Prof. Dr. Carsten Meyer
Faculty of Engineering
Christian-Albrechts-Universität Kiel

Neural Networks and Deep Learning – Summer Term 2018

Exam

1.) Neural networks and deep learning: General concepts

a) For each of the following definitions, find the corresponding technical term.
(For each correct element: 2 points, max: 12 points)

Description	*Technical term*
Set of presynaptic neurons (or inputs) which affect the activity of a considered neuron of a later layer	*Receptive field*
Result of applying a particular filter (kernel) to an input in a convolutional neural network	*Feature map*
Phenomenon in learning where the network learns details of some training patterns which are not relevant for most other patterns	*Overfitting*
Restricting the number of degrees of freedom of a model	*Regularisation*
Transforming each hidden layer activations to have zero mean and unit variance, based on the current mini-batch	*Batch normalization*
Method to stop an iterative training process when the validation error starts to increase	*Early stopping*

b) Insert the missing terms into the following text.
(For each correct element: 1.5 point, max: 6 points)

The [Action potential ✓] is the basic event in neural communication; it is generated if the integrated incoming signal exceeds a threshold. Immediately after generation of such an event, during the [absolute Refractory period] ✓, the neuron cannot generate a second event. After that period, the generation of a second event is inhibited, but not impossible; this period is called the [relative refactory period ✓].

If such an event arrives at the end of the nerve fiber, a chemical substance is released which propagates to the postsynaptic membrane of another neuron; those chemical substances are called [neurotransmitters] ✓.

c) Provide the names of three different differentiable activation functions – which is suitable for the mentioned learning problem – and the name of an appropriate loss function.
(For each correct term 1.5 points, max. 6 points)

Learning problem	Activation function	Loss function
Regression problem (1-dim.)	linear $f(z) = z$	L_{MSE} ✓
Two-class classification problem	sigmoid $f(z) = \frac{1}{1+e^{-z}}$	L_{CE}
Multi-class classification problem	soft max $f(z_i) = e^{z_i} / \sum_k e^{z_k}$	L_{LL} ✓

d) For each question, mark all answers which are correct. Note: There is no limit on the number of correct answers per question.
(1 point for each correct mark, 0.5 points subtracted for any incorrect mark)

i) The following network types are appropriate to handle a one-dimensional regression problem with positive and negative target values:

(1) multi-layer perceptron,
(2) denoising autoencoder,
(3) convolutional neural network with a ReLU activation function in the last layer,
(4) radial basis function network with Gaussian basis functions.

Answer: 3

f

ii) Training of a radial basis function network generally involves

(1) Hebbian learning to compute the number of basis functions,
(2) supervised training of the weights from the hidden to the output layer (with fixed basis functions),
(3) unsupervised training of the basis functions (i.e. of the weights from input to hidden layer),
(4) Hebbian learning of the basis function centers.

Answer: 4, 3

f ✓

iii) The backpropagation algorithm refers to computing

(1) the number of hidden layers,
(2) the number of hidden units,
(3) the synaptic weights and thresholds,
(4) the network outputs.

Answer: 3

✓

iv) The following network types are trained by an unsupervised algorithm:

(1) multi-layer perceptron,
(2) long short term memory,
(3) autoencoder,
(4) fully convolutional network.

Answer: 2, 3

f ✓

v) The backpropagation algorithm (or a variant thereof) is a suitable learning algorithm for the following network types:

(1) multi-layer perceptron,
(2) Hopfield network,
(3) self-organizing map,
(4) convolutional neural network.

Answer: 1

2.) Perceptron learning

O2 C108

Ex. 1:17 5123

a) Consider a binary perceptron with output value $y \in \{0;\ 1\}$, i.e. the function f is the Heaviside function $f(h) = \Theta[h]$. Assume that the initial perceptron parameters are $w_1 = -0.3$, $w_2 = 0.1$, $\theta = 0.4$ and $w_0 = -\theta = -0.4$. (The input component x_0 is always 1.)
The perceptron shall be trained to correctly classify training patterns. Specifically, two training patterns are presented to the perceptron: $\mathbf{x}^{(1)} = (x_1^{(1)}, x_2^{(1)}) = (1,0)$ and $\mathbf{x}^{(2)} = (x_1^{(2)}, x_2^{(2)}) = (0,1)$. After presentation of each training pattern, the weights shall be updated using a learning rate $\eta = 0.25$.

i) Consider the first training pattern $\mathbf{x}^{(1)} = (x_1^{(1)}, x_2^{(1)}) = (1,0)$, for which the target output shall be 0. Calculate the perceptron output $y^{(1)}$ for the first training pattern as well as the new perceptron weights (using the appropriate training algorithm) and insert the found values into the following table (fields marked with "…").

$1 \times -.4 + -.3 + 0$
$= -.7$

μ=1; first training pattern					
Input $\mathbf{x}^{(1)}$	Current weights $\mathbf{w}(t=0)$	Network Output $y^{(1)}$	Target output $d^{(1)}$	Learning rate η	New Weights $\mathbf{w}(t=1)$
$x_0 = 1$	$w_0 = -0.4$	… 0	0	0.25	$w_0 =$ … $-.4$
$x_1 = 1$	$w_1 = -0.3$				$w_1 =$ … $-.3$
$x_2 = 0$	$w_2 = 0.1$				$w_2 =$ … $.1$

the weights will not change because Network output = target output

(For each correct item at fields marked with "…": 1 point, max: 4 points)

4/4

ii) Now consider the second training pattern $\mathbf{x}^{(2)} = (x_1^{(2)}, x_2^{(2)}) = (0,1)$, for which the target output shall be 1. Insert the weights from i) into the field marked with "(*)" and use those weights to calculate the network output $y^{(2)}$ for the second training pattern. Again calculate the new weights (using the training pattern) and insert the found values into the following table (fields marked with "…").

$1 \times .4 + .1$
$= -.3 = \boxed{0}$

μ=2; second training pattern					
Input $\mathbf{x}^{(2)}$	Current weights $\mathbf{w}(t=1)$	Network Output $y^{(2)}$	Target output $d^{(2)}$	Learning rate η	New Weights $\mathbf{w}(t=2)$
$x_0 = 1$	$w_0 =$ (*) $-.4$	… 0	1	0.25	$w_0 =$ … $-.15$
$x_1 = 0$	$w_1 =$ (*) $-.3$				$w_1 =$ … $-.3$
$x_2 = 1$	$w_2 =$ (*) $.1$				$w_2 = .35$

4/4

(For each correct item at fields marked with "…": 1 point, max: 4 points)

$w(t+1) = w(t) + \eta(d - y) \cdot x$

$-.4 + .25 \cdot 1 = -.15$

$= -.3 + .25(1-0) \cdot 0 = -.3$

$.1 + .25 \cdot .1 = .35$

Scanned with CamScanner

iii) Using the resulting perceptron weights calculated in ii), compute the network output for the two input patterns $\mathbf{x}^{(1)} = (x_1^{(1)}, x_2^{(1)}) = (1,0)$ and $\mathbf{x}^{(2)} = (x_1^{(2)}, x_2^{(2)}) = (0,1)$.

Input $\mathbf{x}^{(1)} = (x_1^{(1)}, x_2^{(1)}) = (1,0)$ ⇒ perceptron output $y =$ 0 -.15*1 + .3*1 + .35*1

Input $\mathbf{x}^{(2)} = (x_1^{(2)}, x_2^{(2)}) = (0,1)$ ⇒ perceptron output $y =$ 1 -.15*1 - .3*0 + .35*1

(For each correct network output: 1 point, max: 2 points)

2/

iv) How is this training process called

- with respect to whether the target output is used during training or not (1 point):

  supervised ~~and unsupervised~~

- with respect to the number of training patterns used for weight update (1 point):

  online learning ~~offline learning (Batch)~~

1/2

(For each correct item: 1 point, max: 2 points)

v) Now assume a *linear* function f with slope 2 instead of the Heaviside function: $f(h) = 2h$. Using the resulting perceptron weights (calculated in ii), compute the network output for the two input patterns $\mathbf{x}^{(1)} = (x_1^{(1)}, x_2^{(1)}) = (1,0)$ and $\mathbf{x}^{(2)} = (x_1^{(2)}, x_2^{(2)}) = (0,1)$.

Input $\mathbf{x}^{(1)} = (x_1^{(1)}, x_2^{(1)}) = (1,0)$ ⇒ perceptron output $y =$ -.9 -.15 - .3 = (-.45)x2 ✓

Input $\mathbf{x}^{(2)} = (x_1^{(2)}, x_2^{(2)}) = (0,1)$ ⇒ perceptron output $y =$ +.40 -.15 + .35 = (.20) x2 ✓

(For each correct network output: 1 point, max: 2 points)

2/2

b) Indicate (by checking the appropriate column) whether the following statements are true or false: (For each correct answer: 1 point, for each incorrect answer: 0.5 points subtracted, min.: 0 points, max: 8 points)

Statement	true	false	
A "flat" neuron activation function or error function may pose problems to neural network training since the resulting weight modification may be small.	X		✓
The stochastic gradient descent learning algorithm always finds the global minimum of the loss function.			
Using weight regularization in stochastic gradient descent (e.g. L1 or L2) guarantees to find network weights yielding a low generalization error (e.g., smaller than 0.05).			
Using a ReLU activation function may help to reduce the effect of the vanishing gradient problem.			
The size of the mini-batch in stochastic gradient descent may influence the convergence of the learning algorithm (i.e., whether and how fast it converges).	X		✓
The initial values of the weights and bias parameters may have an influence on the result of the learning algorithm.			
Second order learning methods are especially suited for non-convex loss functions and only few training data.			
Learning with momentum may help to stabilize learning (by reducing the effect of parameter oscillations)			

2/8

c) Indicate (by checking the appropriate column) whether applying the following method may help to reduce the *test error* of a neural network (on a very large test corpus; mark "yes") or not ("no"). (For each correct answer: 1 point, for each incorrect answer: 0.5 points subtracted, min.: 0 points, max: 6 points)

Method (does applying this method may reduce the test error?)	Yes	No
Data augmentation	X	
Normalizing the input data only on the training data (bot not on the test data)		X
Weight regularization	X	
Reducing the size of the training set		X
Applying dropout to fully connected layers	X	
Initializing weights and biases to the same value for all neurons of a given layer		X

6/6

d) Name three extensions of the standard stochastic gradient descent learning algorithm. (2 points for each correct answer; max. 6 points)

~~online gradient learning $w(t+1) = w(t) + \eta \cdot (\hat{y}^M - \hat{y}(w - \hat{x}^M)) \cdot f(\hat{z}^M) \cdot \hat{x}^M$~~

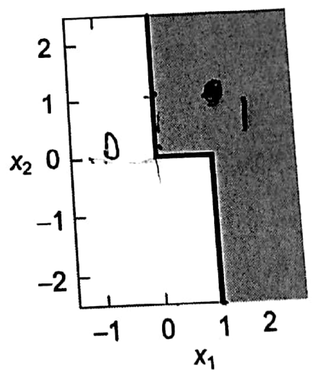
~~batch learning $w(t+1) = w(t) - \eta \cdot \nabla_w L_{mse}(w)$~~ 0/6

~~regularization by adding R(f) to loss function~~ Ex2: 27/34

by L_1 Regularization or L_2 Regularization

3.) Multi-layer perceptron: Determination of weights

Consider a multi-layer perceptron with two inputs, one hidden layer and a single output unit. The values at the inputs are assumed to be real. The activation function of the output unit is the Heaviside function. Thus, the multi-layer perceptron classifies every two-dimensional input as either being 1 or 0. The following figure shows the separation line between the two classes (the dark area should be classified as 1, the light area as 0; both areas extend to infinity to the left and right, respectively).



a) What is the minimum number of hidden units needed to realize the given decision boundary? (2 points)

Answer: 1

7/02

b) What weights and thresholds should be used for the hidden units in order to implement the given classification problem? (Give all weight and threshold parameters for all hidden units assuming that the activation function of the hidden units is the Heaviside function; note that the dark area corresponds to output 1 and the light area to output 0 – and not vice versa!) (max.: 9 points)

$y = 1$

$x_1 > 1$

or ($x_1 > 0$ and $x_2 > 0$)

$y = 0$

$x_1 < 0$ or $x_1 < 1$ and $x_2 < 0$

x_1 x_2 y

x_1	x_2	y
0	0	
1	0	
		1

00/08

c) **What weights and threshold should be used for the output unit to implement the classification problem?** **(max.: 8 points)**

00/08

d) A (fully connected) multi-layer feedforward network network has 4 input units, one hidden layer with 3 units, and 2 output units. How many weights (excluding threshold / bias) does this network have? Indicate how you calculate this number and the final result. (max: 3 points)

Answer: 4 * 3 * 2 = 24

\# of inputs * # of outputs * # of hidden layers

Ex 3: 00/22

4.) Convolutional neural networks

a) Explain three key differences between a convolutional neural network (CNN) and a multi-layer perceptron (MLP).
(2 points for each correct answer; max. 6 points)

CNN has kernel and receptive field but in MLP normal inputs connected

CNN does convolutional layer ~~and~~ and also pooling layer but MLP doesn't have

02/06

b) Consider a CNN with the following architecture:

INPUT → Conv → ReLU → MaxPool ,

where "Conv" denotes a convolutional layer with the filter (kernel) given below, stride 1 and no padding, "ReLU" denotes the rectified linear unit activation and "MaxPool" denotes a pooling layer with 2 × 2 max pooling, stride 2 and no padding.

Given is the following input and the following filter (kernel) function:

S=1 P=0

Input:

2	0	3	1	–2
3	1	0	–1	4
–2	–3	1	2	0
0	1	–2	0	–3
1	–4	0	3	2

Filter (kernel):

1	2
0	–1

input (5 x 5)
filter (2 x 2)

pooling
layer
x2
=2
P=0

Further assume that all biases are set to 0 for this exercise. Calculate the output of the CNN (including *all* intermediate stages). (max.: ~~28~~ points)
14

output size $N - F + 2P / S + 1$

$5 - 2 + 0 / 2$

$3/2$

00/14

9

c) Consider a CNN with the following architecture:

INPUT → Conv1 → Pool → Conv2

N - f + 2P / S+1

32 - 5 + 0 / 2

and the following specifications:

Layer	Specification	Dimension (width × height × depth)
INPUT	32 × 32 gray scale image	32 × 32 × 1
Conv1	5 × 5 filter, no padding, stride 1, 6 feature maps	
Pool	2 × 2 filter, no padding, stride 2, max pooling	
Conv2	3 × 3 filter, no padding, stride 1, 10 feature maps	
Conv3	1 × 1 filter, no padding, stride 1, 2 feature maps	

For each layer, insert the dimension of the *output* of that layer in the form (width × height × depth) into the table above. Then, calculate the number of trainable parameters (synaptic weights *plus* biases) involved in each layer and insert the corresponding equation and the result into the table below: (max. 14 points)

(size of kernal + padding) × number of feature map

Layer	Number of trainable parameters (calculation)	Trainable parameters (result)
Conv1	5×5 × 6 f	150 f
Pool	no trainable parameters in pool layer	0 ✓
Conv2	3×3×10 = 90 f	90 f
Conv3	1×1×2 f	2 f

0

d) Name 4 well-known CNN architectures (which were presented in the lecture) and order them with respect to the number of layers (which roughly corresponds to the time of invention), i.e. from small to large number of layers. For each CNN, name a characteristic feature of the CNN (which has been implemented by this CNN *for the first time*), using the following features: residual connections / 3 × 3 filters / inception module / 1 × 1 convolutions. (max. 8 points)

CNN (from small to high number of layers)	Feature (used by the CNN *for the first time*)
ALex ✓	for images
VGG ✓	small filter and more layers
Rasnet ✓	no drop out on fully connected layer

0

5.) Applications and properties of neural networks

a) For each of the applications mentioned below, name a neural network type which is suitable for this application (all network types must be *different*!). For your neural network type, also mention the corresponding learning mode.

(max. 14 points)

Application	Neural network type	Learning mode
Regression (low-dim., given features)	single layer perceptron ✓	supervised ✓
Image classification (many pixels, many classes)	CNN ✓	Supervised learning ✓
Function approximation (sum of Gaussians)	Radial function ✓	—
Time series prediction (variable length)	MLP f	Supervised learning (✓)
Associative memory	Radial ~~basis~~ function f	—
Feature extraction	Recurrent neural network f	supervised (✓)
Clustering	—	unsupervised learning (✓)

8

b) Indicate (by checking the appropriate column) whether the following statements are true or false: (For each correct answer: 1 point, for each incorrect answer: 0.5 points subtracted, min.: 0 points, max: 10 points)

Statement	true	false	
A radial basis function network is an example of a feedforward network.		X	f
A competitive network with 5 output neurons can be used to divide the input space into 5 classes.	X		✓
A self-organizing map provides a topology-preserving map from an input space to a (generally lower-dimensional) output space.			
After training a self-organizing map, output neurons that win for similar inputs are usually far apart from each other in the map.		X	✓
A Hopfield network can be trained with the Hebbian learning rule.			
A Hopfield network may approach a stationary state.	X		✓
In a recurrent neural network in asynchronous dynamics more than one neuron is updated at each time step.		X	✓
The type of network dynamics in a fully connected recurrent neural network may have an influence on the resulting asymptotic network behavior.			
In a long short term memory (LSTM) unit the cell state is constructed such that the vanishing / exploding gradient problem is alleviated compared to a standard RNN.	X		✓
When applying the backpropagation through time learning rule to a (layered) recurrent neural network, the length of the sequence is completely irrelevant.	X		f

c) Imagine you want to apply a layered recurrent neural network (consisting of an input layer, two hidden layers and an output layer) to the task of classifying a series of video frames (corresponding to about a minute of data, at a frame rate of 60 frames per second) to a sentiment. What would be an appropriate learning algorithm for this task (give the *complete* name)? (max. 3 points)

0/

Answer:

Ex5: 12/