Nadzorovano modeliranje

Amon Stopinšek (63150273)

10. maj 2017

1 Uvod

V nalogi smo uporabili enostavne metode za napovedovanje. Glavni cilj naloge je bila napoved ocen filmov posameznega uporabnika na podlagi ostalih uporabnikov. Problem smo najprej rešili s pomočjo regresije, kjer smo napovedovali dejanske vrednosti. Nato smo se problema lotili še s klasifikacijo, kjer smo ocene filmov razdelili v dva razreda in za danega uporabnika napovedali v kateri razred bi uvrstil posamezni film.

2 Podatki

Pri nalogi sem uporabili podatkovno zbirko Movie Lens.

3 Metode

3.1 Regresija

Pred gradnjo in vrednotenjem modelov je bilo potrebno podatke spraviti v ustrezno obliko.

Podatke sem prebral in jih pretvoril v matriko filmov in uporabnikov.

```
dfRatings = pd.read_csv('../data/ratings.csv')

# transform df
df = dfRatings.pivot(index='movieId', columns='userId', values='rating')
```

Iz podatkov sem nato odstranil vse filme in uporabnike z manj kot 100 ocenami.

```
# get users with more than 100 ratings
dfU = df.dropna(axis=1, how='any', thresh=100, subset=None)

# drop movies with less than 100 ratings
df.dropna(axis=0, how='any', thresh=100, subset=None, inplace=True)

# drop users with less than 100 ratings
df.drop([col for col in df.columns if col not in dfU.columns], axis=1, inplace=True)
```

Pred začetkom granjem modelov je bilo potrebno poskrbeti še za neznane vrednosti. Le te sem zamenjal z oceno 0.

```
# replace null values with 0
df = df.fillna(0)
```

Za regresijo sem izbral tri modele Linear regression, Lasso in Bayesian Ridge.

Učenja modelov sem se lotil tako, da sem za vsakega uporabnika iz matrike trikrat naključno izbral primere za učno in testno množico brez primerov filmov, ki jih uporabnik ni ocenil, naučil model s podatki iz učne množice in ga ovrednotil s pomočjo testne množice. Uporabnika za katerega sem gradil model, sem uporabil le kot ciljno spremenljivko, zato sem njegove ocene iz učne in testne množice odstranil.

```
for user in df.columns:
    for x in range(3):
        uDf = df[df.loc[:, user] > 0]

        # split df into training and test sets
        trainDf = uDf.sample(frac=.75)
        testDf = uDf[~uDf.index.isin(trainDf.index)]

        # get ratings of current user
        trainUserDf = trainDf.loc[:, user].copy()
        testUserDf = testDf.loc[:, user].copy()

        # drop current user from training and test sets
        trainDf = trainDf.drop(user, 1)
        testDf = testDf.drop(user, 1)

        ### learn and evaluate models ###
```

Za ovrednotenje modelov sem izbral mero Mean Absolute Error, ker se mi je ta mera zdela najlažja za razumevanje kako dobro oz. slabo napoveduje model. Dobljena ocena MAE pove, za koliko se v povprečju zmoti model pri napovedovanju novih vrednosti.

```
def learnPredictScore(model, trainX, trainY, testX, testY):
    model.fit(trainX, trainY)

    # predict ratings for current user
    prediction = model.predict(testX)

    # MAE
    return mean_absolute_error(prediction, testY)
```

3.2 Klasifikacija

Priprava podatkov za klasifikacijo je potekala na enak način kot pri regresiji. Vrednosti v ciljni spremenljivki uporabnika sem pretvoril v dva razreda, 0 če je bila ocena manjša ali enaka 3, sicer 1.

```
# binarize current user data
trainUserDf = trainUserDf.apply(lambda x: 1 if x > 3 else 0)
testUserDf = testUserDf.apply(lambda x: 1 if x > 3 else 0)
```

Za klasifikacijo sem izbral metodo podpornih vektorjev. Metoda vsak primer iz učne množice predstavi kot vektor pri katerem je znan ciljni razred, komponente vektorja pa predstavljajo ostali atributi. Metoda v danem prostoru poišče hiperravnino ki najbolje loči prostor glede na ciljni razred. Glavna parametra, ki ju lahko nastavimo sta koeficient pri ceni ob napaki in tip funkcije za ločitev prostora (linearna, polinomska...).

Za vsakega uporabnika iz matrike sem trikrat naključno izbral primerke za učno in testno množico, naučil model ter ga ovrednotil s pomočjo klasifikacijske točnosti in ocene F1.

3.3 Bonus

V matriko filmov in uporabnikov sem dodal novega uporabnika z mojimi ocenami filmov. Ocenil sem 16 filmov, ostali filmi v matriki so bili tako kot pri ostalih uporabnikih ocenjeni z oceno 0.

```
myRatings = [4, 5, 0, 0, 0, 0, 0, 0 ,0, 4, 0, 0, 0, 0,
             3, 0, 0, 0, 0, 1, 0, 5, 0, 0, 0, 0, 0, 0,
             4, 0, 5, 0, 0, 4, 4, 2, 0, 4, 5, 5, 5, 0, 5]
myRLen = len(myRatings)

for x in range(len(df)-myRLen):
    myRatings.append(0)

user = 9999

df2 = pd.DataFrame({user:myRatings}, index = df.index)

df = pd.concat([df,df2], axis=1)
```

Za novega uporabnika sem zgradil regresijski model s pomočjo metode Lasso in napovedal ocene za preostale filme.

4 Rezultati

4.1 Regresija

V tabeli 1 so prikazani rezultati vseh treh modelov. Vsi zgrajeni modeli so se izkazali za skoraj enakovredne.

Tabela 1: Rezultati modelov za regresijo

model	mean absolute error
Linear regression	0.6613
Lasso	0.6552
BayesianRidge	0.6611

4.2 Klasifikacija

Z modelom SVM sem uspel napoved v primerjavi z večinskim klasifikatorjem izboljšati za približno 20%.

Klasifikacijski problem je lažji od regresijskega, saj je pri klasifikaciji nabor vrednosti precej manjši kot pri regresiji, kjer napovedujemo vrednost v zveznem prostoru.

Tabela 2: Rezultati modela SVM za klasifikacijo

mera	rezultat
večinski klasifikator	0.5467
klasifikacijska točnost	0.7415
F1	0.7722

4.3 Bonus

Ocenil sem 16 od 151 filmov, kar se je izkazalo za premalo, da bi lahko model vrnil uporabne napovedi. Večina ocen se je gibala med 1 in 2. Algoritem je pravilno višje ocenil animirane filme, ni pa uspel razpoznati povezave med nadaljevanji filmov. Napoved se pri majhnem številu ocenjenih filmov ni izkazala za uporabno.

5 Izjava o izdelavi domače naloge

Domačo nalogo in pripadajoče programe sem izdelal sam.

Priloge

A Podrobni rezultati poskusov

B Programska koda