Projekt finansowany w ramach programu Ministra Nauki i Szkolnictwa Wyższego pod nazwą „Regionalna Inicjatywa Doskonałości" w latach 2019 - 2022 nr projektu 020/RID/2018/19 kwota finansowania 12 000 000 PLN

Ćwiczenie SNA

Wielowarstwowy perceptron jako aproksymator funkcji

Część teoretyczna

Wykład na temat sieci neuronowych do aproksymacji funkcji.

Zadania pomocnicze

1. Zapoznaj się z programami wymienionymi poniżej (uruchomienie programów następuje poprzez wpisanie ich nazw w oknie Matlaba).
   - programy **nnd2n1** oraz **nnd2n2** modelują działanie prostego neuronu z różnymi funkcjami aktywacji
   - program **nnd11gn** pokazuje, w jaki sposób sieć dopasowuje funkcję aproksymującą do danych uczących
   - program **nnd12sd1** pokazuje wrażliwość algorytmu uczenia (wstecznej propagacji błędu) na powierzchnię błędu (jej stromość i płaskość)
   - program **nnd12mo** pokazuje to samo, co program nnd12sd1, ale w algorytmie uczenia można zmieniać współczynnik uczenia (learning rate) i wartość momentum
   - w programie **nnd12vl** zademonstrowano uczenie z adaptacyjnym współczynnikiem uczenia
   - w programie **nnd12cg** do uczenia sieci zastosowano metodę gradientów sprzężonych
   - w programie **nnd12m** zastosowano metodę Levenberga – Marquardta

   Ww. algorytmy uczenia opisano w [Osowski96].
2. Zapoznaj się z funkcjami `newff`, `train` i `sim` (help lub dokumentacja Matlaba).

Zadania do wykonania

Zaprojektuj sieć neuronową (wielowarstwowy perceptron) do aproksymacji funkcji. Funkcja, którą należy aproksymować dana jest w postaci punktów (x, y). Zadanie polega na znalezieniu takiej architektury sieci neuronowej (wielowarstwowego perceptronu), która da najlepsze wyniki na zbiorze testowym.

1. Generowanie zbiorów danych.

   1.1. Tworzymy funkcję generującą dane i zapisujemy pod nazwą f_aproksymowana.m:

```
function [y,q]=f_aproksymowana(x,nr_gr,rk,szum)
%inicjacja generatorów liczb pseudolosowych
rand('state',nr_gr*rk);
randn('state',nr_gr*rk);
%wzór funkcji
q=1+5*rand;
y=sin(cos(x*pi)*pi+1/(2*q)*cos((x+2*q)*pi*q)*pi)*0.5+randn(1,length(x))*szum;
```

   1.2. Generujemy funkcję docelową (idealną):

   Uwaga - polecenia zamieszczone w kolejnych punktach należy umieścić w jednym skrypcie.

```
nr_gr =
r_k =
x_t=-1:0.01:1;
[y_t,q]=f_aproksymowana(x_t,nr_gr,r_k,0);
```

   gdzie do `nr_gr` przypisz numer swojej sekcji a do `r_k` aktualny rok kalendarzowy.

   1.3. Generujemy zbiór uczący o liczności `l_u = 100` (są to punkty wygenerowane z funkcji docelowej zakłóconej szumem; `szum = 0.1`):

Projekt finansowany w ramach programu Ministra Nauki i Szkolnictwa Wyższego pod nazwą „Regionalna Inicjatywa Doskonałości" w latach 2019 - 2022 nr projektu 020/RID/2018/19 kwota finansowania 12 000 000 PLN

```
l_u=100;
rand('state',nr_gr*rk);
x_u=rand(1,l_u)*2-1;
y_u=f_aproksymowana(x_u,nr_gr,rk,0.1);
```

1.4. Pokazujemy dane na wykresie:

```
figure(1);
plot(x_t,y_t,'b--','linewidth',1);
hold on;
plot(x_u,y_u,'.','markersize',10);
legend('funkcja docelowa','punkty uczace');
title(['Nr gr = ',num2str(nr_gr),', q = ',num2str(q)]);
xlabel('x');
ylabel('y');
```

2. Tworzymy i uczymy sieć neuronową z `l_n` neuronami w warstwie ukrytej:

```
l_n=1; %liczba neuronów w warstwie ukrytej
%utworzenie sieci
net=newff([-1 1],[l_n 1],{'tansig','tansig'},'trainscg');
%pierwszy argument reprezentuje zakresy danych wejściowych,drugi - liczbę
neuronów w warstwie ukrytej i wyjściowej, trzeci - typy funkcji aktywacji w
tych warstwach, czwarty - metodę uczenia sieci; funkcja zwraca obiekt sieci
net
net.trainParam.epochs = 1000; %liczba epok uczenia
%trening sieci
net=train(net,x_u,y_u);
```

3. Testujemy sieć i wyznaczamy błędy aproksymacji:

```
y_u1= sim(net,x_u); %test na danych uczących
mse_u = mean((y_u1 - y_u).^2) %błąd na danych uczących

y_t1=sim(net,x_t); %test na danych testowych
mse_t = mean((y_t1 - y_t).^2) %błąd na danych testowych
```

4. Pokazujemy funkcję aproksymującą utworzoną przez sieć na wykresie:

```
figure(1);
plot(x_t,y_t1,'r','linewidth',2);
legend('funkcja docelowa','punkty uczace','funkcja aproksymujaca');
hold off;
```

5. Zaobserwuj jak zmieniają się wyniki w zależności od liczby neuronów ukrytych. Zmieniaj `l_n` w granicach od 1 do 15. Błędy zamieść w tabeli. Zaznacz wariant najlepszy. Pokaż wykres dla tego wariantu. Dodatkowo przeprowadź eksperymenty z 300 neuronami. Pokaż wykres dla tego wariantu.

6. Zaobserwuj jak zmieniają sie wyniki w zależności od funkcji aktywacji. W kodzie zamieszczonym w p. 2 zmieniaj funkcje aktywacji: `tansig`, `logsig`, `purelin`. Przetestuj każdy możliwy układ par tych trzech funkcji aktywacji. Eksperymenty wykonaj przy optymalnej liczbie neuronów `l_n` (najlepszy wariant z p. 5). Błędy zamieść w tabeli. Zaznacz wariant najlepszy. Pokaż wykres dla tego wariantu.

7. Zaobserwuj jak zmieniają sie wyniki w zależności od metody uczenia sieci. W kodzie zamieszczonym w p. 2 zamiast `trainscg` wpisz inną nazwę (nazwy funkcji implementujących metody uczenia znajdziesz w helpie, np. po wpisaniu „trainscg" ukażą się także inne nazwy zaczynające się od train…). Eksperymenty wykonaj dla pięciu metod uczenia, przy optymalnej liczbie neuronów `l_n` i najlepszych funkcjach aktywacji (najlepszy wariant z p. 5 i 6). Dla każdej metody uczenia pokaż wykres błędu w kolejnych epokach (wykres "Performance"). Błędy zamieść w tabeli. Zaznacz wariant najlepszy. Pokaż wykres aproksymacji dla tego wariantu.

8. Zaobserwuj jak zmieniają sie wyniki w zależności od liczby punktów uczących. W kodzie zamieszczonym w p. 1.3 zmieniaj `l_u` od 10 do 100 z krokiem 10 i dalej do 1000 z krokiem 100. Eksperymenty wykonaj dla optymalnego wariantu sieci (najlepszy wariant z p. 5, 6 i 7). Błędy zamieść w tabeli. Sporządź wykresy błędów `mse_u` i `mse_t` w zależności od `l_u`.

Zawartość sprawozdania

Sprawozdania powinny być sporządzone według wzoru zamieszczonego na stronie i zawierać:
A) Cel ćwiczenia.
B) Treść zadania.
C) Opis sieci neuronowej używanej w zadaniu (nie kopiuj treści wykładu, poszukaj w literaturze i Internecie).
D) Metodyka rozwiązania zadania.
E) Zestawienie wyników (wykresy, tabele z komentarzem).
F) Wnioski końcowe.
G) Wydruk programu.

Zadania dodatkowe dla ambitnych

Wybrane zadanie student wykonuje indywidualnie, po uzgodnieniu z prowadzącym. Zadania nie są obligatoryjne. Z zadania sporządzamy sprawozdanie.

1. Sprawdź jak uczy się sieć na danych uczących z różną zawartością szumu. Jakie błędy aproksymacji wtedy się obserwuje?
2. Sprawdź jak działa nauczona sieć dla x-ów spoza zakresu (-1, 1). Jakie błędy aproksymacji wtedy się obserwuje?
3. Oprogramuj zadanie aproksymacji funkcji trzech zmiennych y = f(x1, x2, x3). Dobierz eksperymentalnie parametry sieci.
4. Oprogramuj przykład 4.4 z [Żurada96]. Wykonaj obliczenia i wykreśl powierzchnie błędów opisane w tym przykładzie. Porównaj wyniki.
5. Oprogramuj zadanie 4.15 z [Żurada96]. Wykonaj symulacje dla 20 punktów testowych, policz błędy. Zwizualizuj funkcję i aproksymantę utworzoną przez sieć.
6. Oprogramuj przykład ze str. 296 i 297 z [Żurada96] – przetwornik współrzędnych biegunowych na kartezjańskie. Wykonaj symulacje i wykreśl mapy błędów (jak na rys. 8.12).
7. Wykonaj podobne ćwiczenie w innym środowisku, np. R, Python, Statistica, C#, ...

Przykładowe zagadnienia i pytania zaliczeniowe

1. Narysuj model sztucznego neuronu.
2. Narysuj model sieci neuronowej użytej w ćwiczeniu.
3. Na czym polega przeuczenie/niedouczenie sieci.
4. Narysuj i objaśnij wykres ze sprawozdania.
5. Na czym polega aproksymacja funkcji.
6. Funkcje aktywacji neuronów.
7. Wsteczna propagacja błędów.
8. Problem generalizacji.
9. Miary dopasowania modelu.

Do przygotowania na następne zajęcia

1. Zapoznać się z instrukcją do kolejnego ćwiczenia.
2. Zapoznać się z częścią teoretyczną do kolejnego ćwiczenia.
3. Wykonać zadania pomocnicze do kolejnego ćwiczenia.