Trabalho 1 - Sistemas Inteligentes

Matheus Schmitz Oliveira
15/0018371

Pedro Aurélio Coelho de Almeida
14/0158103

17 de Abril de 2019

1 Introdução

O reconhecimento de padrões é um dos principais objetivos do subconjunto da inteligência artificial conhecido como *Machine Learning*. Existem diferentes tipos de algoritmos que possuem a capacidade de otimizar seu desempenho a partir da experiência obtida. Para o seguinte projeto será utilizada a técnica de Redes Neurais Artificiais, mais especificamente um modelo simples e rudimentar, denominado *Perceptron*.

1.1 Neurônio Artificial

Parcialmente baseado na estrutura e funcionamento de um neurônio natural, o neurônio artificial recebe um conjunto de entradas numéricas X, um conjunto de pesos W e produz uma saída y.

Soma-se um *bias* (b) ao produto interno de suas entradas $(x_1, x_2, ..., x_m)$ com os respectivos pesos $(w_1, w_2, ..., w_m)$, decidindo, de acordo com uma função de ativação (φ), o resultado obtido em sua saída y. É importante ressaltar que b é um valor de polarização considerado como o peso de uma entrada de valor constante 1. Um exemplo de neurônio artificial com função de ativação é mostrado na Figura 1.

Os neurônios artificiais podem ser conectados em uma (*Perceptron Simples*) ou mais camadas (*Perceptron Multicamada*), gerando redes cada vez mais complexas.

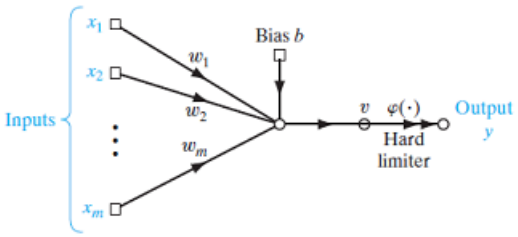


Figure 1: Exemplo de Neurônio Artificial [2]

1.2 Perceptron

Em sua estrutura inicial, a rede *Perceptron* possui apenas uma camada, na qual os neurônios geralmente possuem uma função de ativação do tipo degrau unitário. Cada neurônio desta irá informar se as entradas fornecidas pertencem ou não à uma determinada classe.

Por se tratar de um algoritmo de classificação supervisionado, a rede possui a informação das saídas desejadas para cada conjunto de dados de treinamento fornecido. A partir disso, as saídas produzidas são comparadas e, caso uma diferença entre os resultados desejados e obtidos aconteça, uma atualização no conjunto de pesos W é efetuada a fim de melhor classificar o conjunto de entradas.

O conjunto de equações a seguir descreve o funcionamento do *Perceptron*:

$$v = \sum_{i=1}^{m} w_i x_i + b \tag{1}$$

$$\varphi(v) = \begin{cases} 1, & \text{se } v > 0 \\ 0, & \text{caso contrário} \end{cases} \tag{2}$$

$$y = \varphi(v) \tag{3}$$

$$W' = W + \eta * (\underbrace{d^i - y}_{erro}) * X^i \tag{4}$$

onde v representa o somatório ponderado das entradas, φ é a função de ativação, caracterizada como um degrau unitário , y é a saída obtida de acordo com a função de ativação escolhida φ, η representa a taxa de aprendizado e W' é o novo conjunto de pesos calculado.

1.3 Base de Dados

A base de dados *Sonar* [1] consiste num conjunto de 208 observações (111 para minas e 97 para rochas), cada uma com 60 atributos respresentados em valores no intervalo $[0, 1]$ e devidamente rotuladas entre duas classes. Os atributos são obtidos através de sinais coletados referentes à energia dentro de uma determinada frequência de banda, integrada durante certo período de tempo.

Dado que os valores estão no intervalo descrito anteriormente, torna-se dispensável o uso de técnicas de normalização ou padronização dos dados.

Vale ressaltar que os exemplos da base de treinamento (145 observações) são linearmente separáveis, o que possibilita a um modelo linear atingir erro 0 após uma quantidade finita de épocas.

2 Metodologia

Uma vez que o problema descrito em 1.3 pode ser resumido em classificar as entradas como pertencentes ou não pertencentes a uma determinada classe (a entrada fornecida pertence ou não à classe de rochas, por exemplo), a estrutura padrão do *Perceptron* com neurônio unitário é capaz de efetuar a classificação.

Todos os dados presentes na base de treinamento foram utilizados para treinar o classificador. A cada observação analisada, a rede pode realizar a atualização de seus pesos.

A base de treino foi utilizada para configurar os meta-parâmetros da rede neural: escolhendo-se a taxa de aprendizado (0.04) e o número de épocas (30000). A base de teste foi utilizada somente para avaliar o desempenho do classificador após o treinamento atingir 100% de acurácia.

A implementação do *Perceptron* foi feita na linguagem Python$^{\text{TM}}$ uma vez que esta facilita o desenvolvimento sem causar grandes prejuízos de desempenho para a solução criada.

As seguintes bibliotecas foram utilizadas:

- pandas para a leitura e manipulação dos arquivos de treino e teste;
- numpy para realizar cálculos numéricos como o produto escalar;
- sklearn para obter a matriz de confusão;
- matplotlib para visualizar os resultados de erro e acurácia com relação à época de forma gráfica

O *Github* foi utilizado como repositório para armazenar a implementação do sistema [3]. A aplicação *Jupyter Notebook* foi utilizada para o desenvolvimento do código do projeto.

3 Resultados

A imagem a seguir apresenta o gráfico da evolução do erro quadrático obtido após a execução de cada época. Nota-se que inicialmente o erro é elevado, mas decai conforme ocorrem novas iterações.

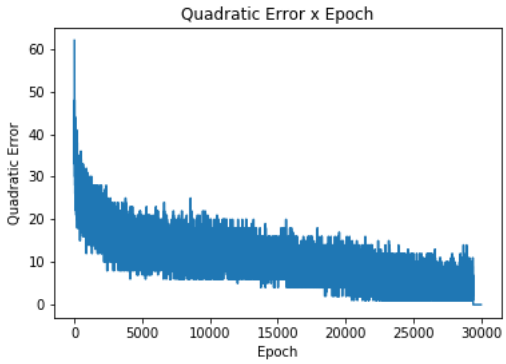


Figure 2: Número de Épocas x Erro Quadrático

O segundo gráfico obtido apresenta a evolução da acurácia obtida nos exemplos de treinamento. De forma oposta ao erro quadrático, a acurácia inicial é baixa e evolui durante as novas iterações.

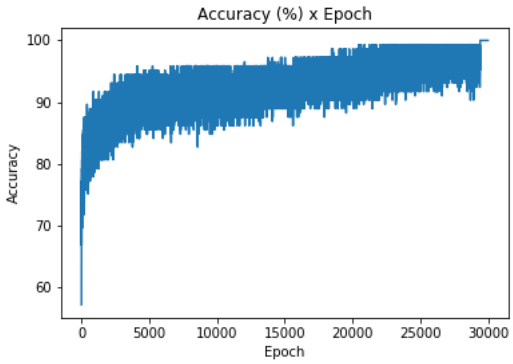
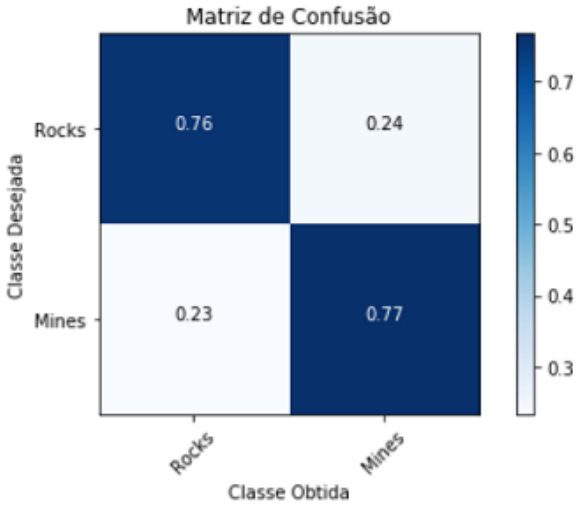


Figure 3: Número de Épocas x Acurácia

A seguinte matriz de confusão foi obtida ao aplicar os exemplos da base de teste (63 observações) ao neurônio configurado a partir da base de treinamento.

Figure 4: Matriz de Confusão Obtida



A tabela abaixo apresenta medidas de desempenho importantes para a avaliação do modelo:

Classes	Precision	Recall	F1-Score	Support
Rocks	0.78	0.76	0.77	33
Mines	0.74	0.77	0.75	30
Total	0.76	0.76	0.76	63

As demais informações (Erro x Época, Acurácia x Época e o conjunto final de pesos) estão explicitadas no código e podem ser visualizadas no arquivo *Trabalho 1 - Source Code.ipynb* presente no repositório do projeto.

4 Análise

Os resultados da fase de treinamento da rede, apresentados nas Figuras 2 e 3 , estão de acordo com o esperado, uma vez que o erro efetivamente chegou a 0 após, aproximadamente, 30000 épocas. Outro resultado esperado foi o comportamento geral descrescente do erro até sua estabilização a partir do momento que o sistema conseguiu separar completamente os dados.

Para a base de teste, obteve-se uma acurácia média de 76%. A precisão da classe de rochas é de 78% e da classe de minas é de 74%, como mostrado na Tabela anterior.

Uma vez que esse conjunto de dados não é linearmente separável, um classificador linear, como o *Perceptron* de uma camada utilizada, não é capaz de atingir grandes valores de acurácia e precisão. Devido à não linearidade dos problemas, os resultados obtidos foram considerados adequados, uma vez que dificilmente acurácias acima de 80% seriam atingindas com classsificadores lineares.

5 Conclusão

Conclui-se que, apesar de ser um modelo relativamente simples e elementar, a rede *Perceptron* mostrou-se capaz de realizar a identificação de padrões e classificar corretamente, com acurácia média esperada, classes distintas em um grupo de observações.

Para obter melhores resultados, pode-se utilizar diferentes algoritmos supervisionados de classificação, tais como: KNN, SVM, Naive Bayes, dentre outros.

References

[1] Dheeru Dua and Casey Graff. UCI machine learning repository. "http://archive.ics.uci.edu/ml", acessado em 15/04/2019, 2017.

[2] S. Haykin. *Neural Networks and Learning Machines*. Pearson, 2009.

[3] Matheus Schmitz Oliveira and Pedro Aurélio Coelho de Almeida. Repositório para implementação do projeto 1 - sonar. "https://github.com/mso13/trabalho1-sonar", acessado em 15/04/2019", 2019.