Ćwiczenie WPK

Wielowarstwowy perceptron jako klasyfikator

Część teoretyczna

Wykład 6: Sztuczne sieci neuronowe – klasyfikacja.

Zadania pomocnicze

Zapoznaj się z funkcjami `newff`, `train` i `sim` (dokumentacja pakietu Octave z sieciami neuronowymi neuralNetworkPackageForOctaveUsersGu.pdf (Google)).

Zadania do wykonania

Dane są zbiory pacjentów: chorych A oraz zdrowych B. Każdy obiekt (człowiek) w tych zbiorach opisany jest dwiema cechami x1 i x2 (np. temperatura ciała, parametry krwi, występowanie lub brak określonych alleli genów itp.). Zaprojektuj klasyfikator neuronowy na bazie perceptronu do rozróżniania klas A i B. Zadanie wykonaj w trzech wariantach:

- zbiory A i B separowalne liniowo,
- zbiory A i B separowalne nieliniowo,
- zbiory A i B częściowo pokrywające się.

1. Generowanie zbiorów danych.

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

```
function [A,B]=gener_zbiorow(nr_gr,wariant,lp,rk)
%nr_gr - nr sekcji np. 1.1, 1.2, ...
%wariant - 1 - zbiory A i B separowalne liniowo, 2 - separowalne
%nieliniowo, 3 - częściowo pokrywające się
%lp - liczba punktów w zbiorach A i B
%rk – rok kalendarzowy

rand('state',nr_gr*rk);
if wariant==1
    Ap=[0.1 0.9; 0.5+rand*0.5 0.5-rand*0.5];
    [A,B]=gener_AB(Ap,Ap,lp,lp,1,1);
elseif wariant==2
    Ap=[0.1 0.4 0.9; 0.5-rand*0.5 0.5+rand*0.2 0.5-rand*0.5];
    [A,B]=gener_AB(Ap,Ap,lp,lp,2,2);
elseif wariant==3
    Ap=[0.1 0.4 0.9; 0.5-rand*0.5 0.5+rand*0.3 0.5-rand*0.5];
    Bp=[0.1 0.25 0.55 0.7 1; 0.5+rand*0.1 0.4-rand*0.3 0.4+rand*0.2 0.5 0.1];
    [A,B]=gener_AB(Ap,Bp,lp,lp,2,4);
 end;
 title(['Nr gr = ', num2str(nr_gr)]);

 function [A,B]=gener_AB(Ap,Bp,Al,Bl,Ast,Bst)
     hold off;

     Aw=polyfit(Ap(1,:),Ap(2,:),Ast);
     Bw=polyfit(Bp(1,:),Bp(2,:),Bst);

     rand('state',sum(100*clock));
     ll=0; i=0;
     while (ll<Al)&&(i<Al*100)
         x=rand;
         y=rand;
         ymax=polyval(Aw,x);
         if (ymax>=0)&&(ymax<=1)&&(y<ymax)
```

```
            ll=ll+1;
            A(1,ll)=x;
            A(2,ll)=y;
        end;
        i=i+1;
    end;

    ll=0; i=0;
    while (ll<Bl)&&(i<Bl*100)
        x=rand;
        y=rand;
        ymin=polyval(Bw,x);
        if (ymin>=0)&&(ymin<=1)&&(y>ymin)
            ll=ll+1;
            B(1,ll)=x;
            B(2,ll)=y;
        end;
        i=i+1;
    end;

    figure(2);
    plot(A(1,:),A(2,:),'ro');
    hold on;
    plot(B(1,:),B(2,:),'bx');
    plot(Ap(1,:),Ap(2,:),'r.');
    plot(Bp(1,:),Bp(2,:),'b.');
    v=0:0.02:1;
    plot(v,polyval(Aw,v),'r--');
    plot(v,polyval(Bw,v),'b--');
    xlim([0 1]); ylim([0 1]);
    xlabel('x1'); ylabel('x2');
```

1.2. Generujemy dane dla zbiorów separowalnych liniowo:

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

```
[A_u,B_u]=gener_zbiorow(nr_gr,1,100,rk); %dane uczące
[A_t,B_t]=gener_zbiorow(nr_gr,1,100,rk); %dane testowe
```

gdzie za `nr_gr` wstaw numer swojej sekcji, a za `rk` rok kalendarzowy.

1.3. Na podstawie wygenerowanych zbiorów tworzymy macierze wejściowe. Tworzymy wektory pożądanych odpowiedzi:

```
%zbiór uczący
x_u=[A_u B_u]; %wektory wejściowe
t_u=[zeros(1,size(A_u,2))+1 zeros(1,size(B_u,2))-1]; %pożądane odpowiedzi: +1
dla zbioru A i -1 dla zbioru B

%zbiór testujący
x_t=[A_t B_t]; %wektory wejściowe
t_t=[zeros(1,size(A_t,2))+1 zeros(1,size(B_t,2))-1]; %pożądane odpowiedzi: +1
dla zbioru A i -1 dla zbioru B
```

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([0 1;0 1],[l_n 1],{'tansig','tansig'},'trainlm');
%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 = 200; %liczba epok uczenia
[net]=train(net,x_u,t_u); %trening sieci
```

3. Testujemy sieć i wyznaczamy błędy klasyfikacji na zbiorze uczącym:

```
%symulacja nauczonej sieci na zbiorze uczącym
y=sim(net,x_u);
for i=1:length(y)
    if y(i)>0.8 y(i)=1; %jeśli odpowiedź sieci jest w zakresie od 0.8 do 1,
                        %przyjmujemy +1
    elseif y(i)<-0.8 y(i)=-1; %jeśli odpowiedź sieci jest w zakresie od -0.8
                              %do -1, przyjmujemy -1
    else y(i)=0; end; %jeśli odpowiedź sieci jest w zakresie od -0.8 do 0.8,
                      %przyjmujemy 0, co oznacza brak decyzji
end;

%błędy
lb_u=0;%liczba punktów źle zaklasyfikowanych
ln_u=0;%liczba punktów niezaklasyfikowanych do żadnej klasy
for i=1:length(y)
    if y(i)==0 ln_u=ln_u+1;
    elseif y(i)~=t_u(i) lb_u=lb_u+1;
    end;
end;
lb_u=lb_u/length(y)*100; %lb_u wyrażone w procentach
ln_u=ln_u/length(y)*100; %ln_u wyrażone w procentach
```

Uwaga – powyżej przyjęto zasadę, że:

- jeśli odpowiedź sieci jest w zakresie od 0.8 do 1, przyjmujemy +1, co oznacza klasę A,
- jeśli odpowiedź sieci jest w zakresie od –0.8 do –1, przyjmujemy –1, co oznacza klasę B,
- jeśli odpowiedź sieci jest w zakresie od –0.8 do 0.8, przyjmujemy 0, co oznacza brak decyzji.

4. Testujemy sieć i wyznaczamy błędy klasyfikacji na zbiorze testowym:

```
%symulacja nauczonej sieci na zbiorze testowym
y=sim(net,x_t);
for i=1:length(y)
    if y(i)>0.8 y(i)=1;
    elseif y(i)<-0.8 y(i)=-1;
    else y(i)=0; end;
end;
%błędy
lb_t=0;%liczba punktów źle zaklasyfikowanych
ln_t=0;%liczba punktów niezaklasyfikowanych do żadnej klasy
for i=1:length(y)
    if y(i)==0 ln_t=ln_t+1;
    elseif y(i)~=t_t(i) lb_t=lb_t+1;
    end;
end;
lb_t=lb_t/length(y)*100; %lb_t wyrażone w procentach
ln_t=ln_t/length(y)*100; %ln_t wyrażone w procentach
```

5. Prezentacja wyników:

```
fprintf('Odsetek poprawnych, złych i nierozstrzygniętych klasyfikacji\n')
fprintf('             %5.2f%%   -   %5.2f%%   -   %5.2f%%  w zbiorze
treningowym  \n',(100-lb_u-ln_u),lb_u,ln_u)
fprintf('             %5.2f%%   -   %5.2f%%   -   %5.2f%%  w zbiorze testowym
\n',(100-lb_t-ln_t),lb_t,ln_t)

%wykreślenie obszarów decyzyjnych utworzonych przez sieć
a=0:0.01:1;
i=1;
for b=0:0.01:1
    x=[a;zeros(1,length(a))+b];
    yy(i,:)=sim(net,x);
    i=i+1;
end;
```

```
figure(1);
surf(a,a,yy,'EdgeColor','none');
colorbar;
title('Powierzchnia decyzyjna utworzona przez sieć');
xlabel('x1'); ylabel('x2');
view(2);
grid off;
figure(2);
yy(find(yy>0.8))=1; yy(find(yy<-0.8))=-1; yy(find(yy>=-0.8&yy<=0.8))=0;
contour(a,a,yy);
title('Obszary decyzyjne:');
```

6. Zaobserwuj jak zmieniają się wyniki w zależności od liczby neuronów ukrytych. Zmieniaj `l_n` w granicach od 1 do wartości, przy której nie obserwuje się już poprawy rezultatów. Błędy zamieść w tabeli. Zaznacz wariant najlepszy. Pokaż wykresy dla tego wariantu.

7. 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. 6). Błędy zamieść w tabeli. Zaznacz wariant najlepszy. Pokaż wykresy dla tego wariantu.

8. Powtórz eksperymenty dla danych separowanych nieliniowo. W tym celu w kodzie zamieszczonym w p. 1.2 wprowadź `2` zamiast `1` jako drugi parametr funkcji `gener_zbiorow`. Wykonaj ponownie p. 6 i 7 dla nowych danych.

9. Powtórz eksperymenty dla danych nieseparowanych. W tym celu w kodzie zamieszczonym w p. 1.2 wprowadź `3` zamiast `2` jako drugi parametr funkcji `gener_zbiorow`. Wykonaj ponownie p. 6 i 7 dla nowych danych.

Zawartość sprawozdania

Sprawozdania powinny być sporządzone według wzoru zamieszczonego na stronie i zawierać:

A) Cel ćwiczenia.
B) Treść zadania.
C) Opis problemu klasyfikacji, sieci neuronowe stosowane do klasyfikacji (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

Te zadania studenci wykonują 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 (szum dodaj do wektorów wejściowych x_u). Jakie błędy klasyfikacji wtedy się obserwuje?
2. Oprogramuj przykład 3.2 z [Żurada96]. Wykonaj obliczenia i zobrazuj działanie dychotomizatora wykresami (rys. 3.12). Porównaj wyniki.
3. Oprogramuj przykład 3.3 z [Żurada96]. Wykonaj obliczenia i zobrazuj działanie klasyfikatora ciągłego wykresami (rys. 3.16). Porównaj wyniki.
4. Oprogramuj przykład 3.4 z [Żurada96]. Wykonaj obliczenia i zobrazuj działanie klasyfikatora ciągłego wykresem (rys. 3.18). Porównaj wyniki.
5. Oprogramuj przykład 4.3 z [Żurada96]. Wykonaj obliczenia i wykreśl linie decyzyjne opisane w tym przykładzie. Porównaj wyniki.
6. Wykonaj podobne ćwiczenie w innym środowisku, np. R, Python, Statistica, C#, ...

Przykładowe zagadnienia i pytania zaliczeniowe

1. Narysuj model sztucznego neuronu.
2. Neuron jako klasyfikator.
3. Narysuj model sieci neuronowej użytej w ćwiczeniu.
4. Na czym polega klasyfikacja danych.
5. Narysuj i objaśnij wykres ze sprawozdania.
6. Funkcje aktywacji neuronów.
7. Wsteczna propagacja błędów.
8. Metody oceny klasyfikatora.

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.