Reconhecimente de números usando Redes Neurais Artificiais

Mario Moura Pedro Honório

20 de abril de 2021

Resumo

Um dos tópicos clássicos que exploram o poder das redes neurais sem se perder na complexidade é a identificação de símbolos e hieroglifos. Tarefa que a olhos destreinados pode parecer extremamente trivial, entretanto quando examinada mais de perto se torna algo inimaginável para se traduzir de uma maneira imperativa. A técnica de representar problemas com pseudo neurónios e alterar o processamento deles através do *backpropagation* veio para auxiliar em tarefas muito abstratas. Utilizando essas técnicas é possível identificar números simples de zero à nove.

Palavras-chave: Redes Neurais. Inteligência Artificial. Python.

1 Problema

O problema da identificação de algarismos de zero(0) à nove(9) tem como objetivo a classificação de imagens, com resolução $28 \times 28 pixels$, em uma das 10 categorias. Cada uma representando um número de um dígito. São necessários então imagens de dígitos escritos as mão e suas respectivas respostas. O *DeepAi*[1] disponibiliza um conjunto de dados batizado de "*MNIST*", contendo 60000 exemplos para treinamento e 10000 exemplos para teste.

1.1 Arquitetura RNA

Como as imagens possuem 28×28 de resolução, cada uma representa 784 *pixels* obtendo-se 784 neurônios de entrada e como são dez possíveis classificações foi escolhido 10 neurônios como saída. O algoritmo de *stochastic gradient descent* foi escolhido utilizando uma função sigmoidal como processamento dos neurônios.

Todos os neurônios aceitam pontos flutuantes de 0.00 à 1.00 e retornam também números de 0.00 à 1.00.

[1] https://deepai.org

1.2 Tratamento de Codificação de dados

O *set* de dados do MNIST provém vetores representando cada, os *pixels* das imagens, variando entre 0 e 255 numa escala de quão cinza é aquele pixel, sendo 0 para totalmente preto e 255 para branco. Esses são os arquivos intitulados "*images*". Os outros arquivos, intitulados de "*labels*" são as repostas das suas respectivas imagens, também num formato vetorial.

O Vetor de imagens é transformado num vetor de x entradas sendo x o número de imagens no arquivo e cada entrada sendo um outro vetor contendo os *pixels* de uma imagem em sequência, então as respostas serão transformadas em vetores com dez posições zeradas onde apenas a posição cujo o índice é a resposta recebe o valor um. Finalmente a matriz de *pixels* é associada ao vetor de respostas(lembrando que cada entrada no vetor de respostas é um vetor de 10 posições) no formato de lista.

- Matriz de treino (60000×784): [[0,0,...,254,...,0],[0,0,...,125,194,255,...,0,0],...]
- Matriz de respostas (60000×10): [[0,0,0,0,0,0,0,1,0,0,],...,[0,0,1,0,0,0,0,0,0,0,]]
- Matriz de teste (10000×784): [[0,0,...,254,...,0],[0,0,...,125,194,255,...,0,0],...]
- Vetor de respostas (10000): [2,5,4,7,3,0,9,1,6,3,...,4,1,3,2,0,7,8,3,7,4]

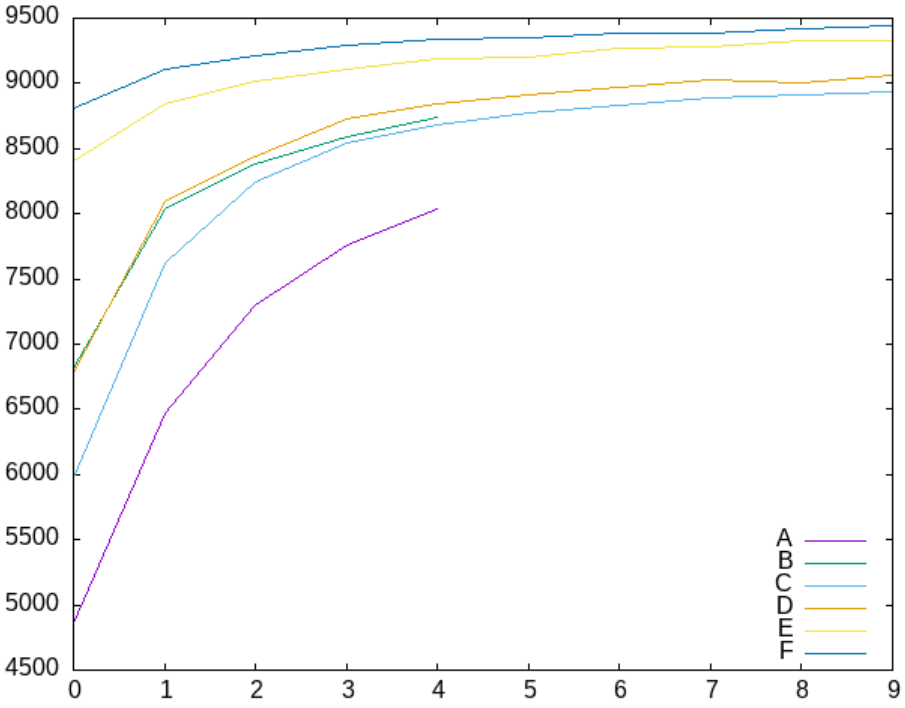
Todas as matrizes podem ser encontradas no site do *DeepAi*[2] na forma vetorial de e num arquivo binário.

2 Implementação

Rede	Camada Intermediária	Taxa de aprendizagem	Épocas	Taxa de Acerto
A	10	0.1	5	80.38%
B	15	0.2	5	87.37%
C	15	0.2	10	89.37%
D	15, 15	0,3	10	90.64%
E	30, 30	0,5	10	93.28%
F	50, 50	0,7	10	94.43%

[2] https://deepai.org/dataset/mnist

Figura 1 – Evolução das redes por época



3 Análise dos Resultados

Todos os resultados foram relativamente aceitáveis, entretanto devido a grande quantidade de dados não foi necessário muitas épocas. Foi escolhido nesse caso utilizar-se de múltiplas camadas ao invés de uma única camada gigante justamente pela grande camada de neurônios iniciais. Uma grande dificuldade do modelo é superar taxas de aceto superiores à 90%, à solução foi justamente aumentar o numero de camadas intermediárias juntamente com a taxa de aprendizado levemente.

Interessante notar as curvas das redes no gráfico 1, que obedecem um padrão de declínio. Outro detalhe interessante à se notar é a simplicidade do algorítimo que se faz eficaz justamente pela grande disponibilidade de dados para ser treinado.

APÊNDICE A – Scripts