Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
Fakulteta za matematiko in fiziko

Demian Bucik

Primerjava latentnih modelov za priporočilne sisteme v strojnem učenju

diplomsko delo

interdisciplinarni univerzitetni
študijski program prve stopnje
računalništvo in matematika

Mentor: prof. dr. Zoran Bosnić

Ljubljana, 2019

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V diplomskem delu naj kandidat primerja uporabo matrične faktorizacije in Boltzmannovih strojev za priporočanje v priporočilnih sistemih. Pri tem naj v uvodu dela opravi pregled ustreznih teoretičnih osnov strojnega učenja in naj predlaga ustrezno prilagoditev Boltzmannovega stroja za namen izvedbe naloge priporočanja. Oba modela naj primerja na dveh realnih podatkovnih množicah ter evalvira uspešnost rezultatov.

Zahvaljujem se vsem profesorjem, ki se trudijo in izboljšujejo kvaliteto študija.

Kazalo

Seznam uporabljene notacije

a	Skalar
$\boldsymbol{a}$	Vektor
$\boldsymbol{A}$	Matrika
$\boldsymbol{I}$	Identiteta (matrika)
a	Slučajna spremenljivka
$\mathbf{a}$	Slučajni vektor
$[a, b)$	Interval realnih števil od vključno a do b, brez b
$\mathcal{G}$	Graf
$Pa_{\mathcal{G}}(\mathrm{x}_i)$	Starši vozlišča x_i v grafu $\mathcal{G}$
a_i	Element i vektorja $\boldsymbol{a}$
a_{ij}	Element matrike $\boldsymbol{A}$ v i-ti vrstici in j-tem stolpcu
$\boldsymbol{A}_{i,:}$	Vrstica i matrike $\boldsymbol{A}$
$\boldsymbol{A}^\top$	Transponirana matrika
$\boldsymbol{a} \odot \boldsymbol{b}$	Produkt po komponentah (Hadamardov produkt)
$p(x)$	Gostota v zveznem primeru oziroma verjetnost v diskretnem
$\mathrm{x} \sim p(\mathrm{x})$	Slučajna spremenljivka x je porazdeljena po $p(\mathrm{x})$
$\mathbb{E}_{\mathrm{x}\sim p(\mathrm{x})}[\mathrm{x}]$ ali $\mathbb{E}\,\mathrm{x}$	Pričakovana vrednost x porazdeljene po $p(\mathrm{x})$
$\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$	Normalna porazdelitev (ali gostota) v spremenljivki $\boldsymbol{x}$ s pričakovano vrednostjo $\boldsymbol{\mu}$ in kovariančno matriko $\boldsymbol{\Sigma}$
$f'(x)$	Odvod funkcije f po x
$\dfrac{\partial f}{\partial \boldsymbol{x}}$	Parcialni odvod f po $\boldsymbol{x}$ (vektor z $\frac{\partial f}{\partial x_i}$ na i-tem mestu)
$\nabla_{\boldsymbol{x}} f$	Gradient f po $\boldsymbol{x}$
$\lVert \boldsymbol{x} \rVert$	2-norma vektorja $\boldsymbol{x}$
$\mathbf{1}_{pogoj}$	Indikator, ki ima vrednost 1, če je pogoj resničen, in 0 sicer

Povzetek

Naslov: Primerjava latentnih modelov za priporočilne sisteme v strojnem učenju

Avtor: Demian Bucik

Vseprisotnost priporočilnih sistemov v digitalnem ekosistemu korenito spreminja način uporabe spletnih storitev. V poplavi informacij se uporabniki za dostop do zanimivih in koristnih vsebin zanašajo na njihova priporočila. Cilj te diplomske naloge je predstaviti dve vrsti priporočilnih sistemov, ki temeljijo na skupinskem filtriranju. Pri konstrukciji modelov se opremo na določene koncepte strojnega učenja. Modele implementiramo v okolju Python in evalviramo pri napovedovanju števila poslušanj izvajalcev in uporabnikovih ocen vsebine, za kar uporabimo dve podatkovni množici. Ob tem predlagamo način razširitve omejenega Boltzmannovega stroja za priporočanje in modeliranje zveznih ocen ter uporabimo dodatek modelu na osnovi matričnega razcepa, ki omogoča modeliranje prijateljstev med uporabniki. Rezultati nakazujejo, da modeli na osnovi matričnega razcepa delujejo občutno bolje in da se splača več časa investirati v njihov nadaljnji razvoj.

Ključne besede: skupinsko filtriranje, strojno učenje, matrična faktorizacija, omejen Boltzmannov stroj, metoda največje aposteriorne verjetnosti.

Abstract

Title: Comparison of latent models for recommender systems in machine learning

Author: Demian Bucik

The ubiquity of recommender systems in the digital ecosystem is thoroughly changing the way web services are used. In the ceaseless stream of information, users are relying on their recommendations to access interesting and useful content. The aim of this thesis is to present two approaches to collaborative filtering based recommender systems. Developing the models, we borrow certain concepts from the field of machine learning. All models are implemented in Python and evaluated on two datasets, one containing the numbers of plays and the other containing users' ratings of content. We propose a way to model continuous ratings with restricted Boltzmann machine and apply an addition to the matrix factorization model that makes it possible to model friendships between users. The results suggest that matrix factorization models work considerably better and that more time should be invested in their further development.

Keywords: collaborative filtering, machine learning, matrix factorization, restricted Boltzmann machine, maximum a posteriori estimation.

Poglavje 1

Uvod

Z besedo priporočilni sistemi (angl. *recommender systems*) označujemo sisteme oziroma algoritme, ki skušajo napovedati uporabnikovo preferenco glede nekega objekta. Ti objekti so lahko filmi, glasba oziroma celotni seznami predvajanja, posamezne novice, izdelki v spletnih trgovinah, učna gradiva... V grobem se algoritmi za priporočanje delijo na dva tipa. Prva vrsta so metode na osnovi skupinskega filtriranja (angl. *collaborative filtering*, nekateri to prevajajo tudi kot metoda izbiranja s sodelovanjem). Drugi tip pa so vsebinsko osnovani sistemi (angl *content-based filtering*).

Metode skupinskega filtriranja delujejo pod predpostavko, da ljudje, ki so se strinjali v preteklosti in imeli radi podobne objekte, se bodo strinjali tudi v prihodnosti. Ključna prednost tu je, da ni potrebno razumevanje kompleksnih objektov, da bi jih lahko priporočali. Vsebinsko osnovane metode pa temeljijo ravno na razumevanju objektov, torej izgradnji opisov objektov in uporabnikov na podlagi njihovih lastnosti. Ta pristop ima svoje korenine v področju iskanja informacij (angl. *information retrieval*).

Priporočilni sistemi ponujajo uporabnikom personalizirana priporočila vsebine ali izdelkov. Uporabniki se za dostop do kvalitetne vsebine v poplavi informacij na njih zanašajo, za podjetja pa predstavljajo eno glavnih komponent za uspešnost na svetovnem spletu. Zaradi masovnega zbiranja podatkov postajajo takšni sistemi vse inteligentnejši in boljši pri priporočanju. Priporočilni sistemi so vroče raziskovalno področje, ki je dobilo veliko pozornosti in pomembnih odkritij po gostovanju tekmovanja v napovedovanju ocen filmov s strani podjetja Netflix. V tej

diplomski nalogi se posvetimo napovedovanju števila predvajanj glasbe določenega izvajalca in napovedovanju ocen določenega filma s strani posameznega uporabnika. V osrednjih poglavjih predstavimo dva pristopa za napovedovanje uporabnikovih preferenc in ponujanje personalizirane vsebine. Pri konstrukciji modelov pa se opremo na nekaj teoretičnih osnov strojnega učenja (angl. *machine learning*), ki jih predstavimo v uvodnem poglavju.

V diplomski nalogi se posvetimo samo sistemom na osnovi skupinskega filtriranja. Natančneje, sistemom, ki za uporabnike in objekte v latentnem prostoru sestavijo nek profil (vektor), v katerem pomena posameznih elementov ne bomo poznali, ki nam bo kljub temu koristil za napovedovanje. Izbrali in preučili bomo dva načina za izgradnjo latentnih profilov in napovedovanje ocen. Prva oblika modelov temelji na matričnem razcepu, druga pa na osnovi omejenih Boltzmannovih strojev (angl. *restricted Boltzmann machine, RBM).* Pri modelih na osnovi matričnega razcepa uporabimo več dopolnitev algoritmov, na primer upoštevanje prijateljstev med uporabniki za natančnejše napovedovanje. Pri drugem tipu modelov pa predlagamo drugačno obliko modela in učnega algoritma od splošno uporabljene [23].

V okviru te naloge bi radi preverili, če se modeli RBM lahko kosajo z uspešnimi modeli na osnovi matrične faktorizacije. V članku [23] avtorjem po številnih vpeljanih izboljšavah navsezadnje uspe doseči malenkost boljši rezultat od osnovnega modela matrične faktorizacije. Zadovoljili se bomo že, če se bo model izkazal za kompetitivnega in uporabnega. Radi bi odgovorili na vprašanje, katera vrsta modelov je bolj perspektivna za uporabo v praksi in kateri vrsti modelov se splača po zaključku diplome še namenjati čas.

V prvem delu diplomske naloge uvedemo nekaj osnovnih pojmov strojnega učenja in statistike, iz katerih kasneje izhajamo pri konstrukciji modelov za priporočanje. V drugem delu predstavimo podatke in uporabljene metrike. Metode ovrednotimo na dveh podatkovnih množicah in vizualiziramo rezultate. Implementacija metod v okolju Python z uporabo knižnjic NumPy [19] in SciPy [12] je na voljo na `https://github.com/demianbucik/collaborative-filtering-recommender-systems`.

Poglavje 2

Osnove uporabljenih metod strojnega učenja

Ena od nalog strojnega učenja in statističnega modeliranja je zajem odvisnosti med spremenljivkami. Z modelom, ki zajema te odvistnosti, lahko, ob znanih vrednostih nekaterih spremenljivk, odgovarjamo na vprašanja o neznanih spremenljivkah.

Dve področji strojnega učenja, ki bosta obravnavani v sklopu te diplomske naloge, sta nadzorovano (angl. *supervised*) in nenadzorovano učenje (angl. *unsupervised learning*). Pri nadzorovanem učenju bi radi na osnovi označenih učnih primerov zgradili model, ki bo na novih, še nevidenih podatkih znal napovedati ciljno spremenljivko. Podatkovna množica, ki jo označimo z $\mathbb{X}$ ima obliko $\{(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}), (\boldsymbol{x}^{(2)}, \boldsymbol{y}^{(2)}), ..., (\boldsymbol{x}^{(m)}, \boldsymbol{y}^{(m)})\}$, kjer $\boldsymbol{x}$ predstavlja neodvisno spremenljivko, $\boldsymbol{y}$ pa pripadajočo ciljno spremenljivko. Na podlagi podatkovne množice $\mathbb{X}$ se model izuri v napovedovanju $\boldsymbol{y}$, ko mu pokažemo pripadajoči $\boldsymbol{x}$. Pri nenadzorovanem učenju naši učni primeri $\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, ..., \boldsymbol{x}^{(m)}\}$ niso označeni. Cilji algoritma so lahko zelo različni, na primer generiranje novih primerov $\boldsymbol{x}$ ali pa odkrivanje skupin.

Algoritmi strojnega učenja si navadno zgradijo matematičen model podatkov, ki ga nato uporabijo za napovedovanje. Predpostavimo, da nam podatke generira neka nepoznana porazdelitev $p_{data}(\mathbf{x})$. Označimo s $\hat{p}_{data}(\mathbf{x})$ empirično porazdelitev podatkov, definirano z $\mathbb{X}$, ki nam bo služila kot aproksimacije prave porazdelitve podatkov. Definirajmo še naš model kot družino porazdelitev $p_{model}(\mathbf{x}; \boldsymbol{\theta})$, para-

metriziranih s $\boldsymbol{\theta}$.

Ideja učnega algoritma je, da iz družine porazdelitev $p_{model}(\mathbf{x};\boldsymbol{\theta})$ izbere tisto, ki je najbližja empirični porazdelitvi $\hat{p}_{data}(\mathbf{x})$. Izbor porazdelitve tu dosežemo z izbiranjem parametra $\boldsymbol{\theta}$.

2.1 Kullback-Leiblerjeva divergenca

Da bi določili porazdelitev $p_{model}(\mathbf{x};\boldsymbol{\theta})$, ki je najbližja $\hat{p}_{data}(\mathbf{x})$, moramo najprej izbrati način merjenja podobnosti med porazdelitvama. Tu imamo več možnosti. Odločimo se lahko za popularno mero D_{KL} imenovano Kullback-Leiblerjeva divergenca (tudi relativna entropija) [9, 2]. Definirana je kot

$$D_{KL}(p \parallel q) = \mathbb{E}_{\mathrm{x}\sim p(\mathrm{x})}\left[\log \frac{p(\mathrm{x})}{q(\mathrm{x})}\right], \tag{2.1}$$

kjer $\mathbb{E}_{\mathrm{x}\sim p(\mathrm{x})}$ predstavlja pričakovano vrednost, ko je slučajna spremenljivka x porazdeljena po p, p in q pa sta poljubni verjetnostni porazdelitvi. Pri diskretni slučajni spremenljivki x se pričakovana vrednost izraža z vsoto, pri zvezni pa z integralom. Intuitivno, $D_{KL}(p \parallel q)$ predstavlja količino informacije, ki jo izgubimo, če porazdelitev p aproksimiramo s q. Ima dve pomembni lastnosti, ki sledita iz Jensenove neenakosti

1. $D_{KL}(p \parallel q) \geq 0$,

2. $D_{KL}(p \parallel q) = 0$ natanko tedaj, ko $p = q$.

Ne gre torej za pravo metriko oziroma razdaljo, ker $D_{KL}(p \parallel q) \neq D_{KL}(q \parallel p)$, ravno tako pa ne velja niti trikotniška neenakost.

Za vse učne primere predpostavljamo, da so, če poznamo $\boldsymbol{\theta}$, med seboj neodvisni in enako porazdeljeni (angl. *independent and identically distributed, i.i.d.*).

Zapišimo sedaj optimalno izbiro $\boldsymbol{\theta}$

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \; D_{KL}(\hat{p}_{data} \parallel p_{model}) \tag{2.2}$$

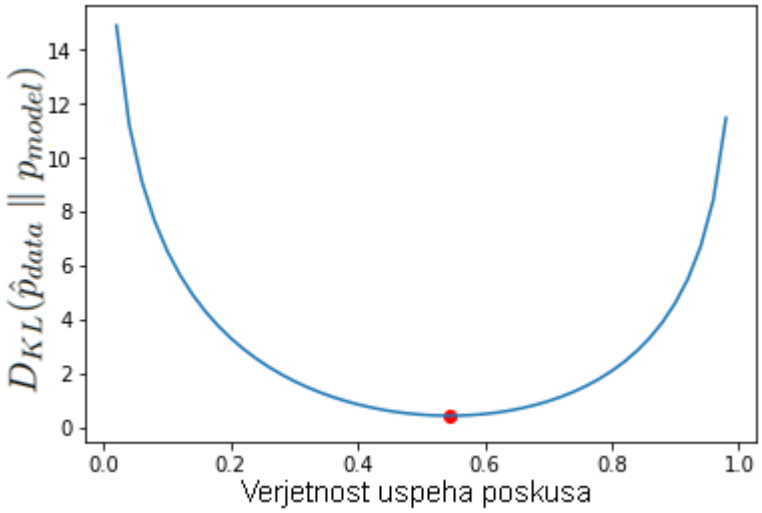
$$= \arg\min_{\boldsymbol{\theta}} \; \mathbb{E}_{\mathbf{x}\sim\hat{p}_{data}(\mathrm{x})}\left[\log \frac{\hat{p}_{data}(\mathbf{x})}{p_{model}(\mathbf{x};\boldsymbol{\theta})}\right] \tag{2.3}$$

$$= \arg\min_{\boldsymbol{\theta}} \; \frac{1}{m}\sum_{j=1}^{m} \log \hat{p}_{data}(\boldsymbol{x}^{(j)}) - \frac{1}{m}\sum_{j=1}^{m} \log p_{model}(\boldsymbol{x}^{(j)};\boldsymbol{\theta}) \tag{2.4}$$

$$= \arg\max_{\boldsymbol{\theta}} \; \sum_{j=1}^{m} \log p_{model}(\boldsymbol{x}^{(j)};\boldsymbol{\theta}) \tag{2.5}$$

$$= \arg\max_{\boldsymbol{\theta}} \; \prod_{j=1}^{m} p_{model}(\boldsymbol{x}^{(j)};\boldsymbol{\theta}). \tag{2.6}$$

Rezultat iz točke (2.6) bomo razložili v naslednjem poglavju. Vse izpeljano velja tudi za drugačne modele, na primer $p_{model}(\mathbf{y} \mid \mathbf{x};\boldsymbol{\theta})$, vendar zaradi enostavnejše notacije računamo s $p_{model}(\mathbf{x};\boldsymbol{\theta})$.



Slika 2.1: Vizualizacija $D_{KL}(\hat{p}_{data} \parallel p_{model})$ [7]. Iz podatkov želimo oceniti parameter θ, ki predstavlja verjetnost, da posamezen Bernoullijev poskus uspe. Na sliki je prikazana vrednost, ki minimnizira Kullback-Leiblerjevo divergenco, ki ustreza vzorčnemu povprečju.

Sedaj še vizualiziramo $D_{KL}(\hat{p}_{data} \parallel p_{model})$ (glej sliko 2.1), kjer je $\hat{p}_{data}(\mathbf{x})$ empirična porazdelitev, $\mathrm{x} \in \{0, 1\}$, $p_{model}(\mathbf{x};\theta)$ pa družina binomskih porazdelitev

in θ verjetnost, da poskus uspe [7]. Vrednost θ, ki minimizira $D_{KL}(\hat{p}_{data} \parallel p_{model})$ je 0,544, kar je intuitivno smiselno, saj to predstavlja ravno vzorčno povprečje podatkov.

Obe obliki priporočilnih sistemov, ki jih predstavimo v diplomski nalogi, bosta temeljili na optimizaciji funkcije, katere glavni del bo predstavljal izraz (2.5). Določanju parametrov po tem izrazu pravimo tudi metoda največjega verjetja, ki je ena najpogosteje uporabljenih metod v nadzorovanem učenju, razložili pa jo bomo v naslednjem poglavju.

2.2 Metoda največjega verjetja

Določanju parametrov $\boldsymbol{\theta}$ po enačbi (2.6) pravimo metoda največjega verjetja (angl. *maximum likelihood estimation, MLE*) [16]. Definirajmo funkcijo verjetja

$$\mathcal{L}(\boldsymbol{\theta}; \mathbb{X}) = p_{model}(\mathbb{X}; \boldsymbol{\theta}) \tag{2.7}$$

$$= \prod_{j=1}^{m} p_{model}(\boldsymbol{x}^{(j)}; \boldsymbol{\theta}), \tag{2.8}$$

kjer je $p_{model}(\mathbb{X}; \boldsymbol{\theta})$ gostota porazdelitve, specificirane z modelom, izračunane na celotni podatkovni množici $\mathbb{X}$; zaradi predpostavke o pogojni neodvisnosti se izraža kot produkt posameznih gostot po učnih primerih $\boldsymbol{x}^{(j)}$. Definirajmo še logaritem verjetja

$$\ell(\boldsymbol{\theta}; \mathbb{X}) = \log \mathcal{L}(\boldsymbol{\theta}; \mathbb{X}) \tag{2.9}$$

$$= \sum_{j=1}^{m} \log p_{model}(\boldsymbol{x}^{(j)}; \boldsymbol{\theta}). \tag{2.10}$$

Metoda največjega verjetja se torej glasi

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \ \mathcal{L}(\boldsymbol{\theta}; \mathbb{X}) = \arg\max_{\boldsymbol{\theta}} \ \ell(\boldsymbol{\theta}; \mathbb{X}), \tag{2.11}$$

torej za oceno parametra $\boldsymbol{\theta}$ izberemo vrednost, ki maksimizira verjetje. V praksi navadno maksimiziramo logaritem verjetja v izogib numeričnim napakam.

Intuitivno je metoda največjega verjetja smiselna. Iz družine porazdelitev $p_{model}(\mathbf{x}; \boldsymbol{\theta})$ izberemo tisto, za katero je verjetnost, da nam je generirala podatke, največja. Z drugimi besedami, izberemo tak $\hat{\boldsymbol{\theta}}$, da je verjetnost podatkov največja.

Za napoved novega primera vstavimo dobljeno vrednost v porazdelitev določeno z modelom $p_{model}(\boldsymbol{x}; \hat{\boldsymbol{\theta}})$.

Kot vidimo, je iskanje parametra, ki maksimizira verjetje, ekvivalentno iskanju parametra, ki minimizira Kullback-Leiblerjevo divergenco med empirično porazdelitvijo in družino porazdelitev specificiranih z modelom [16].

Glavna definicija, vpeljana v tem razdelku, je funkcija logaritma verjetja, ki bo igrala osrednjo vlogo v obeh vrstah priporočilnih sistemov v tej diplomski nalogi.

2.3 Bayesovski pogled

V prejšnjih poglavijh je bilo pomembno poudariti razliko med empirično porazdelitvijo in modelom. Od tu naprej, zaradi enostavnejše notacije, pišemo kar p namesto p_{model}.

Do sedaj smo $\boldsymbol{\theta}$ obravnavali kot deterministično, konstanten, a neznan parameter. V Bayesovi statistiki pa na $\boldsymbol{\theta}$ gledamo kot na slučajno spremenljivko in z verjetnostjo izražamo svojo negotovost (oziroma gotovost) o njeni vrednosti. Namesto točkaste ocene parametra, ki maksimizira verjetnost podatkov, tu vrednosti $\boldsymbol{\theta}$ opišemo z verjetnostno porazdelitvijo [16].

Označimo s $p(\boldsymbol{\theta})$ apriorno znanje o vrednosti $\boldsymbol{\theta}$. Torej znanje, preden opazimo podatke. Ko opazimo podatke $\mathbb{X}$, pa bi radi svoje znanje o $\boldsymbol{\theta}$ primerno popravili. Optimalen način za posodobitev znanja nam podaja Bayesova formula, izpeljana enostavno z dvakratno uporabo definicije pogojne verjetnosti. Posodobitev znanja se torej glasi

$$p(\boldsymbol{\theta} \,|\, \mathbb{X}) = \frac{p(\mathbb{X} \,|\, \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(\mathbb{X})} = \frac{p(\mathbb{X} \,|\, \boldsymbol{\theta}) p(\boldsymbol{\theta})}{\int_{\boldsymbol{\Theta}} p(\mathbb{X} \,|\, \boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta}}. \tag{2.12}$$

Napoved za novi primer $\boldsymbol{x}$ ravno tako izrazimo s porazdelitvijo. Tako za napoved uporabimo celo aposteriorno porazdelitev $\boldsymbol{\theta}$ namesto same točkaste ocene [16]

$$p(\boldsymbol{x} \,|\, \mathbb{X}) = \int_{\boldsymbol{\Theta}} p(\boldsymbol{x}, \boldsymbol{\theta} \,|\, \mathbb{X}) d\boldsymbol{\theta} \tag{2.13}$$

$$= \int_{\boldsymbol{\Theta}} p(\boldsymbol{x} \,|\, \boldsymbol{\theta}) p(\boldsymbol{\theta} \,|\, \mathbb{X}) d\boldsymbol{\theta}. \tag{2.14}$$

To je optimalen način napovedovanja, vendar v veliko primerih za aposteriorno porazdelev $\boldsymbol{\theta}$ in aposteriorno napovedno porazdelitev nimamo rešitve zaprte oblike.

Zato se poslužujemo raznih tehnik za približni izračun, naj si bodo to strukturne aproksimacije ali pa različni načini vzorčenja. To je izjemno zanimivo področje, vendar se z njim v tej diplomski nalogi ne bomo ukvarjali.

Ubrali bomo neko srednjo pot, parameter $\boldsymbol{\theta}$ bomo točkasto ocenili, ob tem pa zanj vseeno uporabili neko apriorno porazdelitev. To nas pripelje do naslednje metode.

2.4 Metoda največje aposteriorne verjetnosti

Kot že ime pove, metoda največje aposteriorne verjetnosti (angl. *maximum a posteriori estimation, MAP*) narekuje, da $\boldsymbol{\theta}$ točkasto ocenimo iz njene aposteriorne porazdelitve [16], ob tem pa predpostavimo apriorno porazdelitev $p(\boldsymbol{\theta})$

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \; p(\boldsymbol{\theta} \,|\, \mathbb{X}) \tag{2.15}$$

$$= \arg\max_{\boldsymbol{\theta}} \; p(\mathbb{X} \,|\, \boldsymbol{\theta}) p(\boldsymbol{\theta}) \tag{2.16}$$

$$= \arg\max_{\boldsymbol{\theta}} \; \ell(\boldsymbol{\theta}; \mathbb{X}) + \log p(\boldsymbol{\theta}). \tag{2.17}$$

V zadnji vrstici smo izraz logaritmirali, kar nam bo kasneje prišlo prav. Kot vidimo, je tu kriterijska funkcija sestavljena iz vsote logaritma verjetja in logaritma apriorne porazdelitve $\boldsymbol{\theta}$. Prvi člen nam je že znan, drugi pa, s pravo izbiro apriorne porazdelitve, nadzoruje kompleksnost modela in s tem preprečuje preveliko prileganje učnim podatkom. Velikokrat se za apriorno porazdelitev uporabi kar normalna porazdelitev $\mathcal{N}(\boldsymbol{\theta} \,|\, \mathbf{0}, \sigma^2 \boldsymbol{I})$, kjer je σ^2 parameter, ki ga nastavimo na primer z uporabo prečnega preverjanja. Pri polni Bayesovski obravnavi pa lahko tudi za σ^2 uporabimo apriorno porazdelitev in nato po njej integriramo. Tako se izognemo ročnemu nastavljanju in tudi to določimo direktno iz podatkov.

Bolje kot najverjetnejšo točko (oziroma tako z največjo gostoto), bi bilo vzeti pričakovano vrednost ali mediano aposteriorne porazdelitve $\boldsymbol{\theta}$ za izbiro točkaste ocene. V praksi pa se navadno uporablja metoda MAP, ker se problem tako poenostavi na optimizacijo [16]. Ko imamo enkrat oceno $\hat{\boldsymbol{\theta}}$, za napovedi uporabimo $p(\boldsymbol{x} \,|\, \hat{\boldsymbol{\theta}})$, ki služi kot aproksimacija $p(\boldsymbol{x} \,|\, \mathbb{X})$. Če imamo neomejeno količino podatkov, se metoda MAP in Bayesovska obravnava približujeta rešitvi po metodi največjega verjetja.

Za konstrukcijo in določitev vrednosti parametrov vseh modelov za napovedovanje ocen, bomo v tej diplomski nalogi uporabili metodo največje aposteriorne verjetnosti.

2.5 Gradientni spust

V prejšnjih razdelkih smo omenjali optimizacijo funkcij. Gradientni spust (angl. *gradient descent*) je preprost algoritem, ki nam poišče parametre, ki minimizirajo dano funkcijo [9]. Če bi radi poiskali parametre, ki maksimizirajo neko funkcijo, lahko to seveda storimo z minimizacijo te iste funkcije, pomnožene z -1.

Denimo, da bi radi poiskali vrednost $\boldsymbol{\theta}^*$, pri kateri funkcija $J : \mathbb{R}^d \to \mathbb{R}$ doseže najmanjšo vrednost. Na primer, pri uporabi metode največjega verjetja ima funkcija J obliko

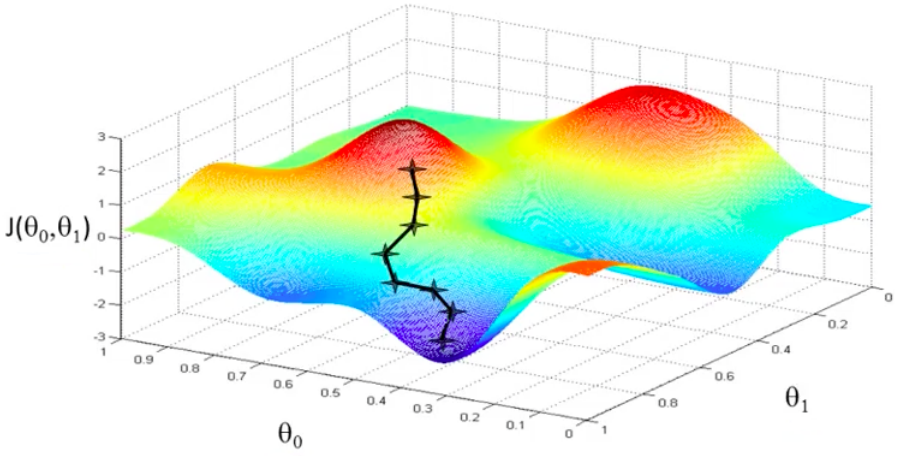
$$J(\boldsymbol{\theta}) = -\frac{1}{m}\sum_{j=1}^{m} \log p(\boldsymbol{x}^{(j)}; \boldsymbol{\theta}), \tag{2.18}$$

ki predstavlja negativno povprečno vrednost logaritma verjetja na učni množici, sestavljeni iz primerov $\boldsymbol{x}^{(j)}$, verjetje je parametrizirano s parametrom $\boldsymbol{\theta}$. Konstruiramo algoritem, ki začne pri neki vrednosti parametrov, naj bo to $\boldsymbol{\theta}^{(0)}$, nato pa se, korak za korakom, pomika v nasprotni smeri gradienta [9]. Posodobitev parametrov v koraku $k+1$ se glasi

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^{(k)}). \tag{2.19}$$

Vrednost α kontrolira velikost korakov in se imenuje stopnja učenja. Prikaz primera korakov gradientnega spusta je prikazan na sliki 2.2. Takšno posodabljanje parametrov ponavljamo do konvergence. Če je funkcija J konveksna, odvedljiva in je njen gradient Lipschitzov ter je α primerno majhna, bo, po dovoljšnjem številu korakov k, $\boldsymbol{\theta}^{(k)}$ poljubno blizu $\boldsymbol{\theta}^*$. Če funkcija ni konveksna, se bomo načeloma morali zadovoljiti z lokalnimi minimumi, v kolikor se seveda uspemo izogniti točkam, kot so sedla in platoji.

Osnovni algoritem lahko izboljšamo s številnimi dodatki. Denimo, da je naša kriterijska funkcija negativen logaritem verjetja. Izračun vrednosti gradienta take funkcije, ki je potreben v vsakem koraku optimizacije, vključuje seštevanje preko vseh učnih primerov, kar je lahko zelo časovno potratno. Poleg tega, natančnost



Slika 2.2: Prikaz korakov gradientnega spusta [17].

izračuna narašča sorazmerno s korenom števila uporabljenih primerov, porabljen čas pa linearno. V tem primeru lahko učno množico razdelimo na majhne skupine (angl. *mini-batches*). Vrednost gradienta v nekem koraku algoritma lahko aproksimiramo tako, da pri izračunu upoštevamo le eno od teh skupin. Popularen pristop predlaga celo uporabo skupin velikosti 1 za izračun približka. Ob tem lahko primere vzorčimo ali pa jih obiskujemo po vrsti (učno množico pred tem premešamo).

Ob uporabi majhnih skupin bomo tako za en korak porabili manj časa. Ta metoda ima še dodatno prednost. Ker na vsakem koraku gradient aproksimiramo in ker konvergenco preverjamo šele ob obhodu vseh podatkov, se včasih lahko celo izognemo nekaterim lokalnim minimumom.

Naslednja dopolnitev je dodatek zagona (angl. *momentum*) [9]. Navadno se uporablja s prej omenjeno modifikacijo. Označimo z $\boldsymbol{m}^{(k+1)}$ vrednost, ki jo bomo v koraku $k+1$ pomnožili s stopnjo učenja in uporabili za posodobitev parametrov. V tem primeru ima posodobitev obliko

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \alpha \boldsymbol{m}^{(k+1)}, \tag{2.20}$$

$\boldsymbol{m}^{(k+1)}$ pa izračunamo kot uteženo povprečje gradienta in posodobitve iz prejšnjega koraka [9]

$$\boldsymbol{m}^{(k+1)} = (1-\gamma)\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^{(k)}) + \gamma \boldsymbol{m}^{(k)}, \tag{2.21}$$

kjer je parameter $\gamma \in (0,1)$. Tako definiran zagon $\boldsymbol{m}^{(k+1)}$ trenutni popravek izračuna kot eksponentno uteženo povprečje vseh prejšnjih vrednosti gradientov

$$\boldsymbol{m}^{(k+1)} = (1-\gamma)\sum_{i=0}^{k}\gamma^i \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^{(k-i)}). \tag{2.22}$$

To formulo smo dobili z rekurzivnim vstavljanjem zagona iz prejšnjega koraka v (2.21).

Uporaba zagona je še posebej koristna pri uporabi majhnih skupin, saj ublaži oscilacije in zmanjša šum v aproksimaciji gradienta ter bolje sledi trendu iz prejšnjih korakov, podoben učinek ima tudi pri prečkanju sotesk (glej sliko 2.3). S tem pospeši konvergenco in se lažje izogne nekaterim lokalnim minimumom in platojem.

Slika 2.3: Levo so koraki gradientnega spusta v soteski. Desno so koraki ob uporabi zagona [8].

V vseh zgornjih algoritmih smo uporabljali stopnjo učenja α, ki jo je potrebno pred učenjem nastaviti. Če je premajhna, bo konvergenca zelo počasna oziroma lahko celo ne bomo prišli do minimuma. Po drugi strani, če je prevelika, bodo posledično tudi koraki preveliki in algoritem ravno tako ne bo konvergiral.

Namesto uporabe fiksne vredosti Ng [18] predlaga dva pristopa, ki stopnjo učenja čez čas zmanjšujeta. Algoritem na začetku z večjimi koraki lažje pride v okolico nekega minimuma, nato pa se mu z drobnejšimi koraki čim natančneje približa. Eksponentni upad stopnje učenja ima v koraku $k+1$ obliko

$$\alpha^{(k+1)} = \rho^k \alpha^{(0)}, \tag{2.23}$$

kjer je $\alpha^{(0)}$ začetna stopnja učenja in kjer navadno uporabimo $\rho \in [0{,}8, 1)$. Poznamo tudi linearni upad

$$\alpha^{(k+1)} = \frac{1}{1+\beta k}\alpha^{(0)}, \tag{2.24}$$

ob tem ponavadi izberemo $\beta \in (0, 2]$.

Velja omeniti, da obstajajo tudi bolj napredni algoritmi, kot sta Adam in RMSprop, ki dodelita vsakemu parametru θ_i svojo stopnjo učenja in jo tudi samodejno prilagajata med učenjem. Optimizacijske metode, kot so Newtonove in kvazi-Newtonove, pa se, kljub višjemu redu konvergence, v praksi redkeje uporabljajo zaradi časovnega in prostorskega računskega bremena povezanega s Hessejevo matriko, poleg tega pa se rade ustavijo v sedlih in lokalnih maksimumih, saj so v osnovi namenjene iskanju ničel [9].

Algoritmi, ki jih v diplomski nalogi uporabimo za iskanje parametrov modelov za napovedovanje ocen, bodo temeljili na na gradientnem spustu in opisanih razširitvah. Funkcija, ki jo bomo optimizirali, pa bo logaritem aposteriorne verjetnosti.

2.6 Zgodnje ustavljanje

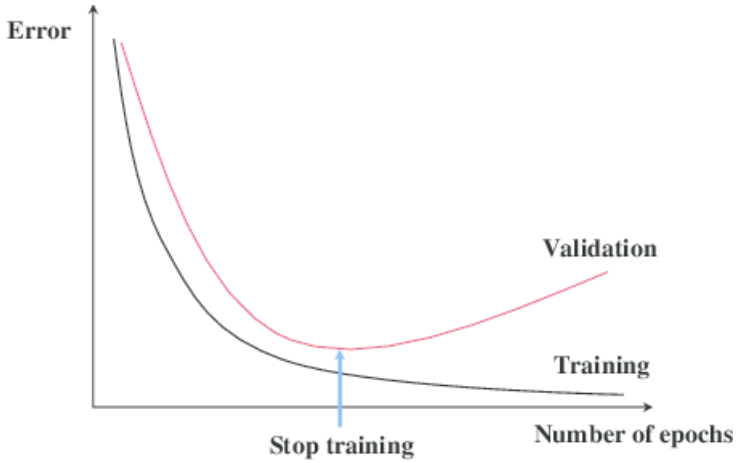
V prejšnjem poglavju smo opisali algoritem za optimizacijo funkcij. Vendar pa se strojno učenje razlikuje od čiste optimizacije [9]. Cilj matematične optimizacije je zgolj poiskati parametre $\boldsymbol{\theta}$, pri katerih kriterijska funkcija J doseže minimalno vrednost. Eden glavnih ciljev nadzorovanega učenja pa je uspešnost modela na novih, še nevidenih podatkih. Optimizacija kriterijske funkcije je tu le sredstvo za dosego tega.

Z minimizacijo kriterijske funkcije zmanjšujemo pričakovano napako modela na učni množici. V resnici pa želimo čim manjšo pričakovano napako pri posploševanju (angl. *generalization*). Tipično to napako ocenimo z uporabo testne množice. To so primeri, ki jih algoritmu med učenjem nismo pokazali.

Za dosego dobrega posploševanja je pomemba izbira modela s pravo kompleksnostjo [9]. Modeli z nizko kompleksnostjo se niso sposobni naučiti kompleksnih vzorcev iz podatkov (angl. *underfitting*). Po drugi strani, prekompleksni modeli imajo tendenco prevelikega prileganja učni množici (angl. *overfitting*). Kompleksnost modela lahko na primer reguliramo z izbiro pravega števila parametrov in uporabo regularizacije (apriornih verjetnosti).

Naslednji dejavnik, ki močno vliva na posploševanje, pa je pravočasno ustavljanje optimizacijskega algoritma [9]. Namesto da algoritem ustavimo ob prihodu

v minimum kriterijske funkcije, ga lahko ustavimo prej (glej sliko 2.4). V ta namen je pred učenjem koristno podatke razdeliti na učno in validacijsko množico. Kriterijska funkcija je v tem primeru sestavljena samo iz učnih podatkov. Ob vsakem obhodu podatkov izračunamo napako na validacijski množici in če se je ta povečala, optimizacijo prekinemo. Temu pravimo zgodnje ustavljanje (angl. *early stopping*). Ob tem pa moramo biti pazljivi, če naša funkcija ni konveksna. Kajti možno je, da napaka po več iteracijah povečevanja spet začne padati in doseže nižjo vrednost kot kadarkoli prej.



Slika 2.4: Prikaz ideje zgodnjega ustavljanja [14].

Za primerjavo različnih modelov je ravno tako primerno uporabiti testno množico ter na njej izmeriti želeno metriko. Res je, da velikokrat metrik, ki nas zares zanimajo, ni možno izmeriti na tak način. Na primer, metrika, ki nas zares zanima pri priporočanju video vsebin, je pričakovani porabljeni čas uporabnika na platformi. Veselje ali korist, ki jo imajo uporabniki od ogleda vsebin, je težko neposredno izmeriti, všečki oziroma ocene pa tega ne odražajo neposredno. Poleg tega pa je porabljeni čas ključen za poslovno uspešnost. Za njegovo merjenje je potrebno modele uporabiti v produkcijskem okolju in narediti test A/B. Kljub temu pa je uporaba metrik, kot so pričakovana napaka in točnost, na testnih množicah, dober indikator kvalitete modela.

Za oceno pričakovane napake pri posploševanju in primerjavo modelov je koristna uporaba prečnega preverjanja (angl. *cross validation*) [2]. Navadno nam, še

posebej ko nimamo ogromne količine podatkov, nudi natančnejšo oceno, kot uporaba ene same učne in testne množice. Izvedemo ga tako, da podatkovno množico razdelimo na k delov, nato pa k-krat naučimo model na $k-1$ delih in ga evalviramo na preostalem delu. Tako za evalvacijo uporabimo vsak del natanko enkrat. Na koncu kot oceno napake vzamemo povprečje vseh evalvacij. Prečnega preverjanja v taki obliki seveda ne smemo uporabljati, če imamo časovno odvisne podatke. Natančneje, z modelom ne smemo napovedovati na primerih, ki so časovno pred podatki, uporabljenimi za učenje.

2.7 Verjetnostni grafični modeli

V strojnem učenju in statističnem modeliranju igra verjetnost pomembno vlogo. V splošnem, če želimo predstaviti porazdelitev slučajnega vektorja $\mathbf{x}$ z n diskretnimi spremenljivkami, od katerih lahko vsaka zavzame k vrednosti, bi naivni pristop vključeval uporabo preglednice z zapisom verjetnosti za vsako od k^n možnih kombinacij. V praksi je tak pristop neizvedljiv, zato je pomembna drugačna formulacija modela, kjer specificiramo samo direktne interakcije med spremenljivkami [9].

Verjetnostni grafični modeli (angl. *probabilistic graphical models*, PGM) nam omogočajo izražanje interakcij med slučajnimi spremenljivkami [2, 9]. Porazdelitev opišemo z uporabo grafa, kjer vozlišča predstavljajo slučajne spremenljivke, direktne interakcije med njimi pa povezave. To nam omogoča enostaven način vizualizacije modela, ki nam nudi vpogled v njegove lastnosti. Olajšajo pa nam tudi konstrukcijo novih modelov in sklepanje. Z usmerjenim modelom si bomo pomagali pri razlagi priporočilnih sistemov na osnovi matrične faktorizacije. Model za napovedovanje ocen na osnovi omejenega Boltzmannovega stroja, pa bo temeljil na uporabi neusmerjenega grafičnega modela.

2.7.1 Usmerjeni modeli

V grobem lahko grafične modele delimo na dva tipa – usmerjene in neusmerjene. Usmerjeni modeli ali Bayesovske mreže (angl. *Bayesian network*) specificirajo faktorizacijo skupne porazdelitve v produkte lokalnih pogojnih porazdelitev [2]. Uporaba usmerjenih povezav v grafu je priročna, ko želimo izraziti vzročno posledična razmerja ali modelirati proces, ki je generiral podatke.

Slučajna spremenljivka v grafu je neodvisna od ostalih, če poznamo njene starše. Od tod sledi formula za faktorizacijo skupne porazdelitve

$$p(\mathbf{x}) = \prod_{\mathrm{x} \in \mathcal{G}} p(\mathrm{x} \mid Pa_{\mathcal{G}}(\mathrm{x})), \tag{2.25}$$

kjer smo z $Pa_{\mathcal{G}}(\mathrm{x})$ označili množico staršev vozlišča x v grafu $\mathcal{G}$.

2.7.2 Neusmerjeni modeli

Neusmerjeni modeli ali Markovska slučajna polja (angl. *Markov random field*) so predstavitve verjetnostnih porazdelitev z neusmerjenimi grafi. Skupna verjetnost konfiguracije slučajnih spremenljivk je izražena kot produkt potencialov klik v grafu [2, 9]. Klika je podmnožica vozlišč, ki tvorijo polno povezan graf, torej vozlišča, kjer je med vsakim parom direktna povezava. Potencial klike $\mathcal{C}$ je funkcija, ki ji priredi nenegativno vrednost $\phi(\mathcal{C})$. Potencial izraža harmonijo med vrednostmi slučajnih spremenljivk v kliki. Večjo vrednost kot ima, večja je verjetnost pojavitve te kombinacije vrednosti, ko izvlečemo vzorec iz porazdelitve modela. Za modeliranje interakcij med slučajnimi spremenljivkami navadno v potenciale vključimo tudi parametre, ki jih nato ocenimo iz podatkov. V tej diplomski nalogi bomo to storili z uporabo metode MAP. Iz potencialov tako definiramo skupno nenormalizirano verjetnost kot produk potencialov klik

$$\tilde{p}(\mathbf{x}) = \prod_{\mathcal{C} \in \mathcal{G}} \phi(\mathcal{C}). \tag{2.26}$$

Da dobimo pravo verjetnost, pa $\tilde{p}$ še normaliziramo

$$p(\mathbf{x}) = \frac{1}{Z} \tilde{p}(\mathbf{x}), \tag{2.27}$$

kjer je

$$Z = \sum_{\boldsymbol{x}} \tilde{p}(\boldsymbol{x}). \tag{2.28}$$

Znak za vsoto tu pomeni seštevanje preko vseh možnih kombinacij vrednost slučajnega vektorja $\mathbf{x}$. Pri zveznih slučajnih spremenljivkah vsoto zamenjamo z integralom po celem definicijskem območju. Nas bodo v resnici zanimali samo potenciali, ki so strogo pozitivni. Tako ima vsaka konfiguracija $\boldsymbol{x}$ pozitivno verjetnost. To nam tudi zagotavlja, da faktorizacija verjetnosti na potenciale klik obstaja. Naj

opomnimo, Markovska slučajna polja navadno definiramo preko lokalnih lastnosti Markova. Na primer prva takšna lastnost je, da sta dve nesosednji spremenljivki neodvisni, če poznamo vrednosti vseh ostalih. Od tod potem lahko upravičimo smiselnost faktorizacije v produkte potencialov.

Poglavje 3

Priporočanje z matričnim razcepom

Naj bo $\boldsymbol{R} \in \mathbb{R}^{n \times m}$ matrika vseh ocen in $r_{ui} = \boldsymbol{R}_{u,i}$ ocena, ki jo uporabnik u dodeli objektu i. Čeprav govorimo o ocenah, bodo v našem eksperimentalnem delu ocene dejansko predstavljale števila ogledov. V praksi je matrika ocen redka, poznanih vrednosti je navadno le nekaj odstotkov. Matriko $\boldsymbol{R}$ bi radi razcepili na produkt dveh matrik $\boldsymbol{R} = \boldsymbol{PQ}$, kjer ti dve matriki določimo tako, da čim bolje apriksimirata znane ocene oziroma še raje, da čimbolje napovedujeta neznane ocene. Vrstico u iz matrike $\boldsymbol{P}$ označimo s $\boldsymbol{p}_u^\top$ in jo poimenujemo *profil uporabnika*, ravno tako stolpec i iz $\boldsymbol{Q}$ označimo s $\boldsymbol{q}_i$ in ga poimenujemo *profil objekta* (filma, glasbe itd.).

3.1 Osnovni model

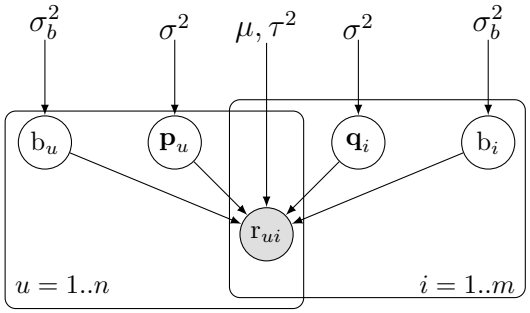
Uporabnike in objekte bi radi predstavili z latentnimi profili. Latentni profil je vektor vložitve uporabnika v latentni vektorski prostor. Želeli bi, da nam profil uporabnika omogoča čimboljše napovedovanje njegovih neznanih ocen. Verjetnostno ozadje problema, razloženo v nadaljevanju diplomske naloge, avtorji predstavijo v člankih [21, 22, 1], ob tem pa uporabijo drugačne in kompleksnejše modele. Oblika osnovnega modela, ki ga bomo uporabili, je predstavljena v knjigi [20].

Predpostavimo, da je ocena r_{ui} sestavljena iz splošnega povprečja, pristranskosti uporabnika, pristranskosti objekta in interakcije med uporabnikom in objektom.

Pristranskost (angl. *bias*) predstavlja povprečni odmik od povprečja. Interakcijo izrazimo kot skalarni produkt med latentnima profiloma. Pri znanih ravnokar navedenih parametrih oceno modeliramo kot [20]

$$r_{ui} = \mu + b_u + b_i + \boldsymbol{p}_u^\top \boldsymbol{q}_i, \tag{3.1}$$

pri čemer sta $\boldsymbol{p}_u$ in $\boldsymbol{q}_i$ vektorja dimenzije d, b_u in b_i pristranskosti, μ pa splošno povprečje. Teh parametrov ne poznamo, zanje pa bi radi vrednosti, s katerimi bo moč čimbolje napovedati nove ocene. Za začetek narišimo grafični model osnovne matrične faktorizacije, ki je prikazan na sliki 3.1 in ga pojasnjujemo v nadaljevanju. Da nekoliko olajšamo notacijo, uvedemo oznako $\boldsymbol{\Theta} = \{\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{b}_u, \boldsymbol{b}_i\}$, kjer



Slika 3.1: Prikaz grafičnega modela osnovne matrične faktorizacije. Vozlišče ocene r_{ui} je osenčeno, kar pomeni da gre za opaženo slučajno spremenljivko. V modelu predpostavimo, da so oceno r_{ui} generirale skrite spremenljivke $\mathbf{p}_u$, $\mathbf{q}_i$, b_u, b_i, kar na sliki predstavimo s puščicami. Vse slučajne spremenljivke pa so odvisne še od konstant, σ^2 predstavlja varianco latentnih vektorjev, σ_b^2 varianco pristranskosti, μ povprečno oceno in τ^2 varianco posamezne ocene. Pravokotnik, imenovan pladenj (angl. *plate*), narisan okrog slučajnih spremenljivk, označuje ponavljanje, v konkretnem primeru, ko indeks u preteče vrednosti od 1 do n oziroma indeks i od 1 do m.

sta $\boldsymbol{P}$ in $\boldsymbol{Q}$ matriki z vsemi profili, $\boldsymbol{b}_u$ in $\boldsymbol{b}_i$ pa vektorja z vsemi pristranskostmi. Predpostavimo normalno porazdelitev ocen in pogojno neodvisnost med ocenami, ki je razvidna iz grafa, ter zapišimo verjetje [22]

$$p(\boldsymbol{R} \,|\, \boldsymbol{\Theta}, \mu, \tau^2) = \prod_{u=1}^{n} \prod_{i=1}^{m} \Big[\mathcal{N}(r_{ui} \,|\, \mu + b_u + b_i + \boldsymbol{p}_u^\top \boldsymbol{q}_i, \tau^2) \Big]^{\mathbf{1}_{ui}}. \tag{3.2}$$

Tu je $\mathbf{1}_{ui}$ indikator, ki ima vrednost 1, če je uporabnik u ocenil objekt i, in 0 sicer.

V tej točki lahko uporabimo metodo največjega verjetja za ocenjevanje parametrov, vendar se v naših eksperimentih izkaže, da se dobljeni model preveč prilega učnim podatkov. Torej znane ocene, na katerih se je učil, dobro rekonstruira, slabše pa napoveduje nove ocene. Vendar nas ravno slednje najbolj zanima. Zato se odločimo za uporabo apriornih porazdelitev na parametrih, te bodo pa kontrolirale kompleksnost modela. S tem upamo, da bo model bolje posploševal na še nevidenih podatkih. Definirajmo torej apriorne verjetnosti

$$p(\boldsymbol{b}_u \,|\, \sigma_b^2) = \prod_{u=1}^{n} \mathcal{N}(b_u \,|\, 0, \sigma_b^2), \tag{3.3}$$

$$p(\boldsymbol{b}_i \,|\, \sigma_b^2) = \prod_{i=1}^{m} \mathcal{N}(b_i \,|\, 0, \sigma_b^2), \tag{3.4}$$

$$p(\boldsymbol{P} \,|\, \sigma^2) = \prod_{u=1}^{n} \mathcal{N}(\boldsymbol{p}_u \,|\, \boldsymbol{0}, \sigma^2 \boldsymbol{I}), \tag{3.5}$$

$$p(\boldsymbol{Q} \,|\, \sigma^2) = \prod_{i=1}^{m} \mathcal{N}(\boldsymbol{q}_i \,|\, \boldsymbol{0}, \sigma^2 \boldsymbol{I}), \tag{3.6}$$

kjer za vse parametre predpostavljamo normalno porazdelitev s pričakovano vrednostjo 0. Zaradi predpostavke o neodvisnosti, razvidne iz grafičnega modela 3.1, se apriorne verjetnosti izražajo kot produkt posameznih gostot. Da zmanjšamo število hiperparametrov, smo se odločili, da si pristranskosti delijo varianco σ_b^2, prav tako pa si varianco σ^2 delijo tudi profili objektov in uporabnikov . Po Bayesovi formuli zapišimo, čemu je sorazmerna aposteriorna verjetnost parametrov, po tem ko opazimo podatke

$$\begin{aligned} &p(\boldsymbol{\Theta} \,|\, \boldsymbol{R}, \sigma^2, \sigma_b^2, \tau^2, \mu) \propto \\ &p(\boldsymbol{R} \,|\, \boldsymbol{\Theta}, \mu, \tau^2) p(\boldsymbol{b}_u \,|\, \sigma_b^2) p(\boldsymbol{b}_i \,|\, \sigma_b^2) p(\boldsymbol{P} \,|\, \sigma^2) p(\boldsymbol{Q} \,|\, \sigma^2). \end{aligned} \tag{3.7}$$

Tu predpostavimo, da splošno povprečje μ poznamo oziroma ga na začetku kar ocenimo iz podatkov, lahko bi pa tudi zanj uporabili apriorno verjetnost.

Po metodi *MAP* izberemo parametre, ki maksimizirajo logaritem aposteriorne

verjetnosti [21]

$$\hat{\boldsymbol{\Theta}} = \arg\max_{\boldsymbol{\Theta}} \log p(\boldsymbol{\Theta} \mid \boldsymbol{R}, \sigma^2, \sigma_b^2, \tau^2, \mu) \tag{3.8}$$

$$= \arg\max_{\boldsymbol{\Theta}} \log p(\boldsymbol{R} \mid \boldsymbol{\Theta}, \mu, \tau^2) + \log p(\boldsymbol{P} \mid \sigma^2) + \log p(\boldsymbol{Q} \mid \sigma^2) + \log p(\boldsymbol{b}_u \mid \sigma_b^2) + \log p(\boldsymbol{b}_i \mid \sigma_b^2) \tag{3.9}$$

$$= \arg\min_{\boldsymbol{\Theta}} \frac{1}{2} \sum_{(u,i)\in\mathcal{R}} (r_{ui} - \hat{r}_{ui})^2 + \frac{\lambda}{2} \sum_{u=1}^{n} \|\boldsymbol{p}_u\|^2 + \frac{\lambda}{2} \sum_{i=1}^{m} \|\boldsymbol{q}_i\|^2 + \frac{\lambda_b}{2} \|\boldsymbol{b}_u\|^2 + \frac{\lambda_b}{2} \|\boldsymbol{b}_i\|^2, \tag{3.10}$$

kjer je $\lambda = \frac{\tau^2}{\sigma^2}$, $\lambda_b = \frac{\tau^2}{\sigma_b^2}$, $\mathcal{R}$ množica vseh parov (u, i), za katere poznamo ocene in $\hat{r}_{ui}$ pričakovana vrednost

$$\hat{r}_{ui} = \mathbb{E}\left[\mathrm{r}_{ui} \mid \boldsymbol{p}_u, \boldsymbol{q}_i, b_u, b_i, \mu\right] = \mu + b_u + b_i + \boldsymbol{p}_u^\top \boldsymbol{q}_i. \tag{3.11}$$

Definirajmo še kriterijsko funkcijo J kot izraz iz (3.10)

$$J(\boldsymbol{\Theta}) = \frac{1}{2} \sum_{(u,i)\in\mathcal{R}} (r_{ui} - \hat{r}_{ui})^2 + \frac{\lambda}{2} \sum_{u=1}^{n} \|\boldsymbol{p}_u\|^2 + \frac{\lambda}{2} \sum_{i=1}^{m} \|\boldsymbol{q}_i\|^2 + \frac{\lambda_b}{2} \|\boldsymbol{b}_u\|^2 + \frac{\lambda_b}{2} \|\boldsymbol{b}_i\|^2. \tag{3.12}$$

Vidimo, da je sestavljena iz vsote kvadratov razlik med pravo in napovedano oceno, ki izvira iz verjetja, ter členov ℓ_2 regularizacije, ki izvirajo iz apriornih verjetnosti. Za optimizacijo se bomo poslužili gradientnega spusta s skupinami velikosti 1. Za izvedbo postopka potrebujemo odvode kriterijske funkcije po parametrih pri enem samem učnem primeru. Denimo, da gre za uporabnika u, ki je ocenil objekt i z oceno r_{ui}

$$\frac{\partial J(\boldsymbol{\Theta})}{\partial \boldsymbol{p}_u} = -(r_{ui} - \hat{r}_{ui})\boldsymbol{q}_i + \lambda \boldsymbol{p}_u, \tag{3.13}$$

$$\frac{\partial J(\boldsymbol{\Theta})}{\partial \boldsymbol{q}_i} = -(r_{ui} - \hat{r}_{ui})\boldsymbol{p}_u + \lambda \boldsymbol{q}_i, \tag{3.14}$$

$$\frac{\partial J(\boldsymbol{\Theta})}{\partial b_u} = -(r_{ui} - \hat{r}_{ui}) + \lambda_b b_u, \tag{3.15}$$

$$\frac{\partial J(\boldsymbol{\Theta})}{\partial b_i} = -(r_{ui} - \hat{r}_{ui}) + \lambda_b b_i. \tag{3.16}$$

Parametroma λ in λ_b pravimo regularizacijska parametra.

Lahko bi že začeli pri definiciji kriterijske funkcije kot vsote kvadratov razlik med pravimi in napovedanimi ocenami, kasneje pa dodali še regularizacijo. Odločili smo se za osnovno razlago verjetnostnega ozadja, ker nam omogoča večjo svobodo pri modeliranju in razširitvah.

Zapišimo še posodobitev parametrov pri uporabi gradientnega spusta s skupinami velikosti 1. Do konvergence se sprehajamo čez učno množico in ob vsakem učnem primeru parameter $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ popravimo

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \alpha^{(epoch)} \frac{\partial J(\boldsymbol{\Theta}^{(k)})}{\partial \boldsymbol{\theta}}, \tag{3.17}$$

kjer uporabimo prej zapisane odvode pri enem samem učnem primeru. Stopnjo učenja zmanjšamo po vsakem obhodu cele podatkovne množice z uporabo enačbe

$$\alpha^{(epoch+1)} = \rho \alpha^{(epoch)}. \tag{3.18}$$

3.2 Razširitve

Po tem, ko smo razložili idejo osnovnega modela, predlagamo še nekaj razširitev, ki jih bomo preizkusili. Vsako razširitev predstavimo samostojno, čeprav jih bomo v iskanju najprimernejšega modela tudi združevali.

3.2.1 Uteži na latentnih faktorjih

Napoved ocene r_{ui} smo sestavili iz pristranskosti in interakcije med uporabnikom in objektom, izražene kot skalarni produkt. Uporabniški profil oziroma vektor je sestavljen iz različnih latentnih faktorjev. Čeprav ne vemo, katere lastnosti predstavljajo faktorji, se zdi smiselno, da so določeni faktorji pomembnejši od ostalih. To je motivacija za naslednji pristop, ki združuje matrični razcep z linearno regresijo tako, da latentne faktorje uteži z vektorjem $\boldsymbol{w}$ [4]. Vektor $\boldsymbol{w}$ je tako še en parameter, ki ga dobimo s hkratno optimizacijo kriterijske funkcije J preko vseh parametrov, torej $\boldsymbol{\Theta} = \{\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{b}_u, \boldsymbol{b}_i, \boldsymbol{w}\}$. Oblika kriterijske funkcije ostaja enaka kot v (2.23), spremeni pa se parametrizacija napovedi ocene [4]

$$\hat{r}_{ui} = \mu + b_u + b_i + (\boldsymbol{w} \odot \boldsymbol{p}_u)^\top \boldsymbol{q}_i, \tag{3.19}$$

tu oznaka $\odot$ predstavlja množenje po komponentah, poznano pod imenom Hadamardov produkt. Zapišimo še potrebne odvode za optimizacijo

$$\frac{\partial J(\boldsymbol{\Theta})}{\partial \boldsymbol{w}} = -(r_{ui} - \hat{r}_{ui})(\boldsymbol{p}_u \odot \boldsymbol{q}_i) + \lambda \boldsymbol{w}, \tag{3.20}$$

$$\frac{\partial J(\boldsymbol{\Theta})}{\partial \boldsymbol{p}_u} = -(r_{ui} - \hat{r}_{ui})(\boldsymbol{w} \odot \boldsymbol{q}_i) + \lambda \boldsymbol{p}_u, \tag{3.21}$$

$$\frac{\partial J(\boldsymbol{\Theta})}{\partial \boldsymbol{q}_i} = -(r_{ui} - \hat{r}_{ui})(\boldsymbol{w} \odot \boldsymbol{p}_u) + \lambda \boldsymbol{q}_i. \tag{3.22}$$

Odvoda po b_u in b_i iz (2.28) in (2.29) se, ob upoštevanju nove parametrizacije $\hat{r}_{ui}$, ne spremenita.

3.2.2 Upoštevanje prijateljstev

Metoda, opisana v naslednjem poglavju, je predstavljena v [20, 4], vendar jo uporabijo za modeliranje implicitnih informacij (angl. *implicit feedback*), ki jih nosijo uporabnikove izbire ogleda ali ocenjevanja vsebine.

V nekaterih primerih imamo na voljo tudi podatke o prijateljstvih med uporabniki. To bi radi na smiseln način vključili v model in s tem izboljšali napovedi. Vpliv prijateljev na nekega uporabnika lahko predstavimo s popravljanjem uporabniškega profila. Lahko bi za napoved ocene uporabniškemu profilu $\boldsymbol{p}_u$ prišteli še uporabniške profile vseh njegovih prijateljev $\boldsymbol{p}_v$ pred skalarnim produktom s $\boldsymbol{q}_i$. To bi pomenilo, da vsaka oseba svoje prijatelje naredi malo podobnejše sebi. Poleg tega bi profil $\boldsymbol{p}_u$ predstavljal mešanico uporabnikovih preferenc in njegovih vplivov na prijatelje. Od tod se porodi ideja, da modeliramo uporabnikove preference ločeno od njegovega vpliva na prijatelje. To omogoča več svobode, med drugim tudi prej opisano možnost. Torej, označimo uporabnikov vpliv na prijatelje z vektorjem $\boldsymbol{f}_u$. Ko napovemo oceno, prištejmo uporabniškemu profilu vse vektorje vplivov njegovih prijateljev. Oziroma, da ne delamo prevelikih razlik med ljudmi z različnim številom prijateljev, vsoto še normalizirajmo

$$\hat{r}_{ui} = \mu + b_u + b_i + \left(\boldsymbol{p}_u + |\mathcal{F}_u|^{-\alpha} \sum_{j \in \mathcal{F}_u} \boldsymbol{f}_j \right)^\top \boldsymbol{q}_i. \tag{3.23}$$

Oznaka $\mathcal{F}_u$ predstavlja množico vseh prijateljev uporabnika u. Najpogosteje uporabljen α je $\frac{1}{2}$ ali pa 1, kar predstavlja povprečenje. Sedaj je naša množica parametrov $\boldsymbol{\Theta} = \{\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{b}_u, \boldsymbol{b}_i, \boldsymbol{F}\}$, kjer $\boldsymbol{F} \in \mathbb{R}^{n \times d}$ predstavlja matriko, v katero so

zloženi vsi vektorji $\boldsymbol{f}_u$. Odvodi kriterijske funkcije pri enem samem učnem primeru so

$$\frac{\partial J(\boldsymbol{\Theta})}{\partial \boldsymbol{p}_u} = -(r_{ui} - \hat{r}_{ui})\boldsymbol{q}_i + \lambda \boldsymbol{p}_u, \tag{3.24}$$

$$\frac{\partial J(\boldsymbol{\Theta})}{\partial \boldsymbol{q}_i} = -\left(r_{ui} - \hat{r}_{ui}\right)\left(\boldsymbol{p}_u + |\mathcal{F}_u|^{-\alpha} \sum_{j \in \mathcal{F}_u} \boldsymbol{f}_j\right) + \lambda \boldsymbol{q}_i, \tag{3.25}$$

$$\frac{\partial J(\boldsymbol{\Theta})}{\partial \boldsymbol{f}_j} = -\left(r_{ui} - \hat{r}_{ui}\right)|\mathcal{F}_u|^{-\alpha}\boldsymbol{q}_i + \lambda \boldsymbol{f}_j. \tag{3.26}$$

Odvoda po b_u in b_i se ne spremenita.

3.2.3 Aktivacijska funkcija

Ocene, ki jih napovedujemo, navadno zavzamejo le omejen nabor števil. Naj bo to prvih pet naravnih števil ali celoten interval med 0 in 1. Pa tudi če napovedujemo vrednost, ki ni omejena, kot so recimo števila ogledov, je v nekaterih primerih uporabe smiselno, da vrednost skrčimo na interval $[0, 1]$. Zato bi bilo smiselno tudi napoved skrčiti na tak interval. V nadaljevanju brez škode za splošnost predpostavljamo, da gre za interval $[0, 1]$.

V splošnem je model ocene

$$\hat{r}_{ui} = g\left(\mu + b_u + b_i + \boldsymbol{p}_u^\top \boldsymbol{q}_i\right), \tag{3.27}$$

kjer je g aktivacijska funkcija. Najenostavnejša rešitev je modeliranje ocene kot neomejene vrednosti, če pa napoved preseže mejne vrednosti, jo primerno odrežemo

$$id(x) = \begin{cases} 1; & x > 1 \\ x; & 0 \leq x \leq 1 \\ 0; & x < 0, \end{cases} \qquad id'(x) = 1. \tag{3.28}$$

Opazimo, da smo odvod izračunali, kot če funkcije ne bi odrezali. Druga možnost za izbiro aktivacijske funkcije je logistična funkcija [21], ki ima obliko

$$\sigma(x) = \frac{1}{1 - e^{-x}}, \qquad \sigma'(x) = \sigma(x)\left(1 - \sigma(x)\right), \tag{3.29}$$

tretja možnost pa je prirejena funkcija ReLU

$$relu(x) = \begin{cases} 1; & x > 1 \\ x; & 0 \leq x \leq 1 \\ 0; & x < 0, \end{cases} \qquad relu'(x) = \begin{cases} \varepsilon; & x > 1 \\ 1; & 0 \leq x \leq 1 \\ \varepsilon; & x < 0, \end{cases} \tag{3.30}$$

kjer je $0 \leq \varepsilon < 1$.

Ker je vrednost ocene omejena med 0 in 1, je zaloga vrednosti vseh treh aktivacijskih funkcij interval $[0, 1]$. Pri definiciji odvoda logistične funkcije σ ni težav, ko definiramo odvoda funkcij id in $relu$, pa se pretvarjamo, da je zaloga vrednosti cela realna os. Čeprav gre tu za matematično nekonsistentnost, to upravičimo s smiselnostjo modelov.

Potrebujemo še odvode kriterijske funkcije J. Za olajšanje notacije definirajmo naprej

$$y = \mu + b_u + b_i + \boldsymbol{p}_u^\top \boldsymbol{q}_i, \tag{3.31}$$

$$\hat{r}_{ui} = g(y). \tag{3.32}$$

Ker gre za enako obliko pri vseh parametrih, zapišimo odvod J pri enem samem učnem primeru po parametru $\boldsymbol{\theta}$, ki lahko predstavlja katerega koli od $\boldsymbol{p}_u, \boldsymbol{q}_i, b_u, b_i$

$$\frac{\partial J(\boldsymbol{\Theta})}{\partial \boldsymbol{\theta}} = -\left(r_{ui} - \hat{r}_{ui}\right) g'(y) \frac{\partial y}{\partial \boldsymbol{\theta}} + \lambda_\theta \boldsymbol{\theta}, \tag{3.33}$$

kjer λ_θ predstavlja regularizacijski parameter, ki pripada $\boldsymbol{\theta}$.

Poglavje 4

Priporočanje z Boltzmannovim strojem

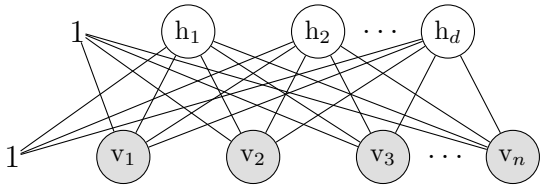
V tem poglavju opišemo drugačen pristop k izdelavi uporabniških profilov v okviru skupinskega filtriranja. Predstavimo vrsto neusmerjenega grafičnega modela, imenovanega omejen Boltzmannov stroj (angl. *restricted Boltzmann machine, RBM*) in ga prilagodimo za napovedovanje ocen. Primer uporabe omejenega Boltzmannovega stroja za skupinsko filtriranje je opisan v [23], vendar se v naših poskusih predlagani model ni obnesel, zato predlagamo drugačen pristop. Poleg tega avtorji modelirajo ocene kot diskretne slučajne spremenljivke, v tej diplomski nalogi pa jih obravnavamo kot zvezne vrednosti.

4.1 Omejen Boltzmannov stroj

Omejen Boltzmannov stroj je v osnovi generativen model. To pomeni, da modelira porazdelitev podatkov s $p(\mathbf{x})$ in je po učenju sposoben generiranja novih primerov. Seveda se ga lahko uporabi tudi za nadzorovano učenje.

Najprej bomo predstavili navaden model omejenega Boltzmannovega stroja, nato pa predlagali potrebne razširitve za modeliranje uporabnikov in napovedovanje ocen. Za začetek si poglejmo sliko 4.1, iz katere vidimo, da ima grafični model obliko dvodelnega grafa [9]. Ravno ta lastnost ga loči od modela splošnega Boltzmannovega stroja, ki ne zahteva dvodelnosti, in nam hkrati nekoliko olajša

učenje.



Slika 4.1: Grafični model omejenega Boltzmannovega stroja. Z v_i so označene opažene slučajne spremenljivke, s h_j pa skrite. Vozlišča s konstanto 1 uporabimo za modeliranje pristranskosti.

V grafu so opažene (angl. *visible*) slučajne spremenljivke osenčene in označene z v_i. Po vrsti jih zložimo v vektor $\mathbf{v}$, ki bo kasneje predstavljal uporabnikove ocene filmov. Skrite (angl. *hidden*) slučajne spremenljivke označimo z vektorjem $\mathbf{h}$, ki bo služil kot latentna predstavitev vektorja $\mathbf{v}$. Imamo tudi dve vozlišči, označeni z 1, ki se uporabljata za modeliranje pristranskosti. Vsaka povezava v grafu ima določeno vrednost oziroma utež. Vse vrednosti na povezavah med skritimi in vidnimi spremenljivkami zložimo v matriko $\boldsymbol{W} \in \mathbb{R}^{n \times d}$, vrednosti na povezavah med 1 in vidnimi spremenljivkami v vektor $\boldsymbol{b} \in \mathbb{R}^n$, med 1 in skritimi spremenljivkami pa v vektor $\boldsymbol{c} \in \mathbb{R}^d$.

Ker so v večini literature slučajne spremenljivke osnovnega modela RBM binarne, se bomo v prvem delu tudi mi tega držali. Razširitev na omejene zvezne spremenljivke je enostavna, saj vse, kar bomo zapisali z vsotami, bo veljalo tudi ob uporabi integralov.

Iz uvodnega poglavja vemo, da neusmerjeni modeli specificirajo faktorizacijo skupne porazdelitve v produkte potencialov klik. Označimo s $\mathcal{C}_i$ kliko med 1 in v_i, s $\mathcal{C}_j$ kliko med 1 in h_j in s $\mathcal{C}_{ij}$ kliko med v_i in h_j ter definirajmo funkcije ϕ imenovane potenciali, ki vsaki kliki priredijo pozitivno vrednost

$$\phi(\mathcal{C}_i) = \exp(v_i b_i), \tag{4.1}$$

$$\phi(\mathcal{C}_j) = \exp(h_j c_j), \tag{4.2}$$

$$\phi(\mathcal{C}_{ij}) = \exp(v_i w_{ij} h_j), \tag{4.3}$$

Potencial izraža harmonijo med vrednostmi slučajnih spremenljivk v kliki. Večjo kot ima vrednost, večja je verjetnost, da opazimo takšno kombinacijo vrednosti

spremenljivk. Tako definirani potenciali, v katerih uporabimo parametre modela, so enostaven način modeliranja interakcij med slučajnimi spremenljivkami (glej uvodno poglavje o neusmerjenih grafičnih modelih). Od tod sledi formula za skupno verjetnost

$$p(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} \prod_{i=1}^{n} \exp(v_i b_i) \prod_{j=1}^{d} \exp(h_j c_j) \prod_{i=1}^{n} \prod_{j=1}^{d} \exp(v_i w_{ij} h_j) \tag{4.4}$$

$$= \frac{1}{Z} \exp\left(\sum_{i=1}^{n} v_i b_i + \sum_{j=1}^{d} h_j c_j + \sum_{i=1}^{n} \sum_{j=1}^{d} v_i w_{ij} h_j \right) \tag{4.5}$$

$$= \frac{1}{Z} \exp\left(\boldsymbol{b}^\top \boldsymbol{v} + \boldsymbol{c}^\top \boldsymbol{h} + \boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h} \right). \tag{4.6}$$

Produkt potencialov lahko zavzame katerokoli pozitivno realno število, zato ga je potrebno normalizirati s konstanto Z, da postane prava verjetnost. Priročno je označiti nenormalizirano verjetnost s simbolom $\tilde{p}$, tako lahko zapišemo skupno verjetnost kot

$$p(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} \tilde{p}(\boldsymbol{v}, \boldsymbol{h}). \tag{4.7}$$

V vrstici (4.6) smo v eksponentu dobili poseben izraz, imenovan energija [9, 16]

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\boldsymbol{b}^\top \boldsymbol{v} - \boldsymbol{c}^\top \boldsymbol{h} - \boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h}, \tag{4.8}$$

od tod verjetnost dobi obliko

$$p(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} \exp\left(-E(\boldsymbol{v}, \boldsymbol{h})\right). \tag{4.9}$$

Z vidika strojnega učenja dodani minusi ne služijo ničemur, saj lahko učni algoritem prosto določa vrednosti parametrov in predznak energijske funkcije. Pišemo jih zgolj zaradi kompatibilnosti s fizikalno literaturo, od koder to izvira.

Energija je mera kompatibilnosti med stanji spremenljivk modela. Stanja z nižjo energijo ustrezajo bolj verjetnim stanjem, in obratno, stanja z visoko energijo ustrezajo manj verjetnim stanjem. Nadzorovano učenje tu ustreza iskanju energijske funkcije, ki asociira nizko vrednost energije s pravilnimi vrednostimi in visoko z nepravilnimi [15].

Zapišimo še normalizacijsko konstanto, imenovano particijska funkcija

$$Z = \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v}, \boldsymbol{h})), \tag{4.10}$$

kjer vsota po $\boldsymbol{v}$ oziroma $\boldsymbol{h}$ predstavlja vsoto preko vseh možnih stanj. Konstanti Z pravimo particijska funkcija, ker je odvisna od parametrov modela, ki so skriti v energiji.

Zaradi dvodelne oblike grafa RBM so skrite spremenljivke med seboj neodvisne ob znanih vidnih spremenljivkah in obratno, vidne spremenljivke so med seboj neodvisne, če poznamo skrite. To lepo lastnost lahko uporabimo pri računanju pogojnih verjetnosti [9], ki jih bomo kasneje uporabili pri vzorčenju Gibbs

$$p(\boldsymbol{v} \,|\, \boldsymbol{h}) = \prod_{i=1}^{n} p(v_i \,|\, \boldsymbol{h}), \tag{4.11}$$

$$p(\boldsymbol{h} \,|\, \boldsymbol{v}) = \prod_{j=1}^{d} p(h_j \,|\, \boldsymbol{v}), \tag{4.12}$$

kjer se posamične verjetnosti izražajo kot [9]

$$p(v_i = 1 \,|\, \boldsymbol{h}) = \sigma\left(b_i + \boldsymbol{W}_{i,:}\boldsymbol{h}\right), \tag{4.13}$$

$$p(h_j = 1 \,|\, \boldsymbol{v}) = \sigma\left(c_j + \boldsymbol{v}^\top \boldsymbol{W}_{:,j}\right). \tag{4.14}$$

Vzorčenje Gibbs je postopek za pridobitev zaporedja vzorcev, ki aproksimirajo večrazsežno porazdelitev. Kasneje bomo potrebovali vzorce iz $p(\mathbf{v}, \mathbf{h})$, začeli bomo pri učnem primeru $\boldsymbol{v}^{(0)}$, nato izvlekli vzorec $\boldsymbol{h}^{(0)} \sim p(\mathbf{h} \,|\, \boldsymbol{v}^{(0)})$, nato $\boldsymbol{v}^{(1)} \sim p(\mathbf{v} \,|\, \boldsymbol{h}^{(0)})$ in nadaljevali v tem slogu. Takšnemu zaporedju vzorcev pravimo Markovska veriga (angl. *Markov chain*), ker je porazdelitev posameznega vzorca odvisna samo od njegovega neposrednega predhodnika.

Za učenje bomo uporabili metodo MAP, torej iskali bomo parametre $\boldsymbol{W}$, $\boldsymbol{b}$, $\boldsymbol{c}$, ki maksimizirajo aposteriorno verjetnost. Označimo parametre s $\boldsymbol{\Theta} = \{\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{c}\}$ in učno množico z $\mathbb{V} = \{\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, ..., \boldsymbol{v}^{(m)}\}$. Zapišimo logaritem verjetja pri enem samem učnem primeru $\boldsymbol{v}$ [24], ob tem pa izpustimo pogojevanje na $\boldsymbol{\Theta}$, da poenostavimo notacijo

$$\ell(\boldsymbol{\Theta}; \boldsymbol{v}) = \log p(\boldsymbol{v}) \tag{4.15}$$

$$= \log \tilde{p}(\boldsymbol{v}) - \log Z. \tag{4.16}$$

Apriorne verjetnosti bomo določili kasneje in jih dodali logaritmu verjetja kot regularizacijo. Za učenje najprej potrebujemo odvode logaritma verjetja po parametrih modela. Odvod po parametru $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ je sestavljen iz dveh členov

$$\frac{\partial \ell(\boldsymbol{\Theta}; \boldsymbol{v})}{\partial \boldsymbol{\theta}} = \frac{\partial \log \tilde{p}(\boldsymbol{v})}{\partial \boldsymbol{\theta}} - \frac{\partial \log Z}{\partial \boldsymbol{\theta}}. \tag{4.17}$$

Izračunajmo najprej prvi člen [24]

$$\frac{\partial \log \tilde{p}(\boldsymbol{v})}{\partial \boldsymbol{\theta}} = \frac{1}{\tilde{p}(\boldsymbol{v})} \frac{\partial \tilde{p}(\boldsymbol{v})}{\partial \boldsymbol{\theta}} \tag{4.18}$$

$$= \frac{1}{\tilde{p}(\boldsymbol{v})} \frac{\partial \sum_{\boldsymbol{h}} \tilde{p}(\boldsymbol{v}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} \tag{4.19}$$

$$= \frac{1}{\tilde{p}(\boldsymbol{v})} \sum_{\boldsymbol{h}} \frac{\partial \exp(-E(\boldsymbol{v}, \boldsymbol{h}))}{\partial \boldsymbol{\theta}} \tag{4.20}$$

$$= -\frac{1}{\tilde{p}(\boldsymbol{v})} \sum_{\boldsymbol{h}} \tilde{p}(\boldsymbol{v}, \boldsymbol{h}) \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} \tag{4.21}$$

$$= -\sum_{\boldsymbol{h}} \frac{Z\tilde{p}(\boldsymbol{v}, \boldsymbol{h})}{Z\tilde{p}(\boldsymbol{v})} \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} \tag{4.22}$$

$$= -\sum_{\boldsymbol{h}} p(\boldsymbol{h} \mid \boldsymbol{v}) \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} \tag{4.23}$$

$$= -\mathbb{E}_{\mathbf{h} \sim p(\mathbf{h} \mid \boldsymbol{v})} \left[\frac{\partial E(\boldsymbol{v}, \mathbf{h})}{\partial \boldsymbol{\theta}} \right]. \tag{4.24}$$

Odvod energije po parametru $\boldsymbol{\theta}$ je enostavno izračunati in ga bomo zapisali kasneje. Kot vidimo je prvi člen odvoda logaritma verjetja enak negativni pričakovani vrednosti odvoda energije po porazdelitvi skritih spremenljivk $\boldsymbol{h}$, ko poznamo vidne spremenljivke $\boldsymbol{v}$.

Zapišimo še, čemu je enak drugi člen [24]

$$\frac{\partial \log Z}{\partial \boldsymbol{\theta}} = \frac{1}{Z} \frac{\partial Z}{\partial \boldsymbol{\theta}} \tag{4.25}$$

$$= \frac{1}{Z} \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} \frac{\partial \exp(-E(\boldsymbol{v}, \boldsymbol{h}))}{\partial \boldsymbol{\theta}} \tag{4.26}$$

$$= -\frac{1}{Z} \sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v}, \boldsymbol{h})) \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} \tag{4.27}$$

$$= -\sum_{\boldsymbol{v}} \sum_{\boldsymbol{h}} p(\boldsymbol{v}, \boldsymbol{h}) \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} \tag{4.28}$$

$$= -\mathbb{E}_{(\mathbf{v}, \mathbf{h}) \sim p(\mathbf{v}, \mathbf{h})} \left[\frac{\partial E(\mathbf{v}, \mathbf{h})}{\partial \boldsymbol{\theta}} \right]. \tag{4.29}$$

Kot vidimo, je tu pričakovana vrednost izračunana glede na skupno porazdelitev modela. Ob upoštevanju teh dveh izračunov se odvod logaritma verjetja glasi [24]

$$\frac{\partial \ell(\boldsymbol{\Theta}; \boldsymbol{v})}{\partial \boldsymbol{\theta}} = -\mathbb{E}_{\mathbf{h} \sim p(\mathbf{h} \mid \boldsymbol{v})} \left[\frac{\partial E(\boldsymbol{v}, \mathbf{h})}{\partial \boldsymbol{\theta}} \right] + \mathbb{E}_{(\mathbf{v}, \mathbf{h}) \sim p(\mathbf{v}, \mathbf{h})} \left[\frac{\partial E(\mathbf{v}, \mathbf{h})}{\partial \boldsymbol{\theta}} \right]. \tag{4.30}$$

Naj opozorimo, da gre za logaritem verjetja ob enem samem učnem primeru $\boldsymbol{v}$. Če nas zanima odvod pri celi učni množici, moramo sešteti odvode preko vseh učnih primerov. Čeprav imamo v (4.30) lepo enačbo za izračun odvoda, je natančen izračun v praksi neizvedljiv. Pričakovana vrednost iz drugega člena namreč vključuje seštevanje preko vseh možnih kombinacij vseh skritih in vidnih spremenljivk. Zato se poslužimo naslednje aproksimacije [24, 10]

$$\frac{\partial \ell(\boldsymbol{\Theta}; \boldsymbol{v})}{\partial \boldsymbol{\theta}} \approx -\left\langle \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} \right\rangle_0 + \left\langle \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} \right\rangle_S, \tag{4.31}$$

kjer $\langle . \rangle_S$ predstavlja oceno pričakovane vrednosti oziroma vzorčno povprečje po številu korakov S bločnega vzorčenja Gibbs, ki tvori Markovsko verigo vzorcev. Povprečje v prvem členu pri številu korakov 0 se računa glede na porazdelitev podatkov. Povprečje v drugem členu pa po S korakih vzorčenja Gibbs iz pogojnih porazdelitev, kjer en korak predstavlja vzorčenje vidnih in skritih spremenljivk.

Spomnimo se, da je učenje po metodi največjega verjetja ekvivalentno minimizaciji Kullback-Leiblerjeve divergence med empirično porazdelitvijo podatkov in družino porazdelitev, sprecificiranih z modelom. Izkaže se, da aproksimacija iz (4.31) še bolje kot odvodu logaritma verjetja sledi odvodu druge funkcije, imenovane kontrastna divergenca (angl. *contrastive divergence*) [10, 3, 26], ki ima obliko

$$D_{CD_S}(p_0 \parallel p_\infty) = D_{KL}(p_0 \parallel p_\infty) - D_{KL}(p_S \parallel p_\infty). \tag{4.32}$$

Tu veljajo oznake $p_0 = \hat{p}_{data}$ in $p_\infty = p_{model}$, namenoma smo raje uporabili zapise s številkami, ker predstavljajo število korakov vzorčenja Gibbs. Torej empirično porazdelitev podatkov imamo že na začetku preden začnemo vzorčiti, porazdelitev modela pa ko Markovska veriga vzorcev doseže stacionarno porazdelitev.

Kontrastna divergenca $D_{CD_S}(p_0 \parallel p_\infty)$ je vedno nenegativna, ker je p_S vedno S korakov bližje p_∞ kot je p_0, razen če je seveda $p_0 = p_\infty$. Od tod sledi $D_{KL}(p_0 \parallel p_\infty) \geq D_{KL}(p_S \parallel p_\infty)$. Dodatno, če imajo vsi prehodi v Markovski verigi pozitivno verjetnost, potem iz $p_0 = p_S$ sledi $p_0 = p_\infty$. Z drugimi besedami, če se porazdelitev ni spremenila v prvih S korakih, potem je veriga že v stacionarni porazdelitvi in je model popoln. Velja opomniti, da kontrastno divergenco lahko naredimo zelo majhno že samo če uporabimo Markovsko verigo, ki se zelo počasi meša, kar nam pri učenju modela ne koristi. Zato je zelo pomembno, da zagotovimo dobro mešanje. Pri tem nam lahko pomagata tudi regularizacija oziroma uporaba apriornih verjetnosti [10].

Zapišimo še odvode energije po parametrih modela

$$\frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial \boldsymbol{W}} = -\boldsymbol{v}\boldsymbol{h}^\top, \tag{4.33}$$

$$\frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial \boldsymbol{b}} = -\boldsymbol{v}, \tag{4.34}$$

$$\frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial \boldsymbol{c}} = -\boldsymbol{h}. \tag{4.35}$$

Za izračun odvoda aposteriorne verjetnosti moramo opredeliti še apriorne porazdelitve. Za vse parametre bo to, tako kot pri matričnem razcepu, kar normalna porazdelitev

$$p(\boldsymbol{W} \,|\, \sigma_w^2) = \prod_{i=1}^{n} \prod_{j=1}^{d} \mathcal{N}(w_{ij} \,|\, 0, \sigma_w^2), \tag{4.36}$$

$$p(\boldsymbol{b} \,|\, \sigma_b^2) = \prod_{i=1}^{n} \mathcal{N}(b_i \,|\, 0, \sigma_b^2), \tag{4.37}$$

$$p(\boldsymbol{c} \,|\, \sigma_c^2) = \prod_{j=1}^{d} \mathcal{N}(c_j \,|\, 0, \sigma_c^2), \tag{4.38}$$

$$\tag{4.39}$$

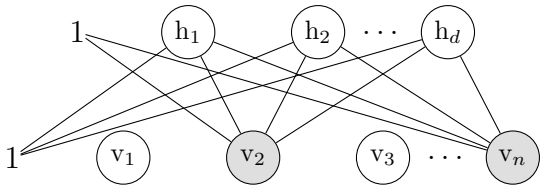
kjer σ_w^2 predstavlja varianco posamezne uteži w_{ij}, σ_b^2 in σ_c^2 pa varianci pripadajočih pristranskosti. Tako definirane apriorne verjetnosti bojo h kriterijski funkciji, po tem ko se znebimo konstant, prispevale člene regularizacije ℓ_2.

4.2 Razširitev za priporočanje

V standardnem modelu omejenega Boltzmannovega stroja smo predpostavili, da poznamo vse vrednosti $v_i^{(u)}$ posameznega učnega primera $\boldsymbol{v}^{(u)}$. V kontekstu priporočilnih sistemov pa temu ni tako. Učni primer $\boldsymbol{v}^{(u)}$ je vektor ocen, ki jih uporabnik dodeli objektom, torej $v_i^{(u)}$ je ocena, ki jo je uporabnik u dodelil objektu i. Z uporabo omejenega Boltzmannovega stroja želimo vektor uporabnikovih ocen $\boldsymbol{v}^{(u)}$ predstaviti v latentnem prostoru z vektorjem $\boldsymbol{h}^{(u)}$, ki ga nato uporabimo za rekonstrukcijo celotnega vektorja $\boldsymbol{v}^{(u)}$, torej tudi manjkajočih ocen.

Problema se bomo lotili tako, da bomo vsakemu uporabniku dodelili svoj RBM, ki bo vključeval samo povezave med poznanimi vrednostmi ocen in skritimi spremenljivkami (glej sliko 4.2). Si bodo pa vsi Boltzmannovi stroji med seboj delili

vrednosti uteži na povezavah [23]. Torej, če sta dva uporabnika ocenila isti objekt, morata imeti iste uteži na povezavah med tem objektom in vsemi skritimi spremenljivkami. Vrednosti skritih spremenljivk pa se seveda razlikujejo med uporabniki. Želeli bi, da model z uporabo skritih spremenljivk predstavi ključne lastnosti uporabnika, ki jih lahko uporabimo za napoved ali rekonstrukcijo ocen.



Slika 4.2: Grafični model omejenega Boltzmannovega stroja za uporabnika z manjkajočimi ocenami, kjer ocen v_1 in v_3 ne poznamo. Vektor $\mathbf{h}$, sestavljen iz h_j, predstavlja latentno predstavitev uporabnika, omogoča pa nam rekonstrukcijo in napovedovanje neznanih ocen.

Osnovni model RBM predpostavlja, da so vrednosti vidnih spremenljivk binarne, torej 0 ali 1. V našem primeru pa vrednosti ocen ležijo na intervalu $[0, 1]$, če ne, pa jih primerno skaliramo. Ob upoštevanju tega se vse vsote preko vidnih spremenljivk $\boldsymbol{v}$ spremenijo v integrale. Problem se pojavi pri izračunu odvoda particijske funkcije Z, ko želimo zamenjati vrstni red odvajanja in integriranja. Teorija mere nam postavlja pogoje, pod katerimi lahko to storimo, na tem mestu pa jih ne bomo navajali. Povzemimo le, da so ti pogoji v našem primeru in pri večini ostalih modelov strojnega učenja izpolnjeni, zato je zamenjava vrstnega reda upravičena [9].

Parametre modela bomo določili z maksimizacijo kriterijske funkcije, ki je sorazmerna logaritmu aposteriorne verjetnost

$$J(\boldsymbol{\Theta}) = \ell(\boldsymbol{\Theta}; \mathbb{V}) - \frac{\lambda_w}{2}\|\boldsymbol{W}\|_F^2 - \frac{\lambda_b}{2}\|\boldsymbol{b}\|^2 - \frac{\lambda_c}{2}\|\boldsymbol{c}\|^2, \tag{4.40}$$

kjer je $\|.\|_F$ Frobeniusova norma (koren iz vsote kvadratov vseh elementov, kot če bi matriko raztegnili v vektor in vzeli 2-normo). Postopek izračuna kriterijske funkcije iz logaritma aposteriorne verjetnosti je enak kot pri matričnem razcepu, zato ga nismo zapisovali.

Za optimizacijo bomo uporabili gradientni vzpon (obratno od spusta, pomikamo se v smeri gradienta), ob tem pa bomo za aproksimacijo gradienta uporabili

majhne skupine in zagon. Ob tem moramo biti pozorni, saj so v vsaki skupini uporabniki, ki so ocenili različne filme, tako bo za izračun približka odvoda po nekem parametru potrebno upoštevati dejansko število ocen iz skupine, ki jih izbrani parameter povezuje s skritimi spremenljivkami.

Denimo, da gre za korak $k+1$ gradientnega vzpona z uporabo zagona in skupin. Označimo z $|\mathbb{V}|$ število vseh učnih primerov in z $|\mathbb{B}|$ število primerov v trenutni skupini. Stopnja učenja je oblike

$$\alpha^{(epoch)} = \rho\alpha^{(epoch-1)}, \tag{4.41}$$

$$\alpha_{\mathbb{B}}^{(k+1)} = \frac{|\mathbb{B}|}{|\mathbb{V}|}\alpha^{(epoch)}, \tag{4.42}$$

kjer je ρ parameter, ki določa upad stopnje učenja, in *epoch* trenutna številka obhoda celotne učne množice. Dejansko bomo za učenje uporabljali $\alpha_{\mathbb{B}}^{(k+1)}$, torej stopnjo učenja pomnoženo z $\frac{|\mathbb{B}|}{|\mathbb{V}|}$. Tako naredimo velikosti posodobitev v enem obhodu podatkov neodvisno od velikosti skupin in posledično je določanje primerne stopnje učenja in primerne velikosti skupin lažje.

Označimo najprej z $\mathcal{I}_{\mathbb{B}} \in \mathbb{R}^n$ vektor, ki ima na i-tem mestu vrednost $\frac{1}{\#_i}$, kjer je $\#_i$ število pojavitev ocene v_i v skupini $\mathbb{B}$. Če se ocena v_i ni pojavila v tej skupini, je na i-tem mestu vrednost 0. Izračunajmo sedaj ocene pričakovanih vrednosti iz (4.31) za skupino $\mathbb{B}$

$$\langle \boldsymbol{v} \rangle_{\mathbb{B},s} = \mathcal{I}_{\mathbb{B}} \odot \sum_{\boldsymbol{v}\in\mathbb{B}} \langle \boldsymbol{v} \rangle_s \,, \tag{4.43}$$

$$\langle \boldsymbol{h} \rangle_{\mathbb{B},s} = \frac{1}{|\mathbb{B}|} \sum_{\boldsymbol{v}\in\mathbb{B}} \langle \boldsymbol{h} \rangle_s \,, \tag{4.44}$$

$$\left\langle \boldsymbol{v}\boldsymbol{h}^\top \right\rangle_{\mathbb{B},s} = \langle \boldsymbol{v} \rangle_{\mathbb{B},s} \langle \boldsymbol{h} \rangle_{\mathbb{B},s}^\top \,, \tag{4.45}$$

kjer je $s \in \{0, S\}$, za aproksimacijo odvodov logaritma verjetja namreč potrebujemo ocene iz obeh korakov. Oznaka $\langle . \rangle_s$ predstavlja oceno pričakovane vrednosti po s korakih vzorčenja Gibbs, opisanega v prejšnjem razdelku. Posodobitev parametrov $\boldsymbol{W}$ se tako glasi

$$\boldsymbol{m}_w^{(k+1)} = (1-\gamma)\left(\left\langle \boldsymbol{v}\boldsymbol{h}^\top \right\rangle_{\mathbb{B},0} - \left\langle \boldsymbol{v}\boldsymbol{h}^\top \right\rangle_{\mathbb{B},S} - \lambda_w \boldsymbol{W}^{(k)}\right) + \gamma \boldsymbol{m}_w^{(k)}, \tag{4.46}$$

$$\boldsymbol{W}^{(k+1)} = \boldsymbol{W}^{(k)} + \alpha_{\mathbb{B}}^{(k+1)} \boldsymbol{m}_w^{(k+1)}, \tag{4.47}$$

kjer $\boldsymbol{m}_w^{(k+1)}$ označuje zagon v tem koraku, γ pa parameter zagona. Posodobitev parametrov pristranskosti $\boldsymbol{b}$ se glasi

$$\boldsymbol{m}_b^{(k+1)} = (1-\gamma)\left(\langle \boldsymbol{v}\rangle_{\mathbb{B},0} - \langle \boldsymbol{v}\rangle_{\mathbb{B},S} - \lambda_b \boldsymbol{b}^{(k)}\right) + \gamma \boldsymbol{m}_b^{(k)}, \tag{4.48}$$

$$\boldsymbol{b}^{(k+1)} = \boldsymbol{b}^{(k)} + \alpha_{\mathbb{B}}^{(k+1)} \boldsymbol{m}_b^{(k+1)}, \tag{4.49}$$

posodobitev parametrov pristranskosti $\boldsymbol{c}$ pa

$$\boldsymbol{m}_c^{(k+1)} = (1-\gamma)\left(\langle \boldsymbol{h}\rangle_{\mathbb{B},0} - \langle \boldsymbol{h}\rangle_{\mathbb{B},S} - \lambda_c \boldsymbol{c}^{(k)}\right) + \gamma \boldsymbol{m}_c^{(k)}, \tag{4.50}$$

$$\boldsymbol{c}^{(k+1)} = \boldsymbol{c}^{(k)} + \alpha_{\mathbb{B}}^{(k+1)} \boldsymbol{m}_c^{(k+1)}. \tag{4.51}$$

Omejen Boltzmannov stroj je generativen model, torej modelira porazdelitev podatkov in je po učenju sposoben generiranja novih primerov iz modela porazdelitve. Ob tem igra pomembno vlogo ozko grlo pretoka informacij, ki ga ustvari dejstvo, da so skrite spremenljivke binarne. Torej v vsakem koraku Markovske verige, ki jo uporabimo tudi za aproksimacijo gradienta, vzorčimo vrednosti skritih spremenljivk iz pogojne porazdelitve $p(\mathbf{h} \,|\, \boldsymbol{v})$. To ozko grlo igra podobno vlogo kot regularizacija [11, 16]. V naših eksperimentih se je vzorčenje skritih spremenljivk izkazalo za slabo. Namesto vzorčenja iz pogojne porazdelitve smo uporabili izračunane verjetnosti aktivacij za nadaljnje korake. To nam omogoča natančnejšo aproksimacijo gradienta v manj korakih. In ker naš končni cilj ni generiranje novih primerov ampak čim natančnejše napovedovanje ocen, se nam dejstvo, da smo izpustili vzorčenje, ne zdi sporno.

Sedaj, ko smo z optimizacijo dobili parametre $\hat{\boldsymbol{W}}$, $\hat{\boldsymbol{b}}$, $\hat{\boldsymbol{c}}$, pa bi radi za nekega uporabnika napovedali ocene, ki jih še ne poznamo. Denimo, da imamo vektor uporabnikovih ocen $\boldsymbol{v}$, v katerem nekaterih vrednosti ne poznamo (na njihovo mesto zapišemo ničle). Izračunajmo najprej pričakovane vrednosti skritih spremenljivk [24], ki jih označimo z vektorjem $\hat{\boldsymbol{h}}$

$$\hat{\boldsymbol{h}} = \mathbb{E}\left[\mathbf{h} \,|\, \boldsymbol{v}\right] \tag{4.52}$$

$$= \sigma(\hat{\boldsymbol{c}} + \hat{\boldsymbol{W}}^\top \boldsymbol{v}), \tag{4.53}$$

kjer smo upoštevali enačbo (4.14) in dejstvo, da je pričakovana vrednost Bernoullijeve slučajne spremenljivke enaka njeni verjetnosti. Logistično funkcijo σ smo uporabili na vsaki komponenti vektorja posamično. Zdaj, ko imamo $\hat{\boldsymbol{h}}$, pa z

uporabo (4.13) izračunamo pričakovane vrednosti vseh vidnih spremenljivk [24]

$$\hat{\boldsymbol{v}} = \mathbb{E}\left[\mathbf{v} \mid \hat{\boldsymbol{h}}\right] \tag{4.54}$$

$$= \sigma(\hat{\boldsymbol{b}} + \hat{\boldsymbol{W}}\hat{\boldsymbol{h}}). \tag{4.55}$$

Tako imamo v vektorju $\hat{\boldsymbol{v}}$ napovedi vseh ocen. Idealno bi bilo izračunati $p(\hat{\boldsymbol{v}} \mid \boldsymbol{v})$, vendar to zahteva marginalizacijo skritih spremenljivk. Izračun bi nam sicer prinesel malenkost bolj natančne rezultate, vendar je uporaba prej opisane točkaste ocene skritih spremenljivk veliko hitrejša.

Poglavje 5

Evalvacija in rezultati

Predstavimo dve podatkovni množici, na katerih bomo uporabili modele iz prejšnjih poglavij za napovedovanje in priporočanje. Pri prvi gre za števila poslušanj iz spletne strani Last.fm, pri drugi pa za ocene filmov iz MovieLens, obe sta dostopni na [25]. V nadaljevanju predlagamo še nekaj smiselnih metrik za evalvacijo rezultatov in predstavimo ter vizualiziramo rezultate.

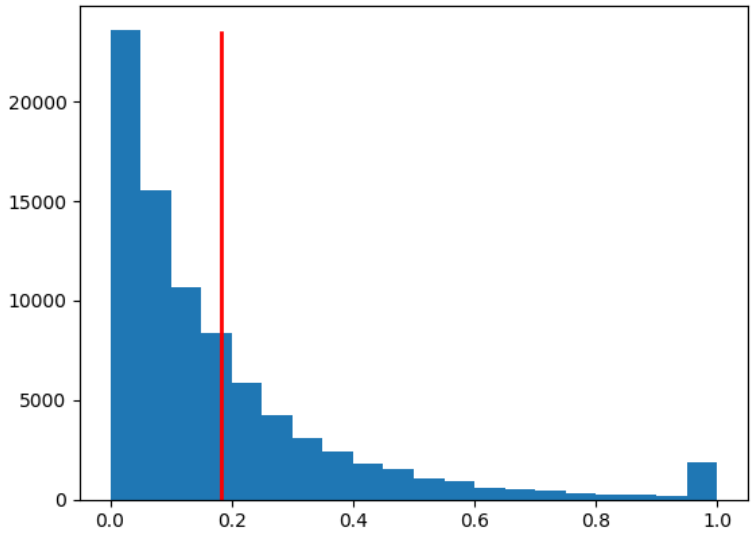
5.1 Podatkovna množica Last.fm

V podatkovni množici Last.fm imamo podan seznam trojk (u, i, r) dolžine 83.550. Podatki predstavljajo število predvajanj glasbe določenega izvajalca. V tem primeru je uporabnik u poslušal skladbe izvajalca i r-krat. Kot že takoj vidimo, so števila predvajanj agregirana, zato za posamezen zapis v podatkovni množici ni časovnega žiga. Imamo 1.892 uporabnikov in 16.532 izvajalcev. Podatke lahko predstavimo z redko matriko, v kateri poznamo le 0.27% vseh vrednosti.

Število predvajanj je lahko katerokoli naravno število. Porazdelitev števila predvajanj dobro sledi Paretovi porazdelitvi, ki se v naravi pojavlja precej pogosto. Paretova porazdelitev je poznana tudi kot pravilo 80-20, ki pravi, da pri mnogih dogodkih približno 80% učinkov izhaja iz 20% vzrokov. Seveda je to pravilo lahko bolj ali manj ekstremno, točno to razmerje pa ustreza Paretovi porazdelitvi z vrednostjo parametra, ki določa njeno obliko, približno 1,16 [5].

Posameznim uporabnikom bi radi priporočili izvajalce, ki jim bodo čim bolj všeč. Število poslušanj veliko bolje izraža uporabnikovo afiniteto do izvajalca kot

eksplicitne ocene. Števila poslušanj skrčimo na interval $[0, 1]$ in jih pred tem primerno odrežemo. Povprečje tako predprocesiranih podatkov je 0,1821, standardni odklon pa 0,2082, kot je prikazano na histogramu na sliki 5.1.

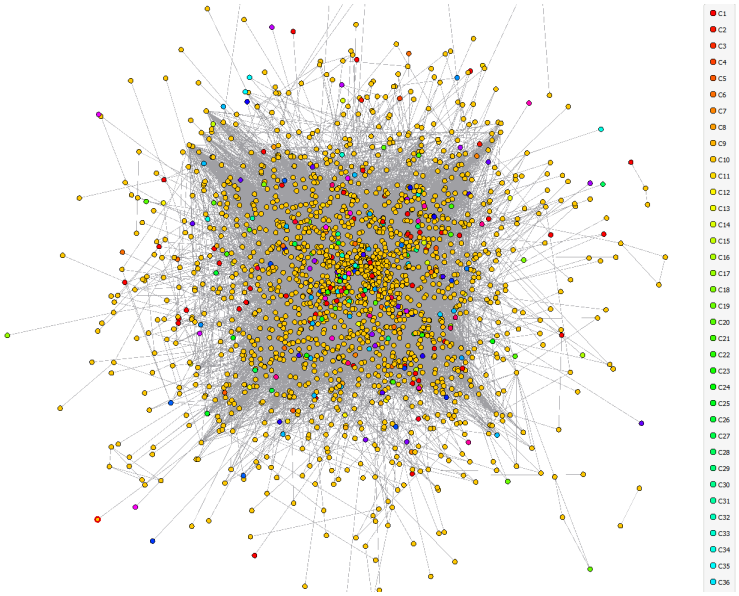


Slika 5.1: Histogram z 20 intervali predprocesiranih števil poslušanj iz podatkovne množice Last.fm. Povprečje 0,1821 je označeno z rdečo črto.

Na voljo imamo tudi podatke o prijateljstvih med uporabniki. Gre za seznam parov (u, v), ki jih interpretiramo kot neusmerjene povezave v grafu prijateljstev. V grafu imamo 1.892 vozlišč in 12.717 povezav, povprečna stopnja vozlišča je tako 13,44, gostota grafa pa 0,0071. Graf, prikazan na sliki 5.2, vizualiziramo z orodjem Orange [6].

5.2 Podatkovna množica MovieLens

V podatkovni množici MovieLens je na voljo 100.836 trojk (u, i, r), za vsako trojko imamo tudi časovni žig nastanka. Skupaj imamo 610 uporabnikov, ki so ocenjevali 9.724 filmov. Od tod sledi, da poznamo 1.70% vseh ocen. Ocene se gibljejo od 0,5 do 5 s korakom 0,5, razvrstimo pa jih lahko na 10 intervalov. Ocene ravno tako

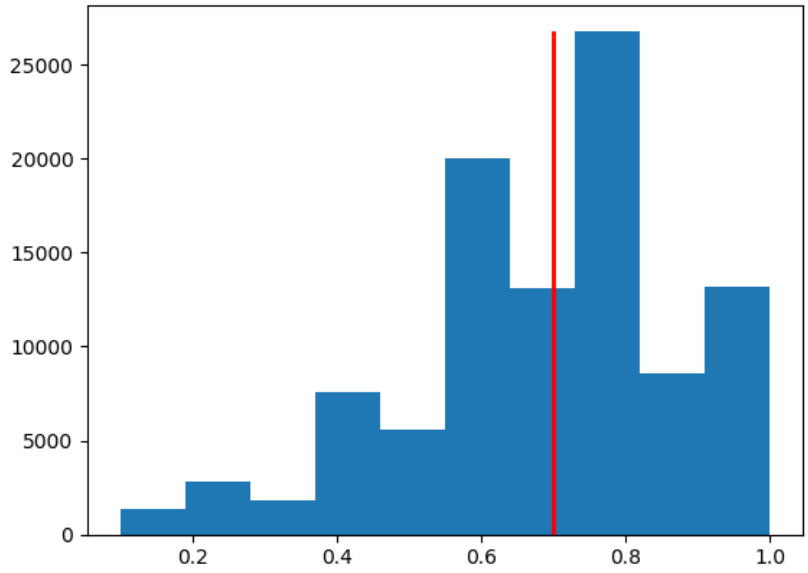


Slika 5.2: Prikaz prijateljstev med uporabniki iz podatkovne množice Last.fm z uporabo grafa.

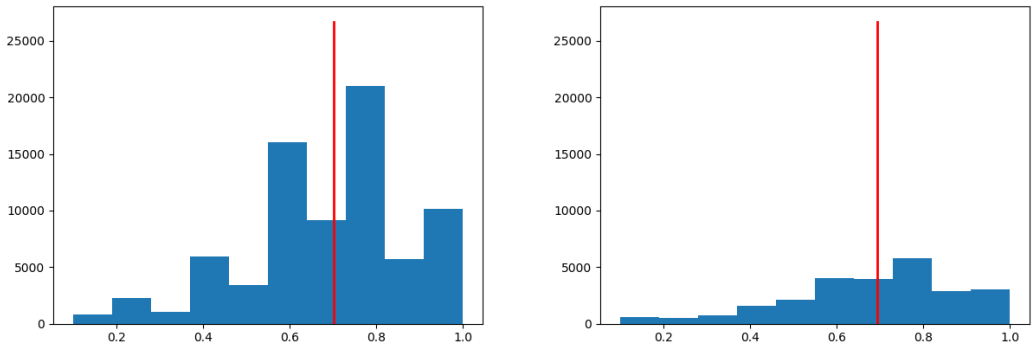
preslikamo na interval [0, 1] in vizualiziramo (glej sliko 5.3).

Povprečna ocena, označena z rdečo črto v grafu, je 0,7003, standardni odklon pa 0,2085. Zanimivo je, da uporabniki raje uporabljajo celoštevilske ocene, v histogramu 5.3 med celoštevilski ocenami namreč vidimo vrzeli, ki ustrezajo ocenam, ki se končajo z 0,5.

Podatke iz te množice smo razvrstili po naraščajoči vrednosti časovnega žiga, začetnih 75% vzeli kot učno množico, preostalih 25% pa kot testno in porazdelitev ocen vizualizirali na sliki 5.4. Opazimo lahko veliko spremembo v porazdelitvi ocen skozi čas (angl. *concept drift*). Ta pojav navadno negativno vpliva na uspešnost modelov, poleg tega pa se je z njim težko spoprijeti. V našem primeru bi lahko pristranskosti, namesto kot konstante, modelirali kot funkcije časa [20]. Poleg tega je v realnem okolju koristno spremljati uspešnost modela skozi čas.



Slika 5.3: Histogram z 10 intervali predprocesiranih ocen filmov iz podatkovne množice MovieLens. Povprečje 0,7003 je označeno z rdečo črto.



Slika 5.4: Levo je porazdelitev ocen iz učne množice, desno pa porazdelitev ocen iz testne množice MovieLens. Na obeh slikah je narisano povprečje z enako visoko rdečo črto. Opazimo lahko veliko spremembo v porazdelitvi podatkov skozi čas.

5.3 Metrike

Prva metrika, ki jo bomo uporabili za ocenjevanje učenja, je relativni koren srednje kvadratne napake (angl. *relative root mean squared error, RRMSE*). Za njen izračun moramo najprej definirati koren srednje kvadratne napake (angl. *root mean squared error, RMSE*):

$$rmse = \sqrt{\frac{1}{|\mathcal{R}|} \sum_{(u,i)\in\mathcal{R}} (r_{ui} - \hat{r}_{ui})^2}, \tag{5.1}$$

kjer je r_{ui} prava vrednost ocene, $\hat{r}_{ui}$ napovedana vrednost in $\mathcal{R}$ množica parov (u, i), za katere poznamo ocene. Metrika RMSE je bila uporabljena za ocenjevanje modelov v tekmovanju, ki ga je gostilo podjetje Netflix. Pogosto se uporablja tudi pri ocenjevanju regresijskih modelov. Kvadriranje povzroči, da metrika močneje kaznuje večja odstopanja. Pove nam, kako natančno v povprečju napovedujemo ocene oziroma kakšno napako lahko pričakujemo. Definirajmo še relativno napako RRMSE kot

$$rrmse = \frac{rmse}{rmse_{\mu}}, \tag{5.2}$$

kjer je v števcu za model izračunana metrika RMSE iz enačbe (5.1), v imenovalcu pa RMSE, ko za napoved ocene vedno uporabimo povprečno vrednost. Tako definirana relativna napaka nam nudi boljši vpogled v uporabnost modela.

Srednja kvadratne napaka je metrika, ki jo modeli na osnovi matrične faktorizacije direktno minimizirajo, sorazmerna je namreč negativnemu logaritmu verjetja. V verjetju modelov na osnovi Boltzmannovega stroja pa se ne pojavlja. V obeh primerih jo bomo uporabili za zgodnje ustavljanje.

V priporočilnih sistemih bi radu uporabniku priporočili le nekaj objektov, za katere upamo, da mu bodo všeč. Če napovedujemo števila poslušanj določenega izvajalca ali pa oceno filma, mu priporočimo le nekaj tistih z napovedano najvišjo oceno. Tako kot je metrika RMSE dobra za ocenjevanje splošne napake, nam tu ne odraža našega cilja. Kolikšna je napaka pri objektih, ki uporabniku niso všeč, nas ne zanima, razen če je seveda napaka tako velika, da mu posledično te objekte priporočimo. Bolj pomembno je torej točno napovedovanje objektov, ki so uporabniku všeč.

Naslednja uporabljena metrika je kombinacija natančnosti (angl. *precision*) in priklica (angl. *recall*). Uporaba natančnosti in priklica v spletnem oglaševanju je opisana v [13]. Označimo z N število priporočil, ki jih bomo prikazali uporabniku (kot na primer YouTube prikaže 12 splošnih priporočil na začetku strani). Označimo s $topN_u$ množico teh priporočil za uporabnika u, z rel_u pa množico uporabniku relevantnih objektov. Definirajmno metriko $pr@N$ kot

$$pr@N = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{|topN_u \cap rel_u|}{min\,(N, |rel_u|)}, \tag{5.3}$$

kjer je $\mathcal{U}$ množica vseh uporabnikov, za katere bomo merili uspešnost modela. V števcu smo izračunali število relevantnih objektov, ki smo jih priporočili. To vrednost nato delimo s številom objektov iz manjše množice.

Navadno v literaturi definirajo dve metriki, v eni delijo s številom priporočenih objektov, v drugi pa s številom relevantnih. Nam se zdi primernejša uporaba (5.2), ker imamo tako vedno možnost dobiti rezultat 1, če smo priporočili vse relevantne objekte. Poleg tega se zgodi, da imamo včasih ogromno relevantnih objektov, včasih pa le nekaj, priporočimo pa jih vedno le N.

Ima pa tudi ta metrika slabost, kajti relevantnih objektov po večini ne poznamo, če bi jih namreč sploh priporočilnega sistema ne bi potrebovali. V nadaljevanju bomo množico relevantnih objektov sestavili iz deleža najbolje ocenjenih ali največ poslušanih objektov iz testne množice. Nič nam pa ne jamči, da v testni množici ni objektov, ki so uporabniku izjemno všeč, pa jih še ni odkril. Vseeno pa nam lahko nudi vpogled v delovanje modela, pri čemer je pomembno, da izvedemo sistematičen način izračuna metrike na vseh modelih (na primer uporabimo enak N, enake delitve na učne in testne množice).

5.4 Rezultati

Modele iz prejšnjih poglavij preizkusimo na dveh podatkovnih množicah – Last.fm in MovieLens.

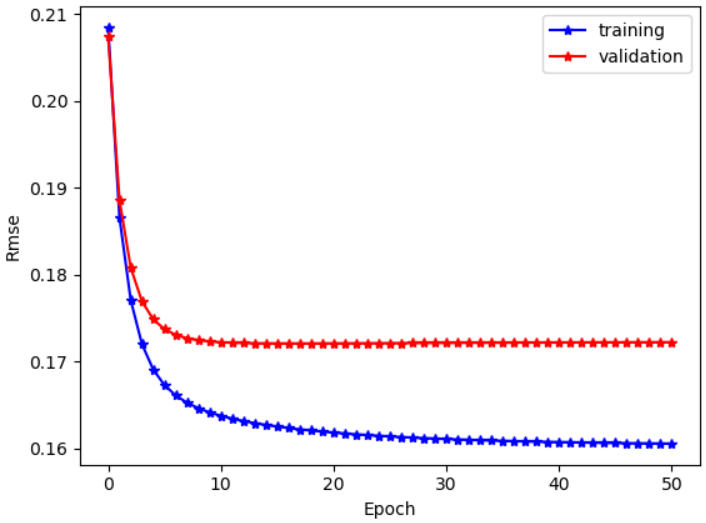
5.4.1 Rezultati na množici Last.fm

Najprej predstavimo tabelo 5.1, v kateri so zbrane uporabljene vrednosti parametrov algoritmov. Zapišemo le vrednosti, pri katerih so bili rezultati najboljši.

Posamezne zanimive izbire bomo komentirali v nadaljevanju.

Matrični razcep		Boltzmannov stroj	
regularizacija	$\lambda = 0{,}02$	regularizacija	$\lambda_w = 0{,}0025$
regularizacija	$\lambda_b = 0{,}01$	regularizacija	$\lambda_b = 0$
		regularizacija	$\lambda_c = 0{,}02$
stopnja učenja	$\alpha = 0{,}01$	stopnja učenja	$\alpha = 0{,}75$
eksponentni upad	$\rho = 0{,}94$	eksponentni upad	$\rho = 0{,}9998$
dim. latentnega prostora	$d = 9$	št. skritih sprem.	$d = 40$
		št. korakov Gibbs	$S = 1$
		zagon	$\gamma = 0{,}75$
		velikost skupin	$\lvert\mathbb{B}\rvert = 100$

Tabela 5.1: Tabela uporabljenih vrednosti parametrov učnih algoritmov na množici Last.fm.

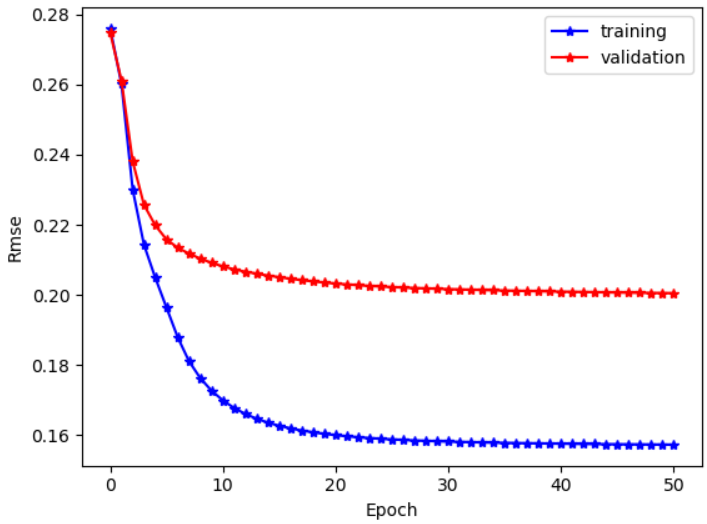


Slika 5.5: Prikaz metrike RMSE matričnega modela na učni in validacijski množici po vsakem obhodu podatkov Last.fm.

Poglejmo potek učenja matričnega modela na naboru podatkov Last.fm, torej

spreminjanja metrike RMSE na učni in validacijski množici v vsaki iteraciji (glej sliko 5.5). Slike so narejene ob eni delitvi podatkovne množice 75% na učno in preostalih 25% na validacijsko množico, brez prečnega preverjanja.

Vidimo lahko, da na validacijski množici RMSE precej zgodaj neha padati, vendar ne začne ponovno naraščati, kot smo to prikazali v poglavju o zgodnjem zaustavljanju. Nekaj zaslug za to lahko pripišemo dobro izbranim vrednostim parametrov, kot so stopnja regularizacije, dimenzija latentnega prostora in upad stopnje učenja. Kljub temu pa lahko opazimo precej veliko vrzel med napakama, izmerjenima na učni in validacijski množici, kar vseeno nakazuje na preveliko prileganje. Če se niti napaka na učni množici ne bi uspela dovolj znižati, pa bi pomenilo, da je kapaciteta modela premajhna.



Slika 5.6: Prikaz metrike RMSE modela RBM na učni in validacijski množici po vsakem obhodu podatkov Last.fm.

Ko pogledamo potek učenja modela na osnovi omejenega Boltzmannovega stroja lahko opazimo še večji razkorak med napako na učni in napako na testni množici (glej sliko 5.6). Napaka na učni množici je še manjša od napake, ki jo imajo matrični modeli. Tega razkoraka med učno in validacijsko množico pa nismo uspeli zmanjšati tudi z zelo močno regularizacijo in z zelo majhnim številom skritih kom-

ponent. V našem eksperimentu model RBM dobro rekonstruirajo podatke, slabo pa posplošujejo na še nevidenih podatkih. Izkaže se, da če modelu RBM pokažemo vektor ocen uporabnika $\boldsymbol{v}^{(new)}$, ki ga med učenjem še ni videl, ga ravno take zelo dobro rekonstruira. Zanimivo bi bilo primerjati njegovo uspešnost z uspešnostjo avtomatskih kodirnikov. Omenimo lahko, da so tudi avtorji članka [23] dobili slabši rezultati pri uporabi osnovnega modela RBM v primerjavi z matričnim razcepom. Njihov končni model, ki pa je imel rahlo nižjo vrednost RMSE, je vključeval številne dodatke, kot je pogojevanje na ocene iz učne množice in tiste, za katere so vedeli, da so v testni množici. Dodatno, matriko uteži $\boldsymbol{W}$ so aproksimirali s produktom dveh matrik nižjega ranga, tako so lahko uporabili večje število skritih spremenljivk.

Model	$rrmse$	$pr@N$
navaden matrični model	0,838945	0,028519
navaden matrični model z logistično f.	0,967499	/
matrični model z utežmi	0,838930	0,028519
matrični model s prijateljstvi in utežmi	0,838032	0,028516
omejen Boltzmannov stroj	0,970121	0,000730
omejen Boltzmannov stroj – rekonstr. novih $\boldsymbol{v}^{(new)}$	0,804345	/

Tabela 5.2: Prikaz rezultatov modelov na podatkovni množici Last.fm.

Poglejmo si tabelo 5.2, v kateri so zbrani rezultati različnih modelov. Prikažemo le modele, ki služijo za primerjanje, in nato nekaj najboljših oziroma najbolj zanimivih. Pri vseh modelih smo uporabili 4-kratno prečno preverjanje, ker bi 10-kratno prečno preverjanje potrebovalo preveč časa. Osnovno množico Last.fm smo torej razdelili na 4 dele, v vsakem koraku prečnega preverjanja pa smo 75% podatkov uporabili za učenje, ostalih 25% pa kot validacijsko množico za zgodnje ustavljanje, ocena napake pa je bila nato izračunana na preostalem delu iz delitve za prečno preverjanje.

Pri vseh modelih matričnega razcepa smo za aktivacijsko funkcijo uporabili identiteto id, če ne piše drugače. Ta funkcija je imela praktično enake rezultate kot $relu$. Logistična funkcija σ pa je rezultat v vseh poskusih precej poslabšala. Kot zanimivost, zadnja vrstica tabele 5.2 predstavlja napako pri rekonstrukciji

vektorjev $\boldsymbol{v}^{(new)}$, katerih nobene vrednosti nismo pokazali modelu med učenjem.

Vidimo lahko, da modeli na osnovi matričnega razcepa v nalogi napovedovanja ocen in priporočanja v dotičnem eksperimentu občutno boljši od modela RBM. Sicer pa ima model z upoštevanjem prijateljstev malenkost boljši RMSE od ostalih dveh. Metrika $pr@N$, kjer smo uporabili $N = 10$, je pri vseh matričnih modelih skoraj enaka in občutno boljša v primerjavi z modelom RBM. Za vsakega uporabnika smo nabor relevantnih objektov sestavili iz zgornje tretjine najbolje ocenjenih, tako se namreč lahko bolje izognemo pristranskostim ocenjevanja različnih uporabnikov.

5.4.2 Rezultati na množici MovieLens

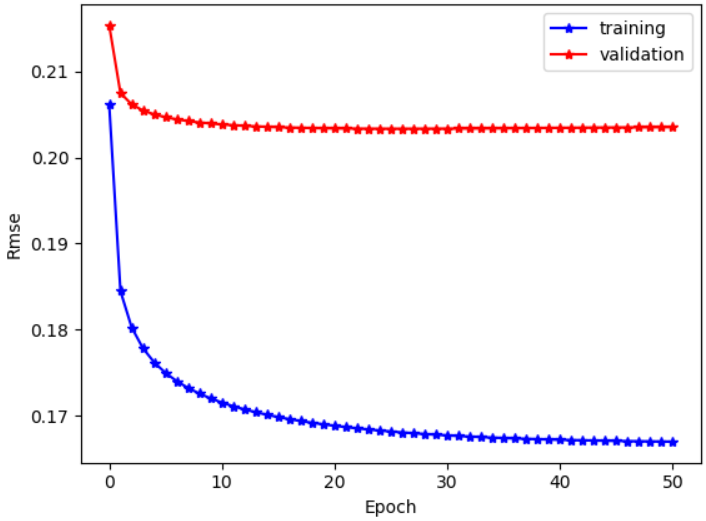
Zapišimo uporabljene vrednosti parametrov učnih algoritmov v tabelo 5.3, ki smo jih uporabili pri učenju na množici MovieLens. Tokrat je bila najprimernejša dimenzija latentnega prostora kar $d = 80$, kar je ogromna razlika v primerjavi s številom 9 ob učenju na podatkih Last.fm. Temu primerno pa smo povišali tudi vrednosti parametrov regularizacije.

Matrični razcep		Boltzmannov stroj	
regularizacija	$\lambda = 0{,}85$	regularizacija	$\lambda_w = 0{,}4$
regularizacija	$\lambda_b = 0{,}35$	regularizacija	$\lambda_b = 0{,}4$
		regularizacija	$\lambda_c = 0{,}25$
stopnja učenja	$\alpha = 0{,}012$	stopnja učenja	$\alpha = 1{,}15$
eksponentni upad	$\rho = 0{,}94$	eksponentni upad	$\rho = 0{,}97$
dim. latentnega prostora	$d = 80$	št. skritih sprem.	$d = 60$
		št. korakov Gibbs	$S = 2$
		zagon	$\gamma = 0{,}88$
		velikost skupin	$\lvert\mathbb{B}\rvert = 100$

Tabela 5.3: Tabela uporabljenih vrednosti parametrov učnih algoritmov na množici MovieLens.

Pri testiranju modelov na podatkovni množici MovieLens smo najprej trojke (u, i, r) razvrstili po naraščajoči vrednosti časovnega žiga, nato pa prvih 75% vzeli

za učno množico, preostalih 25% pa za testno. Iz slik 5.7 in 5.8, ki prikazujeta

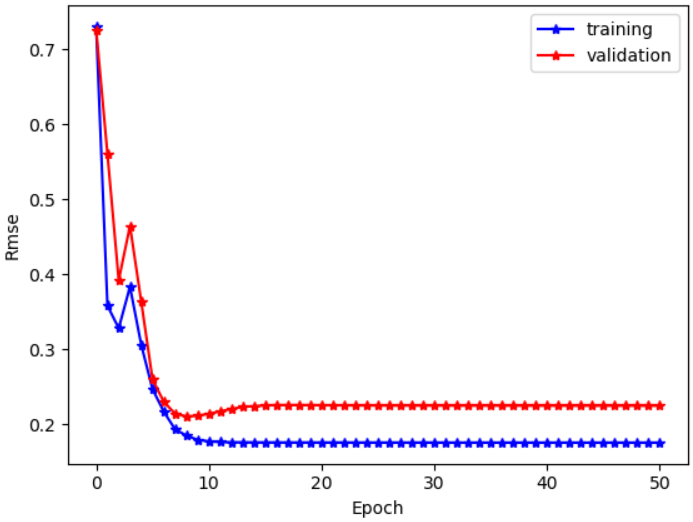


Slika 5.7: Prikaz metrike RMSE matričnega modela na učni in validacijski množici po vsakem obhodu podatkov MovieLens.

potek učenja, lahko v obeh primerih vidimo še večji razkorak med napako na učni in testni množici. Razlog je najbrž v tem, da se je porazdelitev podatkov skozi čas precej spremenila. Če uporabimo prečno preverjanje se namreč vrednost napake približa tisti iz Last.fm

V poteku učenja modela RBM vidimo, da sta napaki po nekaj korakih učenja poskočili, razlog za to je uporaba velike vrednosti zagona. Opazimo lahko, da se je napaka na testni množici po devetem koraku trajno povišala, to nakazuje na smiselnost zgodnjega ustavljanja.

Tabela 5.4 prikazuje vrednosti metrik obeh tipov modelov na tej podatkovni množici. Opazimo, da oba tipa dosežeta nižji RRMSE na množici Last.fm v primerjavi z množico MovieLens. Kljub temu pa imata oba modela za nekaj krat višjo vrednost metrike $pr@N$, razlog je najbrž v tem, da je delež znanih podatkov v množici MovieLens precej višji.



Slika 5.8: Prikaz metrike RMSE modela RBM na učni in validacijski množici po vsakem obhodu podatkov MovieLens.

Model	$rrmse$	$pr@N$
matrični model	0,944378	0,066558
omejen Boltzmannov stroj	0,977849	0,005634

Tabela 5.4: Prikaz rezultatov modelov na podatkovni množici MovieLens.

Poglavje 6

Sklepne ugotovitve

V diplomskem delu predstavimo in izpeljemo dva tipa modelov za napovedovanje ocen in priporočanje. Modele implementiramo v okolju Python z uporabo knižnjic NumPy in SciPy. V nadaljevanju bi bilo smiselno preizkusiti modele na večjih podatkovnih množicah oziroma še bolje, jih uporabiti v praktičnem okolju.

Modeli na osnovi matrične faktorizacije se v naših poskusih izkažejo za občutno boljše od modelov na osnovi omejenih Boltzmannovih strojev. Rezultati modelov RBM so po našem mnenju nekoliko pod pričakovanji oziroma pod mejo kompetitivnosti. Čeprav v model RBM nismo vključili vseh znanih razširitev menimo, da se splača več časa investirati v nadaljnji razvoj matričnih modelov, ki omogočajo tudi več fleksibilnosti.

Koristno bi bilo model matrične faktorizacije prilagoditi učenju iz časovno odvisnih podatkov in modelirati pristranskosti kot funkcije časa. Zanimivo bi bilo preizkusiti pristope iz [1, 21, 22], kjer za modeliranje ocen konstruirajo svojo porazdelitev in uporabijo Bayesovski pristop za samodejno določanje hiperparametrov, kot so stopnje regularizacije, iz podatkov. Začeli smo z eksperimentiranjem z uporabo globokih vložitev pri matričnih modelih. Torej funkcijo, ki preslika indeks uporabnika oziroma objekta v njegov latentni profil, smo parametrizirali kot globoko nevronsko mrežo. Začetki eksperimentov kažejo, da model tako dosega boljše rezultate pri posploševanju.

Model matrične faktorizacije smo tudi sami preizkusili tako, da smo iz podatkovne množice Last.fm izbrali nekaj izvajalcev, ki so nam všeč, in nekaj takšnih, ki jih ne maramo. Med priporočili je bilo nekaj dobrih in nam prej neznanih

izvajalcev, prav tako pa nismo dobili nobenega slabega priporočila.

Literatura

[1] Alex Beutel, Kenton Murray, Christos Faloutsos, and Alexander J. Smola. Cobafi: Collaborative bayesian filtering. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 97–108, New York, NY, USA, 2014. ACM.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, Berlin, Heidelberg, 2006.

[3] Miguel A. Carreira-Perpinan and Geoffrey E. Hinton. On contrastive divergence learning. 2005.

[4] Hung-Hsuan Chen. Weighted-svd: Matrix factorization with weights on the latent factors. *CoRR*, abs/1710.00482, 2017.

[5] Wikipedia contributors. Pareto distribution — wikipedia, the free encyclopedia. Dosegljivo: `https://en.wikipedia.org/wiki/Pareto_distribution`. [Dostopano 23. 02. 2019].

[6] Janez Demšar, Tomaž Curk, Aleš Erjavec, Črt Gorup, Tomaž Hočevar, Mitar Milutinovič, Martin Možina, Matija Polajnar, Marko Toplak, Anže Starič, Miha Štajdohar, Lan Umek, Lan Žagar, Jure Žbontar, Marinka Žitnik, and Blaž Zupan. Orange: Data mining toolbox in python. *Journal of Machine Learning Research*, 14:2349–2353, 2013.

[7] Thushan Ganegedara. Intuitive guide to understanding kl divergence. Dosegljivo: `https://towardsdatascience.com/light-on-math-machine-learning-intuitive-guide-to-understanding-kl-divergence-2b382ca2b2a8`. [Dostopano: 31. 12. 2018].

[8] Thushan Ganegedara. Optimizations of gradient descent. Dosegljivo: http://dsdeepdive.blogspot.com/2016/03/optimizations-of-gradient-descent.html. [Dostopano: 13. 02. 2019].

[9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[10] Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14(8):1771–1800, August 2002.

[11] Geoffrey E. Hinton. A practical guide to training restricted boltzmann machines. In *Neural Networks: Tricks of the Trade - Second Edition*, pages 599–619. 2012.

[12] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python. Dosegljivo: http://www.scipy.org/, 2001–. [Dostopano: 07. 03. 2019].

[13] Domen Košir. *Profiliranje spletnih uporabnikov v spletnem oglaševanju: doktorska disertacija*. PhD thesis, Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, 2015.

[14] Tuan-Ho Le. Early stopping method. Dosegljivo: https://www.researchgate.net/figure/Early-stopping-method_fig3_283697186. [Dostopano: 15. 02. 2019].

[15] Yann LeCun, Sumit Chopra, Raia Hadsell, Fu Jie Huang, and et al. A tutorial on energy-based learning. In *PREDICTING STRUCTURED DATA*. MIT Press, 2006.

[16] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[17] Andrew Ng. Gradient descent. Dosegljivo: https://www.slideshare.net/kandelin/optimizationgradient-descent. [Dostopano: 04. 02. 2019].

[18] Andrew Ng. Optimization algorithms. Dosegljivo: https://cs230.stanford.edu/files/C2M2.pdf. [Dostopano: 02. 03. 2019].

[19] Travis E. Oliphant. *Guide to NumPy*. CreateSpace Independent Publishing Platform, USA, 2nd edition, 2015.

[20] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor. *Recommender Systems Handbook*. Springer-Verlag, Berlin, Heidelberg, 1st edition, 2010.

[21] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, pages 1257–1264, USA, 2007. Curran Associates Inc.

[22] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 880–887, New York, NY, USA, 2008. ACM.

[23] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 791–798, New York, NY, USA, 2007. ACM.

[24] Dustin Stansbury. Maximum likelihood for boltzmann machines. Dosegljivo: `https://theclevermachine.wordpress.com/2014/09/23/derivation-maximum-likelihood-for-boltzmann-machines/`. [Dostopano: 14. 02. 2019].

[25] Grouplens team. Grouplens datasets. Dosegljivo: `https://grouplens.org/datasets/movielens/`. [Dostopano: 23. 12. 2018].

[26] Oliver Woodford. Notes on contrastive divergence. Dosegljivo: `http://www.robots.ox.ac.uk/~ojw/files/NotesOnCD.pdf`. [Dostopano: 14. 02. 2019].