Rukom pisani brojevi
Seminarski rad
Umjetna inteligencija

Jelena Držaić

13. siječnja 2016.

1 Uvod

Ljudski vidni sustav pravo je čudo evolucije. Pretpostavimo da nam je dan rukom napisan broj  . Većina će nas bez previše razmišljanja prepoznati da je dan broj jednak 504192. Na prvi pogled, problem prepoznavanja rukom pisanih brojeva, i znakova općenito, čini se kao lagan zadatak.
Složenost danog problema dolazi do izražaja kod pokušaja izrade efikasnog računalnog programa za rješavanje istog.
Poteškoća koja je već na prvi pogled vidljiva je predstavljanje zahtjeva koji su oblika "broj je jednak 3 ako se sastoji od dva polukružna oblika, položena vertikalno" u programu.
Nadalje, možemo se i zapitati kako "natjerati" računalo da razlikuje npr. rukom napisan broj 1 i 7?
U ovom seminaru prezentirano je nekoliko osnovnih tehnika kojima se dolazi do prihvatljivih rješenja problema, od kojih su neke potkrijepljene i rezultatima testiranja.

2 Motivacija

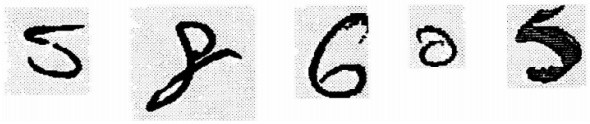
Problem prepoznavanja rukom pisanih znakova, uključujući i brojeva, puno je proučavan upravo iz razloga što nalazi svoju primjenu u mnogo stvarnih situacija. Neke od primjena su:

- Prepoznavanje poštanskih brojeva kod sortiranja pisama u poštanskim službama
- Detektiranje rukom napisanih iznosa kod bankovnih uplata
- Prepoznavanje rukom napisanih matematičkih formula

3 Varijante problema

3.1 Prepoznavanje izoliranih znamenaka

Ova varijanta problema osnovna je verzija na koju se u tijeku procesiranja svedu i kompliciranije varijante. Želimo svaku sliku znamenke koju zadajemo kao unos pripojiti odgovarajućoj vrijednosti od 0 do 9. Jednostavnije, prepoznajemo rukom pisane znamenke sa slika.
Rješenje ovako zadanog problema možemo direktno primjeniti kod prepoznavanja brojeva na "formama" u kojima već postoje definirana mjesta za upisivanje pojedinih znamenaka (npr. pravokutnici na uplatnici).

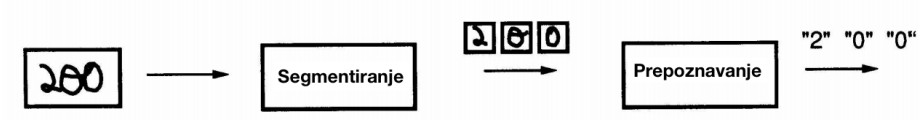


Slika 1: Primjer izoliranih znamenaka

Algoritmi koji su testirani u sklopu ovog seminara rješavaju ovu varijantu problema.

3.2 Prepoznavanje višeznamenkastih brojeva

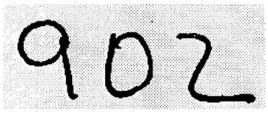
U praksi, najčešće je potrebno prepoznati višeznamenkaste brojeve (npr. poštanske brojeve).
Pretpostavimo da smo razvili algoritam za prepoznavanje izoliranih znamenaka, te ga želimo primjeniti na rješavanje ove varijante problema. Lako je uočiti da nam je za to potrebna još jedna dodatna faza pretprocesiranja.
Njena je zadaća da iz slike koja sadržava niz znamenaka izolira pojedinačne znamenke. Tu fazu nazivamo *segmentiranje* (engl. segmentation).



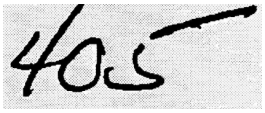
Slika 2: Jednostavan proces prepoznavanja višeznamenkastog broja

3.2.1 Primjer algoritma segmentiranja

Objasnimo jedan od mogućih načina za segmentiranje, koji, iako nije primjeren za univerzalnu upotrebu, služi kao demonstracijski primjer.
Segmentaciju možemo provesti na način da promatramo "komponente povezanosti" na danoj slici, tj. kontinuirane skupove crnih piksela.
Vidimo da u slučaju slike *(3a)* zaista dobivamo korektne segmente - izolirane znamenke. Nažalost, vidljivo je da sliku *(3b)* na ovaj način ne bismo mogli segmentirati, već bi za to bio potreban neki sofisticiraniji pristup.



Slika 3: (a)



Slika 3: (b)

4 Osnovni pristupi problemu

U nastavku su dana dva osnovna tipa algoritama prema načinu zadavanja ulaza.

4.1 Offline prepoznavanje

Prepoznavanje znakova nakon što je pisanje istih dovršeno. Uzorak kod offline prepoznavanja sadrži samo statičke podatke, tj. iz uzorka dobivamo samo informaciju o boji (svjetlini) pojedinog piksela.
Ne koristi se dinamička informacija o zapisu uzorka (potez olovke, redoslijed pisanja brojeva...).

4.2 Online prepoznavanje

Nasuprot offline prepoznavanju, online tehnike prepoznavanja evaluiraju korisnikov unos za vrijeme zadavanja istog, tj. koristimo dinamičku informaciju - potez olovke i sl.
Ova tehnika koristi se npr. kod pisanja rukom ili olovkom na tabletu ili mobilnim uređajima.

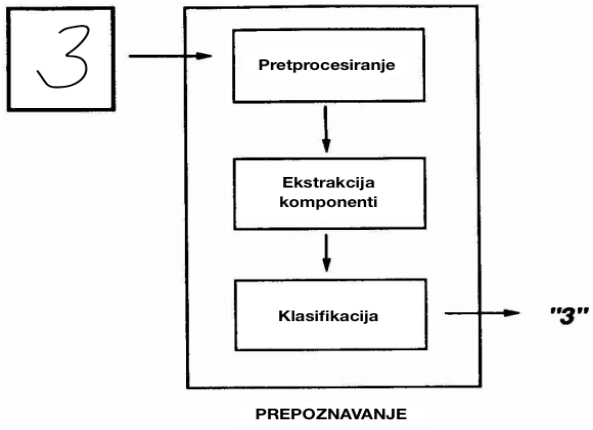
Slika 4: (a) Offline prepoznavanje

Slika 4: (b) Online prepoznavanje

U nastavku je dan detaljniji pregled komponenti algoritama za prepoznavanje rukom pisanih znamenaka te konkretni algoritmi za rješavanje tog problema.

5 Struktura algoritma

U ovom odjeljku dana je tipična struktura algoritma za prepoznavanje rukom pisanih znamenaka.



Slika 5: Struktura algoritma za prepoznavanje znamenaka

5.1 Pretprocesiranje

Pretprocesiranje bismo mogli slikovito predočiti kao "odvajanje signala od okolne buke", kako bi se olakšalo daljnje procesiranje. U slučaju prepoznavanja znamenaka, to nisu samo vanjski čimbenici kao što je odsjaj na slici ili "mrlje", već i neke, na prvi pogled neočigledne značajke. Neke od njih su:

- Skaliranje slika na standardnu veličinu, odgovarajuću za daljnje korake algoritma.
- "Grey-level normalization" - Zatamnjivanje/posvjetljivanje piksela na odgovarajući način.
- Svođenje slike na oblik u kojem "potez olovke" ima odgovarajuću širinu (npr. 1 px).

Način pretprocesiranja uvelike ovisi o potrebama kasnijih koraka algoritma.

5.2 Ekstrakcija komponenti

U ovom koraku algoritma izdvaja se skup značajki koje će u fazi klasifikacije služiti kao kriteriji za razlikovanje skupa pojedinih znamenaka.
Ova faza daje takozvani *vektor značajki* ("feature vector").

Kod nekih algoritama ovaj dio se preskače i klasifikacija se vrši direktno na zadanim slikama. Tada *vektor značajki* možemo poistovjetiti s vektorom čija jedna komponenta opisuje jedan piksel slike.

5.3 Klasifikacija

Klasifikacija nam daje pripadnost ulazne znamenke jednoj od danih klasa (konkretno, jednoj od znamenaka 0-9). Navedimo tri osnovne vrste klasifikatora.

5.3.1 Podudaranje s predloškom ("template matching")

Karakteriziramo vektor značajki (danu znamenku) s obzirom na njegovu *udaljenost* od skupova već otprije poznatih uzoraka. Znamenku pridružujemo skupu kojem je najbliža, dok sigurnost s kojom to radimo računamo kao neku funkciju dobivenih udaljenosti do skupova. Primjer jednog takvog algoritma dan je u 6.2.

5.3.2 Statistički klasifikatori

Koriste skup primjera (u ovom slučaju skup rukom pisanih znamenaka) kako bi se odredila distribucija u različite klase. Nakon što je uzorak "istreniran", za novi primjer se određuje vjerojatnost da pripada nekoj klasi.
Jedan od predstavnika ove klase algoritama su i *algoritmi neuronskih mreža*, čiji je jedan primjer detaljnije objašnjen i testiran u 6.1., po predlošku iz [1].

5.3.3 Sintaktičke metode

Opisuju svaku klasu (svaku znamenku 0-9) pomoću specifičnih kombinacija značajki. Skup "legalnih" kombinacija za neku klasu definiran je zasebnim pravilima. Kao primjer, mogli bismo opisati znamenku 3 kao dvije polukružnice, položene jedna na drugoj.

6 Testiranje

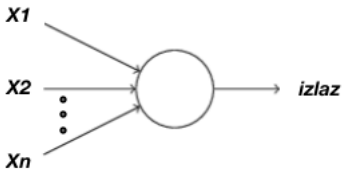
Kao zadnji dio seminara, testirana su dva algoritma za prepoznavanje rukom pisanih znamenaka.

6.1 Algoritam neuronskih mreža

Za razumijevanje algoritma objasnimo prvo nekoliko osnovnih pojmova.

Perceptron

Tip umjetnog neurona, razvijen 1950-ih i 1960-ih godina od strane znanstvenika Franka Rosenblatta, a inspiriran ranijim radom W. McCullocha i W. Pittsa. Perceptron kao unos prima bitove x_1, x_2, ..., x_n te kao izlaz ima bit *izlaz*. Rosenblatt uvodi pojam težina $w_1, w_2, ..., w_n$ kojima označavamo kolika je važnost ulaza x_i za izlaz. Izlaz perceptrona jednak je 1 ili 0, ovisno o sumi $\sum_{i=1}^{n} w_i x_i$.

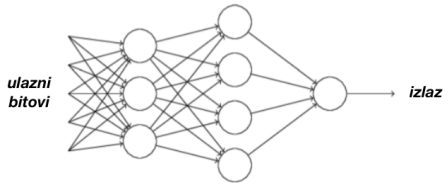


Slika 6: Model perceptrona

Točnije, $izlaz = \begin{cases} 0, ako \sum_{i=1}^{n} w_i x_i \leq prag \\ 1, ako \sum_{i=1}^{n} w_i x_i > prag \end{cases}$,

gdje je *prag* neka zadana vrijednost.

Prikažimo sada shematski model neuronske mreže.

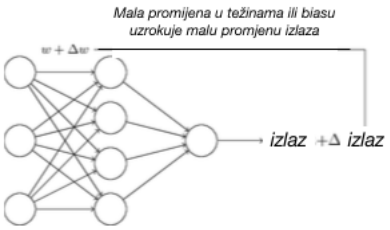


Slika 7: Shematski prikaz jednostavne neuronske mreže

Vidimo da prvi(lijevi) sloj perceptrona donosi odluke na temelju ulaza, srednji sloj kao ulaz prima izlaz prvog sloja itd. Primjetimo da izlaz jednog perceptrona moze biti ulaz za vise preceptrona.
Definirajmo sada nagib ("bias") $b = -prag$. Gornji uvjet za izlaz perceptrona sada možemo zapisati kao $output = \begin{cases} 0, w \cdot x + b \leq 0 \\ 1, w \cdot x + b > 0 \end{cases}$ Vidimo da nam b zapravo kaže koliko je teško dobiti 1 kao izlaz perceptrona. Ako je b velik, vjerojatnost za to je visoka, ako je b relativno mali, vjerojatnost za to je niska. Drugi način na koji možemo promatrati perceptrone je i kao jednostavne logičke sklopove.

Sigmoidni neuroni

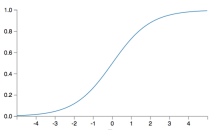
Cilj je razviti neuronsku mrežu za prepoznavanje rukom pisanih znamenaka. Jedan problem koji se javlja kod perceptrona je taj da mala promjena u težini može promijeniti izlaz za veliki iznos (0 u 1). Dobro svojstvo neurona bilo bi to da mala promjena u težinama uzrokuje male promjene u izlazu, budući da bismo tada mogli kontrolirano podešavati parametre neuronske mreže. Upravo taj uvjet dobivamo uvođenjem *sigmoidnih neurona*. Struktura te vrste neurona jednaka je strukturi sa slike (6), uz modifikaciju u odnosu na perceptrone da izlaz nije bit, već realni broj između 0 i 1, rezultat funkcije $(1 + e^{-w \cdot x - b})^{-1}$. Skiciranjem funkcije vidimo da je to zapravo "zaglađena" verzija step funkcije.



Slika 6: Model perceptrona

Neuronska mreža

Mreža za prepoznavanje rukom pisanih znamenaka koju konstruiramo je "feedforward" mreza i sastojat će se od 3 sloja. "Feedforward" označava svojstvo mreže da je izlaz neurona iz i-tog sloja uvijek ulaz neurona j-tog sloja, $j > i$, tj. nema "vraćanja unazad" kroz mrežu.



Slika 7: funkcija sigmoida

Prvi sloj mreže sastojat će se od piksela slike
zadanog primjera - vrijednost 0.0 za bijeli piksel, 1.0 za crni piksel, vrijednost iz $\langle 0, 1\rangle$ za sive piksele. Srednji ("hidden") sloj sadržavat će varijabilni broj neurona. Zadnji, izlazni sloj sastojat će se od 10 neurona (po jedan za svaku znamenku 0-9).
Neka je x jedan primjer za treniranje, $y := y(x)$ definiramo kao vektor dimenzije 10 koji označava stvarnu vrijednost primjera. Npr. ako je dana znamenka 4, $y = (0, 0, 0, 0, 1, 0, 0, 0, 0, 0)^T$. Vidimo da problem sada možemo formirati kao problem minimizacije funkcije $C(w, b) = \frac{1}{2n} \sum_x ||y(x) - a(w, b, x)||^2$, tzv. "funkcije cijene". Ovdje je n broj primjera za testiranje, $a(w, b, x)$ vektor izlaza algoritma s parametrima (w, b) za ulaz x, a suma je po svim x iz skupa primjera za testiranje.
Za minimizaciju koristimo stohastičku metodu najbržeg silaska. Metoda funkcionira na način da odaberemo slučajnim odabirom manji skup primjera iz skupa za treniranje, tzv. "mini-batch", treniramo neuronsku mrežu tim primjerima; postupak ponavljamo dok ne iscrpimo sve primjere - završimo jednu epohu. Treniranje neuronske mreže završava nakon zadanog broja epoha. Promotrimo sada rezultate algoritma za određene parametre.

Rezultati

U tablici su dani rezultati testiranja. U prvom stupcu dan je broj neurona u srednjem sloju neuronske mreže, u drugom stupcu broj epoha, u trećem broj neurona u svakom "mini-batch" skupu, a u posljednjem tzv. "learning rate" parametar η korišten kod računanja funkcije silaska.

n-srednji sloj	epohe	η	"mini-batch"	točnost
30	25	3.0	10	95.16%
100	30	3.0	10	96.49%
50	50	3.0	10	95.64%
50	100	3.0	15	95.7%

Resursi

Za testiranje algoritama korištena je baza rukom pisanih znamenaka MNIST, dostupna na `http://yann.lecun.com/exdb/mnist/`. Slike znamenaka u toj bazi su skalirane i centrirane te su zbog toga prikladne za testiranje algoritama.

6.2 Algoritam temeljen na [2]

U članku [2] predstavljen je algoritam za pretraživanje tekstualnih dokumenata. Bazira se na SVD dekompoziciji, te aproksimaciji matrice matricom manjeg ranga. Sličan algoritam može se prilagoditi za problem prepoznavanja rukom pisanih znamenki. Aproksimacija matricom nižeg ranga koristi se kako bi se iz ulaznih primjera eliminirala "buka". U nastavku je dana ideja algoritma i rezultati testiranja.

Ideja algoritma

Neka A=[S_1 S_2 ... S_n] matrica, u kojoj je j-ti stupac, S_j, reprezentacija

jedne znamenke (po jedna komponenta za svaki piksel) iz skupa primjera za treniranje. Definiramo X_i kao matricu sastavljenu od stupaca matrice A koji reprezentiraju znamenku i. Sada koristeći SVD i aproksimaciju matricom nižeg ranga r dobivamo potprostore dimenzije r za pojedine znamenke kao slike tih aproksimacija.
Potprostor kojem pripada neki test primjer određujemo tako da računamo udaljenost primjera (reprezentiranog kao vektor stupac) od svakog od konstruiranih potprostora i odaberemo potprostor do kojeg je udaljenost najmanja.

Rezultati

rang aproksimacije	točnost
5	90.28%
6	90.53%
10	93.17%
14	93.92%
15	94.16%
18	94.17%

7 Kodovi

Uz ovaj seminar priloženi su i kodovi testiranih algoritama.

- network.py - python implementacija neuronske mreže.
- mnist.py - učitavanje primjera iz MNIST baze, python kod.
- digits.m - MATLAB kod algoritma iz 6.2.

8 Literatura

[1] Michael A. Nielsen, *Neural Networks and Deep Learning*, Determination Press, 2015.

[2] M. W. Berry, Z. Drmač, E.R. Jessup, *Matrices, Vector Spaces, and Information Retrieval*, Society for Industrial and Applied Mathematics, 1999.

[3] O. Matan, J. Bromley, C.J.C. Burges, J.S. Denker, L.D. Jackel, Y.L. Cun, E.P.D. Pednault, W.D. Satterfield, C.E. Stenard, T.J. Thompson, *Reading Handwritten Digits: A Zip Code Recognition System*, 1991.

[4] `https://en.wikipedia.org/wiki/Handwriting_recognition`, (pristupano 9.1.2016.)