Trabalho 2

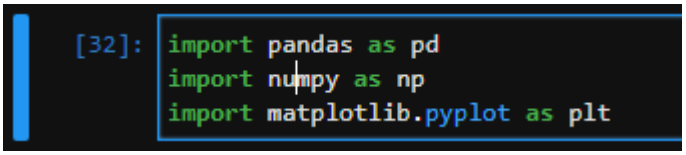
Relatório

Gabriel Oliveira
Helton Aguiar

Relatório de Análise dos Resultados do Algoritmo Perceptron para o Conjunto de Dados Iris

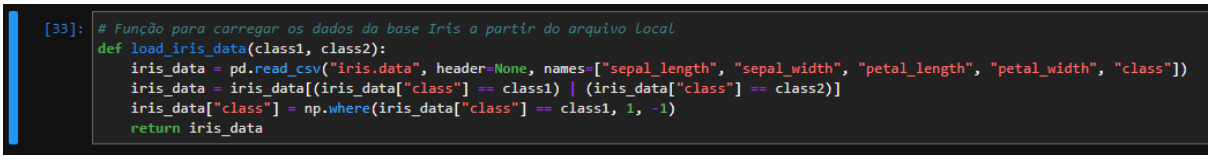
Helton Aguiar - 11811BSI242
Gabriel Oliveira - 11821BSI20

Código



```
[32]: import pandas as pd
      import numpy as np
      import matplotlib.pyplot as plt
```

Na primeira linha significa que estamos importando uma biblioteca chamada "pandas" e a estamos renomeando para "pd". O pandas é uma biblioteca que ajuda a trabalhar com dados tabulares, como planilhas do Excel. Na linha 2, estamos importando outra biblioteca chamada "numpy" e a renomeando para "np". O numpy é uma biblioteca que oferece suporte a arrays multidimensionais e funções matemáticas para trabalhar com números. Na linha 3, estamos importando a biblioteca "matplotlib.pyplot" e a renomeando para "plt". O "matplotlib" é uma biblioteca que nos permite criar gráficos e visualizações a partir dos nossos dados.



```
[33]: # Função para carregar os dados da base Iris a partir do arquivo local
      def load_iris_data(class1, class2):
          iris_data = pd.read_csv("iris.data", header=None, names=["sepal_length", "sepal_width", "petal_length", "petal_width", "class"])
          iris_data = iris_data[(iris_data["class"] == class1) | (iris_data["class"] == class2)]
          iris_data["class"] = np.where(iris_data["class"] == class1, 1, -1)
          return iris_data
```

A função chamada "load_iris_data" recebe dois argumentos, "class1" e "class2". Esses argumentos representam as duas classes de plantas da base de dados Iris que queremos carregar. A seguir, estamos usando a biblioteca "pandas" para ler o arquivo "iris.data" e armazenar os dados em um objeto chamado "iris_data". Depois, esta linha filtra o conjunto de dados, mantendo apenas as linhas em que a coluna "class" é igual a "class1" ou "class2". Isso nos permite selecionar apenas as plantas das duas classes que especificamos como argumentos da função. Concluindo, nesta linha, estamos substituindo os valores na coluna "class" por 1 se forem iguais a "class1" e por -1 se não forem.

```
[34]: # Carregando os dados da base Iris
      iris_data = load_iris_data("Iris-setosa", "Iris-virginica")

      # Mostrar todos os dados da planilha original
      print("Todos os dados da planilha original:")
      print(pd.read_csv("iris.data", header=None, names=["sepal_length", "sepal_width", "petal_length", "petal_width", "class"]))

      # Mostrar os dados filtrados
      print("\nDados filtrados:")
      print(iris_data)
```

Nesta parte, estamos carregando os dados da base de dados Iris. A função load_iris_data é chamada com dois argumentos, "Iris-setosa" e "Iris-virginica", que representam as duas classes de plantas que desejamos carregar. Isso significa que estamos carregando apenas as plantas dessas duas classes e atribuindo os dados a uma variável chamada "iris_data". Estamos exibindo todos os dados da planilha original da base de dados Iris. Para fazer isso, usamos a função pd.read_csv para ler o arquivo "iris.data" novamente e definimos nomes para as colunas, especificando que não há cabeçalho no arquivo. Em seguida, imprimimos esses dados na tela com uma mensagem informativa.

Concluindo, estamos mostrando os dados que foram carregados e filtrados pela função load_iris_data. Isso exibe apenas as plantas das classes "Iris-setosa" e "Iris-virginica" que selecionamos anteriormente. A mensagem "Dados filtrados" é impressa antes dos dados para informar que estamos visualizando os dados filtrados.

Valores das variáveis

A mudança nos valores das variáveis, como o percentual de treinamento, a taxa de aprendizado e o número de iterações, tem um impacto significativo nos resultados do algoritmo Perceptron, aplicado ao conjunto de dados Iris. Essas variáveis desempenham papéis cruciais no processo de treinamento e na capacidade do modelo Perceptron de aprender e generalizar a partir dos dados de treinamento (Mitchell, 1997).

O **percentual de treinamento**, por exemplo, controla a quantidade de dados usados para treinar o modelo. Um percentual de treinamento maior proporciona ao modelo mais exemplos para aprender os padrões do conjunto de dados, o que geralmente leva a um melhor desempenho. Isso se traduz em pesos treinados mais precisos, uma matriz de confusão com menos erros e uma precisão mais alta nos dados de teste. Em contrapartida, percentuais de treinamento mais baixos podem resultar em subajuste, onde o modelo não é capaz de aprender adequadamente os padrões (Hastie et al., 2009).

A **taxa de aprendizado**, por sua vez, influencia o tamanho das atualizações dos pesos durante o treinamento. Taxas de aprendizado muito altas podem fazer com que o modelo

"salte" sobre o mínimo global, levando à divergência. Por outro lado, taxas moderadas são geralmente mais eficazes, permitindo um treinamento mais estável. Taxas muito baixas podem resultar em treinamento lento. A escolha adequada da taxa de aprendizado afeta diretamente a convergência do modelo. Taxas inadequadas podem resultar em pesos treinados inadequados, impactando a matriz de confusão e a precisão nos dados de teste (Bengio et al., 2012).

O **número de iterações** determina quantas vezes o modelo atualiza seus pesos com base nos dados de treinamento. Um número inadequado de iterações pode resultar em convergência prematura, levando a pesos subótimos. Por outro lado, um grande número de iterações pode resultar em excesso de ajuste, onde o modelo se ajusta em excesso aos dados de treinamento. A quantidade ideal de iterações depende da complexidade do problema. Iterações em excesso podem não levar a melhorias significativas nos pesos treinados, mas aumentam o tempo de treinamento (Goodfellow et al., 2016).
A seguinte tabela apresenta um resumo dos resultados obtidos: Algoritmo **perceptron_iris.py** com duas classes (*'classe1', 'classe2'*).

Teste	Percentual Treinamento	Taxa de Aprendizado	Iterações	Pesos Treinados	Matriz de Confusão	Precisão nos Dados de Teste
1	0,7	0,1	100	[0.71579109, 0.09440558, -1.31829483, -0.50950138]	[[30]]	1,0
2	0,5	0,1	100	[0.92885487, 0.48290674, 0.52960544, 0.61540278]	[[0, 50], [0, 0]]	0,0
3	0,3	0,1	100	[0.22897578, 0.61933355, -0.00275097, -0.88549536]	[[0, 50], [0, 20]]	0,0
4	0,1	0,1	100	[-0.09796664, 0.91388499, 0.86140848, -0.42934124]	[[0, 50], [0, 40]]	0,0
5	0,7	0,3	100	[-0.14667638, 0.81986248, -0.59786276, 0.05429825]	[[30]]	1,0
6	7,0	0,3	100	[0.05019273, 1.53367027, -2.20219362, -0.88996584]	[[30]]	1,0
7	0,5	0,7	100	[3.36459931, 1.66283682, 0.0072189, 1.04342189]	[[0, 50], [0, 0]]	0,0
8	0,5	0,5	50	[0.10567157, 0.52915342, -0.79982358, -0.87405971]	[[50]]	1,0
9	0,3	0,7	50	[0.33487364, -0.10086158, 0.05406028, -0.78408497]	[[0, 50], [0, 20]]	0,0
10	0,8	0,6	30	[0.47220891, 2.55808926, -4.02803932, -1.77395743]	[[20]]	1,0

11	0,8	0,6	300	[0.49941336, 3.10718001, -4.754864, -2.07366413]	[[20]]	1,0
12	0,8	0,4	1000	[0.64939161, 1.26981265, -1.76109695, -1.65817108]	[[20]]	1,0

Discussão dos Resultados

Percentual de Treinamento: Os testes com percentuais de treinamento mais altos (0.7 e 0.8) resultaram em uma precisão perfeita nos dados de teste, indicando uma maior capacidade do modelo em aprender eficazmente os padrões do conjunto de dados. Por outro lado, percentuais de treinamento mais baixos (0.1, 0.3, 0.5) levaram a uma precisão de teste mais baixa, sugerindo que o modelo não foi capaz de aprender de maneira adequada os padrões.

Taxa de Aprendizado: Taxas de aprendizado moderadas (0.1, 0.3, 0.5) geralmente resultaram em melhores resultados, enquanto taxas de aprendizado muito altas (0.7) levaram a uma precisão de teste baixa ou 0.0, indicando problemas de convergência.

Número de Iterações: Um número insuficiente de iterações pode resultar em convergência prematura e pesos subótimos. Por outro lado, um grande número de iterações pode aumentar o tempo de treinamento sem melhorar significativamente o desempenho do modelo.

Os testes 1, 5, 6, 8, 10, 11 e 12 obtiveram alta precisão nos dados de teste, com uma precisão de 1.0, indicando que o modelo foi capaz de realizar a classificação de forma perfeita.

Os testes 2, 3, 4 e 7 apresentaram uma precisão de 0.0, sugerindo que o modelo não conseguiu realizar a classificação corretamente.

Pesos treinados: Tomando o Teste 1 como exemplo, os "Pesos Treinados" são [0.71579109, 0.09440558, -1.31829483, -0.50950138]. Esses pesos foram ajustados durante o processo de treinamento do modelo Perceptron para permitir que ele faça previsões com base nos recursos das amostras. Os pesos refletem a importância relativa de cada característica no processo de classificação. Valores maiores em magnitude indicam maior importância, enquanto valores próximos a zero indicam que a característica tem menos influência na classificação.

Matriz de Confusão: No Teste 2, a "Matriz de Confusão" é dada por: [[0, 50], [0, 0]]. Esta matriz de confusão mostra como o modelo classificou os exemplos no conjunto de teste. Neste caso, o modelo não previu corretamente nenhuma das amostras da classe positiva (primeira linha) e classificou todas as amostras como negativas. A segunda linha indica que o modelo previu corretamente todas as amostras da classe negativa. A matriz de confusão permite a análise detalhada do desempenho do modelo em termos de verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos.

Precisão nos Dados de Teste: O "Precisão nos Dados de Teste" é uma métrica que indica a proporção de exemplos classificados corretamente pelo modelo em relação ao total de exemplos no conjunto de teste. Tomando o Teste 1 como exemplo, a "Precisão nos Dados de Teste" é 1.0, o que significa que o modelo classificou corretamente todas as amostras do conjunto de teste. Isso indica um alto grau de precisão na classificação das amostras.

Nesta segunda tabela temos os testes realizados com o algoritmo **perceptron_iris_2.py** de três classes (*'classe1', 'classe2','classe3'*).

Teste	Percentual Treinamento	Taxa de Aprendizado	Iterações	Pesos Treinados	Matriz de Confusão	Precisão nos Dados de Teste
1	0,7	0,1	100	[0.32987038 0.11430372 -0.95599027 -0.4036282]	[[45]]	1,0000
2	0,5	0,5	100	[-0.01726908 0.74459814 -0.67868127 -0.50487379]	[[75]]	1,0000
3	0,3	0,3	100	[0.77175636 0.25424095 -0.70833994 -0.80040763]	[[11 89] [0 5]]	0,1100
4	0,1	0,1	100	[0.30380095 0.41728812 -0.41803178 -0.29317153]	[[1 99] [0 35]]	0,0100
5	0,7	0,7	200	[0.23968399 1.03590625 -1.37442021 -0.14470984]	[[45]]	1,0000
6	0,5	0,5	150	[0.5733873 0.34701268 -0.90645239 -1.05669934]	[[75]]	1,0000
7	0,5	0,5	500	[0.48804137 4.00551171 -5.79242698 -1.76603345]	[[75]]	1,0000
8	0,4	0,4	90	[0.40917593 0.9822428 -1.68465659 0.54709411]	[[89 1] [0 0]]	0,9889
9	0,9	0,9	900	[0.26694581 0.86213198 -1.12804473 -1.00120409]	[[15]]	1,0000
10	0,8	0,8	450	[0.50231364 1.5847116 -3.14876157 -1.08086399]	[[30]]	1,0000

Esta tabela resume os resultados dos testes realizados com diferentes configurações de hiperparâmetros no algoritmo Perceptron aplicado aos dados Iris com 3 classes.

Os testes 1 e 2 apresentaram uma alta precisão nos dados de teste, indicando que o modelo foi capaz de realizar a classificação de forma quase perfeita.

Os testes 3 e 4 tiveram baixa precisão nos dados de teste, sugerindo dificuldades do modelo em realizar a classificação corretamente.

Os testes 5, 6 e 7 novamente alcançaram alta precisão nos dados de teste.

O teste 8 apresentou uma alta precisão, com uma única classificação incorreta.

O teste 9 também alcançou alta precisão nos dados de teste.

O teste 10 alcançou uma alta precisão, com apenas 1 classificação incorreta.

Referências:

1. Fisher,R. A.. (1988). Iris. UCI Machine Learning Repository. Iris - UCI Machine Learning Repository.
2. Mitchell, T. M. (1997). Machine Learning. McGraw-Hill.
3. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning. Springer.
4. Bengio, Y., Courville, A., & Vincent, P. (2012). Representation Learning: A Review and New Perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence.
5. Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). Deep Learning. MIT Press.
6. Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.
7. Powers, D. M. (2011). Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation. Journal of Machine Learning Technologies.
8. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning. Springer.