Politechnika Częstochowska

Podstawy Sztucznej Inteligencji

Laboratorium 3
Perceptron wielowarstwowy

Autor:
Piotr Filek
101311
I grupa

Prowadzący:
dr inż. Artur Starczewski

11 grudnia 2013

1 Cel laboratorium

Celem laboratorium było wykonanie *Perceptronu wielowarstwowego* w programie *Scilab*.

2 Przebieg laboratorium

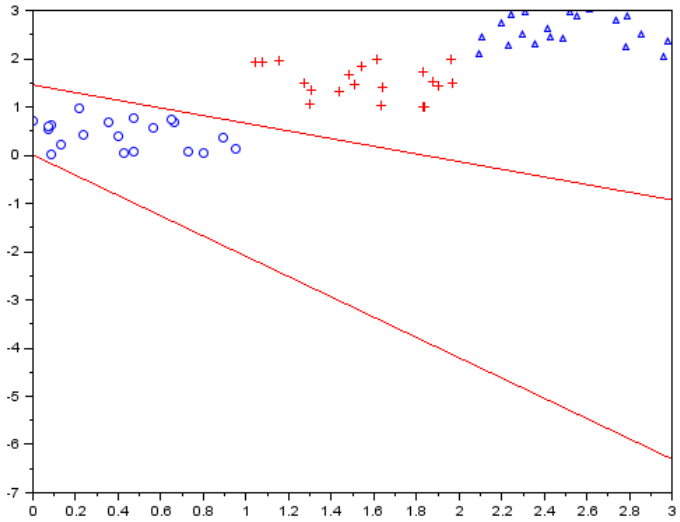
2.1 Kod źródłowy programu

```
clear;
clf();
// przygotowanie elementow
X = [rand(2, 20), rand(2, 20) + 1, rand(2,20)+2];
// wartosci elementow
D1 = [ones(1, 20), ones(1, 20), zeros(1,20)];
D2 = [zeros(1, 20), ones(1, 20), ones(1,20)];
D = [D1; D2]
// wyswietlenie elementow
plot(X(1, 1:20), X(2, 1:20), 'po');
plot(X(1, 20+1:40), X(2, 20+1:40), 'r+');
plot(X(1,40+1:60),X(2,40+1:60),'b^');
// generwowanie wag
w = rand(1, 3)*(0.01);
w2 = rand(1, 3)*(0.01);
// wyswietlenie prostej 1 - przed nauczeniem
k = 0;
for i = 0:0.01:3
  k = k + 1;
  Xw(k) = i;
  Yw(k) = -(w(1) * i - w(3)) / w(2);
end;
plot2d(Xw, Yw, style=[color('red')]);
// wyswietlenie prostej 2 - przed nauczeniem
k = 0;
for i = 0:0.01:3
  k = k + 1;
  Xw(k) = i;
  Yw(k) = -(w2(1) * i - w2(3)) / w2(2);
end;
plot2d(Xw, Yw, style=[color('red')]);
alfa = 0.2; // wspolczynnik alfa
blad = 1; //zmienne poczatkowe
net = zeros(2, 60);
y = zeros(2, 60);
// proces uczenia;
while(blad == 1)
  blad = 0; // zerowanie bledu
  for i = 1:60 // przebieg uczenia
    net(1,i) = X(1, i) * w(1) + X(2, i) * w(2) + (-1) * w(3);
    net(2,i) = X(1, i) * w2(1) + X(2, i)* w2(2) + (-1) * w2(3);
    // zastosowanie funkcji unipolarnej
    if net(1, i) >= 0 then
      y(1,i) = 1;
      if net(2,i) >= 0 then
```

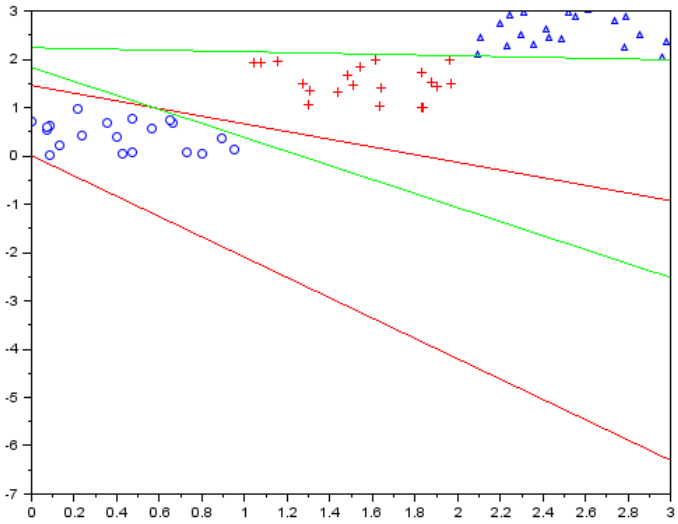
```
  y(2,i) = 1;
      else
  y(2,i) = 0;
      end
    else
      y(1,i) = 0;
      if net(2,i)>= 0 then
  y(2,i) = 1;
      else
  y(2,i) = 0;
      end
    end
    // sprawdzenie
    if D(1,i) <> y(1,i) then
      blad = 1;
    end
    if D(2,i) <> y(2,i) then
      blad = 1;
    end
    // korekta wag;
    w(1) = w(1) + alfa * (D(1,i) - y(1,i)) * X(1, i);
    w(2) = w(2) + alfa * (D(1,i) - y(1,i)) * X(2, i);
    w(3) = w(3) + alfa * (D(1,i) - y(1,i)) * -1;
    // korekta wag 2
    w2(1) = w2(1) + alfa * (D(2,i) - y(2,i)) * X(1, i);
    w2(2) = w2(2) + alfa * (D(2,i) - y(2,i)) * X(2, i);
    w2(3) = w2(3) + alfa * (D(2,i) - y(2,i)) * -1;
  end
end
sleep(2000)
// wyswietlenie prostej 1 - po nauczeniu
k = 0;
for i = 0:0.01:3
  k = k + 1;
  Xw(k) = i;
  Yw(k) = -(w(1) * i - w(3)) / w(2);

end;
plot2d(Xw, Yw, style=[color('green')]);
// wyswietlenie prostej 2 - po nauczeniu
k = 0;
for i = 0:0.01:3
  k = k + 1;
  Xw(k) = i;
  Yw(k) = -(w2(1) * i - w2(3)) / w2(2);
end;
plot2d(Xw, Yw, style=[color('green')]);
```

PerceptronWielowarstwowy.sce

2.2 Wynik działania programu



Rysunek 1: Wykres przedstawiający linie decyzyjne przed nauczeniem perceptrona.



Rysunek 2: Wykres przedstawiający linie decyzyjne po nauczeniu perceptrona. (czerwone przed, zielone po)

3 Wnioski

Na powyższych screenach można zobaczyć działanie perceptrona wielowarstwowego. W sposób równoległy (wiele warstw) wyodrębnia on różne klasy. Rozwinięcie programu perceptrona prostego, który by dzielił klasy w dwóch etapach zamiast równolegle, nie byłoby poprawnym zaimplementowaniem perceptrona wielowarstwowego. Na drugim screenshocie możemy zobaczyć zarówno nienauczony perceptron (czerwone linie), jak i już poprawnie działający, nauczony perceptron (zielone linie).