Universidade Federal do Ceará
Departamento de Computação
Disciplina: Inteligência Artificial
Prof. João Paulo do Vale Madeiro

Aula Prática: Introdução ao Deep Learning com MLP

Objetivo: Nesta prática, você irá construir e experimentar redes neurais do tipo Perceptron Multicamadas (MLP), observando o efeito de diferentes configurações de hiperparâmetros sobre o desempenho do modelo. As implementações podem ser feitas utilizando a biblioteca **PyTorch**.

1 – MLP para problema lógico XOR

- (a) Implemente uma rede neural sem camada oculta e observe se ela consegue aprender a função XOR.
- (b) Implemente uma rede com uma camada oculta de 4 neurônios e função de ativação `Sigmoid`. Treine com taxa de aprendizado $\alpha = 0{,}1$ e 500 épocas.
- (c) Compare graficamente a acurácia de ambos os modelos e discuta a importância da não-linearidade introduzida pela camada oculta.

2 – Variação do número de neurônios na camada escondida

Utilize o dataset `Iris` (com as 3 classes originais).

- (a) Implemente uma rede com apenas uma camada oculta variando o número de neurônios: 4, 8 e 16.
- (b) Utilize ReLU como função de ativação e mantenha o número de épocas fixo em 100.
- (c) Plote as curvas de perda e acurácia para cada configuração e discuta a relação entre capacidade da rede e desempenho.

3 – Análise de funções de ativação e épocas

Com base no modelo de melhor desempenho da questão anterior:

- (a) Treine a rede com diferentes funções de ativação: `Sigmoid`, `Tanh` e `ReLU`.
- (b) Para cada função de ativação, treine o modelo com 50, 100 e 300 épocas.
- (c) Analise graficamente as curvas de perda e acurácia e identifique as melhores combinações. Comente sobre a velocidade de convergência.

Observações:

- Os dados devem ser normalizados antes do treinamento.
- Separe dados de treino e teste (ex: 80/20).
- Salve e anexe os gráficos das curvas de treinamento nas respostas.

Apêndice – Carregamento e Normalização de Dados

Problema XOR:
Use diretamente os dados abaixo:

```
X = [[0,0],[0,1],[1,0],[1,1]]
y = [0,1,1,0]
```

Dataset Iris:
Você pode carregar com:

```
from sklearn.datasets import load_iris
iris = load_iris()
X = iris.data
y = iris.target
```

Separar treino e teste:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)
```

Normalização (recomendado):

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Link oficial do Iris Dataset (UCI): `https://archive.ics.uci.edu/ml/datasets/iris`