ZACHODNIOPOMORSKI UNIWERSYTET
TECHNOLOGICZNY W SZCZECINIE
INSTYTUT TECHNOLOGII MECHANICZNEJ

Metody Sztucznej Inteligencji

Sztuczne Sieci Neuronowe

Wstęp

Sieci neuronowe są sztucznymi strukturami, których budowa i działanie zostały zaprojektowane w sposób modelujący działanie naturalnego układu nerwowego, w szczególności mózgu. Cieszą się bardzo dużym zainteresowaniem. Jednym z głównych czynników mających na to wpływ jest możliwość stosowania ich w bardzo wielu dziedzinach życia do rozwiązywania problemów, gdzie użycie innych metod jest trudne lub niemożliwe. Znakomicie sprawdzają się w problemach klasyfikacji, predykcji, związanych ze sterowaniem, analizą danych. Są często wykorzystywane w geologii, fizyce, ekonomii, dla celów wojskowych oraz medycznych. Na tak wielką popularność wpływają podstawowe cechy oferowane przez sieci neuronowe. Pozwalają one rozwiązywać problemy dla opisów których nie ma modeli matematycznych. Przygotowanie sieci do pracy jest dwuetapowe. Pierwszym jest proces uczenia, w którym sieć uczy się na podstawie danych empirycznych jak reagować na zadany bodziec. Gdy sieć zostanie wytrenowana można przejść do etapu pracy właściwej (drugi etap), podając na jej wejścia dowolne sygnały.

Zastosowanie

- ***Przykład 1 – Dopasowywanie funkcji***

Tworzymy wektory uczące A oraz B:

A=-5:0.1:5;

B=tan(A);

plot(A,B,'-.');

(średnik postawiony na końcu wpisywanego polecenia powoduje, że wartość tego wyrażenia nie zostanie wyświetlona)

Tworzymy dwuwarstwową sieć neuronową o nazwie neu składającą się z pięciu neuronów w warstwie wejściowej oraz jednego neuronu w warstwie wyjściowej:

neu=newff([-5 5],[5 1],{'tansig' 'purelin'});

(tansig, logsig oraz purelin są to najczęściej stosowane funkcje przejścia w sieciach z propagacją wsteczną. Jeśli jako funkcję przejścia w ostatniej warstwie użyjemy tansig lub logsig, które są funkcjami sigmoidalnymi, to dane wyjściowe będą ograniczone do niewielkiego zakresu. Jeśli użyjemy funkcji purelin, która jest funkcją liniową, to dane wyjściowe mogą przyjąć dowolną wartość.)

Nadajemy wagom sieci przypadkowe wartości:

neu=init(neu);

Sprawdzamy odpowiedź sieci niewytrenowanej na wektor wejściowy. Wykres opisujemy „5 1 – sieć niewytrenowana":

C=sim(neu,A);

figure

plot(A,B,'-.',A,C,'-o');

title('5 1 - siec niewytrenowana')

Określamy maksymalną liczbę epok:

neu.trainParam.epochs=5;

Trenujemy sieć:

neu=train(neu,A,B);

Sprawdzamy odpowiedź wytrenowanej sieci na wektor wejściowy na wykresie. Wykres opisujemy „5 1 – 5 epok":

C=sim(neu,A);

figure

plot(A,B,'-.',A,C,'-o');

title('5 1 - 5 epok')

Oceniamy dokładność dopasowania, nie zamykamy wykresu!

Ponownie nadajemy wagom sieci przypadkowe wartości, a następnie zwiększamy maksymalną liczbę epok do 25:

neu=init(neu);

neu.trainParam.epochs=25;

Ponownie trenujemy sieć, a następnie sprawdzamy jej odpowiedź na wektor wejściowy. Na podstawie wykresu oceniamy dokładność dopasowania.

neu=train(neu,A,B);

C=sim(neu,A);

figure

plot(A,B,'-.',A,C,'-o');

title('5 1 - 25 epok')

Powtarzamy procedurę dla 50, 100, 250 oraz 500 epok, wyciągamy początkowe wnioski. Czyścimy pamięć ze wszystkich danych:

clear

Tworzymy identyczne jak poprzednio wektory uczące A oraz B:

A=-5:0.1:5;

B=tan(A);

plot(A,B,'-.');

Tworzymy dwuwarstwową sieć neuronową o nazwie neo składającą się z 20 neuronów w warstwie wejściowej oraz jednego neuronu w warstwie wyjściowej:

neo=newff([-5 5],[20 1],{'tansig' 'purelin'});

Nadajemy wagom sieci przypadkowe wartości:

neo=init(neo);

Sprawdzamy odpowiedź sieci niewytrenowanej na wektor wejściowy:

C=sim(neo,A);

figure

plot(A,B,'-.',A,C,'-o');

title('20 1 - siec niewytrenowana')

Określamy maksymalną liczbę epok:

neo.trainParam.epochs=5;

Trenujemy sieć:

neo=train(neo,A,B);

Sprawdzamy odpowiedź wytrenowanej sieci na wektor wejściowy na wykresie:

C=sim(neo,A);

figure

plot(A,B,'-.',A,C,'-o');

title('20 1 - 5 epok')

Oceniamy dokładność dopasowania, nie zamykamy wykresu!

Ponownie nadajemy wagom sieci przypadkowe wartości, a następnie zwiększamy maksymalną liczbę epok do 25:

neo=init(neo);

neo.trainParam.epochs=25;

Ponownie trenujemy sieć, a następnie sprawdzamy jej odpowiedź na wektor wejściowy. Na podstawie wykresu oceniamy dokładność dopasowania.

neo=train(neo,A,B);

C=sim(neo,A);

figure

plot(A,B,'-.',A,C,'-o');

title('20 1 - 25 epok')

Powtarzamy procedurę dla 50, 100, 250 oraz 500 epok, wyciągamy wnioski. Czyścimy pamięć ze wszystkich danych:

clear

Tworzymy identyczne jak poprzednio wektory uczące A oraz B:

A=-5:0.1:5;

B=tan(A);

plot(A,B,'-.');

Tworzymy trójwarstwową sieć neuronową o nazwie ne3 składającą się z 10 neuronów w pierwszej warstwie, 10 w drugiej warstwie oraz jednego neuronu w warstwie wyjściowej:

ne3=newff([-5 5],[10 10 1],{'tansig' 'tansig' 'purelin'});

Nadajemy wagom sieci przypadkowe wartości:

ne3=init(ne3);

Sprawdzamy odpowiedź sieci niewytrenowanej na wektor wejściowy:

C=sim(ne3,A);

figure

plot(A,B,'-.',A,C,'-o');

title('10 10 1 - siec niewytrenowana')

Określamy maksymalną liczbę epok:

ne3.trainParam.epochs=5;

Trenujemy sieć:

ne3=train(ne3,A,B);

Sprawdzamy odpowiedź wytrenowanej sieci na wektor wejściowy na wykresie:

C=sim(ne3,A);

figure

plot(A,B,'-.',A,C,'-o');

title('10 10 1 - 5 epok')

Oceniamy dokładność dopasowania, nie zamykamy wykresu!

Ponownie nadajemy wagom sieci przypadkowe wartości, a następnie zwiększamy maksymalną liczbę epok do 25:

ne3=init(ne3);

ne3.trainParam.epochs=25;

Ponownie trenujemy sieć, a następnie sprawdzamy jej odpowiedź na wektor wejściowy. Na podstawie wykresu oceniamy dokładność dopasowania.

ne3=train(ne3,A,B);

C=sim(ne3,A);

figure

plot(A,B,'-.',A,C,'-o');

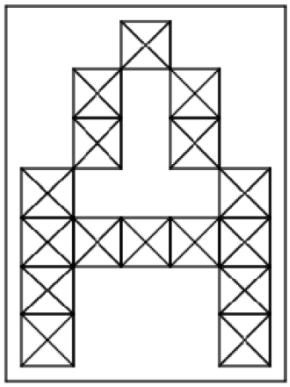
title('10 10 1 - 25 epok')

Powtarzamy procedurę dla 50, 100, 250 oraz 500 epok, wyciągamy wnioski.

- ***Przykład 2 – Rozpoznawanie znaków***

Wywołujemy definicje liter alfabetu oraz ich reprezentacje:

[alphabet,targets]=prprob;



Rys. 1. Przykład reprezentacji litery A.

Tworzymy nową sieć neuronową z 35 neuronami w warstwie wejściowej (ilość pól mogących tworzyć kształt litery) oraz 26 neuronami w warstwie wyjściowej. W warstwie ukrytej umieścimy na początek 10 neuronów.

net=newff(alphabet,targets,10,{'tansig' 'tansig'});

Definiujemy liczbę epok:

net.trainParam.epochs=250;

Trenujemy sieć:

net.divideFcn=''; */apostrof x 2/*

[net,tr]=train(net,alphabet,targets);

Wywołujemy przykładową literę z alfabetu, może to być B:

B=alphabet(:,2);

Wyświetlamy ją:

plotchar(B);

Odpytujemy sieć, aby rozpoznała wybraną literę i wyświetliła swoją odpowiedź:

y=sim(net,B);

y=compet(y);

answer=find(compet(y)==1);

figure;

plotchar(alphabet(:,answer));

Powtarzamy to samo dla innej litery, np. X:

X=alphabet(:,24);

figure;

plotchar(X);

y=sim(net,X);

y=compet(y);

answer=find(compet(y)==1);

figure;

plotchar(alphabet(:,answer));

Czyścimy obszar roboczy ze zbędnych danych:

clear;

Oceniamy jakość rozpoznawania liter przez sieć, a następnie zwiększamy ilość neuronów w warstwie ukrytej i powtarzamy całą procedurę. Powtórzenie procedury należy wykonać dla 50 oraz 75 neuronów w warstwie ukrytej.