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 2

Luiz Henrique Souza Caldas
email: lhscaldas@cos.ufrj.br

25 de julho de 2024

Questão 1

O objetivo deste trabalho é entender como construir um modelo preditivo simples (de Redes Neurais) usando uma ou mais camadas ocultas, e se familiarizar com os códigos em Python. Nesta tarefa, você deverá construir um modelo de rede neural simples para prever a nota de cada aluno em uma turma com base em duas features:

- Fração de palestras assistidas
- Número de horas estudadas por semana (até o máximo de 8 horas em uma semana).

Sobre a Rede Neural e programa:

- A sua rede neural deverá ter uma camada oculta com 3 neurônios. Portanto teremos 2 entradas (as 2 features), uma camada oculta e, na camada de saída, um neurônio.
- Os dados de entrada são fornecidos em uma planilha .ods (libreoffice), com os dados de 500 estudantes.
- Uma vez lido, o seu dataset deve ser aleatoriamente dividido de forma a que 80% seja para treino do modelo e os restantes 20% para teste.
- No seu programa:
  - Use um Scaler padrão do sklearn para dimensionar os dados de treinamento. Explique o motivo de escalonar os dados de entrada.

    Explicação:

    Escalonar os dados de entrada normaliza suas distribuições, levando a média para zero e o desvio padrão para o valor unitário, permitindo uma convergência mais rápida e estável do modelo de machine learning, pois o modelo não precisa lidar com features de ordem de grandeza diferentes.
  - Para a camada oculta, use a função de ativação ReLU (Rectified Linear Unit).
  - A camada de saída usa a ativação linear padrão (explique o que é).

    Explicação:

    A ativação linear padrão retorna a entrada sem modificações ($f(x) = x$), usada em problemas de regressão para prever valores contínuos.
  - Use o algoritmo de otimização Adam (Adaptive Moment Estimation). Explique bem resumidamente as vantagens em relação ao Stochastic Gradient Descent padrão.

    Explicação:

    Adam ajusta a taxa de aprendizagem para cada parâmetro individualmente e utiliza momentos dos gradientes, permitindo convergência mais rápida e estável.
  - Use mean square error para a função de perda. Explique bem resumidamente o objetivo da função de perda.

    Explicação:

    A função de perda mede a diferença entre as previsões do modelo e os valores reais. Durante o aprendizado, o ajuste dos pesos é feito de forma a minimizar o valor da função de perda, melhorando a acurácia do modelo.

Responda:

1. Como o modelo de rede neural está estruturado? Explique a arquitetura.

   Resposta:

   O modelo possui duas entradas (as duas features), uma camada oculta com 3 neurônios usando a função de ativação ReLU, e uma camada de saída com um neurônio e ativação linear.

2. Explique o papel da função de ativação usada na camada oculta.

   Resposta:

   A função ReLU (Rectified Linear Unit) ajuda a introduzir não-linearidade ao modelo, permitindo que ele aprenda relações complexas entre as entradas e as saídas.

3. Treine o modelo com o conjunto de dados fornecido. Qual o erro quadrático médio nos dados de teste?

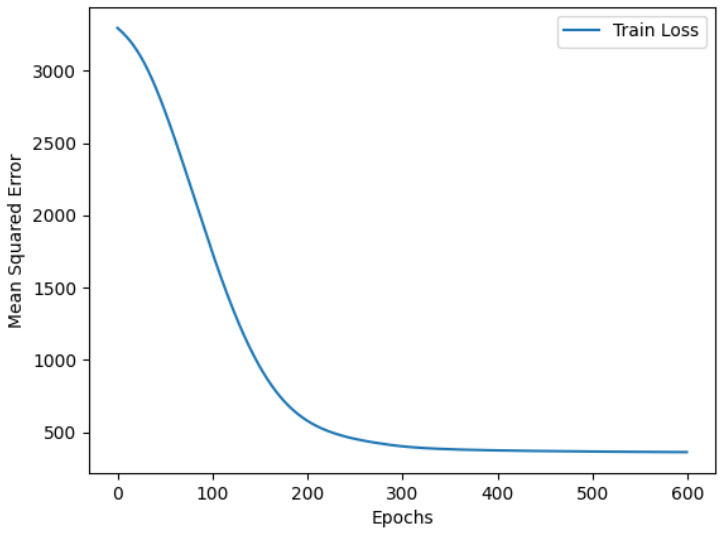
   Resposta:

   O modelo foi treinado por 600 épocas e no conjunto de teste o menor erro médio quadrático obtido foi de 307.94.

4. Trace o erro quadrático médio em função das "épocas" (dos passos para a convergência). Descreva a tendência que você observa.

   Resposta:

Figura 1: Eveloução do erro de treinamento ao longo das épocas.



Pelo gráfico acima, é posssível observar que o erro médio quadrático tende a cair com o aumento das épocas até que esta queda deixa de ser relevante, o que ocorre por volta de 400 épocas.

5. Mostre os pesos e bias de cada camada após o treinamento, isto é, mostre os parâmetros aprendidos do modelo.

   Resposta:

Tabela 1: Pesos e Bias das Camadas

Camada	Tipo	Valores
1	Pesos	2.3686342 — 2.3783615 — 0.71419466 2.0336158 — -2.3631034 — 1.5801256
1	Bias	3.7342753 — 4.462739 — 4.0898366
2	Pesos	4.112546 — 3.063474 — 4.479588
2	Bias	2.6367683

6. Use o modelo treinado para prever as notas a partir dos dados de novos alunos (com um segundo dataset fornecido sem as notas). Mostre as previsões feitas pelo modelo e explique os resultados.

   Resposta:

Tabela 2: Resultados da previsão a partir dos dados de novos alunos.

Presença	HorasEstudo	Nota
0.3745401188	2.974337759	32.95745
0.9507143064	5.628494665	82.800064
0.7319939418	3.183951849	57.951267
0.5986584842	2.175392597	44.790115
0.1560186404	7.931297538	44.161804
0.1559945203	4.548447399	24.69879
0.05808361217	5.135056409	20.509508
0.8661761458	2.150503628	62.7456
0.6011150117	3.911234551	52.116886
0.7080725778	5.221845179	64.74353
0.0205844943	2.698012845	17.904613
0.9699098522	5.751396037	84.60283
0.8324426408	3.79872262	67.26796
0.2123391107	4.4166125	27.958336
0.1818249672	3.796586776	23.457981
0.1834045099	8.704556369	51.34989
0.304242243	4.973005551	36.457355
0.5247564316	2.884578142	42.72716
0.4319450186	6.645089824	51.97556
0.2912291402	2.5583127	27.386648

7. Modifique o programa para adicionar mais camadas e/ou maior número de neurônios ocultos. Como suas modificações afetam o desempenho do modelo?

Resposta:

Foram testados dois novos modelos, um com 3 camadas ocultas com 3 neurônios cada e outro com uma camada oculta com 9 neurônios.

Enquanto o modelo original (uma camada oculta com 3 neurônios) obteve um erro médio quadrático de 307.94, o modelo de 3 camadas de 3 neurônios obteve um erro de 315.00 e o modelo com uma camada de 9 neurônios obteve um erro de 309.56.

Foi testado então um novo modelo com uma camada oculta de 100 neurônios (valor padrão do tensor flow). O erro médio quadrático obtido foi de 312.28.

Nenhuma das 3 novas arquiterturas testadas superou o desempenho da arquitetura original, com uma camada oculta de 3 neurônios.

8. Quais seriam algumas melhorias potenciais ou recursos adicionais que poderiam ser adicionados ao modelo para melhorar sua precisão preditiva?

   Resposta:

   Para melhorar a precisão preditiva, poderia-se adicionar regularização (como Dropout), usar técnicas de aumento de dados, ou experimentar diferentes arquiteturas de rede.

9. Faria sentido usar a função de ativação sigmoid no modelo? Explique em poucas palavras.

   Resposta:

   A função sigmoid é mais adequada para tarefas de classificação binária. Para problemas de regressão, como o apresentado, a ativação linear na saída é mais apropriada.

Código

O código abaixo encontra-se no repositório https://github.com/lhscaldas/cps769-ai-gen, bem como o arquivo LaTex com o relatório.

Código 1: código fornecido completo com algumas modificações

```
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import random

class NeuralNetworkModel:
    def __init__(self, data_path, seed=42):
        self.data_path = data_path
        self.model = None
        self.history = None
        self.scaler = StandardScaler()
        self.seed = seed
        self._set_seed()

```

```
    def _set_seed(self):
        np.random.seed(self.seed)
        random.seed(self.seed)
        tf.random.set_seed(self.seed)

    def load_data(self):
        self.data = pd.read_csv(self.data_path, decimal=',', dtype=float)
        self.X = self.data[['Presença', 'HorasEstudo']]
        self.y = self.data['Nota']

    def preprocess_data(self):
        self.X_train, self.X_test, self.y_train, self.y_test = train_test_split
            (self.X, self.y, test_size=0.2, random_state=self.seed)
        self.X_train_scaled = self.scaler.fit_transform(self.X_train)
        self.X_test_scaled = self.scaler.transform(self.X_test)

    def build_model(self):
        self.model = tf.keras.Sequential([
            tf.keras.layers.InputLayer(input_shape=(2,)),
            tf.keras.layers.Dense(3, activation='relu'),
            tf.keras.layers.Dense(1)
        ])
        self.model.compile(optimizer='adam', loss='mean_squared_error')

    def train_model(self, epochs=600):
        self.history = self.model.fit(self.X_train_scaled, self.y_train, epochs
            =epochs, verbose=0)

    def evaluate_model(self):
        test_loss = self.model.evaluate(self.X_test_scaled, self.y_test)
        print(f'Mean Squared Error on Test Data: {test_loss}')

    def plot_loss(self):
        plt.plot(self.history.history['loss'], label='Train Loss')
        plt.xlabel('Epochs')
        plt.ylabel('Mean Squared Error')
        plt.legend()
        plt.show()

    def show_weights_and_biases(self):
        for layer in self.model.layers:
            weights, biases = layer.get_weights()
            print(f'Pesos da camada: {weights}')
            print(f'Bias da camada: {biases}')

    def predict_new_data(self, new_data_path):
        new_data = pd.read_csv(new_data_path, decimal='.', dtype=float)
        new_X = new_data[['Presença', 'HorasEstudo']]
        new_X_scaled = self.scaler.transform(new_X)
        predictions = self.model.predict(new_X_scaled, verbose=0)
        new_data['Nota'] = predictions
        new_data.to_csv('lista_2/lista_2-students_data_new.csv', index=False)
        print(f'Previsões salvas no arquivo lista_2-students_data_new.csv')

```

```
    def modify_and_train_new_model(self, new_layers, epochs=600):
        modified_model = tf.keras.Sequential()
        modified_model.add(tf.keras.layers.InputLayer(input_shape=(2,)))

        for units in new_layers:
            modified_model.add(tf.keras.layers.Dense(units, activation='relu'))

        modified_model.add(tf.keras.layers.Dense(1))
        modified_model.compile(optimizer='adam', loss='mean_squared_error')

        modified_history = modified_model.fit(self.X_train_scaled, self.y_train
            , epochs=epochs, verbose=0)
        modified_test_loss = modified_model.evaluate(self.X_test_scaled, self.
            y_test)
        print(f'Mean Squared Error on Test Data with Modified Model: {
            modified_test_loss}')

if __name__ == "__main__":
    data_path = 'lista_2/lista_2-students_data.csv'
    new_data_path = 'lista_2/lista_2-students_data_new.csv'

    nn_model = NeuralNetworkModel(data_path)
    nn_model.load_data()
    nn_model.preprocess_data()
    nn_model.build_model()
    nn_model.train_model()
    nn_model.evaluate_model()
    # nn_model.plot_loss()
    # nn_model.show_weights_and_biases()
    # nn_model.predict_new_data(new_data_path)
    nn_model.modify_and_train_new_model([3, 3, 3])
    nn_model.modify_and_train_new_model([9])
    nn_model.modify_and_train_new_model([100])
```