Instituto Federal de Minas Gerais
Campus Bambuí
Departamento de Engenharia e Computação
Engenharia de Computação

DNN para classificação de mãos do pôquer

Nesta atividade da disciplina de Redes Neurais Artificiais você deverá trabalhar individualmente. Com base no código fonte pronto que implementa uma *Deep Neural Network*(DNN) com otimização via *Adam* visto em sala, você deverá implementar uma arquitetura de rede neural profunda usando Keras e Python para classificar corretamente mãos do pôquer com base nas cartas que um jogador tem em mãos. Este *dataset* faz parte do repositório da UCI[1] e está em anexo.

Perceba que o *dataset* completo da UCI contém 1025010 amostras. Você pode baixar esse *dataset* e tentar treinar com ele se tiver um bom computador ou uma placa de vídeo dedicada, a ideia é que sua rede se comporte melhor se for treinada no conjunto completo. No entanto, a versão do *dataset* fornecida em anexo para este trabalho é um subconjunto contendo 25010 amostras. Sua rede deve dar conta do recado usando esse *dataset* menor. Outra opção é usar o Google Colab[2] caso você queira usar o completo ou apenas ganhar desempenho no menor.

Quando for implementar o modelo, observe que existe neste *dataset* um valor em decimal para representar a pinta da carta (1 a 4) e outro para o valor da carta (1 a 13). Esse código em decimal deve ser convertido para binário[3] de forma que esse modelo receba apenas um vetor contendo valores 0 ou 1. Como cada mão tem cinco cartas, então teremos 30 valores de entrada, sendo 2 bits para representar a pinta e 4 para representar o valor da carta. A saída também deverá ser binária usando quatro bits para representar a mão do pôquer que o jogador tem (consulte a planilha em anexo que traz um melhor detalhamento dessa estrutura). O modelo receberá 30 valores de entrada em binário e deve gerar 4 valores em binário na saída. Os valores em binário na saída devem ser interpretados de forma que a taxa de acerto deve ser calculada no final.

Existem várias abordagens para solucionar esse problema usando modelos de redes neurais profundas. Se você quiser fazer algo diferente, sugiro usar uma função *softmax* na saída do modelo para estimar a probabilidade de ser uma mão ou outra. Nesse caso sua rede terá 9 saídas possíveis. Vasculhe a documentação do *scikit-learn* se decidir fazer isso.

Neste trabalho as seguintes tarefas deverão ser desenvolvidas:

1. Crie um Jupyter Notebook e carregue o *dataset*.

2. Analise e trate os dados. Mas trate apenas se julgar que isso realmente é necessário e promoverá uma melhoria na classificação. Já adianto que remover *outliers* e estabilizar variância não dá muito certo nesse experimento.

[1] https://archive.ics.uci.edu/ml/datasets/Poker+Hand
[2] https://colab.research.google.com/
[3] Procure pela função *OneHot Encoding* do *scikit-learn*.

3. Divida o *dataset* nos conjuntos de treino, validação e teste.

4. Elabore um modelo de DNN da forma como quiser usando o Keras.

5. Treine o modelo nos dados de treino e valide ele nos dados de validação.

6. Verifique se a curva de convergência está se comportando como esperado, senão, ajuste os parâmetros até achar que o modelo está sendo treinado corretamente.

7. Por fim, aplique o modelo ao conjunto de teste.

8. Calcule a taxa de acerto no conjunto de teste. O que ela mostra? O modelo esta conseguindo realizar a classificação corretamente?

Você deverá entregar no AVA dois arquivos. O primeiro é o seu notebook do Jupyter. O segundo é um relatório em PDF contendo uma descrição da arquitetura que você escolheu, a curva de convergência e uma discussão dos resultados da classificação.

A data de entrega é a que consta no AVA.