Aprendizado de Máquina
Trabalho Prático 1

Luís Felipe Ramos Ferreira
2019022553

May 17, 2023

1 Introdução

O Trabalho Prático 1 da disciplina de Aprendizado de Máquina teve como tema a criação de uma rede neuronal para classificação de dígitos escritos a mão, mais especificamente o conhecido conjunto de dados MNIST. Os objetivos do trabalho envolviam analisar como o modelo da rede neuronal iria variar na convergência do erro empírico conforme são modificadas diferentes variáveis de configuração da rede, como a taxa de aprendizado, a quantidade de neurônios na camada oculta e diferentes algoritmos de cálculo de gradiente.

2 Implementação

A linguagem escolhida para o desenvolvimento do trabalho foi Python (versão 3.10), devida a sua grande variedade de bibliotecas úteis para ciência de dados e aprendizado de máquina. A modelagem da rede neuronal foi feita com o uso da API keras disponibilizada na biblioteca TensorFlow, uma vez que se tratava de uma ferramenta extremamente completa para todos os objetivos do trabalho que permitia grande flexibilidade na modelagem da rede.

Para organizar o ambiente, que englobava várias bibliotecas diferentes, foi utilizado o gerenciador de pacotes Anaconda, o que tornou muito mais fácil trabalhar com os pacotes de ciência de dados citados.

3 Experimentos

Os experimentos foram realizados sobre uma subparte da base de dados MNIST, disponibilizados no enunciado do trabalho. Essa parte possui um total de 5000 instâncias, que foram divididas em conjunto de treino (90%) e conjunto de teste (10%).

Conforme especificado, foram testados e comparados os resultados da rede neuronal na classificação dos dígitos para diferentes parâmetros de modelagem. Mais especificamente, todas as permutações das seguintes configurações foram utilizadas:

- Número de neurônios na camada oculta
  - 25
  - 50
  - 100
- Algoritmo de cálculo de gradiente
  - Gradient Descent (O gradiente é calculado após cada época, neste caso, 5000 entradas)
  - Stochastic Gradient Descent (O gradiente é calculado após cada entrada)
  - Mini batch (O gradiente é calculado após um certo número de entradas, neste caso, 10 e 50)
- Taxa de aprendizado

– 0.5
– 1.0
– 10.0

O resultado obtido para cada umas das 36 configurações foi armazenado em um arquivo JSON, de modo que sua leitura e manipulação fosse facilitada, e assim bibliotecas de análise de dados como Pandas pudessem ser utilizadas para interpretar como se comportou cada parametrização do modelo durante o treino.

4 Análise dos resultados

De maneira geral, os dados gerados deixaram muito claro quais as configurações que permitiam uma boa e uma má performance do modelo. As duas tabelas a seguir mostram os parâmetros das 5 configurações que obtiveram o menor valor de acurácia no conjunto de treino, assim como as 5 configurações com o maior valor de acurácia.

5 piores desempenhos

Tamanho da camada oculta	Tamanho do lote	Taxa de aprendizado	Acurácia
25	1	10.0	0.094
100	4500	10.0	0.100
100	1	10.0	0.108
50	1	10.0	0.146
25	4500	10.0	0.226

5 melhores desempenhos

Tamanho da camada oculta	Tamanho do lote	Taxa de aprendizado	Acurácia
100	10	1.0	0.944
100	50	1.0	0.942
50	10	1.0	0.940
50	50	1.0	0.936
50	50	0.5	0.934

Em primeiro lugar, evidencia-se o fato de que todos os 5 piores valores de acurácia obtidos nos dados de treino ocorreram em modelos com uma taxa de aprendizado igual a 10. Isso indica que uma taxa de aprendizado alta não é uma boa opção para o treino de uma rede neural, devido ao fato de que uma alta taxa leva a uma grande quantidade de oscilações durante o treino, gerando alta divergência do erro empírico.

Outro fato relevante a se notar é o tamanho dos lotes que geraram os piores resultados. Os modelos em questão utilizavam ou o algoritmo Gradient Descent (tamanho de lote igual a 4500)