

# Einführung in die Neuroinformatik

Tim Luchterhand, Paul Nykiel (Gruppe P)

20. Juni 2018

## 1 Learning Slowdown

### 1.1

(a)

$$\begin{aligned}w_2 &= 0.6 \\b_2 &= 0.9 \\T &= 0 \\ \Rightarrow \frac{\partial E}{\partial b_2}(w, b) &= -2 \left( 0 - \frac{1}{1 + e^{-1.5}} \right) \frac{1}{1 + e^{-1.5}} \left( 1 - \frac{1}{1 + e^{-1.5}} \right) \\ &\approx 0.24\end{aligned}$$

(b)

$$\begin{aligned}w_2 &= 2 \\b_2 &= 2 \\T &= 0 \\ \Rightarrow \frac{\partial E}{\partial b_2}(w, b) &= -2 \left( 0 - \frac{1}{1 + e^{-4}} \right) \frac{1}{1 + e^{-4}} \left( 1 - \frac{1}{1 + e^{-4}} \right) \\ &\approx 0.035\end{aligned}$$

- (c) Der Gradient ist sehr klein, wodurch jede Epoche nur minimalen Lernfortschritt bringt. Problematisch ist hierbei die Ableitung der Sigmoid-Funktion, da  $f'(x) \in [0, \frac{1}{4}] \quad \forall x \in \mathbb{R}$ , und verschwindet für betragsmäßig große  $x$ .

## 1.2

(a)

$$\begin{aligned}
 \frac{\partial E}{\partial b_1} &= 2(T - y_2) \cdot (-1) \cdot \frac{\partial y_2}{\partial b_1} \\
 &= -2(T - y_2) \cdot \frac{\partial}{\partial b_1} f(w_2 \cdot f(w_1 \cdot x + b_1) + b_2) \\
 &= -2(T - y_2) \cdot f'(w_2 \cdot f(w_1 \cdot x + b_1) + b_2) \cdot w_2 \cdot f'(w_1 \cdot x + b_1) \\
 &= -2(T - y_2) \cdot f'(u_2) \cdot w_2 \cdot f'(u_1)
 \end{aligned}$$

Pro weiterer Netzwerkschicht kommen weitere  $f'$ -Terme hinzu die wiederum  $\in [0, \frac{1}{4}]$  sind. Dadurch wird der Gradient tendentiell noch kleiner und das Lernen verlangsamt sich zusätzlich zu Neuronen näher zur Eingangsschicht.

- (b) Wie in (a) bereits erläutert, verschlimmern weitere Schichten das Problem.
- (c) In jeder Epoche werden die Gewichte und der Bias nur marginal angepasst wodurch der Gradientenabstieg sehr langsam erfolgt.

## 1.3

(a)

$$\begin{aligned}
 x &= 1 \\
 T &= 0 \\
 w_1 &= 100 \\
 b_1 &= -100 \\
 w_2 &= 100 \\
 b_2 &= -50 \\
 u_1 &= 100 - 100 = 0 \\
 y_1 &= f(u_1) = \frac{1}{2} \\
 u_2 &= 100 \cdot y_1 - 50 = 0 \\
 y_2 &= f(u_2) = \frac{1}{2} \\
 \frac{\partial E}{\partial b_2} &= 2 \cdot y_2 \cdot f'(u_2) = \frac{1}{4} \\
 \frac{\partial E}{\partial b_1} &= \frac{\partial E}{\partial b_2} \cdot 25 = \frac{25}{4}
 \end{aligned}$$

- (b) Der Gradient der aktuellen Schicht setzt sich aus dem Gradienten der nachfolgenden Schicht multipliziert mit einem Faktor  $\gg 1$ . Dadurch werden die Gradienten in den Schichten nah am Eingang extrem groß.

- (c) Ein zu großer Gradient kann dazu führen, dass Minima übersprungen werden wodurch der Lernprozess oszilliert und sich das Lernen in die Länge zieht.

## 1.4

Bei der Cross-Correlation-Funktion tritt immer ein  $f'$ -Term weniger auf als bei der Sigmoid-Funktion. Das Problem der zu kleinen Gradienten fällt deswegen weniger stark ins Gewicht.

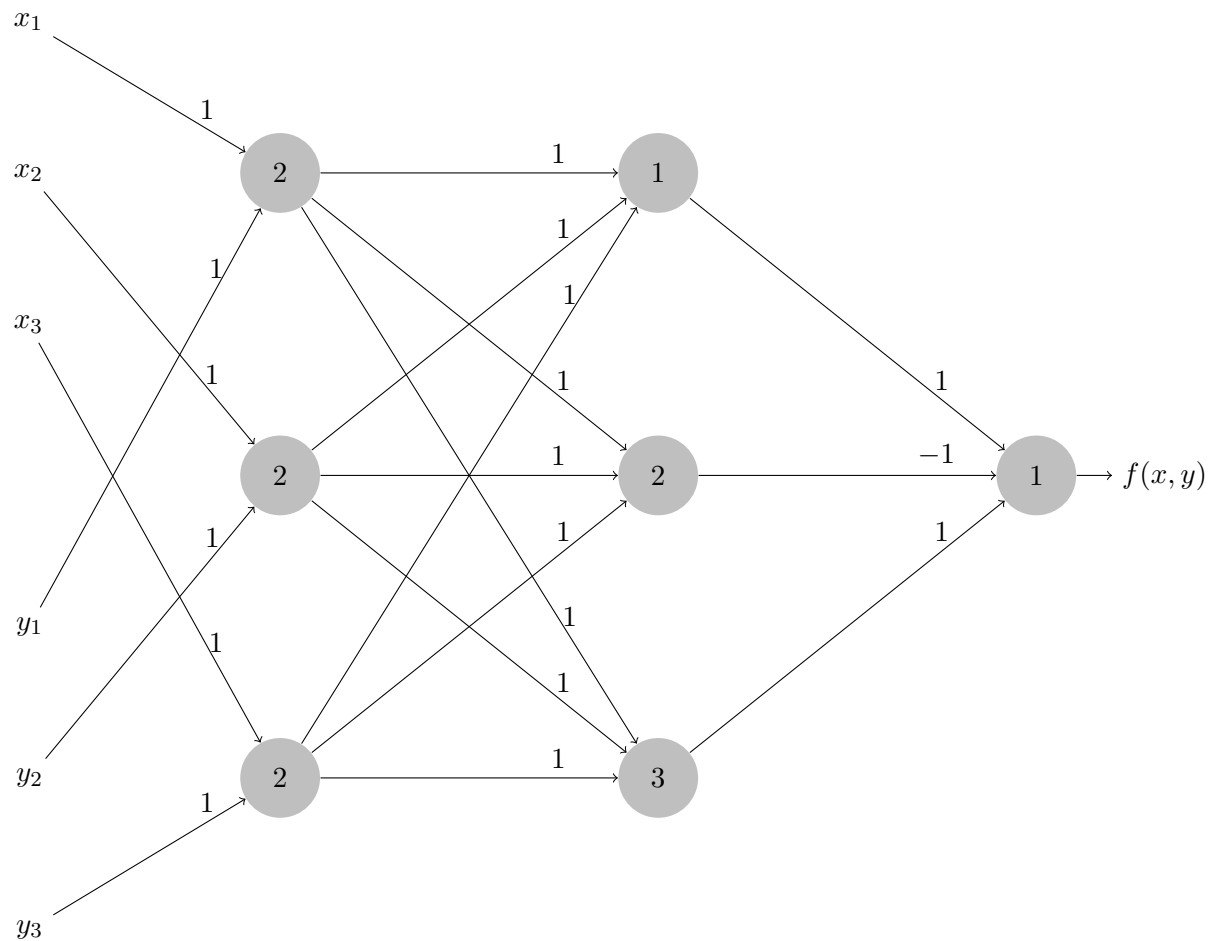
## 2 Flat vs. Deep Networks

### 2.1

- (a)

$$\begin{aligned}
 x &= (0, 0) \\
 y &= (1, 1) \\
 y_1^{(1)} &= 0 \\
 y_2^{(1)} &= 0 \\
 y_1^{(2)} &= 0 \\
 y_2^{(2)} &= 0 \\
 f(x, y) &= 0 \\
 \langle x, y \rangle_2 &= (0 \ 0) \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix} \mod 2 = 0
 \end{aligned}$$

- (b) Die Neuronen in der ersten Schicht fungieren als AND-Gatter und berechnen das Skalarprodukt. Die Neuronen in der zweiten Schicht bilden zusammen ein XOR-Gatter und führen die Modulo-Operation durch.
- (c)



## 2.2

(a) Für das Gradientenabstiegsverfahren sind differenzierbare Funktionen  $f$  notwendig, die Sprunfunktion ist nicht differenzierbar.

(b)

```

1 % Aufgabe b
2 % Generate input data
3 n = 4;
4 nInput = 2 * n;
5 combinations = de2bi(0:2^nInput-1)';
6
7 % Each column of both vectors represents an input sequence
  (this is contrary to the usual convention but Matlabs
  implementation of neural networks expects the input in
  this format)

```

```

8 X = combinations(1:n, :);
9 Y = combinations(n+1:end, :);
10
11 T = zeros(1, size(X,2));
12 for c = 1:size(X,2)
13     x = X(:,c);
14     y = Y(:,c);
15     currentT = mod(transpose(x) * y, 2);
16     T(:,c) = currentT;
17 end

```

(c)

```

1 function [bestError] = trainNetworks(combinations, targets,
    hidden)
2 %trainNetworks trains the network multiple times and
    returns the best result
3 % Arguments:
4 %     - combinations: input data organized as features x
    observations (each column defines a data point). This is
    the required format for Matlabs train() function
5 %     - targets: label for each data point (as defined by
    the binary dot product function)
6 %     - hidden: number of hidden neurons to use per layer
    (as vector if multiple hidden layers should be used)
7 %
8 % Return value:
9 %     - bestError: the value for the best error (MSE)
    over all trained networks
10 %
11     assert(isrow(targets), 'targets should be a row vector'
    );
12     assert(isscalar(hidden) || isrow(hidden), 'hidden
    should either be a scalar value (number of hidden
    neurons in the flat hierarchy) or a vector defining
    the number of hidden neurons per layer');
13
14     rng(1337, 'combRecursive');
15
16     minError = 100000;
17     for c = 1:20
18         net = fitnet(hidden);
19         net.divideParam.trainRatio = 1;
20         net.divideParam.valRatio = 0;

```

```

21         net.divideParam.testRatio = 0;
22         net.trainParam.epochs = 300;
23         net = train(net, combinations, targets);
24
25         error = perform(net, net(combinations), targets);
26         if error < minError
27             minError = error;
28         end
29     end
30
31     bestError = minError;
32 end

```

(d)

```

18
19 % Aufgabe c
20 fhidden = 1:2^n+5;
21 dhidden = 1:n+5;
22
23 ferrors = zeros(size(fhidden));
24 derrors = zeros(size(dhidden));
25
26 parfor h = fhidden
27     ferrors(h) = trainNetworks(combinations, T, h);
28 end
29
30 parfor h = dhidden
31     derrors(h) = trainNetworks(combinations, T, [h h]);
32 end
33
34 figure();
35 subplot(2,1,1);
36 plot(fhidden, ferrors);
37 title('Fehlerfunktion Flat-Netzwerk');
38 xlabel('Anzahl der Neuronen');
39 ylabel('Minimaler Fehler');
40
41 subplot(2,1,2);
42 plot(dhidden, derrors);
43 title('Fehlerfunktion Deep-Netzwerk');
44 xlabel('Anzahl der Neuronen');
45 ylabel('Minimaler Fehler');
46

```

```
47 print( 'b07a02.eps', '-depsc' );
```

(e)

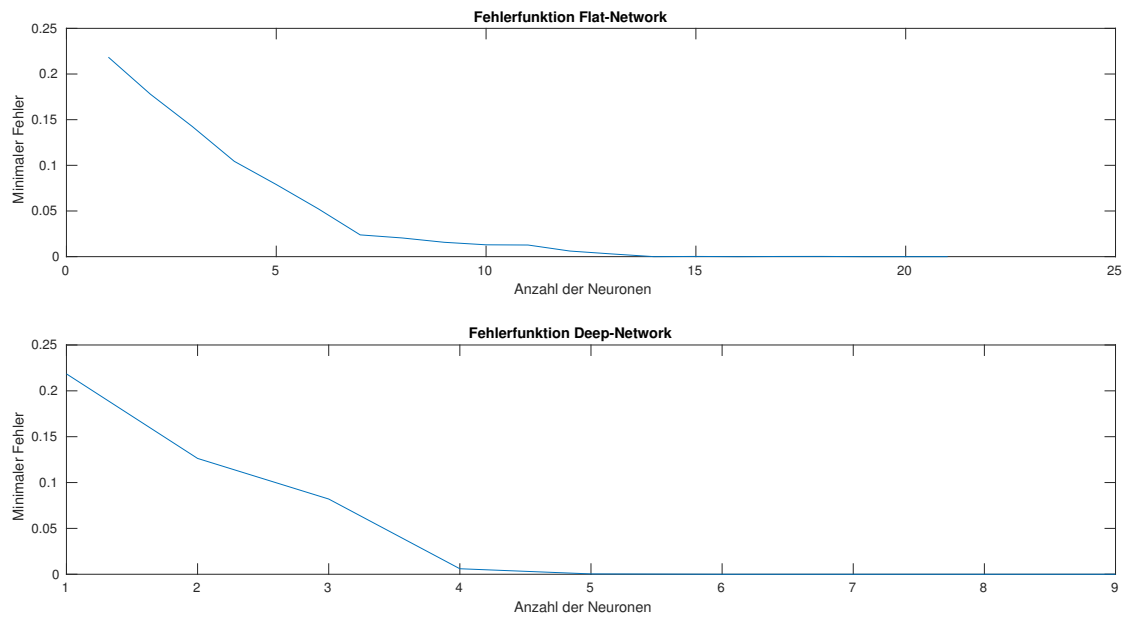


Abbildung 1: Verlauf der Fehlerfunktion