iný. Pokračujeme kastrovanie, kým nebudeme mat 1.5 od začiatočných 20 klastrov, tu už spomalíme a robíme kópiu predchádzajúceho zoznamu klastrov. Potom prepočítame vzdialenosť všetkých bodov jednotného klastra, ak je väčšia ako 500 vrátime sa do predchádzajúceho zoznamu.

Potom ohodnotím úspešnosť všetkých klastrov a dáme grafické dáta na obrazovku.

Porovnávanie algoritmov

Pre maticu rozmermi n x n zložitosť matice je (O(n^2) pre n bodov). Čo je veľmi neefektívne pre použitie, keď duplikujeme jednu polovicu matice na dve strany matice, preto pri implementácii používame len polovicu matice.

Navrhnutá implementácia algoritmus je dosť efektívna pri malom pocte bodov, ale nie je dosť efektívnym pre veľkom. S pomocou tabuľky najbližších klastrov snažím sa zmenšiť proces nájdenia minimálnej vzdialenosti, aby neprechádzať celu polovicu matice. Nie je veľmi úspešný algoritmus a vyžadoval by zlepšenia. V procese skúšania rôznych implementácii vyskúšal som nájdenie najbližších susedov cez prechádzanie celej matice, nebol to veľmi účinný metód. Taktiež vyskúšal metód vypočítania matice vzdialenostnej, vytvorenie najbližších unikátnych par a usporiadanie ich podlá vzdialeností od najmenšej k najväčšej. Potom podľa pocta klastrov spojím určitý počet najlepších klastrov. Čo môže byt oveľa efektívnejšie než dva predchádzajúce metódy. Nevýhodou týchto metód je veľký straty pamäti pri tvorbe matice a súvisiacich s tými konštrukciami. Napriek tým metódam efektívnejšie bolo by nepoužívať maticu vzdialenostnej v takej implementácii.

S najpopulárnejších metrík single, complete, average , ward a iných jednej z najrýchlejší bude single linkage t.j. pretože vyžaduje iba sledovanie najbližších bodov.

Testovanie

Počas testovania používal rôzne metódy vypočítania vzdialeností medzi klastrami, výsledky zo vstupnými parametrami a ohodnotením budú uvedene pri jednotných testoch.

Test 1

Vstup: Centroid, začiatočne body: 20, počet bodov 1020. Podmienka ukončenia: žiaden klaster nemá priemernú vzdialenosť bodov od stredu viac ako 500. Average Linkage.

1. Зображення, що містить текст, схема, ряд, Графік

   Автоматично згенерований описЗображення, що містить текст, схема, знімок екрана, Графік

   Автоматично згенерований описЗображення, що містить текст, знімок екрана, схема

   Автоматично згенерований опис

Výstup: Čas: ~3-5s. Úspešnosť: Successful clustering

Na uvedenom príklade vidíme, že sa dá zvýšiť minimálna vzdialenosť medzi klastrami. Pridal som červený bod na základnú plochu, aký označuje priemerný bod medzi všetkými centroidmi alebo metoidmi. Ten bod ukazuje v ktorej časti sa nachádza najväčší počet klastrov. A da možnosť ďalšie manipulovať zo centroidmi alebo metoidmi.

Test 2

Vstup: Metoid, začiatočne body: 20, počet bodov 1020. Podmienka ukončenia: žiaden klaster nemá priemernú vzdialenosť bodov od stredu viac ako 500. Average Linkage.

1. Зображення, що містить текст, схема, ряд, знімок екрана

   Автоматично згенерований описЗображення, що містить текст, схема, Графік, знімок екрана

   Автоматично згенерований описЗображення, що містить текст, знімок екрана, схема, Графік

   Автоматично згенерований описЗображення, що містить текст, знімок екрана, Барвистість, схема

   Автоматично згенерований опис

Výstup: Čas: 3.92s. Úspešnosť: Successful clustering

Ako aj v predchádzajúcemu príklade, sa dá zvýšiť minimálna vzdialenosť medzi klastrami.

Test 3

Vstup: Centroid, začiatočne body: 20, počet bodov 5020. Podmienka ukončenia: žiaden klaster nemá priemernú vzdialenosť bodov od stredu viac ako 500. Average Linkage.

1. Зображення, що містить текст, схема, ряд, знімок екрана

   Автоматично згенерований описЗображення, що містить текст, схема, знімок екрана, Графік

   Автоматично згенерований описЗображення, що містить текст, знімок екрана, Барвистість

   Автоматично згенерований опис

Výstup: Čas: 321.09s. Úspešnosť: Unsuccessful clustering

Unsuccessful clustering môže nastáť z podmienky v kóde, že keď mame if len(clusters) < 20, len vtedy začnem kontrolovať priemernú vzdialenosť bodov od stredu.

Test 4

Vstup: Metoid, začiatočne body: 20, počet bodov 5020. Podmienka ukončenia: žiaden klaster nemá priemernú vzdialenosť bodov od stredu viac ako 500. Average Linkage.

1. Зображення, що містить текст, схема, ряд, Графік

   Автоматично згенерований описЗображення, що містить текст, схема, знімок екрана, Графік

   Автоматично згенерований описЗображення, що містить знімок екрана, текст

   Автоматично згенерований опис

Výstup: Čas: 306.13s. Úspešnosť: Successful clustering

Zlepšili sme podmienky a plánujeme skúsiť iní Linkage.

Test 5

Vstup: Centroid, začiatočne body: 20, počet bodov 5020. Podmienka ukončenia: žiaden klaster nemá priemernú vzdialenosť bodov od stredu viac ako 500. Single Linkage.

1. Зображення, що містить знімок екрана

   Автоматично згенерований описЗображення, що містить текст, знімок екрана

   Автоматично згенерований опис

Výstup: Čas: 267.05s. Úspešnosť: Unsuccessful clustering

S tohto príkladu vidíme, že s Single funguje to rýchlejšie.

Test 6

Vstup: Metoid, začiatočne body: 20, počet bodov 5020. Podmienka ukončenia: žiaden klaster nemá priemernú vzdialenosť bodov od stredu viac ako 500. Single Linkage.

1. Зображення, що містить знімок екрана, текст, схема

   Автоматично згенерований опис

Výstup: Čas: 267.18 s. Úspešnosť: Successful clustering

Test 7

Vstup: Metoid, začiatočne body: 20, počet bodov 10020. Podmienka ukončenia: žiaden klaster nemá priemernú vzdialenosť bodov od stredu viac ako 500. Average Linkage.

1. Зображення, що містить текст, знімок екрана, програмне забезпечення

   Автоматично згенерований опис

Výstup: Čas: 2967.84s. Úspešnosť: Unsuccessful clustering

Test 8

Vstup: Metoid, začiatočne body: 20, počet bodov 10020. Podmienka ukončenia: žiaden klaster nemá priemernú vzdialenosť bodov od stredu viac ako 500. Single Linkage.

1. Зображення, що містить текст, схема, знімок екрана, Графік

   Автоматично згенерований описЗображення, що містить текст, знімок екрана, схема

   Автоматично згенерований опис

Výstup: Čas: 3257.17s. Úspešnosť: Successful clustering

Boli trochu zmenene podmienky.

Test 8

Vstup: Metoid, začiatočne body: 20, počet bodov 10020. Podmienka ukončenia: žiaden klaster nemá priemernú vzdialenosť bodov od stredu viac ako 500. Single Linkage.

1. Зображення, що містить текст, схема, ряд, Графік

   Автоматично згенерований описЗображення, що містить текст, схема, знімок екрана, Графік

   Автоматично згенерований описЗображення, що містить текст, знімок екрана, Барвистість, схема

   Автоматично згенерований опис

Výstup: Čas: 2486.12s. Úspešnosť: Successful clustering

Ďalej som skúšal rôzne podmienky ukončenia a prišiel k tomu, algoritmus neveľmi správne robil kópiu predchádzajúcich klastrov, preto som použil knižnicu pre spoľahlivé kopírovanie.

Test 9

Vstup: Centroid, začiatočne body: 20, počet bodov 50020. Podmienka ukončenia: žiaden klaster nemá priemernú vzdialenosť bodov od stredu viac ako 500. Single Linkage.

1. Зображення, що містить текст, схема, ряд, знімок екрана

   Автоматично згенерований описЗображення, що містить текст, схема, знімок екрана, Графік

   Автоматично згенерований описЗображення, що містить текст, знімок екрана, схема, Барвистість

   Автоматично згенерований опис

Pre lepší pochopenie vytvorených klastrov vypisal som priemernú vzdialenosť bodov od stredu:  
Cluster 1 with 1390 points, Centroid: [-2002.28057554 1978.64964029], Mean distance: 497.9250935438831

Cluster 2 with 321 points, Centroid: [4056.43302181 3021.13084112], Mean distance: 391.1596334582285

Cluster 3 with 210 points, Centroid: [ 710.23809524 -656.12857143], Mean distance: 135.89011232593825

Cluster 4 with 624 points, Centroid: [ 1448.38782051 -1868.45032051], Mean distance: 169.71591222229299

Cluster 5 with 977 points, Centroid: [ 304.58341863 -4518.16683726], Mean distance: 292.3880084849645

Cluster 6 with 423 points, Centroid: [3065.36170213 759.64539007], Mean distance: 159.55055627716283

Cluster 7 with 257 points, Centroid: [-394.6692607 220.87937743], Mean distance: 136.1529779945107

Cluster 8 with 229 points, Centroid: [2259.73362445 2062.2139738 ], Mean distance: 139.68105626894885

Cluster 9 with 39 points, Centroid: [-3086.56410256 4061.02564103], Mean distance: 90.86083419156598

Cluster 10 with 15 points, Centroid: [-2305.93333333 163.33333333], Mean distance: 84.14517989041111

Cluster 11 with 62 points, Centroid: [ 4009.98387097 -1875.5 ], Mean distance: 129.50646349303162

Cluster 12 with 236 points, Centroid: [ 72.62711864 2410.73305085], Mean distance: 117.20169477651596

Cluster 13 with 34 points, Centroid: [ 4255.82352941 -2816.79411765], Mean distance: 107.9730294847895

Cluster 14 with 181 points, Centroid: [-3999.51933702 -4290.29834254], Mean distance: 152.32101832807635

Cluster 15 with 21 points, Centroid: [ 2094.19047619 -4126.14285714], Mean distance: 73.45965673340626

Cluster 16 with 1 points, Centroid: [-3835. 4494.], Mean distance: 0.0

Výstup: Čas: 347.15s. Success rate of clustering: 100.00%

Test 10

Vstup: Centroid, začiatočne body: 20, počet bodov 20020. Podmienka ukončenia: žiaden klaster nemá priemernú vzdialenosť bodov od stredu viac ako 500. Average Linkage. Зображення, що містить знімок екрана, текст

Автоматично згенерований опис

1. Зображення, що містить текст, знімок екрана

   Автоматично згенерований описЗображення, що містить текст, знімок екрана

   Автоматично згенерований описЗображення, що містить текст, знімок екрана, Барвистість

   Автоматично згенерований описЗображення, що містить текст, знімок екрана, схема

   Автоматично згенерований опис

Výstup: Čas: 17291.66s. Úspešnosť: Successful clustering

Tento test ukazuje, že zlučovač vie spraviť dobre kastrovanie, len potrebuje zbytočne veľa času. Musíme spraviť jeho optimalizáciu.

Záverečný test na správnosť vytvoreného systému.

Vstup: Metoid, začiatočne body: 20, počet bodov 10020. Podmienka ukončenia: žiaden klaster nemá priemernú vzdialenosť bodov od stredu viac ako 500. Single Linkage.

1. Зображення, що містить текст, схема, ряд, знімок екрана

   Автоматично згенерований описЗображення, що містить текст, схема, знімок екрана, Графік

   Автоматично згенерований описЗображення, що містить текст, знімок екрана, схема

   Автоматично згенерований опис

Výstup:

Cluster 1 with 2381 points, Metoid: [-3921 -996], Mean distance: 165.3357867505383

Cluster 2 with 2014 points, Metoid: [-503 -849], Mean distance: 199.5323749227025

Cluster 3 with 1415 points, Metoid: [ 2101 -4275], Mean distance: 430.0821229405799

Cluster 4 with 496 points, Metoid: [ 3826 -4591], Mean distance: 258.6831106991255

Cluster 5 with 269 points, Metoid: [4444 -105], Mean distance: 158.0258348738357

Cluster 6 with 1109 points, Metoid: [-2162 -4736], Mean distance: 151.56590571857228

Cluster 7 with 815 points, Metoid: [ 321 3489], Mean distance: 151.92723579842516

Cluster 8 with 308 points, Metoid: [-3209 4813], Mean distance: 132.62549651934023

Cluster 9 with 253 points, Metoid: [-1813 2388], Mean distance: 145.6943508144369

Cluster 10 with 658 points, Metoid: [1852 3298], Mean distance: 172.0752765360418

Cluster 11 with 196 points, Metoid: [-2105 -2775], Mean distance: 153.7218920657062

Cluster 12 with 106 points, Metoid: [-3304 736], Mean distance: 188.24232623466614

Success rate of clustering: 100.00%

Program execution time: 2725.91 seconds

Porovnanie s inými implementáciami.

Pre porovnaní použil som tri rôzne spôsoby implementácii, poväzujem, že všetky návrhy majú rovnaký základný postup pre ukončovaciu podmienku:

* Vlastne navrhnutie agloerativneho zhlukovania
* Zhlukovanie prvých n lepších par zo usporiadaného zoznamu najbližších bodov
* Zhlukovanie rôznych častej paralelne a zhlukovanie vytvorených klastrov zo všetkých tých častej.

Výsledky:

1. 47.33s.
2. 24.27s
3. 27.25s.

Prvý navrhnutý postup ma veľkú časovú zložitosť v porovnaní s inými. Druhy vyžaduje oveľa väčšiu kapacitu pamäti, ako prvý a je oveľa rýchlejšie. Tretia možnosť kombinuje lepšie aspekty s prvého aj druhého. Je dosť rýchla a nepotrebuje dosť veľkú spotrebu pamäti.

**Zhodnotenie:**

Vytvorený algoritmus je určený na klasifikáciu pomocou aglomeratívneho zhlukovania. Algoritmus splna podmienku, aby žiaden klaster nemal priemernú vzdialenosť bodov od stredu viac ako 500 s veľmi vysokou šancou na úspech, čo umožňuje spoľahlivo klasifikovať body k najvhodnejším klastrom a zobraziť výsledok na obrazovke.

Algoritmus je rýchly a schopný nájsť optimálne alebo blízke k optimálnemu riešeniu pri adekvátnych podmienkach. Malý počet klastrov dával možnosť rýchlo a správne kastrovať, len pri veľkom počte bolo to veľmi náročne s časového hľadiska a viedlo k veľkým spotrebám pamäti.

Taktiež program umožňuje konfigurovať konfiguráciu pred behom programu. Čo výrazne zrýchľuje prístup k jednotným špecifikáciám a prispôsobeným nástrojom pre určite potreby klienta. Ako napríklad používateľ môže si zvoliť metriku, aká najviac mu vyhovuje.

Dosť dôležitou úlohou bolo správne vybrať dobrý postup pre zjednotenie klastrov. Algoritmus bol testovaný rôznymi spôsobmi a boli vygenerovane najvhodnejšie parametre pre klasifikáciu a overenie podmienky ukončenia zhlukovania. Boli porovnané rôzne spôsoby implementácie, ako napríklad rôzne metriky na porovnávanie vzdialeností medzi klastrami, vďaka čomu je zrozumiteľne nevýhody a výhody daného návrhu implementácie zhlukovania.

**Vylepšovanie a dolaďovanie**

Možnosti sú prepísať logiku algoritmu pre menšie pamäťové využitie, prípadne pridať viacvláknovosť pre paralelne spracovanie rôznych časti mapy. Napríklad jednou z možnosti je rozdeliť mapu na strži kvadrante a paralelne spracovať ich, výsledne klastre štyroch kastrovaní dát do jedného zoznamu a skúsiť pokračovať spojenie klastrov. Pri porovnaní rôznych implantácii zhlukovania tato možnosť ukázala výrazne zvýšenie výkony s časového hľadiska.