Sztuczna Inteligencja

Projekt

Realizacja radialnej sieci neuronowej uczonej algorytmem wstecznej propagacji błędu (learnbp) uczącej się rozpoznawania kwiatów Irysa

Miśkowicz Brajan
163976, EF-DI, P4

Rzeszów, 2021

Spis treści

Rozdział 1

Wstęp

1.1 Opis projektu

Tematem projektu jest Realizacja radialnej sieci neuronowej uczonej algorytmem wstecznej propagacji błędu (learnbp) uczącej się rozpoznawania kwiatów Irysa. Polega on na napisaniu odpowiednich skryptów pozwalających na wykorzystanie sieci neuronowej - nauczenie jej rozpoznawania kwiatów Irysa częścią danych - danymi uczącymi, oraz przetestowanie sieci pozostałymi danymi - testującymi. Ponadto dzięki analizie otrzymanych wyników kolejnych eksperymentów należy wyznaczyć takie wartości parametrów występujących w algorytmie wstecznej propagacji, aby sieć działała możliwie najlepiej, co zostanie opisane szerszej podczas eksperymentów.

1.2 Przedstawienie danych uczących

Używanym zbiorem podczas projektu jest zbiór danych kwiatów irysa dostępny na stronie: `http://archive.ics.uci.edu/ml/datasets/Iris`. W zbiorze znajdują się cztery parametry opisujące trzy gatunki kwiatów irysa: Iris Setosa, Iris Versicolour, oraz Iris Virginica. Wśród opisu zbioru można znaleźć informacje, że znajduje się w nim 150 instancj, po 50 z każdego gatunku. Wspomniane wcześniej parametry opisujące kwiaty to:

- długość kielicha [cm],
- szerokość kielicha [cm],
- długość płatka [cm],
- szerokość płatka [cm].

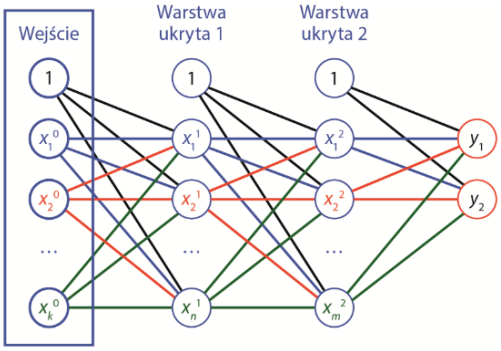
W zbiorze nie pojawiają się braki danych co ułatwia ich normalizację. Na potrzeby projektu gatunki Iris Setosa, Iris Versicolour, oraz Iris Virginica zostały oznaczone podczas testów odpowiednio 1, 2 i 3. Z racji rozbierzności danych zostały one znormalizowane do zakresu wartości od -1 do 1.

Rozdział 2

Teoria

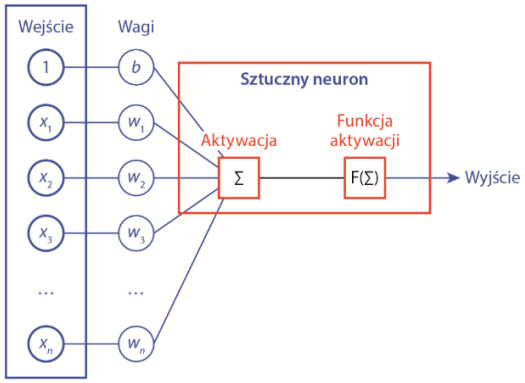
2.1 Opis neuronu i sieci neuronowej

Neuron reprezentuje jednostkę przetwarzania, która przyjmuje wartość wejściową i, zgodnie z określonymi ragułami, przekazuje inną wartość na wyjściu. Sieci neuronowe to struktury składające się z neuronów połączonych synapsami. Potraktować można każdy neuron jak swego rodzaju procesor, który sumuje z odpowiednimi wagami sygnały wejściowe pochodzące z innych neuronów, następnie tworzy nieliniową funkcję sumy i przekazuje tę wartość do innych neuronów z nim powiązanych - w podobny sposób działają sztuczne sieci neurononwe. Stosowana jest w tym projekcie taka, jedna z najpowszechniejszych sieci - jednokierunkowa wielowarstwowa. W sieciach jednowarstwowych w przeciwieństwie do rekurencyjnych występuje tylko jeden kierunek przepływu sygnału, to znaczy, że każdy sygnał przechodzi przez dany neuron tylko raz w cyklu. Sieci wielowarstwowe posiadają: warstwę wejściową, co najmniej jedną warstwę ukrytą i jedną warstwę wyjściową. Każda warstwa neuronów posiada macież wag, wektor przesunięć, funkcję aktywacji i wektor sygnałów wyjściowych. Typowa sieć wielowarstwowa jednokierunkowa posiada pełne połączenia między warstwami, tj. wyjście neuronu z jednej warstwy przekazywane jest na wejście wszystkich na warstwie kolejnej.



Rysunek 2.1: Ogólna postać sieci neuronowej wielowarstwowej[1]

Przedstawiona została powyżej sieć neuronowa posiadająca dwie warstwy ukryte, z k neuronami na wejściu, n neuronów w pierwszej warstwie, m neuronów w drugiej warstwie i dwa wyjścia y. Dla uproszczenia na rysunku nie zostały oznaczone wagi, jednak każde z tych połączeń między poszczególnymi neuronami posiada swoją wagę. Każda zmiana wagi w neuronu odbywa się proporcjonalnie do iloczynu jego sygnału wejściowego oraz wyjściowego. U góry obrazka można zauważyć tak zwany biasy, które mają stałą wartość (na ogół równą 1) i mają za zadanie zmieniać wartość zwracaną przez neutrony.



Rysunek 2.2: Budowa sztucznego neuronu[1]

Powyższy rysunek pokazuje model neuronu posiadający n wejść x i jedno wyjście. Najpierw sumowane są wszystkie składowe przemnożone przez ich wagi w, a później wynik sumowania poddawany jest działaniu funkcji aktywacji, która generuje wartość na wyjście. Ponownie pojawia się tutaj bias z wagą oznaczoną jako b. Mając taki obraz neuronu i jego działania można wyprowadzić wzór opisujący neuron:

$$y = f\left(\sum_{i=1}^{n} x_i w_i + b\right)$$

y jest tutaj wyjściem neuronu, natomiast f oznacza funkcję aktywacji, ale można zapisać ją w prostrzy sposób zastępując sumowanie kolejnych wartości macierzami:

$$y = f\left(xw + b\right)$$

Mając ten wzór oraz wiedzę na temat budowy sieci możeny wyznaczyć wzory matematyczne poszczególnych warstw - dwóch ukrytych i wyjściowej:

$$y^{(1)} = f^{(1)}\left(xw^{(1)} + b^{(1)}\right)$$

$$y^{(2)} = f^{(2)}\left(y^{(1)}w^{(2)} + b^{(2)}\right)$$

$$y^{(3)} = f^{(3)}\left(y^{(2)}w^{(3)} + b^{(3)}\right)$$

Oznaczenia tutaj są identyczne, jak wcześniej. Górne indeksy mówią natomiast, z której warstwy brane są wartości.

2.2 Propagacja wsteczna

Każda sieć neuronowa jest przybliżeniem funkcji, więc różni się od niej o pewną wartość. Ta wartość to błąd, a naszym celem powinno być dążenie do zminimalizowania go. Ponieważ w sieci neuronowej błąd jest funkcją wag, to chcemy więc zminimalizować błędy w stosunku do wag. W algorytmie wstecznej propagacji najpierw obliczany jest sygnał wyjściowy, a na jego podstawie błąd warstwy wyjściowej. Następnie odwórcony zostaje kierunek przepływu sygnałów, a funkcje aktywacji zostają zastąpione przez ich pochodne, wagi są aktualizowane o wartość współczynnika uczenia pomnożoną przez kierunek minimalizacji.Wzór opisujący wartość nowej wagi k+1 jest więc postaci: $w(t+1) = w(t) + \Delta w$, gdzie t oznacza epokę, czyli numer cyklu uczenia. Kierunek minimalizacji tworzony jest na podstawie przestrzeni wielowymiarowej w taki sposób, aby wagi podążały w kierunku punktu o najmniejszej wartości błędu. Stosowany do wyznaczenia kierunku jest wektor gradientu względem wag wszystkich warstw sieci. Zmiana wartości wag jest równa: $\Delta w = \eta p(w)$, gdzie η jest współczynnikiem uczenia, a p(w) kierunkiem minimalizacji.

Rozdział 3

Eksperymenty

3.1 Eksperyment 1 - podział danych

```
clear
disp_freq = 100;
max_epoch = 40000;
err_goal = 1e-40;%0.01;
max_fail = 10000;
load iris

S1=10;      %for ind_S1 % METAPARAMETR
S2=9;      %for ind_S2 % METAPARAMETR
lr = 0.1;   %for ind_lr % METAPARAMETR

averages = zeros([1,9]);

for tries = 1:1:20
    for x = 1:9
        r = 1;
        t = 1;
        for o = 1:3
            [traind] = crossvalind('Holdout', 50, x/10);
            for k = 1:50
               if (traind(k) == 1)
                   Ptest(:,t) = Pn(:,k+50*(o-1));
                   Ttest(1,t) = T(1,k+50*(o-1));
                   t = t + 1;
               else
                   Plearn(:,r) = Pn(:,k+50*(o-1));
                   Tlearn(1,r) = T(1, k+50*(o-1));
                   r = r + 1;
               end
            end
        end
        net = feedforwardnet([S1, S2],'traingd');   %definicja perceptonu
        net.trainParam.epochs = max_epoch;  %maksymalna liczba epok
        net.trainParam.goal = err_goal; %cel wydajnosci
        net.trainParam.lr = lr; %learning rate
        net.trainParam.max_fail = max_fail; %maksymalna ilosc bledow walidacji
        net.trainParam.showWindow = false;  %czy pokazac okno uczenia
        net.divideParam.trainRatio=1;   %ilosc danych do uczenia
        net.divideParam.valRatio=0; %ilosc danych do walidacji
        net.divideParam.testRatio=0;    %ilosc danych do testowania
        [net,tr] = train(net,Plearn,Tlearn); %uczenie sieci neuronowej
        u = net(Ptest);
        result = (1-sum(abs(Ttest-u)>=0.5)/length(Ttest))*100;
        averages(x) = averages(x)+result;
        clear Plearn; clear Ptest; clear Tlearn; clear Ttest;
    end
end
percents = [10, 20, 30, 40, 50, 60, 70, 80, 90];
plot(percents, averages/20);
title('Zaleznosc poprawnosci klasyfikacji od podzialu zbioru');
xlabel('wielkosc zbioru uczacego [% calosci zbioru]') ;
ylabel('poprawnosc klasyfikacji [%]');
grid;
```

Rozdział 4

Wnioski

Bibliografia

[1] Zocca, Valentino ; Spacagna, Gianmario ; Slater, Daniel ; Roelantsi, Peter: *Deep Learning. Uczenie głębokie z językiem Python. Sztuczna inteligencja i sieci neuronowe.* 2018. – ISBN 978–83–283–4174–6