**SVEUČILIŠTE U ZAGREBU**

**PRIRODOSLOVNO-MATEMATIČKI FAKULTET**

**MATEMATIČKI ODSJEK**

Daniel Jelušić

Andrea Perčinlić

Jure Šiljeg

Validacija 4 algoritma o rukom pisanim brojevima

Korisnička dokumentacija

Zagreb, veljača 2014.

**O problemu:**

Ideja za projektnu temu je dobivena nakon 2. zadaće iz kolegija Umjetna inteligencija gdje smo morali samostalno napraviti validaciju k-means i knn algoritama za prepoznavanje rukom pisanih brojeva. Ovaj projekt se temelji na analizi tri vrste klasifikatora pomoću kojih ćemo pokušati prepoznavati (te ocijeniti i mjeriti uspješnost svakog prepoznavanja) rukom pisane znamenke za sva tri klasifikatora i pokušati dobiti puno bolje rezultate od onih koji su bili predviđeni u regularnoj zadaći, a radit će se s naivnim Bayesovskim klasifikatorom, SVD dekompozicijom velikog skupa podataka iz MNIST-ove baze podataka i konvolucijskom neuronskom mrežom.

**O rješenjima :**

Konvolucijska neuronska mreža:

Za početak, što je to konvolucijska neuronska mreža? To je neuronska mreža čija se arhitektura sastoji od konvolucijskog sloja, softmax regresijskog sloja i izlaznog sloja. Glavni koraci algoritma su:

0) inicijalizacija parametara i pohranjivanje podataka iz MNIST-ove baze podataka

1) implementacija računanja funkcije troška i gradijenta

2) učenje parametara (treniranje modela)

3) testiranje performansi modela i ispisivanje rezultata

U 0. koraku smo inicijalizirali parametre kao što su imageDim,

numClasses, filterDim,numFilters, poolDim, povukli podatke s MNIST-ove baze i napraviti kategorizaciju znamenki. Cijeli prvi korak se svodi na implementiranje funkcije cnnCost gdje dobivamo više rezultata. Dobijemo ukupno odstupanje od pravog rezultata, sve gradijente koji su nam potrebni za gradijentni spust te predviđanja modela (klasifikaciju). Sljedeći korak nam daje uvid u način učenja parametara. Tu je osnovna stvar SGD petlja koja nam ubrzava samo izvršavanje koristeći gradijentni spust (napredna vrsta optimizacije koja bitno utječe na brzinu izvršavanja) umjesto postupnog računanja cost funkcije i pojedinih gradijenata. Također se koristi kaljenje inicijalnog koeficjenta učenja da se dobije bolja konvergencija k rješenju kao što se koristi i miješanje podataka prije svake epohe (korisno za izbjegavanje lokalnih minimuma). Za kraj ostaje još testirati rješenja.

Osim samog algoritma mogu se spomenuti i neka dobra svojstva na koja smo naišli prilikom gradnje konvolucijske neuronske mreže. Za početak možemo reći kako smo umjesto sigmoidne sktivacijske funkcije mogli koristiti i tangens hiperbolni (gotovo pa reskalirana verzija sigmoidne funkcije). Sama ideja traženja minimuma počiva na metodi gdje se postupno približavamo minimumu uz pomoć našeg koeficjenta učenja i verzije derivacije (koja ako je negativna gura pretragu “natrag”, dok u suprotnom smjeru gura pretragu “naprijed” prema minimumu). Prilikom samog kodiranja smo dosta koristili vektorizaciju[[1]](#footnote-1) (što je jedna dobra značajka MATLAB-a) koja nam je dodatno pomogla pri dobivanju brzine. Još neke tehnike optimizacije su bili *pooling* i česta uporaba konvoluiranja slika s nekim maskama (filterima). Dosta formalnih (pa i animiranih) zapisa se može naći po web-u[[2]](#footnote-2).

Za veliki dio implementacije konvolucijske neuronske mreže smo koristili razne tutorijale[[3]](#footnote-3) i opširan opis problema koju je na web-u omogućilo američko sveučilište Stanford[[4]](#footnote-4). Pri radu smo koristili brojne optimizacije s prethodno navedenih stranica. Našom implementacijom smo uspjeli dobiti impresivna rješenja i vrlo dobre postotke (uspješnost na samo 3 epohe je bila 97.75% točnosti uz ~20 min trajanje izvršavanja). Također, usporedili smo našu troslojnu konvolucijsku mrežu s još jednom mrežom[[5]](#footnote-5) koja je bila pobjednik u Competition "MFC/C++ Nov 2006"

gdje se implementirala peteroslojna mreža s uspješnosti od 99.18%. Dakle, rezultat (obzirom na broj linija koda i složenost) nije loš, ali može i bolje, dakako, u ovisnosti o broju implementiranih slojeva.

**Izvori :**

1. [**https://github.com/PedroCV/UFLDL-Tutorial-Solutions/tree/master/Additional\_3\_Convolutional\_Neural\_Network**](https://github.com/PedroCV/UFLDL-Tutorial-Solutions/tree/master/Additional_3_Convolutional_Neural_Network)
2. [**http://ufldl.stanford.edu/wiki/index.php/UFLDL\_Tutorial**](http://ufldl.stanford.edu/wiki/index.php/UFLDL_Tutorial)
3. [**http://www.codeproject.com/Articles/16650/Neural-Network-for-Recognition-of-Handwritten-Digi**](http://www.codeproject.com/Articles/16650/Neural-Network-for-Recognition-of-Handwritten-Digi)

Višeslojni perceptron :

Raspoznavanje rukom pisanih brojeva pomoću višeslojnog perceptrona MLP (*MultiLayer* *Perceptron*)

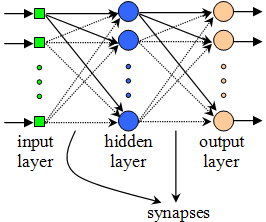
Opis zadatka:

Definirati strukturu višeslojnog perceptrona MLP za raspoznavanje rukom pisanih brojeva, opisati postupak učenja takve neuronske mreže te ocijeniti rezultate klasifikacije na velikom broju uzoraka.

Osnovni predmet proučavanja ovog poglavlja je posebna vrsta umjetnih neuronskih mreža – višeslojni perceptron (MLP). Uloga perceptrona očituje se u klasifikaciji i raspoznavanju uzoraka, a u ovom slučaju rukom pisanih brojeva.

MLP se sastoji od dva ili više slojeva procesnih elemenata koje nazivamo neuronima, a za proizvoljno kompleksan problem raspoznavanja dovoljna su tri sloja neurona. Aktivacijska funkcija koja opisuje ulazno – izlazni funkcijski odnos nelinearnosti pridružene neuronu je obično sigmoidalna: (derivabilna funkcija).

U ovom zadatku izvedba takve neuronske mreže ostvarena je u najmanje 3 sloja. Ta višeslojna mreža je potpuno povezana i aciklička, tj. nema povratnih veza između slojeva neurona. Ulazni sloj mreže sastoji se od 784 neurona koji predstavljaju piksele slike uzorka za učenje i klasifikaciju ( piksela), a izlazni sloj mreže ima 10 neurona koji predstavljaju razrede uzoraka: 0, 1, 2, ..., 9. Preostali neuroni čine skrivene slojeve mreže. U mjerenjima je obično trenirana i testirana mreža s jednim skrivenim slojem od 10 neurona. Također, svojstva MLP – a su propagacija funkcijskog signala unaprijed kroz mrežu te propagacija signala greške unatrag kroz mrežu.



sinapse mreže

ulazni sloj

skriveni sloj

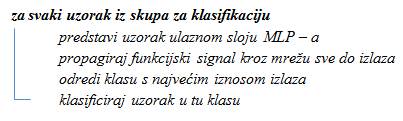
izlazni sloj

Slika. Strukturalni graf MLP – a

Učenje MLP – a temelji se na algoritmu učenja propagacije greške unatrag. U osnovi, učenje propagacijom greške unatrag sastoji se od dva prolaza kroz različite slojeve mreže: prolazak unaprijed i prolazak unatrag. U prolasku unaprijed ulazni vektor postavlja se na ulazne čvorove mreže i njegov učinak širi se dalje kroz svaki sloj mreže. Na kraju se generira skup izlaza kao konkretan odziv mreže. Tijekom prolaska unaprijed sve sinaptičke težine mreže su nepromjenjive. Tijekom prolaska unatrag sinaptičke težine se podešavaju u skladu s pravilom učenja korekcijom greške. Razlika željenog (ciljanog) odziva i trenutnog konkretnog odziva mreže predstavlja signal greške. Taj se signal greške dalje širi unatrag kroz mrežu. Sinaptičke težine podešavaju se tako da realni odziv mreže bude bliže željenom odzivu u statističkom smislu.

Ne postoje dobro definirani uvjeti zaustavljanja prethodno opisanog algortima, no postoje određeni razumni uvjeti koji se mogu iskoristiti za zaustavljanje procesa prilagođavanja težina kao što su: broj epoha, postignuta određena točnost raspoznavanja ili postignut minimalni iznos prosječne kvadratne greške po epohi. U ovom zadatku kriterij zaustavljanja procesa učenja je broj epoha, odnosno, svaki uzorak iz skupa za učenje mreži se predstavlja određeni broj puta. Parametri učenja mreže su brzina učenja η = 0.025, moment učenja γ = 0.05 (η i γ određeni eksperimentalno), 6 000 uzoraka svake pojedine znamenke iz skupa za učenje i njihov redoslijed predstavljanja ulaznim čvorovima mreže.

Ovako naučeni višeslojni perceptron korišten je u daljnjoj klasifikaciji rukom pisanih brojeva iz ispitne baze podataka. Pritom se postupak klasifikacije može prikazati pseudokodom u nastavku.



Slika. Pseudokod postupka klasifikacije pomoću MLP – a

Višeslojnom perceptronu predočeno je ukupno 70 000 uzoraka za raspoznavanje – 60 000 različitih uzoraka iz skupa za učenje i 10 000 uzoraka iz skupa za ispitivanje. Od toga broja točno je klasificiranih 58 102 uzoraka što čini statistički pokazatelj od 83 % točno raspoznatih i pravilno klasificiranih rukom pisanih brojeva. Za različit odabir parametara učenja i veličine skupa za učenje dobiveni su različiti iznosi ukupno uspješno raspoznatih brojeva. Pokazuje se vrlo bitnim koristiti što veći skup uzoraka za učenje sa što reprezentativnijim uzorcima pojedine znamenke.

U tablicama koje slijede prikazana je točnost raspoznavanja MLP – a u dva mjerenja s različitim brojem epoha i veličinom skupa za učenje.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ispravno klasificirani | neispravno klasificirani | ukupno | točnost raspoznavanja |
| skup za učenje | 5 507 | 4 493 | 10 000 | 55.07 % |
| skup za ispitivanje | 5 472 | 4528 | 10 000 | 54.72 % |
| ukupno | 10 979 | 9 021 | 20 000 | 54.90 % |

Tablica. Točnost raspoznavanja MLP – a (200 epoha, η = 0.025, γ = 0.05)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ispravno klasificirani | neispravno klasificirani | ukupno | točnost raspoznavanja |
| skup za učenje | 49 716 | 10 284 | 60 000 | 82.86 % |
| skup za ispitivanje | 8 386 | 1 614 | 10 000 | 83.86 % |
| ukupno | 58 102 | 11 898 | 70 000 | 83.00 % |

Tablica. Točnost raspoznavanja MLP – a (500 epoha, η = 0.025, γ = 0.05)

Iz prikazanoga može se zaključiti da veći broj uzoraka za učenje povećava uspješnost raspoznavanja ovog klasifikatora, a mogućnosti daljnjeg poboljšanja treba tražiti u predobradi ulaznih uzoraka (izdvajanje značajki uzoraka) te korištenju većeg broja MLP – a.

Literatura:

1. http://www.doc.ic.ac.uk/~nd/surprise\_96/journal/vol4/cs11/report.html#Pattern%20Recognition%20-%20an%20example
2. http://www.cs.sun.ac.za/~kroon/courses/machine\_learning/lecture5/mlp.pdf
3. http://www.zemris.fer.hr/projects/LicensePlates/english/Diplomski/kraupner-diplomski.pdf

SVD dekompozicija:

Raspoznavanje rukom pisanih brojeva pomoću SVD

Opis zadatka:

Razviti algoritam koji klasificira rukom pisane znamenke pomoću dekompozicije matrice, kojom je prikazan ulazni skup podataka, na singularne vrijednosti (SVD).

U oblikovanju željenog algoritma za raspoznavanje rukom pisanih brojeva pomoću SVD koriste se sljedeći rezultati.

Neka je A proizvoljna m n realna matrica, m n, kojom je prikazan ulazni skup podataka. Za matricu A postoji dekompozicija A = , gdje je U ortonormalna m n i V ortogonalna n n matrica, a = diag( su singularne vrijednosti.

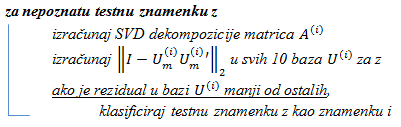
Stupci matrice U = [, … nazivaju se lijevi singularni vektori, a stupci matrice V = [, … desni singularni vektori. Lako se vidi da vrijedi: A =

Najbolja aproksimacija matrice A matricom ranga k je .

U ovom zadatku razmatra se primjena SVD – a na klasifikaciju rukom pisanih brojeva. Cilj je razviti algoritam koji u kratkom vremenu provodi klasifikaciju s visokom točnošću raspoznavanja. Svaka znamenka iz ispitne baze poistovjećuje se sa slikom rezolucije 28 28 piksela u *grayscale* formatu. Dakle, svakoj se znamenci može pridružiti matrica realnih brojeva iz segmenta [0, 1] dimenzije 28 28. Nadalje, svaka 28 28 matrica iz podskupa , gdje je skup za vježbu koji se sastoji od matrica koje predstavljaju istu znamenku, i = 0, 1, …, 9, može se pretvoriti u vektor dimenzije 784. Dobivenih vektora tvori stupce matrice dimenzija 784 .

Na ovaj način dobiveno je deset matrica , i = 0, 1, …, 9, te svaka rukom pisana znamenka *i* iz skupa odgovara nekom stupcu matrice . Koristeći dekompoziciju matrice zaključuje se da se svaka znamenka iz testne skupine može približno prikazati kao linearna kombinacija dominantnih singularnih vektora (kažemo da su singularni vektori dominantni smjerovi ako su singularne vrijednosti velike u odnosu na ). Preostalo je odrediti koliko je dobro neka nepoznata znamenka *z* iz skupa za testiranje opisana u deset različitih baza. Problem se svodi na računanje SVD dekompozicije matrica , čiji stupci predstavljaju znamenke iste vrste iz skupa za vježbu, te računanje norme reziduala u svih deset baza prema formuli: , gdje je matrica čiji su stupci prvih m dominantnih singularnih vektora koji pripadaju znamenci *i*. Ako je rezidual u bazi manji od ostalih, tada se nepoznata znamenka klasificira kao znamenka *i*.

Nakon detaljnog razmatranja jednostavno je oblikovati pseudokod algoritma:



Slika. Pseudokod algoritma za raspoznavanje rukom pisanih brojeva pomoću SVD – a

Algoritam je testiran na znamenkama iz baze MNIST. Skup za vježbu sastojao se od 60 000 klasificiranih rukom pisanih znamenki, a skup za testiranje sadržavao je 10 000 rukom pisanih znamenki. Algoritam je pokazao sjajne rezultate koji slijede u nastavku. Može se primijetiti da točnost raspoznavanja znamenki ovisi o broju m singularnih vektora koji se koriste pri računanju reziduala.

Dijagram. Točnost raspoznavanja rukom pisanih brojeva pomoću SVD – a

Literatura: 1. http://e.math.hr/br24/NovakEtAl

Naivni Bayesovski klasifikator :

Pretpostavimo da promatramo objekte sa različitih atributa i želimo svakog od njih klasificirati u jednu od kategorija iz skupa . Nazovimo skup objekata koje želimo klasificirati testni skup, i pretpostavimo da je zadan još jedan trening skup već klasificiranih objekata.

Objekt želimo klasificirati tako da mu pridružimo klasu za koju vrijedi

, odnosno

Direktno modeliranje ovih vjerojatnosti bi trajalo izuzetno dugo te bi nam trebala ogromna količina trening primjera, pa kod Naivnog Bayesovog klasifikatora imamo dodatnu pretpostavku da su atributi objekta međusobno nezavisni u odnosu na razred, odnosno

U stvarnosti, ova pretpostavka može često biti narušena, no pokazuje se da unatoč tome klasifikator daje dobre rezultate.

**NB klasifikator za problem prepoznavanja**

**rukom pisanih znamenki**

U našem problemu, slike znamenki su reprezentirane poljem bitova , gdje je ako je -ti piksel slike crne boje, a 0 ako je bijele boje.

Treniranje naivnog Bayesovog klasifikatora se svodi na računanje aproksimacija potrebnih vjerojatnosti na sljedeći način:

, gdje je T trening skup

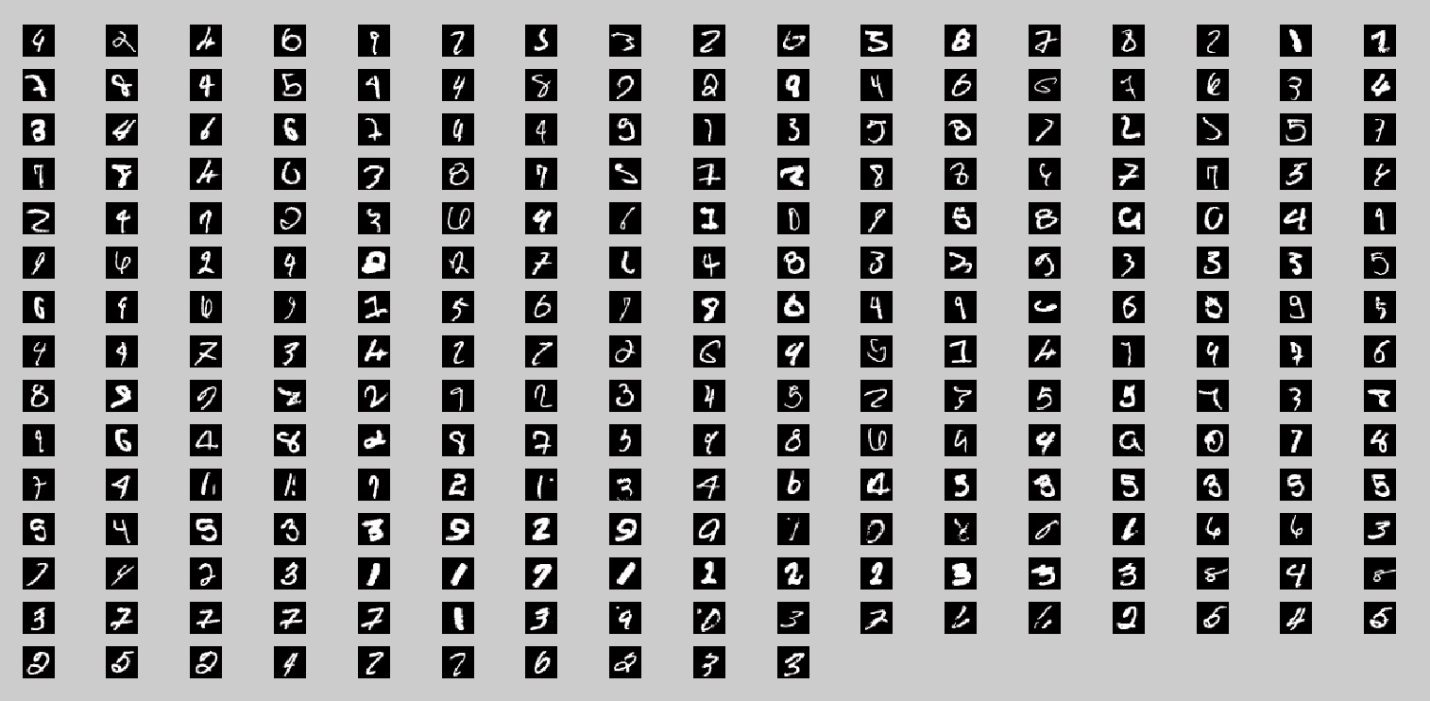
Ovakav model primijenjen na MNIST trening i testne skupove daje točnost od otprilike 84.5%.

Izvori :

1. <http://en.wikipedia.org/wiki/Naive_Bayes_classifier>
2. <http://software.ucv.ro/~cmihaescu/ro/teaching/AIR/docs/Lab4-NaiveBayes.pdf>
3. <http://www.youtube.com/watch?v=ZAfarappAO0>

**Kratka analiza:**

Za kraj se može pružiti i kratak pregled rezultata. Sama usporedba nam je bitna samo u segmentima gdje se klasificiraju znamenke. Naime, svugdje ćemo u praksi morati odraditi dio gdje se obavlja trening, ali kada to jednom uradimo gotovi smo (u pravilu trening najdulje i traje, ali smo rekli da nas to ne zanima). Rezultati pokazuju da je Naivni Bayesov klasifikator najsporiji kod klasifikcije (~140s za cijeli testni skup), njega slijedi konvolucijska neuronska mreža s ~45s, dok je najbrži SVD s ~35s. Uočiti možemo kako je jako mala razlika između SVD-a i konvolucijske neuronske mreže, ali bismo mogli pobjednikom proglasiti konvolucijsku mrežu (jer je točnija ~2% što u praksi ipak puno i znači). Prednost Bayesovog klasifikatora jest ta što je u njegovoj pozadini „najjednostavnija matematika“ (primijetimo da je kod u SVD implementaciji kraći, ali je Bayesovski model puno intuitivniji i, usudili bismo se reći, jednostavniji u odnosu na SVD). Također, Bayesov klasifikator bi mogao poslužiti kao dobra polazišna točka za bolje i brže algoritme.



Slika pokazuje znamenke koje je konvolucijska mreža krivo klasificirala (njih otprilike 200). Iz primjera je očito da bi dosta njih i čovjek netočno klasificirao (možda iz konteksta eventualno), pa nije razumno kriviti algoritam za te grješke, dok s druge strane moramo biti iskreni, pa reći da doista nije jasno kako pojedine znamenke (koje su doista lako i prepoznatljive) mreža nije uspjela ispravno klasificirati.

1. Činjenica jest da MATLAB radi s matricama samo i da brzina i efikasnost može ovisiti o vještini implementiranog koda. [↑](#footnote-ref-1)
2. Pogledati npr. : <http://ufldl.stanford.edu/wiki/index.php/Feature_extraction_using_convolution>

   ili <http://ufldl.stanford.edu/wiki/index.php/Pooling> [↑](#footnote-ref-2)
3. Npr. ovaj : https://github.com/PedroCV/UFLDL-Tutorial-Solutions/tree/master/Additional\_3\_Convolutional\_Neural\_Network [↑](#footnote-ref-3)
4. Radi se o ovim stranicama : 1) <http://ufldl.stanford.edu/wiki/index.php/UFLDL_Tutorial>

   2) http://openclassroom.stanford.edu/MainFolder/CoursePage.php?course=MachineLearning [↑](#footnote-ref-4)
5. Više detalja pogledati ovdje : http://www.codeproject.com/Articles/16650/Neural-Network-for-Recognition-of-Handwritten-Digi [↑](#footnote-ref-5)