Deep Learning - Übungsblatt 4

Recurrent Neural Networks und LSTM

Fachhochschule Südwestfalen

23. Oktober 2025

Voraussetzungen

- Übungsblätter 1-3 sollten erfolgreich bearbeitet worden sein
- Verständnis von Backpropagation und CNNs
- Grundkenntnisse in Sequenz-Datenverarbeitung
- Mathematische Grundlagen: Zeitreihenanalyse, Rekurrenz

Lernziele

Nach erfolgreicher Bearbeitung dieser Übung können Sie:

- RNN-Architekturen verstehen und mathematisch beschreiben
- Das Vanishing Gradient Problem in RNNs erklären
- LSTM- und GRU-Mechanismen implementieren und anwenden
- Sequenz-zu-Sequenz Modelle für verschiedene Aufgaben verwenden
- Attention-Mechanismen grundlegend verstehen

1 RNN-Grundlagen

1.1 Vanilla RNN Forward Pass

Aufgabe 1.1: Gegeben sei ein einfaches RNN mit folgenden Parametern:

$$W_{xh} = \begin{pmatrix} 0.5 & 0.3 \\ -0.2 & 0.4 \end{pmatrix}, \quad W_{hh} = \begin{pmatrix} 0.1 & -0.3 \\ 0.6 & 0.2 \end{pmatrix} \tag{1}$$

$$W_{hy} = \begin{pmatrix} 0.7 & -0.1 \end{pmatrix}, \quad b_h = \begin{pmatrix} 0.1 \\ -0.2 \end{pmatrix}, \quad b_y = 0.3 \tag{2}$$

Input-Sequenz: $x_1 = (1, 0)^T$, $x_2 = (0, 1)^T$, $x_3 = (1, 1)^T$

RNN-Gleichungen:

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h) \tag{3}$$
$$y_t = W_{hy}h_t + b_y \tag{4}$$

(a) Berechnen Sie $h_0 = (0, 0)^T$ (Initialzustand)

(b) Berechnen Sie h_1 und y_1 für Input x_1

(c) Berechnen Sie h_2 und y_2 für Input x_2

(d) Berechnen Sie h_3 und y_3 für Input x_3

(e) Interpretieren Sie: Wie entwickelt sich der Hidden State über die Zeit?

1.2 Backpropagation Through Time (BPTT)

Aufgabe 1.2: Vereinfachtes BPTT

Betrachten Sie ein RNN mit nur einer Hidden Unit für 3 Zeitschritte.

Gegeben: - $W_{xh} = 0.5$, $W_{hh} = 0.8$, $W_{hy} = 1.0$ - $b_h = 0$, $b_y = 0$ - Targets: $\hat{y}_1 = 1$, $\hat{y}_2 = 0$, $\hat{y}_3 = 1$ - Loss: $L_t = \frac{1}{2}(y_t - \hat{y}_t)^2$

(a) Berechnen Sie den Forward Pass für $x_1 = 1$, $x_2 = 0$, $x_3 = 1$

(b) Berechnen Sie $\frac{\partial L_3}{\partial W_{hy}}$

(c) Berechnen Sie $\frac{\partial L_3}{\partial h_3}$

(d) Zeigen Sie die Kettenregel für $\frac{\partial L_3}{\partial W_{hh}}$:

$$\frac{\partial L_3}{\partial W_{hh}} = \frac{\partial L_3}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial W_{hh}} + \frac{\partial L_3}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial W_{hh}}$$

(e) Erklären Sie das Vanishing Gradient Problem anhand dieser Berechnung

2 LSTM-Mechanismen

2.1 LSTM-Zell-Mathematik

Aufgabe 2.1: Analysieren Sie eine LSTM-Zelle mit folgenden Gleichungen:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad \text{(Forget Gate)} \tag{5}$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad \text{(Input Gate)} \tag{6}$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad \text{(Candidate Values)} \tag{7}$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad \text{(Cell State)} \tag{8}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad \text{(Output Gate)} \tag{9}$$
$$h_t = o_t * \tanh(C_t) \quad \text{(Hidden State)} \tag{10}$$

(a) Erklären Sie die Funktion jeder Komponente (Gates und States)

(b) Gegeben seien vereinfachte 1D-Parameter:

- $W_f = [0.2, 0.1]$, $b_f = 0.5$
- $W_i = [0.3, 0.4]$, $b_i = 0.2$
- $W_C = [0.1, 0.5]$, $b_C = 0.1$
- $W_o = [0.4, 0.2]$, $b_o = 0.3$

(c) Berechnen Sie alle Zwischenwerte für $h_{t-1} = 0.5$, $x_t = 1.0$, $C_{t-1} = 0.3$

(d) Interpretieren Sie: Welche Information wird "vergessenünd welche wird ërinnert"?

2.2 LSTM vs. Vanilla RNN

Aufgabe 2.2: Vergleichende Analyse

(a) Berechnen Sie die Anzahl der Parameter für:

- Vanilla RNN mit Hidden Size 128 und Input Size 64
- LSTM mit Hidden Size 128 und Input Size 64

(b) Erklären Sie, warum LSTMs das Vanishing Gradient Problem besser handhaben

(c) Diskutieren Sie: Wann würden Sie ein Vanilla RNN gegenüber einem LSTM bevorzugen?

(d) Beschreiben Sie das GRU (Gated Recurrent Unit) als Kompromiss zwischen RNN und LSTM

3 Sequenz-zu-Sequenz Modelle

3.1 Sequence-to-Sequence Architekturen

Aufgabe 3.1: Encoder-Decoder Design

(a) Skizzieren Sie eine Seq2Seq-Architektur für Übersetzung:

- Input: "Hello World"(Englisch)
- Output: "Hallo Welt"(Deutsch)

(b) Erklären Sie die Rolle des **Context Vectors**

(c) Diskutieren Sie das **Information Bottleneck** Problem

(d) Wie kann **Teacher Forcing** beim Training helfen?

(e) Beschreiben Sie den Unterschied zwischen Training und Inference

3.2 Attention-Mechanismus

Aufgabe 3.2: Grundlagen der Attention

(a) Erklären Sie das Problem fester Context-Vektoren bei langen Sequenzen

(b) Beschreiben Sie den Bahdanau Attention-Mechanismus:

$$e_{t,i} = \text{align}(s_{t-1}, h_i) \tag{11}$$

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{j=1}^{T_x} \exp(e_{t,j})} \tag{12}$$

$$c_t = \sum_{i=1}^{T_x} \alpha_{t,i} h_i \tag{13}$$

(c) Interpretieren Sie die Attention-Gewichte $\alpha_{t,i}$

(d) Diskutieren Sie: Wie löst Attention das Information Bottleneck Problem?

4 Programmieraufgaben

4.1 RNN von Grund auf implementieren

Aufgabe 4.1: Vanilla RNN Implementation

Listing 1: RNN Grundgerüst

```python
import numpy as np
import matplotlib.pyplot as plt

class SimpleRNN:
    def __init__(self, input_size, hidden_size, output_size):
        self.hidden_size = hidden_size

        # TODO: Initialisieren Sie die Gewichtsmatrizen
        # self.Wxh = ...  # Input zu Hidden
        # self.Whh = ...  # Hidden zu Hidden
        # self.Why = ...  # Hidden zu Output
        # self.bh = ...   # Hidden Bias
        # self.by = ...   # Output Bias

    def forward(self, inputs):
        """
        Forward Pass durch das RNN

        Args:
            inputs: Liste von Input-Vektoren [x1, x2, ..., xT]

        Returns:
            outputs: Liste von Output-Vektoren
            hidden_states: Liste von Hidden States
        """
```

```
        h = np.zeros((self.hidden_size, 1))
        hidden_states = []
        outputs = []

        for x in inputs:
            # TODO: Implementieren Sie den Forward Pass
            # h = tanh(Wxh @ x + Whh @ h + bh)
            # y = Why @ h + by
            pass

        return outputs, hidden_states

    def backward(self, inputs, targets, outputs, hidden_states):
        """
        Backpropagation Through Time

        Returns:
            gradients: Dictionary mit Gradienten
        """
        # TODO: Implementieren Sie BPTT
        pass

def test_rnn():
    """Test des RNN mit einer einfachen Sequenz"""
    rnn = SimpleRNN(input_size=2, hidden_size=3, output_size=1)

    # Beispiel-Sequenz
    inputs = [np.array([[1], [0]]),
              np.array([[0], [1]]),
              np.array([[1], [1]])]

    outputs, hidden_states = rnn.forward(inputs)

    print("RNN Outputs:")
    for i, output in enumerate(outputs):
        print(f"t={i+1}: {output.flatten()}")

if __name__ == "__main__":
    test_rnn()
```

Teilaufgaben:

(a) Implementieren Sie die `__init__` Methode mit Xavier-Initialisierung

(b) Vervollständigen Sie den `forward` Pass

(c) Implementieren Sie `backward` für BPTT (vereinfacht)

(d) Testen Sie mit verschiedenen Sequenzen

(e) Visualisieren Sie die Hidden States über die Zeit

4.2 LSTM mit TensorFlow/Keras

Aufgabe 4.2: Zeitreihen-Vorhersage

Listing 2: LSTM für Zeitreihen

```python
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import numpy as np
import matplotlib.pyplot as plt

def generate_sine_data(seq_length, num_samples):
    """
    Generiert Sinus-Zeitreihen für Training

    Args:
        seq_length: Länge jeder Sequenz
        num_samples: Anzahl der Beispiele

    Returns:
        X: Input-Sequenzen (num_samples, seq_length, 1)
        y: Targets (num_samples, 1)
    """
    # TODO: Generieren Sie Sinus-Daten
    # Tipp: np.sin(np.linspace(...))
    pass

def create_lstm_model(seq_length):
    """
    Erstellt ein LSTM-Modell für Zeitreihen-Vorhersage

    Args:
        seq_length: Länge der Input-Sequenzen

    Returns:
        model: Keras-Modell
    """
    model = keras.Sequential([
        # TODO: Implementieren Sie die LSTM-Architektur
        # - LSTM Layer(s)
        # - Dense Layer für Output
        # - Geeignete Aktivierungsfunktionen
    ])

    return model

def plot_predictions(model, test_data, test_targets):
    """Visualisiert Vorhersagen vs. tatsächliche Werte"""
    predictions = model.predict(test_data)

    plt.figure(figsize=(12, 6))
    plt.plot(test_targets[:100], label='Tatsächliche Werte', alpha
```

```
        =0.7)
    plt.plot(predictions[:100], label='Vorhersagen', alpha=0.7)
    plt.legend()
    plt.title('LSTM Zeitreihen-Vorhersage')
    plt.show()

def main():
    # Parameter
    seq_length = 20
    num_samples = 10000

    # Daten generieren
    X, y = generate_sine_data(seq_length, num_samples)

    # TODO: Train/Test Split
    # TODO: Modell erstellen und trainieren
    # TODO: Evaluation und Visualisierung

if __name__ == "__main__":
    main()
```

Teilaufgaben:

(a) Implementieren Sie die Sinus-Datengenerierung

(b) Erstellen Sie ein LSTM-Modell mit 1-2 LSTM-Schichten

(c) Trainieren Sie das Modell für Sinus-Vorhersage

(d) Erweitern Sie auf komplexere Zeitreihen (mehrere Sinuswellen, Rauschen)

(e) Implementieren Sie Multi-Step-Vorhersage

(f) Vergleichen Sie LSTM vs. GRU vs. Simple RNN

4.3 Text-Generierung

Aufgabe 4.3: Character-Level RNN

Listing 3: Text-Generierung mit RNN

```
import tensorflow as tf
from tensorflow import keras
import numpy as np
import string

class CharRNN:
    def __init__(self, vocab_size, embedding_dim, rnn_units):
        self.vocab_size = vocab_size
        self.embedding_dim = embedding_dim
        self.rnn_units = rnn_units
        self.model = self._build_model()

    def _build_model(self):
```

```
14          """Erstellt das Character-RNN Modell"""
15          model = keras.Sequential([
16              # TODO: Implementieren Sie die Architektur
17              # - Embedding Layer
18              # - LSTM/GRU Layer(s)
19              # - Dense Layer mit vocab_size Outputs
20          ])
21          return model
22
23      def prepare_data(self, text):
24          """
25          Bereitet Text-Daten für Training vor
26
27          Args:
28              text: Input-Text als String
29
30          Returns:
31              dataset: TensorFlow Dataset
32              char_to_idx: Character zu Index Mapping
33              idx_to_char: Index zu Character Mapping
34          """
35          # TODO: Implementierung
36          # - Eindeutige Charaktere extrahieren
37          # - Character-zu-Index Mappings erstellen
38          # - Text in Sequenzen aufteilen
39          pass
40
41      def generate_text(self, seed_text, num_generate=100, temperature
            =1.0):
42          """
43          Generiert Text basierend auf Seed-Text
44
45          Args:
46              seed_text: Start-Text
47              num_generate: Anzahl zu generierender Charaktere
48              temperature: Sampling-Temperature (Kreativität)
49
50          Returns:
51              generated_text: Generierter Text
52          """
53          # TODO: Implementierung
54          # - Seed-Text in Indices konvertieren
55          # - Iterativ Charaktere sampeln und vorhersagen
56          pass
57
58  def main():
59      # Beispiel-Text (Sie können auch eigene Texte verwenden)
60      sample_text = """
61      Machine learning is a subset of artificial intelligence that
            focuses on
62      algorithms that can learn from data. Deep learning is a subset of
```

```
        machine
    learning that uses neural networks with multiple layers.
    """

    # TODO: CharRNN erstellen und trainieren
    # TODO: Text generieren und evaluieren

if __name__ == "__main__":
    main()
```

Teilaufgaben:

(a) Implementieren Sie die Datenvorverarbeitung für Character-Level Text

(b) Erstellen Sie ein RNN für Text-Generierung

(c) Trainieren Sie auf einem kleinen Text-Korpus

(d) Experimentieren Sie mit verschiedenen Temperaturen beim Sampling

(e) Implementieren Sie Beam Search für bessere Generierung

(f) Erweitern Sie auf Word-Level Generation

5 Erweiterte Konzepte

5.1 Bidirectional RNNs

Aufgabe 5.1: Bidirectional LSTM

(a) Erklären Sie die Motivation für bidirectionale RNNs

(b) Skizzieren Sie die Architektur eines Bidirectional LSTM

(c) Berechnen Sie die Anzahl Parameter im Vergleich zu einem unidirektionalen LSTM

(d) Implementieren Sie ein bidirectionales LSTM für Sentiment-Analyse

(e) Diskutieren Sie: Wann sind bidirectionale RNNs nicht geeignet?

5.2 Sequence-to-Sequence mit Attention

Aufgabe 5.2: Attention-Implementierung

Listing 4: Einfacher Attention-Mechanismus

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

class AttentionLayer(layers.Layer):
    def __init__(self, units):
        super(AttentionLayer, self).__init__()
        self.units = units
```

```
9
10      def build(self, input_shape):
11          # TODO: Definieren Sie die Gewichtsmatrizen für Attention
12          # W1, W2, V für Bahdanau Attention
13          pass
14
15      def call(self, query, values):
16          """
17          Berechnet Attention-Gewichte und Context-Vektor
18
19          Args:
20              query: Decoder Hidden State (batch_size, hidden_size)
21              values: Encoder Hidden States (batch_size, seq_len,
                    hidden_size)
22
23          Returns:
24              context_vector: Gewichtete Summe der Encoder States
25              attention_weights: Attention-Gewichte
26          """
27          # TODO: Implementieren Sie Attention-Berechnung
28          pass
29
30  def create_seq2seq_with_attention():
31      """Erstellt Seq2Seq-Modell mit Attention"""
32      # TODO: Implementierung
33      pass
```

6 Verständnisfragen

Aufgabe 6.1: RNN-Limitationen

(a) Erklären Sie das **Vanishing Gradient Problem** in RNNs:

- Mathematische Ursache
- Auswirkungen auf das Training
- Warum sind lange Sequenzen besonders problematisch?

(b) Beschreiben Sie das **Exploding Gradient Problem**:

- Unterschied zu Vanishing Gradients
- Lösungsansätze (Gradient Clipping)

(c) Diskutieren Sie **Computational Efficiency**:

- Warum können RNNs nicht vollständig parallelisiert werden?
- Vergleich mit CNNs und Transformers

Aufgabe 6.2: LSTM-Mechanismen

(a) Erklären Sie detailliert, wie jedes LSTM-Gate funktioniert:

- **Forget Gate:** Welche Information wird ”vergessen”?
- **Input Gate:** Wie wird neue Information ausgewählt?
- **Output Gate:** Wie wird der Output gesteuert?

(b) Beschreiben Sie den **Cell State** vs. **Hidden State**:

- Unterschiedliche Rollen
- Informationsfluss durch das Netzwerk

(c) Vergleichen Sie **LSTM vs. GRU**:

- Anzahl Parameter
- Computational Complexity
- Performance-Unterschiede

7 Zusatzaufgaben (Optional)

Aufgabe 7.1: Multi-Modal Seq2Seq

(a) Implementieren Sie ein Modell für Image Captioning:

- CNN-Encoder für Bilder
- RNN-Decoder für Text-Generierung
- Attention zwischen visuellen Features und Text

(b) Erweitern Sie auf Video Captioning mit 3D-CNNs

Aufgabe 7.2: Transformer vs. RNN

(a) Recherchieren Sie die Transformer-Architektur

(b) Implementieren Sie einen einfachen Transformer-Block

(c) Vergleichen Sie RNN vs. Transformer für:

- Parallelisierbarkeit
- Memory-Effizienz
- Lange Sequenzen
- Training-Zeit

(d) Diskutieren Sie: Ättention is All You Need stimmt das?

Hinweise und Tipps

Zu den Mathematik-Aufgaben

- **Dimensionen:** Prüfen Sie immer Matrix-Dimensionen bei Multiplikationen
- **Aktivierungsfunktionen:** tanh für Hidden States, σ für Gates
- **Gradienten:** Verwenden Sie die Kettenregel systematisch
- **Numerische Stabilität:** Beachten Sie Overflow bei exp() in Softmax

Zu den Programmieraufgaben

- **Sequence Length:** Beginnen Sie mit kurzen Sequenzen (10-20 Zeitschritte)
- **Batch Size:** RNNs benötigen oft kleinere Batches als CNNs
- **Learning Rate:** Verwenden Sie kleinere Learning Rates für RNNs (0.001-0.01)
- **Gradient Clipping:** Implementieren Sie Gradient Clipping bei Training

Häufige Fehler vermeiden

- **Sequence Dimension:** Achten Sie auf (batch, time, features) Anordnung
- **Stateful vs. Stateless:** Verstehen Sie, wann States zwischen Batches übertragen werden
- **Return Sequences:** `return_sequences=True` für Seq2Seq, `False` für Classification
- **Masking:** Verwenden Sie Masking für variable Sequenzlängen

Weiterführende Ressourcen

- **Bücher:**
  - ”Deep Learning Goodfellow et al., Kapitel 10
  - SSpeech and Language Processing Jurafsky & Martin
- **Papers:**
  - ”Long Short-Term Memory Hochreiter & Schmidhuber (1997)
  - ”Learning Phrase Representations using RNN Encoder-Decoder Cho et al.
  - ”Neural Machine Translation by Jointly Learning to Align and Translate Bahdanau et al.
  - Ättention Is All You Need Vaswani et al.
- **Online:**
  - Ünderstanding LSTM Networks Christopher Olah
  - ”The Unreasonable Effectiveness of RNNs Andrej Karpathy
  - CS224n: Natural Language Processing with Deep Learning