Zadanie č. 3a: Klasifikácia*

Hlib Kokin ID: 117991
Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
xkokin@stuba.sk

21.11.2022

*Dokumentacia treteho zadania v predmete Umelá Inteligencia, ak. rok 2022/23, vedenie: Ivan Kapustík

Abstrakt

Dokumentácia k tejto úlohe obsahuje podmienky samotnej úlohy, popis použitých algoritmov a ich vlastností, testy a grafy s konkrétnymi výsledkami, na základe ktorých je možné posúdiť efektívnosť k-NN algoritmu.

Obsah

1 Zadanie

Podľa podmienok zadania som musel použiť klasifikačný algoritmus k-NN (K Nearest Neighbors) na klasifikáciu bodov generovaných v rôznych bodoch mapy (10000x10000 bodov) podľa farieb, červenej, zelenej, modrej a fialovej.

2 Rozbehanie Projektu

Svoju úlohu som sa rozhodol implementovať v jazyku Python, vo vývojovom prostredí PyCharm, na spustenie testovacej funkcie je potrebné nainštalovať balík MatPlotLib verzie 3.5.3, aby ste mohli zobraziť grafy novo vygenerovaných riešení

3 Inicializácia mapy a špecializácia riešenia

3.1 Reprezentácia údajov

Pri implementácii mojej úlohy som vytvoril dve triedy, jednu na definovanie sektora, ktorá sa používa iba v inicializačnej funkcii, sú tam dve premenné, ktoré ukladajú stred sektora, ako aj funkcia, ktorá vracia prvok druhej triedy so súradnicami stredu sektora, aby sa aktuálny bod dal porovnať so stredom sektora. Druhá trieda je trieda pre bod, obsahuje premenné obsahujúce súradnice samotného bodu, ako aj premenné, v ktorých je vyznačená farba bodu (1 - červená, 2 - zelená, 3 - modrá, 4 - fialová), číslo časti, v ktorej bol bod vygenerovaný (1 - časť červených bodiek, 2 - zelená atď.), ako aj číslo sektora, v ktorom sa tento bod nachádza.

3.2 Optimalizácia riešenia

Pre optimalizáciu riešenia bola zvolená metóda sektorovania, keďže na určenie najbližších bodov by sme potrebovali kontrolovať každý klasifikovaný bod, ktorého počet sa bude zakaždým zvyšovať, čo značne spomaľuje proces. Preto bude mapa riešenia rozdelená na n sektorov, v riešení je mapa rozdelená na 2500 sektorov, 50 sektorov za sebou a spolu 50 riadkov.

3.3 Inicializácia mapy a štartovych bodov

Bodová mapa je znázornená zoznamovým spôsobom so sektormi. Riešenie umožňuje meniť počet vytvorených bodov a rozdelenie na sektory (dá sa určiť konkrétny počet sektorov v jednom riadku), podľa normy stojí 50 sektorov v jednom rade, čo nám dáva 2500 sektorov na celú mapu, čo znamená, že v jednom sektore môže byť 40 000 bodov. Na úplnom začiatku pre každý počiatočný bod určíme sektor, v ktorom sa bude nachádzať, na to musíme sektory najskôr inicializovať, inicializácia prebieha vytvorením zoznamu s objektmi triedy sektora, každý objekt označuje stred sektora. sektore. Aby sme pochopili, v ktorom sektore sa bod nachádza, zoradíme zoznam všetkých sektorov podľa parametra najkratšej vzdialenosti do stredu sektora. Potom už vieme, v ktorom sektore sa bude nachádzať začiatočný bod, pridáme ho do zoznamu konkrétneho sektora a tiež jeho súradnice pridáme do zoznamu zablokovaných bodov. Na výstupe

teda dostaneme mapu s inicializovanými prázdnymi sektormi, okrem tých, do ktorých padli naše východiskové body, zoznam s blokovanými bodmi, ktorý sa pri vygenerovaní nových bodov rozšíri, ako aj zoznam so sektormi

3.4 Generacia bodov

Body sú generované v cykle n opakovaní, kde n je špecifikovaných podľa 100 000 (bodov). v tejto slučke počítadlo striedavo prepíname z 1. farby (červená) na 4. (fialová), keď dosiahneme 4. farbu, počítadlo sa vynuluje na 1. farbu. v každej iterácii zavoláme funkciu generate_point, v ktorej vyberieme bod z jeho časti farby s 99-percentnou pravdepodobnosťou (farba sa prenesie do argumentov), 1 % - vyberieme bod z celej mapy. Potom sa súradnice bodu pridajú do zoznamu blokovaných bodov a vytvorí sa bodový objekt s prázdnym farebným poľom a prázdnym sektorovým poľom, tento objekt funkcia vráti. Keď funkcia vráti bod, zavolá sa funkcia find_sector (index, ktorý vráti, zapíšeme do bodového poľa s názvom sektora) a bod sa pridá do zoznamu s nekvalifikovanými bodmi. Potom v slučke zavoláme kvalifikačnú funkciu pre každý bod v zozname nekvalifikovaných bodov.

4 k-NN algoritmus

Algoritmus funguje tak, že k nášmu aktuálnemu objektu vezmeme K najbližších prvkov a na základe ich vlastností zaradíme náš objekt do jednej zo skupín, v tomto prípade nájdeme 1, 3, 7 alebo 15 najbližších susedov k vybraný bod, spočítame počet susedov každej zo 4 farieb a následne náš bod zaradíme do skupiny najčastejšie sa vyskytujúcej farby.

4.1 Hľadanie suseda

Aby sme bod kvalifikovali, musíme najprv nájsť najbližších susedov bodu.

Aby sme teda našli k susedov bodu, pozrieme sa na susedné sektory a nie na všetky existujúce body, ktorých počet v určitom bode dosiahne 40 000. Vyhľadávanie v sektoroch je implementované tak, že algoritmus najprv berie sektory vo vzdialenosti 1 sektora od aktuálneho sektora, čím odtiaľ zbiera všetky body, ak by bodov nebolo dosť, pokračuje a pozerá sa na sektory vo vzdialenosti 2, s týmto konceptom hľadania sektorov s bodmi , je dôležité zabezpečiť, aby sa sektory indexu nevyberali viac a nie menej ako zoznam sektorov, preto, keď sme našli sektory okolo nás, všetky negatívne indexy a indexy väčšie ako posledný index v zozname (počet sektory - 1) sú odstránené.

Pri hľadaní sektorov na diaľku môžeme podmienečne prejsť za hranice mapy, čím sa dostaneme na inú mapu mapy, aby sme sa tomu vyhli, prestaneme brať sektory, keď sa dostaneme do sektora, ktorého index je deliteľný bez zvyšok počtom sektorov v rade (to bude znamenať, že sme dosiahli ľavý okraj, resp. nepohneme sa doľava), ale ak dosiahneme index, ktorý sa bezo zvyšku vydelí počtom sektory v rade mínus 1, tiež prestaneme ísť ďalej (na pravú stranu, zatiaľ čo prezeranie sektorov naľavo bude pokračovať). Keď sa teda dostaneme na okraj mapy, už nedostaneme ideálny štvorec sektorov v určitej vzdialenosti od aktuálneho sektora, ale výrez tohto štvorca

4.2 Funkcia klasifikacií

Keď nájdeme dostatok bodov zo susedných sektorov, zoradíme ich podľa vzdialenosti od zvoleného bodu a zozbierame štatistiku farieb prvých k bodov, pričom počet bodov určitej farby zapíšeme do zodpovedajúcich 4 premenných, potom zoraďte zoznam týchto 4 premenných, kde na prvom mieste bude farba, ktorá sa najčastejšie vyskytuje. Potom jednoducho priradíme túto farbu aktuálnemu bodu a pridáme ju do príslušného sektora mapy špecifikovaného v poli bodového objektu.

5 Testovacia funkcia

Na začiatku testovacej funkcie zavoláme inicializačné funkcie a vygenerujeme body v slučke, potom klonujeme mapy a nekvalifikované body trikrát, pre k = 3, 7 a 15, potom vytvorím zoznam n-tic troch prvkov , prvým prvkom je mapa s prvkami , druhým je hodnota k a tretím sú nekvalifikované body. Prejdením každej zo 4 n-tic získame všetky nekvalifikované body, ktoré sa v priebehu procesu pridajú na mapu podľa ich sektorov. Po kvalifikovaní všetkých bodov pre aktuálnu mapu zavoláme funkciu na zobrazenie mapy. Pre túto funkciu zbierame všetky body z každého sektora v prvom zozname a v druhom ich farby umyjeme a pomocou matplotlib.scatter nakreslíme mapu bodu

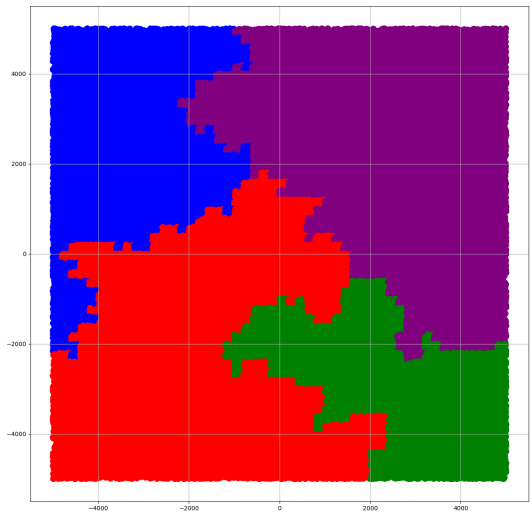
6 Výsledky testov

Testy boli na klasifikáciu 100 000 bodov, čo je o niečo viac ako stanovené minimum 40 000 bodov, bolo to urobené preto, aby bolo lepšie vidieť rozdiel v testovacích časoch s rôznymi hodnotami k.

Priemerný čas inicializácie pre 4 kopie máp a 4 kopie 100 000 nekvalifikovaných bodov trvá v priemere 300 sekúnd. Keďže pre každé k potrebujeme nove objekty, nenašiel som spôsob, ako znížiť časovú a pamäťovú náročnosť inicializácie.

6.1 k = 1

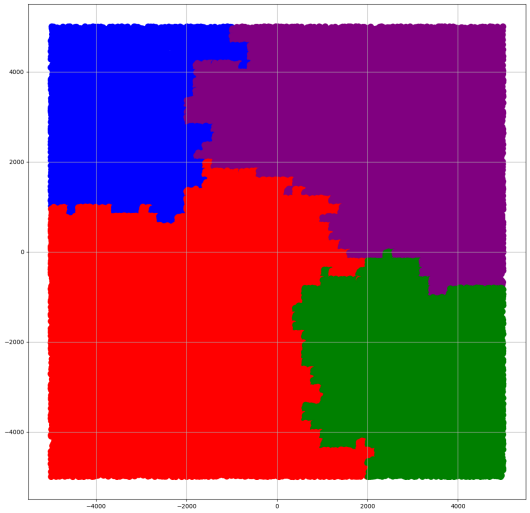
Pre k = 1 sa riešenie nájde v priemere za 3,4 sekundy, s presnosťou kvalifikácie 71%. Nižšie vidíte výsledok testu algoritmu k-NN pre k = 1. Okamžite vás upúta, že farby presahujú svoje štvorce a tvary rôznych farieb vyzerajú skôr ako náhodný tvar než obdĺžniky. Môžeme povedať, že kvalifikácia s pozornosťou 1 suseda nie je dostatočne presná.



Obr. 1: k-NN algoritmus pri k = 1

6.2 k = 3

Pre k = 3 sa situácia s výsledkom príliš nelíši od testu s k = 1, body sú stále kvalifikované na základe príliš malého počtu susedov, čo ovplyvňuje výsledky tak, že figúrky z rôznych farieb vyliezajú za svoje inicializačné zóny. Test sa skončil za 3,6 sekundy, čo je o niečo viac ako predchádzajúci, vzhľadom na to, že sú tam ďalší 2 susedia. Úspešnosť kvalifikácie 73%.

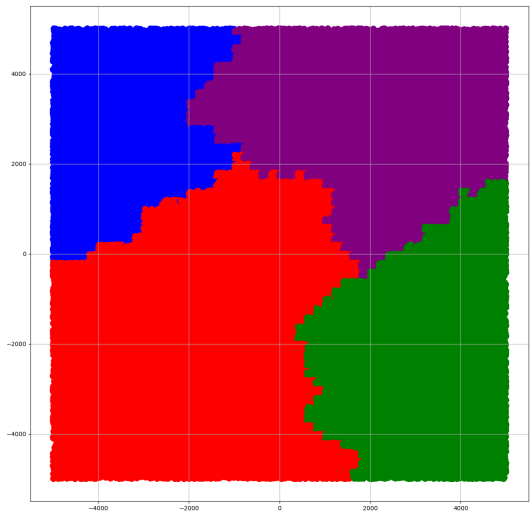


Obr. 2: k-NN algoritmus pri k = 3

6.3 k = 7

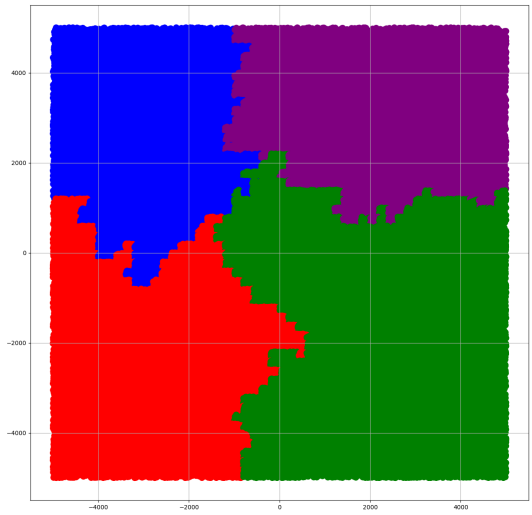
Výsledok pre k = 7 už vyzerá oveľa lepšie ako výsledky predchádzajúceho testu. Tu už vidíme, že zóny sú vedené viac monoliticky, bez toho, aby do seba po častiach liezli. V tomto teste sa ukázalo, že bodky červenej farby majú výhodu v množstve oproti bodkám iných farieb, ale to bolo len vďaka náhode. Po pohľade na výsledky testu pre k = 7 môžeme povedať, že tento počet susedov je oveľa úspešnejší ako 1 alebo 3, takže kvalifikácia nepríde s falošnými výsledkami v dôsledku pohľadu na príliš málo susedov.

Test sa skončil za 4,43 sekundy, trval relatívne dlhšie ako predchádzajúce dva vzhľadom na to, že počet susedov na testovanie je už o 4 viac ako predchádzajúci. Správnosť kvalifikácie - 75%

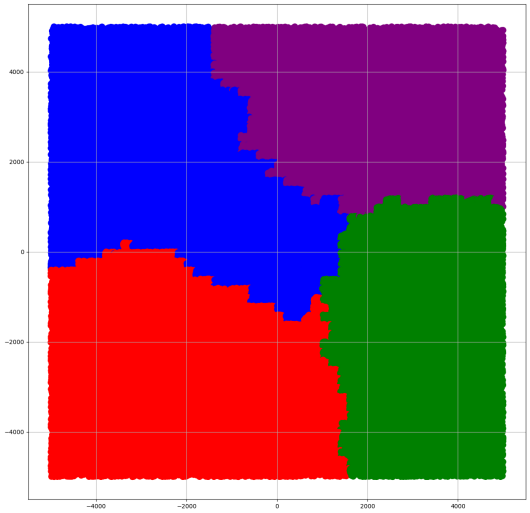


Obr. 3: k-NN algoritmus pri k = 1

Hoci v niektorých prípadoch náhodného generovania bodov môže byť ideálny počet susedov rovný 3, tak v jednom z testov dopadla kvalifikácia s k = 7 horšie ako s k = 3 (74%, resp. 76%)



Obr. 4: k-NN algoritmus pre k = 3

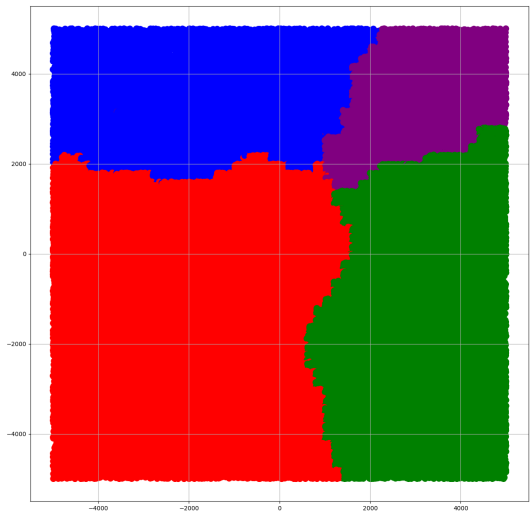


Obr. 5: k-NN algoritmus pre k = 7

6.4 k = 15

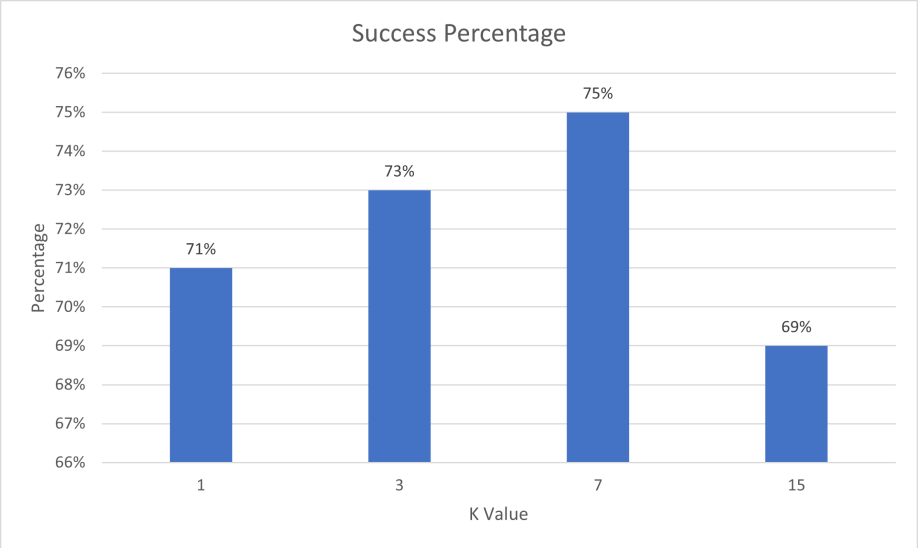
V testoch na k = 15 sa body kvalifikovali tak, že na konci testu jedna alebo dve farby z karty zmizli a jedna z dvoch alebo troch zostávajúcich farieb využila výhodu. Na obrázku nižšie môžete vidieť, že z bodkovej mapy nezmizla ani jedna farba, ale je tam oveľa menej fialových bodiek ako bodiek iných farieb, hoci tvary, ktoré sa získajú kvalifikovaním bodiek do rôznych farieb, stále vyzerajú ako obdĺžniky a nepremiešavaju sa ako tekutiny.

Posledný test skončil za 7,3 sekundy, čo je ešte väčší časový rozdiel medzi predchádzajúcimi dvoma. Správnosť kvalifikácie 69%



Obr. 6: k-NN algoritmus pri k = 15

6.5 Diagram uspešnosti



Obr. 7: Diagram uspešnosti pre rôzne hodnoty k

7 Zaver

Po dostatočnom počte testov možno konštatovať, že pri realizácii môjho riešenia sú najlepšie hodnoty k 3 a 7, relatívne rýchly čas kvalifikácie a dobrý výsledok správnej kvalifikácie. Stojí za zmienku, že kvalifikácia s k = 1 trvala najmenej a v niektorých prípadoch bola jej úspešnosť vyššia ako pri k = 3, ale tam sa body ležiace blízko okraja ich zóny niekedy chybne kvalifikujú na inú farbu.

Zatiaľ čo kvalifikáciu s k = 15 možno označiť za najneefektívnejšiu v porovnaní s predchádzajúcimi tromi, zaberie viac času a má nižšie percento správnosti kvalifikácie.