SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 3947

PREDSTAVLJANJE SLIKA OGRANIČENIM BOLTZMANNOVIM NEURONSKIM MREŽAMA

Ivan Relić

Zagreb, lipanj 2015.

Sadržaj

1. Uvod

Računalni vid područje je umjetne inteligencije koje ugrubo možemo podijeliti na klasifikaciju i rekonstrukciju. Klasifikacijski dio bavi se utvrđivanjem i prepoznavanjem objekata koje slika sadrži i daljnjom obradom dobivenih saznanja. Rekonstrukcijski dio bavi se rekonstrukcijom 3D modela scene s obzirom na više slika same scene.

Područje ovog rada je predstavljanje slika u vektorskom prostoru niže dimenzionalnosti, tj. kompresija slika – dobivanje vjerne reprezentacije stvarne slike u što je manje moguće podataka. Prostor svih mogućih slika neke razdiobe ima puno manju dimenzionalnost od prostora svih mogućih slika svih mogućih razdioba koje su unutar $\mathbb{R}^{(w \cdot h)}$ prostora, gdje su *w* i *h* dimenzionalnosti vektora koji opisuju širinu i visinu slike. Upravo taj prostor svih mogućih slika neke razdiobe nalazi se na zakrivljenoj mnogostrukosti[1] $M \subset \mathbb{R}^{(w \cdot h)}$. Lokalne koordinate mnogostrukosti puno bolje opisuju udaljenost slika od globalnih koordinata nad $\mathbb{R}^{(w \cdot h)}$ prostorom. Upravo iz tog razloga opravdano je nadati se kako će, primjerice, klasifikacija biti preciznija u koordinatama mnogostrukosti nego u originalnom prostoru.

Preciznost preslikavanja između stvarnog skupa podataka i skupa podataka smanjene dimenzionalnosti možemo mjeriti tako da usporedimo stvarni skup podataka i skup podataka koji dobijemo rekonstrukcijom iz podataka smanjene dimenzionalnosti – što je pogreška manja, to je naše preslikavanje bolje.

Faze koje ovaj rad opisuje jesu opis matematičkog modela koji se koristi za preslikavanje između stvarnog skupa podataka (slike) i kompresiranog skupa podataka, odnosno slika u nižoj dimenzionalnosti. Nadalje, opisano je stvaranje takvog matematičkog modela i rezultati dobiveni njegovim korištenjem. Model koji će se koristiti za preslikavanje bit će ograničena Boltzmannova neuronska mreža koju će biti potrebno trenirati na skupu podataka izgledom sličnom onome kakav ćemo željeti koristiti prilikom preslikavanja stvarnih podataka u podatke niže

1 Mnogostrukost (engl. *manifold*) – apstraktan prostor u kojem točke lokalno imaju Euklidska svojstva, ali globalno to ne mora vrijediti; npr. dvodimenzionalna mnogostrukost sfere predstavljena je intrinsičnim koordinatama mnogostrukosti (Θ, Φ)

dimenzionalnosti . Iz tog razloga će treniranje i ispitivanje same neuronske mreže biti provedeno nad skupom slika rukom pisanih brojeva pod nazivom MNIST (eng. *Mixed National Institute of Standards and Technology database*). Uspješnost neuronske mreže pri postupku preslikavanja uvelike ovisi o njenom učenju nad podacima za učenje, ali je očekivana preciznost vrlo visoka. Učenje mreže bit će prikazano i na jednostavnijem skupu podataka – zašumljenim podacima s pravca, tj. točkama u dvije dimenzije koje ćemo svoditi na jednu dimenziju i pratiti kako se mreža ponaša i kakva je razdioba rekonstrukcijskih podataka tijekom učenja.

Korištenje ograničene Boltzmannove neuronske mreže za preslikavanje između stvarnih podataka i podataka u nižoj dimenzionalnosti je odabrano jer ima potencijal izraziti nelinearna preslikavanja između podataka. Koristeći nelinearna preslikavanja imamo mogućnost postići preslikavanje podataka u nižu dimenzionalnost uz manju rekonstrukcijsku pogrešku od nekog linearnog postupka. Ako bi koristili linearnu transformaciju, optimalan postupak za to bi bio PCA[2] jer je u mogućnosti to napraviti s najmanjom rekonstrukcijskom pogreškom [1]. Završna faza bit će uporaba dobivene mreže nad ispitnim podacima i usporedba slika rekonstruiranih iz podataka niže dimenzionalnosti s originalnim podacima.

2 PCA (engl. *Principal Component Analysis*) – postupak pretvorbe podataka u linearnu kombinaciju linearno nezavisnih vektora koji se nazivaju svojstvene komponente (engl. *principal components*) http://en.wikipedia.org/wiki/Principal_component_analysis

2. Algoritmi i metode

Model koji želimo razviti treba imati takva svojstva da računalnim postupkom preslikava stvarne podatke u podatke smanjene dimenzionalnosti koji dobro aproksimiraju stvarne podatke i omogućuju kvalitetnu rekonstrukciju. Također, želimo koristiti model koji koristi nelinearna preslikavanja između stvarnih podataka i podataka u smanjenoj dimenzionalnosti zbog moguće bolje ekspresivnosti, odnosno veće preciznosti u odnosu na ranije spomenut postupak PCA koji je najbolji postupak u domeni linearnih preslikavanja. Iz takvog zahtjeva proizlazi uporaba neuronske mreže zbog njenih zanimljivih svojstava.

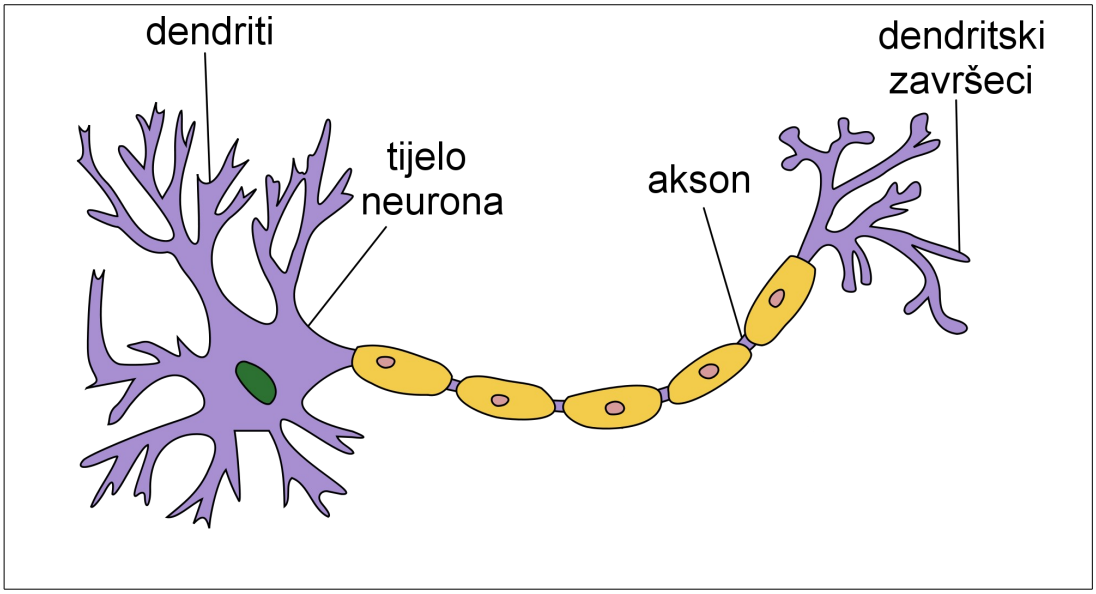
2.1. Neuronska mreža

Razvoj neuronskih mreža (punog naziva umjetne neuronske mreže) započeo je kao pokušaj da se objasni inteligentno ponašanje ljudi i životinja. Kako je jedan od načina objašnjenja zaključivanja i izvođenja novog znanja u umjetnoj inteligenciji simbolički pristup, a takav način funkcioniranja bioloških organizama nije dokazan do danas, temelj drugog pristupa objašnjenju funkcioniranja bioloških organizama naziva se *konektivizam*. Navedeni pristup temelji svoja objašnjenja inteligentnog ponašanja na tome da ono izranja iz velikog broja procesnih jedinica (neurona) u mozgu koji podatke obrađuju masovno paralelno. Primjerice, istraživanja ljudskog mozga pokazala su da se u njemu nalazi po prilici 10^{11} neurona koji su povezani s oko 10^{15} veza.

Neuronske mreže pokazale su se kao izuzetno uspješan model za aproksimaciju funkcija, klasifikaciju podataka, predviđanje trendova i slično. Posebno su zanimljive zbog robusnosti i otpornosti na šum te mogućnosti učenja pa su i danas u fokusu istraživanja i primjene.

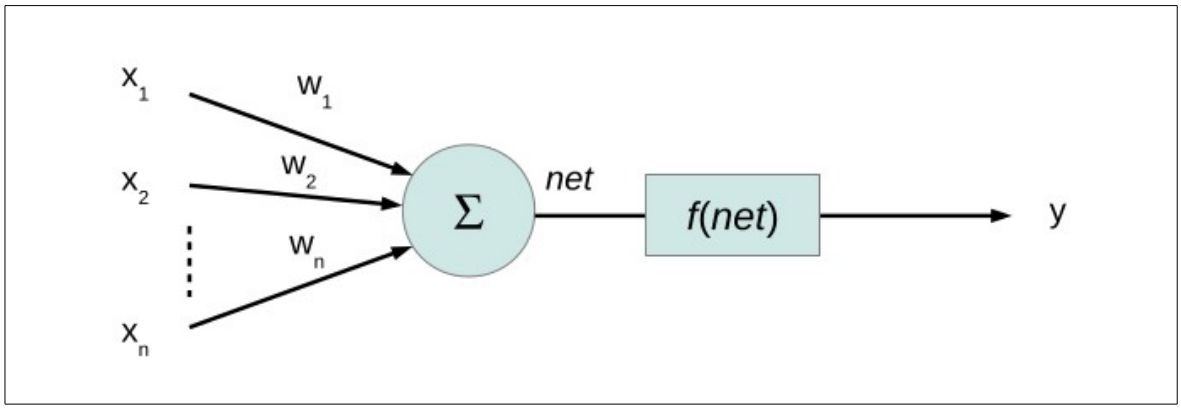
Biološki neuron se, pojednostavljeno, sastoji od dendrita, tijela neurona i aksona. Svaki je neuron, u prosjeku, povezan s 10^4 drugih neurona preko dendrita.

Neuron preko dendrita prikuplja električke impulse koji mijenjaju ukupni potencijal neurona. Nakon što se akumulira određena količina naboja (određena nekim pragom), neuron akumulirani naboj šalje kroz svoj akson prema drugim neuronima. Na taj način se električki impuls propagira kroz mrežu neurona – npr. mozak.



Slika 2.1.1. Građa biološkog neurona

Koristeći ovako pojednostavljeni model biološkog neurona, možemo definirati osnovni matematički model neurona. Model se sastoji od ulaza x_1 do x_n koji predstavljaju dendrite iz biološkog modela, težina w_1 do w_n koje određuju u kojoj se mjeri neuron pobuđuje za svaki od ulaza, sumatora u ulozi tijela neurona koje računa ukupnu pobudu svih ulaza i prijenosne funkcije u ulozi aksona koja ukupnu pobudu obrađuje i prosljeđuje na izlaz neurona.



Slika 2.1.2. Matematički model biološkog neurona [2]

Ukupna pobuda neurona *net* računa se prema sljedećem izrazu:

$$net = -\Phi + w1x1 + w_2x_2 + \ldots + w_nx_n \qquad (2.1.1.)$$

gdje Φ predstavlja prag okidanja neurona, ali se zbog lakše računalne implementacije tretira kao još jedna težina u opisu neurona. Prijenosna funkcija *f(net)* definira ponašanje neurona i u nastavku će biti dano nekoliko najkorištenijih prijenosnih funkcija.

Linearna funkcija definirana je na sljedeći način:

$$f(net) = k \cdot net + l \qquad (2.1.2.)$$

Korištenjem linearne funkcije u svim neuronima neuronske mreže ne možemo dobiti nikakvo složenije nelinearno preslikavanje jer se čitava mreža koristeći linearnu funkciju pretvara u jedan složeniji linearni element. Kako se u ovom radu koncentriramo na predstavljanje slika u nižoj dimenzionalnosti, ne želimo se ograničiti samo na linearne transformacije. Kao što je ranije navedeno, koristeći nelinearne transformacije imamo potencijal za postizanjem bolje kvalitete kompresije. Neke nelinearne prijenosne funkcije koje se u praksi koriste dane su u

nastavku:

Funkcija skoka (engl. *step function*) definirana je na sljedeći način:

$$f(net)=\begin{Bmatrix}0, net\leq 0\\ 1, inače\end{Bmatrix} \quad (2.1.3.)$$

ReLU (engl. *rectified linear unit*) funkcija definirana je na sljedeći način:

$$f(net)=max(0, net) \quad (2.1.4.)$$

Funkcija tangens hiperbolni definirana je na sljedeći način:

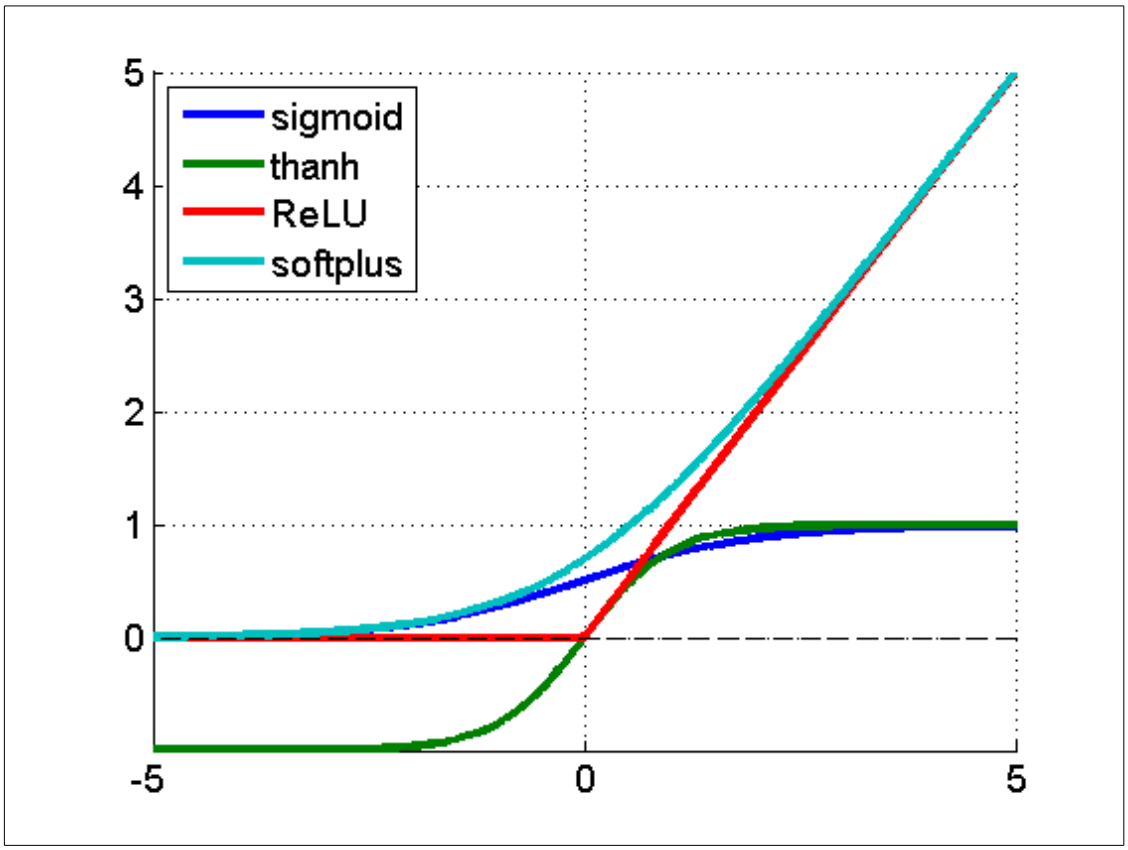
$$f(net)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \quad (2.1.5.)$$

Softplus funkcija definirana je na sljedeći način:

$$f(net)=\ln(1+e^{x}) \quad (2.1.6.)$$

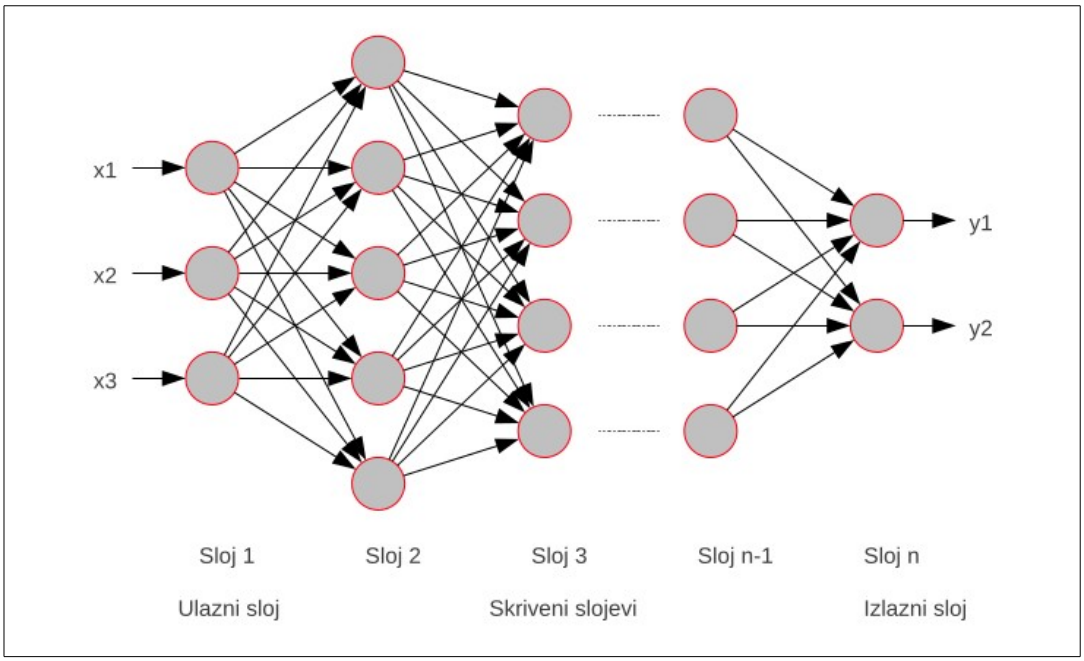
Sigmoidalna funkcija definirana je u nastavku i upravo ona će biti korištena kao prijenosna funkcija u ovom radu.

$$f(net)=\frac{1}{1+e^{-net}} \quad (2.1.7.)$$



Slika 2.1.3. Nelinearne prijenosne funkcije

Povezivanjem više matematičkih modela biološkog neurona u različite strukture dobivamo umjetnu neuronsku mrežu – baš kao što su i u prirodi biološki neuroni povezani u skupine i tvore upravljačku jedinicu.



Slika 2.1.4. Struktura umjetne neuronske mreže

Prikazana mreža sastoji se od ulaznog sloja (s lijeve strane) koji se sastoji od 3 neurona (ulaz mreže je 3-dimenzijski vektor podataka) i koji svoj ulaz samo prenose dalje u mrežu i ne obavljaju nikakvu obradu. Izlazni sloj sastoji se od 2 neurona (izlaz mreže je 2-dimenzijski vektor podataka) i rezultate svoje obrade daju kao izlaz korisniku mreže. Mreža ima i nekoliko sakrivenih slojeva koji obavljaju propagiranje i obradu podataka od ulaza do izlaza mreže.

2.2. Vjerojatnosni modeli i učenje vjerojatnosnih modela

Kako je model koji će se koristiti u ovom radu baziran na vjerojatnosnom modelu, u nastavku su ukratko objašnjeni neki pojmovi iz teorije vjerojatnosti važni za shvaćanje takvih modela.

2.2.1. Teorija vjerojatnosti

Pojam vjerojatnosti vežemo uz slučajne događaje čiji ishod unaprijed ne možemo znati. Ishod pokusa nazivamo elementarnim događajem i označavamo ga simbolom *ω.* Prilikom svakog pokusa mogu se dogoditi različiti događaji, a vjerojatnost je vrijednost koja nam govori s kojom sigurnošću će se događaj dogoditi, odnosno neće dogoditi. Što je događaj izvjesniji, to će njegova vjerojatnost biti bliža jedinici dok će događaji koji su malo vjerojatni imati vjerojatnost blisku nuli. Skup svih mogućih događaja nekog pokusa označavamo simbolom *Ω* i nazivamo ga sigurnim događajem.

2.2.1.1. Vjerojatnost i uvjetna vjerojatnost. Nezavisnost događaja

Vjerojatnost je preslikavanje

$$P: \mathcal{F} \rightarrow [0,1] \qquad (2.2.1.1.1.)$$

definirano na algebri događaja $\mathcal{F}$. Broj *P(A)* nazivamo vjerojatnost događaja *A*. Izvedimo dalje neka dodatna svojstva vjerojatnosti. Neka su *A* i *B* događaji, *A, B* $\subset$ *Ω.* Događaj koji se ostvaruje ako se ostvario barem jedan od događaja *A* ili *B* nazivamo unijom događaja i označavamo ga s *A* $\cup$ *B.* Događaj koji se ostvaruje ako su se ostvarili i događaj *A* i događaj *B* nazivamo presjekom događaja i označavamo ga s *A* $\cap$ *B.* Događaj koji se ostvaruje ako se ostvario događaj *A,* a nije se ostvario događaj *B* nazivamo razlikom događaja i označavamo ga s *A\B.* Iz navedenih izraza možemo napisati izraz za vjerojatnost unije događaja:

$$P(A \cup B) = P(A) + P(B) - P(A \cap B) \qquad (2.2.1.1.2.)$$

Uvjetna vjerojatnost je vjerojatnost nekog događaja uz uvjet da nam je poznato da se ostvario neki drugi događaj koji je povezan s događajem čiju uvjetnu vjerojatnost tražimo i koji je povoljan za njega. Uvjetnu vjerojatnost definiramo sljedećim izrazom:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \qquad (2.2.1.1.3.)$$

Izraz 2.2.1.1.3. predstavlja vjerojatnost da se dogodi događaj *A* uz uvjet da nam je poznato da se dogodio događaj *B.*

Koristeći uvjetnu vjerojatnost možemo pokazati i nezavisnost događaja. Dva su događaja nezavisna ako vrijedi $P(A|B) = P(A)$ ili $P(B|A) = P(B)$. Nadalje, nuždan i dovoljan uvjet da vrijedi da su dva događaja nezavisna je $P(A \cap B) = P(A)P(B)$.

Potpuna vjerojatnost je opis nekog događaja pomoću svih događaja s kojima taj događaj nije nezavisan. Događaje o kojima događaj kojeg opisujemo ovisi nazivamo hipotezama – $\{H_1, H_2, \ldots, H_n\}$. Potpunu vjerojatnost tada možemo izraziti pomoću tih hipoteza:

$$P(A) = \sum_{i=1}^{n} P(H_i) P(A|H_i) \qquad (2.2.1.1.4.)$$

Kada želimo znati vjerojatnost aposteriornih vjerojatnosti pojedinih hipoteza koristimo Bayesovu formulu. Nakon ishoda pokusa, nestaje neizvjesnost i znamo koji se događaj dogodio. Pretpostavimo da ne znamo koji se elementarni događaj dogodio, ali znamo da se dogodio događaj *A*. U tom slučaju ne znamo koja od hipoteza je nastupila, ali informacija da se dogodio događaj *A* mijenja apriorne vrijednosti pojedinih hipoteza i pomoću Bayesove formule dane u nastavku

računamo aposteriorne vrijednosti pojedinih hipoteza [3]:

$$P(H_i|A)=\frac{P(H_i)P(A|H_i)}{\sum_{j=1}^{n}P(H_j)P(A|H_j)} \qquad (2.2.1.1.5.)$$

2.2.1.2. Slučajne varijable, normalna razdioba i stohastički procesi

Slučajnom varijablom u teoriji vjerojatnosti nazivamo preslikavanje određeno sljedećim izrazom:

$$X:\Omega\rightarrow R \qquad (2.2.1.2.1.)$$

Funkcija razdiobe slučajne varijable *X* određena je sljedećim izrazom:

$$F(x):=P(X<x). \qquad (2.2.1.2.2.)$$

Funkcija gustoće razdiobe vjerojatnosti slučajne varijable *X* u točkama neprekinutosti definirana je sljedećim izrazom:

$$f(x):=\frac{dF(x)}{dx} \qquad (2.2.1.2.3.)$$

Koristeći funkciju gustoće ili funkciju razdiobe, možemo računati vjerojatnost da slučajna varijabla poprimi određen interval. Izraz u nastavku povezuje vjerojatnost s funkcijom razdiobe i funkcijom gustoće.

$$P(x_1 < X < x_2) = F(x_2) - F(x_1) = \int_{x1}^{x2} f(t)\,dt \qquad (2.2.1.2.4.)$$

Iz izraza 2.2.1.2.4. također slijedi da integral funkcije gustoće nad intervalom svih realnih brojeva mora iznositi 1.

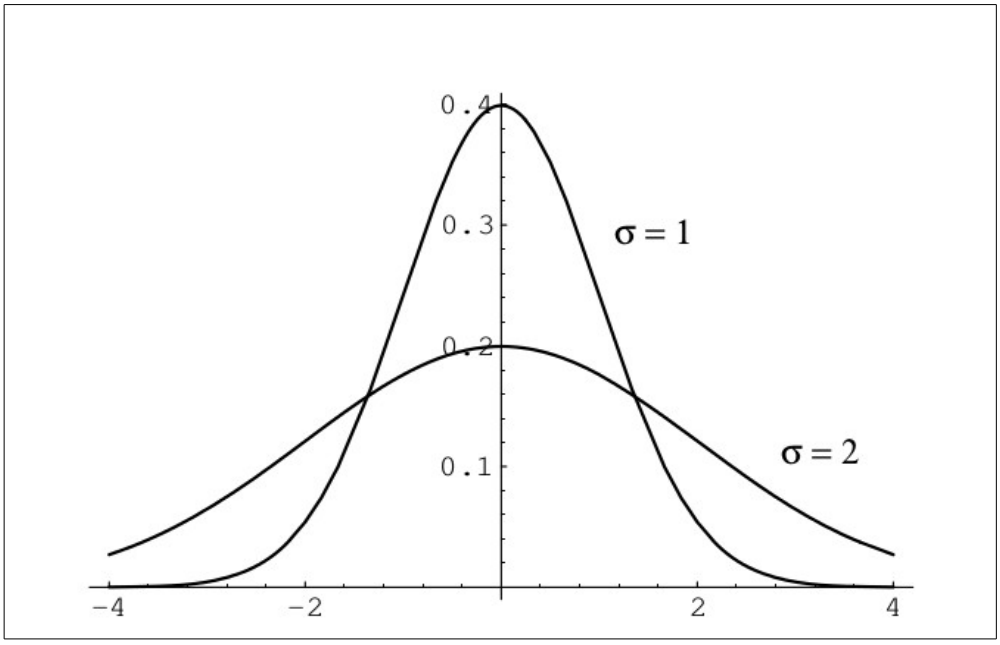
Normalna (Gaussova) razdioba jedna je od najvažnijih neprekinutih razdioba. Razlog važnosti normalne razdiobe leži u tome da je zbroj velikog broja međusobno nezavisnih varijabli upravo normalno distribuiran. Funkcija gustoće normalne razdiobe s parametrima *a* i σ^2 definirana je sljedećim izrazom:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \qquad (2.2.1.2.5.)$$

Graf funkcije gustoće normalne razdiobe zvonolika je krivulja koju nazivamo Gaussova krivulja. Zanimljiva činjenica kod normalne razdiobe je *pravilo tri sigma.* To pravilo govori da je vjerojatnost da varijabla čija je razdioba normalna poprimi vrijednost iz intervala ($a-3\sigma$, $a+3\sigma$) jednaka 0.9973, tj. 99.73%. Dobivena vrijednost računa se pomoću funkcije razdiobe:

$$P(|X - a < 3\sigma|) = F(3\sigma + a) - F(3\sigma - a) \qquad (2.2.1.2.6.)$$

Parametar *a* kod normalne razdiobe predstavlja očekivanje vrijednosti, a σ^2 predstavlja standardnu devijaciju vrijednosti slučajne varijable distribuirane po normalnoj razdiobi.



Slika 2.2.1.2.1. Graf funkcije gustoće normalne razdiobe uz parametar *a = 0*

Pojam slučajne varijable neovisan je o vremenu. Mnogi procesi koji se odvijaju u vremenu zahtijevaju da se koncept slučajne varijable poopći tako da uključuje i vremensku komponentu. Stohastički proces označava familiju slučajnih varijabli koja ovisi o vremenu. Skup *T* označava skup vremena u kojem se proces odvija, a skup *Ω,* kao i kod slučajnih varijabli, označava skup elementarnih događaja. Stohastičke procese možemo podijeliti u skupine na temelju prirode skupova *T* i *S.* Markovljevi procesi spadaju u skupinu stohastičkih procesa kojima su skupovi *T* i *S* diskretni [4] [5].

2.2.2. Markovljevi lanci prvog reda

Niz diskretnih slučajnih varijabli $X_0, X_1, X_2, \ldots, X_n$ zovemo stohastičkim lancem. Diskretne slučajne varijable kod Markovljevih lanaca uzimaju vrijednosti iz skupa S koji predstavlja moguća stanja u kojima se sustav može nalaziti. Te su varijable međusobno povezane i trenutno stanje sustava ima utjecaj na sljedeće stanje u vremenu. Stohastički lanac je Markovljev lanac prvog reda ako i samo ako stanje u budućnosti ovisi samo o sadašnjem stanju, a ne i o prošlim stanjima, tj. načinu na koji je sustav dospio u trenutno stanje. Zahtjev iskazujemo sljedećim izrazom:

$$P(X_{n+1}=i_{n+1}|X_n=i_n,\ldots,X_0=i_0)=P(X_{n+1}=i_{n+1}|X_n=i_n) \qquad (2.2.2.1.)$$

Veza između slučajnih varijabli X_{n+1} i X_n zadana je prijelaznim vjerojatnostima. Vjerojatnost prijelaza iz stanja i u stanje j određena je sljedećim izrazom:

$$p_{ij}:=P(X_{n+1}=j|X_n=i) \qquad (2.2.2.2.)$$

Prijelazne vjerojatnosti ne ovise o trenutku u kojem se prijelaz događa nego samo o stanjima u kojima se sustav nalazi. Matrica s elementima p_{ij} naziva se matrica prijelaznih vjerojatnosti i označava se s Π. Elementi ove matrice su nenegativni i zbroj svakog retka jednak je jedinici. Kako matrica Π daje vjerojatnosti prijelaza samo za jedan korak Markovljeva lanca, sljedeći izrazi daju vjerojatnosti da sustav prijeđe iz stanja i u stanje j u m koraka.

$$p_{ij}(m)=\sum_k p_{ik}(r)p_{kj}(m-r) \qquad (2.2.2.3.)$$

$$\Pi(m)=\Pi(r)\Pi(m-r) \qquad (2.2.2.4.)$$

$$\Pi(m)=\Pi^m \qquad (2.2.2.5.)$$

Izrazi vrijede za svaki *r = 1, 2, …, m – 1.*

Stanje u trenutku t_0 odgovara vjerojatnosti da sustav u trenutku $t=t_0$ poprimi vrijednost $i \in S$ i definirano je sljedećim izrazom:

$$p_i(0)=P(X_0=i) \qquad (2.2.2.6.)$$

Razdioba stanja u trenutku t_n određena je umnoškom razdiobe stanja u trenutku $t=t_0$ i n-tom potencijom matrice prijelaznih vjerojatnosti. Razdioba stanja u trenutku $t=t_n$ predstavljena je vektorom $p(n)=[p_i(n)], i \in S$.

Izrazi dani u nastavku opisuju razdiobu stanja sustava u trenutku t_n i vezu između razdioba stanja u uzastopnim vremenskim trenucima t_n i t_{n-1}.

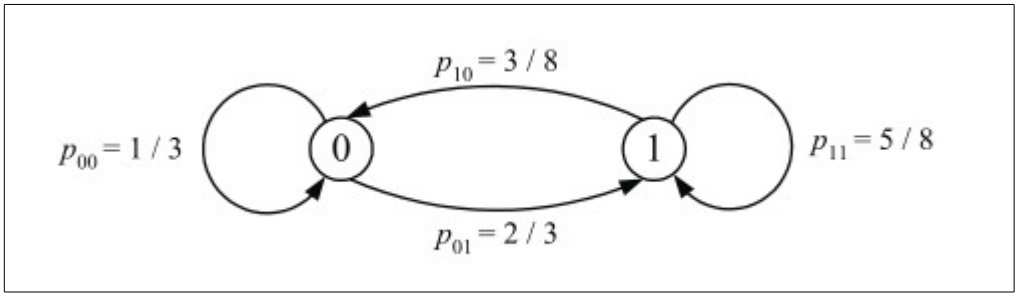
$$p(n)=p(0)\Pi^n \qquad (2.2.2.7)$$

$$p(n)=p(n-1)\Pi \qquad (2.2.2.8)$$

Stacionarna vjerojatnost je vjerojatnost da će se u nekom dalekom trenutku (kad se izgubi utjecaj početnog stanja) sustav nalaziti u stanju *j*. Ta se vjerojatnost može interpretirati i kao prosječni dio vremena koje sustav provodi u stanju *j*.

$$\pi_j=\lim_{n\to\infty} p_{ij}(n) \qquad (2.2.2.9.)$$

Markovljev lanac za kojeg izraz 2.2.2.9. postoji naziva se regularan. Stacionarna vjerojatnost π_j ne ovisi o vrijednosti izraza *i* i jedinstvena je za Markovljev lanac [5].



Slika 2.2.2.1. Primjer Markovljevog lanca

Na slici vidimo primjer Markovljevog lanca s dva stanja *S* = {0, 1}. Iz vjerojatnosti prijelaza među stanjima, lako je napisati matricu prijelaznih vjerojatnosti:

$$\Pi = \begin{bmatrix} 1/3 & 2/3 \\ 3/8 & 5/8 \end{bmatrix} \qquad (2.2.2.10.)$$

Želimo izračunati vektor stacionarnih vjerojatnosti za dani lanac. Prvi korak je pronaći *n* – tu potenciju matrice Π.

$$\Pi^n = \begin{bmatrix} \frac{9}{25} + \frac{1}{25}\left(\frac{-1}{3}\right)^n 2^{4-3n} & \frac{16}{25} - \frac{1}{25}\left(\frac{-1}{3}\right)^n 2^{4-3n} \\ \frac{9}{25} - \frac{1}{25}\left(\frac{-1}{8}\right)^n 3^{2-n} & \frac{16}{25} + \frac{1}{25}\left(\frac{-1}{8}\right)^n 3^{2-n} \end{bmatrix} \qquad (2.2.2.11.)$$

Iz izraza 2.2.2.11. lako je pronaći vrijednost limesa kada *n* teži u beskonačnost. Kako su oba retka jednaka, uzimamo bilo koji od njih i dobivamo vektor stacionarnih vjerojatnosti $\pi = \begin{bmatrix} \frac{9}{25} & \frac{16}{25} \end{bmatrix}^T$. Dobiveni vektor zapravo znači da će u nekom dalekom koraku u beskonačnosti Markovljev lanac biti u stanju 0 s vjerojatnosti 9/25, a u stanju 1 s vjerojatnosti 16/25.

2.2.3. Markovljeva slučajna polja

Neusmjereni grafovi određeni su uređenim parom *G = (V, E),* gdje je *V* konačan skup čvorova, a *E* je konačan skup neusmjerenih bridova grafa. Brid čini par čvorova iz skupa *V.* Ukoliko između dva čvora *v* i *w* postoji brid, tada čvor *v* pripada susjedstvu čvora *w*, i obratno, upravo zbog svojstva neusmjerenosti. Podgraf grafa *G* koji čine čvorovi od kojih su svaka dva susjedi naziva se klika. Maksimalnom klikom nazivamo onu kliku kojoj ne možemo dodati više niti jedan element, a da ona i dalje bude klika. Skup maksimalnih klika označavamo s *C.*

Neka je p_X razdioba vektorske slučajne varijable *X.* Grafičke modele možemo koristiti za reprezentaciju strukture združene razdiobe višedimenzionalne slučajne varijable. Za uređeni par (G, p_x) kažemo da je Markovljevo slučajno polje ako i samo ako je za svaki čvor $v \in V$ zadovoljeno lokalno Markovljevo svojstvo:

$$P(X_v | X_{V \setminus v}) = p(X_v | X_{N(v)}) \qquad (2.2.3.1.)$$

gdje je *N(v)* skup susjednih čvorova čvora *v.* Izraz 2.2.3.1. ustvari traži da je svaka komponenta *v* slučajne varijable *X* nezavisna o komponentama slučajne varijable *X* koje na grafu nisu susjedne s komponentom *v*.

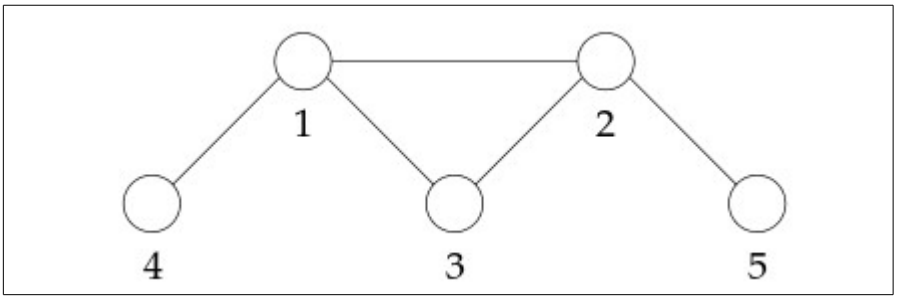
Razdioba vjerojatnosti nad neusmjerenim grafom naziva se Gibbsovom razdiobom ako se može faktorizirati pomoću pozitivnih funkcija definiranih preko ranije spomenutih klika koje pokrivaju sve čvorove i bridove grafa *G*. To možemo prikazati sljedećim izrazom:

$$P(X) = \frac{1}{Z} \prod_{c \in C} \phi_c(X_c) = \frac{1}{Z} e^{\sum_{c \in C} \ln \phi_c(X_c)} = \frac{1}{Z} e^{-E(x)} \qquad (2.2.3.2.)$$

gdje je *C* skup svih maksimalnih klika grafa *G,* a izraz *Z* predstavlja normalizacijsku konstantu određenu izrazom u nastavku. Funkciju *E(x)* nazivamo funkcijom energije.

$$Z = \sum_{x} \prod_{c \in C} \phi_c(X_c) = \sum_{x} e^{-E(x)} \qquad (2.2.3.3.)$$

Teorem *Hammersley – Clifford*[3] dokazuje da su Markovljeva slučajna polja i Gibbsova razdioba nad neusmjerenim grafom ekvivalentni [6].



Slika 2.2.3.1. Markovljevo slučajno polje

Markovljevo slučajno polje na slici 2.2.3.1. možemo prikazati preko ranije spomenutih funkcija klika, i to sljedećom faktorizacijom: $f_{123}(x_1,x_2,x_3)f_{14}(x_1,x_4)f_{25}(x_2,x_5)$. Takva faktorizacija je moguća jer čvorovi 1 i 4 čine jednu kliku, čvorovi 1, 2 i 3 čine jednu kliku i čvorovi 2 i 5 čine jednu kliku, odnosno ti čvorovi čine skup u kojem su svi čvorovi međusobno povezani.

3 Dokaz teorema *Hammersley – Clifford* http://www.vis.uky.edu/~cheung/courses/ee639/Hammersley-Clifford_Theorem.pdf

2.2.4. Učenje vjerojatnosnih modela

Učenje vjerojatnosnih modela može se bazirati na više pristupa. Pristup koji je zanimljiv za ovaj rad temelji se na maksimiziranju izglednosti (engl. *MLE*[4]). Uzimamo skup podataka za učenje $S = \{x_1, x_2, \ldots, x_n\}$ za koje je pretpostavljeno da su nezavisni i da pripadaju identičnom izvoru podataka s nekom nepoznatom razdiobom. Maksimiziranje izglednosti podešava parametre modela koji koristimo za prikazivanje podataka tako da poveća izglednost razdiobe modela. Primjerice, koristeći maksimiziranje izglednosti za Markovljeva slučajna polja (engl. *MRF – Markov Random Fields*), to odgovara pronalaženju parametara θ takvih da se minimizira razlika između nepoznate razdiobe skupa podataka za učenje i razdiobe p koja odgovara Markovljevom slučajnom polju koje koristimo kao vjerojatnosni model. Izglednost je definirana izrazom u nastavku koji želimo maksimizirati.

$$L(\theta|S)=\prod_{i=1}^{n} p(x_i|\theta) \qquad (2.2.4.1.)$$

Kako je maksimiziranje logaritma izglednosti identično maksimiziranju izglednosti, možemo tražiti i maksimum te funkcije što može biti računski efikasnije:

$$\ln(L(\theta|S))=\ln(\prod_{i=1}^{n} p(x_i|\theta))=\sum_{i=1}^{n} \ln(p(x_i|\theta)) \qquad (2.2.4.2.)$$

Za Gibbsovu razdiobu Markovljevih slučajnih polja u pravilu nije moguće pronaći parametre maksimalne izglednosti analitički pa se u takvim slučajevima radi gradijentni spust nad logaritmom izglednosti definiran sljedećim izrazom:

4 MLE – Maximum – likelihood estimation

$$\theta^{(t+1)} = \theta^{(t)} + \eta \frac{\partial}{\partial \theta^{(t)}} \left(\sum_{i=1}^{n} \ln\left(p\left(x_i | \theta^{(t)} \right) \right) \right) \qquad (2.2.4.3.)$$

Parametar η označava stopu učenja. Kasnije ćemo uvesti još neke dodatne parametre pri korištenju gradijentnog spusta za učenje modela koji koristimo u ovom radu i detaljnije ih objasniti.

Učenje modela koji se koristi u ovom radu koristi Gibbsovo uzorkovanje za aproksimaciju gradijenta funkcije logaritma izglednosti. Gibbsovo uzorkovanje je Monte Carlo metoda za Markovljeve lance. Gibbsovo uzorkovanje traži Markovljev lanac koji konvergira u stacionarno stanje sa željenom razdiobom koja odgovara razdiobi iz koje uzimamo podatke za učenje. Konvergencija u željenu razdiobu je moguća ako se prisjetimo izraza 2.2.2.9. koji govori da će sustav u nekom trenutku s određenom vjerojatnosti biti u stanju *j* sa željenom razdiobom (krenuvši iz nekog nasumično odabranog stanja). Osnovna ideja je osvježavati svaku varijablu na temelju uvjetne vjerojatnosti s obzirom na vrijednosti drugih varijabla. U praksi se izbor slučajne varijable koja mijenja stanje ne vrši nasumično koristeći razdiobu nad Markovljevim lancem nego se to vrši unaprijed određenim redom nad unaprijed određenim slučajnim varijablama i to se naziva periodičkim Gibbsovim uzorkovanjem [6]. Metoda kojom učimo naš model radi upravo takvo uzorkovanje jer se promjene vrše unutar 2 sloja, točno određenim redoslijedom.

3. Ograničena Boltzmannova neuronska mreža i algoritam kontrastne divergencije

U nastavku će biti objašnjena struktura i metode izračuna stanja modela koji koristimo za predstavljanje podataka u nižoj dimenzionalnosti. Također, bit će objašnjen praktičan postupak za učenje takvog modela nad nekim skupom podataka.

3.1. Ograničena Boltzmannova neuronska mreža

Ograničena Boltzmannova neuronska mreža (engl. *RBM – Restricted Boltzmann Machine*) je umjetna neuronska mreža koja se strukturno sastoji od samo dva sloja – vidljivog (ulaznog) binarnog sloja i skrivenog (izlaznog) binarnog sloja. Takva mreža ustvari pripada i skupu vjerojatnosnih modela baziranih na Markovljevim slučajnim poljima koja su objašnjena ranije – možemo ju predstaviti kao bipartitni graf. Zajedničko stanje vidljivog i skrivenog binarnog sloja (v, h) ima energiju koja se računa sljedećim izrazom:

$$E(v,h) = -\sum_{i \in visible} b_i v_i - \sum_{j \in hidden} c_j h_j - \sum_{i,j} v_i h_j w_{ij} \qquad (3.1.1.)$$

gdje su v_i i h_j binarna stanja ulazne komponente i i izlazne komponente j, b_i i c_j su ranije pojašnjeni pragovi *i-tog* i *j-tog* neurona (w_0 komponente), a w_{ij} je težina između ulazne i izlazne komponente. Ukoliko ograničenu Boltzmannovu mrežu promatramo kao grafički model, odnosno Markovljevo slučajno polje, izraz 3.1.1. možemo interpretirati kao funkciju nad klikama, kao i u izrazu 2.2.3.2. Mreža pridjeljuje združenu vjerojatnost svakom paru ulaznog i izlaznog vektora koristeći funkciju za izračun energije iz izraza 3.1.1.

$$p(v,h)=\frac{1}{Z}\cdot e^{-E(v,h)} \qquad (3.1.2.)$$

Združena vjerojatnost računa se izrazom 3.1.2. gdje je funkcija Z definirana na sljedeći način:

$$Z=\sum_{v,h} e^{-E(v,h)} \qquad (3.1.3.)$$

Vjerojatnost koju mreža pridjeljuje ulaznom vektoru definirana je zbrojem vjerojatnosti svih izlaznih vektora i definirana je na sljedeći način:

$$p(v)=\frac{1}{Z}\sum_{h} e^{-E(v,h)} \qquad (3.1.4.)$$

Vjerojatnost ulaznog vektora može se povećati tako da podesimo težinsku matricu w i pragove okidanja ulaznog i izlaznog sloja. Podešavanjem parametara želimo postići smanjenje energije ulaznog vektora i povećanje energija drugih vektora, posebno onih koji imaju nisku energiju i samim time visok utjecaj na vrijednost izraza Z (3.1.3.). Sniženje energije ulaznog vektora, odnosno povećanje energija drugih vektora radimo jer nekom izlaznom vektoru želimo pridružiti ulazni vektor koji mu zaista i odgovara, a sve ostale ulazne vektore razdiobe želimo sa što manjom vjerojatnošću.

Kako u ograničenoj Boltzmannovoj neuronskoj mreži nema direktnih veza između pojedinih ulaznih neurona, izračun vjerojatnosti da stanja ulaznih neurona poprime vrijednost 1 za dani izlazni vektor nije kompliciran i računa se sljedećim izrazom:

$$p(v_l=1|h)=\sigma\left(b_l+\sum_j h_j w_{lj}^T\right) \tag{3.1.5.}$$

Izraz 3.1.5. ustvari predstavlja uvjetnu vjerojatnost koja je definirana sljedećim izrazom:

$$p(v_l=1|h)=\frac{p(v_l=1,v_{-l},h)}{p(v_{-l},h)} \tag{3.1.6.}$$

Da bismo izračunali taj izraz, prvo ćemo definirati vektor v_{-l} kao vektor stanja svih ulaznih neurona osim *l* – tog. Također, definirat ćemo sljedeće izraze:

$$\alpha_l(h)=-\sum_i w_{il}h_i-b_l \tag{3.1.7.}$$

$$\beta(v_{-l},h)=-\sum_j\sum_{i\neq l} w_{ij}h_j v_i-\sum_{i\neq l} b_i v_i-\sum_j c_j h_j \tag{3.1.8.}$$

Energijsku funkciju sada možemo zapisati kao:

$$E(v,h)=\beta(v_{-l},h)+v_l\alpha_l(h) \tag{3.1.9.}$$

gdje $v_l\alpha_l(h)$ označava sva stanja, uključujući i stanje *l.*

Izraz 3.1.6. sada možemo raspisati kao:

$$\begin{aligned}
p(v_l=1|h)&=\frac{e^{-E(v_l=1,v_{-l},h)}}{e^{-E(v_l=1,v_{-l},h)}+e^{-E(v_l=0,v_{-l},h)}}=\\
&=\frac{e^{-\beta(v_{-l},h)-1\alpha_l(h)}}{e^{-\beta(v_{-l},h)-1\alpha_l(h)}+e^{-\beta(v_{-l},h)-0\alpha_l(h)}}=\\
&=\frac{e^{-\beta(v_{-l},h)}e^{-\alpha_l(h)}}{e^{-\beta(v_{-l},h)}e^{-\alpha_l(h)}+e^{-\beta(v_{-l},h)}}=\\
&=\frac{e^{-\beta(v_{-l},h)}e^{-\alpha_l(h)}}{e^{-\beta(v_{-l},h)}\left(e^{-\alpha_l(h)}+1\right)}=\\
&=\frac{e^{-\alpha_l(h)}}{e^{-\alpha_l(h)}+1}=\\
&=\frac{1}{1+e^{\alpha_l(h)}}
\end{aligned} \tag{3.1.10.}$$

iz čega je vidljivo kako to upravo odgovara izrazu 3.1.5.

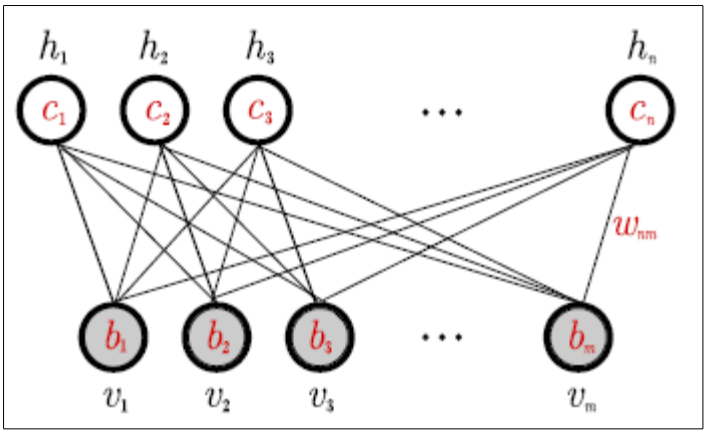
Kako u ograničenoj Boltzmannovoj neuronskoj mreži nema niti direktnih veza između pojedinih izlaznih neurona, izračun vjerojatnosti da stanja izlaznih neurona poprime vrijednost 1 za dani ulazni vektor također nije kompliciran i računa se prema sljedećem izrazu:

$$p\left(h_l = 1 \mid v\right) = \sigma\left(c_l + \sum_i v_i \cdot w_{il}\right) \qquad (3.1.11.)$$

Izvod izraza 3.1.11. sličan je izvodu izraza 3.1.5. iz razloga što je ograničena Boltzmannova neuronska mreža primjer modela baziranog na neusmjerenom grafu pa su izvodi simetrični [6].

U oba izraza, *σ(x)* predstavlja prijenosnu funkciju pojedinog neurona, a u slučaju ovog rada, kao što je već ranije navedeno, traži se nelinearno preslikavanje pa se koristi sigmoidalna prijenosna funkcija.

Podešavanje parametara neuronske mreže obavljamo kroz učenje i u nastavku će biti objašnjen jedan od načina učenja ograničene Boltzmannove neuronske mreže, a koji je vrlo brz i efikasan za većinu problema [7].



Slika 3.1.1. Neusmjereni graf ograničene Boltzmannove neuronske mreže

3.2. Algoritam kontrastne divergencije

Učenje modela koji koristimo u ovom radu temelji se na maksimiziranju izglednosti, odnosno minimiziranju razlike između razdiobe stvarnih podataka i razdiobe našeg modela za dane podatke. To možemo zamisliti kao Markovljev lanac koji započinje u trenutku $t=t_0$ nad razdiobom stvarnih podataka, a završava u trenutku $t=\infty$ s razdiobom koja odgovara stacionarnom stanju. Razliku između razdiobe stvarnih podataka i razdiobe našeg modela u stacionarnom stanju definiramo sljedećim izrazom:

$$F_0 - F_\infty \tag{3.2.1.}$$

gdje F_0 označava slobodnu energiju (engl. *free energy)* sustava kada koristimo stvarnu razdiobu podataka kao razdiobu modela, a F_∞ označava slobodnu energiju sustava kada koristimo razdiobu modela u stacionarnom stanju. Slobodna energija definira se kao razlika energije sustava i entropije razdiobe sustava:

$$F_t = E_t - S_t \tag{3.2.2.}$$

Minimiziranje razlike između razdiobe stvarnih podataka i razdiobe našeg modela odgovara maksimiziranju logaritma izglednosti modela. Za energijske grafičke modele gradijent logaritma izglednosti možemo prikazati preko energije modela kako slijedi:

$$\begin{aligned} \frac{\partial}{\partial \theta} L(\theta|v) &= \frac{\partial}{\partial \theta}\left(\ln \sum_h e^{-E(v,h)}\right) - \frac{\partial}{\partial \theta}\left(\ln \sum_{v,h} e^{-E(v,h)}\right) = \\ &= -\sum_h p(h|v) \frac{\partial E(v,h)}{\partial \theta} + \sum_{v,h} p(v,h) \frac{\partial E(v,h)}{\partial \theta} \end{aligned} \tag{3.2.3.}$$

Kako je vidljivo u izrazu 3.2.3., gradijent logaritma izglednosti sastoji se od razlike dvaju vjerojatnosti: očekivanih vrijednosti energijske funkcije nad razdiobom modela i nad uvjetnom vjerojatnosti izlaznog vektora za dani ulazni vektor, odnosno uzorak za učenje. Dio izraza 3.2.3. koji ovisi o razdiobi modela možemo aproksimirati razdiobom modela u stacionarnom stanju. Za računanje razdiobe u stacionarnom stanju potrebno je računati dovoljno dug Markovljev lanac koji će osigurati konvergenciju u stacionarno stanje. Kako je računska složenost takvog postupka prevelika, obavljaju se dodatne aproksimacije prilikom učenja modela.

Algoritam naziva kontrastna divergencija (engl. *contrastive divergence*) (*k*) je algoritam koji je osmislio Geoffrey Hinton [7]. Algoritam se bazira na aproksimaciji gradijenta logaritma izglednosti ograničene Boltzmannove neuronske mreže i to tako da umjesto računanja dovoljno dugog Markovljevog lanca koji bi osigurao konvergenciju modela u stacionarno stanje, obavlja Gibbsovo uzorkovanje od *k* koraka za aproksimaciju razdiobe modela. Uzorkovanje se obavlja na sljedeći način: inicijaliziramo Gibbsov lanac uzorkom za učenje $v^{(0)}$ i uzorkujemo sve dok ne dobijemo uzorak $v^{(k)}$. Jedan korak uzorkovanja sastoji se od toga da uzorkujemo vrijednosti $h^{(t)}$ iz vjerojatnosti $p(h|v^{(t)})$ i zatim uzorkovanja vrijednosti $v^{(t+1)}$ iz vjerojatnosti $p(v|h^{(t)})$.

Ovakvo relativno jednostavno Gibbsovo uzorkovanje posljedica je toga da se stanja svih varijabli u jednom sloju mogu uzorkovati združeno zbog neovisnosti varijabli unutar jednog sloja (varijable skrivenog sloja nisu međusobno povezane, isto tako niti varijable vidljivog sloja).

Izraz za gradijent logaritma izglednosti kod algoritma kontrastne divergencije tada se svodi na sljedeći izraz:

$$CD_k(\theta, v^{(0)}) = -\sum_h p(h|v^{(0)}) \frac{\partial E(v^{(0)}, h)}{\partial \theta} + \sum_{v,h} p(h|v^{(k)}) \frac{\partial E(v^{(k)}, h)}{\partial \theta} \quad (3.2.4.)$$

Vrijednost koja se nadodaje na trenutnu vrijednost težinske matrice *w* računa se derivacijom izraza 3.2.4. po parametru težine i definirana je sljedećim izrazom:

$$\Delta w_{ij} = \varepsilon \cdot \left(p\left(h_j = 1 \mid v^{(0)} \right) \cdot v_i^{(0)} - p\left(h_j = 1 \mid v^{(k)} \right) \cdot v_i^{(k)} \right) \qquad (3.2.5.)$$

Vrijednost koja se nadodaje pragovima okidanja ulaznog sloja računa se derivacijom izraza 3.2.4. po parametru pragova okidanja ulaznog sloja i definirana je sljedećim izrazom:

$$\Delta b_i = \varepsilon \cdot \left(v_i^{(0)} - v_i^{(k)} \right) \qquad (3.2.6.)$$

Vrijednost koja se nadodaje pragovima okidanja izlaznog sloja računa se derivacijom izraza 3.2.4. po parametru pragova okidanja izlaznog sloja i definirana je sljedećim izrazom:

$$\Delta c_j = \varepsilon \cdot \left(h_j^{(0)} - h_j^{(k)} \right) \qquad (3.2.7.)$$

U praksi se pokazalo da je za dobre rezultate za parametar *k* dovoljno odabrati vrijednost 1 pa je sam algoritam vrlo efikasan.

Parametar ε koji se pojavljuje u sva 3 izraza za osvježavanje parametara mreže naziva se stopa učenja (engl. *learning rate*). Nije preporučljivo pretjerati sa stopom učenja jer bi prilikom učenja mreže vrijednosti težinske matrice mogle naglo divergirati. Svakako, nije loše dinamički osvježavati stopu učenja – u početku je postaviti na neku početnu razinu i zatim je povećavati ili smanjivati nekom funkcijom za vrijeme učenja. Preporuka je započeti sa stopom učenja vrijednosti 0.1 [7].

Odluka koja se također mora donijeti je kada vršiti osvježavanje težinske matrice *w* i pragova okidanja neurona novim vrijednostima. To možemo raditi za

svaki ulazni vektor nad kojim učimo našu mrežu, možemo akumulirati vrijednost za sve ulazne vektore skupa podataka za učenje pa skalirati dobivenu vrijednost s brojem ulaznih vektora u skupu podataka ili možemo koristiti tzv. *mini – batch*, osvježavanje parametara nakon što smo akumulirali vrijednosti za osvježavanje nad nekim brojem ulaznih vektora iz skupa podataka za učenje. Preporuka je za *mini – batch* koristiti vrijednosti između 10 i 100 [7].

Još jedan parametar koji se uvodi naziva se težinska cijena (engl. *weight – cost*) i vezan je uz pojam smanjenja težina (engl. *weight – decay*). Smanjenje težina radi tako da dodaje dodatnu stavku u gradijent logaritma izglednosti neuronske mreže. Dodatna stavka koja se dodaje u gradijent je derivacija funkcije kazne koja kažnjava velike težine neuronske mreže. Funkciju kazne je važno pomnožiti sa stopom učenja, inače se promjenom stope učenja mijenja funkcija koja se optimira, a jedino što se treba promijeniti je način optimizacije. Jedna od najvećih dobiti ovog postupka je regularizacija. Regularizacija se odnosi na to da se uvodi dodatna informacija koja kažnjava kompleksnost rješenja, odnosno, postići ćemo to da težinski parametri imaju male apsolutne vrijednosti – čuvamo jednostavnost modela. Regularizacijom sprječavamo prenaučenost mreže. Mreža je prenaučena kada postiže vrlo dobre rezultate nad podacima za učenje, ali jako loše nad novim podacima koji su slični podacima za učenje, ali su različiti od njih. Također, korištenjem smanjenja težina dobivamo i to da “otkvačimo” neurone skrivenog sloja koji imaju jako velike težine pa su uglavnom ili uvijek uključeni ili uvijek isključeni. Jedna od preporučenih funkcija kazne za korištenje zove se *L2* funkcija koja je definirana sljedećim izrazom:

$$L2\left(w_{ij}\right)=\varepsilon\cdot w_{ij}^{2}\cdot 0.5\cdot\kappa \qquad (3.2.8.)$$

Kako se gradijentu dodaje derivacija funkcije kazne, potrebno je derivirati funkciju određenu izrazom 3.2.4. tako da se dobije sljedeći izraz:

$$L2'\left(w_{ij}\right)=\varepsilon\cdot w_{ij}\cdot\kappa \qquad (3.2.9.)$$

κ je novouvedeni parametar težinske cijene i preporuka je da bude između vrijednosti 0.01 i 0.00001.

Još jedan koristan parametar koji uvodimo u algoritam za učenje je *momentum*. Momentum ili inerciju koristimo za modifikaciju smjera pretraživanja algoritma, tj. ne krećemo se direktno u smjeru aproksimacije gradijenta. To daje sustavu za učenje vremena da se koristeći manje početne vrijednosti momentuma giba parametarskim prostorom prije nego se vrijednost momentuma poveća i započne se usmjerena pretraga. Takva modifikacija omogućuje izbjegavanje zaglavljivanja u lokalnim optimumima, bržu konvergenciju i omogućuje korištenje većih stopa učenja. Modifikacija je definirana sljedećim izrazom:

$$v^{(new)} = \alpha v^{(old)} + \Delta(\theta) \qquad (3.2.10.)$$

gdje je $v^{(new)}$ nova vrijednost koja se nadodaje parametru mreže koji se osvježava, α je iznos momentuma, $v^{(old)}$ je stara vrijednost koja se dodavala parametru mreže pri prošlom osvježavanju, a izraz $\Delta(\theta)$ je neki od prijašnje definiranih izraza: $\Delta w, \Delta b, \Delta c.$ Preporuka je za početna osvježavanja parametara držati momentum na vrijednosti od 0.5, a kasnije na vrijednosti 0.9 [7].

Konačno, uzimajući u obzir sve parametre i stavke algoritma, možemo napisati pseudokod:

Algoritam 1 Kontrastna divergencija(1)

```
dok nije zadovoljen uvjet zaustavljanja radi {
 za svaki ulazni vektor iz skupa podataka za učenje {
     ako si osvježio parametre batch_size puta {
         dodaj weight_decay težinskim parametrima
         dodaj momentum parametrima
         dodaj parametre parametrima mreže
         resetiraj parametre
     }
     izračunaj skriveni0 iz ulaznog vektora
     izračunaj ulazni1 iz skriveni0
     izračunaj skriveni1 iz ulazni1
     osvježi parametre težina sa {
         ulazni vektor * transponirano(skriveni0) – ulazni1 *
         transponirano(skriveni1)
     }
     osvježi parametre ulaznih pragova okidanja sa {
         ulazni vektor – ulazni1
     }
     osvježi parametre izlaznih pragova okidanja sa {
         skriveni0 – skriveni
     }
 }
 osvježi momentum
}
```

Uvjet zaustavljanja učenja mreže može se odnositi na broj iteracija koji smo spremni čekati ili na minimalnu rekonstrukcijsku pogrešku za koju smo spremni prihvatiti naučenu mrežu. U pseudokodu se vide svi ranije navedeni elementi – osvježavanje parametara mreže tek nakon *batch_size* osvježavanja lokalnih parametara gradijentom između ulaznih podataka i rekonstrukcije, dodavanje *weight_decay* korekcije težinskim parametrima prije nego se lokalni parametri dodaju parametrima mreže te osvježavanje vrijednosti momentuma nakon što se završi jedna epoha (nakon što se prođu svi ulazni vektori skupa podataka za učenje).

4. Podaci

Učenje i ispitivanje mreže u ovom radu bazira se na skupu podataka MNIST [8] (engl. *Mixed National Institute of Standards and Technology*). Iz navedenog skupa podataka koristit ćemo 2 datoteke: *train-images-idx3-ubyte* koja predstavlja 60000 slika za učenje i *t10k-images-idx3-ubyte* koja sadrži 10000 slika za ispitivanje. Obje datoteke su pohranjene u IDX formatu koji je definiran tako da prvo dolazi 4 bajta podataka i to *magic* – *number*, zatim 4 bajta podataka koji definiraju broj slika, sljedećih 4 bajta definira broj redaka svake slike, sljedećih 4 bajta definira broj stupaca svake slike i svi podaci koji slijede predstavljaju podatke samih slika.

Slike su predstavljene vrijednostima piksela od 0 – 255 gdje vrijednost 0 označava bijelu boju, a 255 crnu.

4.1. Učitavanje i normalizacija skupa podataka

Učitavanje skupa podataka temelji se na čitanju datoteke koja sadrži slike i formatiranja u prikladne strukture podataka. Kako se na ulaz ograničene Boltzmannove neuronske mreže postavlja binarni vektor ili vektor realnih vrijednosti iz intervala [0, 1] (takve realne vrijednosti ustvari će predstavljati vjerojatnosti da neki neuron bude u stanju 1), prikladna struktura za pohranjivanje svake pojedine slike bit će niz brojeva koji sadrži vrijednosti između 0 i 1. Sljedeći algoritmi će težiti tome da na kraju svaku sliku imamo predstavljenu kao jedan niz vrijednosti između 0 i 1.

Algoritam 2 Učitavanje skupa podataka

```
otvori datoteku(putanja do dataseta)
pročitaj broj slika, broj redaka i broj stupaca u svakoj
 slici iz datoteke
pročitaj podatke koji predstavljaju slike
podijeli pročitane podatke na blokove veličine broj redaka *
 broj stupaca
vrati kreirane blokove
```

Algoritmu predajemo putanju do datoteke koja sadrži skup podataka i natrag dobivamo skup nizova od kojih svaki predstavlja jednu sliku.

Algoritam 3 Normalizacija skupa podataka

```
za svaki blok koji predstavlja sliku {
 minimum = min(blok)
 za svaki element bloka {
     element -= minimum
 }
 maksimum = max(blok)
 za svaki element bloka {
     element /= maksimum
 }
}
```

Algoritam normalizira skup podataka na vrijednosti iz intervala [0, 1] na način da se pronađe minimalni element bloka podataka koji se zatim oduzme svakom elementu bloka. Vrijednosti koje tada imamo su iz intervala [0, *x*] gdje je *x* najveći element tog bloka. Svaki element bloka podataka zatim dijelimo elementom *x* da dobijemo vrijednosti iz intervala [0, 1].

5. Programska implementacija

Programska implementacija ostvarena je u programskom jeziku Python uz korištenje poznatog skupa biblioteka za korištenje u znanosti – SciPy. Skup biblioteka SciPy sadrži nekoliko biblioteka, a nama zanimljive su NumPy, SciPy i Matplotlib. Biblioteka NumPy je korištena zbog podrške za višedimenzionalne nizove i brze operacije nad takvim nizovima čije brzine izračuna su sumjerljive onima koje bi bile kada bismo ih implementirali u jeziku C. Biblioteke SciPy i Matplotlib korištene su za pohranjivanje slika i prikazivanje raznih podataka na grafu. Skup biblioteka SciPy je otvorenog koda i dostupan je za korištenje besplatno. Upute za instalaciju skupa biblioteka SciPy dostupne su na stranici http://www.scipy.org/install.html.

Konačna implementacija bit će sadržana u dvije datoteke. Program sadržan u prvoj datoteci će učitati i normalizirati skup podataka za učenje, kreirati ograničenu Boltzmannovu neuronsku mrežu s nasumično odabranim početnim parametrima, pokrenuti učenje algoritmom kontrastne divergencije (1) i pohraniti naučene parametre. Program sadržan u drugoj datoteci će učitati parametre mreže, kreirati mrežu i postaviti joj učitane parametre, učitati i normalizirati skup podataka za ispitivanje, nad učitanim podacima provjeriti učinkovitost naučene mreže i generirati rekonstrukcijske podatke iz podataka smanjene dimenzionalnosti uz izvještaje o minimalnim i maksimalnim pogreškama te razdiobom pogreške. Kao pomoć su korištene implementacije [9] i [10].

5.1. Ograničena Boltzmannova neuronska mreža

Implementacija ograničene Boltzmannove neuronske mreže dana je u jednoj datoteci – *rbm.py.* Sama datoteka *rbm.py* sadrži i jedan razred – *RBM* koji predstavlja ograničenu Boltzmannovu neuronsku mrežu. U nastavku će biti dan dijagram razreda i kratak opis pojedinih funkcija.

RBM
+number_of_visible +number_of_hidden +visible_biases +hidden_biases +weight_matrix +activation_matrix
+__init__(self, visible_biases, hidden_biases, weight_matrix, activation_function) +visible_states_to_hidden_states(self, visible_states) +visible_states_to_hidden_probabilities(self, visible_states) +hidden_states_to_visible_states(self, hidden_states) +hidden_states_to_visible_probabilities(self, hidden_states)

`__init__(self, visible_biases, hidden_biases, weight_matrix, activation_function)` – konstruktor razreda *RBM* koji prima pragove okidanja za vidljive neurone, pragove okidanja za sakrivene neurone, matricu težina između vidljivih i skrivenih neurona te prijenosnu funkciju koja se koristi za izračun vjerojatnosti, odnosno stanja vidljivih i skrivenih neurona

`visible_states_to_hidden_states(self, visible_states)` – funkcija koja za dane vrijednosti vidljivih neurona vraća vjerojatnosti i stanja pridružena skrivenim neuronima

`visible_states_to_hidden_probabilities(self, visible_states)` – funkcija koja za dane vrijednosti vidljivih neurona vraća vjerojatnosti pridružene skrivenim neuronima

`hidden_states_to_visible_states(self, hidden_states)` – funkcija koja za dane vrijednosti skrivenih neurona vraća vjerojatnosti i stanja pridružena vidljivim neuronima

`hidden_states_to_visible_probabilities(self, hidden_states)` – funkcija koja za dane vrijednosti skrivenih neurona vraća vjerojatnosti pridružene vidljivim neuronima

Kako konstruktor razreda *RBM* kao argument prima i prijenosnu funkciju, a u ovom radu se za prijenosnu funkciju koristi sigmoidalna funkcija, njena implementacija je dana u datoteci *activation_functions.py*. Implementacija sigmoidalne funkcije je kako slijedi:

```
def sigmoid_function(array):
 return 1/(1+numpy.exp(-array))
```

Funkcija kao argument prima niz nad kojim ćemo za svaki element izračunati vrijednost sigmoidalne funkcije.

5.2. Algoritam kontrastne divergencije (1) nad ograničenom Boltzmannovom neuronskom mrežom

Implementacija algoritma kontrastne divergencije (1) dana je u jednoj datoteci – *contrastive_divergence.py.* Sama datoteka *contrastive_divergence.py* sadrži i jedan razred – *ContrastiveDivergence* koji predstavlja algoritam za učenje ograničene Boltzmannove neuronske mreže – kontrastnu divergenciju (1). U nastavku će biti dan dijagram razreda i kratak opis pojedinih funkcija.

ContrastiveDivergence
+dataset +batch_size +max_epochs +learning_rate +momentum_updater +momentum +old_weight_updates +old_visible_biases_updates +old_hidden_biases_updates +weight_decay_calculator +rbm_updated_notifier
+__init__(self, dataset, batch_size, learning_rate, momentum_updater, weight_decay_calculator, max_epochs, rbm_updated_notifier) +train(self, rbm) +update_rbm(self, rbm, weight_updates, batched_size, visible_biases_updates, hidden_biases_updates)

`__init__(self, dataset, batch_size, learning_rate, momentum_updater, weight_decay_calculator, max_epochs, rbm_updated_notifier)` – konstruktor razreda *ConstrastiveDivergence* koji prima skup podataka za učenje ograničene Boltzmannove neuronske mreže, veličinu *batch* nakon koje se radi osvježavanje parametara mreže, stopu učenja, objekt *momentum_updater* koji zna vratiti sljedeći iznos ranije spominjanog momentuma koji će se koristiti, objekt *weight_decay_calculator* koji će znati izračunati vrijednost ranije spominjanog postupka smanjenja težina prilikom osvježavanja parametara mreže, maksimalni broj epoha i objekt *rbm_updated_notifier* koji će biti obaviješten na kraju svake epohe

`train(self, rbm)` – funkcija uči predanu mrežu s parametrima algoritma koji su postavljeni kroz konstruktor

`update_rbm(self, rbm, weight_updates, batched_size, visible_biases_updates, hidden_biases_updates)` – funkcija koja osvježava parametre predane mreže s predanim podacima – vrijednosti koje treba nadodati težinskoj matrici mreže, pragovima okidanja vidljivih i pragovima okidanja skrivenih neurona; dodatan parametar je *batched_size* koji govori koliko podataka je mreži predočeno od zadnjeg osvježavanja parametara

Kako kroz konstruktor razreda *ConstrastiveDivergence* predajemo nekoliko objekata čije ponašanje nije striktno definirano i kako je programski jezik Python strogo dinamički tipiziran, ovdje govorimo o strategiji. Objekt *momentum_updater* mora imati definiranu metodu `next_value()` kojom se dohvaća sljedeća vrijednost momentuma nakon svake epohe u postupku učenja. Objekt *weight_decay_calculator* mora imati definiranu metodu `calculate_weight_decay(weight_matrix, learning_rate)` koja se poziva u trenutku kada se osvježavaju parametri mreže i potrebno je izračunati vrijednost smanjenja težina. Funkcija kao parametre prima trenutnu težinsku matricu mreže i trenutnu stopu učenja a vraća vrijednosti koje treba oduzeti od vrijednosti koje će se pribrojiti težinskoj matrici mreže. Objekt

rbm_updated_notifier mora imati metodu `rbm_updated(rbm, epoch, error)` koja se poziva nakon svake epohe u postupku učenja. Funkcija kao parametre prima mrežu koja se uči, indeks trenutne epohe i ukupnu apsolutnu rekonstrukcijsku pogrešku trenutne epohe nad podacima za učenje. Svrha joj je klijentu ponuditi da u postupku učenja dohvaća i prikazuje neke podatke mreže. Konkretni objekti koji se koriste u ovom radu bit će prikazani u nastavku.

5.3. Učenje mreže nad MNIST skupom podataka

Učenje i evaluacija nad MNIST skupom podataka izvedeno je kroz 2 datoteke – *train_mnist_rbm.py* i *existing_mnist_rbm.py*. Prvi program omogućuje nam učenje ograničene Boltzmannove mreže algoritmom kontrastne divergencije (1). Pokretanje učenja mreže obavlja se na sljedeći način: smjestimo se u direktorij u kojem se nalazi datoteka *train_mnist_rbm.py* i u komandnu liniju unesemo naredbu *python "*train_mnist_rbm.py" "train_mnist_rbm.cfg" (bez navodnika),gdje je *train_mnist_rbm.cfg* putanja do konfiguracijske datoteke iz koje datoteka za učenje mreže učitava parametre. Format konfiguracijske datoteke je sljedeći:

```
[configuration]
train_dataset_location = putanja do datoteke sa skupom podataka za učenje
learning_weights_save_location = putanja za pohranjivanje težinske matrice u
                                 obliku slika tijekom postupka učenja
pickle_save_folder = putanja za pohranjivanje naučenih parametara
weights_random_mean = očekivanje za početne vrijednosti težinske matrice
weights_random_stddev = devijacija za početne vrijednosti težinske matrice
lowered_dimensionality_components = broj komponenata u koje preslikavamo
                                    podatke iz MNIST skupa podataka
mini_batch_size = veličina batch za osvježavanje parametara
learning_rate = stopa učenja mreže
max_epochs = maksimalni broj epoha
starting_momentum = početni momentum koji se održava prvih
                    momentum_change_step epoha
ending_momentum = završni momentum koji se održava ostatak epoha
momentum_change_step = indeks epohe na kojoj se događa promjena
                       momentuma s početnog na završni
weight_cost = parametar težinske cijene za računanje smanjenja težina
```

Program za učenje mreže parsira predanu datoteku i koristi ju za inicijalizaciju algoritma za učenje i početnih vrijednosti parametara mreže. Putanja za

pohranjivanje težinske matrice u obliku slika će na kraju svake epohe spremiti težinsku matricu mreže u niz slika tako da je moguće vidjeti kako model napreduje i interpretirati parametre težinske matrice kroz izvođenje učenja. Također, na kraju svake epohe ispisuje se pogreška između stvarnih i rekonstruiranih podataka nad skupom podataka za učenje. Program kao prijenosnu funkciju koristi ranije spominjanu sigmoidalnu funkciju koja se uz početne vrijednosti parametara postavlja pri inicijalizaciji mreže. Nakon što je inicijalizirana mreža i algoritam učenja, učitava se skup podataka za učenje mreže i normalizira se, pokreće se učenje mreže i nakon što završi, na lokaciju koja je zapisana u konfiguracijskoj datoteci pohranjuju se naučeni parametri mreže koji se mogu koristiti u drugom programu - *existing_mnist_rbm.py.*

Drugi program pokrećemo slično kao i prvi: smjestimo se u direktorij u kojem se nalazi datoteka *existing_mnist_rbm.py* i u komandnu liniju unesemo naredbu *python "*existing_mnist_rbm.py" "existing_mnist_rbm.cfg" (bez navodnika), gdje je *existing_mnist_rbm.cfg* putanja do konfiguracijske datoteke iz koje datoteka za pokretanje mreže učitava parametre. Format konfiguracijske datoteke je sljedeći:

```
[configuration]
weights_load_file = putanja do datoteke iz koje se učitavaju težine mreže
visible_load_file = putanja do datoteke iz koje se učitavaju pragovi okidanja
                    vidljivih neurona
hidden_load_file = putanja do datoteke iz koje se učitavaju pragovi okidanja
                   skrivenih neurona
test_dataset_location = putanja do datoteke sa skupom podataka za ispitivanje
images_save_folder = putanja do direktorija u koji će se pohranjivati originalni
                     testni podaci i podaci dobiveni rekonstrukcijom u obliku
                     slika
```

Program za pokretanje mreže parsira predanu datoteku i koristi ju za pronalaženje i učitavanje pohranjenih parametara mreže iz kojih ju i inicijalizira. Program zatim učitava skup podataka za ispitivanje, normalizira ga i pokreće ispitivanje naučene mreže – svaku sliku iz normaliziranog skupa podataka za ispitivanje postavlja na ulaz mreže, smanjuje joj dimenzionalnost, ponovno ju rekonstruira, pamti pogrešku

između originalne i rekonstruirane slike i konačno pohranjuje i originalnu i rekonstruiranu sliku na lokaciju koja je zapisana u konfiguracijskoj datoteci. Nakon što se obavi ispitivanje nad svakom slikom iz normaliziranog skupa podataka za ispitivanje, na ekran se ispisuje prosječna preciznost prilikom rekonstrukcije slika, najboljih i najgorih 10 rekonstrukcija te graf s prikazom razdiobe pogrešaka prilikom rekonstrukcije.

6. Eksperimentalni rezultati

6.1. Učenje mreže nad jednostavnim skupom podataka

Učenje mreže, dobivene rezultate i interpretaciju parametara možemo provesti nad nekim jednostavnijim skupom podataka. Kako na ulaz ograničene Boltzmannove neuronske mreže možemo postaviti ili binarni vektor ili vektor realnih vrijednosti iz intervala [0, 1], učenje mreže možemo prikazati nad skupom podataka koji prikazuju pravac $y = 2x$ i to tako da ograničimo vrijednosti upravo na skup [0, 1]. Kreiranje takvog skupa podataka prepušteno je sljedećoj funkciji:

```
def create_noisy_line_data(a, max_noise, size):
 dataset = []
 max_x = 1.0 / float(a)
 for i in range(size):
    point_x = max_x * random.random()
    point_y = min(point_x * a + max_noise *
          random.random(), 1)
    point_x = min(point_x + max_noise *
          random.random(), 1)
    data = [point_x, point_y]
    dataset.append(data)
 return dataset
```

Funkcija kao argumente prima parametar *a* (parametar jednadžbe pravca $y = ax$), maksimalan šum koji se dodaje točkama pravca i veličinu skupa podataka koji je potrebno kreirati. Kako obje koordinate točaka moraju biti iz intervala [0, 1], odabire se maksimalna *x* koordinata koju je moguće dobiti iz izraza $max\_x = 1 / a$. Taj izraz je dobiven iz osnovne jednadžbe pravca $y = ax$ i zahtjeva da maksimalna vrijednost koordinate *y* bude 1. Parametar *b* je upravo zato i izostavljen iz argumenata funkcije.

Učenje mreže provedeno je nad 10000 takvih podataka, a ispitivanje također nad 10000 podataka. Mreža je inicijalizirana sa 2 neurona u vidljivom sloju tako da joj na ulaz možemo postaviti 2D točku i sa 1 neuronom u skrivenom sloju tako da vršimo smanjivanje dimenzionalnosti. Parametri koji se koriste za učenje mreže odabrani su kako slijedi:

***batch_size* =** 100

***learning_rate* =** 0.25

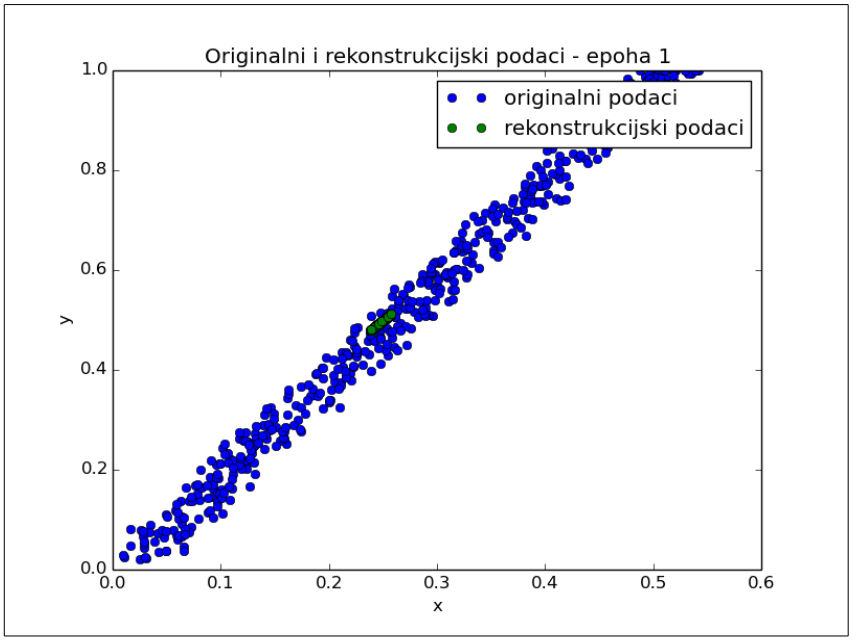
***momentum_updater* =** 0.5 prve dvije epohe, 0.9 ostale

***weight_decay_calculator* =** ranije spominjana *L2* funkcija kazne s *weight_cost* vrijednosti od 0.001

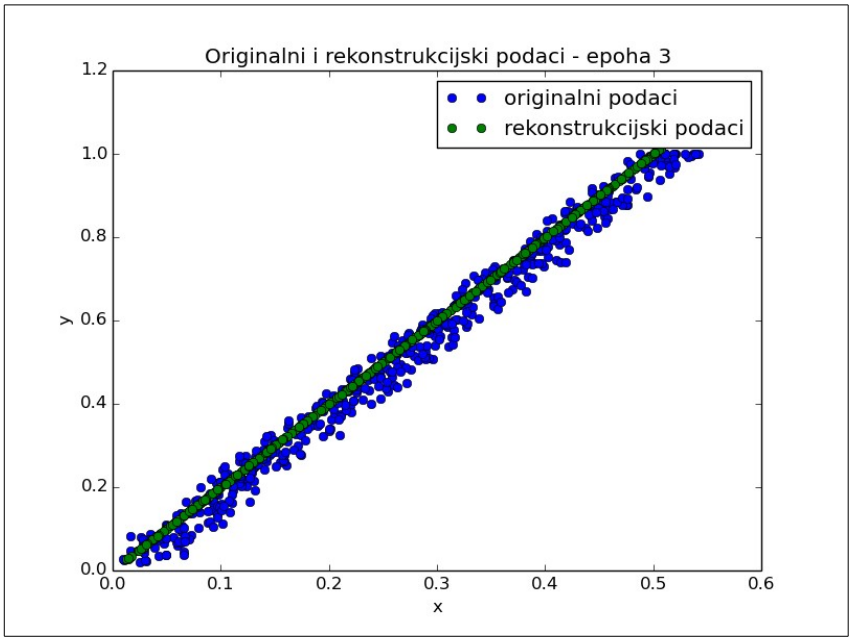
***max_epochs* =** 15

Kako smo za naš model nad skupom podataka MNIST tražili neko nelinearno preslikavanje koje bi nam dalo najbolje rezultate, situacija je ovdje drukčija. Našim modelom želimo smanjivati dimenzionalnost točaka pravca i jasno je da ćemo bolje rezultate postići koristeći neku linearnu funkciju kao prijenosnu funkciju neurona. Za ovaj problem odabrani su *ADALINE (*engl. *Adaptive Linear Element*) neuroni kod kojih je prijenosna funkcija oblika *f(net) = net,* tj. izlaz neurona se samo prosljeđuje dalje, bez obrade. Korištenjem sigmoidalne prijenosne funkcije ne može se ostvariti linearna kombinacija ulaznih podataka na izlazu pa je nemoguće ostvariti razdiobu rekonstrukcijskih podataka koja odgovara stvarnom pravcu s kojeg su točke.

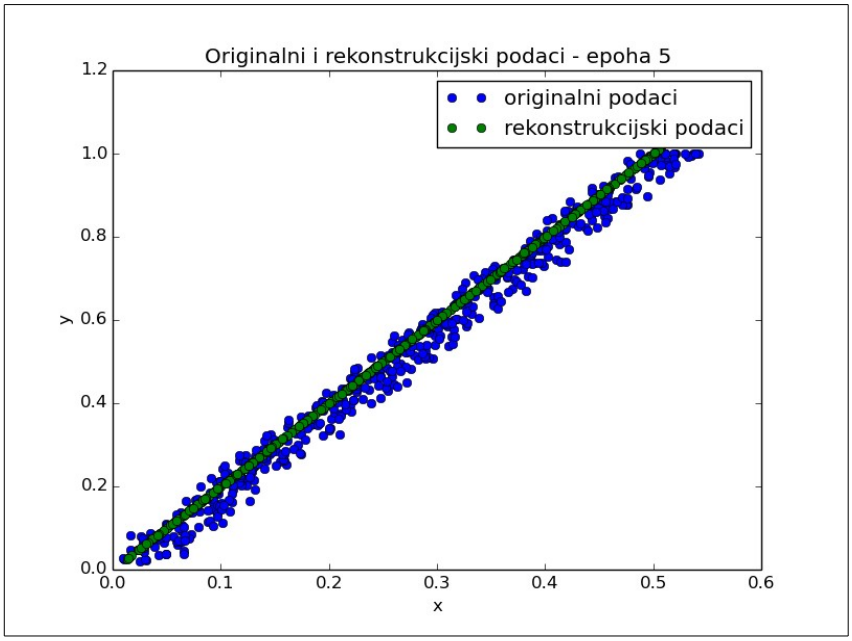
Nizom sljedećih slika bit će prikazano kako se mreža učila, tj. kakva je bila razdioba rekonstrukcije tijekom učenja.



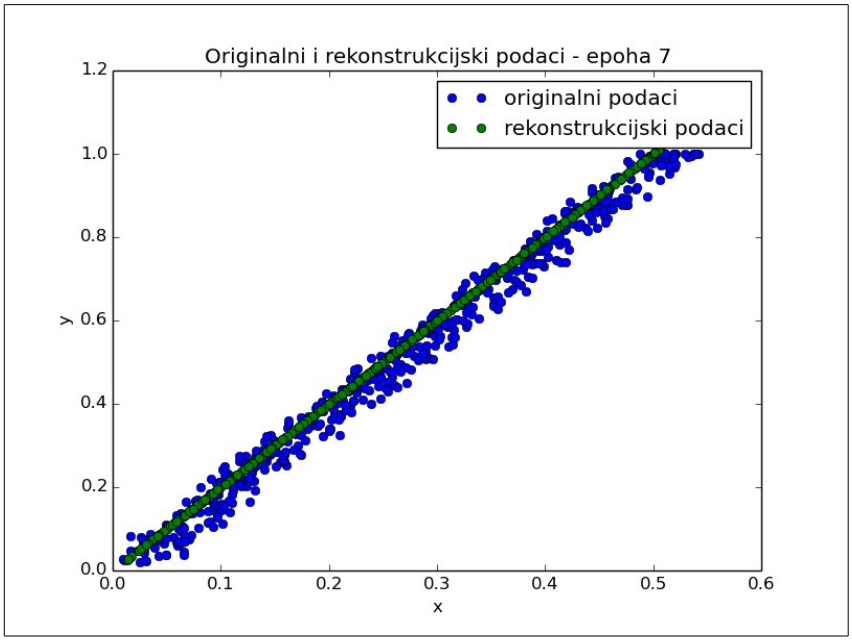
Slika 6.1.1. Razdioba originalnih i rekonstrukcijskih podataka – epoha 1



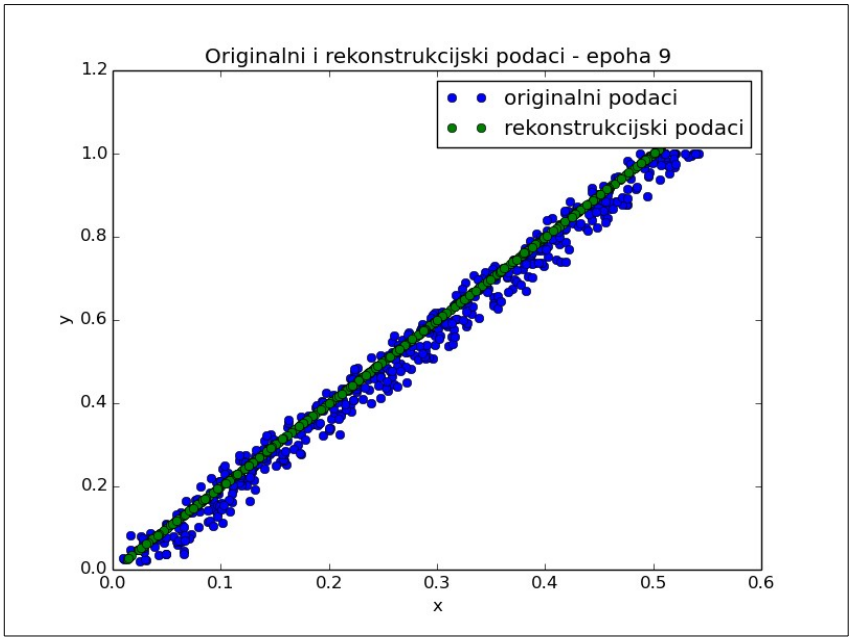
Slika 6.1.2. Razdioba originalnih i rekonstrukcijskih podataka – epoha 3



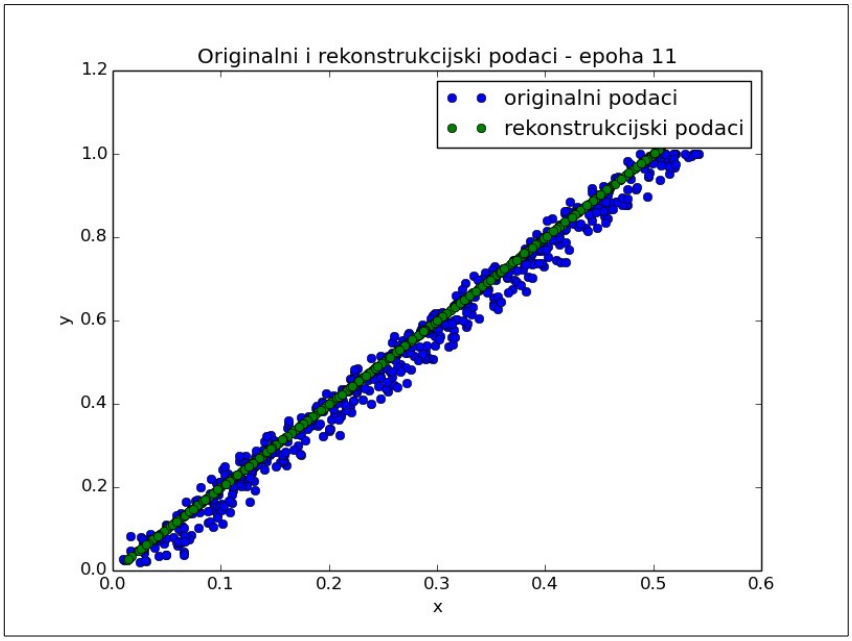
Slika 6.1.3. Razdioba originalnih i rekonstrukcijskih podataka – epoha 5



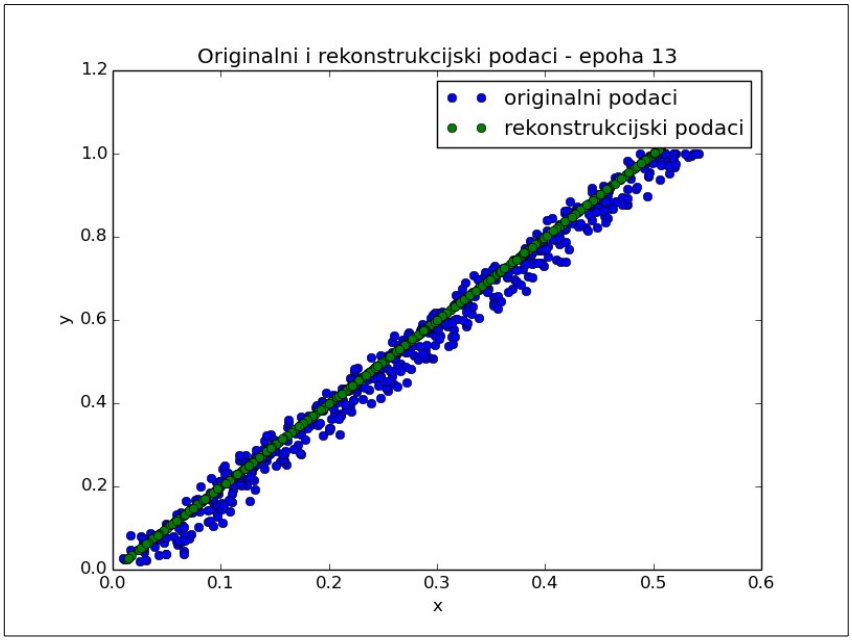
Slika 6.1.4. Razdioba originalnih i rekonstrukcijskih podataka – epoha 7



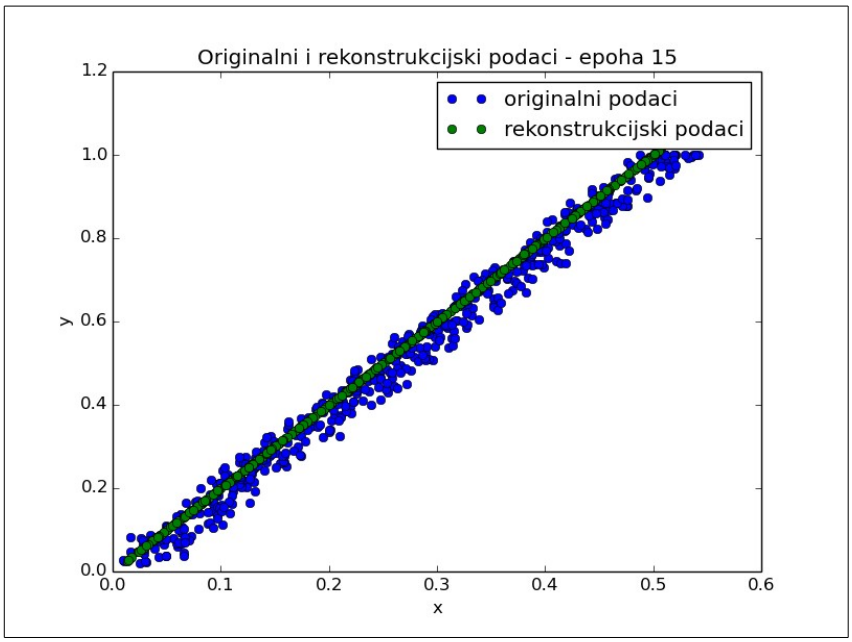
Slika 6.1.5. Razdioba originalnih i rekonstrukcijskih podataka – epoha 9



Slika 6.1.6. Razdioba originalnih i rekonstrukcijskih podataka – epoha 11



Slika 6.1.7. Razdioba originalnih i rekonstrukcijskih podataka – epoha 13



Slika 6.1.8. Razdioba originalnih i rekonstrukcijskih podataka – epoha 15

Iz slijeda slika se vidi kako u početnim epohama razdioba rekonstrukcijskih podataka ne odgovara originalnim podacima u najboljoj mjeri, a kako postupak učenja odmiče, razdioba rekonstrukcijskih podataka sve je bliža razdiobi originalnih podataka i vidljivo je kako se mreža uči.

Korištenjem mreže nad testnim podacima došlo se do srednje pogreške od **0.010040274**. Statistika rekonstrukcije točaka je sljedeća:

Najboljih 10 rekonstrukcija:

Originalna točka: (0.229585, 0.459167), rekonstrukcijska točka: (0.229693, 0.459141), pogreška: 0.000134637595685

Originalna točka: (0.220052, 0.440049), rekonstrukcijska točka: (0.220188, 0.440129), pogreška: 0.000215948768107

Originalna točka: (0.242434, 0.484263), rekonstrukcijska točka: (0.242238, 0.484231), pogreška: 0.000228948627414

Originalna točka: (0.240048, 0.479380), rekonstrukcijska točka: (0.239819, 0.479393), pogreška: 0.000242563658054

Originalna točka: (0.211081, 0.421693), rekonstrukcijska točka: (0.211097, 0.421948), pogreška: 0.000271306845637

Originalna točka: (0.236234, 0.471733), rekonstrukcijska točka: (0.236017, 0.471789), pogreška: 0.000272939233421

Originalna točka: (0.240512, 0.480231), rekonstrukcijska točka: (0.240251, 0.480256), pogreška: 0.000286491814646

Originalna točka: (0.254015, 0.507829), rekonstrukcijska točka: (0.253921, 0.507597), pogreška: 0.000325991277145

Originalna točka: (0.229823, 0.458868), rekonstrukcijska točka: (0.229622, 0.458998), pogreška: 0.00033110485544

Originalna točka: (0.255152, 0.510062), rekonstrukcijska točka: (0.255036, 0.509827), pogreška: 0.000350916777787

Najgorih 10 rekonstrukcija:

Originalna točka: (0.231698, 0.364264), rekonstrukcijska točka: (0.192344, 0.384441), pogreška: 0.0595315713731

Originalna točka: (0.076322, 0.055732), rekonstrukcijska točka: (0.038637, 0.077020), pogreška: 0.058973076632

Originalna točka: (0.083053, 0.069624), rekonstrukcijska točka: (0.045505, 0.090756), pogreška: 0.0586806685606

Originalna točka: (0.355608, 0.612521), rekonstrukcijska točka: (0.315802, 0.631362), pogreška: 0.0586479006121

Originalna točka: (0.302531, 0.506994), rekonstrukcijska točka: (0.263243, 0.526241), pogreška: 0.0585357255374

Originalna točka: (0.172944, 0.248905), rekonstrukcijska točka: (0.134742, 0.269234), pogreška: 0.058531645453

Originalna točka: (0.054800, 0.013803), rekonstrukcijska točka: (0.017667, 0.035079), pogreška: 0.0584094897368

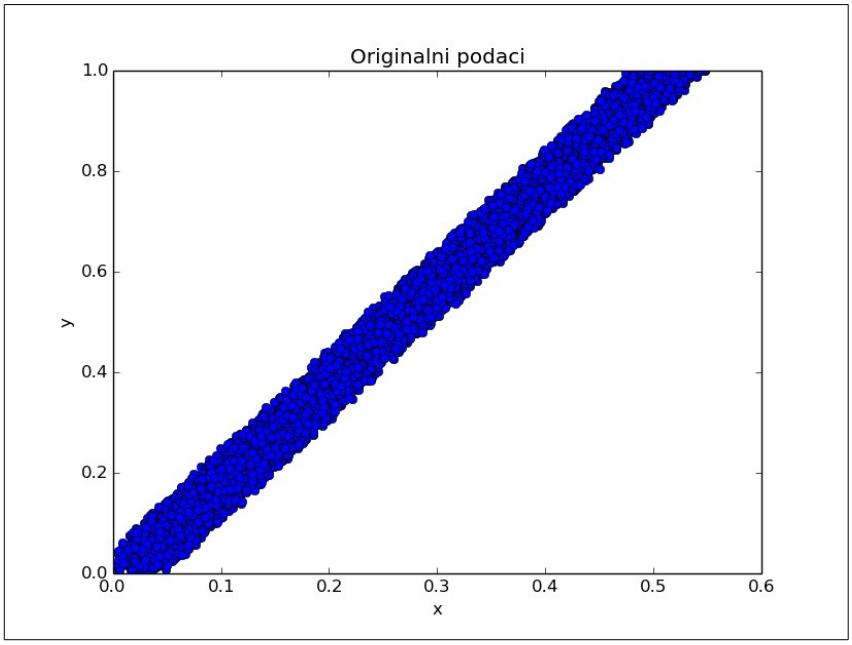
Originalna točka: (0.062748, 0.029730), rekonstrukcijska točka: (0.025588, 0.050921), pogreška: 0.0583508783234

Originalna točka: (0.141918, 0.187533), rekonstrukcijska točka: (0.104144, 0.208036), pogreška: 0.0582781200875

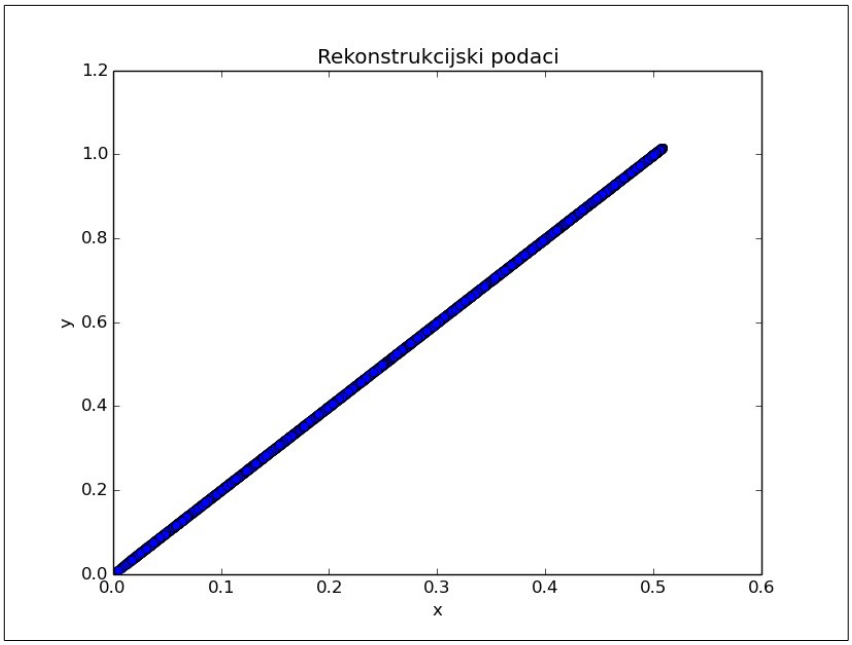
Originalna točka: (0.165537, 0.234798), rekonstrukcijska točka: (0.127654, 0.255058), pogreška: 0.0581432063273

Pogreška predstavlja zbroj apsolutne razlike između originalne točke i rekonstrukcijske točke po obje komponente.

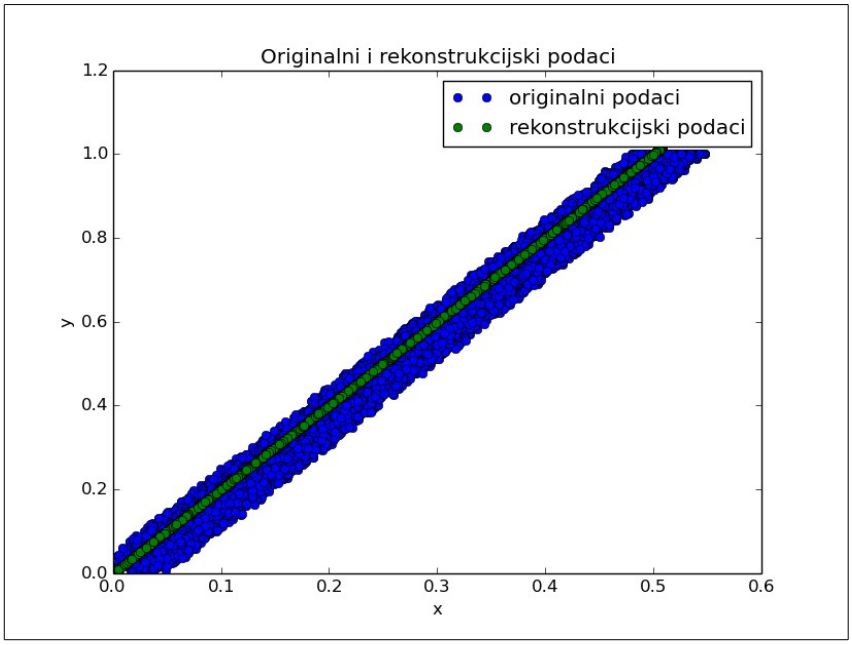
Niz sljedećih slika prikazuje razdiobu originalnih podataka, razdiobu rekonstruiranih podataka i razdiobu rekonstrukcijske pogreške.



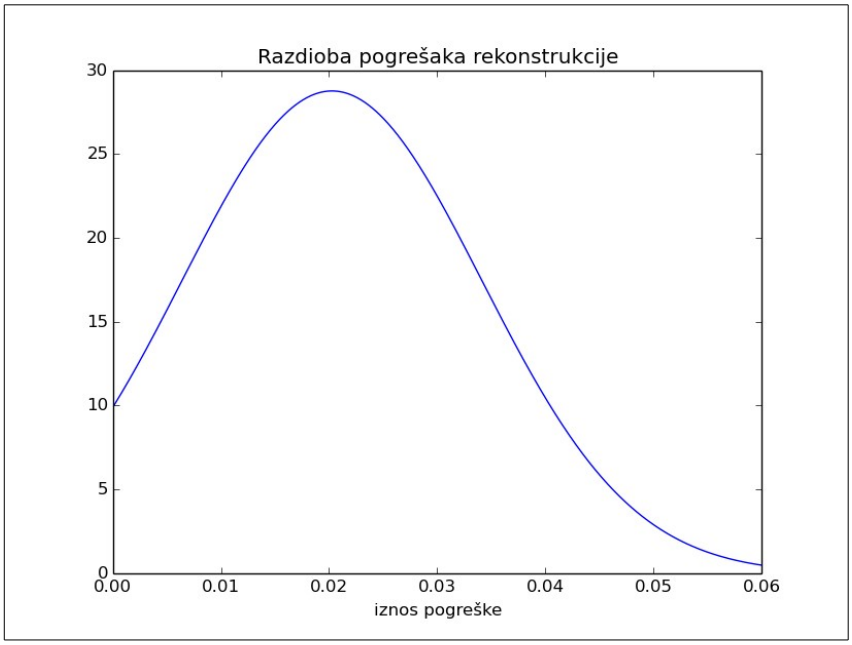
Slika 6.1.9. Razdioba originalnih podataka



Slika 6.1.10. Razdioba rekonstruiranih podataka



Slika 6.1.11. Zajednička razdioba originalnih i rekonstruiranih podataka



Slika 6.1.12. Razdioba pogrešaka rekonstrukcije

Kako vidimo iz slika, razdioba rekonstrukcijskih podataka dobro prati originalni pravac po kojem su se generirale točke. Iz najboljih i najgorih 10 rekonstrukcija i slike koja prikazuje razdiobu rekonstrukcijskih pogrešaka vidimo da je većina rekonstrukcija s malom pogreškom pa možemo reći da naučeni model dobro smanjuje dimenzionalnost i dobro prikazuje točke koje pripadaju 2D pravcu. Dobiveni parametri mreže su sljedeći:

$$w = \begin{bmatrix} -0.44610137 \\ -0.89220839 \end{bmatrix}$$

$$b = \begin{bmatrix} 0.09421949 & 0.18818159 \end{bmatrix}$$

$$c = \begin{bmatrix} -0.20835128 \end{bmatrix}$$

gdje je *w* matrica težina između vidljivih i skrivenih neurona, *b* je matrica pragova okidanja vidljivih neurona, a *c* je matrica pragova okidanja skrivenih neurona. Matricu težina možemo i lako interpretirati – kako vidimo da je omjer $w_{21} : w_{11}$ = -0.89220839 : -0.44610137 = 2.000012665, a to je upravo i koeficijent smjera pravca čije točke smo prevodili u prostor smanjene dimenzionalnosti. Kada postavimo podatke na skriveni neuron tako naučene mreže, vidimo da će vrijednost na prvom vidljivom neuronu, odnosno koordinata *x,* biti dvostruko manja od vrijednosti na drugom vidljivom neuronu, odnosno koordinate *y,* naravno, uz sitne korekcije koje vrše pragovi okidanja neurona, što upravo odgovara podacima koje želimo – pravcu $y = 2x$.

Također, vidljivo je kako rekonstrukcijski podaci bolje prate originalni pravac. Iz toga slijedi da je mreža bila sposobna pronaći razdiobu originalnih podataka unatoč tome što su bili zašumljeni, a to nam izvrsno odgovara jer i u praksi želimo da mreža bude sposobna iz podataka koji ne odgovaraju originalnim podacima u potpunosti, odnosno zašumljeni su, izvući ono što ti podaci predstavljaju. Upravo iz tog razloga su i pogreške prilikom rekonstrukcije bile visoke – mreža je naučila da točke pripadaju pravcu, a nije tražila direktno preslikavanje u zašumljene točke.

6.2. Metaparametri mreže i algoritma za učenje nad MNIST skupom podataka

Kako u algoritmu za učenje ograničene Boltzmannove neuronske mreže postoje brojni parametri koje trebamo postaviti, možemo reći da svaki skup parametara zajedno s generičkim algoritmom učenja koji koristi taj skup čini jedan poseban algoritam za učenje mreže. Cilj je pronaći takav skup parametara koji daje optimalne rezultate pri smanjivanju dimenzionalnosti podataka, tj. onaj koji će imati najmanju rekonstrukcijsku pogrešku nad ispitnim skupom podataka. Kako je ranije opisano, za učenje mreže se parametri učitavaju iz konfiguracijske datoteke i koriste se u postupku učenja. Parametri koje moramo postaviti su sljedeći: očekivanje distribucije početnih vrijednosti težina mreže, standardna devijacija distribucije početnih vrijednosti težina mreže, broj neurona u skrivenom sloju mreže, tj. broj komponenata podataka u smanjenoj dimenzionalnosti, veličina *batcha*, tj. broja uzoraka koji se prikazuju mreži prije nego se osvježavaju parametri, stopa učenja, maksimalni broj epoha, početni momentum, završni momentum, indeks epohe na kojoj se momentum postavlja s početnog na konačni te parametar težinske cijene koji se koristi u ranije opisanoj *L2* funkciji za postupak smanjenja težina, tj. za kažnjavanje težina mreže koje su prevelikog apsolutnog iznosa. Od navedenih parametara, samo neki će biti uključeni u pretragu za optimalnim skupom dok će ostatak biti fiksno postavljen prema preporukama [7]. Parametri koji će biti fiksno postavljeni su iznos početnog i završnog momentuma, očekivanje i standardna devijacija distribucije početnih vrijednosti težina i broj epoha. Parametar koji će također biti postavljen fiksno je broj neurona u skrivenom sloju jer je očigledno da će veći broj neurona u skrivenom sloju svakako uzrokovati veću ekspresivnost mreže koju učimo i dati nam manju rekonstrukcijsku pogrešku, ali uz manju kompresiju pa je za taj parametar nad MNIST skupom podataka odabrana vrijednost 100 što nam daje kompresiju faktora 7.84 (784 komponenata originalne slike, 100 komponenata podataka smanjene dimenzionalnost).

Pronalazak optimalnih parametara iz skupa parametara koji nisu postavljeni fiksno obavljeno je pomoću datoteke *metaparameters_calibration_mnist.py*. Uz tu datoteku, u datoteci *metaparameters_calibration_mnist.cfg* potrebno je postaviti

kako će se optimalni parametri tražiti. Potrebno je postaviti početne vrijednosti parametara, završne vrijednosti parametara i korak promjene vrijednosti parametara. Tako je za stopu učenja za početnu vrijednost odabrana vrijednost 0.08, za završnu vrijednost odabrana je vrijednost 0.15, za korak promjene odabrana je vrijednost 0.005. Za početnu vrijednost indeksa epohe na kojem se momentum mijenja odabrana je vrijednost 1, za završnu vrijednost odabrana je vrijednost 6, za korak promjene odabrana je vrijednost 1. Kako se učenje obavlja kroz 50 epoha, završna vrijednost odgovara postotku od 12% ukupnog broja epoha. Konačno, za početnu vrijednost parametra težinske cijene odabrana je vrijednost 0.00001, za završnu vrijednost odabrana je vrijednost 0.01, za korak promjene odabrana je vrijednost 0.0001. Za svaku kombinaciju skupa parametara mreža se učila kroz 50 epoha nad smanjenim skupom podataka za učenje i zatim se ispitivala nad također smanjenim skupom podataka za ispitivanje (radi ubrzanja pronalaska optimalnih parametara) i pratila se preciznost, tj. pogreška prilikom rekonstrukcije nad skupom testnih podataka. Uz svaki od parametara zapisala se i navedena pogreška, taj podatak pohranjen je u memoriju i na kraju postupka je u silaznom redoslijedu ispisan od rekonstrukcije s najmanjom pogreškom do one s najvećom i utvrđen je sljedeći skup parametara: stopa učenja vrijednosti 0.125, promjena momentuma s početne na završnu vrijednost na indeksu 4, tj. na indeksu koji čini 8% ukupnog broja epoha i *weight – cost* parametar vrijednosti 0.00081. Preciznost koja se dobiva za takav skup parametara nad skupovima za učenje i ispitivanje smanjene veličine je **0.91397**. Takav skup parametara će se koristiti i za učenje i ispitivanje nad cjelokupnim skupom podataka.

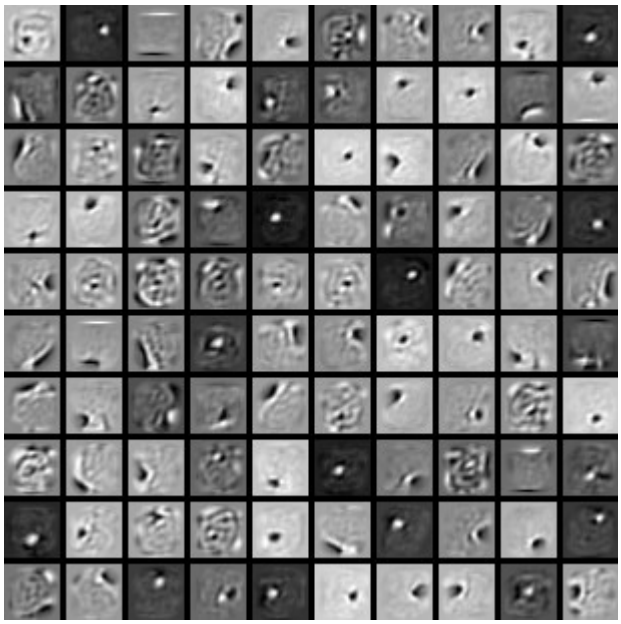
6.3. Rekonstrukcija slika MNIST skupa podataka iz podataka smanjene dimenzionalnosti

Učenje mreže provedeno je nad 60000 podataka iz MNIST skupa podataka za učenje. Ispitivanje mreže provedeno je nad 10000 podataka iz MNIST skupa podataka za ispitivanje, različitih od onih koji su se koristili za učenje mreže. Mreža je inicijalizirana sa 784 neurona u vidljivom sloju tako da joj na ulaz možemo postaviti sliku dimenzija 28x28, i sa 100 neurona u skrivenom sloju tako da vršimo smanjivanje dimenzionalnosti s faktorom 7.84. Prijenosna funkcija koja se koristi između slojeva je, kako je ranije dogovoreno, sigmoidalna funkcija. Parametri koji se koriste za učenje mreže odabrani su kako slijedi:

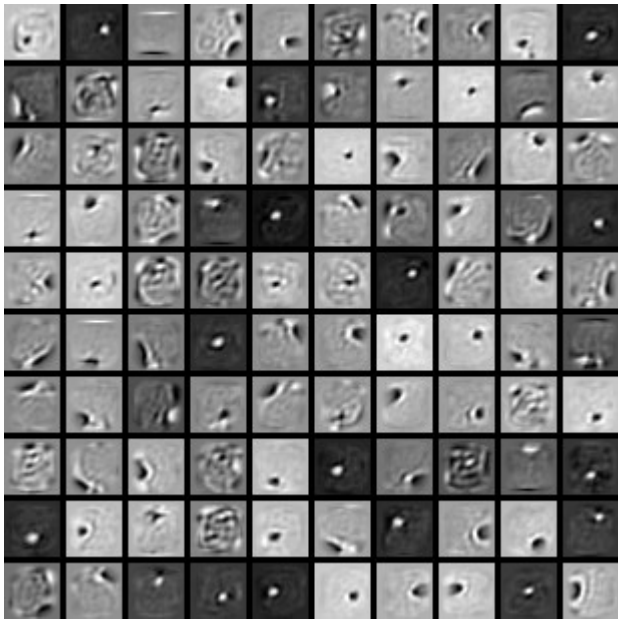
[configuration]
weights_random_mean = 0
weights_random_stddev = 0.01
lowered_dimensionality_components = 100
mini_batch_size = 100
learning_rate = 0.125
max_epochs = 300
starting_momentum = 0.5
ending_momentum = 0.9
momentum_change_step = 24
weight_cost = 0.00081

Nizom sljedećih slika bit će prikazano kako se mreža učila, tj. prikazat će se težine između svakog od skrivenog neurona sa svakim od vidljivih neurona i njihova interpretacija nizom slika dimenzija 28x28 koje prikazuju koja područja originalnih slika aktiviraju promatrane skrivene neurone. Kako je korištena mreža sa 784 vidljiva i 100 skrivenih neurona, težine će biti prikazane slikama koje sadrže 100 manjih slika od kojih je svaka dimenzija 28x28, tj. predstavlja svaku od 784 težina svakog skrivenog neurona.

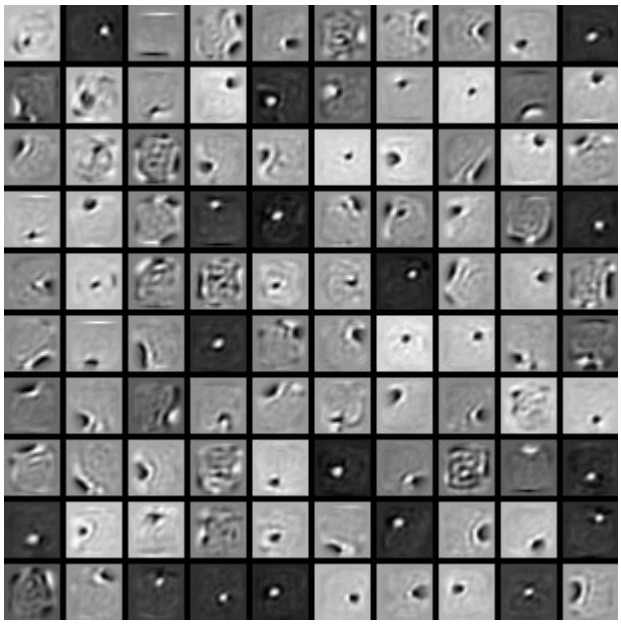
Slika 6.3.1. Prikaz naučenih težina nakon 1. epohe



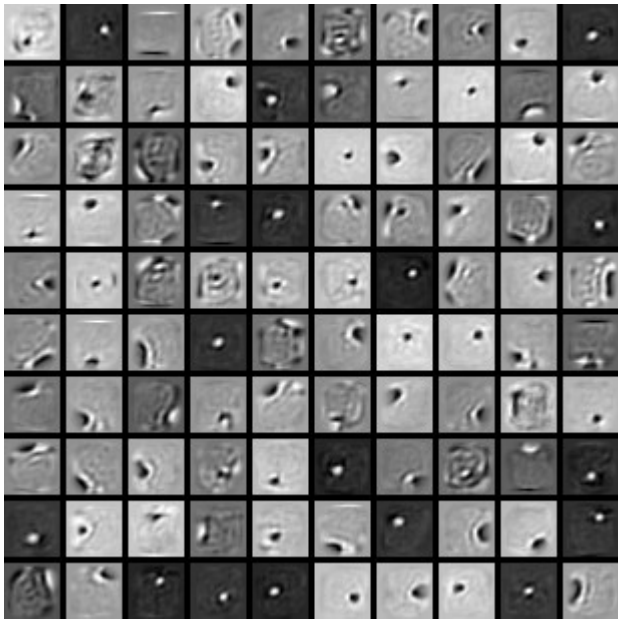
Slika 6.3.2. Prikaz naučenih težina nakon 50. epohe



Slika 6.3.3. Prikaz naučenih težina nakon 100. epohe



Slika 6.3.4. Prikaz naučenih težina nakon 200. epohe



Slika 6.3.5. Prikaz naučenih težina nakon 300. epohe

Kako vidimo iz slijeda slika, kako iteracije algoritma za učenje mreže odmiču, model i parametri koji predstavljaju model mijenjaju se. Nakon posljednje epohe vidljivo je kako je mreža naučena tako da se svaki od neurona aktivira za pojedine elemente originalnih slika. Dijelovi koji su najsvjetliji na svakom pojedinom elementu na slici odgovaraju onim pikselima za čije je prepoznavanje svaki od skrivenih neurona zadužen. Primjerice, promotrimo sliku 6.3.5. - dio slike koji odgovara 5. slici u 3. redu možemo tumačiti kao filter koji će prepoznavati početni dio brojke 2 (početak znamenke, bez završnog vodoravnog dijela).

Glavni dio uporabe naučene mreže upravo je predstavljanje slika iz skupa podataka u nižoj dimenzionalnosti – slike iz originalnih 784 piksela predstavljamo kao 100 komponentni vektor vrijednosti između 0 i 1. Korištenjem mreže nad testnim podacima ostvarena je preciznost **0.966957185408** , tj. **96.695718541%**, odnosno prosječna pogreška mreže je **0.033042815**, tj. **3.304281459%**. Pogreška je izračunata apsolutnim zbrojem odstupanja rekonstrukcije od originalne slike za svaki piksel. U prosjeku, to znači da je po slici prosječno odstupanje **25.90556696** piksela (svaki pojedini piksel prikazan je kao realna vrijednost između 0 i 1).

Statistika pogrešaka s najboljim i najgorim rekonstrukcijama je sljedeća:

Najboljih 10 rekonstrukcija:
Slika 5211, pogreška 6.75458193071
Slika 1876, pogreška 6.85537699987
Slika 5524, pogreška 6.89522022335
Slika 5811, pogreška 6.94350357191
Slika 178, pogreška 6.9696679411
Slika 6819, pogreška 7.04113895651
Slika 3648, pogreška 7.04621699875
Slika 6397, pogreška 7.09665934121
Slika 7280, pogreška 7.1205983585
Slika 2357, pogreška 7.13021744971

Najgorih 10 rekonstrukcija:
Slika 3364, pogreška 70.0676406502
Slika 3768, pogreška 69.3665426674
Slika 8602, pogreška 65.2781359012
Slika 2462, pogreška 65.0271863683
Slika 8553, pogreška 63.8117090083
Slika 6161, pogreška 63.5210972864
Slika 1610, pogreška 63.085102003
Slika 1352, pogreška 62.779120712
Slika 4140, pogreška 62.66560896
Slika 4086, pogreška 62.1573269574

Sve navedene pogreške predstavljaju, kao i za ukupnu prosječnu pogrešku naučene mreže, apsolutni zbroj odstupanja rekonstrukcije od originalne slike za svaki piksel. Slike su indeksirane redom kako su čitane iz skupa podataka, počevši s indeksom 0.

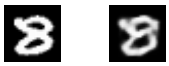
Nizom sljedećih slika prikazane su originalne i rekonstrukcijske slike najboljih i najgorih rekonstrukcija naučene mreže. Kako svaka od prvih 10 najboljih rekonstrukcija predstavlja rekonstrukciju znamenke 1 iz prostora smanjene dimenzionalnosti, prikazane su najbolja i najgora rekonstrukcija od 10 najboljih – dakle, slike 5211 i 2357.

Najbolje rekonstrukcije:

Slika 6.3.6. Slika 5211 – original lijevo; rekonstrukcija desno

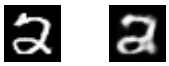
Slika 6.3.7. Slika 2357 – original lijevo; rekonstrukcija desno

Najgore rekonstrukcije:



Slika 6.3.8. Slika 3364 – original lijevo; rekonstrukcija desno

Slika 6.3.9. Slika 3768 – original lijevo; rekonstrukcija desno



Slika 6.3.10. Slika 8602 – original lijevo; rekonstrukcija desno

Slika 6.3.11. Slika 2462 – original lijevo; rekonstrukcija desno

Slika 6.3.12. Slika 8553 – original lijevo; rekonstrukcija desno

Slika 6.3.13. Slika 6161 – original lijevo; rekonstrukcija desno

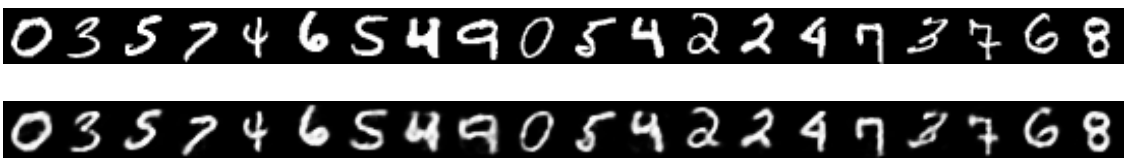
Slika 6.3.14. Slika 1610 – original lijevo; rekonstrukcija desno

Slika 6.3.15. Slika 1352 – original lijevo; rekonstrukcija desno

Slika 6.3.16. Slika 4140 – original lijevo; rekonstrukcija desno

Slika 6.3.17. Slika 4086 – original lijevo; rekonstrukcija desno

Kako je vidljivo iz rezultata, najgore rekonstrukcije koje je mreža obavila i dalje su prihvatljive – iz gotovo svih rekonstrukcija moguće je prepoznati stvarnu znamenku koju su podaci u nižoj dimenzionalnosti predstavljali. Na sljedećoj slici prikazan je niz originalnih znamenki u jednom redu i rekonstrukcija u drugom za slučajno odabrane znamenke iz skupa podataka za ispitivanje.

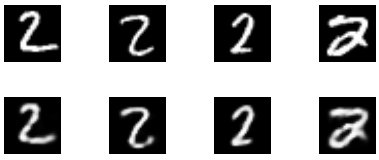


Slika 6.3.18. Slijed slučajno odabranih slika – original gore; rekonstrukcija dolje

Korištenjem ograničene Boltzmannove neuronske mreže za predstavljanje slika u nižoj dimenzionalnosti dobivamo nelinearno preslikavanje između originalnih podataka i podataka smanjene dimenzionalnosti koje je izrazito otporno na šumove u podacima, sposobno za prepoznavanje i ispravno smanjivanje dimenzionalnosti podataka te rekonstrukciju iz niskodimenzionalnih podataka. Otpornost na šumove i varijacije prikazat ćemo kroz niz različitih varijacija za svaku klasu znamenaka i ispravno smanjivanje dimenzionalnosti i rekonstrukciju za svaku od varijacija.

Slika 6.3.19. Varijacije znamenke 0 – original gore; rekonstrukcija dolje

Slika 6.3.20. Varijacije znamenke 1 – original gore; rekonstrukcija dolje



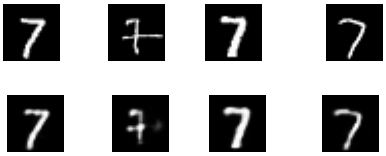
Slika 6.3.21. Varijacije znamenke 2 – original gore; rekonstrukcija dolje

Slika 6.3.22. Varijacije znamenke 3 – original gore; rekonstrukcija dolje

Slika 6.3.23. Varijacije znamenke 4 – original gore; rekonstrukcija dolje

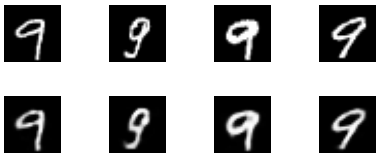
Slika 6.3.24. Varijacije znamenke 5 – original gore; rekonstrukcija dolje

Slika 6.3.25. Varijacije znamenke 6 – original gore; rekonstrukcija dolje



Slika 6.3.26. Varijacije znamenke 7 – original gore; rekonstrukcija dolje

Slika 6.3.27. Varijacije znamenke 8 – original gore; rekonstrukcija dolje



Slika 6.3.28. Varijacije znamenke 9 – original gore; rekonstrukcija dolje

Iz rekonstrukcija za svaku pojedinu klasu znamenaka je vidljivo kako je naučen model izrazito otporan na varijacije i šumove u podacima i sposoban je izvući prave podatke iz onoga što mu je predočeno na ulazu.

U početku smo htjeli naučiti model koji će nelinearnim preslikavanjem pokušati smanjiti dimenzionalnost stvarnih podataka i nadali se rezultatima boljima od PCA linearnog preslikavanja. Ponašanje PCA metode nad MNIST skupom podataka ispitano je korištenjem *mlpy* modula za strojno učenje u programskom jeziku Python [11].

Korištenjem naučenog modela nad testnim podacima ostvarena je preciznost **0.959073230396**, tj. **95.9073230396%**, odnosno prosječna pogreška mreže je **0.04092677**, tj. **4.092677%**. Pogreška je izračunata apsolutnim zbrojem

odstupanja rekonstrukcije od originalne slike za svaki piksel. U prosjeku, to znači da je po slici prosječno odstupanje **32.08658737** piksela (svaki pojedini piksel prikazan je kao realna vrijednost između 0 i 1).

Statistika pogrešaka s najboljim i najgorim rekonstrukcijama je sljedeća:

Najboljih 10 rekonstrukcija:
Slika 5524, pogreška 13.0491729603
Slika 3919, pogreška 13.8469223648
Slika 6276, pogreška 14.2246295079
Slika 1876, pogreška 14.3949328687
Slika 3314, pogreška 14.4604187702
Slika 4708, pogreška 14.4853495689
Slika 2541, pogreška 14.5596671243
Slika 735, pogreška 14.897869997
Slika 3648, pogreška 14.9144671552
Slika 4774, pogreška 14.9411869782

Najgorih 10 rekonstrukcija:
Slika 1017, pogreška 61.8903398852
Slika 4140, pogreška 59.8383790366
Slika 1170, pogreška 59.3762089641
Slika 1610, pogreška 57.3322510558
Slika 1698, pogreška 56.7988005903
Slika 8116, pogreška 56.6421800435
Slika 1569, pogreška 56.4633674003
Slika 402, pogreška 56.1772460101
Slika 54, pogreška 56.1221472723
Slika 5165, pogreška 56.1075772304

Preciznost je na isti način mjerena za oba modela i vidljivo je kako ograničena Boltzmannova neuronska mreža ima nešto bolji rezultat od PCA metode. U nastavku će kroz slijed slika biti prikazane slučajno izabrane originalne i rekonstrukcijske slike za obje navedene metode.

Slika 6.3.29. Slijed slučajno odabranih slika i rekonstrukcije

Na slici 6.3.29. vidljiv je slijed slučajno odabranih slika i rekonstrukcija svakom od metoda. Prvi redak predstavlja stvarne slike, srednji redak predstavlja rekonstrukcije iz 100 – dimenzionalnog vektora podataka pomoću ograničene Boltzmannove neuronske mreže, a posljednji redak predstavlja rekonstrukcije iz 100 – dimenzionalnog vektora podataka pomoću PCA metode. Vidljivo je kako je smanjivanje dimenzionalnosti stvarnih slika uz rekonstrukciju nešto bolje korištenjem ograničene Boltzmannove neuronske mreže.

7. Zaključak

U okviru ovog rada obavljeno je učenje modela koji se koristi za smanjivanje dimenzionalnosti slika. Model koji je korišten je ograničena Boltzmannova neuronska mreža. Eksperimentalni rezultati pokazali su da uspješnost učenja modela u jednoj mjeri ovisi i o skupu parametara koje algoritam za učenje koristi. U drugom dijelu rada naučen model koristio se za smanjivanje dimenzionalnosti slika i rekonstrukciju iz smanjene dimenzionalnosti u stvarnu. Pokazano je da je rekonstrukcija provedena uspješno i da pogreške nisu velike. Naučen model dobro predstavlja stvarne podatke u smanjenoj dimenzionalnosti s nešto boljom preciznošću od metode PCA.

Kako je u ovom radu korišten samo jedan sloj ograničene Boltzmannove neuronske mreže, u budućem radu bi se kvaliteta rekonstrukcije i faktor smanjenja dimenzionalnosti mogli poboljšati korištenjem višeslojne arhitekture – spajanjem više ograničenih Boltzmannovih neuronskih mreža tako da je izlaz jedne ulaz u drugu mrežu. Takva višeslojna arhitektura bi dala bolje rezultate zbog veće ekspresivnosti izražene kroz veći broj parametara (težina).

8. Literatura

[1] – Hinton G.E., Salakhutdinov R.R. *Reducing the Dimensionality of Data with Neural Networks*. *Science.* 28.06.2006. broj 313

[2] – Čupić M., Dalbelo – Bašić B., Golub M. *Neizrazito, evolucijsko i neuroračunarstvo: Umjetne neuronske mreže.* Fakultet elektrotehnike i računarstva, Zagreb, 2013.

[3] – Elezović N., *Diskretna vjerojatnost.* Element, Zagreb, 2007.

[4] – Elezović N., *Slučajne varijable.* Element, Zagreb, 2007.

[5] – Elezović N., *Matematička statistika i stohastički procesi.* Element, Zagreb, 2007.

[6] – Igel C., Fischer A., *An introduction to Restricted Boltzmann Machines.* Institut für Neuroinformatik, Ruhr – Universität Bochum, Germany. Department of Computer Science, University of Copenhagen, Dennmark.

[7] – Hinton G. *A Practical Guide to Training Restricted Boltzmann Machines.* Department of Computer Science, University of Toronto, 2010.

[8] – LeCun Y., Cortes C., Burges C. J.C. *The MNIST Database of handwritten digits*. URL: http://yann.lecun.com/exdb/mnist/

[9] - Implementacija Contrastive Divergence algoritma http://deeplearning.net/tutorial/rbm.html#contrastive-divergence-cd-k

[10] – Implementacija Contrastive Divergence algoritma http://www.cs.toronto.edu/~hinton/code/Autoencoder_Code.tar

[11] – Albanese D., Visintainer R., Merler S., Riccadonna S., Jurman G., Furlanello C. *mlpy: Machine Learning Python*, 2012. arXiv:1202.6548 [bib]

Predstavljanje slika ograničenim Boltzmannovim neuronskim mrežama

Sažetak

Rad opisuje postupak učenja modela koji omogućuje predstavljanje slika u nižoj dimenzionalnosti – kompresiju slika. Kao skup slika koji se predstavlja u nižoj dimenzionalnosti koristi se MNIST skup podataka rukom pisanih brojeva. MNIST skup podataka službeno se sastoji od dva disjunktna skupa – skupa za učenje koji koristimo za učenje same mreže i skupa za ispitivanje koji koristimo za provjeru preciznosti naučenog modela. Skup podataka se prvo normalizira tako da ima vrijednosti između 0 i 1, uključivo. Model koji učimo je ograničena Boltzmannova neuronska mreža. Model učimo algoritmom kontrastne divergencije. Učenje modela takvim algoritmom prikazano je i nad vrlo jednostavnim skupom podataka – zašumljenim podacima s pravca. Eksperimentalni rezultati preciznosti modela nad MNIST skupom podataka smanjene veličine prikazani su koristeći različite parametre algoritma za učenje. Najbolji dobiveni parametri korišteni su za učenje nad cjelokupnim skupom podataka. Dobivena naučena mreža je potom ispitana na testnom skupu podataka uz prikaz najboljih i najgorih rekonstrukcija, prosječnu pogrešku te usporedbu s metodom PCA.

Ključne riječi: ograničena Boltzmannova neuronska mreža, kontrastna divergencija, MNIST, normalizacija, smanjivanje dimenzionalnosti, kompresija, Python, NumPy, SciPy

Representing images with restricted Boltzmann networks

Abstract

This paper describes training of model that enables mapping of original images into lowered dimensionality ones - compressing. As dataset that is represented in lowered dimensionality is used MNIST handwritten digits dataset. MNIST dataset is officially divided into two disjunctive sets – training set which is used for training the model and testing set which is used to check the accuracy of trained model. First thing to do is normalizing the dataset so it contains values from 0 to 1, inclusive. The model that is trained is restricted Boltzmann network. Algorithm used for training is Contrastive Divergence algorithm. Model training with mentioned algorithm is also shown on very simple dataset – noisy line data. Experimental results of model accuracy on reduced size MNIST dataset were shown using different sets of algorithm parameters. The best parameters were used to train the model on complete MNIST dataset. Trained model was then tested on test dataset showing best and worst reconstruction results, average reconstruction error and comparation to PCA method.

Keywords: restricted Boltzmann network, Contrastive Divergence, MNIST, normalization, dimensionality reduction, compression, Python, NumPy, SciPy