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:

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

```
# Modify the Python Script to Disable GPU
import os
os.environ['CUDA_VISIBLE_DEVICES'] = '-1'

# to run the python script: python3 rnn_cyclic_sequence.py

import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt

# Define the square path coordinates
square_path = np.array([
    [0.25, 0.25],
    [0.75, 0.25],
    [0.75, 0.75],
    [0.25, 0.75],
    [0.25, 0.25]
])

# Generate the training data by repeating the square path
num_repeats = 4
```

```
data = np.tile(square_path, (num_repeats, 1))

# Prepare training data
x_train = data[:-1].reshape(-1, 1, 2)
y_train = data[1:].reshape(-1, 2)

# Define the RNN model
model = models.Sequential([
    layers.LSTM(50, activation='relu', input_shape=(num_repeats, 2)),
    layers.Dense(2)
])

# Compile the model
model.compile(optimizer='adam', loss='mse')

# Train the model
model.fit(x_train, y_train, epochs=300, verbose=0)

# Generate predictions
predictions = model.predict(x_train[:5])

# Plot the results
plt.figure(figsize=(10, 6))
plt.plot(data[:, 0], data[:, 1], label='Original Path', linestyle='dashed',
    color='gray')
plt.plot(predictions[:, 0], predictions[:, 1], label='Predicted Path', color='
    blue')
plt.scatter(square_path[:, 0], square_path[:, 1], color='red')
plt.legend()
plt.show()
```