Elementy Sztucznej Inteligencji
Laboratorium #1

Podstawy sztucznych sieci neuronowych. Perceptron. Zadania klasyfikacji

Grupa: _____ Data: _____

Nazwisko i imię: _____

Nazwisko i imię: _____

Nazwisko i imię: _____

Zadanie #1: Badanie prostego neuronu

1. Uruchom program demonstracyjny w Matlabie ilustrujący działanie neuronu z dwoma wejściami. W tym celu wykonaj następujące czynności:
   a. Uruchom program Matlab
   b. Wpisz polecenie **nnd**
   c. Naciśnij przycisk **Table of contents**
   d. Rozwiń listę **Chapter 2 demos**
   e. Wybierz opcję **Two-input neuron**

2. Zbadaj zachowanie się neuronu (sygnał n na wyjściu elementu sumacyjnego oraz sygnał a na wyjściu neuronu) dla następujących parametrów:
   a. p(1)=1, p(2)=1, w(1,1)=1, w(1,2)=1, b=0, F=Hardlim
      n= _____, a= _____
   b. p(1)=1, p(2)=1, w(1,1)= −2, w(1,2)=1, b=0, F=Hardlim
      n= _____, a= _____
   c. p(1)=1, p(2)=1, w(1,1)=1, w(1,2)=1, b=0, F=Purelin
      n= _____, a= _____
   d. p(1)=1, p(2)=1, w(1,1)=1, w(1,2)=1, b=0, F=Tansig
      n= _____, a= _____

   Przebadaj zachowanie się wyjścia neuronu dla różnych funkcji aktywacji przy płynnej zmianie wartości parametrów. Zwróć uwagę na różnice działania neuronu z dyskretnym i ciągłym sygnałem wyjściowym.

   Dla jakich parametrów możliwe jest uzyskanie sygnału wyjściowego a=6 ?
   e. p(1)= ____, p(2)= ____, w(1,1)= ____, w(1,2)= ____, b= ____, F= _____

Zadanie #2: Klasyfikacja z dwuwejściowym perceptronem

1. Uruchom program demonstracyjny **nnd4db**. W tym celu albo:
   a. W menu programów demo wybierz opcję **Decision boundaries** w **Chapter 4**
   b. Albo podaj polecenie **nnd4db** w oknie Matlab Commands

2. Dla domyślnego ustawienia czterech punktów (w tym jednego czarnego) dokonaj takiego przemieszczenia granicy decyzyjnej perceptronu, aby uzyskać separację (klasyfikację) punktów białych od punktu czarnego (przeczytaj wskazówki z prawej strony okna). Podaj wagi i bias dla rozwiązania:
   w(1,1)= _____, w(1,2)= _____, b=_____
3. Ustaw nową konfigurację punktów według podanych niżej współrzędnych x i y:
   punkty białe: (−1.5,1.5), (−1, −1), (2,0)
   punkt czarny: (−0.5,0)
4. Spróbuj dokonać separacji dla nowej konfiguracji punktów. Opisz i wyjaśnij rezultat:
   _____
   _____
   _____
   _____
   _____

Zadanie #3: Uczenie perceptronu do zadań klasyfikacji

1. Uruchom program demonstracyjny **Perceptron rule** (**nnd4pr**)
2. Ustaw punkty według następujących współrzędnych (x i y):
   punkty białe: (0, 1), (1, −1)
   punkt czarny: (0, 2)
3. Wykonaj uczenie klasyfikatora według następujących kroków:
   a. Wybierz bias=0 (**no bias**)
   b. Ustal losową wartość wag (**random**)
   c. Zastosuj regułę uczenia perceptronu (**learn**)
   d. Wykonaj kilka sekwencji uczenia wciskając kilkakrotnie przycisk **train** i zaobserwuj rezultaty. Podaj wnioski:
      _____
      _____
      _____
4. Wykonaj uczenie tak samo jak w p.3, ale z włączoną opcją **bias**. Podaj parametry perceptronu po nauczeniu go klasyfikacji:

   w(1,1)= _____, w(1,2)= _____, b= _____

Zadanie #4: Projekt programu symulacji neuronowego klasyfikatora typu perceptron

1. Napisz nowy program symulacji sieci neuronowej według podanego niżej wzoru:
   a. Wybierz opcję **File** a potem **New**
   b. W oknie edytora Matlab wpisz tekst następującego programu:

```
p = [2 1 -2 -1;2 -2 2 1];
t = [0 1 0 1];
net = newlin( [-2 2; -2 2],1);
net.trainParam.goal= 0.1;
[net, tr] = train(net,p,t);
```

   c. Zapisz utworzony program jako nowy plik m-file
      i. Umieść plik w katalogu MATLABR11/WORK
      ii. Nadaj mu własną nazwę (np. my_file1.m)

d. Przeanalizuj dokładnie podany program i spróbuj szczegółowo zbadać znaczenie kolejnych linii. Wskazówka: posłuż się systemem Help i określ znaczenie poszczególnych poleceń.
e. Odpowiedz na pytania:
    i. Co oznacza macierz p _____
    ii. Co oznacza macierz t _____
    iii. Ile neuronów zawiera sieć _____
    iv. Jakiego rodzaju jest funkcja aktywacji _____
    v. Ile wejść ma sieć _____
    vi. Ile wyjść ma sieć _____
    vii. Jaką funkcję spełnia sieć _____

f. Uruchom program wybierając opcje **Tools** a następnie **Run.** Przeanalizuj obraz uzyskany w oknie graficznym oraz w oknie Matlab Command. W której iteracji sieć osiąga założone kryterium błędu uczenia? _____

2. Zmień kryterium błędu uczenia (zmniejsz wartość graniczną) i zaobserwuj co się dzieje. Dlaczego po znacznym zmniejszeniu błędu granicznego sieć nie osiąga zadanego celu? Spróbuj to zmienić. Wskazówka: wykorzystaj parametr net.trainParam.epochs. Jaka jest wartość domyślna tego parametru?

3. Spróbuj wykonać tak dużo modyfikacji i ulepszeń tego programu, ile potrafisz. W szczególności rozważ następujące propozycje:

    a. Spróbuj wstawić do programu fragment, który przedstawi w postaci graficznej dane wejściowe. Wskazówka: wykorzystaj funkcję **plot.**
    Punkty, dla których t=0 i punkty dla których t=1 przedstaw różnymi kolorami. Wskazówka: wykreśl kolejne punkty w pętli **for**, do określenia koloru użyj instrukcji **if**.
    b. Spróbuj zmienić dane wejściowe na inne. W szczególności spróbuj zmienić rozmiar wektorów.
    c. Przy pomocy Help przeanalizuj zastosowaną metodę uczenia sieci. Zastanów się nad możliwością zmiany parametrów uczenia, w tym zwłaszcza współczynnika szybkości uczenia **lr.**
    d. Zastanów się nad możliwością graficznej prezentacji błędu uczenia sieci w trakcie sesji uczenia. Funkcja **train** wykonuje to zadanie automatycznie. Spróbuj zrobić to samo używając funkcji **plot.**

Zadanie #5: Sieć neuronów liniowych z opóźnieniami (adaptacyjna).

Uruchomić plik wsadowy ***demolin8.m.*** Zapoznać się z opisem sieci w pliku pomocy.

1. Zmieniając szybkość uczenia zaobserwuj czas osiągania celu. Zapisz wnioski.

2. Zmień postacie sygnałów (zapisz w sprawozdaniu). Wykonaj polecenie punktu 1.

Zadanie #6: Dla ambitnych: Rozbudować program z zadania 4 tak, aby mógł klasyfikować obiekty nie na dwie klasy, ale na większą liczbę klas.

Wskazówki:

1. Trzeba zmodyfikować strukturę sieci i wyposażyć ją w większą liczbę wyjść.
2. Dane wejściowe powinny uwzględniać zróżnicowanie populacji na większą liczbę klas

```
p = [2 1 -2 -1;2 -2 2 1];
t = [0 1 0 1];
hold on;
for i=1:length(t),
   if t(i)==0
      plot(p(1,i),p(2,i),'go');
   else
      plot(p(1,i),p(2,i),'ro');
   end
end;
pause;
net = newlin([-2 2; -2 2],1,[0],0.001);
net.trainParam.epochs=200;
[net, tr] = train(net,p,t);
plot(tr.epoch,tr.perf);
```