Lista 5 - Inteligência Computacional II

Data de entrega: 16 de setembro de 2022

Filipe Silva
Universidade Federal do Rio de Janeiro (UFRJ)
Programa de Engenharia de Sistemas e Computação (COPPE/PESC)
Rio de Janeiro, Brasil
filipe@cos.ufrj.br

***Resumo*—O trabalho tem o propósito de explorar os conceitos aprendidos em sala de aula por meio de suas respectivas implementações de modelos de Inteligência Artificial (IA). Assim, explora-se o uso de técnicas mais antigas e estabelecidas - como árvore de decisão -, assim como técnicas mais recentes com a utilização de SVM e TensorFlow.**

***Index Terms*—Inteligência Artificial, Aprendizado de Máquina, Redes Neurais**

I. Introduction

Em meio à crescente demanda direta e indireta por dados, tem-se como papel neste trabalho implementar e analisar dois experimentos computacionais com uso de aprendizado de máquina e fazendo uso da linguagem de programação Python. O código utilizado pode ser encontrado no repositório do Github[1].

II. Ferramentas utilizadas

Foram utilizados os seguintes *softwares* para realização dos experimentos:

1) **Google Data Studio** - plataforma online para exploração mais customizáveis dos dados resultantes
2) **Spyder** - IDE para programação científica em Python (3.9.7). Os principais pacotes utilizados foram Sklearn, Tensorflow e Pandas

III. Descrição dos experimentos

A. Experimento 1

O objetivo de avaliar a capacidade de classificação do SVM na base de dados sintética da KEEL conhecida como "Banana dataset"; assim chamada pois as instâncias pertencem a vários clusters que se apresentam no formato de uma banana. A base[2] possui 5300 observações, 2 atributos, e 2 classes.

Serão apresentados os seguintes pontos neste experimento:

1) Gráfico de dispersão da base de dados.
2) Utilização do SVM com os kernels sigmoide, linear, RBF e polinomial. Para sigmoide e RBF, será variada a utilização do parâmetro gamma (γ): 1, 0.5 e 0.01; para o polinomial, grau 3. Para cada kernel, será feita a validação cruzada k-fold com três valores para k: 2, 5 e 10. Além disso, uma tabela será colocada com os valores de acurácia e erro.
3) Gráficos que indiquem os vetores de suporte para o modelo kernel/k-fold de melhor e pior desempenho.
4) Utilização de árvore de decisão em comparação com os modelos SVM.

B. Experimento 2

O objetivo é construir uma rede neural convolucional para classificar imagens de algarismos. A base[3] utilizada contém dígitos manuscritos, sendo um conjunto de 60.000 exemplos de treinamento e outro de 10.000 exemplos de teste.

1) Exemplo de imagens de números de forma sintética.
2) Utilização de rede neural convolucional capaz de classificar os algarismos com um nível razoável de acurácia.
3) Gráficos com a quantidade de épocas necessárias para atingir o valor máximo de acurácia e mínimo de erro.
4) Gráfico de acurácia ao longo das épocas (1-5) de treinamento.

IV. Resultados

A. Experimento 1

Gráfico de dispersão da base de dados

Pode-se perceber na figura 1 que o conjunto avaliado tem a distribuição em "bananas". Além disso, podemos afirmar que a base de dados está relativamente equilibrado ao avaliar os histogramas de ambos atributos.

Tratando-se das opções de treinamento, podemos observar também que a figura 1 não apresenta uma distribuição linear, tampouco um polinômio de terceiro grau - como sugerido pelo roteiro do trabalho - será suficiente para se ajustar aos dados.

Utilização do SVM com os kernels sigmoide, linear, RBF e polinomial[4]

É possível visualizar que apenas as três primeiras linhas da tabela da figura 2 possui acurácia fora da amostra (15%) ou K10 $\geq 80\%$ que configura um bom limiar a ser superado. Além disso, como dito e esperado anteriormente, nota-se uma

[1]https://github.com/dfilipeaugusto/pesc_ic2_lista_5

[2]https://sci2s.ugr.es/keel/dataset.php?cod=182

[3]https://www.tensorflow.org/datasets/catalog/mnist

[4]Foi colocado o "kernel"*tree* na tabela somente para uma visualizaçao completa dos dados. Na verdade, o *tree* corresponde ao modelo de classificação de árvore de decisão, onde o valor de gamma não se aplica.

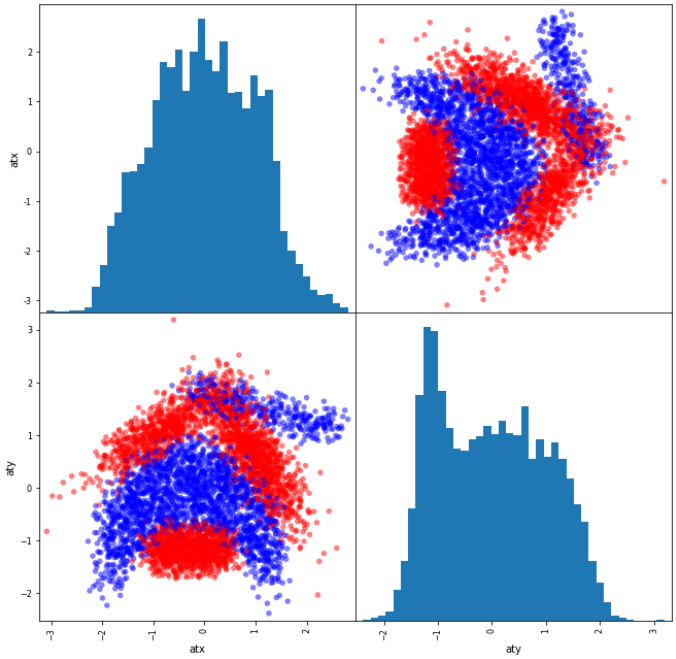

Figura 1. Experimento 1 - Dispersão da base

kernel	gamma	Acc. IN (85%)	Acc. OUT (15%)	Acc. K2	Acc. K5	Acc. K10
rbf	1	90,94%	89,81%	90,58%	90,68%	90,68%
rbf	0.5	90,54%	90,19%	90,38%	90,25%	90,23%
tree	0	100,00%	88,30%	86,87%	86,72%	87,74%
poly	0.500...	63,46%	63,02%	63,68%	63,81%	63,91%
rbf	0.01	59,11%	61,89%	56,96%	58,53%	59,83%
sigmoid	0.01	54,78%	57,36%	55,17%	55,17%	55,17%
linear	0.500...	54,78%	57,36%	55,17%	55,17%	55,17%
sigmoid	0.5	28,37%	31,07%	29,23%	29,26%	29,25%
sigmoid	1	27,79%	30,19%	28,19%	28,21%	28,11%

Figura 2. Experimento 1 - Tabela de acurácia com e sem validação cruzada

abrupta queda no valor de acurácia para o modelo polinomial e kernel linear.

Sobre a variação do hiperparâmetro gamma, é interessante ver o seu comportamento, pois o mesmo é definido antes da etapa de treinamento. O gamma decide quanta curvatura queremos em um limite de decisão; se temos um alto gamma, significa mais curvatura, por outro lado, se temos baixa gama, então menos curvatura - vide o exemplo da figura 3.

Gráficos que indiquem os vetores de suporte

Pode-se ver o melhor e o pior modelos avaliados nas figuras 4 e 5 respectivamente. É esperado também que o kernel RBF tenha um desempenho em acurácia melhor, pois o mesmo usa base radial, ou seja, curva sobre os pontos de dados, o que é compatível as ”bananas”do conjunto de dados.

Curiosamente, também pode-se observar o mesmo tipo de gráfico para o kernel linear na figura 6. Percebe-se que o

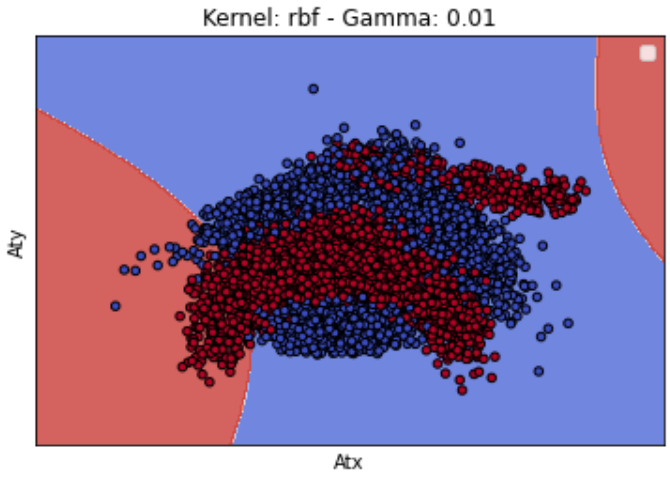

Figura 3. Experimento 1 - Vetores de suporte - RBF - Gamma: 0.01

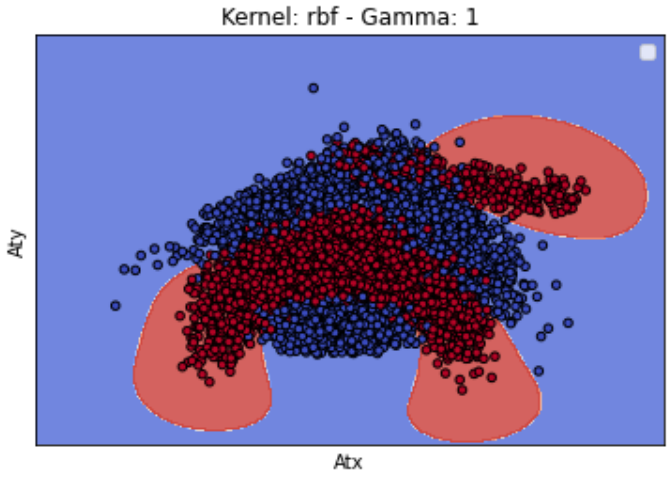

Figura 4. Experimento 1 - Vetores de suporte - RBF - Gamma 1.0

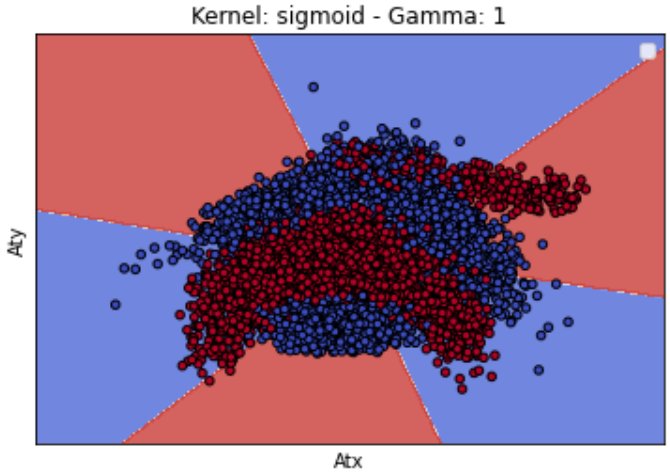

Figura 5. Experimento 1 - Vetores de suporte - Sigmoide - Gamma 1.0

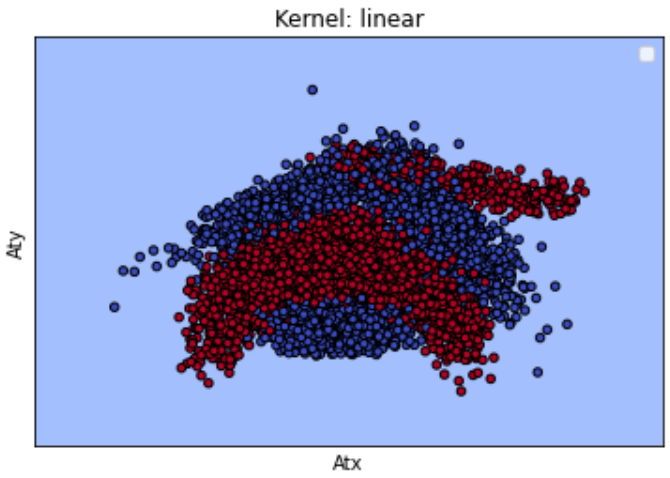

Figura 6. Experimento 1 - Vetores de suporte - Linear

algoritmo basicamente categorizou todos os dados com uma classe tão somente. Desta forma, considerando que há apenas duas classes relativamente equilibradas, é razoável afirmar que a acurácia de tal modelo ficaria próximo de 50% - como é possível verificar na tabela da figura 2.

Utilização de árvore de decisão em comparação com os modelos SVM

A opção por tal utilizada foi somente pela razão de que o modelo de árvore de decisão é um método antigo vigente - ou seja, ainda em uso na literatura e no mercado -, foi apresentado em aula e confere bons índices de acurácia com baixo uso de recursos computacionais.

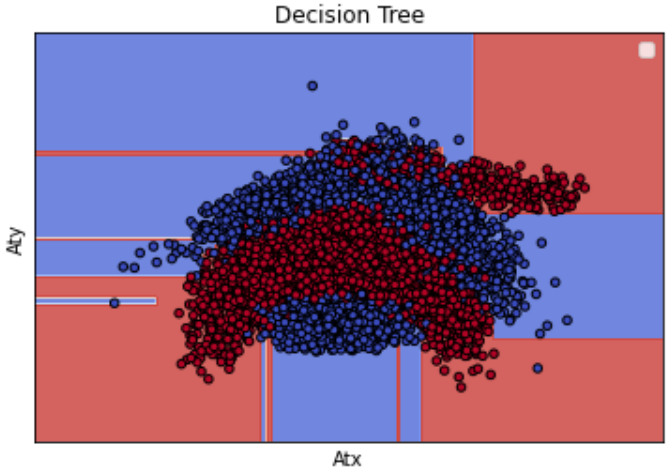

Figura 7. Experimento 1 - Árvore de Decisão

Apesar do sobreajuste natural do modelo nos dados de treinamento, pode-se ver que, fora da amostra, o kernel RBF com $\gamma = 1$ ainda possui uma acurácia maior em todos os outros casos (incluindo validação cruzada). Entretanto, o modelo de Árvore se mantém no pódio dos melhores ajustes, pois ocupa o terceiro lugar na tabela da figura 2, que está ordenada de forma decrescente pela acurácia K10.

B. Experimento 2

Exemplo de imagens de números de forma sintética

Figura 8. Experimento 2 - Exemplos de dígitos manuscritos - MNIST. Fonte: Towards Data Science

A estrutura do Keras já contém o conjunto de dados MNIST que pode ser baixado em tempo de execução do algoritmo Python. Conforme dito em III-B, a base contém 60.000 imagens manuscritas que podem ser usadas para treinar uma rede neural.

Construa uma rede neural convolucional capaz de classificar

Foi utilizado Tensorflow para criação de uma rede com quatro camadas como é possível ver na figura 9. Além disso, é importante ressaltar que foi escolhido o otimizador Adam, pois é um bom método de otimização para descida de gradiente pelo baixo uso de memória em comparação com outros otimizadores e ainda assim é eficiente.

Gráficos com a quantidade de épocas necessárias para atingir o valor máximo de acurácia e mínimo de erro

Durante o treinamento, foram utilizadas 100 épocas (figura 9). É possível notar que na 25ª época foi o primeiro a ser alcançado o melhor de acurácia e perda. Além disso, é importante notar que nas épocas 39 e 71 (figura 10) pequenos vales são aparentes devido o *overfitting* do processo de treinamento.

Desta forma, é possível afirmar que a melhor época para parada do treinamento muito possivelmente estará antes da 25ª, pois foi atingido o pico de acurácia neste momento. Para otimização da escolha, é importante que tivéssemos um outro conjunto de acurácia e perda para dados fora da amostra - que não está no roteiro do trabalho -, pois seria fundamental determinar o limiar.

```
model = tf.keras.models.Sequential()

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(128, tf.nn.relu))
model.add(tf.keras.layers.Dense(128, tf.nn.relu))
model.add(tf.keras.layers.Dense(10, tf.nn.softmax))

model.compile(

    optimizer="adam",
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]

)

result = model.fit(

    x_train,
    y_train,
    batch_size=200,
    epochs=100,
    callbacks=[Iteration_monitor()]

)
```

Figura 9. Experimento 2 - Camadas da rede neural

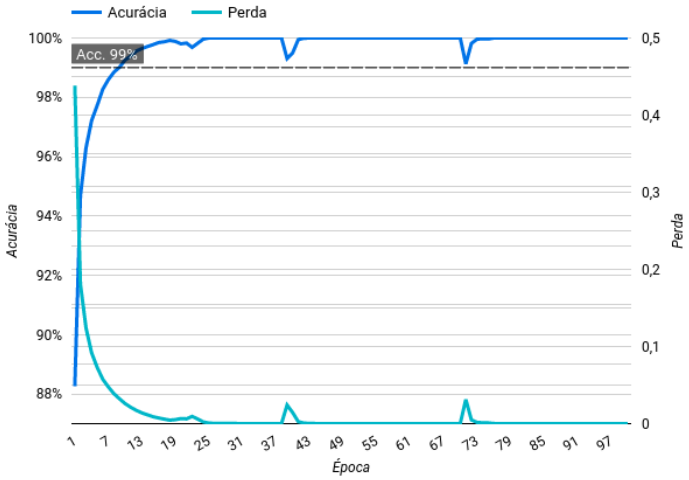

Figura 10. Experimento 2 - Acurácia e perda - Época $\leq$ 100

O gráfico da figura 11 representa os mesmos valores da figura 10, tendo apenas a época limitada a 30 para melhor visualização do comportamento durante o treinamento.

Gráfico de acurácia ao longo das épocas (1-5) de treinamento

Neste sentido, é interessante notar também que, enquanto pode-se notar um ponto de partida em torno de 88% para a época 1 na figura 11, tem-se que os *batches* desta época (figura 12) possuem a maior variação de acurácia entre todas. É possível observar pela última figura colocada que a época 1 começa com 10,5% de acurácia e finaliza com 88%; a

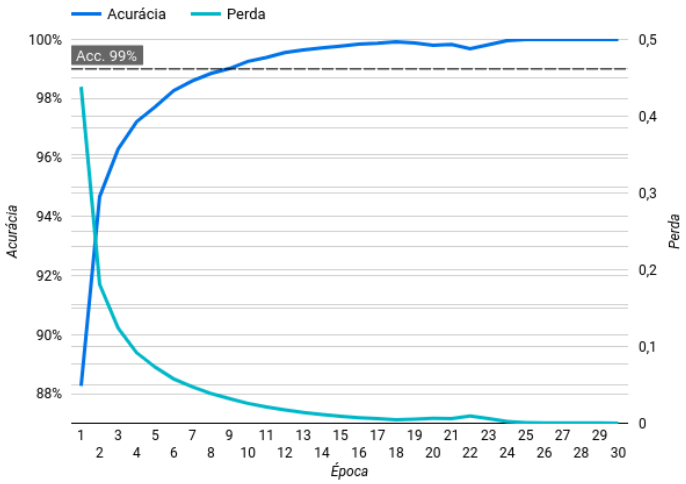

Figura 11. Experimento 2 - Acurácia e perda - Época $\leq$ 30

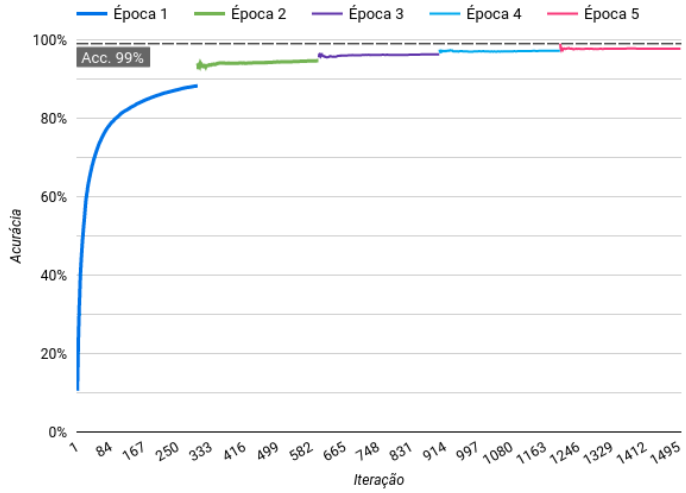

Figura 12. Experimento 2 - Acurácia e perda por época e *batch*

partir desse momento, as demais épocas (iterações) têm um crescimento linear ao invés de exponencial.

Referências

[1] Learning From Data MOOC - The Lectures. Acesso em 15/09/2022. https://work.caltech.edu/lectures.html

[2] Evsukoff, Alexandre G. Inteligência Computacional I. Disciplina de graduação da Escola Politécnica (UFRJ).

[3] Almeida, Heraldo L. Introdução ao Aprendizado de Máquina. Disciplina de graduação da Escola Politécnica (UFRJ).

[4] Handwritten digit recognition with MNIST on iOS with Keras — by Eridy Lukau — Towards Data Science. Acesso em 15/09/2022. https://towardsdatascience.com/handwritten-digit-recognition-with-mnist-on-ios-with-keras-e85e194f9fa5