

Universidade Federal do Rio de Janeiro
Instituto Alberto Luiz Coimbra de
Pós-Graduação e Pesquisa de Engenharia



Programa de Engenharia de Sistemas e
Computação

CPS769 - Introdução à Inteligência Artificial e Aprendizagem Generativa

Prof. Dr. Edmundo de Souza e Silva (PESC/COPPE/UFRJ)

Profa. Dra. Rosa M. Leão (PESC/COPPE/UFRJ)

Participação Especial: Gaspare Bruno (Diretor Inovação, ANLIX)

Lista de Exercícios 1a

Luiz Henrique Souza Caldas

email: lhscaldas@cos.ufrj.br

9 de julho de 2024

Questão 1

Esse exemplo simples é para auxiliar a discussão do artigo “Serial Order A Parallel Distributed Processing Approach” que todos já devem ter lido. O objetivo é prever um padrão de figura, por exemplo um quadrado, usando uma Rede Neural Recorrente (RNN). Fornecemos o código em Python de um exemplo de geração do padrão 2-D de quadrados e treinamento de uma RNN para prever a sequência cíclica $[0, 25, 0, 25]$, $[0, 75, 0, 25]$, $[0, 75, 0, 75]$, $[0, 25, 0, 75]$, $[0, 25, 0, 25]$.

1. Entenda o código e explique qual a RNN que ele modela (faça o desenho). Explique a parte do código que define a RNN.

Resposta:

O código pode ser explicado dividindo-o em 6 partes:

- (a) Definição do caminho quadrado: O código define um conjunto de coordenadas que formam um caminho quadrado na variável *square_path*.

```
1 square_path = np.array([
2     [0.25, 0.25],
3     [0.75, 0.25],
4     [0.75, 0.75],
5     [0.25, 0.75],
6     [0.25, 0.25]
7 ])
```

- (b) Geração dos dados de treinamento: O caminho quadrado é repetido várias vezes para formar os dados de treinamento.

```
1 num_repeats = 4
2 data = np.tile(square_path, (num_repeats, 1))
3 x_train = data[:-1].reshape(-1, 1, 2)
4 y_train = data[1:].reshape(-1, 2)
```

- (c) Definição e compilação do modelo RNN: O modelo RNN é definido usando uma camada LSTM (*long short-term memory*) seguida de uma camada densa e depois é compilado configurando o algoritmo ADAM como otimizador e o Erro Médio Quadrático como função de perda.

```
1 model = models.Sequential([
2     layers.LSTM(50, activation='relu', input_shape=(num_repeats, 2)),
3     layers.Dense(2)
4 ])
5 model.compile(optimizer='adam', loss='mse')
```

- (d) Treinamento do modelo: O modelo é treinado com os dados gerados, utilizando inicialmente 300 épocas.

```
1 model.fit(x_train, y_train, epochs=300, verbose=0)
```

- (e) Geração das previsões: As previsões são geradas.

```
1 predictions = model.predict(x_train[:5])
```

- (f) Plotagem dos resultados: As previsões são plotadas e comparadas com o caminho original.

```
1 plt.plot(data[:, 0], data[:, 1], label='Original Path', linestyle='
    dashed', color='gray')
2 plt.plot(predictions[:, 0], predictions[:, 1], label='Predicted Path',
    color='blue')
3 plt.scatter(square_path[:, 0], square_path[:, 1], color='red')
4 plt.legend()
5 plt.show()
```

2. Treine a rede. Aprenda como fazer, e explique.

Resposta:

3. Faça a previsão de algumas trajetórias, quando o ponto inicial varia. O que você conclui?

Resposta:

4. Modifique a RNN usada e observe o que acontece.

Resposta:

5. Quais os pontos principais que você concluiu do artigo “Serial Order A Parallel Distributed Processing Approach”?

Resposta:

Código 1: código fornecido completo

```
1 # Modify the Python Script to Disable GPU
2 import os
3 os.environ['CUDA_VISIBLE_DEVICES'] = '-1'
4
5 # to run the python script: python3 rnn_cyclic_sequence.py
6
7 import numpy as np
8 import tensorflow as tf
9 from tensorflow.keras import layers, models
10 import matplotlib.pyplot as plt
11
12 # Define the square path coordinates
13 square_path = np.array([
14     [0.25, 0.25],
15     [0.75, 0.25],
16     [0.75, 0.75],
17     [0.25, 0.75],
18     [0.25, 0.25]
19 ])
20
21 # Generate the training data by repeating the square path
22 num_repeats = 4
23 data = np.tile(square_path, (num_repeats, 1))
24
```

```

25 # Prepare training data
26 x_train = data[:-1].reshape(-1, 1, 2)
27 y_train = data[1:].reshape(-1, 2)
28
29 # Define the RNN model
30 model = models.Sequential([
31     layers.LSTM(50, activation='relu', input_shape=(num_repeats, 2)),
32     layers.Dense(2)
33 ])
34
35 # Compile the model
36 model.compile(optimizer='adam', loss='mse')
37
38 # Train the model
39 model.fit(x_train, y_train, epochs=300, verbose=0)
40
41 # Generate predictions
42 predictions = model.predict(x_train[:5])
43
44 # Plot the results
45 plt.figure(figsize=(10, 6))
46 plt.plot(data[:, 0], data[:, 1], label='Original Path', linestyle='dashed',
47          color='gray')
48 plt.plot(predictions[:, 0], predictions[:, 1], label='Predicted Path', color='
49          blue')
50 plt.scatter(square_path[:, 0], square_path[:, 1], color='red')
51 plt.legend()
52 plt.show()

```