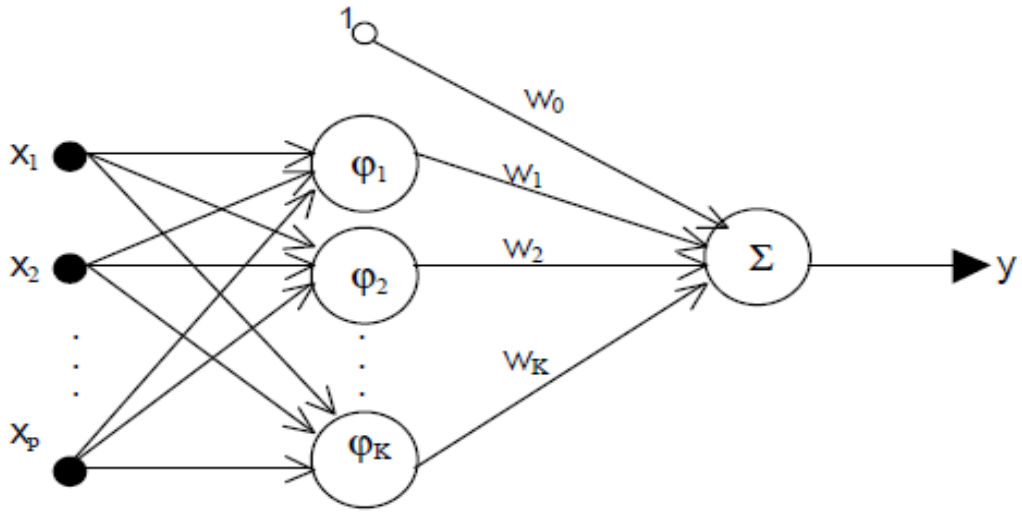
Sieci neuronowe typu RBF

Wstęp

Sieci RBF są przykładem sieci neuronowych typu Feed Forward, czyli sieci jednokierunkowych. Obok sieci MLP są one najpopularniejszymi sieciami neuronowymi, które znalazły szerokie zastosowanie praktyczne. Cechą charakterystyczną sieci RBF jest to iż składaja się one zawsze z dwóch warstw – warstwy neuronów radialnych oraz warstwy neuronów wyjściowych.



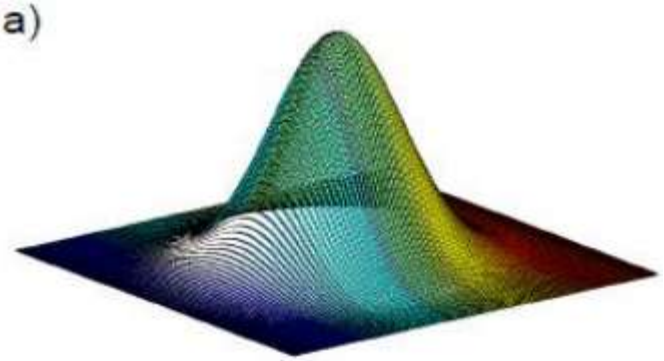
Co można wyrazić zależnością:

$$F(x) = \sum_{i=1}^{K} w_i G(\|x - t_i\|_A)$$

Gdzie zwykle

$$G(\|x - t\|) = \exp\left(\frac{-\|x - t\|^2}{2\sigma^2}\right)$$

a)



Analiza równania *F(x)* wskazuje na parametry które można poddać adaptacji podczas procesu uczeni sieci RBF. Są nimi przede wszystkim wartości wag w_i oraz parametry neuronów radialnych czyli *A* oraz t_i .

W praktyce treningu radialnych sieci neuronowych definiuje się trzy etapu ich uczenia:

1) Etap 1 – uczenie jedynie wag neuronu liniowego w_i. Proces ten zwykle realizowany jest poprzez minimalizację błędu średniokwadratowego, co sprowadza się do wyznaczenia wag *w* z zależności

   $\mathbf{\Phi w} = \mathbf{y}$ skad: $\mathbf{w} = \mathbf{\Phi}^{-1}\mathbf{y}$

   Ponieważ macierz $\mathbf{\Phi}$ zwykle jest prostokątna w celu jej odwrócenia zwykle stosuje się macierz pseudoodwrotną wyznaczaną z zalezności

   $\Phi^{+}=(\Phi^{T}\Phi)^{-1}\Phi^{T}$

   By ostatecznie wyznaczyć *w*

   $\mathbf{w} = \Phi^{+}\mathbf{y}$
2) Etap 2 – odpowiedni dobór centrów funkcji radialnych (niezależnie od uczenia wag). W etapie tym zwykle wykorzystuje się algorytmy selekcji prototypów typu ENN lub CNN albo też algorytmy klasteryzacji w celu wyznaczenia położenia centrów funkcji radialnych czyli parametrów t_i. Po wyznaczeniu parametrów centrów funkcji radialnych używa się ich w celu przekształcenia danych, tak by w kolejnym kroku dokonać uczenia warstwy wyjściowej zgodnie z zasadami opisanymi w etapie 1.
3) Etap 3 – w etapie tym uczeniu poddawane są jednocześnie wszystkie parametry sieci RBF z wykorzystaniem algorytmów gradientowych.

W pierwszym jak i w drugim etapie parametr σ^2 zwykle definiowany jest ręcznie poprzez przeszukanie odpowiedniej jego wartości.

Sieci neuronowe typu RBF znalazły zastosowanie zarówno w problemach klasyfikacji jak i regresji. W tych pierwszych należy jednak zwrócić uwagę iż pojedynczy neuron powinien być uzyty jedynie do rozpoznawania par klas lub odpowiednio wg. zalezności klasa-reszta. Powoduje to że w przypadkach wieloklasowych liczba neuronów wyjściowych powinna być co najmniej równa liczbie klas.

Do zrobienia w Matlabie

Zaimplementuj funkcje:

M = rbf_ucz(trening,typ_selekcji,rbf_param,typ)

Oraz

[wyn] = rbf_test(M,test);

Gdzie

M – Nauczony model sieci RBF składający się z pól: M.X – zbiór centrów neuronów radialnych, M.W zbiór wag poszczególnych neuronów, M.rbf_param –parametr szerokości funkcji radialnej
Typ_selekcji – string opisujący algorytm selekcji wektorów referencyjnych: 'enn_sel(te,3);' , 'cnn_sel(te);', 'vq…'
rbf_param – parametr (szerokość) neuronu radialnego
typ_selekcji – string opisujacy funkcję uzytą do selekcji centrów funkcj radialnych
typ – typ wartości przewidywanej – czy problem klasyfikacj czy regresji

Budowa funkcji rbf_ucz

```
Function  M = rbf_ucz(trening,typ_selekcji,rbf_param,typ)
P = eval(typ_selekcji); %Wywołanie selekcji wektorów referencyjnych (położenia neuronów funkcji radialnych)
%Oblicz macierz odległości D pomiędzy każdym wektorem zbioru treningowego i każdym prototypem z P
%D(i,j) powinno odpowiadać i – numer wiersza, czyli maksymalnie i = liczba wektorów uczących z „trening”, j – numer kolumny, czyli maksymalnie j = liczba wektorów uzyskanych z P
X = exp(- rbf_param * D); % przejście z odległości do podobieństwa
Y = trening.Y
%Oblicz wagi W wg zależności z części teoretycznej korzystając z X i Y
M.X = P.X;
M.rbf_param = rbg_param
M.W = W;
M.typ = typ;
```

UWAGA

Funkcja eval powoduje wykonanie poleeń Matlaba zapisanych w postaci stringu.

Budowa funkcji rbf_test

```
function wyn = rbf_test(M,test)
%Oblicz macierz odległości D pomiędzy każdym prototypem z P i każdym wektorem zbioru testowego
%D(i,j) powinno odpowiadać i – numer wiersza, czyli maksymalnie i = liczba wektorów testowych z „test”, j – numer kolumny, czyli maksymalnie j = liczba neuronów radialnych
X = exp(- rbf_param * D); % przejście z odległości do podobieństwa
wyn = X * W;
```

```
if strcmp(M.typ,'klas')
  wyn = round(wyn);
end
```

Przykładowe wykorzystanie

Badań dokonaj w oparciu o skrypt:

```
clear; clc;
d = load('  … ');
acc = zeros(1,10);
for I = 1:10
    [trening, test] = podziel(d,0.7);
    M = rbf_ucz(trening,'cnn_sel()',1,'klas');
    wyn = rbf_test(M,test);
    acc[i] = dokladnosc(test,wyn);
end;
disp( [ num2str(mean(acc)) '+-' num2str(std(acc))]);
```

Powyższy skrypt umożliwia obliczenie średniej dokładności oraz jej odchylenia standardowego w zależności od tego jak funkcja podziel podzieli się zbiór dany.

Zadania

1. Zbadaj wpływ parametru rbf_param na jakość uzyskiwanych wyników
2. Zbadaj wpływ metody selekcji wektorów referencyjnych (położenia neuronów radialnych) na jakość uzyskiwanych wyników
3. Narysuj wykres osobno dla każdej z metod selekcji wektorów referencyjnych zależności parametru rbf_param i dokładności klasyfikacji
4. Pobierz najlepszy wynik uzyskany dla danej metody selekcji prototypów i wykreśl wykres słupkowy pokazujący dokładność w zależności od wybranej metody selekcji
5. Zastanów się jak rozwiązać problem klasyfikacji wieloklasowej za pomocą sieci RBF i zaimplementuj wersję sieci RBF dla tego typu problemów.
6. Zastanów się jak i policz błąd średniokwadratowy sieci RBF