Implementação de Rede Percepetron

1st Hans Herbert Schulz
Programa de Pós-Graduação em Engenharia de Sistemas Eletrônicos
Universidade Federal de Santa Catarina
Joinville, Brasil
hansherbert99@gmail.com

I. Introdução

O presente relatório visa demonstrar o processo de formulação de uma rede Percepetron de camada única e seus respectivos testes e treinamentos em bases de dados linearmente e não linearmente separáveis.

II. Formulação Teórica

No contexto do desenvolvimento de redes neurais e problemas de classificação, pode-se inferir que um Percepetron é a instância mais singela possível. Para melhor compreender a essência do mesmo, compara-se um Percepetron um neurônio presente no cérebro humano.

O Percepetron aceita múltiplas entradas e estas entradas estarão associadas a um 'peso' que representa a importância que a referida entrada tem na saída desejada. O resultado da somatória das entradas será submetido a uma função de ativação e a mesma fornecerá o resultado de saída do Percepetron. Deve-se então avaliar o resultado obtido em relação ao resultado esperado na fase de treinamento e realizar um processo iterativo para ajustar os 'pesos' até que a resposta encontrada seja tão próxima quanto se queira da resposta esperada. A Figura 1 abaixo ilustra a estrutura de um Percepetron.

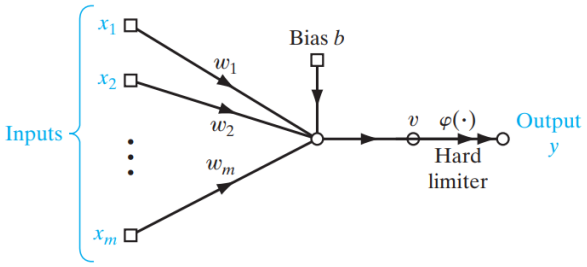

Fig. 1. Ilustração estrutura de Percepetron.[1]

III. Implementação

A implementação foi realizada na linguagem Python na versão 3.8.10. Visando generalidade e facilidade de manipulação do objeto, foram criadas 2 classes e 1 função respectivamente:

- *neuronio*;
- *layer*;
- *treinar*;

Na classe *neuronio*, utilizou-se um conjunto de funções que atribuem o peso às entradas, calculam o campo induzido e retornam a resposta do Percepetron. Já na classe *layer*, define-se a quantidade de neurônios utilizados em uma única camada. Finalmente, a classe *validar* possuem um conglomerado de funções que randomizam os pesos a cada iteração para evitar o fenômeno de *overfitting*, comparam as respostas encontradas com as respostas esperadas, retornam o erro entre as respostas previamente citadas e ajustam o peso para cada entrada de acordo com equação 1 abaixo:

$$w_{n+1} = w_n + \eta_n(d_n - y_n)x_n \tag{1}$$

onde o termo η simboliza a taxa de aprendizado, y_n é a saída obtida, d_n é a resposta esperada e x_n é o valor de entrada.

IV. Resultados

A. Base linearmente separável

Após a implementação inicial, foi necessário avaliar a qualidade de classificação do Percepetron. Para isso, foi criada uma base de dados linearmente separável fictícia com duas classes e duas características. As classes são 'laranja' e 'maçã' e as respectivas características a serem avaliadas são o pH de cada fruta e sua massa em gramas. Para realizar o processo de classificação propriamente dito, inicializou-se uma camada denominada 'cerebro' que possui dois neurônios (um para identificar cada classe individualmente). O critério de parada estabelecido foi o erro quadrático médio ser menor que 0,001 ou o número de épocas igual à 100.000. O erro foi 'printado' em tela a cada 1000 épocas. Utilizou-se ainda 3 valores de eta para a comparação do erro: $\eta = 0,01$, $\eta = 0,05$ e $\eta = 0,1$. Adicionalmente, utilizou-se a função sigmoide como função de ativação padrão e testou-se também a função degrau para a ativação. Em seguida, utilizou-se a biblioteca *matplotlib* para plotar a reta que separa ambas as classes. O resultado obtido está demonstrado na Figura 2.

Para uma base de dados fictícia com poucos dados como a descrita neste experimento, é esperado que haja flutuação considerável durante o processo iterativo. Observa-se nas Figuras 3 e 4 esta flutuação e ademais, percebe-se a relação diretamente proporcional entre o valor atribuído para η no erro obtido. Adicionalmente, a Tabela 1 mostra a quantidade de iterações necessárias para convergência.

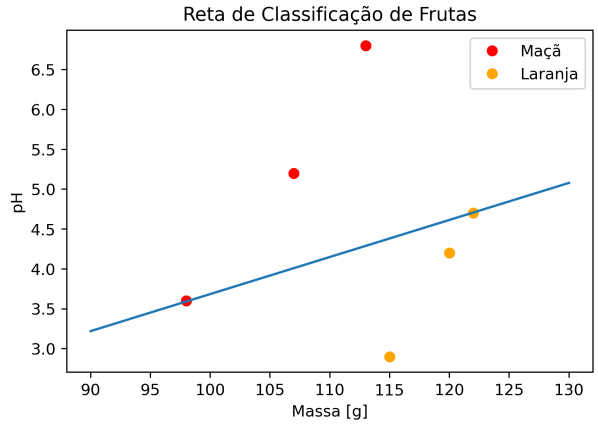

Fig. 2. Classificação obtida pelo Percepetron

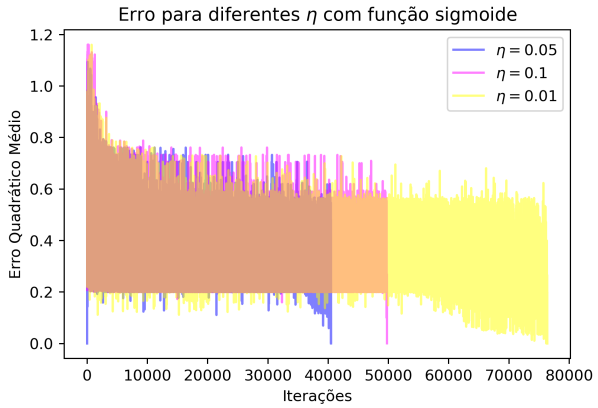

Fig. 3. Comparação do erro para diferentes valores de η utilizando a função sigmoide para ativação.

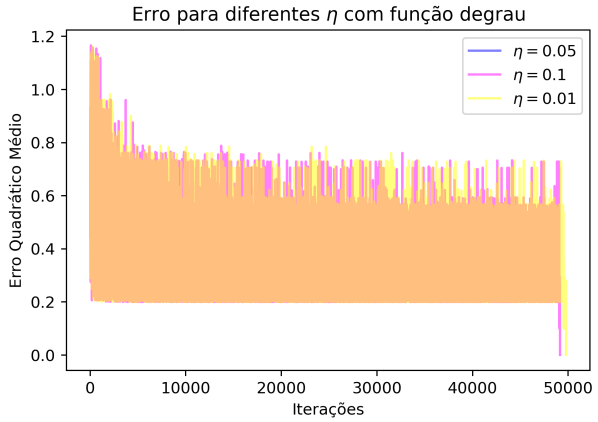

Fig. 4. Comparação do erro para diferentes valores de η utilizando a função degrau para ativação.

TABLE I
NÚMERO DE ITERAÇÕES PARA DIFERENTES η_s

Valores de η	Iterações Sigmoide	Iterações Degrau
0.01	76368	49781
0.05	40479	1
0.1	49764	49154

Após observar a reta obtida e o erro quadrático médio entre a resposta esperada e a reposta obtida, infere-se visualmente que houve convergência do Percepetron e a classificação foi realizada corretamente.

B. *Extensão da base linearmente separável*

Após utilizar a base mencionada acima para uma formulação inicial, criou-se uma nova base linearmente separável com 49 elementos a serem classificados e cada elemento possuindo também 2 atributos. A nova base foi intitulada de *citrus*. Em seguida, aplicou-se a nova base ao Percepetron com os mesmos parâmetros citados no experimento anterior. A Figura 5 representa a reta de classificação, as Figuras 6 e 7 o erro plotado para diferentes valores de η e funções de ativação e a Tabela 2 mostra o número de iterações para cada valor.

TABLE II
NÚMERO DE ITERAÇÕES PARA DIFERENTES η_s PARA BASE COM 49 ELEMENTOS

Valores de η	Iterações Sigmoide	Iterações Degrau
0.01	49389	40184
0.05	42220	40532
0.1	41335	40623

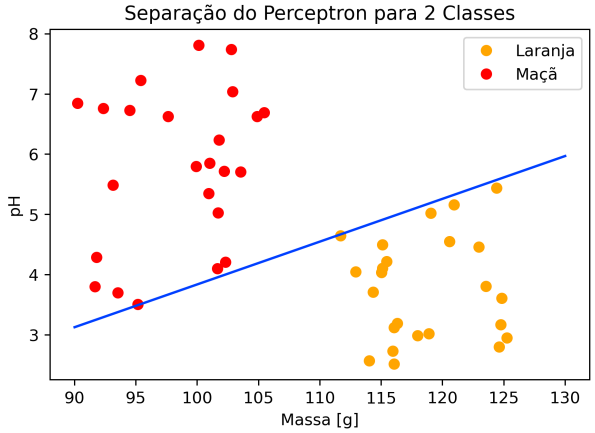

Fig. 5. Classificação obtida pelo Percepetron para base com 49 elementos.

C. *Base não-linearmente separável*

A seguir, o Percepetron foi testado para a base de dados *Iris* que consiste de três classes): *Iris-setosa, Iris-versicolor e Iris-virgínica*. Cada classe possui quatro atributos:comprimento de

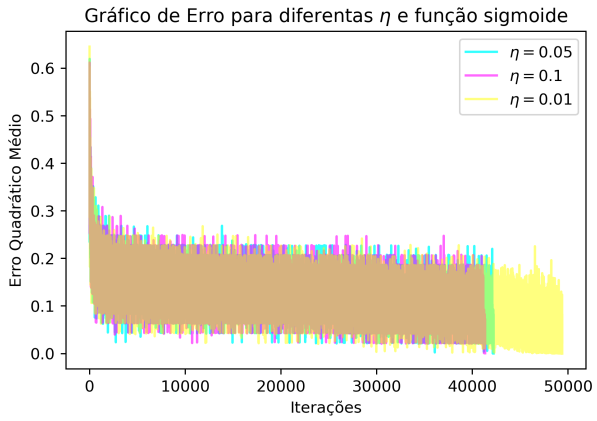

Fig. 6. Erro por iteração para cada valor de η para base com 49 elementos e função sigmoide.

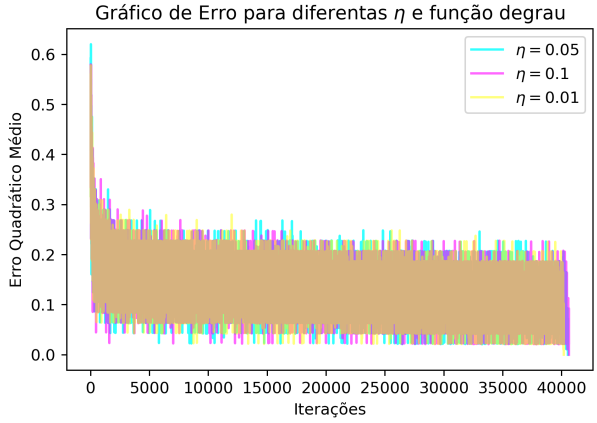

Fig. 7. Erro por iteração para cada valor de η para base com 49 elementos e função degrau.

pétala, largura de pétala, comprimento de sépala, e largura de sépala.

Tendo em vista que a base não é linearmente separável, entende-se que o Percepetron não convergirá totalmente. Diante disso, optou-se por utilizar apenas 10000 épocas e uma taxa de aprendizado $\eta = 0.2$. A função de ativação utilizada foi a função sigmoide novamente. Por se tratar de uma base linearmente não separável com 3 classes, não convém realizar a representação gráfica da mesma. Adicionalmente, foi definido um valor de *threshold*, ou seja, uma espécie de "tolerância" de classificação de 0.8. Em suma, a função deve classificar entre 0 ou 1, porém o valor de *threshold* considerará valores a partir 0.8 como 1 para a classificação. A qualidade dos resultados será avaliada na seção subsequente.

V. Validação

A. Avaliação do Percepetron por Bootstrap

Para definir a qualidade de avaliação do Percepetron, optou-se por utilizar o método de re-amostragem denominado "bootstrap" apresentado por Efron e Tibshirani em 1993. Em suma, o método utiliza uma amostra como população finita e, a partir dela, novas amostras aleatórias são geradas para estimar características da população e portanto, poder avaliar a qualidade de classificação obtida.

Para a base de dados linearmente separável artificial *citrus*, o resultado está disposto na Tabela 3.

TABLE III
Resultado da avaliação por Bootstrap

Erro	Desvio Padrão	Iterações Bootsrap
5.91%	4.10%	10

Diante do exposto, pode-se concluir que o Percepetron classifica corretamente 94.09% dos casos e tem uma margem de erro de 2.05% para mais ou para menos.

Já para a base linearmente não separável *Iris*, o procedimento do método Bootstrap foi realizado apenas três vezes, uma vez que, o tempo requerido para cada iteração é consideravelmente maior visto que o Percepetron tende a não convergir. Os dados encontrados estão dispostos na Tabela 4.

TABLE IV
Resultado da avaliação por Bootstrap

Erro	Desvio Padrão	Iterações Bootsrap
16.89%	10.45%	3

Os dados expostos na Tabela 4 mostram que, mesmo para uma base linearmente não separável, a qualidade dos resultados do Percepetron aparentam ser bem otimistas. O neurônio classifica corretamente 83.10% dos objetos da base de dados e possui uma margem de erro de 5.22% para mais ou para menos.

É pertinente destacar que, ao observar os valores fornecidos após a aplicação dos neurônios, notou-se que na região de interfaces das classes, o Percepetron apresentou resultados pouco precisos e em alguns casos, não atribuiu o objeto a nenhuma classe.

VI. Considerações Finais

A. Base Linearmente Separável

Após realizar o processo de treinamento em ambas as bases linearmente separáveis, nota-se que a quantidade de iterações reduz-se substancialmente quando a quantidade de objetos é expandida. Ademais, é interessante ressaltar que o valor de erro máximo obtido cai pela metade conforme observa-se na escala dos gráficos das Figuras 3 e 5.

No tocante às funções de ativações utilizadas, é possível observar que, até para valores de η refinados, o número necessário de iterações foi semelhante às demais taxas de aprendizado quando utilizou-se a função degrau. O número maior de iterações para pequenos valores de η deve-se ao fato de a sigmoide ser assintótica, ou seja, jamais atinge o valor de zero. Sendo assim, a resposta vai vagarosamente se aproximando do valor de zero até atingir a tolerância desejada.

B. Base Linearmente Não Separável

Após realizar as análises com o método bootstrap, concluise que o erro obtido durante o processo de classificação é consideravelmente maior do que para bases linearmente separáveis. Contudo, tendo em vista a abordagem de problemas multiclasse utilizada, e portanto, a expectativa de encontrar um alto valor para o erro, o Perceptron aparenta gerar resultados relativamente precisos.

Em um estudo futuro, seria pertinente utilizar outras bases de dados para averiguar a qualidade do Perceptron para diferentes problemas com diferentes classes.

VII. Manual do Usuário

No contexto de base de dados para redes neurais, é comum que não haja um tratamento prévio de dados, ou seja, diferentes bases estarão organizadas de formas distintas e portanto, precisam ser tratadas individualmente. Contudo, visando atribuir a máxima generalidade possível ao código, os parâmetros a serem alterados pelo usuário se encontram nas classes *neuronio*, *layer* e nas funções *treinar e bootstrap_val*.

Na classe *neuronio* os parâmetros são:

- *pesos*: Valor do número de pesos;
- *func_phi*: Função de ativação a ser utilizada. A mesma está armazenada na variável *chosen_one*.

Para a classe *layer* os parâmetros são:

- *nmb_neuronios*: Quantidade de neurônios na camada;
- *input_numb*: Número de classes a serem separadas;
- *act_func*: Função ativação a ser utilizada;

Para a função *treinar*, os parâmetros são:

- *neuronio*: Neurônio criado a partir da classe homônima;
- *dados*: Input de valores da base de dados;
- *resposta*: Output esperado da base de dados;
- *epoca*: Número de épocas desejado;
- *eta*: Taxa de aprendizado;
- *t_plot*: Define o espaçamento de prints de erro entre as épocas;
- *tol*: Tolerância desejada.

Finalmente, para a função *bootstrap_val*, os parâmetros são:

- *test_times*: Quantidade de execuções do método;
- *input_data_set*: Input de valores da base de dados;
- *output_data_set*: Output esperado da base de dados;
- *classes*: Número de classes da base de dados;
- *atributos*: Número de atributos da base de dados;
- *iterations*: Número de iterações passado para a função *treinar*.

References

[1] HAYKIN, Simon. Neural Networks and Learning Machines. 3. ed. Hamilton: Pearson Prentice Hall, 2008. 846 p.

[2] SANTOS, Cristiano de Carvalho. Bootstrap e Jackknife. Departamento de Estatística, Universidade Federal de Minas Gerais (UFMG). Disponível em: http://www.est.ufmg.br/~cristianocs/MetComput/Aula8.pdf. Acesso em: 23 maio 2022.