Projeto 4 FSI

Matheus Eiji Endo
Estudante de Ciência da Computação
Universidade de Brasília
Brasília, Brasil
matheus.endo@hotmail.com

Johannes Peter Schulte
Estudante de Ciência da Computação
Universidade de Brasília
Brasília, Brasil
johpetsc@gmail.com

***Resumo*—Documento referente ao projeto 4 da disciplina Fundamentos de Sistemas Inteligentes.**

***Index Terms*—Aprendizagem, redes neuronais artificiais, perceptron, retropropagação, peso**

I. Introdução

Com o avanço das tecnologias relacionadas à aprendizagem de máquina é natural a tentativa de simular o comportamento humano. Assim surgiu a ideia de Redes Neuronais Artificiais(RNA), que são redes que se baseiam na forma como o cérebro humano se comporta, onde os neurônios são células com núcleo e corpo(soma) responsável pelo processamento de informação, a saída da informação do corpo é feito através de impulsos elétricos pelo axônio, os dentritos recebem sinais de outros neurônios e a sinapse é o ponto de conexão entre os neurônios, que são conectados entre si formando redes, e os modelos de RNAs simulam essas redes biológicas em artificiais.

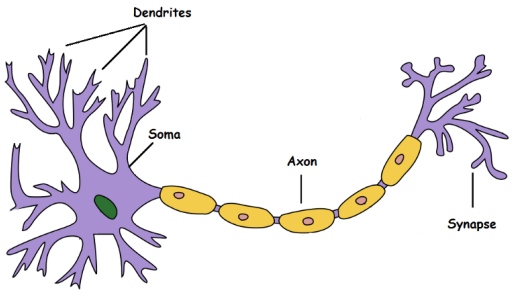

Figura 1. [1]Exemplo Neurônio

II. Redes Neuronais Artificiais

A. Neurônio Artificial

Os primeiros passos em relação ao desenvolvimento de Rede Neuronais Artificias começaram em 1943 quando Warren McCulloth e Walter Pitts escreveram um artigo sobre como os neurônios artificiais possam funcionar. Esses neurônios possuem m entradas binárias(0 ou 1), que possuem pesos iguais, e fazendo uma simples soma dessas entradas o valor resultante é comparado com um limiar e a saída desse neurônio seía 1 somente se o valor resultante for maior que o limiar, nos outros casos ela seía 0. Em 1949 Donald Hebb escreveu um livro chamado *The Organization of Behavior*, que destacou o fato de que as conexões entre os neurônios eram reforçadas toda vez que fossem ativadas.

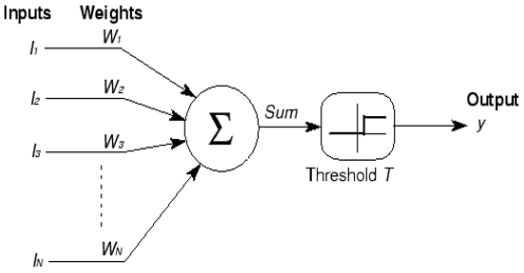

Figura 2. [2]Exemplo Neurônio de McCulloth e Pitts

B. Perceptron

No final da década de 1950 Frank Rosenblatt desenvolveu um modelo baseado nas ideias de McCulloth e Pitts chamado de Perceptron. O Perceptron funciona de forma similar ao neurônio de McCulloth e Pitts, porém os valores de entrada não são só booleanos, os pesos e o limiar não são necessariamente idênticos, os pesos podem ser negativos ou positivos, utiliza-se o limiar como um e a diferença mais importante foi a apresentação de um algoritmo de aprendizagem que adapta os pesos internos do neurônio, de forma que esse modelo possa resolver problemas de classificação linear, desde de que as classes sejam separáveis linearmente.

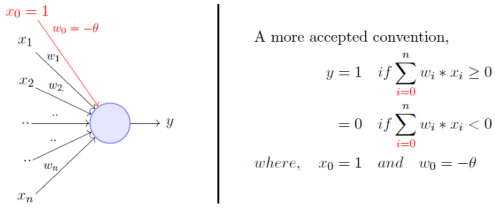

Figura 3. [3]Exemplo Perceptron

[1]https://towardsdatascience.com/mcculloch-pitts-model-5fdf65ac5dd1

[2]http://www.ques10.com/p/13297/what-is-mcculloch-pitts-neuron-model-with-the-he-1/

[3]https://towardsdatascience.com/perceptron-the-artificial-neuron-4d8c70d5cc8d

C. Perceptron MultiCamadas

Apesar desse avanço, as limitações do perceptron foram mostradas por Marvin Minsky e Seymour Papert no livro chamado *Perceptrons*. O livro mostra que um simples problema do tipo ou exclusivo (XOR) não pode ser resolvida pelo perceptron. Em 1986 Rumelhart publicou o algoritmo de retropropagação. A utilização desse algoritmo foi resultante nas redes chamadas Perceptron MultiCamadas (MLP). Nesse tipo de rede existem as camadas de entrada, saída e as chamadas de camadas escondidas ou intermediárias, que não possuem número definido mas o 3 é o número mais usado. Essas camadas são camadas identificadoras de características, seus pesos são uma codificação de características apresentadas nos padrões de entrada. Utiliza-se funções de ativação para determinar a ativação ou não de cada neurônio, a principal função utilizada é a função sigmoid em vez de um limiar, pois a função sigmoid é diferenciável e não decrescente, que é importante pois a maioria das redes MLP utilizam a regra delta generalizada para adaptar os pesos das conexões.

O treinamento é feito através do algoritmo de retropropagação, onde primeiro, um padrão é apresentado à camada de entrada da rede. A atividade resultante passa pela rede, camada por camada, até que a resposta seja obtida pela camada de saída. No segundo passo, a saída resultante é comparada à saída desejada para esse padrão de entrada particular, se esta não estiver correta, o erro é calculado. O erro é propagado a partir da camada de saída até a camada de entrada, e os pesos das conexões das camadas internas vão sendo modificados conforme o erro é retropropagado. Para aprender os pesos das conexões, utiliza-se uma função de erro entre o mapeamento realizado e o desejado, utilizando a distância euclidiana quadrática. Após o treino, a rede pode ser utilizada como uma ferramenta para classificação de novos dados. Para isto, a rede será utilizada como feed-forward, ou seja, novas entradas são apresentadas à camada de entrada, são processadas nas camadas escondidas e os resultados são apresentados na camada de saída, como no treinamento, mas sem a retropropagação do erro.

A rede Perceptron Multicamadas é um aproximador universal de funções, isso quer dizer que desde que os pesos sejam bem adaptados, e a rede tenha neurônios na camada escondida suficientes, o cálculo desejado é atingido. Isso mostra o poder dessa ferramenta, porém existem algumas limitações. Uma limitação é o fato de não se saber o porque a rede chega a determinado resultado, elas podem ser vistas como uma 'caixa preta', não se sabe exatamente o que ocorre dentro dessa caixa. Outro possível problema é o longo tempo de treinamento, pois para poder adaptar os pesos dentro de um erro aceitável pode ser necessário muito tempo e muitos dados. Além disso é preciso fazer uma boa escolha em relação aos pesos iniciais, pois dependendo dos pesos pode ser que na fase de treino o algoritmo fique preso em um mínimo local, para combater isso o mais comum é escolher pesos iniciais de forma aleatória, com distribuição uniforme.

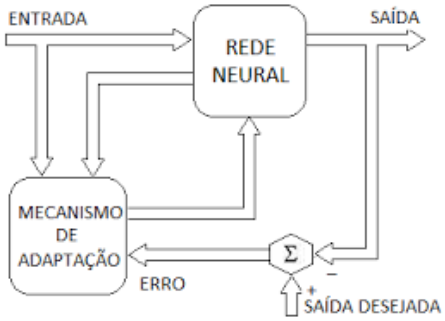

Figura 4. Exemplo Retropropagação.

D. Redes RBF

Funções de base radial são funções da forma:

$$\phi\left(\|X - Xi\|\right) \tag{1}$$

Com ϕ sendo uma função de ativação não-linear, X a entrada e Xi a *i-ésima* posição, ou centro.

Redes RBF, ou rede com função de ativação de base radial, são redes que utilizam uma função com base radial como função de ativação dos neurônios da camada escondida, além disso outras características dessa redes é o fato de haver somente uma camada escondida e os neurônios de saída são sempre lineares.

III. Problema

Nesse projeto devem ser utilizados dois modelos de Redes Neuronais Artificiais para o reconhecimento de e-mails. Os modelos que serão implementados são: MLP(Perceptron Multicamadas) e FBR(Função de base radial) que deverão ser treinados com uma base de dados que indica por +1 e -1 se um e-mail é spam ou não. Os resultados do problema devem ser distribuídos entre treino/teste numa proporção 90/10. Os resultados no final são feitos por validação cruzada.

IV. Implementação

Para a implementação do quarto projeto da disciplina foi utilizada a linguagem de programação *Python(3)*, devido a sua vasta coleção de bibliotecas referentes à aprendizagem de máquinas. A biblioteca *sklearn* foi novamente escolhida para a parte aplicada do projeto, devido a eficiência que teve no primeiro, segundo e terceiro projeto e por apresentar funções específicas para o tema do projeto, Redes Neurais Artificiais.

O script deve ser executado num terminal Linux com as seguintes bibliotecas instaladas: *pandas*, para leitura do arquivo txt; *numpy* para o melhor manuseamento de matrizes no código; *sklearn*, para os cálculos referentes a Rede neural; *matplotlyb*, para gerar a matriz de confusão em gráfico. Funções de sklearn utilizadas:

- *confusion_matriz*, que gera uma matriz de confusão baseada nos dados e resultados dos testes.
- *accuracy_score*, que testa a precisão dos testes realizados.
- *classification_report*, que reporta as principais métricas de classificação.

- *train_test_split*, que divide os dados em 70/30 para teste como requisitado.
- *MLPClassifier*, é a função responsável por realizar os testes para MLP.
- *GaussianProcessClassifier*, para fazer os testes usando seu kernel FBR.

Para a execução do script, o primeiro passo foi ler os dados do arquivo spambase.data.txt com a biblioteca pandas. Foi necessário atribuir o nome para cada atributo devido a utilização desse arquivo em específico. Depois são gerados os labels a partir do valor 0 ou 1 do arquivo. Depois dos dados serem divididos entre X e Y, são aplicados as funções de teste MLPClassifier e GaussianProcessClassifier, em seguida são gerados os resultados com validação cruzada.

V. Resultados

O primeiro teste realizado foi do Perceptron MultiCamadas, que teve a execução relativamente rápida(questão de segundos) e obteve resultados bem satisfatórios, variando de 80% até 94% em alguns casos. O melhor valor para 0(not spam) foi 96% e o pior 81%. Já para o reconhecimento de 1(spam), os valores foram de 100% até 71%, apresentando uma variação muito maior.

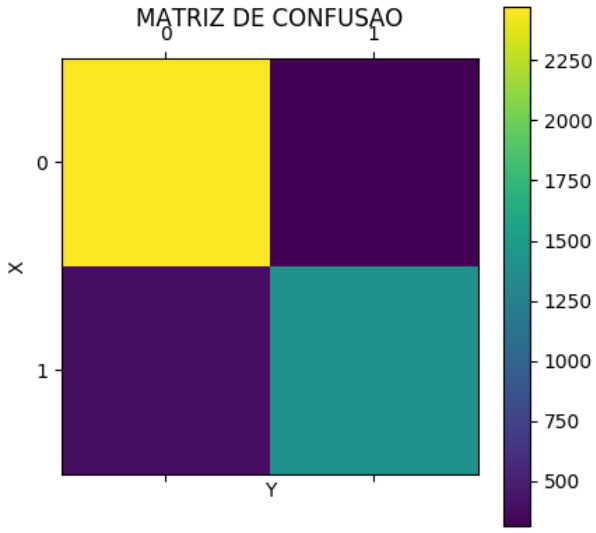

Figura 5. Matriz de Confusão MLP.

Na matriz de confusão MLP é possível observar esses resultados, onde 0-0(reconhecimento do que não é spam) teve os maiores valores, e 1-1(reconhecimento do que é spam) teve um valor um pouco inferior. 0-1 e 1-0 são os erros, que possuíram valores bem baixos.

Já usando a rede RBF, sua execução teve duração muito elevada, durando até vários minutos para terminar todos os treinos, isso pode ter ocorrido devido a escolha da função da biblioteca sklearn, porém foi a única encontrada para realizar o teste RBF, impossibilitando a comparação para comprovar se o problema realmente foi esse.

Mesmo demorando, os testes com RBF também tiveram um bom resultado, tendo em média a precisão de 95% para o reconhecimento de spam/not spam.

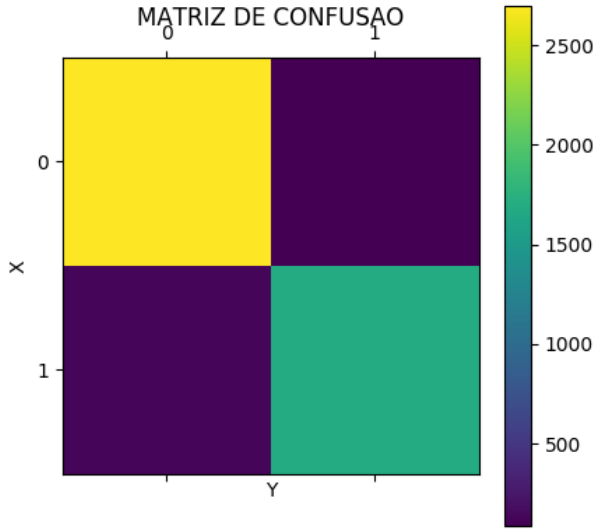

Figura 6. Matriz de Confusão FBR.

Comparando essa matriz de confusão com a matriz de MLP, podemos perceber que o reconhecimento de de não spam(0-0) também foi muito alto, porém a principal diferença foi o na hora de reconhecer o que era spam(1-1) que teve um valor relativamente mais alto.

A partir desses testes realizados com MLP e FBR, é possível dizer que, mesmo com um processamento maior, a rede FBR conseguiu melhores resultados, o que fica bem evidente quando comparamos ambas as matrizes de confusão. Também é possível observar que a matriz de FBR vai de 0 até 2500, enquanto a de MLP tem seu pico em 2250. Mostrando que os valores alcançados por FBR foram maiores.

Referências

1 https://towardsdatascience.com/perceptron-the-artificial-neuron-4d8c70d5cc8d
2 http://www.psych.utoronto.ca/users/reingold/courses/ai/cache/neural4.html