Trabalho Prático I

Igor Lacerda Faria da Silva

igorlfs@ufmg.br

1 Introdução

O trabalho consiste em montar uma rede neural para identificar dígitos escritos à mão, do banco de dados do `MNIST`. O propósito desse relatório é comparar algumas variações dessa rede neural. A rede possui apenas 3 camadas e as variações exploram diferentes algoritmos, taxas de aprendizado e número de neurônios na camada oculta.

A comparação é dividida em 3 partes: primeiro é fixado o algoritmo e a taxa de aprendizado, modificando-se o tamanho da camada oculta. Similarmente, a taxa é alterada e, por fim, o algoritmo. Os 3 algoritmos implementados (*Stochastic Gradient Descent*, *Gradient Descent* e *Mini-Batch*) consistiram apenas em modificações de parâmetros simples da biblioteca.

A biblioteca utilizada foi o `TensorFlow`. Ela permite carregar o banco de dados do MNIST diretamente e possui uma `API` conveniente para criar os modelos.

2 Desenvolvimento

Foram analisadas duas métricas: a acurácia e a perda.

2.1 Tamanho da Camada Oculta

Foi fixada uma taxa de aprendizado $l_R = 1$ e um algoritmo de *Mini-Batch*, com o tamanho da *batch* igual a 50.

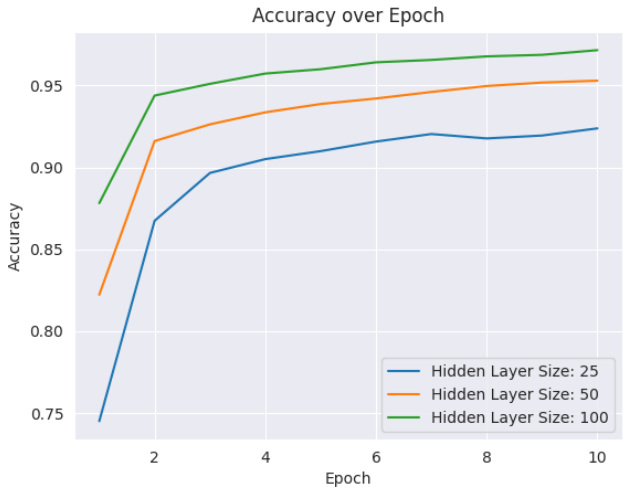

Figura 1: Acurácia por época, variando o tamanho da rede.

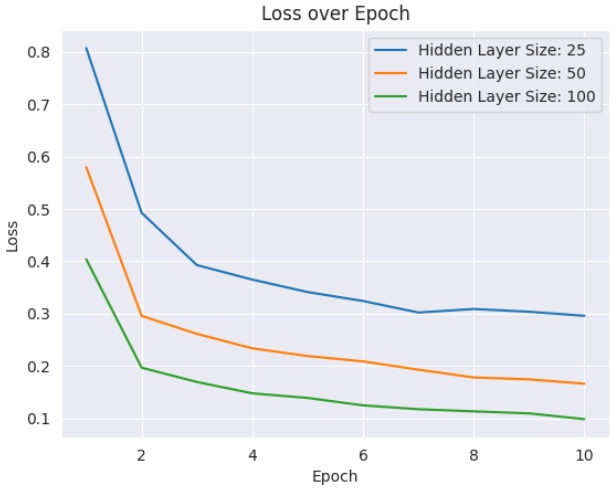

Figura 2: Perda por época, variando o tamanho da rede.

Como é de se esperar, o aumento do tamanho da rede melhorou a performance. Existem execuções em que não é o caso, apesar de ser uma tendência comum. Com mais neurônios é possível fazer ajustes mais sensíveis aos dados. Ao que parece, no entanto, com apenas 100 neurônios ainda não há *overfitting*.

2.2 Taxa de Aprendizado

Foi fixada um tamanho de camada oculta igual a 50, e o algoritmo de *Mini-Batch*, também com tamanho de *batch* igual a 50.

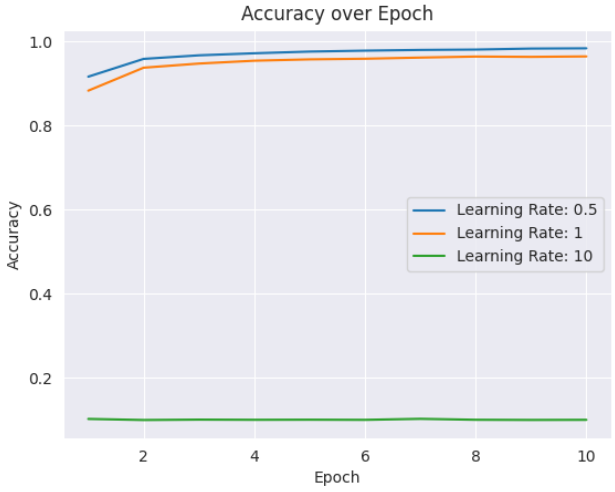

Figura 3: Acurácia por época, variando a taxa de aprendizado.

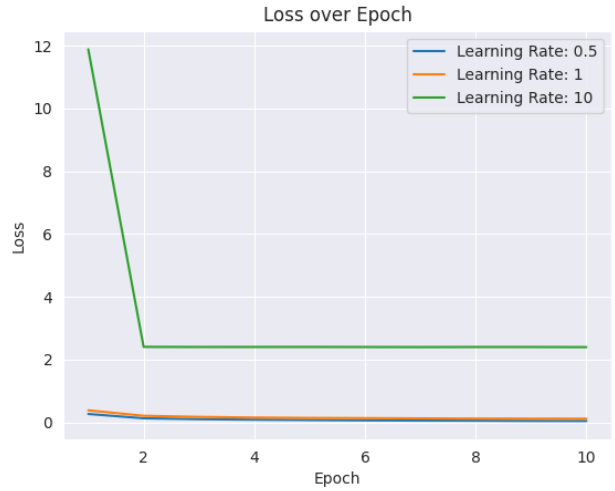

Figura 4: Perda por época, variando a taxa de aprendizado.

Como visto em aula, uma taxa de aprendizado muito alta pode prejudicar a qualidade do modelo. Isso acontece porque a função de perda "pula" possíveis mínimos locais. Apesar de ter convergido nessa execução, o desempenho para

a taxa igual a 10 não é diferente de um modelo que chuta que todos os dígitos são o número 9. Por outro lado, as taxas de 0.5 e 1, para esses parâmetros e, principalmente, essa execução, foram muito próximos. Uma taxa de aprendizado muito baixa também poderia demorar a convergir. O ideal é encontrar um meio termo, que nesse caso parece ser um número próximo de 0.5 e 1.

2.3 Algoritmo

As variações de algoritmo foram controladas alterando-se o parâmetro *batch_size*, da função *fit()* do modelo. Em princípio, todos os algoritmos são o *Mini-Batch*. Para criar um SGD, basta usar uma *batch* de tamanho 1 e para criar um GD, é preciso usar uma taxa de aprendizado igual ao número de dados de treino. O número de épocas também precisou ser ajeitado: é preciso dar mais oportunidades de ajustes no GD, por exemplo (e o tempo de execução do SGD é muito grande). Nos gráficos, o eixo x representa o "progresso" do algoritmo, cada ponto corresponde a uma época. A taxa de aprendizado foi fixada em 1, e o número de neurônios fixado em 50.

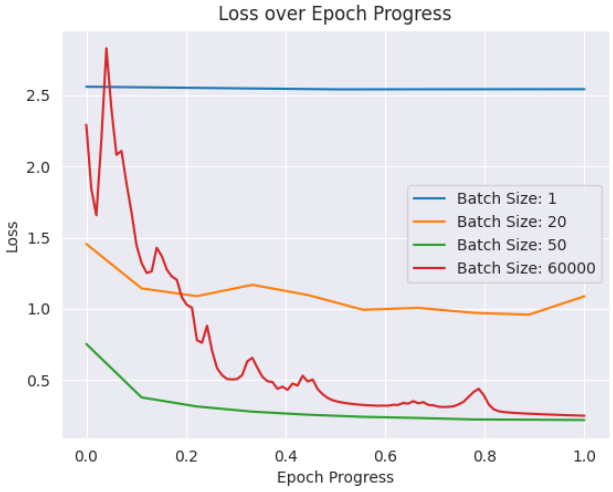

Figura 5: Acurácia por progresso em épocas, variando o algoritmo.

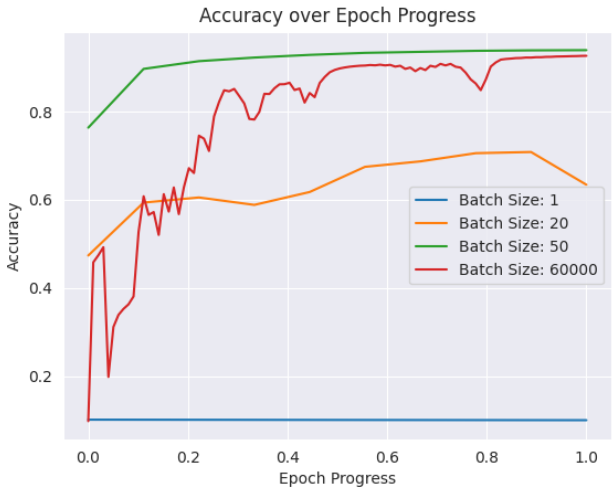

Figura 6: Acurácia por progresso em épocas, variando o algoritmo.

O SGD teve um péssimo desempenho, também indiferente a apenas chutar algum número em particular. Como são feitos ajustes a cada dado, é como se o SGD "decorasse" o dado em questão. O GD, com 100 épocas, teve um desempenho similar ao MB ($batch\_size = 50$). Com a diminuição do tamanho da *batch*, o desempenho piorou consideravelmente.