Raspoznavanje rukom pisanih znakova

Andrea Perčinlić
Matematički odsjek
Prirodoslovno-matematički fakultet
Sveučilište u Zagrebu
Email: andrea.percinlic@gmail.com

Ivana Tkalčec
Matematički odsjek
Prirodoslovno-matematički fakultet
Sveučilište u Zagrebu
Email: ivana.tk@gmail.com

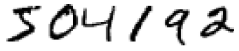
***Sažetak*—U radu se bavimo različitim algoritmima za prepoznavanje znamenaka. To su neuronske mreže, dekompozicija matrice na singularne vrijednosti (SVD) i algoritam k-najbližih susjeda (k-nn). Svaki od algoritama ukratko je objašnjen s teorijske strane te je objašnjena naša primjena na dani problem. Algoritmi su testirani uz različite značajke koje su vezane za svaki od algoritama te su na temelju toga donešeni određeni zaključci. Programska izvedba algoritama je realizirana u MATLAB - u.**

I. Uvod

A. Motivacija

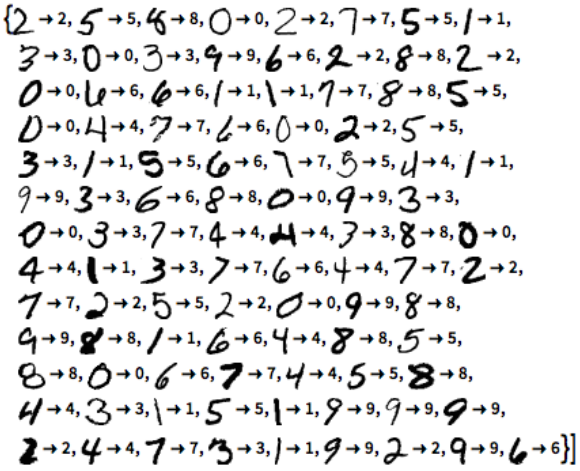
Potreba za računalnim programom koji će prepoznavati brojeve originalno se pojavila u američkoj pošti gdje treba sortirati pisma po poštanskom broju odredišta. U Americi postoji više od 31500 poštanskih ureda te ako svaki ured obrađuje pisma tada je potreban barem jedan radnik za svaki od ureda koji će provesti neko vrijeme sortirajući ih. S obzirom da poštanskim uredima prolazi 23 milijuna pisma svaki sat lako je zaključiti da se postojanjem računalnog programa koji to radi umjesto ljudi uštedi nevjerojatna količina posla, ali i novca.

Ljudski vizualni sustav jedan je od najsavršenijih u prirodi. Pogledajmo sljedeći slijed znamenaka:



Većina ljudi bez imalo truda može prepoznati broj 504192. No jednostavnost prepoznavanja je varljiva. U obje hemisfere našeg mozga imamo primarni vizualni korteks $V1$ koji sadrži 140 milijuna neurona s desecima milijardi veza među njima. K tome, osim korteksa $V1$ tu su još i vizualni korteksi $V2$, $V3$, $V4$ i $V5$ koji rade progresivno na još složenijim procesuiranjima slika. Dakle, moglo bi se reći da u svojim glavama imamo superračunalo podešavano milijunima godina kako bi što bolje razumjelo vizualni svijet. Stoga, mi ljudi imamo nevjerojatno dobar osjećaj za to što vidimo, a sav taj rad obavimo nesvjesno te pritom najčešće ni ne cijenimo kako teške probleme naš vizualni sustav rješava.

Teškoće u prepoznavanju vizualnog uzorka postaju očite želimo li da računalo prepoznaje brojeve. Ono što se dosad činilo lako, najednom postaje nevjerojatno teško. Jednostavna intuicija kojom prepoznajemo oblike (npr. 9 ima kružić gore i polukružni završetak dolje desno) nije više tako jednostavna kad je trebamo izraziti algoritamski. Poteškoće se javljaju i zbog činjenice da računalo treba prepoznati rukom pisane znakove različitih ljudi. Stoga se treba nositi s različitim veličinama, oblicima i stilovima pisanja istih znakova (Slika 1). Također, ti znakovi podliježu slučajnim promjenama koje se nazivaju šumom, osobito blizu rubova pa računalni program znak s velikim šumom može interpretirati kao posve drugi znak.



Slika 1. Prepoznavanje rukom pisanih znamenaka

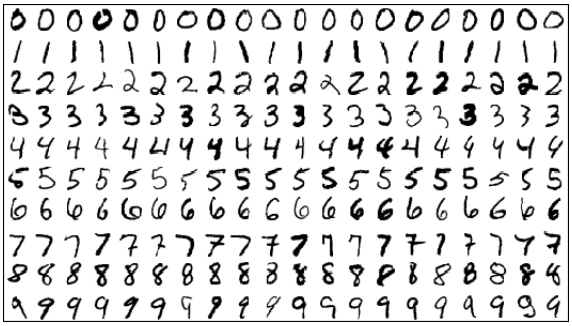
B. Ciljevi

Cilj ovog projekta je pronaći koji od naših algoritama najbolje klasificira dani skup znamenaka. Koristit ćemo tri različita algoritma, a to su neuronska mreža, SVD i k-nn. Pritom ćemo za svaki od algoritama provjeriti uz koje značajke daje najbolje rezultate. Odnosno, za neuronsku mrežu isprobat ćemo koliko je etapa dovoljno da uspješno nauči prepoznavati znamenke te da se ne prenauči. Za SVD ćemo mijenjati broj singularnih vektora kako bismo usporedile koliko je vektora dovoljno, a koliko previše za uspješno prepoznavanje. Dok ćemo za k-nn isprobavati algoritam s različitim k-ovima te donijeti odluku koji je k optimalan za naš problem.

II. Opis problema (skup podataka)

U ovom radu bavimo se problemom koji se proučava već više od trideset godina. Riječ je o problemu raspoznavanja rukom pisanih znakova. Raspoznavanje rukom pisanih znakova je sposobnost računala da interpretira rukom pisane ulaze. Razlikujemo off – line i on – line prepoznavanje znakova. U off – line pristupu podaci koji se koriste su statička reprezentacija znaka, dok su u on – line pristupu podaci signali koji se konvertiraju u kod koji zatim prepoznaje računalo. Ovdje se razmatra off – line pristup za rješavanje opisanog problema.

Skup podataka koji se koristi u ovom radu čine slike iz MNIST[1] – ove baze podataka rukom pisanih znamenki[2]. Svaka slika reprezentira neku od znamenki: $0, 1, 2, 3, 4, 5, 6, 7, 8$ ili 9, i pridružena joj je odgovarajuća oznaka. Ta baza podataka sadrži trening skup od 60000 i test skup od 10000 primjera. Ti skupovi su međusobno disjunktni. MNIST – ova baza podataka je konstruirana iz NIST – ovih baza *Special Database 3* i *Special Database 1*[3]. Originalne binarne slike veličine 128×128 piksela iz NIST – ovih baza su normalizirane tako da stanu u matricu od 28×28 piksela, zato su rezultirajuće slike u MNIST – ovoj bazi u *grayscale* formatu i veličine 28×28 piksela. Metode koje koristimo za rješavanje problema ne zahtijevaju dodatno pretprocesiranje i formatiranje slika iz ovoga skupa podataka.



Slika 2. Primjeri rukom pisanih znamenki iz MNIST - ove baze podataka

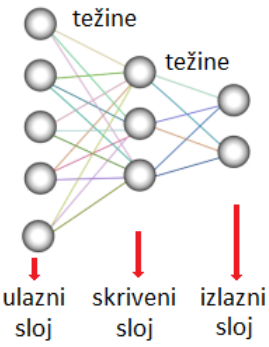
III. Metode za rješavanje problema

A. Neuronska mreža

Za rješavanje uvodnog problema upotrijebili smo posebnu vrstu umjetnih neuronskih mreža – višeslojni perceptron (MLP[4]). Najznačajnija uloga takve neuronske mreže očituje se u klasifikaciji i raspoznavanju uzoraka. MLP se sastoji od dva ili više slojeva procesnih elemenata koje nazivamo neuronima, a za proizvoljno kompleksan problem raspoznavanja dovoljna su tri sloja neurona[5]. Također, svojstva MLP – a su propagacija funkcijskog signala unaprijed kroz mrežu te propagacija signala greške unatrag kroz mrežu. Aktivacijska funkcija koja opisuje ulazno – izlazni funkcijski odnos nelinearnosti pridružene neuronu je sigmoidalna. Sigmoidalna funkcija je oblika:

$$f(x) = \frac{1}{1 + e^{-x}}.$$

U ovom zadatku izvedba MLP – a ostvarena je u 3 sloja. Ta 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 primjera za učenje i testiranje. (Svaka znamenka je reprezentirana slikom veličine 28×28 piksela.) Izlazni sloj mreže ima 10 neurona koji predstavljaju razrede uzoraka: $0, 1, 2, \ldots, 9$. Preostali neuroni čine jedan skriveni sloj mreže. Obično je trenirana i testirana mreža s jednim skrivenim slojem od 30 ili 100 neurona.



Slika 3. 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 težine mreže su nepromjenjive. Tijekom prolaska unatrag 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. Težine se podešavaju tako da realni odziv mreže bude bliže željenom odzivu u statističkom smislu.

Ne postoje dobro definirani uvjeti zaustavljanja prethodno opisanog algoritma, 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 $\eta = 0.025$[6], 60000 primjera iz skupa za učenje i njihov redoslijed predstavljanja ulaznom sloju mreže.

Ovako naučeni model višeslojnog perceptrona korišten je u klasifikaciji rukom pisanih brojeva iz testnog skupa podataka. Postupak klasifikacije može se opisati na sljedeći način:

1) Predstavi primjer iz testnog skupa ulaznom sloju MLP - a.
2) Propagiraj funkcijski signal kroz mrežu sve do izlaza.
3) Odredi klasu s najvećim iznosom izlaza.
4) Klasificiraj uzorak u tu klasu.

[1] MNIST je kratica za *Mixed National Institute of Standards and Technology*.

[2] Baza podataka dostupna je ovdje: http://yann.lecun.com/exdb/mnist/.

[3] *Special Database 3* je prikupljena među zaposlenicima Census Bureau, a *Special Database 1* među učenicima srednjih škola. Primjeri u *SD – 3* su mnogo čišći i jednostavniji za prepoznati, zato MNIST – ov trening skup sadrži 30000 primjera iz *SD – 3* i 30000 primjera iz *SD – 1* od oko 250 različitih pisaca. MNIST – ov test skup sadrži 5000 primjera iz *SD – 3* i 5000 primjera iz *SD – 1*. Ti skupovi su međusobno disjunktni.

[4] MLP (*engl.* MultiLayer Perceptron) je oznaka kojom ćemo označavati višeslojni perceptron u nastavku.

[5] Svaka neuronska mreža ima tri vrste slojeva: ulazni, skriveni i izlazni. Za problem raspoznavanja uzoraka dovoljan je jedan skriveni sloj. Izvor.

[6] Parametar η određen je eksperimentalno.

B. SVD

Dekompozicija matrice na singularne vrijednosti (SVD) važna je dekompozicija matrice i s teorijske i s praktične strane. U sljedećem teoremu dana je definicija SVD dekompozicije te tvrdnja o njezinoj egzistenciji za svaku matricu.

Teorem 3.1: (SVD) Neka su m i n ($m \geq n$) prirodni brojevi te A proizvoljna $m \times n$ realna matrica. Tada postoji dekompozicija $A = U\Sigma V^\tau$, gdje je U ortonormalna $m \times n$ matrica i V ortogonalna $n \times n$ matrica, a $\Sigma = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n)$, sa $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0$.

Definicija 3.2: Stupce matrice $U = [u_1, \ldots, u_n]$ nazivamo lijevi singularni vektori, a stupce matrice $V = [v_1, \ldots, v_n]$ nazivamo desni singularni vektori. Brojeve σ_i nazivamo singularne vrijednosti.

Napomena 1. U slučaju kada je $m < n$, SVD definiramo za matricu A^τ.

Lako se vidi da vrijedi: $A = U\Sigma V^\tau = \sum_{i=1}^{n} \sigma_i u_i {v_i}^\tau$. Najbolja aproksimacija matrice A matricom ranga k je:

$$A_k = \sum_{i=1}^{k} \sigma_i u_i {v_i}^\tau.$$

Nadalje, kažemo da su singularni vektori $u_1^{(i)}, \ldots, u_m^{(i)}$ dominantni smjerovi ako su singularne vrijednosti $\sigma_1, \ldots, \sigma_m$ velike u odnosu na $\sigma_{m+1}, \ldots, \sigma_n$.

S obzirom da se bavimo problemom klasifikacije 10 različitih znamenaka, imamo 10 skupova T_i od kojih svaki sadrži skup odgovarajućih znamenaka $(i-1)$. Skup T_i je predstavljen matricom u kojoj svaki stupac predstavlja jednu sliku odgovarajućeg broja. Prema Teoremu 3.1 slijedi da za svaku matricu postoji dekompozicija pa provođenjem algoritma SVD lako dobijemo matricu $U^{(i)}$ koja predstavlja skup singularnih vektora matrice T_i. Sada se svaka znamenka iz testne skupine može približno prikazati kao linearna kombinacija dominantnih singularnih vektora. Preostalo je odrediti koliko je dobro neka nepoznata znamenka z iz skupa za testiranje opisana u deset različitih baza. Stoga računanamo norme reziduala u svih deset baza $U^{(i)}$ prema formuli: $\|(I - U_m^{(i)} U_m^{(i)\tau})z\|_2$, gdje je $U_m^{(i)} = [u_1^{(i)}, \ldots, u_m^{(i)}]$ matrica čiji su stupci prvih m dominantnih singularnih vektora koji pripadaju znamenci i. Ako je rezidual u bazi $U^{(i)}$manji od ostalih, tada se nepoznata znamenka klasificira kao znamenka i.

Jednostavno je oblikovati pseudokod algoritma:

1) izračunaj SVD dekompozicije matrica T_i
2) izračunaj $\|(I - U_m^{(i)} U_m^{(i)\tau})z\|_2$ u svih 10 baza $U^{(i)}$ za z
3) ako je rezidual u bazi $U^{(i)}$ manji od ostalih, klasificiraj testnu znamenku z kao znamenku i

Algoritam je izvršen za različite vrijednosti broja m.

C. k-nn

Algoritam k-najbližih susjeda (k-nn) jedan je od osnovnih klasifikacijskih algoritama te ga je jednostavno implementirati. Ideja algoritma k-nn je da se novi primjer klasificira tako da se pogledaju njemu najbliži primjeri iz skupa za učenje te on dobiva vrijednost ciljnog atributa koja je najčešća u njegovom susjedstvu. Pogledajmo sada od kojih se koraka sastoji algoritam. Prvo je potrebno izračunati udaljenost između novog primjera X i svih primjera Y iz skupa za učenje T što radimo koristeći neku metriku. Najčešće se koristi euklidska metrika, gdje je formula za udaljenost:

$$d(X, Y) = \sqrt{\sum_{i=1}^{k} (x_i - y_i)^2}$$

no ovisno o skupu primjera može se koristiti i neka druga, primjerice Manhattan:

$$d(X, Y) = \sum_{i=1}^{k} |x_i - y_i|$$

ili općenitije Minkowski:

$$d(X, Y) = \left(\sum_{i=1}^{k} (|x_i - y_i|)^q \right)^{\frac{1}{q}}.$$

Primijetimo kako se ove formule za udaljenost ne mogu koristiti uvijek. Kada imamo kategoričke varijable, primjeri atributa su spol ili brak, tada koristimo Hamming udaljenost:

$$d_H(X, Y) = \sum_{i=1}^{k} |x_i - y_i|$$
$$x = y \Rightarrow d = 0$$
$$x \neq y \Rightarrow d = 1$$

S obzirom da u problemu kojim se mi bavimo imamo samo numeričke atribute koristimo euklidsku udaljenost. Sljedeći korak k-nn algoritma je odrediti k-najbližih susjeda primjera x iz skupa za učenje T. A zadnji korak je pridijeliti primjeru x klasu koja je najčešća između k-najbližih susjeda. Za dani primjer x s nepoznatom klasifikacijom, neka $x_1, x_2, \ldots, x_k$ označavaju k primjera koji su najbliži x. Tada vrati

$$h(x) = \underset{c \in \{0,1,\ldots,k\}}{\arg\max} \sum_{1}^{k} \delta(c, f(x_i))$$

gdje je $\delta(x, y) = 1$ ako $x = y$, a $\delta(x, y) = 0$ inače. Dakle, $h(x)$ je najčešća vrijednost ciljne funkcije koja se pojavljuje među k primjera za učenje koji su najbliži upitu x.

Atributi obično imaju vrlo različite intervale vrijednosti, stoga je potrebno provesti normalizaciju numeričkih atributa na interval $(0, 1)$:

$$a_i' = \frac{a_i - min(a_i)}{max(a_i) - min(a_i)}$$

ili standardizaciju numeričkih atributa:

$$a_i^s = \frac{a_i - \bar{a}_i}{stdev(a_i)}.$$

Ukoliko primjeri imaju velik broj atributa od kojih je puno nevažnih potrebno je odrediti težinski faktor za svaki atribut. U našem algoritmu nije bilo potrebe za time s obzirom da su vrijednosti atributa od 0 do 1.

Kako odabrati broj k? Brojem k definiramo složenost modela. Općenito se rastom k smanjuje varijanca, ali se

povećava pristranost. Prevelik k može potpuno poništiti prednosti lokalne estimacije, jer je prisiljen koristiti sve udaljenije primjere. Algoritam kojim smo testirale naš skup podataka koristio je za k sve vrijednosti iz skupa $\{1, 2, \ldots, 50\}$ kako bismo usporedile koji k daje najbolje, a koji najlošije rezultate.

Zašto koristiti k-nn klasifikator? Veoma je jednostavan za implementaciju, gotovo optimalan za jako velik broj primjera, vrlo je adaptivan (koristi lokalnu informaciju) te je pogodan za paralelnu implementaciju. Mane su trajanje u procesu klasifikacije te činjenica da zahtijeva puno memorije.

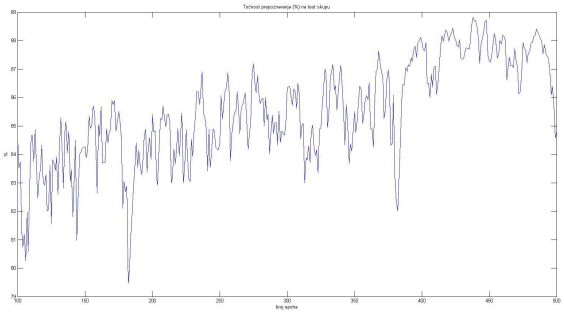
IV. Rezultati

A. Neuronska mreža

Višeslojnom perceptronu predočeno je ukupno 60000 primjera iz trening skupa. Proces učenja mreže zaustavljen je nakon određenog broja epoha. Nakon što je proces učenja zaustavljen, naučenom modelu mreže predočeno je 10000 primjera iz test skupa.

Testirane su dvije različite konfiguracije mreže. Eksperimentalno je utvrđeno da je najbolji parametar brzine učenja 0.025. Prevelike ili premale vrijednosti toga parametra čine mrežu lošijom od slučajnog klasifikatora.

U prvom modelu mreže skriveni sloj sastojao se od 30 neurona, a broj epoha kretao se od 100 do 500. Točnost prepoznavanja na test skupu maksimalno iznosi 88.81%, a postiže se nakon 434 epohe. Grafički prikaz je na Slici 4.



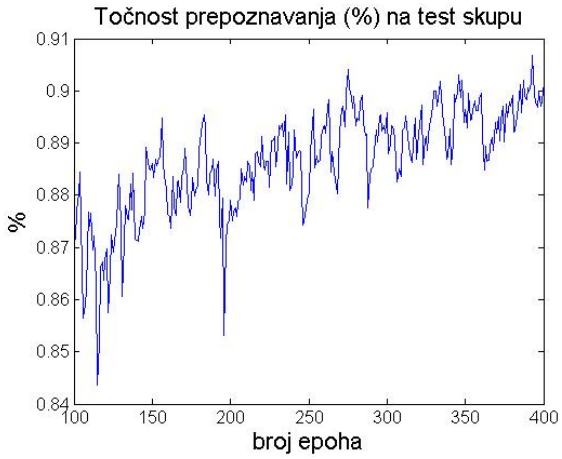
Slika 4. Točnost prepoznavanja (%) prvog modela na test skupu

U drugom modelu mreže skriveni sloj sastojao se od 100 neurona, a broj epoha kretao se od 100 do 500. Točnost prepoznavanja na test skupu maksimalno iznosi 90.67%, a postiže se nakon 394 epohe. Grafički prikaz je na Slici 5.

Jedan od problema koji može nastupiti tijekom učenja neuronske mreže je prenaučenost. Grafički prikazi rezultata ukazuju na to da prvi i drugi model nisu prenaučeni jer točnost prepoznavanja (%) na test skupu *raste* s brojem epoha.

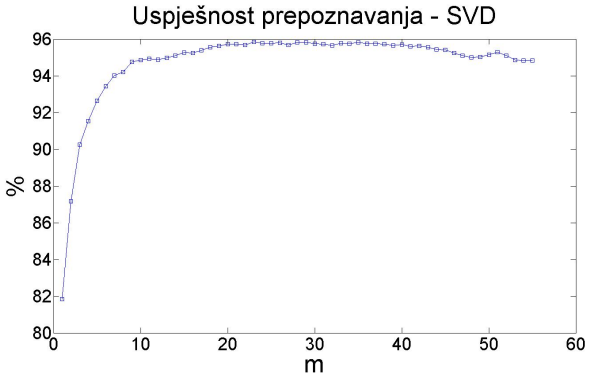
B. SVD

Algoritam je testiran na skupu od 10000 znamenaka te je dao zavidne rezultate. Uspješnost klasifikacije (broj točno klasificiranih znamenki) ovisi o broju m singularnih vektora korištenih pri računanju reziduala. Slika 6 prikazuje uspješnost klasifikacije u ovisnosti o broju vektora korištenih pri računanju reziduala. Možemo vidjeti da su najlošija prepoznavanja



Slika 5. Točnost prepoznavanja (%) drugog modela na test skupu

dobivena pri korištenju samo jednog (81.84%), odnosno dva (87.16%) singularna vektora. Za veći broj vektora uspješnost je veća. Najbolji rezultat dobiven je za $m = 23$ te iznosi 95.85%. Na Slici 7 možemo vidjeti kako se ponaša greška SVD algoritma.



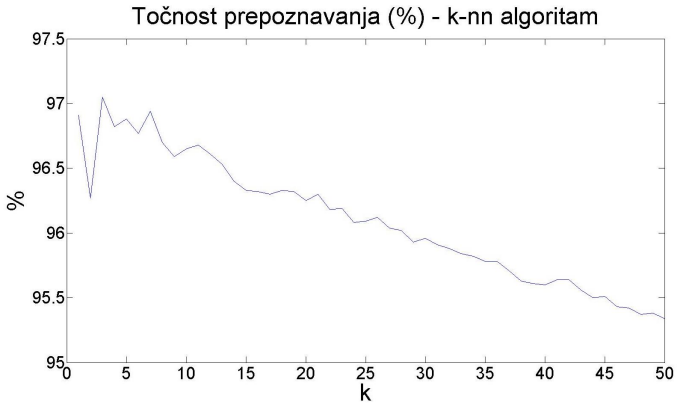
Slika 6. Uspješnost prepoznavanja algoritma SVD

Slika 7. Greška prepoznavanja algoritma SVD

C. k-nn

Provele smo k-nn na testnom skupu od 10000 znamenaka. Kao što je spomenuto ranije, test se izvršio za 50 različitih k-ova. Najbolji rezultat je dobiven za $k = 3$ te je iznosio 97.05%, a najgori rezultat je iznosio 95.34% za $k = 50$. Unatoč

izvrsnim rezultatima i za velike k-ove možemo primijetiti da nije bilo potrebe uzimati k veći od 11, jer je nakon toga prepoznavanje sve lošije. Na Slici 8 možemo vidjeti uspješnost cijelog testiranja.



Slika 8. Uspješnost prepoznavanja k-nn algoritma

V. Osvrt na druge pristupe

Postoje različiti pristupi za prepoznavanje rukom pisanih brojeva i mnogi od njih testirani su na skupu podataka koji koristimo. Originalni autori tog skupa čuvaju popis nekih metoda[7]. U nastavku navodimo neke od njih.

Autori MNIST - ove baze podataka ostvaruju stopu greške od 0.8% koristeći metodu potpornih vektora (SVM[8]). K-NN klasifikator, koji je korišten i u ovom radu, postiže stopu greške od 0.52% uz dodatno pretprocesiranje slika iz trening skupa. Hijerarhijska neuronska mreža s konfiguracijom $(784-2500-2000-1500-1000-500-10)$ ima stopu greške od 0.35%. Konvolucijske neuronske mreže[9] često se koriste u sustavima za prepoznavanje slika, a na ovom skupu podataka ostvarile su stopu greške od 0.23%. Taj je rezultat od veljače 2012. najbolji. Od ostalih neuronskih mreža razlikuju se po tome što omogućuju brže treniranje.

Ovaj kratak pregled metoda za rješavanje uvodnog problema pokazuje da je prepoznavanje rukom pisanih znamenki svedeno na gotovo ljudsku izvedbu.

VI. Mogući budući nastavak istraživanja

U budućem nastavku istraživanja namjera je pronaći konfiguraciju višeslojnog perceptrona koja najuspješnije rješava naš problem. Prvenstveno se tu misli na pronalazak vrijednosti kojima se inicijaliziraju težine u modelu i utvrđivanje najpogodnijeg broja neurona u skrivenom sloju. Dakle, glavni je cilj poboljšati način na koji neuronska mreža uči[10]. Na našem skupu podataka k-nn klasifikator postiže najnižu stopu greške. U budućem nastavku istraživanja namjera je obraditi slike iz trening skupa i utvrditi koja vrsta obrade pospješuje prepoznavanje. Na kraju, skup podataka planira se proširiti drugim rukom pisanim znakovima iz NIST - ove baze podataka *Special Database 19*[11].

VII. Zaključak

Problem raspoznavanja rukom pisanih znakova proučava se već više od trideset godina. To je sposobnost računala da interpretira rukom pisane ulaze. Skup podataka koji se koristi u ovom radu sadrži 60000 primjera za učenje i 10000 primjera za testiranje. Svaka slika iz skupa podataka reprezentira neku od znamenki: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, i pridružena joj je odgovarajuća oznaka. Upotrijebili smo tri različite metode za rješavanje problema: neuronske mreže, dekompoziciju matrice na singularne vrijednosti i algoritam k – najbližih susjeda. U prvom slučaju točnost prepoznavanja na test skupu maksimalno iznosi 90.67%. Primijetili smo da točnost prepoznavanja raste s brojem epoha učenja mreže. U drugom slučaju točnost prepoznavanje raste u odnosu na neuronsku mrežu i maksimalno iznosi 95.85%. Točnost prepoznavanja raste s brojem singularnih vektora m koji se koriste u računanju normi reziduala, no ne valja uzeti prevelik m jer bi se moglo dogoditi da se neka nepoznata znamenka prikaže pomoću stupaca dviju različitih matrica T_i što bi otežalo klasifikaciju. U trećem slučaju algoritam k – najbližih susjeda postiže točnost prepoznavanja od maksimalno 97.05% i to za $k = 3$. To je najbolji rezultat koji je postignut u našim mjerenjima. Rezultati korištenih algoritama podudaraju se s onima u literaturi, no postoje pristupi koji rješavaju ovaj problem mnogo uspješnije, toliko da je njihovo prepoznavanje svedeno na gotovo ljudsku izvedbu. Primjer su konvolucijske neuronske mreže.

LITERATURA

1) http://e.math.hr/br24/NovakEtAl
2) http://neuralnetworksanddeeplearning.com/
3) http://yann.lecun.com/exdb/mnist/
4) http://stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-neural-netw

[7] Popis metoda dostupan je ovdje.

[8] Više o SVM može se pročitati ovdje.

[9] Više o konvolucijskim neuronskim mrežama može se pročitati ovdje.

[10] Kako poboljšati način na koji neuronska mreža uči, detaljnije je opisano u poglavlju knjige koje je dostupno ovdje.

[11] *Special Database 19* dostupna je ovdje.