Questao 4

Codigo Fonte:

```
import numpy as np
import io
from contextlib import redirect_stdout
from fpdf import FPDF
from fpdf.enums import XPos, YPos

def solve_and_capture_output():
    """
    Executa o treinamento e teste da rede neural e captura o output.
    """
    print("Questao 4: Treinamento e Teste de Rede Neural (Perceptron)")
    print("=" * 70)

    # --- Dados de Treinamento ---
    # Coluna 0: Peso (mg), Coluna 1: Acidez (Ph)
    X_train_raw = np.array([
        [370, 7], [220, 3], [180, 4], [400, 7],
        [380, 8], [410, 9], [220, 2], [170, 3]
    ])
    # Laranja: +1, Limão: -1
    y_train = np.array([1, -1, -1, 1, 1, 1, -1, -1])

    print("Parte (a): Treinamento da Rede Neural")
    print("-" * 70)
    print("1. Preparacao dos Dados")
    print("\nDados de Treinamento Brutos (X):")
    print(X_train_raw)
    print("\nLabels de Treinamento (y): [Laranja=1, Limao=-1]")
    print(y_train)

    # 2. Normalização dos Dados (Min-Max para o intervalo [0, 1])
    X_min = X_train_raw.min(axis=0)
    X_max = X_train_raw.max(axis=0)
    X_train_normalized = (X_train_raw - X_min) / (X_max - X_min)

    print("\n2. Normalizacao dos Dados de Entrada (X)")
    print(f"   - Minimos (Peso, Acidez): {X_min}")
    print(f"   - Maximos (Peso, Acidez): {X_max}")
    print("\nDados de Treinamento Normalizados:")
    print(X_train_normalized)

    # 3. Treinamento da Rede
    print("\n3. Treinamento com a Regra Delta")

    # Parâmetros
    learning_rate = 0.1
    epochs = 2000
```

```
np.random.seed(42)
weights = np.random.rand(2) * 0.1 - 0.05
bias = np.random.rand(1) * 0.1 - 0.05

print(f"   - Taxa de Aprendizagem (alpha): {learning_rate}")
print(f"   - Epocas de Treinamento: {epochs}")
print(f"   - Pesos Iniciais (w1, w2): {weights}")
print(f"   - Bias Inicial (w0): {bias}")

for epoch in range(epochs):
    for xi, target in zip(X_train_normalized, y_train):
        net_input = np.dot(xi, weights) + bias
        output = np.tanh(net_input)
        error = target - output
        delta = error * (1 - output**2)
        weights += learning_rate * delta * xi
        bias += learning_rate * delta

print("\nTreinamento Concluido.")
print("\n4. Pesos e Bias Finais Calculados:")
print(f"   - Peso w1 (relativo a 'Peso da Fruta'): {weights[0]:.4f}")
print(f"   - Peso w2 (relativo a 'Acidez'):      {weights[1]:.4f}")
print(f"   - Bias w0:                             {bias[0]:.4f}")

print("\n" + "=" * 70)

# --- Parte (b): Teste da Rede ---
print("Parte (b): Teste da Rede com Novos Dados")
print("-" * 70)

X_test_raw = np.array([
    [380, 2],
    [170, 8]
])

print("1. Dados de Teste Brutos:")
print(X_test_raw)

X_test_normalized = (X_test_raw - X_min) / (X_max - X_min)

print("\n2. Normalizacao dos Dados de Teste (usando min/max do treino):")
print(X_test_normalized)

print("\n3. Resultados da Classificacao:")

for i, xi_test in enumerate(X_test_normalized):
    net_input = np.dot(xi_test, weights) + bias
    output = np.tanh(net_input)
    classification = "Laranja" if output > 0 else "Limao"

    print(f"\n   - Fruta {i+1} (Entrada Bruta: {X_test_raw[i]})")
    print(f"     - Saida da Rede (tanh): {output[0]:.4f}")
    print(f"     - Classificacao: **{classification}**")
```

```
    print("\n" + "=" * 70)

def generate_pdf_report(code_content, output_content):
    """Gera um PDF com o código e o output."""
    pdf = FPDF()
    pdf.add_page()

    pdf.set_font("Courier", 'B', 16)
    pdf.cell(0, 10, 'Questao 4', new_x=XPos.LMARGIN, new_y=YPos.NEXT, align='C')
    pdf.ln(10)

    pdf.set_font("Courier", 'B', 12)
    pdf.cell(0, 10, 'Codigo Fonte:', new_x=XPos.LMARGIN, new_y=YPos.NEXT)
    pdf.set_font("Courier", '', 8)
    pdf.multi_cell(0, 5, code_content)

    pdf.add_page()

    pdf.set_font("Courier", 'B', 12)
    pdf.cell(0, 10, 'Output da Execucao:', new_x=XPos.LMARGIN, new_y=YPos.NEXT)
    pdf.set_font("Courier", '', 9)
    output_content_safe = output_content.encode('latin-1', 'replace').decode('latin-1')
    pdf.multi_cell(0, 5, output_content_safe)

    pdf_file_name = "resultado_questao_4.pdf"
    pdf.output(pdf_file_name)
    return pdf_file_name

# --- Bloco Principal de Execução ---
if __name__ == "__main__":
    # 1. Captura todo o output da função principal para uma variável
    output_buffer = io.StringIO()
    with redirect_stdout(output_buffer):
        solve_and_capture_output()
    output_content = output_buffer.getvalue()

    # 2. (NOVO) Imprime o conteúdo capturado diretamente no terminal
    print("--- [INICIO] Resultado da Execucao no Terminal ---")
    print(output_content)
    print("--- [FIM] Resultado da Execucao no Terminal ---")

    # 3. Lê o conteúdo do próprio script para colocar no PDF
    try:
        with open(__file__, 'r', encoding='utf-8') as f:
            code_content = f.read()
    except Exception as e:
        code_content = f"Nao foi possivel ler o arquivo do codigo: {e}"

    # 4. Gera o arquivo PDF com o código e o output capturado
    try:
        pdf_file = generate_pdf_report(code_content, output_content)
        print(f"\nPDF '{pdf_file}' gerado com sucesso no diretorio atual!")
```

```
except Exception as e:
    print(f"\nOcorreu um erro ao gerar o PDF: {e}")
```

Output da Execucao:

```
Questao 4: Treinamento e Teste de Rede Neural (Perceptron)
========================================================================
Parte (a): Treinamento da Rede Neural
------------------------------------------------------------------------
1. Preparacao dos Dados

Dados de Treinamento Brutos (X):
[[370   7]
 [220   3]
 [180   4]
 [400   7]
 [380   8]
 [410   9]
 [220   2]
 [170   3]]

Labels de Treinamento (y): [Laranja=1, Limao=-1]
[ 1 -1 -1  1  1  1 -1 -1]

2. Normalizacao dos Dados de Entrada (X)
   - Minimos (Peso, Acidez): [170   2]
   - Maximos (Peso, Acidez): [410   9]

Dados de Treinamento Normalizados:
[[0.83333333 0.71428571]
 [0.20833333 0.14285714]
 [0.04166667 0.28571429]
 [0.95833333 0.71428571]
 [0.875      0.85714286]
 [1.         1.        ]
 [0.20833333 0.        ]
 [0.         0.14285714]]

3. Treinamento com a Regra Delta
   - Taxa de Aprendizagem (alpha): 0.1
   - Epocas de Treinamento: 2000
   - Pesos Iniciais (w1, w2): [-0.01254599  0.04507143]
   - Bias Inicial (w0): [0.02319939]

Treinamento Concluido.

4. Pesos e Bias Finais Calculados:
   - Peso w1 (relativo a 'Peso da Fruta'): 3.6327
   - Peso w2 (relativo a 'Acidez'):      2.8643
   - Bias w0:                            -3.1031

========================================================================
Parte (b): Teste da Rede com Novos Dados
------------------------------------------------------------------------
1. Dados de Teste Brutos:
[[380   2]
```

```
 [170   8]]

2. Normalizacao dos Dados de Teste (usando min/max do treino):
[[0.875      0.        ]
 [0.         0.85714286]]

3. Resultados da Classificacao:

   - Fruta 1 (Entrada Bruta: [380   2])
     - Saida da Rede (tanh): 0.0754
     - Classificacao: **Laranja**

   - Fruta 2 (Entrada Bruta: [170   8])
     - Saida da Rede (tanh): -0.5703
     - Classificacao: **Limao**

======================================================================
```