SL PERCEPTRON ALGORITAM
SL PERCEPTRON ALGORITHM

Stefan Tubić
Elektrotehnički fakultet, Univerzitet u Beogradu

Sadržaj – *Ovaj rad prikazuje način funkconisanja Single Layer Perceptron algoritma kao jednog elementa veštačke inteligencije i njegove podoblasti pod nazivom mašinsko učenje. Cilj rada je predstavnjanje načina kojim mašina može da se prilagodi okolini i sama uči.*

Abstract *- This paper presents the way Perceptron algorithm functions as one element of Artificial Intelligence and its subfield called Machine Learning. The objective of this paper is the representation of way that machine can adapt to the environment and learn by itself.*

1. UVOD

Mašinsko učenje je naučna disciplina čiji je cilj konstruisanje algoritama i računarskih sistema koji su sposobni da se adaptiraju na analogne nove situacije i uče na bazi iskustva. Takvi algoritmi funkcionišu tako što izgrade model na bazi ulaza i koriste ga da naprave predviđanja budućnosti, u većem stepenu nego što bi koristili eksplicitno napisane instrukcije i tačno definisan sled stvari koji moraju da odrade.

Mašnisko učenje se može shvatiti kao podoblast veštačke inteligencije, kompjuterske nauke i statistike. Daje različite metode, algoritme i načine rešavanja problema u socijalnim i senzorskim mrežama. Veoma je primenljivo u oblastima: spam-filtering, optical character recognition (OCR), search-engines, computer vision.

U daljem radu biće predstavljen način funkcionisanja Perceptron algoritma kao i njegova uloga u neuralnim mrežama. Takođe su prikazane prednosti i mane datog algoritma, analizirane performanse i predstavljena poboljšanja datog algoritma.

2. ISTORIJAT

Perceptron algortiam je izmišljen 1957. godine u Cornell vazduhoplovnoj laboratoriji od strane Frank Rosenblatta, osnovana od strane američke kancelarije za mornarička istraživanja. Perceptron je namenjen da bude mašina, više nego program iako je njegova prva implementacija bila u softveru za IBM 704.

Zatim je bio implementiran u hardveru kao „Mark 1 perceptron“. Ova mašina je bila dizajnirana za prepoznavanje slika. Imala je niz od 400 fotoćelija, nasumično povezanih na „neurone“. Težine su šifrovane potenciometrima, dok je ažuriranje težina tokom učenja vršeno električnim motorima.

3. ŠTA JE PERCEPTRON

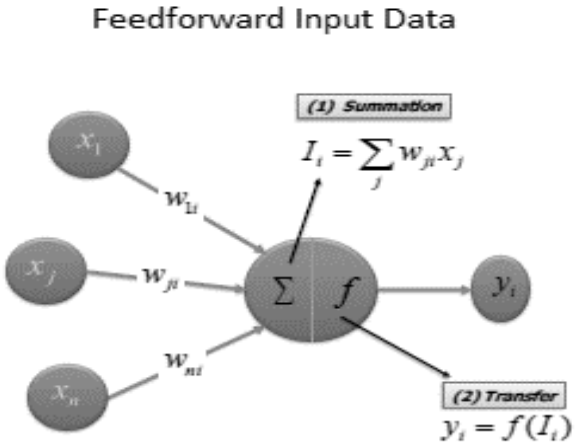
Neuron, poznat i kao nervna ćelija, jeste električno nadraživa ćelija koja procesira i šalje informacije kroz električne i hemijske signale. Ovi signali između neurona se ostvaruju preko sinapsi, specijalizovanih veza sa drugim ćelijama. Neuroni povezani jedni sa drugima čine neuralnu mrežu.

Perceptron je simulacija neurona. Predstavlja jednu ćeliju koja na svom ulazu ima nekoliko ulaznih grana gde svaka grana ima svoju težinu, i ima jednu izlaznu granu. Njegov zadatak je da izvrši klasifikaciju ulaznih podataka i vrši predikciju baziranu na predikcionoj funkciji (transfer funkciji) i ulaznim podacima kombinovanim sa težinama njihovih ulaznih grana.

4. PERCEPTRON ALGORITAM

4.1 Osvnovne akcije

Obavlja dve osnovne akcije. Sumira ulaze, a zatim suma prolazi kroz transfer funkciju čija vrednost predstavlja izlaz.



Slika 1 Propagacija podataka od ulaza do izlaza

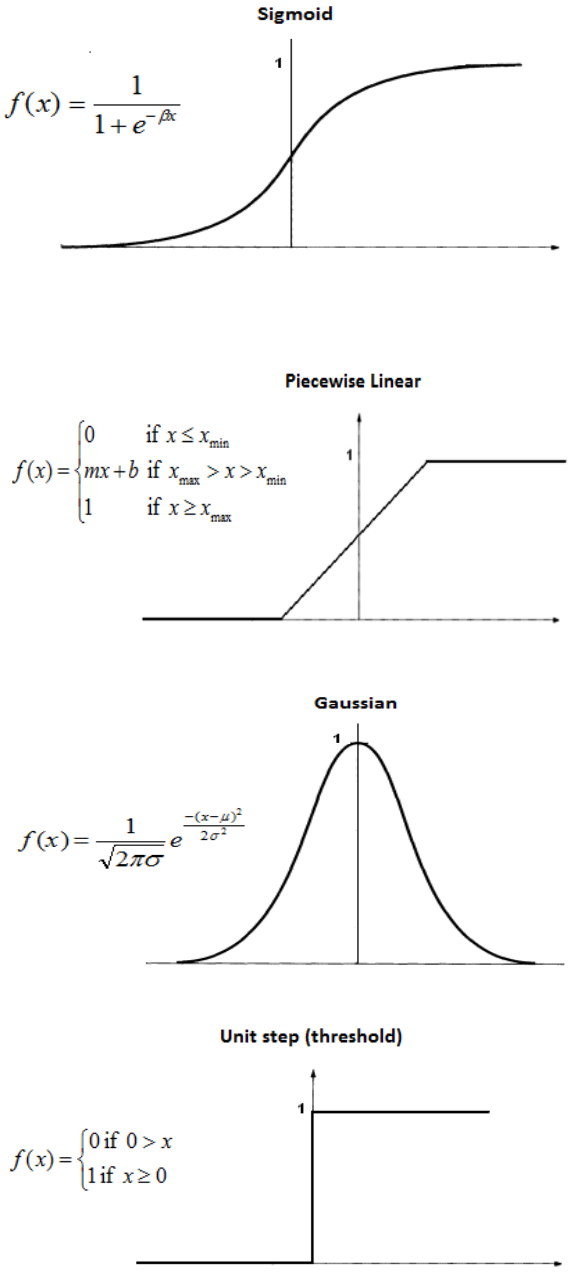
Sumiranje ulaza se vrši po sledećoj formuli:

$$S = \sum_{i=0}^{n} w_i \, x_i \qquad (1)$$

S je suma čiji su sabirci proizvodi ulazne vrednosti grane i njene težine. w_i je težina grane dok je x_i ulazna vrednost te grane. Bitno je napomenuti da se ulazne grane broje

počevši od 1, s tim što se uvodi nulti ulaz čija je ulazna vrednost uvek 1 dok se težina može menjati. Taj ulaz se naziva „bias“ i služi pri podešavanju prave klasifikacije u smislu njenog tranlsatornog kretanja po y osi. Uvodi se radi finije klasifikacije.

Postoji više transfer funkcija koje se mogu koristiti. Neke od njih su:



Slika 2 Transfer funkcije

Poslednja „Unit step“ funkcija je ona koja se najčešće koristi.

$$f(S) = \begin{cases} 0, S < 0 \\ 1, S \ge 0 \end{cases} \quad (2)$$

f(S) predstavlja izlaznu vrednost perceptrona. Ukoliko suma izračunata u prethodnoj akciji prelazi vrednost 0 na izlazu se pojavljuje 1, u suprotnom 0. Iz ovoga proizilazi da se perceptron se mora dovoljno stimulisati da bi se na izlazu pojavio jak signal.

4.2 Način rada

Perceptron se trening vektorima uči da klasifikuje podatke, tj. vektore (atribute) koji dolaze posle trening vektora, u neku od dve klase. U slučaju dva atributa koji opisuju objekat koji je potrebno klasifikovati algoritam ima zadatak da nađe pravu koja će podeliti skup trening objekata u dve grupe. Valjanost klasifikacije definišu upravo trening vektori. Ukoliko je taj skup dobro izabran prava dobijena algoritmom će u budućnosti vršiti dobru klasifikaciju novih podataka.

Algoritam obavlja dve stvari u više iteracija. Prvo vrši propagaciju ulaznih podataka do izlaza. Tada se vrše dve, već opisane, akcije, a to su sumiranje ulaza i prolazak kroz transfer funkciju. Izlaz se u velikom broju slučajeva ne poklapa sa očekivanim tako da se javlja greška. Ukoliko greška postoji ona propagira od izlaza do ulaza (propagacija unazad) kada se redukuju težine ulaznih grana. Ispod se nalazi pseudo kod algoritma:

```
Iteration = 0;
Do{
  Iteration++;
  For(p=0 to NumOfInstances){
    S = CalcSum(weights, inputs);
    Output = F(S);
    LocalError = Output[p] – Output;
    UpdateWeights(LearningRate, LocalError, Weights);
    GlobalError += LocalError^2;
  }
}
While(GlobalError != 0 and Iteration <= MaxIteration)
```

Na početku se broj iteracija postavi na 0. Sve dok globalna greška nije 0 i broj iteracija ne pređe maksimalni broj iteracija radi se sledeće. Pozivom funkcije *CalcSum(weights, inputs)* računa se izlazna suma. Zatim se pozivom *F(S)* dobija izlazna vrednost koja se upoređuje sa očekivanim izlazom i nastaje lokalna greška. Ažuriraju se vrednosti težina ulaznih grana pozivom *UpdateWeights*. Ažuriranje se obavlja po formuli:

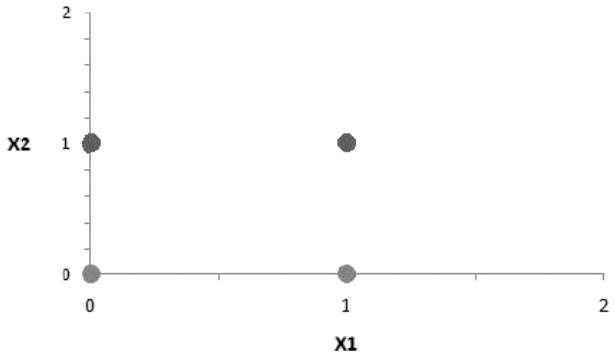
$$w_i = w_i + \Delta w_i \quad (3)$$
$$\Delta w_i = \eta * d * x_i$$

Δw_i je promena težine grane, η je tempo učenja, d je lokalna greška, dok je x_i ulazna vrednost grane. Na kraju se na globalnu grešku doda vrednost lokalne greške.

S obzirom da algoritam može raditi dosta dugo jer učenje neurona može biti spor proces postoji vrednost *MaxIteration* koja služi da ograniči broj iteracija u slučaju da se algoritam dugo izvršava.

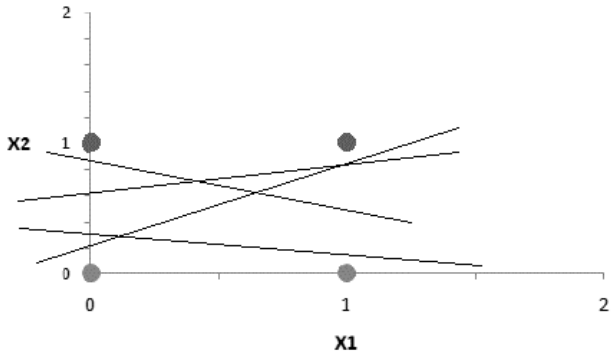
5. DEMONSTRACIJA RADA

U daljem tekstu biće prikazano testiranje i rad ovog algoritma.



Slika 3 Ulazni trening vektori

Testiranje je urađeno nad četiri objekta gde su gornja dva pripadnici klase A dok su druga dva pripadnici klase B. Zadatak programa je da pronađe pravu koja će podeliti ravan na dva dela i tako vršiti predikciju i klasifikovati podatke u budućnosti.



Slika 4 Perceptron može imati više rešenja

Moguće je naći više rešenja, u ovom slučaju beskonačno mnogo što zavisi od početnih težina ulaznih grana, transfer funkcije i tempa učenja.

Početne težine grana korišćene u ovom testiranju su:
w0: 0.37 w1: 0.11 w2: 0.23, i one se nasumično biraju. Trening vektori (ulazi) i očekivani izlazi su sledeći:
x1: 0.0 x2: 0.0 output: 0
x1: 0.0 x2: 1.0 output: 1
x1: 1.0 x2: 0.0 output: 0
x1: 1.0 x2: 1.0 output: 1

Dalje će biti prikazane poslednje dve iteracije while petlje algoritma.

Pretposlednja iteracija:

```
Sum = 0.17
output = 1 localError = -1.0 globalError = 1.0
w0: 0.07 w1: 0.009999999999999995 w2: 0.23

Sum = 0.30000000000000004
output = 1 localError = 0.0 globalError = 1.0
w0: 0.07 w1: 0.009999999999999995 w2: 0.23

Sum = 0.08
output = 1 localError = -1.0 globalError = 2.0
w0: -0.03 w1:-0.09000000000000001 w2: 0.23

Sum = 0.11000000000000001
output = 1 localError = 0.0 globalError = 2.0
w0: -0.03 w1:-0.09000000000000001 w2: 0.23

Iteration 2 : RMSE = 0.7071067811865476
```

Sum i *output* su izračunata suma i izlaz za jedan test vektor. Prikazane su lokalna i globalna greška kao i nove vrednosti težina dobijenih ažuriranjem. RMSE (Root Mean Square Error) je vrednost koja se obično koristi da iskaže grešku dobijenu jednom iteracijom ovog algoritma. RMSE se računa po formuli:

$$RMSE = \sqrt{\frac{\sum_{i=0}^{n}(p_i - a_i)^2}{n}}$$

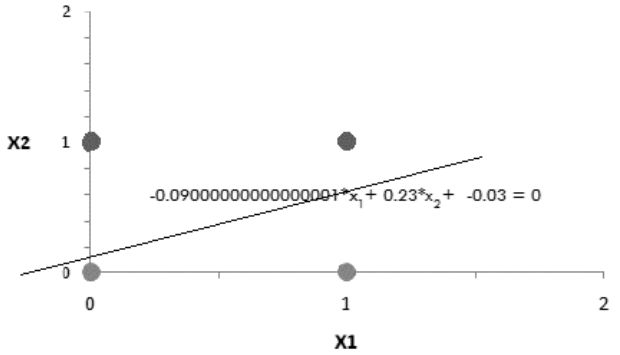
p_i je trenutna vrednost izlaza, dok je a_i očekivana vrednost izlaza. n je broj trening vektora. RMSE nije jednak nuli tako da se nastavlja sa zvršavanjem i ide u narednu iteraciju.

Poslednja iteracija:

```
Sum = -0.03
output = 0 localError = 0.0 globalError = 0.0
w0: -0.03 w1:-0.09000000000000001 w2: 0.23

Sum = 0.2
output = 1 localError = 0.0 globalError = 0.0
w0: -0.03 w1:-0.09000000000000001 w2: 0.23

Sum = -0.12000000000000001
output = 0 localError = 0.0 globalError = 0.0
w0: -0.03 w1:-0.09000000000000001 w2: 0.23

Sum = 0.11000000000000001
output = 1 localError = 0.0 globalError = 0.0
w0: -0.03 w1:-0.09000000000000001 w2: 0.23

Iteration 3 : RMSE = 0.0
```

RMSE je sada 0 tako da je algoritam završen i nađene su krajnje vrednosti težina ulaznih grana a to su:
w0: -0.03 w1:-0.09000000000000001 w2: 0.23

Na slici ispod je nacrtana linija koja je dobijena perceptron algoritmom i koja će vršiti klasfikaciju podataka.



Slika 5 Rezultat rada Perceptron algoritma

6. PREDNOSTI I MANE

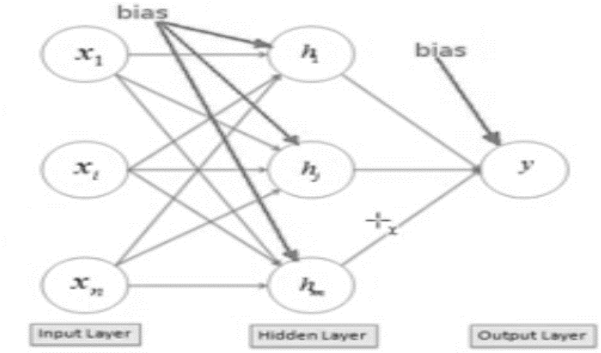
Prednost ovog algoritma je ta što on može vršiti dosta dobru klasifikaciju ulaznih podataka. Koliko će klasifikacija biti dobra zavisi od toga koliko su ulazni trening vektori valjani.

Mana je ta što perceptron sporo uči. Takođe je problem onda kada je nemoguće pronaći pravu liniju koja će razdvojiti sve trening vektore u ravni (ravan u prostoru itd.).

7. POBOLJŠANJA

Algoritam bi se mogao unaprediti uvođenjem rešenja za slučaj kada pravu odlučivanja nije moguće naći, kao i pametnijeg načina za određivanje ulaznih trening vektora.

Kada pravu odlučivanja nije moguće naći, odnosno kada postoji potreba za rešavanjem nelinearnog problema. koristi se MultiLayer Perceptron algoritam koji obično koristi sigmoid kao transfer funkciju i koji se sastoji od više perceptrona povezanih u veštačku neuralnu mrežu gde pored ulaznog i izlaznog sloja postoji i „hidden“ kao međusloj radi dodatne kalkulacije što je prikazano na narednoj slici.



Slika 6 MultiLayer Perceptron

8. ZAKLJUČAK

U prezentovanom radu je pokazano kako Perceptron algoritam funkcioniše i kako može vrlo efikasno da izvrši klasifikaciju ulaznih podataka. Istaknuto je da je dosta bitno stvoriti dobar skup trening vektora i njima učiti neuron.

Priložen algoritam je dosta jednostavan za implementaciju i pokazuje dobre performanse.

U radu je prikazan i proces učenja jednog perceptrona kroz iteracije što je rezultat autorovog rada i njegove implementacije algoritma.

LITERATURA

[1] Jia Li, “Perceptron Learning Algorithm”, Department of statistics, The Pennsylvania State University, United States

[2] Wikipedia,“Perceptron”,http://en.wikipedia.org/wiki/Perceptron, Decembar 2014.

[3] Umesh Vazirani, “The Perceptron Algorithm”, University of California, Berkeley

[4] Wikipedia,“MultiLayer Perceptron”, http://en.wikipedia.org/wiki/ Multilayer_perceptron , Decembar 2014.