Implementação de Aprendizagem de Máquina com o Dataset Íris

Arthur Tassinari Cabral[1], José Augusto Accorsi[2], Leonardo Broch de Morais[3]

Universidade do Vale do Rio dos Sinos (UNISINOS)
93.020-190 – São Leopoldo – RS – Brasil

arthurtcabral@gmail.com[1], augusto.acorsi@gmail.com[2], leobroch036@gmail.com[3]

Abstract. *This paper addresses the study of Machine Learning that have done with the dataset named Íris. It explores conceptual topics, involving either the learning machine or the dataset, and the application, that was developed by using the mentioned dataset.*

Resumo. *Este artigo aborda o estudo da aprendizagem de máquina realizados com a base de conhecimento nomeada Íris. São explorados desde tópicos conceituais, envolvendo a aprendizagem de máquina e o dataset, até a aplicação, desenvolvida utilizando o conjunto de dados mencionado.*

1. Introdução

Este artigo apresenta o estudo da aprendizagem de máquina em conjunto com a utilização de Redes Neurais com o dataset Íris. São explorados, primeiramente, as conceituações referentes à aprendizagem de máquina em si, para que, de modo posterior, seja apresentada a utilização do dataset mencionado.

O objetivo deste artigo é efetuar a representação do conceito de aprendizagem de máquina através de uma abordagem prática, onde utilizou-se uma Rede Neural para classificar a base de conhecimento Íris. Tal pesquisa foi realizada de maneira a aplicar os conhecimentos obtidos ao longo da disciplina de Inteligência Artificial, na Universidade do Vale do Rio dos Sinos (UNISINOS). Também considera-se como parte do objetivo a exploração dos temas propostos.

Como fontes de consulta, foram utilizados artigos e demais websites do cunho da TI. A própria aplicação também é uma fonte de consulta.

2. Conceituações

2.1 Aprendizagem de Máquina

O aprendizado de máquina - em inglês: *Machine Learning* - pode ser concebido como sendo uma metodologia de análise de dados onde se busca aprimorar a capacidade de decisão de uma máquina por meio de uma série de eventos de "tentativa e erro" sobre um tópico específico. É um ramo da inteligência artificial que baseia-se na ideia de que as máquinas podem aprender com os dados que lhe são enviados para processamento. A aprendizagem de máquina busca garantir o aprimoramento da tomada de decisão, por parte de uma máquina, considerando o mínimo de intervenção humana.

A aprendizagem de máquina é capaz de proporcionar experiências valiosas para o consumidor final, uma vez que permite que mecanismos de busca, aplicativos web e demais tecnologias personalizem recomendações e resultados de acordo com as preferências do usuário. Desta forma, as experiências que os consumidores têm são mais caricatas a ele, como se tivessem sido desenvolvidas a ele exclusivamente.

2.2 Redes Neurais

Redes Neurais visam emular o funcionamento do cérebro humano através de nós interconectados que funcionam como neurônios. Através desse algoritmo é possível se descobrir padrões e correlações escondidas dentro de dados brutos, podendo assim agrupá-los e classificá-los, e sendo um algoritmo de aprendizagem de máquina, ele gera conhecimento e melhora através do tempo e da experiência.

A primeira Rede Neural foi concebida em 1943 pelos cientistas Warren McCulloch e Walter Pitts. Eles elaboraram um artigo descrevendo como a Rede Neural deveria funcionar e a experimentaram em circuitos elétricos. Esse modelo estudado pavimentou o caminho para que estudos se aprofundassem nas áreas de processos biológicos no cérebro e também no uso das Redes Neurais nas áreas de inteligência artificial.

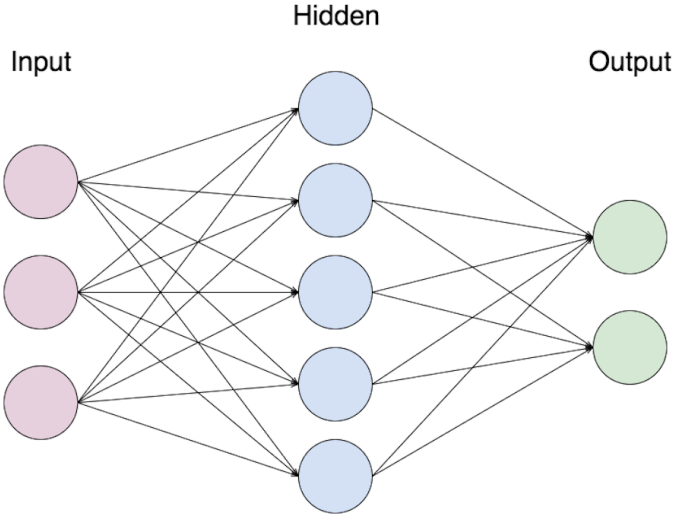

Figura 1. Gráfico representando uma Rede Neural, onde podemos ver uma camada de input, uma camada intermediária e uma camada de output

2.3 Keras

Na implementação a qual esse artigo retrata foi utilizada a biblioteca Keras, que é um projeto open-source escrito em Python que provém uma série de serviços para lidar com Redes Neurais, esta biblioteca roda em cima da ferramenta TensorFlow, ferramenta também open-source que foi desenvolvida na Google. O Keras tem como principal objetivo prover uma camada de alto nível para que o programador trabalhe com Redes Neurais. É bastante utilizada em prototipação, por ter uma curva de aprendizagem muito pequena e também por ser de rápido desenvolvimento.

2.4 Dataset Iris

Como massa de dados onde foi realizada a análise foi utilizado o Dataset Íris, que busca classificar através de alguns atributos o tipo de uma planta da família Íris. No dataset utilizado estão dados de três tipos diferentes de Íris: Íris-setosa, Íris-versicolor e Íris-virgínica.

O dataset possui ao todo seis colunas, cinco delas representando atributos e a última delas representando uma das três classes em que a Íris analisada pode pertencer. Também se fazendo uma análise inicial do dataset, pode se perceber que a primeira coluna representa um id único do registro, não sendo aproveitado para análise.

Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	5.1	3.5	1.4	0.2	Iris-setosa

Figura 2. Esquema do dataset utilizado e um exemplo de registro

Como visto no exemplo da imagem acima, os atributos úteis para análise são quatro: SepaWidthCm, PetalLengthCm, SepaLengthCm, PetalWidthCm. E a última coluna se refere a uma das três classes possíveis, que já foram mencionadas anteriormente neste artigo.

O Dataset utilizado se encontra em um arquivo csv e possui no total 150 registros de Íris, sendo assim um conjunto considerado pequeno para análise, porém com dados suficientes para a experimentação proposta.

3. Implementação

3.1 Preparação do Dataset

Primeiro passo ao implementar uma Rede Neural é analisar o problema a ser solucionado, no nosso caso precisamos classificar plantas do tipo Íris em três espécies diferentes. Para isto contamos com um dataset com seis colunas, cinco atributos e uma coluna que representa a classe a ser classificada. Primeiramente como já descrito nesse artigo, uma das colunas presentes no dataset é um identificador único do registro, que basicamente sinaliza a linha atual do arquivo, este campo por não se tratar de um atributo da Íris em questão obviamente deve ser desconsiderado da análise. Nas figuras abaixo apresentamos as bibliotecas que foram importadas e a preparação do Dataset em questão.

```
import pandas as pd
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Dense
import matplotlib.pyplot as plt
from keras.layers import Dropout
from keras import regularizers
from keras.optimizers import SGD,Adam
```

Figura 3. Bibliotecas que foram importadas e utilizadas no projeto

```
X = dataset.iloc[:,1:5].values
y = dataset.iloc[:,5].values

from sklearn.preprocessing import LabelEncoder
encoder =  LabelEncoder()
y1 = encoder.fit_transform(y)

Y = pd.get_dummies(y1).values
```

Figura 4. Preparação do Dataset para posterior utilização no algoritmo

Na figura acima mostramos que após a leitura do arquivo que contém o Dataset em formato csv e armazená-lo na variável *dataset,* preparamos dois vetores X e Y, que irão conter respectivamente os dados dos atributos analisados e os dados das classes pertencentes a estes dados. Aqui podemos ver que como atributos selecionamos as colunas 2 a 4, excluindo a coluna 1 que representa a linha atual do arquivo. Após selecionarmos a última coluna do dataset a qual representará a classe a ser analisada. No dataset original ela está representada como uma string, porém para análise pela Rede Neural precisamos transformar este campo em um valor numérico, para isto usamos a função *fit_transform,* que transforma os valores das classes em um número inteiro de 0 até o número total de classes menos um.

```
from sklearn.model_selection import train_test_split
X_train, X_val_and_test, Y_train, Y_val_and_test = train_test_split(X_scale, Y, test_size=0.3)
```

Figura 5. Separação do dataset entre registros utilizados para treino e teste do modelo

Na figura acima podemos ver a separação dos dados entre os que serão usados para treino da Rede Neural e os dados que serão utilizados para validação e teste do sistema. Aqui foi utilizado a métrica de 70% dos registros para treino e 30% dos registros para testes, está métrica foi utilizada pois é amplamente difundida na bibliografia consultada e também foi a recomendada no material complementar da disciplina. Essa parcela do datasource separada para testes foi também dividida em duas partes, uma para validação a cada época de treino do algoritmo e a segunda parte para testes ao final da etapa de treino.

3.2 Estrutura da Rede Neural

Após a preparação do dataset a ser analisado partimos então para a estruturação e implementação de fato da Rede Neural. Como já comentado no presente artigo, foi utilizada a biblioteca Keras, que fornece uma abstração de alto nível para a implementação de Redes Neurais utilizando a linguagem Python. A arquitetura da Rede Neural implementada foi a de FeedForward com múltiplas camadas, que de acordo com a bibliografia consultada é a que apresenta melhores resultados para problemas de classificação, como o proposto pelo grupo neste artigo.

Como nosso problema é de classificação dos dados de entrada em uma única classe dentre as várias existentes, podemos determinar que a primeira camada, a camada de entrada, terá um neurônio para cada feature existente, ou seja, para cada coluna do dataset. Neste caso,

como possuímos a definição de quatro atributos no Dataset, nossa primeira camada terá quatro neurônios. Já a camada de saída da Rede Neural possuirá um neurônio para cada classe existente, no Dataset em questão são três. Cada um deles deverá retornar um valor entre 0 e 1, que representa a probabilidade que a classe que este neurônio representa ser a correta para a entrada analisada.

```
model = Sequential()

model.add(Dense(10,input_shape=(4,),activation='elu'))
model.add(Dense(8,activation='elu'))
model.add(Dense(6,activation='elu'))
model.add(Dense(3,activation='softmax'))

model.compile(Adam(lr=0.04),'categorical_crossentropy',metrics=['accuracy'])

model.summary()
```

Figura 6. Criação da estrutura da Rede Neural

Como podemos ver na imagem acima, criamos o modelo inicialmente com a camada de entrada com quatro neurônios, valor informado no *input_shape*, também são criadas três camadas intermediárias, que serão melhor abordadas na seção de resultados obtidos. E em seguida uma camada de saída que possui três neurônios, com a função de ativação *softmax,* que força com que a saída de uma rede neural represente a probabilidade dos dados serem de uma das classes definidas. Na imagem abaixo podemos ver a equação realizada pela função softmax, a qual trabalha de uma forma diferente das demais, uma vez que a saída de um neurônio da camada final depende da saída dos demais neurônios da mesma camada.

$$\phi_i = \frac{e^{z_i}}{\sum_{j \in group} e^{z_j}}$$

Figura 7: Função de ativação softmax

Nas demais camadas intermediárias foi utilizada inicialmente a função *ELU,* que é a qual apresenta melhores resultados quando posta em camadas intermediárias. Na sequência da implementação informamos a função de perda a ser utilizada, juntamente com a função de optimização e métricas a serem extraídas durante o treinamento da Rede.

```
model.compile(optimizer='sgd',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

Figura 8: Definição das funções de optimização e de perda bem como as métricas a serem extraídas do modelo

Na sequência realizamos a chamada para a função de treino fornecida pela biblioteca Keras e então iniciamos de fato a etapa de treino da Rede Neural construída até aqui, conforme podemos ver na imagem abaixo:

```
hist = model.fit(X_train,Y_train,epochs=100, validation_data=(X_val, Y_val))
y_pred = model.predict(X_test)

y_test_class = np.argmax(y_test,axis=1)
y_pred_class = np.argmax(y_pred,axis=1)
```

Figura 9: Treinamento da Rede Neural construída, passando como parâmetros os datasets de treino e de validação

3.3 Análise dos Resultados

Após então o treinamento da Rede Neural construída foi realizada a extração de métricas para demonstrar a assertividade e eficácia do sistema construído, inicialmente, com a estrutura exata descrita e demonstrada até aqui obtivemos os seguintes resultados no processo de treino e validação.

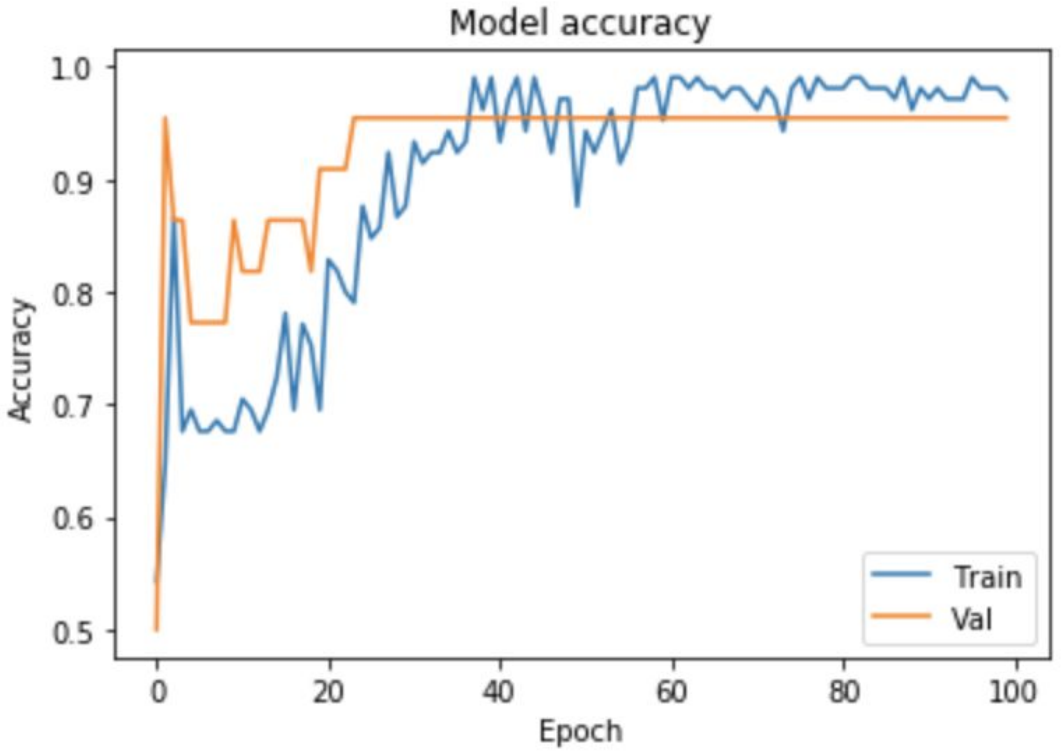

Figura 10: Métricas de assertividade extraídas da Rede Neural demonstrada.

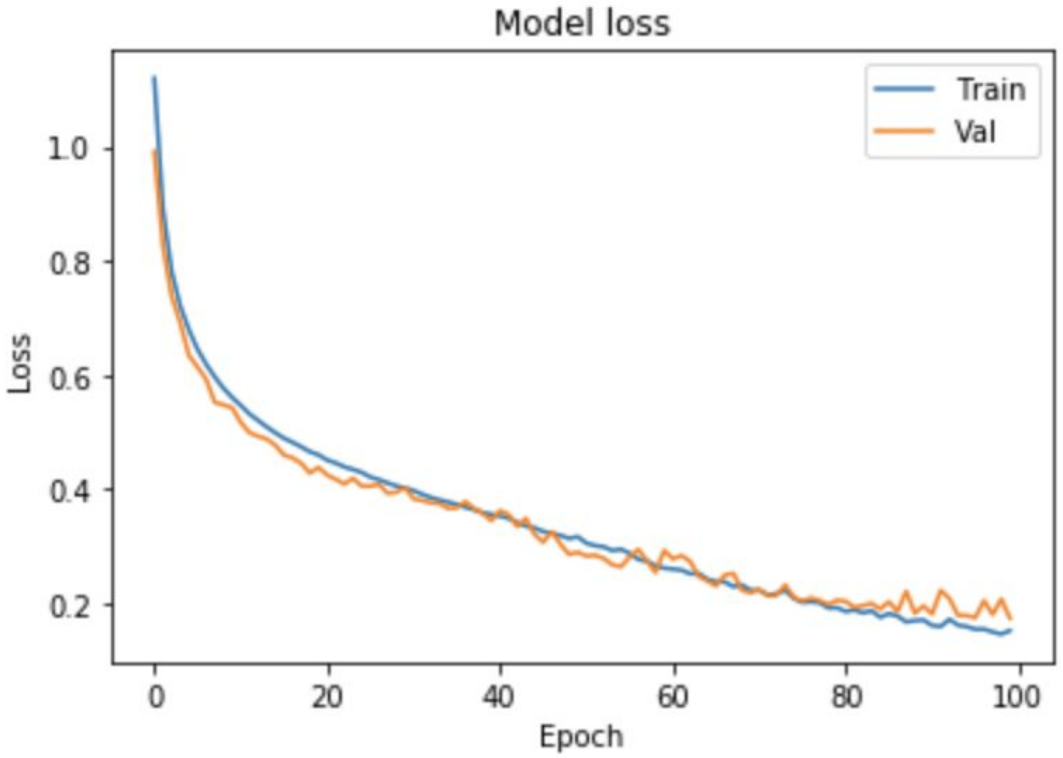

Figura 11: Taxa de perda do modelo

Nos resultados coletados podemos ver que a Rede Neural construída teve resultados satisfatórios, pois sua assertividade durante as etapas de treino e validação ficaram próximas de 100%, e o modelo teve uma taxa de perda razoavelmente baixa. A seguir também foram coletados os resultados da execução da Rede Neural com a mesma configuração aqui descrita, porém com as camadas intermediárias utilizando a função de ativação tanh. Neles se nota uma perda razoavelmente significativa, pois nas etapas de validação da Rede Neural obtivemos resultados próximos de 80% de assertividade, longe do que foi conseguido com o uso do *ELU*.

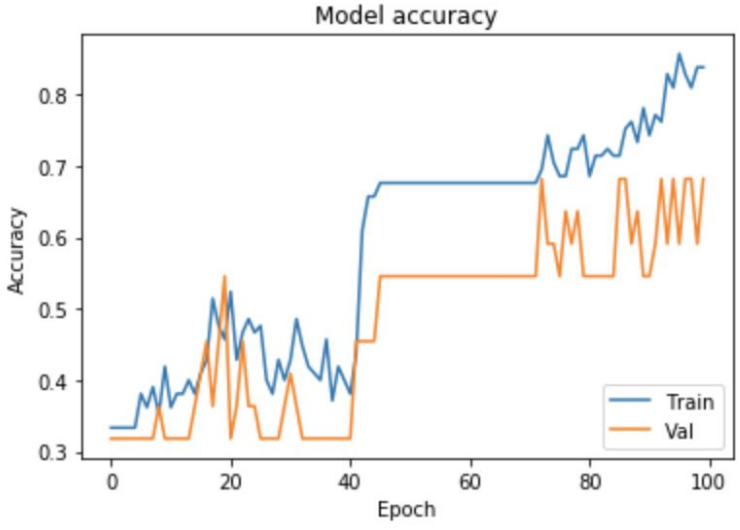

Figura 12. Taxa de assertividade utilizando a função de ativação tanh para as camadas intermediárias

Também foi realizado testes se diminuindo o número de camadas intermediárias, realizando com três e duas camadas, resultados obtidos abaixo.

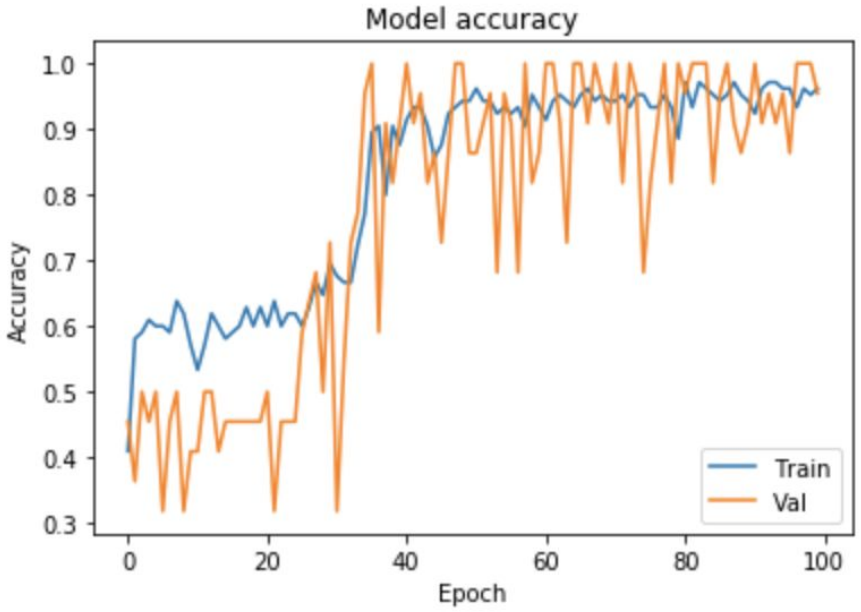

Figura 13. Resultado obtido com três camadas intermediárias

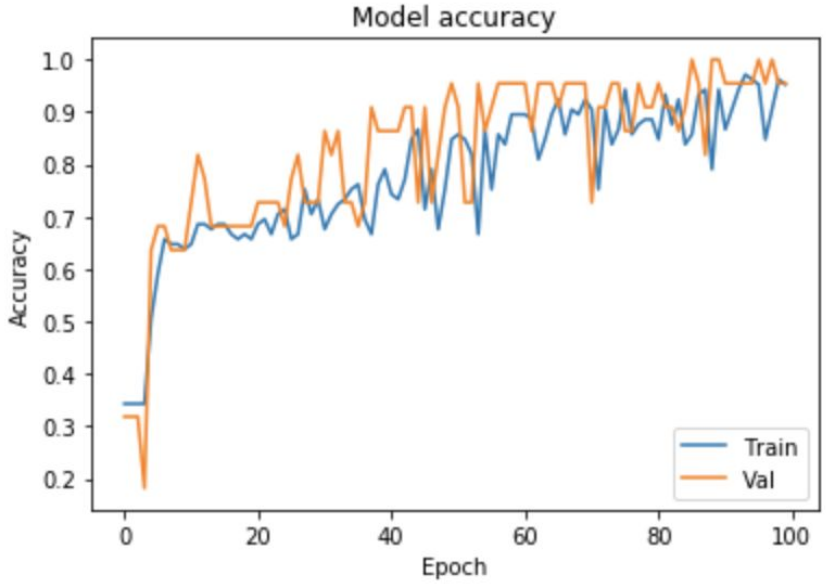

Figura 14. Resultado obtido com duas camadas intermediárias

Nos experimentos com um número menor de camadas intermediárias podemos notar a manutenção dos melhores aproveitamentos, porém notamos também a variância da qualidade dos resultados através das épocas é bem maior.

Referências Bibliográficas

Hewlett Packard Enterprise - O que é aprendizado de máquina? Disponível em: <https://www.hpe.com/br/pt/what-is/machine-learning.html> Acesso em 05 jun 2019.

SAS. Machine Learning - O que é e qual sua importância? Disponível em: <https://www.sas.com/pt_br/insights/analytics/machine-learning.html> Acesso em 05 jun 2019.