# Einführung in die Neuroinformatik

Tim Luchterhand, Paul Nykiel (Gruppe P)

3. Juli 2018

## 1 Dropout

### 1.1

a) Durch Dropout wird die Relevanz einzelner Neuronen reduziert, da das trainiert Netzwerk auch nur mit einer Teilmenge aller Neuronen funktioniert. Es werden quasi mehrere Netze mit weniger Neuronen trainiert und dann der Durchschnitt verwendet. Im Gegensatz zur naiven Implementierung mit mehreren Netzten wird allerdings deutlich weniger Rechenkapazität benötigt.

b) Die Gewichte müssen mit dem Faktor $1 - p = \frac{2}{3}$ skaliert werden.

### 1.2

a)

```
function [netDropout] = networkDropout(input, target, dropoutRate)
%networkDropout creates and trains a neural network for a regression task
%  Arguments:
%       - input: one-dimensional input values (first data dimension)
%       - target: one-dimensional target values (second data dimension)
%       - dropoutRate: probability for dropping out neurons in the second and third layer
%
%  Returns:
%       - netDropout: the trained network object which can be used directly for prediction
```

```matlab
%
    assert(isrow(input), 'The input values must be passed as a row-vector');
    assert(isrow(target), 'The target values must be passed as a row-vector');
    assert(length(input) == length(target), 'Each input value must have an associated target value');
    assert(isscalar(dropoutRate), 'The dropout rate must be a scalar value');
    assert(dropoutRate >= 0 && dropoutRate <= 1, 'The dropout rate must be in the range [0;1]');

    rng(1337, 'combRecursive');

    % Define options for training
    options = trainingOptions('adam',...
        'MaxEpochs',3000,...
        'Shuffle','never',...
        'L2Regularization',0.0,...
        'InitialLearnRate',0.01,...
        'GradientDecayFactor',0.999,...
        'ValidationPatience',inf,...
        'Plots','training-progress',...
        'Verbose',false);

    layers = [
        sequenceInputLayer(1)
        fullyConnectedLayer(100)
        leakyReluLayer(0.01)
        dropoutLayer(dropoutRate)
        fullyConnectedLayer(100)
        leakyReluLayer(0.01)
        dropoutLayer(dropoutRate)
        fullyConnectedLayer(1)
        regressionLayer()
    ];

    netDropout = trainNetwork(input, target,layers,options);
end
```

b)

```matlab
load('data.mat');
```

```
inputs = data(:,1);
outputs = data(:,2);
```

c)

```
netDropout = networkDropout(transpose(inputs), transpose(
    outputs), 0.1);
netWithoutDropout = networkDropout(transpose(inputs),
    transpose(outputs), 0.0);
```

d)

```
x = -10:0.01:10;
yDropout = predict(netDropout,x);
yWithoutDropout = predict(netWithoutDropout, x);
```

e)

```
figure();
plot(x, yDropout);
xlabel("x");
ylabel("y");
hold on;
plot(x, yWithoutDropout);
scatter(inputs, outputs);
legend("Mit Dropout", "Ohne Dropout", "Datenpunkte");

print("b10a01.eps", "-depsc");
```

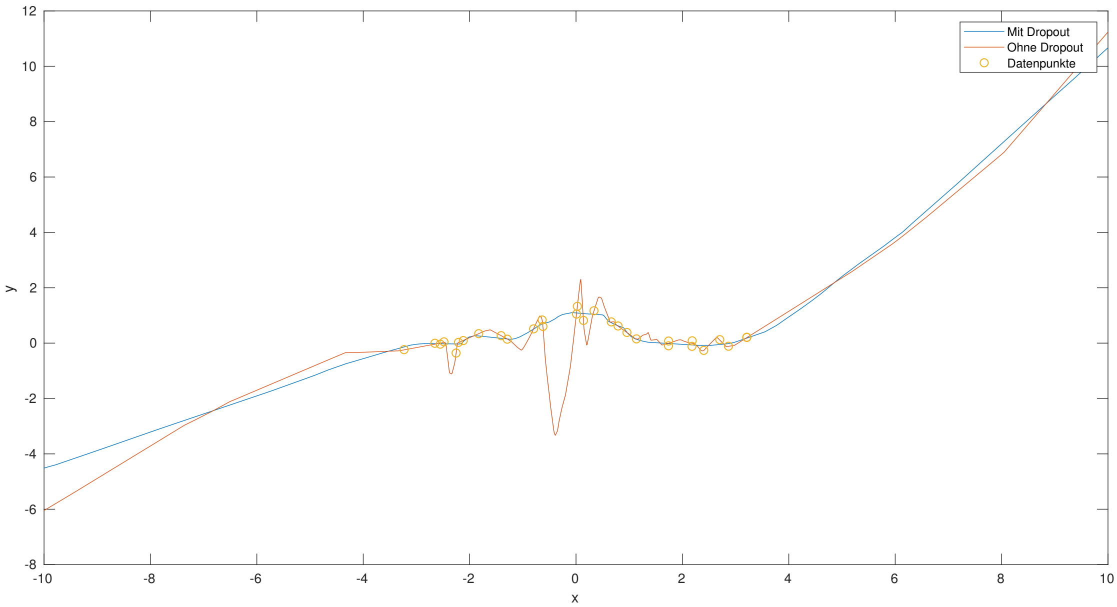


Abbildung 1: Ausgabe des Matlab-Skripts

f)

```
save('b10a01_y', 'yDropout');
save('b10a01_net', 'netDropout');
```

# 2 Uncertainty via Dropout

## 2.1

```
% Initialization
rng(1337, 'combRecursive');

load('data.mat');
load('b10a01_y.mat');
load('b10a01_net.mat');
```

## 2.2

```
x = -10:0.01:10;
W1 = netDropout.Layers(2).Weights;
b1 = netDropout.Layers(2).Bias;
W2 = netDropout.Layers(5).Weights;
b2 = netDropout.Layers(5).Bias;
```

```
W3 = netDropout.Layers(8).Weights;
b3 = netDropout.Layers(8).Bias;

Y = zeros(1000, size(x,2));

for c=1:1000
    Y1 = leakyRelu(W1 * x + b1);
    for i=1:size(Y1,1)
        Y1(i,randperm(size(Y1,2), round(size(Y1,2)*0.1))) = 0;
    end
    Y2 = leakyRelu(W2 * Y1 + b2);
    for i=1:size(Y2,1)
        Y2(i,randperm(size(Y2,2), round(size(Y2,2)*0.1))) = 0;
    end
    Y3 = W3 * Y2 + b3;

    Y(c,:) = Y3;
end
```

## 2.3

```
means = mean(Y);
stds = std(Y);
```

```
function y= leakyRelu(x)
    if x>0
        y = x;
    else
        y = 0.01*x;
    end
end
```

## 2.4

```
plot(x, yDropout);
hold on;
plot(x, means);
plot(x, means+stds);
plot(x, means-stds);
```

### 2.5

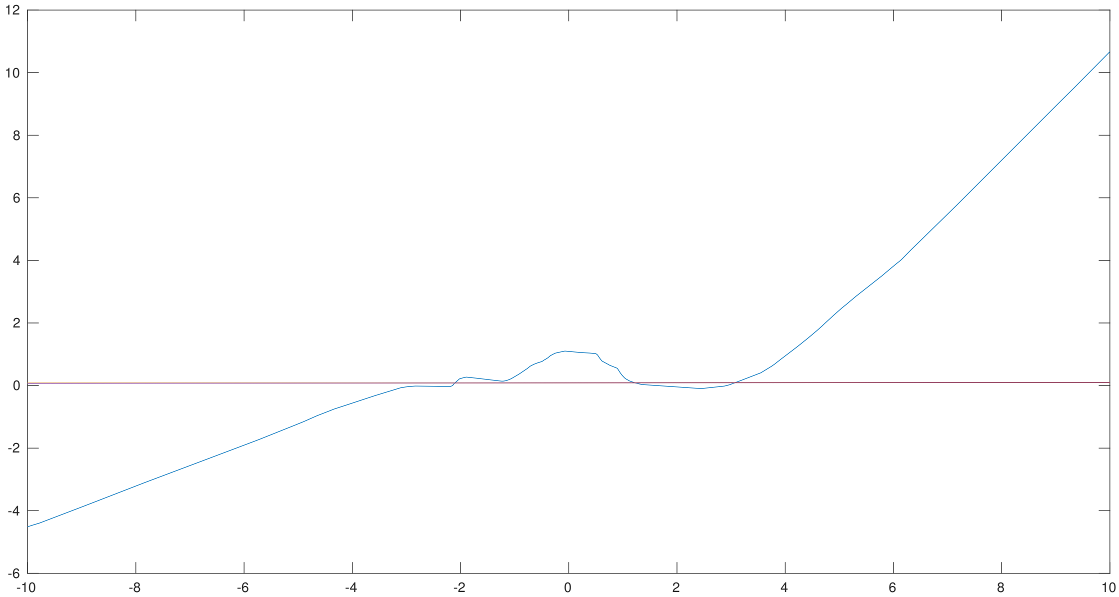


Abbildung 2: Ausgabe des Matlab Scripts

## 3 Radiale Basisfunktionen

### 3.1

$$H = \begin{pmatrix} h(0) & h(1) & h(2) \\ h(1) & h(0) & h(1) \\ 4.001 & h(1) & h(0) \end{pmatrix} = \begin{pmatrix} 0.001 & 1.001 & 4.001 \\ 1.001 & 0.001 & 1.001 \\ 4.001 & 1.001 & 0.001 \end{pmatrix}$$

### 3.2

$$\begin{pmatrix} 0.001 & 1.001 & 4.001 \\ 1.001 & 0.001 & 1.001 \\ 4.001 & 1.001 & 0.001 \end{pmatrix} \begin{pmatrix} 1.752 \\ -8.004 \\ w_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 4 \\ -1 \end{pmatrix}$$

$$\Rightarrow w_3 = 2.252$$

### 3.3

a)

$$g(3) = 1.752 \cdot h(3) - 8.004 \cdot h(2) + 2.252 \cdot h(1) = -14$$

b) Ja, da der Datenpunkt bei $x = 3$ nicht bekannt ist.

## 3.4

$\alpha = -1.0 \Leftrightarrow$ Grafik B. $h$ steigt Quadratisch $\Rightarrow$ Parabelförmig.

$\alpha = -0.3 \Leftrightarrow$ Grafik C. Da $h(0) << 1$.

$\alpha = 0.15 \Leftrightarrow$ Grafik A. Da $h(0) >> 1$.