[PSI3471](https://psi3471.github.io/)

Search

* [Home](https://psi3471.github.io/)
* [Teoria](https://psi3471.github.io/ex_entrega_2.html#)
  + [Introdução](https://psi3471.github.io/introducao.html)
  + [Regressão linear](https://psi3471.github.io/regressao_linear.html)
  + [O algoritmo LMS](https://psi3471.github.io/lms.html)
  + [O modelo do neurônio](https://psi3471.github.io/neuronio.html)
  + [A rede perceptron multicamada](https://psi3471.github.io/mlp.html)
  + [Evitando mínimos locais e *overfitting*](https://psi3471.github.io/hiperparametros.html)
  + [Medidas de desempenho](https://psi3471.github.io/medidas.html)
  + [Rede neural convolucional](https://psi3471.github.io/cnn.html)
* [Exercícios para aula](https://psi3471.github.io/ex_entrega_2.html#)
  + [Exercício 1 - Regressão Linear](https://psi3471.github.io/ex_aula_1.html)
  + [Exercício 2 - O algoritmo LMS](https://psi3471.github.io/ex_aula_2.html)
  + [Exercício 3 - Neurônio de Rosenblatt no modo mini-batch](https://psi3471.github.io/ex_aula_3.html)
  + [Exercício 4 - MLP e *backpropagation*](https://psi3471.github.io/ex_aula_4.html)
  + [Exercício 5 - MLP e o otimizador Adam](https://psi3471.github.io/ex_aula_5.html)
  + [Exercício 6 - MLP com PyTorch](https://psi3471.github.io/ex_aula_6.html)
  + [Exercício 7 - CNN com PyTorch](https://psi3471.github.io/ex_aula_7.html)
* [Exercícios para entrega](https://psi3471.github.io/ex_entrega_2.html#)
  + [Exercício 1 - Regressão Linear](https://psi3471.github.io/ex_entrega_1.html)
  + [Exercício 2 - Perceptron de Rosenblatt](https://psi3471.github.io/ex_entrega_2.html)
  + [Exercício 3 - MLP](https://psi3471.github.io/ex_entrega_3.html)
  + [Exercício 4 - CNN](https://psi3471.github.io/ex_entrega_4.html)
  + [Avaliação - Parte Prática](https://psi3471.github.io/ex_avaliacao.html)
* [Material de apoio](https://psi3471.github.io/ex_entrega_2.html#)
  + [Tópicos de programação com Python](https://psi3471.github.io/python_videos.html)
  + [Exemplo MLP com PyTorch](https://psi3471.github.io/exemplo_pytorch.html)
  + [Exemplo CNN com PyTorch](https://psi3471.github.io/exemplo_pytorch_cnn.html)
* [Moodle](https://edisciplinas.usp.br/enrol/index.php?id=106947)

Exercício 2 - Perceptron de Rosenblatt

Um dos experimentos feitos com o perceptron de Rosenblatt consistia na classificação de caracteres. Um caractere era exposto na frente de uma matriz de sensores, que digitalizava a imagem e os elementos dessa matriz eram utilizados como entradas do perceptron, conforme mostrado na fotografia abaixo:



Rosenblatt’s Classifier

A ideia consistia em alimentar diversos perceptrons com essas entradas, que funcionariam em paralelo, de forma independente uns dos outros. Cada um era treinado para classificar um determinado caractere, fornecendo saída igual a 1 no caso da detecção do caractere em questão ou -1, no caso da detecção de qualquer outro caractere. Assim, era possível obter uma estrutura com várias saídas, cada uma representando o reconhecimento de um determinado caractere.

Neste exercício, você vai trabalhar em uma aplicação semelhante, porém simplificada: um classificador binário usando o perceptron de Rosenblatt para classificar imagens de círculos ou quadrados.

Antes de começar a trabalhar no código, certifique-se de ter baixado o [arquivo CSV com os dados](https://media.githubusercontent.com/media/psi3471/datasets/main/circles_vs_rects_dataset/circles_and_squares.csv) do repositório de materiais.

Para carregar o arquivo CSV e processar os dados, você pode usar a biblioteca [Pandas](https://pandas.pydata.org/). Após a instalação da biblioteca e o download do arquivo CSV, é possível carregar os dados com os seguintes comandos:

import pandas as pd  
data\_df = pd.read\_csv("circles\_and\_squares.csv")

Com esses comandos, você vai criar um *DataFrame* do Pandas chamado data\_df contendo os dados do arquivo CSV. Para ver algumas linhas do *DataFrame*, você pode usar o método .head():

data\_df.head()

|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 391 | 392 | 393 | 394 | 395 | 396 | 397 | 398 | 399 | 400 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | -1.0 |
| 1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | -1.0 |
| 2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | -1.0 |
| 3 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | -1.0 |
| 4 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |

5 rows × 401 columns

É possível converter o *DataFrame* em um *array* NumPy usando o método .to\_numpy():

data = data\_df.to\_numpy()  
data\_df.shape

(1000, 401)

A matriz de dados tem 1000 linhas e 401 colunas. Cada linha se refere a um exemplo no banco de dados. As 400 primeiras colunas se referem aos elementos de uma matriz 20x20 com os dados referentes à digitalização de uma imagem de um círculo ou de um quadrado. A última coluna é um rótulo igual a -1 ou 1 indicando se a imagem se refere a um círculo ou a um quadrado.

Você pode visualizar uma imagem, usando o método .reshape() para reorganizar os 400 coeficientes em uma matriz 20x20 e a função imshow() para exibí-la. Por exemplo, para exibir a imagem referente à primeira linha da matriz:

import matplotlib.pyplot as plt  
img = data[0, :-1].reshape((20,20))  
plt.imshow(img, cmap='gray')

<matplotlib.image.AxesImage at 0x7f0127f3fd60>



O rótulo referente à imagem pode ser obtido com o elemento da última coluna:

data[0, 400]

-1.0

Como você pode observar, o rótulo igual a -1 é usado para identificar os círculos. Além disso, vale notar também que a resolução de 20x20 considerada na digitalização da figura do círculo causa uma distorção considerável. No entanto, vamos utilizar essa resolução baixa para reduzir a dimensionalidade do problema e tornar o treinamento mais rápido.

A quinta linha da matriz (índice 4) representa um quadrado:

img = data[4, :-1].reshape((20,20))  
print(data[4, -1])  
plt.imshow(img, cmap='gray')

1.0

<matplotlib.image.AxesImage at 0x7f0127dd3d00>



No exercício, você vai utilizar as 800 primeiras linhas do banco de dados para fazer o treinamento do perceptron, ajustando os valores dos pesos de forma iterativa e as 200 linhas finais para avaliar o desempenho do seu modelo na classificação de círculos e quadrados. Dessa forma, você pode separar os dados em uma matriz Xd para o treinamento e Xd\_test para o teste:

Xd = data[:800, :]  
Xd\_test = data[800:, :]

Você deve implementar uma função para treinar o perceptron de Rosenblatt de forma iterativa, dados a matriz Xd, um passo de adaptação eta, o número de épocas Ne e o tamanho do *mini-batch* Nb. Após o treinamento, avalie o seu modelo com os dados dos conjuntos de treinamento e de teste, medindo a acurácia do classificador para ambos os casos. A escolha desses hiperparâmetros fica a seu critério, buscando obter um bom desempenho com os dados de teste.

Ao final do exercício, você deverá apresentar:

1. Os códigos utilizados para treinar o modelo e avaliá-lo com os conjuntos de treinamento e de teste;
2. O valor obtido para a acurácia considerando os dados de teste.

A sugestão é que o relatório seja elaborado utilizando um Jupyter Notebook e a linguagem Python, já que essas são as ferramentas que estamos utilizando neste bloco do curso. No entanto, isso não é obrigatório e você pode usar outra linguagem de programação, caso queira.

## Instruções para entrega

* O exercício pode ser feito em dupla ou individualmente;
* A entrega deve incluir:
  + Um vídeo de no **máximo 40s**, mostrando a resolução do exercício;
  + Os **códigos-fontes** dos programas, preferencialmente organizados em um Jupyter Notebook, descrevendo o experimento e mostrando como foram obtidos os resultados solicitados.
* **A correção será feita baseada no vídeo**. Quando o professor/pesquisador ficar com alguma dúvida, serão consultados os códigos-fonte;
* Sobre o vídeo:
  + **Deve incluir áudio** descrevendo o experimento;
  + Gravem a tela do computador usando celular ou usando algum programa de captura de tela (por exemplo Zoom, Google Meet, ou OBS Studio);
  + No início, **deve aparecer o rosto e algum documento do aluno que gravou o vídeo** (como a carteira USP, RG, CNH, etc);
  + No caso de entrega em dupla, **não é necessário que os dois componentes apareçam no vídeo**. No entanto, alternem o apresentador ao longo das entregas dos exercícios e **não esqueçam de incluir os dois nomes no início do vídeo**.
  + Procurem convencer o espectador do vídeo, que vai corrigir o exercício que fizeram os exercícios computacionais solicitados e que eles estão funcionando corretamente. Tentem fazer um bom aproveitamento do tempo para apresentar os resultados solicitados, **respeitando o limite de 40s e não acelerem a velocidade do vídeo**;
* Sobre os códigos-fonte:
  + **Incluir o nome do(s) aluno(s)** no início do programa;
* Sobre o envio no Moodle:
  + Apenas um aluno de cada dupla deve enviar o vídeo no Moodle;
  + Podem ser enviados o arquivo de vídeo (.mkv, .mp4, .avi, etc.) ou um link para o vídeo (Youtube, Google Drive, etc);
    - No segundo caso, certifiquem-se que todos os professores/pesquisadores (magno.silva@usp.br, hae.kim@usp.br, renatocan@lps.usp.br, wesleybeccaro@usp.br) tenham acesso ao seu vídeo.
  + Não se esqueçam de escrever o nome dos componentes da dupla (ou do único aluno, escrevendo: “exercício feito individualmente”) em três lugares diferentes: **no campo “comentários sobre o envio” no Moodle**, **no início do vídeo** e no **início dos códigos-fonte**.