2. Laboratorijska vaja razpoznavanje vzorcev

Mentor: as. dr. Klemen Grm, mag. inž. el.

Avtor: Blaž Ardaljon Mataln Smehov

Datum: 17.12.2021

Vaja 2: razvrščanje s prileganjem

Pri drugi vaji smo izvedli razvrščanje slik s prileganjem. Za izvajanje vaje smo že dobili predlogo kode, v kateri smo morali dopolniti določene funkcije.

Pri nalogi smo imeli vzorčne slike in slike iz učne množice s katerimi smo primerjali vzorec. Pri razvrščanju so nas zanimali elementi in oznake razredov.

Kot predlogo smo dobili 3 dopolnjene kode za rešitev prve laboratorijske vaje in novo kodo, ki smo jo morali dopolniti.

Prvi dve funkciji, ki smo ju morali dopolniti sta namenjeni za računanje kosinusne mere podobnosti in evklidske razdalje med dvema vektorjema.

```
def csm(v1, v2):
    """Kosinusna mera podobnosti med vektorjema v1 in v2."""
    kosinusna_mera_podobnosti = np.dot(v1,
v2)/(np.linalg.norm(v1)*np.linalg.norm(v2))
    return kosinusna_mera_podobnosti
    # TODO: spiši funkcijo

def euc(v1, v2):
    """Evklidska razdalja med vektorjema v1 in v2."""
    evklidska_razdalja = np.linalg.norm(v1 - v2)
    return evklidska_razdalja
     # TODO: spiši funkcijo
```

Predloga kode nam je namignila kateri dve enačbi moramo zapisati v funkciji.

Za računanje obeh enačb sem uporabil funkcijo iz knjižnice numpy.

```
np.linalg.norm()
```

– funkcija nam računa normo vektorjev. Norma je funkcija, ki vsakemu ne ničelnemu vektorju v vektorskem prostoru pripiše pozitivno dolžino. Kot argument sem funkciji podal vektorja s katerima smo računali. Enačbi za izračun Evklidske razdalje in kosinusne podobnosti v pythonu sem našel na spletu.

Naslednja funkcija, ki smo jo morali dopolniti je namenjena za razdelitev podatkovne baze značilk in oznak na N enakih delov.

```
    """N-kratno navzkrižno preverjanje natančnosti razpoznavalnika CSMKNN
    na podatkovni zbirki, podani z vektorji značilk X in vektorjem oznak y."""
    # Najprej naključno premešamo matriko X in vektor y:
    X_premesan, y_premesan = premesaj_vrstice(X, y)

    # podatkovno zbirko enakomerno razdelimo na N kosov:
    X_deli = []
    y_deli = []

    X_deli = np.array_split(X_premesan,N)
    y_deli = np.array_split(y_premesan,N)
    # for i in range(N):
    #     pass
        # TODO: enakomerno razdeli podatkovno bazo na N kosov,
        #       tako, da seznam X_deli vsebuje N matrik značilk oblike
        #       (_, 24), seznam y_deli pa vsebuje N vektorjev oznak
        #       oblike (_,). Vsak del mora vsebovati čim bolj enako
porazdelitev
        #       razredov vzorcev.
```

V predlogi kode je že bila zapisana funkcija za premešanje značilk in oznak z uporabo naključne permutacije. V zgornjem segmentu se kliče ta funkcija in vrednosti dveh premešanih polj se shranita v polju X_premesan in y_premesan.

```
X_premesan, y_premesan = premesaj_vrstice(X, y)
```

Ti dve polji smo torej morali razdeliti na N polj enakih velikosti in ta polja shraniti v seznama X_deli in y_deli. Razdelitev teh polj sem opravil s funkcijo iz knjižnice numpy np.array split:

```
    X_deli = np.array_split(X_premesan,N)
    y_deli = np.array_split(y_premesan,N)
```

dana funkcija nam razdeli podano polje na pod polja, število pod polj določimo z vpisom argumenta v funkcijo. Kot prvi argument funkciji sem podal ime polja, ki ga je potrebno razdeliti, drugi argument v funkciji je podatek o tem na koliko delov želimo razdeliti polje.

V isti funkciji smo morali nato še preveriti uspešnost razpoznavalnika na testni zbirki. Uspešnost merimo kot delež predvidenih oznak na testni zbirki, ki se ujemajo z dejanskimi.

```
# TODO: Izračunaj uspešnost razpoznavalnika na testni zbirki.
        # uspešnost merimo kot delež predvidenih oznak na testni zbirki, ki se
        # ujemajo z dejanskimi:
        uspesnost = 0.0

        y_test1 = np.array(y_test)
        y_hat1 = np.array(y_hat)
        enakosti = y_test1 - y_hat1
        stevilo_enakosti = len(np.where(enakosti==0)[0])
        uspesnost = stevilo_enakosti/len(y_test)
        uspesnosti.append(uspesnost)
```

Preveriti smo torej morali koliko je enakih istoležnih elementov v seznamih y_test1 in y_hat1. Koliko je enakih istoležnih elementov sem preštel tako, da sem med seboj odštel oba seznama in če je vrednost na tistem mestu enaka 0, pomeni da sta elementa bila enaka. Da sem lahko seznama med seboj odštel sem ju najprej moral spremeniti v numpy array, to sem naredil s funkcijo:

```
y_test1 = np.array(y_test)
```

funkcija np.array nam spremeni polje v numpy polje, kjer lahko potem odštevamo vrednosti, kot argument funkciji sem podal ime seznama, ki sem ga želel pretvoriti.

V naslednjem delu kode sem nato med seboj odštel ti dve polji in dobil novo polje, kjer so zapisane razlike istoležnih elementov.

Število ničel sem poiskal z:

```
stevilo_enakosti = len(np.where(enakosti==0)[0])
```

funkcija np.where nam vrne seznam, v katerem so zapisane vse vrednosti, ki smo jih specificirali kot pogoj. Pogoj v tem primeru je ta da imajo elementi vrednost enako 0. S funkcijo len dobimo dolžino tega seznama, dolžina tega seznama pa dejansko ustreza številu enakih istoležnih elementov v obeh preverjenih seznamih. Delež sem nato izračunal kot kvocient med številom enakih elementov in številom vseh elementov.

Dopolniti smo morali tudi funkcijo, ki nam shrani značilke.

```
def shrani_znacilke():
    dn = "slike/"
    fns = sorted(os.listdir(dn))

    slike = []
    oznake = []

    for fn in fns:
        oznake.append(int(fn[3])-1)

        objekt = cv2.imread("slike/"+fn)
        slike.append(objekt)
        # TODO: Iz datotečnega imena preberi oznako objekta na sliki.
        #       Oznako, ki naj bo 0-4, shrani v seznam oznake.
        # TODO: preberi sliko in jo shrani v seznam slike.
```

V seznama oznake in slike sem s funkcijo append dodal specificirane elemente. Funkcija append nam doda element na konec seznama.

V predlogi kode je bil že podan algoritem za preverjanje uspešnosti razpoznavanja za vse kombinacije števil značilk in števil najbližjih sosedov z evklidsko razdaljo. Dodati smo morali enak algoritem, vendar za preverjanje uspešnosti razpoznavanja s kosinusnimi podobnostmi.

```
rezultati_kosinusna = []
    for N_znacilk in range(1, 25):
        if N_znacilk == 24:
            N_znacilk = "all"
        izb = SelectKBest(k=N_znacilk)
        rezultati_za_ta_N_znacilk = []
        for k in [1, 3, 5, 7]:
            # print(N_znacilk, k)
            knn = CSMKNN(k, mera=csm, nacin="max")
            uspesnosti = navzkrizno_preverjanje(X, y, 5, izb, knn)
            rezultati_za_ta_N_znacilk.append(np.mean(uspesnosti))
        rezultati_kosinusna.append(rezultati_za_ta_N_znacilk)

    # TODO: Preverite  še uspešnost razpoznavanja pri vseh kombinacijah z mero
    #       kosinusne podobnosti. Rezultate podajte v tabelah.
```

Algoritem za izračun uspešnosti kosinusne podobnosti je skoraj enak kot ta za evklidske razdalje, vse kar sem spremenil je izbira mere in način, v tem primeru je mera csm iz razreda CSMKNN, način je max. Ko kličemo max nas zanima kateri razred ima največje število podobnosti s testiranim vzorcem.

Končni rezultat je tabela vrednosti kosinusne mere podobnosti in evklidskih razdalj.

Koda za generiranje Excel tabele:

```
    data1 = pd.DataFrame(data = rezultati, index=n_znac, columns=['1',
'3','5','7'])
    data1.to_excel('euc.xlsx',index_label='znacilke/K',sheet_name='euc')


    data2=pd.DataFrame(data = rezultati_kosinusna, index=n_znac, columns=['1',
'3','5','7'])
    data2.to_excel('cos.xlsx',index_label='znacilke/K',sheet_name='cos')
```

Za izris excel tabel sem uporabil panda knjižnico.

Pd.DataFrame je funkcija s te knjižnice. Kot argumente sem funkciji podal katere rezultate naj izpiše, indeks po katerem vpisuje in kako so označeni stolpci.

Data.to_excel – s to funkcijo sem definiral kako se imenuje razpredelnica in kako se imenuje datoteka.

Rezultati:

1. **Kosinusna mera podobnosti:**

znacilke/K	1	3	5	7
1	0,227778	0,227778	0,222222	0,216667
2	0,397222	0,452778	0,425	0,491667
3	0,558333	0,522222	0,494444	0,488889
4	0,577778	0,591667	0,6	0,555556
5	0,625	0,602778	0,580556	0,555556
6	0,641667	0,633333	0,655556	0,597222
7	0,625	0,630556	0,65	0,572222
8	0,713889	0,663889	0,588889	0,602778
9	0,694444	0,630556	0,636111	0,588889
10	0,661111	0,636111	0,6	0,627778
11	0,680556	0,630556	0,658333	0,652778
12	0,725	0,675	0,616667	0,594444
13	0,683333	0,658333	0,613889	0,588889
14	0,688889	0,622222	0,594444	0,575
15	0,669444	0,641667	0,647222	0,588889
16	0,677778	0,602778	0,6	0,619444
17	0,694444	0,652778	0,594444	0,605556
18	0,652778	0,669444	0,636111	0,619444
19	0,691667	0,638889	0,622222	0,552778
20	0,669444	0,652778	0,6	0,605556
21	0,680556	0,638889	0,588889	0,580556
22	0,686111	0,652778	0,619444	0,586111
23	0,683333	0,616667	0,608333	0,583333
24	0,691667	0,627778	0,611111	0,605556

2. **Evklidske razdalje:**

znacilke/K	1	3	5	7
1	0,483333	0,536111	0,566667	0,547222
2	0,575	0,530556	0,544444	0,547222
3	0,675	0,65	0,622222	0,6
4	0,736111	0,688889	0,686111	0,7
5	0,661111	0,691667	0,672222	0,694444
6	0,666667	0,694444	0,627778	0,633333
7	0,663889	0,702778	0,688889	0,641667
8	0,705556	0,669444	0,641667	0,641667
9	0,733333	0,686111	0,658333	0,633333
10	0,733333	0,694444	0,686111	0,661111
11	0,702778	0,725	0,680556	0,663889
12	0,733333	0,727778	0,727778	0,666667
13	0,705556	0,702778	0,705556	0,680556
14	0,741667	0,713889	0,716667	0,708333
15	0,75	0,727778	0,705556	0,691667
16	0,722222	0,713889	0,713889	0,702778
17	0,758333	0,7	0,730556	0,680556
18	0,736111	0,719444	0,725	0,697222
19	0,772222	0,752778	0,711111	0,702778
20	0,755556	0,733333	0,730556	0,7
21	0,772222	0,744444	0,75	0,716667
22	0,766667	0,755556	0,75	0,722222
23	0,775	0,744444	0,727778	0,713889
24	0,797222	0,758333	0,741667	0,727778