Alicja Kapiszka, Informatyka N1, gr. 30C

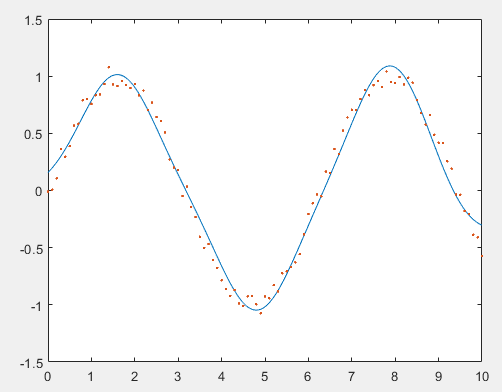
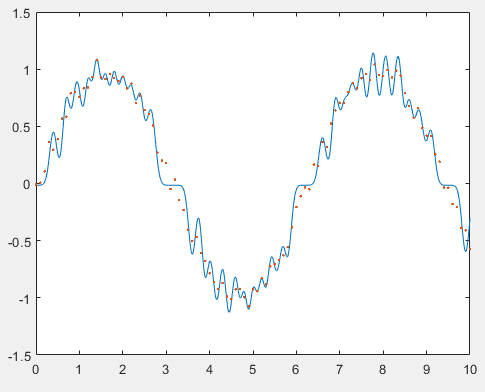
**Sieci neuronowe – sprawozdanie nr 3**

**Zadanie 1.**

Dla wybranych 3 plików z danymi należało: zapoznać się z danymi, dobrać najkorzystniejszą szerokość neuronów RBF (spread) oraz przeprowadzić uczenie i porównać jakość wyników. Wyniki zebrałam w poniższej tabeli, wytłuszczając najlepszy dla danej sieci spread.

|  |  |  |
| --- | --- | --- |
| Dane | Spread | Ilość neuronów |
| dane\_sin1a | **1** | 4 |
| 0.1 | 45 |
| dane\_sin3a | 1 | 6 |
| 0.5 | 10 |
| 2 | 5 |
| **2.5** | 4 |
| percep | **1** | 5 |
| 0.5 | 8 |
| 1.5 | 6 |
| dane\_a | 1 | 23 |
| 1.3 | 21 |
| **1.2** | 19 |
| dane3d | **1** | 2 |

Poniżej przedstawiam przykład uczenia się sieci z różnymi parametrami spread. Po lewej sieć nauczona danych (dane\_sin1a) z parametrem spread = 1 (prawidłowe uczenie sieci), a po prawej ze spread = 0.1 (sieć przeuczona).

**Zadanie 2.**

Należało napisać skrypt, który podzieli dane na część uczącą i testującą w zadanej proporcji oraz dobierze najkorzystniejszą wartość parametru spread sieci przy uczeniu z wykorzystaniem polecenianewrbe lub newgrnn, przy czym dane testujące miały być wykorzystane do określenia, kiedy należy przerwać zmniejszanie parametru spread oraz określenia rzeczywistej dokładności sieci.

close all;

clc;

% loading and random sorting of the data

we = load('dane\_sin1a\_i.txt');

wy = load('dane\_sin1a\_o.txt');

mix = randperm(length(wy));

we = we(mix,:)';

wy = wy(mix)';

% dividing the data to the learning and test data

in\_learn = [];

in\_test = [];

out\_learn = [];

out\_test = [];

procent = 30;

ratio = procent/100 \* length(wy);

for i = 1:length(wy)

if i <= ratio

in\_test = [in\_test we(:,i)];

out\_test = [out\_test wy(i)];

else

in\_learn = [in\_learn we(:,i)];

out\_learn = [out\_learn wy(i)];

end

end

% learning and testing the net

results = [];

all\_err = [];

for i = 1:100

err = [];

spread = logspace(-2, 2, 100)';

for n = 1:length(spread)

net = newgrnn(in\_learn, out\_learn, spread(n));

y = sim(net, in\_test);

d = sum((out\_test - y).^2);

err = [err d];

end

% finding the best spread

[min\_d, spread\_ind] = min(err);

results = [results spread(spread\_ind)];

all\_err = [all\_err; err];

end

% finding the best mean spread and drawing a plot

mean(results)

mean\_err = [];

for i = 1:size(all\_err,2)

mean\_err = [mean\_err mean(all\_err(:,i))];

end

figure; % hold on is off because then the axis' scale is not logarythmic

semilogx(spread, mean\_err); % error plot

% finding the best mean spread and drawing a plot

mean(results)

mean\_err = [];

for i = 1:size(all\_err,2)

mean\_err = [mean\_err mean(all\_err(:,i))];

end

figure; % hold on is off because then the axis' scale is not logarythmic

semilogx(spread, mean\_err); % error plot

Jako, że wagi początkowe są losowane, w celu otrzymania jak najdokładniejszego wyniku, obliczenia wartości spread wykonano 100 razy w pętli. Otrzymano **optymalną wartość spread równą 0,1485** oraz wykres zależności błędu (oś y) od wartości spread (oś x), na którym widać miejsce, gdzie spread jest najmniejszy:

