LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

CENTRUM FÜR INFORMATIONS- UND SPRACHVERARBEITUNG
STUDIENGANG COMPUTERLINGUISTIK

PROBEKLAUSUR ZUM MASTERMODUL „PROFILIERUNGSMODUL II“ WS 2019/2020,

BENJAMIN ROTH, NINA POERNER

KLAUSUR AM

VORNAME:	
NACHNAME:	
MATRIKELNUMMER:	
STUDIENGANG:	☐ M.Sc. Computerlinguistik, ☐ M.Sc. Informatik, ☐ Magister
	☐ anderer:

Die Klausur besteht aus **9 Aufgaben**. Die Punktzahl ist bei jeder Aufgabe angegeben. Die Bearbeitungsdauer beträgt **90 Minuten**. Bitte überprüfen Sie, ob Sie ein vollständiges Exemplar erhalten haben.
Tragen Sie die Lösungen in den dafür vorgesehenen Raum im Anschluss an jede Aufgabe ein. Falls der Platz für Ihre Lösung nicht ausreicht, benutzen Sie bitte **nur** die ausgeteilten Zusatzblätter!
Verwenden Sie einen dokumentenechten Kugelschreiber oder Füller, **keine** Bleistifte. Es sind **keine Hilfsmittel** zugelassen. Geben Sie Programmcode immer in **Python** an. **Sie können Fragen auf Englisch bearbeiten**. Bitte tragen Sie **zuerst**, d.h., bevor Sie die Aufgaben lösen, auf **allen** Seiten Ihren Namen ein und füllen Sie die Titelseite aus.

Aufgabe	mögliche Punkte	erreichte Punkte
1. Gradient Updates	6	
2. Softmax	4	
3. Function Graphs	6	
4. LSTM	4	
5. CNN	4	
6. (Self-)Attention	7	
7. NumPy	3	
8. Keras	5	
9. PyTorch	5	
Summe	44	
Note		

NAME:

Aufgabe 1 Gradient Updates

(a) Provide the formula for one gradient descent update of the parameters of a loss function. Here, you can refer to the gradient of the loss function by $\nabla_{\theta_t}\mathcal{L}(\theta_t)$, where θ_t are the parameters at time t (before the update).

(b) How does the update function change, if l2-regularization is added? (Provide the formula.)

(c) In Pytorch, a gradient update for a `model` with corresponding `loss` can be done with the following lines:

```
loss.backward()
for w in lr_model.parameters():
  w.sub_(w.grad * 0.0001)
```

Extend the code to include l2-regularization.

6 PUNKTE

NAME:

Aufgabe 2 Softmax

Let $\vec{z}$ be a vector that contains a constant c added to the log-probabilities of a categorical distribution stored in a vector $\vec{p}$.
Formally: $\vec{z}_i = c + \vec{y}_i$, i.e., $\vec{z} = [c + \log p_1, c + \log p_2, \cdots, c + \log p_n]^T$.
What is the result of softmax($\vec{z}$)? (Show why.) 4 PUNKTE

Aufgabe 3 Function Graphs

Draw the network graph for the following formula:

$$\mathcal{L}(\vec{w}, W, \vec{x}, y) = (y - \vec{w}^T max(0, W\vec{x}))^2$$

where $y \in \mathbb{R}$, $\vec{w} \in \mathbb{R}^k$, $W \in \mathbb{R}^{k \times n}$, $\vec{x} \in \mathbb{R}^n$

For each node (representing an intermediate result), indicate what function was applied, and what is the resulting shape. 6 PUNKTE

Aufgabe 4 LSTM

The following equations define an LSTM architecture. We have given unusual variable names to its components. Below, indicate which variable corresponds to which component (by putting an "x" in the correct fields of the table).

$$\vec{h}_0 = 0 \tag{1}$$
$$\vec{c}_0 = 0 \tag{2}$$
$$\vec{\alpha}_t = \sigma(\mathbf{W}_\alpha \vec{h}_{t-1} + \mathbf{V}_\alpha \vec{x}_t + \vec{b}_\alpha) \tag{3}$$
$$\vec{\beta}_t = \sigma(\mathbf{W}_\beta \vec{h}_{t-1} + \mathbf{V}_\beta \vec{x}_t + \vec{b}_\beta) \tag{4}$$
$$\vec{\gamma}_t = \sigma(\mathbf{W}_\gamma \vec{h}_{t-1} + \mathbf{V}_\gamma \vec{x}_t + \vec{b}_\gamma) \tag{5}$$
$$\vec{\delta}_t = \tanh(\mathbf{W}_\delta \vec{h}_{t-1} + \mathbf{V}_\delta \vec{x}_t + \vec{b}_\delta) \tag{6}$$
$$\vec{c}_t = \vec{\alpha}_t \circ c_{t-1} + \vec{\beta}_t \circ \delta_t \tag{7}$$
$$\vec{h}_t = \vec{\gamma}_t \circ \tanh(\vec{c}_t) \tag{8}$$

	input gate	output gate	forget gate	candidate state	cell state	hidden state
$\vec{h}$						x
$\vec{c}$					x	
$\vec{\alpha}$						
$\vec{\beta}$						
$\vec{\gamma}$						
$\vec{\delta}$						

4 PUNKTE

Aufgabe 5 CNN

(a) Let $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3$ be CNN layers. Their parameters (their trainable filter banks) are $\mathbf{F}_1 \in \mathbb{R}^{K_1 \times H \times N_1}$, $\mathbf{F}_2 \in \mathbb{R}^{K_2 \times N_1 \times N_2}$, $\mathbf{F}_3 \in \mathbb{R}^{K_3 \times N_2 \times N_3}$, where K_1, K_2, K_3 are filter widths and N_1, N_2, N_3 are the number of filters. The input is a matrix $\mathbf{X} \in \mathbb{R}^{T \times H}$ (e.g., a sentence of length T represented by H-dimensional word vectors). What is the shape of $\mathbf{Y} = \mathcal{F}_3(\mathcal{F}_2(\mathcal{F}_1(\mathbf{X})))$? (Assume that we do not do padding before applying the CNN.)

4 PUNKTE

NAME:

Aufgabe 6 (Self-)Attention

(a) Let $\mathbf{X} \in \mathbb{R}^{T \times H}$ be a sequence of word vectors (e.g., a sentence of length T represented by H-dimensional word2vec vectors). Let $\mathbf{W}^{(q)} \in \mathbb{R}^{H \times D}, \mathbf{W}^{(k)} \in \mathbb{R}^{H \times D}, \mathbf{W}^{(v)} \in \mathbb{R}^{H \times D}$ be trainable linear transformations into the query, key and value space. Use the scaled dot-product self-attention formula to define a new representation $\mathbf{O} \in \mathbb{R}^{T \times D}$. You may use $\mathrm{softmax}(\ldots)$ as a shorthand for "softmax with normalization over the last dimension of a matrix."

(b) The self-attention network that you have defined cannot distinguish between different word orderings (e.g., "cat eat mouse" vs. "mouse eat cat"). Name and describe a possible solution to this problem.

7 PUNKTE

Aufgabe 7 NumPy

Determine the values of the `out` variable for every case:

(a)
```
x = np.arange(6)
out = x < 3
```

(b)
```
x = np.arange(6).reshape(2,3)
out = x[1,2]
```

(c)
```
x = np.arange(6).reshape(-1,2)
out = x.shape
```

(d)
```
x = np.arange(6)
out = x[0] + 1
```

(e)
```
x = np.ones((2,3))
out = x.sum(axis=0)
```

(f)
```
x = np.zeros((5,6))
x[0] = 1
out = x[0,3]
```

3 PUNKTE

Aufgabe 8 Keras

Consider the following model definition:

```
from keras.layers import Embedding, Conv1D, GlobalMaxPooling1D, Dense, Dropout
from keras.models import Sequential

VOCABSIZE = 10000
EMBSIZE = 100
HIDDENSIZE = 10
KERNELSIZE = 5

dropout_L = Dropout(0.25)
embedding_L = Embedding(input_dim = VOCABSIZE, output_dim = EMBSIZE, use_bias=True)
cnn_L = Conv1D(filters = HIDDENSIZE, kernel_size = KERNELSIZE)
pool_L = GlobalMaxPooling1D()
output_L = Dense(units = 1, activation = "sigmoid")
model = Sequential([embedding_L, dropout_L, cnn_L, pool_L, dropout_L, output_L])
```

(a) How many learnable parameters does the `cnn_L` layer contain?

(b) What kinds of tasks would this model architecture be appropriate for? Give at least one specific example.

(c) Redefine the `output_L` layer so that the model can be trained on a dataset with 5 exclusive categorical labels.

5 PUNKTE

Aufgabe 9 PyTorch

Consider a PyTorch code snippet, implementing a model with the following scoring function:

$$\hat{y} = W_2 tanh(W_1 x + b_1) + b_2$$

```
from torch.nn import Linear, Conv1D, LSTM, Tanh, Sigmoid

class MyModule(torch.nn.Module):
    def __init__(self):
        super().__init__()
        self.layer1 = Linear(100, _____________ )    # todo (a)

        self.layer2 = Tanh()

        self.layer3 =  _____________ (10, 1)         # todo (a)


    def forward(x):
        x = self.layer1(x)
        x = self.layer2(x)
        x = self.layer3(x)
        return x


# assume batch data x and batch targets y

model = MyModule()
output = model(x)
```

(a) Complete the definitions of the `layer1` and `layer3` in the underlined positions.

(b) What shape is the input `x` to model(x) restricted to?

(c) What is the output range of the model predictions? What would be an appropriate loss function?

5 PUNKTE