Sustavi za davanje preporuka

Krnetić, Luka

SVEUČILIŠTE U DUBROVNIKU
ODJEL ZA ELEKTROTEHNIKU I RAČUNARSTVO

LUKA KRNETIĆ

SUSTAVI ZA DAVANJE PREPORUKA

ZAVRŠNI RAD

Dubrovnik, rujan, 2019.

SVEUČILIŠTE U DUBROVNIKU
ODJEL ZA ELEKTROTEHNIKU I RAČUNARSTVO

SUSTAVI ZA DAVANJE PREPORUKA

ZAVRŠNI RAD

Studij: Primijenjeno/poslovno računarstvo
Kolegij: Strojno učenje
Mentor: izv.prof.dr.sc Mario Miličević
Student: Luka Krnetić

Dubrovnik, rujan, 2019

SAŽETAK

U početku rada se objašnjavaju određeni razlozi koji čine sustav za davanje preporuka nužnim u današnjem svijetu, kao i područje strojnog učenja, pošto se metode strojnog učenja koriste u svrhu rješavanja problema s kojima se sustavi za davanje preporuka susreću. Prvo se objašnjavaju sustavi za davanje preporuka koji preporuku temelje prema sadržaju, dok se u daljnjem dijelu rada objašnjavaju sustavi koji pružaju preporuke na temelju kolaboracije, to jest, zajedničkih akcija korisnika koji svojom aktivnošću poboljšavaju rad sustava. Sustavi koji davaju preporuke, a temeljeni su na kolaboraciji, bilježe sve veću i veću primjenu u svijetu pa se povodom toga i detaljnije objašnjavaju. Da bismo objasnili sustave za davanje preporuka koje nastaju kao rezultat kolaboracije korisnika, objašnjavamo algoritme linearne regresije i gradijentnog spusta, kao i neke probleme s kojima se navedeni algoritmi susreću. Algoritmi se praktično objašnjavaju kroz sustav za preporučivanje filmova. Objašnjava se kako možemo doći do top-n preporuka za određenog korisnika, kako pronaći slične filmove kao i kako riješiti problem novog korisnika. Poglavlje koje slijedi objašnjava određene metode koje služe kao evaluacija sustava. Na kraju rada je prikazan programski kod koji se koristi za implementaciju sustava objašnjenog u trećem poglavlju kao i evaluacija implementiranog sustava.

Ključne riječi: sustavi za davanje preporuka, strojno učenje, linearna regresija s gradijentom spusta

ABSTRACT

In the beginning of the work we explain some of the reasons why are recommender systems necessary in the world we live today, as well, we explain field of machine learning since methods that stems from the field of machine learning are used by recommender systems in order to resolve problems that are encounter. First we explain recommender systems whose recommend is based on content, whilst as we continue with work we are more into explaining collaborative filtering recommender system whose recommend is based on collaboration, that is, common actions among users which makes recommender system better. As collaborative filtering recommender systems are used more and more in real world, we explain them in more detail. As we have to explain system recommender that are based on collaborative filtering, we have to explain algorithms such as linear regression and gradient descent, as well we explain some of the problems that algorithms encounter. Practically, algorithms are explained through recommender system that recommend movies. We explain how we can get top-n recommends for particular user, how to find similar movies and how to solve problem of new user. In the chapter that follows, we explain evaluation methods for recommender systems. In the end of the work, we explain program code that is used in order to implement recommender system that is explained in third heading, as well, we explain evaluation of the implemented system.

Keywords: system recommender, machine learning, linear regression with gradient descent

SADRŽAJ

1. Uvod

Sustavi za davanje preporuka (eng. „Recommender systems“) su sustavi koji pomoću korištenja određenih metoda i rada s određenim podacima pružaju preporuke korisniku sustava. Obično, rezultati korištenja sustava za davanje preporuka su poboljšano korisničko iskustvo i veći profit. U nastavku ovog rada, radi lakšeg izražavanja i shvaćanja, koristit ću riječ objekt za vrstu sadržaja koju sustav za davanje preporuka nudi svom korisniku. Na primjer, objekt sustava za davanje preporuka koje se tiču filmova je film, dok je objekt sustava za davanje preporuka koje se tiču smještaja smještaj.

Dva su glavna razloga za stvaranje sustava za davanje preporuka. Prvi je sve veća i veća količina informacija, a sve manje i manje vremena da korisnik obradi sve informacije kako bi došao do relevantnog objekta koji odgovara njegovim interesima. Korisnik sustava ako je zainteresiran za određeni proizvod kao na primjer film, vijest, ili kompjuter treba uložiti dosta vremena kako bi došao do što više informacija u vezi proizvoda koji ga zanima. Drugi razlog je povećani broj mogućnosti praćenja određenih aktivnosti kao što su ocjenjivanje proizvoda, praćenje korisnika u smislu da se gleda koje proizvode je korisnik pregledao ili kupio.

U današnje vrijeme, zbog napretka tehnologije, svjedoci smo sve veće zastupljenosti strojnog učenja u raznim područjima, pa tako i u sustavima za davanje preporuka. Strojno učenje, prema Tom Michael Mitchellu (američkom računalnom znanstveniku) je: “Mogućnost kompjuterskog programa da poboljša zadatak T u odnosu na metriku performanse P, temeljeno na iskustvu E“. Kod sustava za davanje preporuka, očekuje se poboljšanje davanja preporuke (T), u odnosu na metriku performanse (obično je metrika performanse kupnja proizvoda, gledanje video sadržaja, klik na određenu vijesti...), temeljeno na podacima o korisnicima (E). Rad strojnog učenja se bazira na generiranju hipoteze, automatizaciji otkrivanja i korištenja pravilnosti u velikim skupovima podataka te istraživanja što se pod određenim uvjetima može naučiti kako bi se dalje na temelju rezultata učenja mogao stvoriti model [1]. Model u strojnom učenju predstavlja matematički izraz koji rješava određeni problem. Podaci za učenje modela su danas dostupni na internetu, često besplatno, dok su kriteriji nad kojim se model razvija definirani ciljnom funkcijom ili konceptom pomoću koje se određuje ispravnost hipoteze. Ciljna funkcija nije jedinstvena, već ovisi o vrsti problema koju algoritam strojnog učenja rješava. Ako uzmemo za primjer da neki web shop posjeduje sustav za preporučivanje i da je rad sustava temeljen na algoritmima strojnog učenja, onda bi ciljna funkcija tog sustava bila preporučiti proizvod koji je relevantan za korisnika.

Neki od najpoznatijih primjera sustava za preporučivanje prema područjima njihovog djelovanja su:

- IMDb.com, MovieLens.org, Netflix.com – filmovi
- Spotify.com, Napster.com, Pandora.com – glazba
- eBay.com, Amazon.com, AliExpress.com – e-trgovina
- Facebook.com, LinkedIn.com - društvene mreže
- YouTube.com, DailyMotion.com, Vimeo.com – video sadržaj
- google.com, Baidu.com – web tražilice
- Airbnb.com, booking.com, trivago.com – booking sustavi za smještaj

[2]

Slika 1. prikazuje problem dugog repa. Problem „dugog repa“ je fenomen koji sustave za davanje preporuka čini nužnim u današnjem svijetu [3]. U današnje vrijeme, zbog manjka ponude i viška neisplativih cijena za određene proizvode, čovjek je sve više i više zainteresiran za potražnju određenih proizvoda online. Prednost online sustava za kupnju određenih proizvoda je ta da navedeni sustavi obično nemaju problem sa smještajem velikog broja proizvoda kao što to imaju poduzeća koja pružaju uslugu kupnje proizvoda samo onih proizvoda koji se nalaze unutar prostora unutar kojeg poduzeće posluje. Online sustavi nemaju problem sa smještajem proizvoda, ali imaju problem s davanjem preporuka. Prema slici je vidljivo da ~ 80% korisnika favorizira ~ 20% proizvoda. Često je slučaj da određenom korisniku nije relevantan proizvod koji je popularan, već proizvod koji odgovara njegovim užim interesima. Jedan od načina kako korisniku preporučiti proizvod koji nije popularan, a koji je samom tom korisniku relevantan jest da koristimo sustav za preporučivanje baziran na algoritmima strojnog učenja.

Slika 1. Problem dugog repa

Podatke za rad sustava može prikupljati:

- Direktno od samog korisnika na temelju određenog zahtjeva. Ako je korisniku dana mogućnost da ocijeni neki objekt, često se događa da korisnik nije zainteresiran za ocjenjivanje, ili ako jest, onda je uglavnom sklon dati ocjenu za one filmove koji su ga pretjerano, bilo negativno, bilo pozitivno, iznenadili. Zbog toga se izbjegava učiti model s eksplicitno, to jest, direktno prikupljenim podacima od strane samog korisnika. [4]
- Implicitno - na temelju ponašanja korisnika. Ako korisnik klikne na određeni YouTube video ili kupi proizvod na Amazon web shopu, onda možemo zaključiti da korisnik preferira određeni video koji je kliknuo, ili određeni proizvod koji je kupio. Na temelju tih informacija možemo dalje preporučiti video sličan onom na kojeg je korisnik kliknuo, ili proizvod sličan onom proizvodu kojeg je korisnik kupio.

Sustave za preporučivanje možemo podijeliti na:

- Personalizirane
- Nepersonalizirane

Nepersonalizirani sustavi su sustavi koji davaju preporuke koje su jednake za svakog korisnika te su jednostavni za implementaciju. Neki od primjera kriterija pomoću kojih nepersonalizirani sustavi preporučuju određeni objekt:

- najnoviji objekt
- najprodavaniji objekt
- najbolje ocijenjen objekt
- najpopularniji objekt

Jedna od glavnih prednosti nepersonaliziranih sustava su jednostavna implementacija sustava kao i lako prikupljanje podataka. Neki od glavnih nedostataka nepersonaliziranih sustava su: proizvod koji je najprodavaniji ili najbolje ocijenjen često nije relevantan za određenog korisnika, proizvod koji je novi na tržištu, a može zanimati korisnika, neće biti ponuđen korisniku. U nastavku ovog rada fokusirat ću se na personalizirane sustave. Personalizirani sustavi su sustavi koji analiziraju korisnika te za davanje preporuke za određeni objekt uzimaju u obzir korisničke aktivnosti, sličnosti s ostalim korisnicima te sličnost određenog objekta s ostalim objektima. Kao takvi, personalizirani sustavi ne daju jednake preporuke za svakog korisnika. Personalizirani sustavi prema načinu davanja preporuka mogu se podijeliti na:

- Sustave za davanje preporuka koji su temeljeni na sadržaju (eng. Content Based Recommender) – fokusiraju se na davanje preporuka za određeni objekt koji je sadržajno sličan ostalim objektima. Na primjer dva filma istog žanra ili dvije nekretnine istih specifikacija.
- Sustave za davanje preporuka baziranih na suradnji to jest kolaboraciji (eng. Collaborative Filtering Recommender System)
- Hibridni sustave – sustavi koji se temelje na sadržaju i kolaboraciji

Prva i najjednostavnija metoda koja se koristila kao temelj rada personaliziranih sustava je metoda najbližih susjeda (eng. k-nearest neighbour – kNN) [2]. Metoda najbližih susjeda preporuku za određeni objekt temelji na sličnosti određenog korisnika s ostalim korisnicima, ili sličnosti određenog proizvoda s ostalima proizvodima koje je korisnik kupio, ocijenio ili na neki drugi način pružio informaciju sustavu da određeni objekt smatra relevantnim. Uzimaju se top n preporuka objekata koji imaju najveću sličnost s objektima koje korisnik favorizira, to jest smatra relevantnim. Pošto korisnik hoće dobiti preporuke u stvarnom vremenu, a metoda najbližih susjeda za pronalaženje sličnosti uzima u obzir sve korisnike što je kompleksno i zahtijeva dosta vremena, metoda najbližih susjeda se ipak ne koristi u praksi. Da bi se smanjila složenost računanja sličnosti određenog korisnika s ostalim korisnicima sustava, računa se sličnost između objekata. Računati sličnosti između objekata je stabilnija metoda od računanja sličnosti između korisnika jer se preferencije korisnika mijenjaju, dok se osobine objekata ne mijenjaju. Pošto se navedene dvije metode primjenjuju na podatke bez procesiranja, onda navedene metode smatramo metodama koje pripadaju grupi sustava temeljenih na memoriji (eng. memory-based). Procesiranje podrazumijeva pretvaranja sirovih podataka u određeni oblik koji je prikladan za računanje.
Sustavi temeljeni na modelu (eng. Model-based) izrađuju model te kasnije na temelju modela, a ne na temelju svih podataka kao što to rade sustavi bazirani na memoriji, davaju preporuke za određene objekte. Modeli koji se koriste su izrađeni pomoću algoritama strojnog učenja (eng. Machine learning) i rudarenja podacima (eng. Data mining). Jedna od prednosti personaliziranih sustava je neprisutnost problema dugog repa jer je princip rada sustava takav da je sustav u stanju pronaći proizvod koji je relevantan za korisnika, a koji nije popularan. Također, personalizirani sustavi nemaju problema s preporučivanjem proizvoda koji su novi na tržištu - korisnik koji ima određene sličnosti s ostalim korisnicima koji na primjer vole isti žanr filma i koji su dali visoku ocjenu za novi film može dobiti preporuku za novi film koji je favoriziran od strane ostalih korisnika koji su slični tom korisniku kojem je preporuka namijenjena. Neki od nedostataka personaliziranih sustava su: nemogućnost rada sustava u slučaju premalo ocjena na početku rada (eng. Cold Start), problem novih korisnika ili objekata - sustav nema informacija o novom korisniku pa se problem često rješava tako da sustav u početku novom korisniku preporuči najpopularniji objekt, nedostatak sličnosti između pojedinih objekata ili korisnika (eng. Black Sheep), problem rijetkog uzorkovanja (eng. Sparsity), problem raznolikosti ili pretendiranja (eng. Diversity) – određeni korisnici su kritičniji ili manje kritični od drugih u smislu ocjenjivanja određenog objekta, problem privatnosti – određeni korisnici radi straha o iskorištavanju podataka u negativne svrhe izbjegavaju pružiti podatke o određenom objektu.

2. Sustavi temeljeni na sadržaju

Glavna ideja sustava za davanje preporuka čiji je rad temeljen na davanju preporuka koje ovise o sadržaju jest preporučiti objekt korisniku X koji je sličan objektu kojeg je korisnik X prethodno ocijenio s visokom ocjenom. Jedan od načina kako da dođemo do objekta koji je sličan objektu X jest da koristimo određene metode za računanje sličnosti. Po pitanju filmova možemo tražiti film sa sličnim glumcima, žanrom ili redateljem, po pitanju vijesti možemo tražiti vijesti sa sličnim ključnim riječima, a po pitanju ljudi možemo preporučiti ljude sa zajedničkim prijateljima, ili zajedničkim osobinama. Na primjer ako je određeni korisnik pogledao filmove kao što su Harry Potter 1 i Harry Potter 2 te je za oba filma dao visoku ocjenu, sustav čije je davanje preporuka bazirano na sadržaju filma bi vjerojatno dao preporuku da korisnik pogleda filmove Harry Potter 3 i Harry Potter 4.

Svaki korisnik i objekt koji postoje u sustavu imaju svoj profil. Profil korisnika i objekta je opisan određenim vrijednostima koje pripadaju određenim atributima. Na primjer, sustav za preporučivanje filmova izrađuje profil filma i korisnika. Profil filma je opisan s određenim vrijednostima koje pripadaju određenim atributima kao što su autor, naslov, glumci, žanr, dok je profil korisnika opisan s određenim vrijednostima kao što su ocjena određenom filmu, broj godina, nacionalnost ili bilo što drugo što se smatra da je relevantno za dati dobru preporuku. Za razumijevanje primjera koji slijedi i koji prikazuje način rada sustava čija se preporuka temelji na sadržaju definiramo sljedeće pojmove:

- Frekvencija pojma (eng. Term Frequency) – broj ponavljanja riječi u dokumentu
- Inverzna frekvencija dokumenta (eng. Inverse Of The Document Frequency) – inverz pojavljivanja dokumenta u cijelom korpusu dokumenata

Ako je broj pojavljivanja određene riječi u dokumentu veći od 0, onda da bi se izračunala težina te riječi, odnosno pojma, koristi se formula:

$$w_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{tf}_{\mathrm{t,d}} & , \text{ako je } \mathrm{tf}_{\mathrm{t,d}} > 0 \\ 0 & , \text{ inače} \end{cases} \quad (1)$$

Formulom 1 se doznaje važnost određene riječi ili pojma koji se nalaze unutar dokumenta. Pošto se pojavljivanje određenih riječi kao što su riječi: „da, ili, ali, kao“ ponavlja dosta puta, a same po sebi navedene riječi nemaju neku posebnu važnost, koristi se logaritam da bi se smanjila važnost navedenih riječi. Nakon računanja TF-IDF, da bismo odredili kojem korisniku je određeni pojam sličan, koristimo Vektorski prostorni model (eng. Vector Space Model) koji računa kosinus razlike između dva kuta vektora, vektora profila korisnika i vektora pojma. Vektori su dužinski normalizirani nakon čega je njihova duljina jednaka 1, dok je kalkulacija kosinusa unutarnji produkt sume dvaju vektora [4].

Da bismo razumjeli navedeno, pretpostavit ćemo da smo u google tražilicu upisali „IoT and analytics“ i prvih šest članaka koji se pojavljuju imaju određenu frekvenciju riječi koja je prikazana u sljedećoj tablici:

Articles	Analytics	Data	Cloud	Smart	Insight
Article 1	21	24	***0***	2	2
Article 2	24	59	2	1	***0***
Article 3	40	115	8	10	19
Article 4	4	28	5	***0***	1
Article 5	8	48	4	3	4
Article 6	17	49	8	***0***	5
DF	5,000	50,000	10,000	5,00,000	7000

Tablica 1 – broj pojavljivanja određenog pojma po određenom članku [5]

U korpusu svih dokumenata koji su se koristili pri pretrazi, 5000 članaka sadrži riječ „analytics“, 50 000 članaka sadrži riječ „data“, 10 000 riječ „Cloud“, 500 000 riječ „Smart“ dok se riječ „Insight“ pojavljuje u 7000 članaka. Pretpostavljamo da je ukupni korpus svih dokumenata 1 000 000. Sljedeća tablica prikazuje rezultate težina određenih pojmova. Na primjer, težina pojma „Analytics“ u prvom članku je $1+ \log_{10} 21 = 2.322$.

Articles	Analytics	Data	Cloud	Smart	Insight	Length of Vector
Article 1	2.322219295	2.380211242	0	1.301029996	1.301029996	**3.800456039**
Article 2	2.380211242	2.770852012	1.301029996	1	0	**4.004460697**
Article 3	2.602059991	3.06069784	1.903089987	2	2.278753601	**5.380804488**
Article 4	1.602059991	2.447158031	1.698970004	0	1	**3.527276247**
Article 5	1.903089987	2.681241237	1.602059991	1.477121255	1.602059991	**4.257450611**
Article 6	2.230448921	2.69019608	1.903089987	0	1.698970004	**4.326697114**

Tablica 2 - težina pojma po određenom članku [5]

Nakon što smo izračunali težine određenih pojmova možemo se baviti izračunavanjem inverzne frekvencije pojma (IDF). IDF se računa prema logaritamskom inverzu frekvencije dokumenta u cijelom korpusu dokumenata. Dakle, ako imamo 1 000 000 dokumenata u kojima se riječ „smart“ pojavljuje u 5 000 000 puta, onda je IDF rezultat riječi „smart“ $\text{Log}_{10}(1000000/500000) = 0.30$. U tablici 3 prikazan je inverz frekvencije za svaki pojam.

DF	5000	50000	10000	500000	7000
IDF	2.301029996	1.301029996	2	0.301029996	2.15490196
N =	**10^6**				

Tablica 3 - prikaz inverzne frekvencije pojma [5]

Duljina vektora svakog članka se računa korijenom sume kvadrata IDF-a svake riječi. Da bismo dobili vektore veličine jedan, podijelimo red pojma s duljinom vektora te dobijemo sljedeće rezultate:

Articles	Analytics	Data	Cloud	Smart	Insight	Sum of Normalized Lengths
Article 1	0.61103701	0.626296217	0	0.342335231	0.342335231	1
Article 2	0.594389962	0.691941368	0.324895184	0.249721517	0	1
Article 3	0.483581962	0.568817887	0.353681311	0.371691632	0.423496822	1
Article 4	0.454191812	0.693781224	0.481666273	0	0.283504872	1
Article 5	0.447002246	0.62977624	0.376295614	0.34694971	0.376295614	1
Article 6	0.51550845	0.621766675	0.439848211	0	0.392671352	1

Tablica 4 - normalizirana tablica [5]

Sada možemo izračunati kosinusovu sličnost kako bismo doznali sličnosti između članaka. Navedeni račun ćemo izvršiti za tri članka i tri pojma. Dobiveni rezultati su prikazani u sljedećoj tablici

Articles	Analytics	Data	Cloud
Article 1	0.61103701	0.626296217	0
Article 2	0.594389962	0.691941368	0.324895184
Article 3	0.483581962	0.568817887	0.353681311

Article Vector	Cosine Values
cos(A1,A2)	0.796554526
cos(A2,A3)	0.795934246
cos(A1,A3)	0.651734967

Tablica 5 - sličnosti između članaka

Prema navedenom rezultatu možemo zaključiti da su članak 1 i 2 sličniji nego članak 1 i 3. Ako pretpostavimo da je korisnik dao dobru ocjenu za članak 1, onda možemo iskoristiti dobivene informacije kako bismo korisniku mogli preporučiti članak 3.

Prednosti sustava temeljenih na sadržaju

- Ne trebaju podaci o drugim korisnicima da bismo nekome dali preporuku
- Moguće je dati preporuku koja odgovara uskim interesima korisnika
- Moguće je preporučiti novi i ne popularni objekt
- Moguće je dobiti objašnjenje zašto je neki objekt preporučen

Nedostatci sustava temeljenih na sadržaju

- Nalaženje potrebnih značajki ili atributa je teško ako se radi o slikama, muzici ili filmovima
- Nemogućnost preporuke objekta koji ne odgovara određenim vrijednostima koje se nalaze unutar profila korisnika
- Problem hladnog početka (eng. Cold-start) – kako izgraditi korisnički profil bez korisničkih aktivnosti

3. Sustavi za temeljeni na kolaboraciji

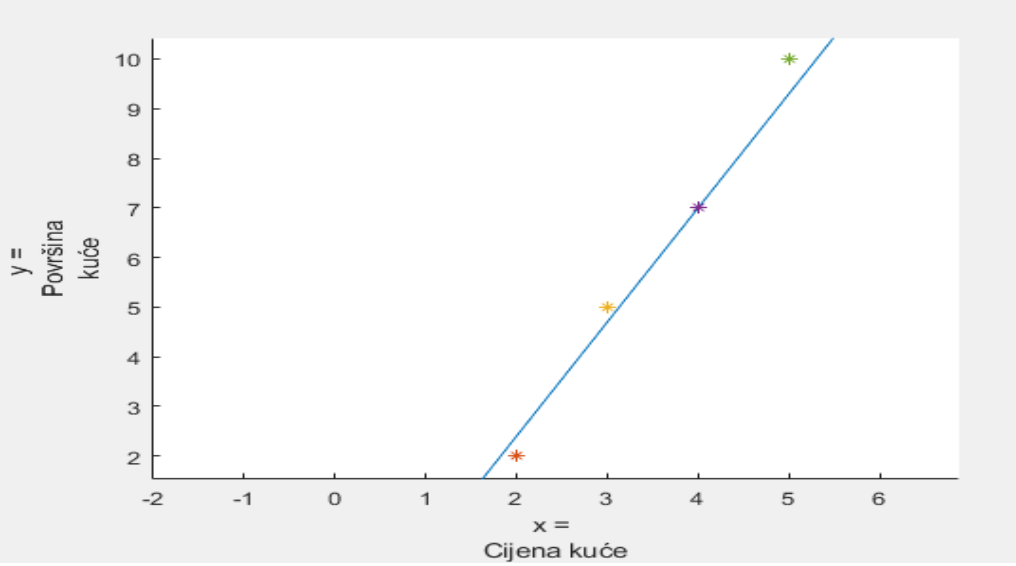
3.1 Linearna regresija i gradijent spusta

U ovom poglavlju ćemo prikazati sustav za preporučivanje filmova temeljen na suradnji (eng. Collaborative Filtering) te ćemo objasniti algoritam linearne regresije kao i algoritam gradijentnog spusta. Kao što je navedeno na početku, model u strojnom učenju predstavlja određeni matematički izraz koji rješava problem iz stvarnog svijeta - u slučaju ovog rada i ovog poglavlja, model rješava problem davanja preporuke filma. Da bismo mogli razvijati model potrebni su nam određeni podaci i kriteriji kako bismo znali da model uistinu rješava ciljni problem. Kriterij kojim se provjerava ispravnost rada modela počiva na ciljnoj funkciji, to jest, ciljna funkcija jest kriterij ispravnosti rada modela. Ciljna funkcija ovisi o vrsti algoritma. U sljedećim primjera, koristit ćemo algoritam linearne regresije i algoritam gradijentnog spusta kako bismo došli do odgovarajuće ciljne funkcije. Linearna regresija pripada skupini algoritama strojnog učenja pod nazivom „Nadgledano učenje“. Tipično je za algoritme nadgledanog učenja da postoji skup podataka koji predstavlja vrijednosti koje bi algoritam trebao aproksimirati. Ako algoritam dobro aproksimira odgovarajuće vrijednosti iz skupa podataka, onda se smatra da algoritam dobro funkcionira, to jest, za neku novonastalu situaciju će se predvidjeti dobro rješenje – u slučaju preporuke filmova, za određenog korisnika koji je pogledao ili ocijenio određeni broj filmova cilj je preporučiti film koji odgovara interesima korisnika. Osim preporuke filmova možemo proučavati određene veze između starosne dobi i krvnog tlaka osobe, odnos prodaje koja nastaje kako se povećanje površine poduzeća povećava ili jakost veze između prodajne i procijenjene vrijednosti nekretnine. Cilj linearne regresije je otkriti odnos između više varijabli, taj odnos izraziti funkcijom čija bi vrijednost izlazne varijable mogla poslužiti kao prognoza. Za sljedeće primjere oznaka $x^{(i)}$ će predstavljati ulaznu varijablu ili ulazni atribut (na primjer površina kuće), dok će oznaka $y^{(i)}$ predstavljati izlaznu vrijednost ili traženu vrijednost (na primjer cijenu). U sljedećim primjerima, skup m n-torki, to jest, $(x^{(i)}, y^{(i)}; i = 1,...,m)$ zovemo skup podataka za treniranje. Ovdje $^{(i)}$ ne znači eksponent, već i-ti podataka. Također, koristit ćemo X za sve vrijednosti ulaznih varijabli, dok ćemo Y koristiti za sve vrijednosti izlaznih varijabli. Da bismo opisali problem algoritama nadgledanog učenja formalnije, definiramo funkciju:

$$h : X \rightarrow Y \qquad (2)$$

Funkciju h nazivamo hipoteza i gledamo da h što bolje aproksimira odgovarajuću vrijednost y. U principu postoji tri skupine podataka: podaci za učenje, podaci za testiranje, dakle podaci za koje se zna klasifikacija te novi podaci s nepoznatom klasifikacijom.

Slika 2. prikazuje grafičku raspoređenost skupa parova $x^{(i)}$, $y^{(i)}$, a ujedno i pravac koji aproksimira vrijednosti cijene nekretnine na temelju vrijednosti cijene kuće. Prikazani pravac je regresijski pravac koji je izražen određenom matematičkom funkcijom. Što je pravac bliže točkama grafa to znači da je naša hipoteza ispravnija, to jest da se za određenu vrijednost ulazne varijable dana dobra procjena za izlaznu vrijednost. Zbog jednostavnosti objašnjenja i prikaza, prikazujemo sliku 2. na kojoj se kao krivulja odabire pravac.



Slika 2. prikaz regresijskog pravca

Ako bismo definirali regresijski pravac funkcijom

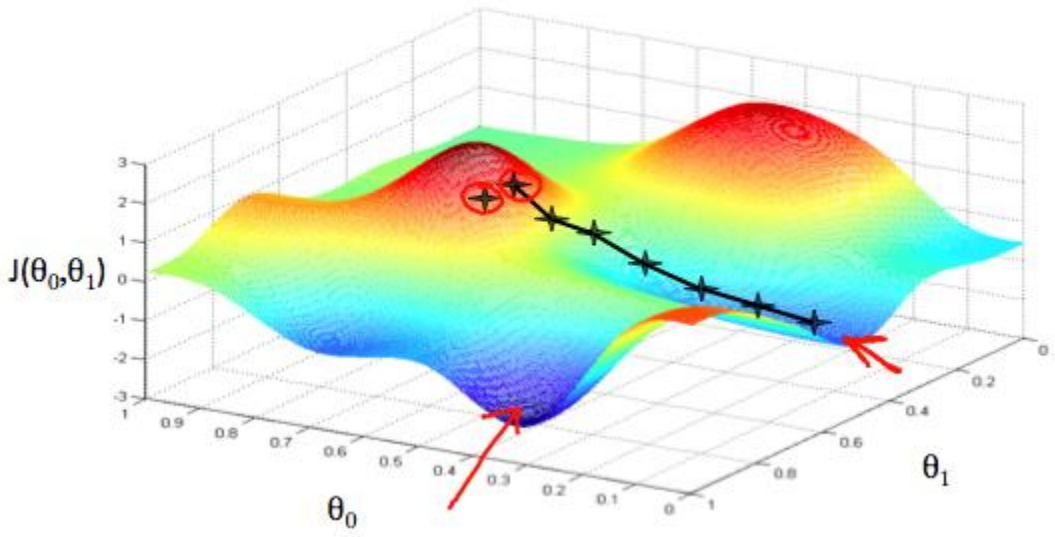
$$h_\theta(x) = \theta_0 + \theta_1 x \tag{3}$$

postavlja se pitanje kako bismo trebali odrediti parametre θ_0 i θ_1 kako bi pravac bio što bliže točkama $(x^{(i)}, y^{(i)})$. Rješenje navedenog problema se izražava sljedećim izrazom:

$$J = \frac{1}{2n}\sum_{i=1}^{n}(h_\theta(x^{(i)}) - y^{(i)})^2 \tag{4}$$

gdje je n broj elementa skupa testnih podataka. Izraz 4. predstavlja ciljnu funkciju. U ovom slučaju ciljna funkcija je zadana kao srednja vrijednost kvadrata razlike pretpostavljene i stvarne vrijednosti. Osim kvadrata srednje vrijednosti razlike pretpostavljene i stvarne vrijednosti koji se često koristi kao sredstvo za rješavanja regresijskih problema, ciljna funkcija se mogla odrediti i na neki drugi način. Cilj je dobiti što manju vrijednost ciljne funkcije, pošto je manja vrijednost ciljne funkcije pokazatelj dobro postavljene hipoteze. U našem slučaju (pošto je hipoteza zadana

izrazom (3)), minimizacija, to jest postizanje što manje vrijednosti ciljne funkcije se ostvaruje odabirom odgovarajućih parametara θ_0 i θ_1. Algoritam kojim ćemo se koristi da bismo dobili što manju vrijednost ciljne funkcije jest algoritam gradijenta spusta.



Slika 3. gradijentni spust [6]

Slika 3. prikazuje krivulju, odnosno vrijednost ciljne funkcije ovisno o određenom odabiru parametara θ_0 i θ_1. Zaokruženi crni križići pokazuju početne točke parametara θ_0 i θ_1, poveznica križića je korak gradijentnog spusta, dok su crvenim strjelicama prikazani lokalni minimumi funkcije J. Vidimo da je početna točka gradijentnog spusta vrh brijega gdje se nalaze križići koji su zaokruženi crvenom bojom. Nama je u interesu pronaći ciljnu funkciju koja kao rezultat korijena kvadrata srednje vrijednosti dava što manji rezultat, a to se upravo postiže korištenjem algoritma gradijentnog spusta pomoću kojeg dolazimo do minimuma funkcije. Gradijent spusta se ostvaruje tako da se simultano računaju parametri θ_0 i θ_1 , sve do ostvarenja konvergencije, odnosno do postizanja minimuma funkcije cilja. Određenim izmjenama ciljne funkcije dobiti ćemo sljedeće izraze:

$$\theta_0 \coloneqq \theta_0 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)\ ) \tag{5}$$

$$\theta_1 \coloneqq \theta_1 - \alpha \frac{1}{m}\sum_{i=1}^{m}((h_\theta(x_i) - y_i)x_i) \tag{6}$$

U izrazima 5. i 6. vrijedi: m je broj elemenata skupa, θ_0 i θ_1 su konstantne koje se simultano mijenjaju, x_i i y_i su vrijednosti iz skupa podataka za testiranje, dok je α korak gradijentnog spusta. Gornji izrazi nakon parametra α su zapravo parcijalne derivacije po θ_0 ili θ_1 – ako računamo θ_0 onda je parcijalna derivacija po θ_0, a ako računamo θ_1, onda je parcijalna derivacija po θ_1 [5].

Ova metoda pri svakom koraku računanja θ_0 i θ_1 uzima u obzir svaki element skupa za testiranje i zove se silazni gradijent spusta (eng. „Batch gradient descent“). Ako bi za razliku od ovog slučaja u kojem smo imali jednu varijablu (veličinu kuće) imali više varijabli (na primjer broj katova, blizina metro stanice, starost kuće, broj soba...) onda bi problem određivanja cijene kuće također bilo moguće riješiti, neovisno što je broj ulaznih varijabli veći od jedan. Tada bismo hipotezu prikazali na sljedeći način:

$$h_\theta(x) = \theta_0 + \theta_1 * x_1 + \theta_2 * x_2 + \theta_3 * x_3 + \cdots + \theta_n * x_n \quad (7)$$

U gornjem izrazu n je broj karakteristika koja određena n-torka (u našem primjeru kuća) posjeduje. Da bismo objasnili ciljnu funkciju trebamo objasniti sljedeće uvedene izraze: $x_j^{(i)}$ predstavlja vrijednost značajke j u i-tom elementu skupa, dok m kao i prije predstavlja broj elemenata. Opći oblik računanja parametra θ_j zadan je sljedećim izrazom::

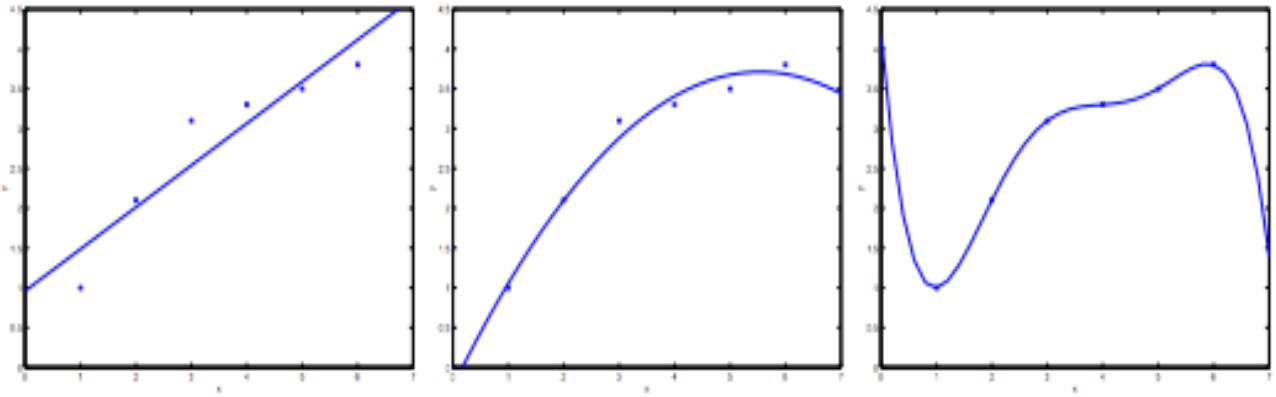
$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} ((h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}) \quad (8)$$

U slučaju raznolikih vrijednosti varijabli, kao na primjer broju površine kvadrata kuće, broju katova kuće te recimo blizini metro stanice, može se dogoditi da algoritam gradijentnog spusta bude usporen. Navedeni problem može se riješiti tako da skaliramo vrijednosti iz skupa podataka tako da se svaka vrijednost koja pripada varijabli x_i podijeli s razlikom najveće i najmanje vrijednosti x_i. Osim toga, potrebno je i napraviti normalizaciju srednje vrijednosti, to jest da svakoj vrijednost x_i oduzmemo prosječnu vrijednost varijable x. Idealno bi bilo da novodobivene vrijednosti budu unutar sljedećeg intervala: $-1 \leq x_i \leq 1$, ili još bolje: $-0.5 \leq x_i \leq 0.5$
Oba dvije navedene metode mogu se postići sljedećim računom:

$$x_i = \frac{x_i - \mu}{s_i} \quad (9)$$

U procesu učenja može se dogoditi da algoritam ostvaruje jako veliku uspješnost, a da se u stvarnim novonastalim situacijama pri pojavi problema dogodi da algoritam ne dava relevantno rješenje za novonastali problem. Navedena situacija se zove pretjerano prilagođavanje podacima (Eng. “Overfitting”). Također, može se dogoditi da se algoritam slabo prilagođava podacima te se ta situacija zove neprilagođavanje podacima (Eng. “Underfitting”).

Sljedeći grafovi prikazani na slici 4. prikazuju situacije pretjeranog prilagođavanja, kao i neprilagođavanja algoritma podacima.



Slika 4 - primjeri pretjeranog prilagođavanja i neprilagođavanja podacima [6]

Pretpostavimo da grafovima prikazanim na slici 4 pokušavamo pogoditi vrijednost y-a na temelju vrijednosti x-a. Ako za treći graf s desne strane koji pogađa sve vrijednosti izlazne varijable y uzmemo za primjer da vrijednost na y osi predstavlja vrijednost kuće, a da vrijednost x-a određuje površinu kuće, onda (zbog opadajućeg dijela grafa nakon zadnje točke koju krivulja pogađa) možemo pretpostaviti da algoritam neće dobro pogađati cijene kuće (pošto se vrijednost y-a koji predstavlja cijenu kuće smanjuje kako vrijednost x koji predstavlja površinu kuće raste, što nije slučaj u praksi). Po pitanju ovih grafova možemo reći da prva dva grafa s lijeve strane prikazuju problem ne prilagođavanja podacima, dok zadnji graf prikazuje problem pretjeranog prilagođavanja podacima.

Navedeni problemi mogu se riješiti na više načina, a neki od njih su:

- uvođenjem što više slučajnosti u postupku učenja, bilo korištenjem unakrsne međuvalidacije, slučajnih šuma ili nečeg sličnog
- Ručno da izaberemo koje ćemo atribute zadržati, to jest nad kojim će se atributima algoritam učiti
- Zadržati sve atribute, ali umanjiti vrijednost parametara θ_j
- Regularizacijom

Regularizacija se ostvaruje umanjivanjem vrijednosti parametara hipoteze kao i dodavanjem regularizacijskog člana. Ako zamislimo da imamo hipotezu zadanu sljedećim izrazom:

$$h_\theta(x) = \theta_0 + \theta_1 * x_1 + \theta_2 * x_2^2 + \theta_3 * x_3^3 \quad (10)$$

te se odlučimo da hoćemo smanjiti veličinu parametara θ_3 i θ_4 približno nuli, onda bi prema hipotezi jednadžbe (10) mogli izraziti ciljnu funkciju sljedećim izrazom:

$$J = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + 1000 * \theta_3^2 + 100 * \theta_4^2 \quad (11)$$

Ako imamo velik broj parametara θ i hoćemo napraviti regularizaciju koja uključuje sve parametre, onda to možemo napraviti tako da ciljnu funkciju izrazimo na sljedeći način:

$$J = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n}\theta_j^2 \quad (12)$$

U jednadžbi 12. parameter λ predstavlja regularizacijski parametar koji određuje razliku koja se ostvari smanjivanjem vrijednosti parametara θ kako bi do što manje vrijednosti J. U gornjem primjeru smo prikazali ciljnu funkciju, ali nismo objasnili kako se uvođenje regularizacije odražava na algoritam gradijentnog spusta. Pošto se θ_0 koristi da bi se postiglo presijecanje krivulje na osi Y (zbog toga je konvencionalno vrijednost $x_0 = 1$), a u uz to ne predstavlja veličinu nekog atributa, uvođenje regularizacije se ne odnosi na parametar θ_0, dok se na sve ostale parametre odnosi. Sljedeći izraz prikazuje način računanja gradijentnog spusta, to jest promjene na svim ostalima parametrima $\theta_1, \theta_2, ..., \theta_n$

$$\theta_j - \alpha\left[\left(\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}\right) + \frac{\lambda}{m}\theta_j\right];\ j \in \{1,2,\dots n\} \quad (13)$$

3.2 Primjer sustava za preporučivanje filmova

U ovom poglavlju ćemo prikazati sustav za preporučivanje filmova temeljen na suradnji koji pripada grupi sustava koja razvija određeni model koji se koristi za davanje preporuka. Algoritam koji ćemo koristiti jest algoritam linearne regresije s gradijentnim spustom.

Uvodimo sljedeće pojmove:

- N_u - broj korisnika u skupu
- N_m - broj filmova u skupu
- r(i, j) - odgovara vrijednosti 1 ako je korisnik j ocijenio film, inače je 0
- $y^{(i,j)}$ - ocjena filmu *i* dana od strane korisnika j
- Ocjene filmova mogu biti u vrijednosti od 0 do 5, ako korisnik nije ocijenio neki film onda je stavljen znak ?
- $\Theta^{(j)}$ vektor parametar koji predstavlja preferencije određenog žanra od strane korisnika
- $X^{(i)}$ vektor značajki određenog filma, to jest zastupljenost određenog žanra u određenom filmu
- $M^{(j)}$ – broj filmova ocijenjenih od strane korisnika j

FILM	Marko(1)	Petra(2)	Lara(3)	Toni(4)	X_1 (Sci-fi)	X_2 (romansa)
The Matrix	5	5	0	0	0.9	0
Alien	5	?	?	0	1.0	0.01
Interstellar	?	4	0	?	0.99	0
Romance forever	0	0	5	4	0.1	1
Titanic	0	0	5	?	0	0.9

Tablica 6 - ocjene filmova

Prema tablici 6 vidljivo je da se tablica sastoji od filmova, ocjena koju su pojedini korisnici dali filmu, kao i značajki (Sc-fi i romansa) filmova. Prema uvedenoj notaciji, N_u bi bio 4, N_m bi bio 5, dok bi $y^{(1,2)}$ bila ocjena 5 korisnice Petre filmu „The Matrix“.

Za značajke filma uvodimo vektor x_i, gdje je prva vrijednost vektora 1 (kao što je navedeno, radi se o konvenciji da je prvi član hipoteze uvijek 1), dok ostale dvije vrijednosti predstavljaju udio Sci-fi i romanse kao žanrova u filmu.

Na primjer, vektor x1, dakle značajke filma „Matrix“ mogli bismo (proizvoljno određeno tako da vrijednost žanra bude u relaciji s vrijednosti ocjene prema izrazu 10) iskazati sljedećim izrazom:

$$x^1 = \begin{pmatrix} 1 \\ 0.9 \\ 0 \end{pmatrix} \tag{14}$$

Prvi problem koji pokušavamo riješiti jest da za svakog korisnika j naučimo parametar $\theta^{(j)} \in \mathbb{R}^3$. Rezultat koji dobijemo (parametar $\theta^{(j)}$) predstavlja koliko koji korisnik favorizira koji žanr. Kada otkrijemo vrijednost parametra $\theta^{(j)}$, onda ćemo dalje koristiti $\theta^{(j)}$ kako bismo izračunali koju bi ocjenu korisnik j dao filmu i. Ocjenu ćemo dobiti kao rezultat sljedećeg izraza

$$\left(\theta^{j}\right)^{T} x^{i} \tag{15}$$

Znak T u izrazu 15. nije eksponent već označava da se vektor $\theta^{(j)}$ transponira, to jest, da stupci i redci vektora zamijene mjesta. Dakle, prvi stupac postane prvi redak, drugi stupac postane drugi redak i tako dalje, sve do n-tog retka kada n-ti redak postaje n-ti stupac. U slučaju davanja preporuka, ako bismo korisniku dali n preporuka onda bi se navedeni račun mogao koristiti gdje bi se gledalo top n ocjena te bi se filmovi s top n ocjena koristili kao preporuka. Na primjer ako bi nas zanimalo koju bi ocjenu Marko dao filmu Interstellar, možemo razmišljati na sljedeći način: Marko je ocijenio s visokom ocjenom filmove „The Matrix“ i „Alien“, što znači da Marko favorizira Sci-fi filmove. Na temelju Markovih preferencija prema određenim žanrovima možemo izraziti vektorom $\theta^{(1)}$ koji bi poprimio sljedeće vrijednosti:

$$\theta^1 = \begin{pmatrix} 0 \\ 5 \\ 0 \end{pmatrix} \tag{16}$$

U izrazu 16. prva vrijednost vektora je 0 pošto se ona množi s vektorom x koji na istoj i,j poziciji ima vrijednost 0, a koji ovdje ne igra neku važnost pri ocjenjivanju filma (a uz to imamo i dva žanra), druga vrijednost (element na poziciji 0,1, dakle broj 5) je vrijednost udjela Sci-fi žanra u filmu, dok je treća vrijednost odraz udjela romanse u filmu. Sada kada smo odredili parametar $\theta^{(1)}$ te znamo da film Interstellar ima vrijednosti 0.99 za Sci-fi i 0.0 za romansu, onda možemo izračunati ocjenu koju bi Marko dao filmu Interstellar množeći vektor $(\theta^{(1)})^T$ s vektorom $x^{(3)}$ te dobiti ocjenu 4.95, što je dosta blizu ostalim ocjenama koje je Marko dao za Sci-fi filmove. Ovo je u suštini regresijski problem – da bismo odabrali parametar θ_j tražimo vrijednost x_i koja pomnožena s θ_j dava što bližu vrijednost ocjeni koja pripada odgovarajućem filmu, to jest ocjenu koju je određeni korisnik dao određenome filmu. Formalnije, problem određivanja parametra $\theta^{(j)}$ možemo zapisati pomoću sljedećeg izraza:

$$\min_{\theta^{(i)}} \ \frac{1}{2}\sum_{i:r(i,j)=1}\left(\left(\theta^{(j)}\right)^{T}\left(x^{(i)}\right)-y^{(i,j)}\right)^{2}+\frac{\lambda}{2}\sum_{k=1}^{n}\left(\theta_{k}^{(j)}\right)^{2} \tag{17}$$

Izraz 17. predstavlja rješenje za minimizaciju pojedinačnog parametra θ koji pripada jednom korisniku, dok izraz 12. označava izraz kojim se minimiziraju svi parametri $\theta^{(1)},\ldots,\theta^{(n)}$, dakle parametri θ svih korisnika.

$$\min_{\theta^{(1)},\ldots,\theta^{n_m}} \ \frac{1}{2}\sum_{j=1}^{n_u}\ \sum_{i:r(i,j)=1}\left(\left(\theta^{(j)}\right)^{T}\left(x^{(i)}\right)-y^{(i,j)}\right)^{2}+\frac{\lambda}{2}\sum_{j=1}^{n_u}\ \sum_{k=1}^{n}\left(\theta_{k}^{(j)}\right)^{2} \tag{18}$$

Uvrštavajući dobivene formule u algoritam gradijentog spusta dobivamo sljedeće izraze

$$\theta_{k}^{j} \coloneqq \theta_{k}^{(j)} - \alpha\left[\sum_{i:r(i,j)=1}\left(\left(\theta^{(j)}\right)^{T}\left(x^{(i)}\right)-y^{(i,j)}\right)x_{k}^{(i)}\right]; \text{za } k = 0 \tag{19}$$

$$\theta_{k}^{j} \coloneqq \theta_{k}^{(j)} - \alpha\left[\sum_{i:r(i,j)=1}\left(\left(\theta^{(j)}\right)^{T}\left(x^{(i)}\right)-y^{(i,j)}\right)x_{k}^{(i)}\right) + \frac{\lambda}{m}\,\theta_{k}^{(j)}\Big]; \text{za k različito od 0} \tag{20}$$

U prethodnom primjeru smo imali slučaj da smo na temelju poznatih vrijednosti vektora x i ocjena pokušali doći do parametara θ. U sljedećem primjeru ćemo na temelju parametara θ pokušati doći do vrijednosti vektora x pomoću izraza 15. Da bismo objasnili postupak kako doći do vrijednosti parametara θ na temelju vrijednosti parametara x koristimo se tablicom 5.

FILM	Marko(1)	Petra(2)	Lara(3)	Toni(4)	X_1 (Sci-fi)	X_2 (romansa)
The Matrix	5	5	0	0	?	?
Alien	5	?	?	0	?	?
Interstellar	?	4	0	?	?	?
Romance forever	0	0	5	4	?	?
Titanic	0	0	5	?	?	?

Tablica 7 - određivanje vrijednosti žanra

Zamislimo da smo na temelju ocjena filmova pokušali zaključiti da prva tri filma pripadaju jednom žanru, dok zadnja dva filma („Romance forever“ i „Titanic“) pripadaju nekom drugom žanru. Zaključivanje koje koristimo možemo opravdati time da su Marko i Petra ocijenili visokim ocjenama prva tri filma (dakle filmove „The Matrix“, „Alien“ i „Interstellar“), dok su Lara i Toni dali jako niske ocjene za prva tri filma filmove. U slučaju zadnja dva filma (dakle filma „Romance forever“ i filma „Titanic“) vidimo da su Lara i Toni dali visoke ocjene, dok su Marko i Petra dali

niske ocjene. Odraz navedenog razmišljanja su vektori θ koji redoslijedno pripadaju broju koji se nalazi kraj imena korisnika u tablici 7.

$$\theta^{(1)} = \begin{pmatrix} 0 \\ 5 \\ 0 \end{pmatrix} \theta^{(2)} = \begin{pmatrix} 0 \\ 5 \\ 0 \end{pmatrix} \theta^{(3)} = \begin{pmatrix} 0 \\ 0 \\ 5 \end{pmatrix} \theta^{(4)} = \begin{pmatrix} 0 \\ 0 \\ 5 \end{pmatrix}$$

Na temelju dobivenih parametara θ, koristeći se izrazom 15. možemo doći do vrijednosti vektora x. Za naše primjere, matematički pojednostavljeno, problematika određivanja vrijednosti atributa vektora $x^{(1)}$ može se prikazati sljedećim sustavom jednadžba:

$$(\theta^{1})^{\mathrm{T}}\left(x^{(1)}\right) \approx 0.5$$
$$(\theta^{2})^{\mathrm{T}}\left(x^{1)}\right) \approx 0.5$$
$$(\theta^{3})^{\mathrm{T}}\left(x^{1)}\right) \approx 0$$
$$(\theta^{3})^{\mathrm{T}}\left(x^{1)}\right) \approx 0$$

Formalnije, matematički izražen problem u kojem imamo vrijednosti $\theta^{(1)}, \theta^{(2)},..., \theta^{(n)}$ i pokušavamo odrediti vrijednosti $x^{(1)}$ može se zapisati na sljedeći način:

$$\min_{x^{(i)}} \frac{1}{2}\sum_{j:r(i,j)=1}\left(\left(\theta^{(j)}\right)^{\mathrm{T}}\left(x^{(i)}\right) - y^{(i,j)}\right)^{2} + \frac{\lambda}{2}\sum_{k=1}^{n}\left(x_{k}^{(i)}\right)^{2} \tag{21}$$

Dok se problem u slučaju u kojem imamo vrijednosti $\theta^{(1)}, \theta^{(2)},..., \theta^{(n)}$ i pokušavamo odrediti vrijednosti $x^{(1)}, x^{(2)},..., x^{(n)}$, može se zapisati na sljedeći način:

$$\min_{x^{(1)},\ldots,x^{n_m}} \frac{1}{2}\sum_{j=1}^{n_u}\sum_{i:r(i,j)=1}\left(\left(\theta^{(j)}\right)^{\mathrm{T}}\left(x^{(i)}\right) - y^{(i,j)}\right)^{2} + \frac{\lambda}{2}\sum_{j=1}^{n_u}\sum_{k=1}^{n}\left(\theta_{k}^{(j)}\right)^{2} \tag{22}$$

Dakle, u jednadžbi 22. smo pokazali da ako imamo vrijednosti $\theta^{(1)}, \theta^{(2)},..., \theta^{(n)}$ da možemo odrediti vrijednosti $x^{(1)}, x^{(2)},..., x^{(n)}$ te također obrnuto – u jednadžbi 18. smo pokazali da ako imamo vrijednosti $x^{(1)}, x^{(2)},..., x^{(n)}$ da možemo doći do vrijednosti $\theta^{(1)}, \theta^{(2)},..., \theta^{(n)}$. Jedan od načina kako bi u stvarnosti (ako bi imali određene ocjene filmova od strane korisnika) mogli pristupiti problemu određivanja nepoznatih ocjena, to jest, kako bi za nekog korisnika napravili prognozu s

kojom bi ocjenom ocijenio određeni film, jest da slučajno inicijaliziramo vrijednosti $\theta^{(1)}$, $\theta^{(2)}$,..., $\theta^{(n)}$ te pomoću inicijaliziranih vrijednosti dobijemo određene vrijednosti $x^{(1)}$, $x^{(2)}$,..., $x^{(n)}$ te pomoću dobivenih vrijednosti $x^{(1)}$, $x^{(2)}$,..., $x^{(n)}$ pokušamo odrediti $\theta^{(1)}$, $\theta^{(2)}$,..., $\theta^{(n)}$ i tako navedene procese naizmjenično ponavljati dok se ne postigne odgovarajuća vrijednost ciljne funkcije [5]. Međutim, u izrazu 18. imamo da vanjska suma uzima u obzir sve korisnike j, a unutarnja suma za izračun sume kvadrata pogreške uzima u obzir sve filmove i koje je korisnik j iz vanjske sume ocijenio. Dok u izrazu 22. imamo isti proces samo na obrnuti način – vanjska suma uzima svaki film i, a unutarnja suma svakog korisnika j koji je ocijenio film i. Ova dva izraza se mogu udružiti u jedan izraz koji simultano minimizira i vrijednosti $\theta^{(1)}$, $\theta^{(2)}$,..., $\theta^{(n)}$ i vrijednosti $x^{(1)}$, $x^{(2)}$,..., $x^{(n)}$:

$$J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)}) = \frac{1}{2}\sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} \left(\left(\theta^{(j)}\right)^T \left(x^{(i)}\right) - y^{(i,j)}\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{n_m} \sum_{k=1}^{n} \left(x_k^{(i)}\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{n_u} \sum_{k=1}^{n} \left(\theta_k^{(j)}\right)^2 \quad (23)$$

Dakle, kolaboracijski algoritam bi se odvijao u sljedećim koracima:

1. Inicijalizacija vrijednosti značajki $x^{(1)}$, $x^{(2)}$,..., $x^{(n)}$ te parametara $\theta^{(1)}$, $\theta^{(2)}$,..., $\theta^{(n)}$ na male vrijednosti
2. Minimiziranje ciljne funkcije $J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)})$ koristeći algoritam gradijentog spusta ili nekog drugog algoritma. Ako bismo koristili gradijenti spust, za vrijednosti j = 1, ..., n_u te i = 1, ... ,n_m prema izrazu (18), računali bismo vrijednosti atributa $x_k^{(i)}$ i parametara $\theta_k^{(j)}$ na sljedeći način:

$$x_k^{(i)} := x_k^{(i)} - \alpha \left[\sum_{j:r(i,j)=1} \left(\left(\theta^{(j)}\right)^T \left(x^{(i)}\right) - y^{(i,j)}\right)\theta_k^{(j)} + \frac{\lambda}{m}\theta_k^{(j)}\right] \quad (24)$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \left[\sum_{i:r(i,j)=1} \left(\left(\theta^{(j)}\right)^T \left(x^{(i)}\right) - y^{(i,j)}\right)x_k^{(i)} + \frac{\lambda}{m}\theta_k^{(j)}\right] \quad (25)$$

3. Za korisnika j odrediti ocjenu za film i prema izrazu 10.

Za sada smo riješili problem određivanja vrijednosti atributa filma, kao i problem stvaranja korisničkog profila, to jest, za svakog smo korisnika uspjeli doći do parametra θ koji predstavlja korisničke preferencije određenog žanra. Dalje smo na temelju poznatih vrijednosti parametara θ i x uspjeli doći do prognoza ocjene koja se tiče filma kojeg korisnik nije ocijenio. Što je ocjena veća za određeni film, to je i veća vjerojatnost da bi taj film bio relevantan korisniku. Također, u slučaju da je sustav za preporučivanje filmova u početku rada (zbog čega onda imamo oskudan

broj ocjena filmova), onda se pretpostavljena ocjena može dodati u skup kojeg čine ocjene svih filmova kako bi nova ocjena riješila problem oskudnosti ocjena, a uz to može i poslužiti za sljedeću iteraciju učenja modela.

Još su nam ostala dva problema koje bi sustav za preporučivanje filmova trebao rješavati: problem nalaženja sličnih filmova kao i problem novog korisnika, to jest, situacije u kojoj novi korisnik nije ocijenio određene filmove te mu je s toga teško dati bilo kakvu preporuku. Prethodno smo objasnili kako možemo doći do značajki određenog filma, a sada te značajke možemo iskoristiti kako bismo saznali jesu li neki filmovi slični. Ako uspoređujemo filmove A i B, onda filmovi A i B imaju određene značajke koje karakteriziraju navedene filmove. Što su značajke sličnije to su i filmovi sličniji. Pošto vektor x opisuje značajke filmova, onda bi se sličnost filmova A i B mogla odrediti pomoću euklidske udaljenosti vektora $x^{(a)}$ koji opisuje film A i vektora $x^{(B)}$ koji opisuje film B (gdje su A i B proizvoljno imenovani filmovi, moglo se napisati i film j = 1 te film j = 2).

Da bismo opisali rješenje problema novog korisnika definiramo matricu Y koja predstavlja ocjene određenih filmova koje smo imali u prethodnoj tablici, s time da smo matrici nadodali jedan redak koji predstavlja ocjene novog korisnika. Također uvodimo vektor μ koji predstavlja prosjek ocjena određenog filma.

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix} \quad (26)$$

$$\mu = \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{bmatrix} \quad (27)$$

Novi korisnik nije ocijenio niti jedan film pa se za njega ne može izračunati parametar θ - pošto se ocjena određenog filma računa prema izrazu 15. Jedno od rješenja je da novom korisniku pridijelimo ocjene filmova koje su rezultat prosjeka ocjena svih korisnika za određeni film za koji postoji ocjena. Da bismo to napravili, svaki element retka i matrice Y se oduzme od odgovarajuće vrijednosti koja stoji na poziciji (i,1) vektora μ.

U slučaju naše matrice (izraza 26.), normalizacijom bi se dobila matrica sa sljedećim vrijednostima:

$$Y = \begin{bmatrix} 2.5 & 2.5 & -2.5 & -2.5 & ? \\ 2.5 & ? & ? & -2.5 & ? \\ ? & 2 & -2 & ? & ? \\ -2.25 & -2.25 & 2.75 & 1.75 & ? \\ -1.25 & -1.25 & 3.75 & -1.25 & ? \end{bmatrix} \quad (28)$$

Prednosti sustava za preporučivanje baziranih na kolaboraciji

- Nije potrebno znanje o domeni područja kojem algoritam rješava određeni problem
- Moguće je otkrivanja određenih specifičnosti objekta iz baze
- Moguće je preporučiti novi i ne popularni objekt
- Moguće je riješiti problem novog korisnika

Nedostatci sustava za preporučivanje baziranih na sadržaju

- Nemogućnost davanja preporuka u situacijama kada imamo oskudan broj vrijednosti objekta kojeg te vrijednosti opisuju – u našem slučaju to su ocjene filmova
- Složenost – da bi algoritam dao određenu preporuku treba vršiti određeni izračun (na primjer sličnost) za svakog korisnika ili objekta
- Ovisi o aktivnosti korisnika

4. Evaluacija sustava za davanje preporuka

Srednja vrijednost sume kvadrata apsolutne vrijednosti razlike pretpostavljene i stvarne vrijednosti (eng. Mean Absolute Error (MAE)) računa se na sljedeći način [6]:

$$MAE = \frac{1}{n}\sum|procjena - stvarna\ vrijednost| \tag{29}$$

Za primjer računanja srednje vrijednosti sume kvadrata apsolutne vrijednosti razlike procijenjene i stvarne ocjene, prikazujemo tablicu 8 s procijenjenim ocjenama i stvarnim ocjenama, kao i pogreškama pri procijeni (pogreška je razlika procijenjene i stvarne ocjene).

PROCJENA OCJENE	STVARNA OCJENA	POGREŠKA
5	3	2
4	1	3
5	4	1
1	1	0

Tablica 8 - Procijenjene i stvarne ocjene - primjer za računanje srednje apsolutne pogreške

Ako uvrstimo podatke iz tablice 8 u formulu 29. možemo izračunati MAE na sljedeći način:

$$MAE = \frac{(2+3+1+0)}{4} = 1.5$$

Dakle, vrijednost srednje apsolutne pogreške za podatke iz tablice 8 bi bila 1.5 Srednja vrijednost korijena kvadrata razlike (eng. Root Mean Squared Error (RMSE)) stvarne vrijednosti i vrijednosti koja je nastala kao pretpostavka algoritma računa se prema izrazu 21. Prednost srednje kvadratne pogreške u odnosu na srednju apsolutnu pogrešku je ta da se veće pogreške strože penaliziraju, to jest, imat ćemo bolji uvid ako je došlo do većih grešaka u pretpostavci, dok je nedostatak srednje kvadratne pogreške u odnosu na srednju apsolutnu pogrešku taj da se može koristi samo za varijable koji imaju istu jediničnu vrijednost [7].

$$RMSE = \sqrt{\sum\frac{procjena - stvarna\ vrijednost}{n}} \tag{30}$$

PROCJENA OCJENE	STVARNA OCJENA	POGREŠKA	POGREŠKA^2
5	3	2	4
4	1	3	9
5	4	1	1
1	1	0	0

Tablica 9 - Procijenjene i stvarne ocjene - primjer za računanje srednje vrijednosti korijena kvadrata razlike

Ako uvrstimo podatke iz tablice 9. u formulu za računanje srednje vrijednosti korijena kvadrata razlike (izraz 30.) imamo sljedeći račun:

$$RMSE = \sqrt{\frac{(4+9+1+0)}{4}} = 1.87$$

Dakle, vrijednost korijena kvadrata razlike stvarne i pretpostavljene vrijednosti za podatke iz tablice 8 bi bila 1.87.

Pokrivenost (eng. coverage) - predstavlja postotak (objekata, korisnika, ocjena) koji je sustav za preporučivanje uspio preporučiti. Nedovoljna pokrivenost je često opravdana nedovoljnim brojem ocjena. Vrijedi pozitivna korelacija između pokrivenosti i slučajnosti preporuke određenog objekta.

Raznovrsnost (eng. diversity) – 1 – S, gdje je S prosječna sličnost preporučenih objekata. Primjer niske sličnosti bi bila preporuka dvije knjige, jedne knjige koja predstavlja prvi dio serijala, dok druga knjiga predstavlja drugi dio serijala. Često se ispostavlja (ali nije pravilo) da je velika raznolikost odraz lošeg sustava za davanje preporuka. Na primjer, ako ponovo prikažemo tablicu 5:

Articles	Analytics	Data	Cloud
Article 1	0.61103701	0.626296217	0
Article 2	0.594389962	0.691941368	0.324895184
Article 3	0.483581962	0.568817887	0.353681311

Article Vector	Cosine Values
cos(A1,A2)	0.796554526
cos(A2,A3)	0.795934246
cos(A1,A3)	0.651734967

Možemo izračunati prosječnu sličnost prema podacima koje imamo u tablici 5 te dobiti da je prosječna sličnost prema sljedećem računu:

$$\frac{(0.796554526 + 0.79593426 + 0.651734967)}{3} = 0.748.$$

Dakle, rezultat raznovrsnosti bi bio 0.252.

Novost (eng. novelty) – predstavlja popularnost objekata koji preporučujemo. Nije dobra ni previsoka, ali ni preniska novost. Sustavi s niskom novosti ne rješavaju problem dugog repa te će zbog toga korisnik uglavnom za preporuku dobiti određeni objekt kojeg ne može pozitivno iznenaditi, objekt koji vjerojatno ne odgovara samim užim interesima koje korisnik posjeduje pa je s toga moguće da korisnik razvije nepovjerenje u sustav. S druge strane, visoka novost podrazumijeva ne preporučivanje popularnih objekata pa je moguće da se na većem broju preporučenih objekata ne pojave objekti koji su relevantni većini korisnika. Jedan od načina kako možemo izračunati novost jest da na novost gledamo kao omjer poznatih i nepoznatih objekata na preporučenom objektu. Prema Baeza-Yatesu i Ribieru-Neti, možemo definirati skup L kao skup objekata koje korisnik preferira (da bismo došli do onih objekata koje korisnik preferira možemo koristiti kriterij sličnosti između onih objekata koje je korisnik ocijenio s visokom ocjenom, na primjer) [6]. Dalje, skup L možemo podijeliti na dva skupa, skup L_k čiji su elementi objekti koji su korisniku poznati (na primjer, kao kriterij poznatosti možemo uzeti da je korisnik ocijenio neki objekt) te skup L_u kojeg čine objekti koji su korisniku nepoznati (kao kriterij nepoznatosti možemo uzeti objekte koje korisnik još nije ocijenio). Novost (N) se onda računa prema sljedećoj formuli:

$$N = \frac{|L_k|}{|L|} \tag{31}$$

Ako bismo zamislili da prema tablici 5 imamo jednog korisnika koji je pozitivno ocijenio prvi članak te dalje imamo kriterij da korisnik preferira članke koji su 60% slični onom članku kojeg je korisnik pozitivno ocijenio, onda bi iznos novosti preporuke za zamišljenog korisnika bila 0.33 (L = 3, pošto smo za kriterij koje bi filmove korisnik preferirao uzeli one filmove koji su slični filmu kojemu je korisnik dao visoku ocjenu, dok je L_k broj filmova koje je korisnik ocijenio - u našem primjeru to je samo jedan film.

5. Implementacija i evaluacija implementiranog sustava za davanje preporuka filmova

5.1 Implementacija sustava

Za implementaciju sustava za davanje preporuke opisanom u poglavlju 3.2 koristimo programski jezik Python kao i module NumPy i SciPy. Podatke koje koristimo za učenje algoritma, a kasnije i za davanje preporuka preuzeti su s GitHub korisničkog računa koji je dostupan na sljedećem linku:

https://github.com/yanfei-wu/ml-ng/tree/master/ex8_Anomaly%20Detection%20and%20Recommender%20Systems

Podaci su spremljeni unutar datoteke „ex8_movies.mat“ te unutar datoteke „ex8_moviesParams.mat“. Podaci koje se nalaze unutar datoteke „ex8_movies.mat“ nalaze se unutar matrice Y koja prikazuje ocjene filmova od strane određenih korisnika i matrice R koja s vrijednostima 1 i 0 indicira je li određeni korisnik ocijenio određeni film; vrijednost 0 znači da nije, dok vrijednost 1 znači da jest. Uključivanje modula kao i učitavanje podataka je prikazano sljedećim programskim kodom prikazanim na slici:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import loadmat
filmovi = loadmat("ex8_movies.mat")
parametri = loadmat("ex8_movieParams.mat")
Y = filmovi["Y"] # 1682 X 943 matrica, sadržava ocijene (1-5) od 1682 filmova i 943 korisnika
R = filmovi["R"] # 1682 X 943 matrica, gdje R(i,j) = 1 ako i samo ako je korisnik j dao ocjenu filmu i
X = parametri["X"] # 1682 X 10 matrica, broj_filmova X broj_znacajki - matrica koja predstavlja značajke filmova
Theta = parametri["Theta"] # 943 X 10 matrica, br_korisnika X br_znacajki matrica značajki korisnika
```

Slika 5 - učitavanje modula i datoteka potrebnih za rad algoritma

Prema slici 5 vidimo da imamo 1682 filma, 943 korisnika, kao i 10 atributa. Atributi nemaju neko specifično ime (bar dok ih ne imenujemo ovisno o nekom zaključivanju koje bi poslužilo kao kriterij imenovanja atributa) – jedino što o njima znamo jest da su numerički i da na kraju kada algoritam završi učenje modela da ti atributi predstavljaju, u slučaju matrice X udio određenog atributa u filmu, dok u slučaju matrice Theta, atributi predstavljaju udio korisničkih preferencija prema određenom atributu.

Programskim kodom koji je prikazan na slici 6. dolazimo do normalizirane matrice kako bismo mogli riješiti problem novog korisnika:

```
def normalizacijaOcjena(Y, R):
    """
    normalizirana matrica Y
    """

    m,n = Y.shape[0], Y.shape[1]
    Y_prosijek = np.zeros((m,1))
    Y_normaliziran = np.zeros((m,n))

    for i in range(m):
        Y_prosijek[i] = np.sum(Y[i,:])/np.count_nonzero(R[i,:])
        Y_normaliziran[i,R[i,:]==1] = Y[i,R[i,:]==1] - Y_prosijek[i]

    return Y_normaliziran, Y_prosijek
```

Slika 6 - normalizacije matrice kako bi se riješio problem novog korisnik

Slika 7. prikazuje programski kod kojim se ostvaruje minimizacija parametara θ i x:

```
def minimizacijaParametara(params, Y, R, br_korisnika, br_filmova, br_atributa, Lambda):
    """
    Povratna vrijednost je funkcija cilja i gradijent te regularizirani gradijent i regularizirana funkcija cilja
    """

    X = params[:br_filmova*br_atributa].reshape(br_filmova,br_atributa)
    Theta = params[br_filmova*br_atributa:].reshape(br_korisnika,br_atributa)

    procjene =  X @ Theta.T
    greška = (procjene - Y)
    J = 1/2 * np.sum((greška**2) * R)

    #izračunaj reguraliziranu funkciju cilja
    reg_X =  Lambda/2 * np.sum(Theta**2)
    reg_Theta = Lambda/2 *np.sum(X**2)
    reg_J = J + reg_X + reg_Theta

    #izračunaj gradijent
    X_grad = greška*R @ Theta
    Theta_grad = (greška*R).T @ X
    grad = np.append(X_grad.flatten(),Theta_grad.flatten())

    #izračunaj regularizirani gradijent
    reg_X_grad = X_grad + Lambda*X
    reg_Theta_grad = Theta_grad + Lambda*Theta
    reg_grad = np.append(reg_X_grad.flatten(),reg_Theta_grad.flatten())

    return J, grad, reg_J, reg_grad
```

Slika 7- programski kod kojim se ostvaruje minimizacija parametara θ i x

Vrijednosti funkcije cilja kada je λ = 0 i kada je λ = 1.5 možemo vidjeti na slici 8.

```
br_korisnika, br_filmova, br_atributa = 943,1682,10
X_test = X[:br_filmova,:br_atributa]
Theta_test= Theta[:br_korisnika,:br_atributa]
Y_test = Y[:br_filmova,:br_korisnika]
R_test = R[:br_filmova,:br_korisnika]
parametri = np.append(X_test.flatten(),Theta_test.flatten())
# Izracunavanje funkcije cilja
J, grad = minimizacijaParametara(parametri, Y_test, R_test, br_korisnika, br_filmova, br_atributa, 0)[:2]
print("Iznos funkcije cilja s lambda = 0 je :",round(J,3))
J2, grad2 = minimizacijaParametara(parametri, Y_test, R_test, br_korisnika, br_filmova, br_atributa, 1.5)[2:]
print("Iznos funkcije cilja s lambda = 0 je :",round(J2,3))
```

```
Iznos funkcije cilja s lambda = 0 je : 76476.32
Iznos funkcije cilja s lambda = 0 je : 82410.25
```

Slika 8 - iznosi ciljnih funkcija ovisno o različitim vrijednostima parametra λ

Vidimo da je iznos ciljne funkcije, dakle u slučaju bez ikakvog treniranja modela kada su vrijednosti parametara θ i x jednake onim početnim iz datoteke „ex8_movieParams.mat“, 76476.32 kada je λ = 0, dok je iznos funkcije cilja kada je λ = 1.5, 84210.25.

Programskim kodom prikazanim na slici 9 izračunava se gradijent za svaki parametar:

```
def izracunaj_gradient(pocetni_parametri, Y, R, br_korisnika, br_filmova, br_atributa, alpha, br_iteracija, Lambda):
    """
    Optimiziranje X i Theta iz primjera
    """
    # unfold the parameters
    X = pocetni_parametri[:br_filmova*br_atributa].reshape(br_filmova,br_atributa)
    Theta = pocetni_parametri[br_filmova*br_atributa:].reshape(br_korisnika,br_atributa)
    J_vrijednosti =[]
    for i in range(br_iteracija):
        parametri = np.append(X.flatten(),Theta.flatten())
        rez_fun_cilja, grad = minimizacijaParametara(parametri, Y, R, br_korisnika, br_filmova, br_atributa, Lambda)[2:]
        X_grad = grad[:br_filmova*br_atributa].reshape(br_filmova,br_atributa)
        Theta_grad = grad[br_filmova*br_atributa:].reshape(br_korisnika,br_atributa)
        X = X - (alpha * X_grad)
        Theta = Theta - (alpha * Theta_grad)
        J_vrijednosti.append(rez_fun_cilja)

    parametri_finalno = np.append(X.flatten(),Theta.flatten())
    return parametri_finalno , J_vrijednosti
```

Slika 9 – programski kod za dobivanje finalnih parametara kao i finalne funkcije cilja

6.2 Evaluacija sustava

Za evaluaciju sustava koristimo tri metode. Evaluacija prve metode podrazumijeva unos ocjena određenim filmovima. Nakon unosa ocjena, sustav ima izgrađen profil te na temelju profila korisnika, sustav može preporučiti top n filmova. Ako se u top-n preporuka prepozna poveći broj filmova koji korisnik smatra relevantnim, onda možemo reći da sustav dobro funkcionira. Druga metoda koja se koristi je metoda korijena kvadrata srednje vrijednosti čija je formula zadana izrazom 21. Ako je rezultat korijena kvadrate srednje vrijednosti nizak, onda se smatra da sustav dobro funkcionira, u suprotnom ako je rezultat korijena kvadrata srednje vrijednosti visok, onda se smatra da sustav ne funkcionira dobro.

Slika 10 prikazuje popis ocjene filmova kao i ime filma kojeg smo ocijenili:

```
Unijeli ste ocjene za sljedeće filmove:

Ocjena 4 za film  5 Copycat (1995)
Ocjena 5 za film  55 Professional, The (1994)
Ocjena 1 za film  66 While You Were Sleeping (1995)
Ocjena 5 za film  76 Carlito's Way (1993)
Ocjena 5 za film  77 Firm, The (1993)
Ocjena 2 za film  78 Free Willy (1993)
Ocjena 5 za film  100 Fargo (1996)
Ocjena 4 za film  264 Mimic (1997)
Ocjena 5 za film  265 Hunt for Red October, The (1990)
Ocjena 5 za film  273 Heat (1995)
Ocjena 2 za film  354 Wedding Singer, The (1998)
Ocjena 2 za film  383 Flintstones, The (1994)
Ocjena 2 za film  392 Man Without a Face, The (1993)
Ocjena 4 za film  397 Striking Distance (1993)
Ocjena 2 za film  405 Mission: Impossible (1996)
Ocjena 2 za film  457 Free Willy 3: The Rescue (1997)
Ocjena 2 za film  495 Around the World in 80 Days (1956)
Ocjena 2 za film  498 African Queen, The (1951)
Ocjena 4 za film  558 Heavenly Creatures (1994)
Ocjena 5 za film  572 Blown Away (1994)
Ocjena 4 za film  573 Body Snatchers (1993)
Ocjena 2 za film  575 City Slickers II: The Legend of Curly's Gold (1994)
Ocjena 4 za film  590 Hellraiser: Bloodline (1996)
Ocjena 4 za film  591 Primal Fear (1996)
Ocjena 2 za film  596 Hunchback of Notre Dame, The (1996)
Ocjena 2 za film  617 Blue Angel, The (Blaue Engel, Der) (1930)
Ocjena 2 za film  623 Angels in the Outfield (1994)
Ocjena 5 za film  628 Sleepers (1996)
Ocjena 2 za film  629 Victor/Victoria (1982)
Ocjena 4 za film  654 Chinatown (1974)
Ocjena 5 za film  693 Casino (1995)
Ocjena 3 za film  701 Wonderful, Horrible Life of Leni Riefenstahl, The (1993)
Ocjena 2 za film  702 Barcelona (1994)
Ocjena 3 za film  839 Loch Ness (1995)
Ocjena 2 za film  867 Whole Wide World, The (1996)
Ocjena 4 za film  885 Phantoms (1998)
Ocjena 2 za film  893 For Richer or Poorer (1997)
Ocjena 1 za film  901 Mr. Magoo (1997)
Ocjena 4 za film  931 Island of Dr. Moreau, The (1996)
Ocjena 2 za film  962 Ruby in Paradise (1993)
Ocjena 2 za film  969 Winnie the Pooh and the Blustery Day (1968)
Ocjena 1 za film  988 Beautician and the Beast, The (1997)
Ocjena 2 za film  989 Cats Don't Dance (1997)
Ocjena 2 za film  999 Clean Slate (1994)
```

Slika 10 - ocjene filmova koje se koriste u svrhu evaluacije sustava

Slika 10 prikazuje preporuke koje sam dobio od strane sustave sustava:

```
Top 20 ocijena:

Procijenjena ocjena 8.8  za film s indexom 174 Raiders of the Lost Ark (1981)
Procijenjena ocjena 8.8  za film s indexom 64 Shawshank Redemption, The (1994)
Procijenjena ocjena 8.8  za film s indexom 12 Usual Suspects, The (1995)
Procijenjena ocjena 8.7  za film s indexom 127 Godfather, The (1972)
Procijenjena ocjena 8.7  za film s indexom 172 Empire Strikes Back, The (1980)
Procijenjena ocjena 8.7  za film s indexom 100 Fargo (1996)
Procijenjena ocjena 8.7  za film s indexom 408 Close Shave, A (1995)
Procijenjena ocjena 8.7  za film s indexom 313 Titanic (1997)
Procijenjena ocjena 8.6  za film s indexom 169 Wrong Trousers, The (1993)
Procijenjena ocjena 8.6  za film s indexom 181 Return of the Jedi (1983)
Procijenjena ocjena 8.6  za film s indexom 318 Schindler's List (1993)
Procijenjena ocjena 8.6  za film s indexom 56 Pulp Fiction (1994)
Procijenjena ocjena 8.5  za film s indexom 114 Wallace & Gromit: The Best of Aardman Animation (1996)
Procijenjena ocjena 8.5  za film s indexom 195 Terminator, The (1984)
Procijenjena ocjena 8.4  za film s indexom 272 Good Will Hunting (1997)
Procijenjena ocjena 8.4  za film s indexom 89 Blade Runner (1982)
Procijenjena ocjena 8.4  za film s indexom 96 Terminator 2: Judgment Day (1991)
Procijenjena ocjena 8.4  za film s indexom 302 L.A. Confidential (1997)
Procijenjena ocjena 8.4  za film s indexom 357 One Flew Over the Cuckoo's Nest (1975)
Procijenjena ocjena 8.3  za film s indexom 483 Casablanca (1942)
```

Slika 11 - preporuke filmova

Mogu reći da sam zadovoljan preporukom koju sam dobio od strane sustava – neke od preporučenih filmova sam gledao prije i svidjeli su mi se, dok sam o ostalim filmovima pročitao na stranici IMDb.com te pogledao kratki film koji ih predstavlja. Prema ovome, to jest ovoj metodi koja je ponovljena više puta, moglo bi se reći da implementirani sustav za davanje preporuka dobro funkcionira.

Sljedeće dvije metode koje koristimo pri evaluaciji sustava su računanje srednje vrijednost sume kvadrata apsolutne vrijednosti razlike pretpostavljene i stvarne vrijednosti (MAE) kao i računanje srednje vrijednosti korijena kvadrata razlike (eng. Root Mean Squared Error (RMSE)) stvarne i pretpostavljene vrijednosti. Slika 12 prikazuje kod kojim se izračunava vrijednost MAE i RMSE kao i rezultate navedenih metoda evaluacije:

```
X = parametri_finalni[:br_filmova*br_atributa].reshape(br_filmova,br_atributa)
Theta = parametri_finalni[br_filmova*br_atributa:].reshape(br_korisnika,br_atributa)

# Procijeni ocjene
procjene_ocjena = X @ Theta.T
moje_ocjene = procjene_ocjena[:,0][:,np.newaxis] + Y_prosijek

rmse = np.sqrt(np.mean((procjene_ocjena-Y)**2))
print("Iznos korijena srednje kvadratne pogreške (RMSE) je : ", round(rmse,2))
mae = np.sum(np.absolute(procjene_ocjena - Y))
print("Iznos srednje apsolutne pogreške (MAE) je: ", round((mae)/(procjene_ocjena.shape[1]*procjene_ocjena.shape
```

```
Iznos korijena srednje kvadratne pogreške (RMSE) je :  2.36
Iznos srednje apsolutne pogreške (MAE) je:  2.1
```

Slika 12 - RMSE i MAE za procijenjene ocjene filmova

Pošto prognoziramo ocjene filmova, bili bismo zadovoljni ako je razlika pretpostavljane i stvarne ocjene približno 0.5, ili 1. Ali ako griješimo 2 ocjene manje ili više u prosjeku ocjenjivanja, onda baš i ne bismo bili zadovoljni. Pošto su rezultati MAE i RMSE (obje metode dakle pokazuju procjene pri greškama, samo što RMSE penalizira veće greške pa s toga dava već rezultat) veći od

2 nakon što se više puta obavila evaluacija, ne možemo reći da bismo prema rezultatima MAE i RMSE bili zadovoljni sustavom. No, da bi rekli da je neki sustav za davanje preporuka dobar ili loš, ne možemo se samo osloniti na rezultate MAE i RMSE. Spomenuto je u uvodu ovog rada da je glavni problem sustava za davanje preporuka problem dugog repa, problem u kojem bi sustavi trebali preporučiti određene objekte koji su relevantni i koji odgovaraju specifične interesima pojedinog korisnika. Također, trendovi preferencija se mijenjaju pa bi se shodno tome trebale promjene preferencija zabilježiti u sustavu – pošto je implementirani sustav za davanje preporuka kolaboracijskog tipa, onda će se te promjene preferencija zabilježiti od strane sustava pa će se shodno tome i davanje preporuka promijeniti (kao i riješiti problem dugog repa pošto sustav sam pravi profil korisnika). Dakle, prema navedenom, ipak se ne bi reklo da je sustav loš - pošto sustav rješava probleme novog korisnika, prati određene promjene trendova, a usput, u slučaju prve metode evaluacije sustav je u više navrata dao dobre preporuke (velika većina preporučenih filmova je odgovarala mojim interesima) - moglo bi se reći da je sustav dobar, pogotovo ako se još uzme u obzir da je zbog regularizacije (da bi se izbjegao problem pretjeranog prilagođavanja podacima) postignuta nešto veća vrijednost RMSE i MAE. Također, bitno je naglasiti da bi se svakako korištenjem određenih naprednijih algoritama kao na primjer neuralnih mreža mogla postići bolja procjena ocjene filma, a zbog toga i manje vrijednosti RMSE I MAE.

6. Zaključak

Ovaj završni rad se bavi sustavima za davanje preporuka. U samom početku rada je opisan glavni problem i razlog zbog kojeg su sustavi za davanje preporuka nužni u današnjem svijetu – problem dugog repa - sve se više određena potražnja proizvoda odvija online, online sustavi nemaju problema sa smještajem proizvoda za prodaju pa je problem koji nastaje kako doći do specifičnih interesa koji odgovaraju pojedinom korisniku.

Sustave za davanje preporuke dijelimo na sustave čija se preporuka temelju na sadržaju, sustave čija se preporuka temeljeni na kolaboraciji te na hibridne sustave – sustave koji nastaju kada se preporuka temelji na kolaboraciji, kao i sadržaju. Kroz rad su opisane prednosti i nedostatci sustava za preporučivanje ovisno o tome na čemu se temelji preporuka. Ne možemo izričito reći da su neki tipovi sustava bolji od drugih, ali pošto se kolaboracijski sustavi sve više koriste, mogli bismo reći da su kolaboracijski sustavi pogodniji za korištenje u generalnim slučajevima (dokaz toga je i sve veća korištenost kolaboracijskih sustava u svijetu).

Dublje se ulazi u područje strojnog učenja, pošto je kolegij kojem pripada ovaj završni rad strojno učenje. Nakon objašnjenja algoritma linearne regresije kao i gradijentnog spusta, objašnjavaju se određeni problemi kao na primjer problem pretjeranog (ne)prilagođavanja podacima iz skupa podataka koji služe za učenje modela, kao i problem raznolikih vrijednosti atributa koje otežavaju rad gradijentnog spusta.

Znanje o linearnoj regresiji i gradijentom spustu objašnjava se na primjeru sustava za preporučivanje filmova. Navedeni algoritmi pomažu kako bi se prvo objasnilo, a kasnije i implementirao sustav koji je baziran na kolaboraciji. Implementirani sustav rješava problem novog korisnika, dugog repa, a može i procijeniti ocjenu filma (koja služi za davanje preporuka), kao i sličnosti između filmova. Objašnjena je i evaluacija implementiranog sustava te je na kraju zaključeno da sustav nije loš, ali da bi bio implementiran u stvarnosti vjerojatno bi trebao doživjeti određene promjene.

Zanimljivo je da se iskorišteno rješenje sustava za preporučivanje filmova može iskoristiti kao generalno rješenje za bilo koji sustav koji preporučuje objekt koji se opisuje pomoću numeričkih podataka.

Navedeno znanje koje se steklo radeći ovaj rad može se iskoristiti kao sredstvo za učenje nekih kompliciranijih algoritama strojnog učenja, a također može i poslužiti kao početna točka za ulaženje dublje u problematiku sustava za davanje preporuka.

7. Literatura

[1] [Online]. Available: https://stats.stackexchange.com/questions/131267/how-to-interpret-error-measures.

[2] J. Frost, "https://statisticsbyjim.com," [Online]. Available: https://statisticsbyjim.com/regression/overfitting-regression-models/.

[3] Leskovec Jure, Rajaraman Anand and jef Ullman, "http://infolab.stanford.edu".

[4] »https://github.com/,« [Mrežno]. Available: https://github.com/yanfei-wu/ml-ng.

[5] Folasade Isinkaye, Yetunde Folajimi, Bolanle Ojokoh, »Recommendation systems: Principles, methods and evaluation«.

[6] "Stackoverflow," [Online]. Available: https://stackoverflow.com/.

[7] J. Brownlee, "https://machinelearningmastery.com," [Online]. Available: https://machinelearningmastery.com/what-is-machine-learning/.

[8] Z. Damijanić, "https://repozitorij.pmf.unizg.hr," [Online]. Available: https://repozitorij.pmf.unizg.hr/islandora/object/pmf%3A4555.

[9] S. Das, "https://www.analyticsvidhya.com," [Online]. Available: https://www.analyticsvidhya.com/blog/2015/08/beginners-guide-learn-content-based-recommender-systems/.

[10] B. Lau, "https://towardsdatascience.com," [Online]. Available: https://towardsdatascience.com/andrew-ngs-machine-learning-course-in-python-anomaly-detection-1233d23dba95.

[11] A. Ng, "https://www.coursera.org/," [Online]. Available: https://www.coursera.org/learn/machine-learning/.

[12] C. Pinela, "https://medium.com," [Online]. Available: https://medium.com/@cfpinela/how-to-evaluate-recommender-systems-874a313d7724.

[13] L. Zhang, "The definition of Novelty in Recommendation System," *JOURNAL OF Enigneering Science and Technology Review.*

8. Prilozi

8.1 Popis tablica

8.2 Popis slika

IZJAVA

Izjavljujem pod punom moralnom odgovornošću da sam završni rad izradio samostalno, isključivo znanjem stečenim na studijima Sveučilišta u Dubrovniku, služeći se navedenim izvorima podataka i uz stručno vodstvo izv.prof.dr.sc Maria Miličevića, kome se još jednom srdačno zahvaljujem.

Luka Krnetić