Marcelina Bałamut, IO, gr. 1

Temat ćwiczenia:

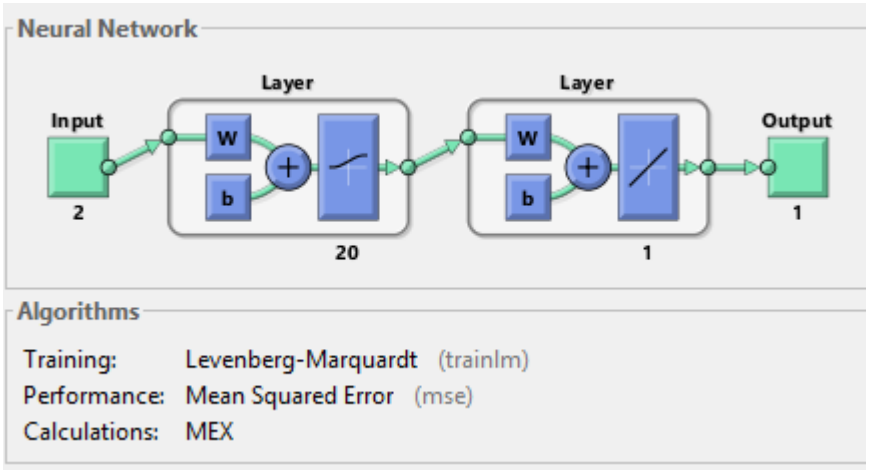
Budowa i działanie perceptronu.

Cel ćwiczenia:

Celem ćwiczenia jest poznanie budowy i działania perceptronu poprzez implementację oraz uczenie perceptronu realizującego wybraną funkcję logiczną dwóch zmiennych.

1.Opis budowy perceptronu oraz wykorzystanego algorytmu uczenia:

Działanie perceptronu opiera się na klasyfikowaniu danych na wejściu i ustawianiu do tego wartości wyjścia. Tworzony perceptron składa się z dwóch wejść, którymi jest tablica. Natomiast na wyjściu znajduje się wynik operacji funkcji logicznej NOR. Tak przygotowany zestaw uczący będzie służył do trenowania perceptronu poprzez wykorzystanie parametru trainlm w strukturze NET, zawierającej parametry sieci wielowarstwowej jednokierunkowej.



Funkcja trainlm opiera się na algorytmieLevenberga-Marquardta. Jest to bardzo popularna metoda uczenia sieci neuronowych słynąca z dużej szybkości działania i dużego zapotrzebowania na pamięć. Nadaje się tylko dla sieci z pojedynczym neuronem wyjściowym i funkcją błędu w postaci sumy kwadratów odchyleń. Algorytm Levenberga-Marquardta jest algorytmem iteracyjnym i wykorzystuje metodę znajdowania miejsc zerowych funkcji, czyli

metodę Newtona. W pobliżu minimum posiada cechy metody Newtona natomiast wraz z oddalaniem się punktu od poszukiwanego minimum, sposób jej działania zbliża się do metody najszybszego spadku.

Wzór opisujący algorytm jest następujący:

$$\mathbf{w}^{t} = \mathbf{w}^{t-1} - (\mathbf{H} + \mu\mathbf{I})^{-1}\nabla E(\mathbf{w}^{t-1})$$

Ważnym elementem algorytmu Levenberga-Marquardta jest fakt, że do hesjanu dodawana jest macierz jednostkowa, pomnożona przez dodatni współczynnik, którego wartość modyfikowana jest w trakcie uczenia. Podczas zwiększania μ wzrasta znaczenie kierunku poprawy wyznaczonego w oparciu o gradient funkcji błędu. Po każdym udanym kroku μzostaje zmniejszony, a jego wzrost następuje, gdy wstępny krok zwiększy wydajność.

2.Zestawienie otrzymanych wyników dla funkcji logicznej NAND z zestawem uczącym:

Pierwsza zmienna: 1 1 0 0

Druga zmienna: 1 0 0 1

Wynik operacji NAND: 0 1 1 1

```
przed_treningiem =

   -1.4382   -0.2364   -0.1030   -1.4259

po_treningu =

    0.0000    1.0000    1.0000    1.0000
```

Jak widać powyżej, dla przygotowanego zestawu uczącego, wyniki przed uczeniem są różnorakie, niezgodne z prawdziwym działaniem funkcji NAND. Wynika to z tego, że nie zostało przeprowadzone jeszcze uczenie. Niżej zostały wyświetlone wyniki po treningu i jak widać na załączonym screenie są one prawidłowe i pokrywają się z wynikami z zestawu uczącego.

3.Analiza i dyskusja błędów uczenia i testowania opracowanego perceptronu w zależności od wartości współczynnika uczenia oraz liczby danych uczących.

Z testu działania perceptronu wynika, że zmiana wartości poszczególnych czynników znacząco zmienia proces uczenia. Błąd uczenia zasadniczo zależy od wartości współczynnika uczenia, co pokazały testy po jego zmianie na wartość odpowiednio mniejszą i większą. Na przykład, dla współczynnika wynoszącego 0.2, czas uczenia był inny niż dla 0.1, co za tym idzie liczba wywołań potrzebna do pożądanego wyniku programu była inna. Ważnym aspektem jest również zależność między liczbą danych uczących, a współczynnikiem. Przy takiej samej wartości błędu oraz współczynnika czas uczenia jest różny dla większego zbioru danych i dla mniejszego. Testując na zbiorze cztero- i ośmioliczbowym czasy były znacząco różne. Dla zboru ośmioliczbowego udało się osiągnąć wynik już po dwóch wywołaniach, czego nie można uzyskać na zbiorze mniejszym.

Wywołanie pierwsze:

```
przed_treningiem =

   -2.1163   -0.6696    0.3714   -1.5382   -2.1163   -0.6696    0.3714   -1.5382

po_treningu =

    0.0001    0.9965    1.0005    0.9989    0.0001    0.9965    1.0005    0.9989
```

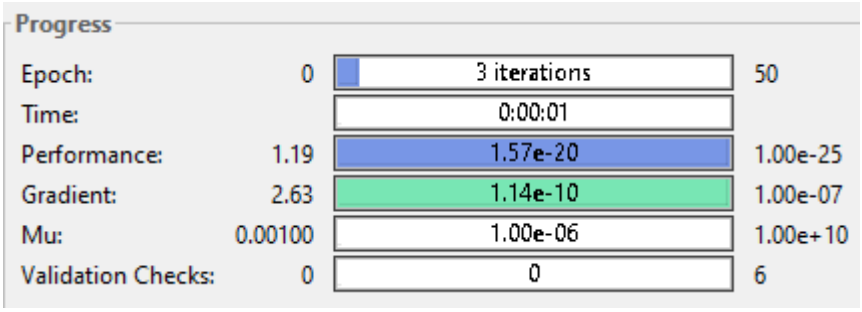
Wywołanie drugie:

```
przed_treningiem =

    0.1286   -0.3614   -0.8257   -1.5037    0.1286   -0.3614   -0.8257   -1.5037

po_treningu =

    0.0000    1.0000    1.0000    1.0000    0.0000    1.0000    1.0000    1.0000

>>
```

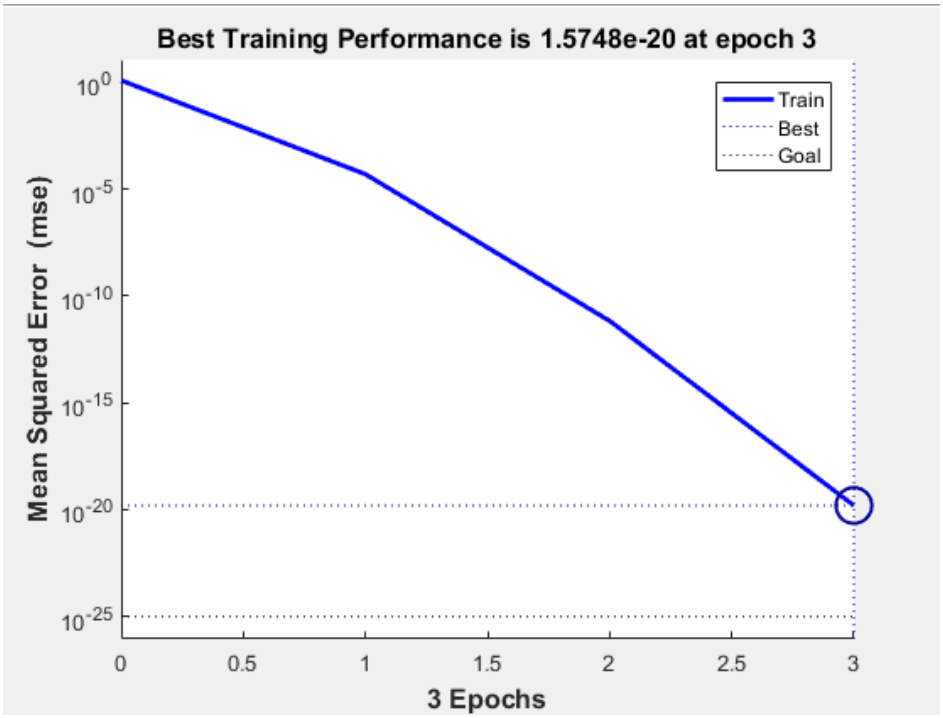
Komenda odpowiadająca za wartość współczynnika.

```
net.trainparam.lr=0.2;
```

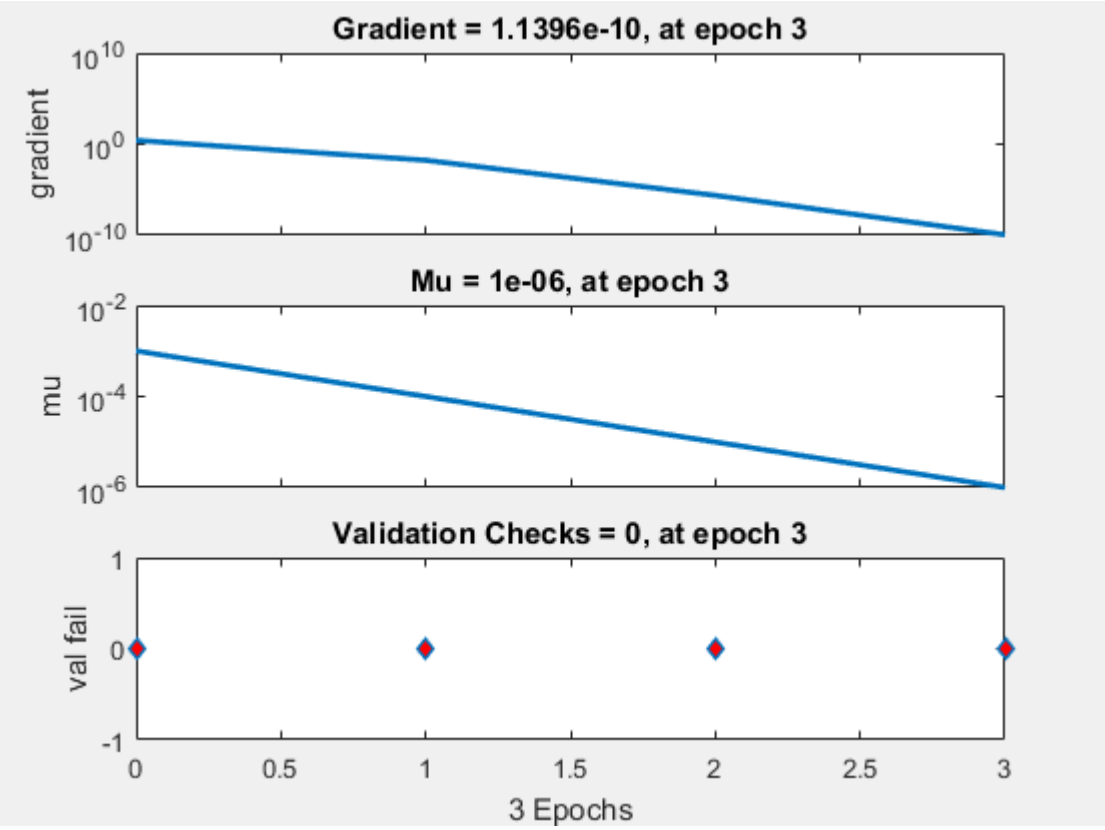
Parametry procesu uczenia przedstawiają się następująco:



Wykres wydajności, gdzie najlepsza wynosi: 6.2006e-23



Pozostałe wykresy:



4.Wnioski:

Po przeprowadzonych testach i analizie można wywnioskować, iż większa wartość współczynnika uczenia skraca czas uczenia perceptronu, co potwierdza fakt, że dla większego współczynnika ilość potrzebnych wywołań trenowania była mniejsza i prawidłowy wynik został osiągnięty wcześniej. Jeśli współczynnik ma wysoką wartość, perceptron uczy się szybciej, jednak zbyt wysoka wartość prowadzi do sporych błędów. Ważnym wnioskiem jest również to, że cały proces uczenia zależy znacząco od liczby danych uczących. Im więcej danych, tym proces uczenia jest krótszy. Podczas kolejnego uruchamiania procesu uczenia, wartości coraz bardziej zbliżają się do pożądanych, jednak jeśli wynik jest zadowalający, należy zakończyć proces, ponieważ może dojść do przetrenowania i kolejne wyniki będą odbiegać od prawidłowych.

5.Listing kodu programu

```
x(1 ,:)=[1 ,1, 0, 0 ];
x(2 ,:)=[1 ,0, 0, 1];

y=[0, 1 , 1, 1];
net=newff(minmax(x),[20 1], {'logsig', 'purelin', 'trainlm'});
przed_treningiem=sim(net,x);

net.trainparam.epochs=50;
net.trainparam.goal=0.00001;
net.trainparam.lr=0.1;
net=train(net,x,y);
[net, tr]= train(net, x, y);
po_treningu=sim(net,x);
```