Lab 6 Supervised Learning with Tensorflow and Deep Neural Networks

Martin Gustafsson (margu424) and Axel Gard (axega544)

2020-10-08

Question 1) In the gradient descent learning code below, please complete the gradient computation by inputing the correct variables where there are question marks. We are using Tensorflow to automatically compute the gradient with GradientTape (tape) similar to how you were taught above, but to do supervised learning which tensorflow node do we compute the gradient of, and with regard to which variable? Please fill this in below (and in your lab report). If you get stuck, the slides on training models in the first ML lecture might be helpful.

Answer:

We compute the gradient of the loss function with regards to the parameters. which means we input loss and parameters to the gradient function.

```
gradient = tape.gradient(loss, parameters)
```

Question 2) Show the math for why the first Dense layer has 100 480 parameters with these inputs and number of neurons. Remember, a neuron is just a non-linear transformation (e.g. sigmoid/ReLU) of a linear model, implying one parameter for each input dimension, + 1 for the line intercept/constant (also called neuron "bias" in NN slides from the first ML lecture).

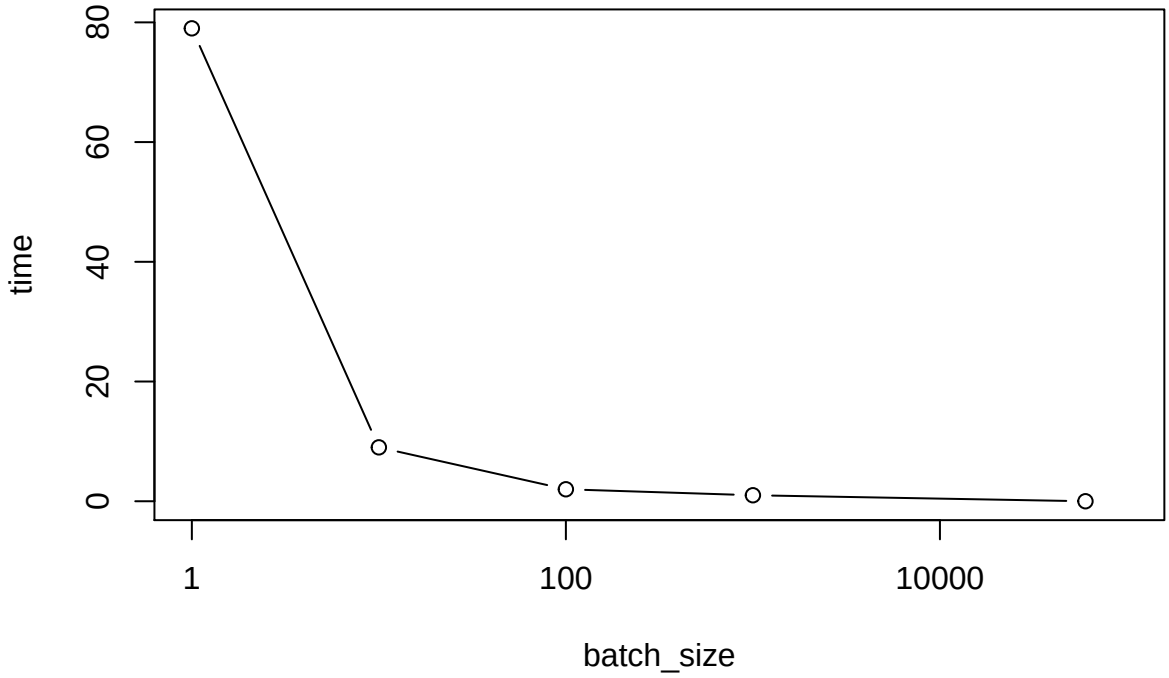
Answer: first Dense layer parameters = 128 * (28*28 + 1) = 100480

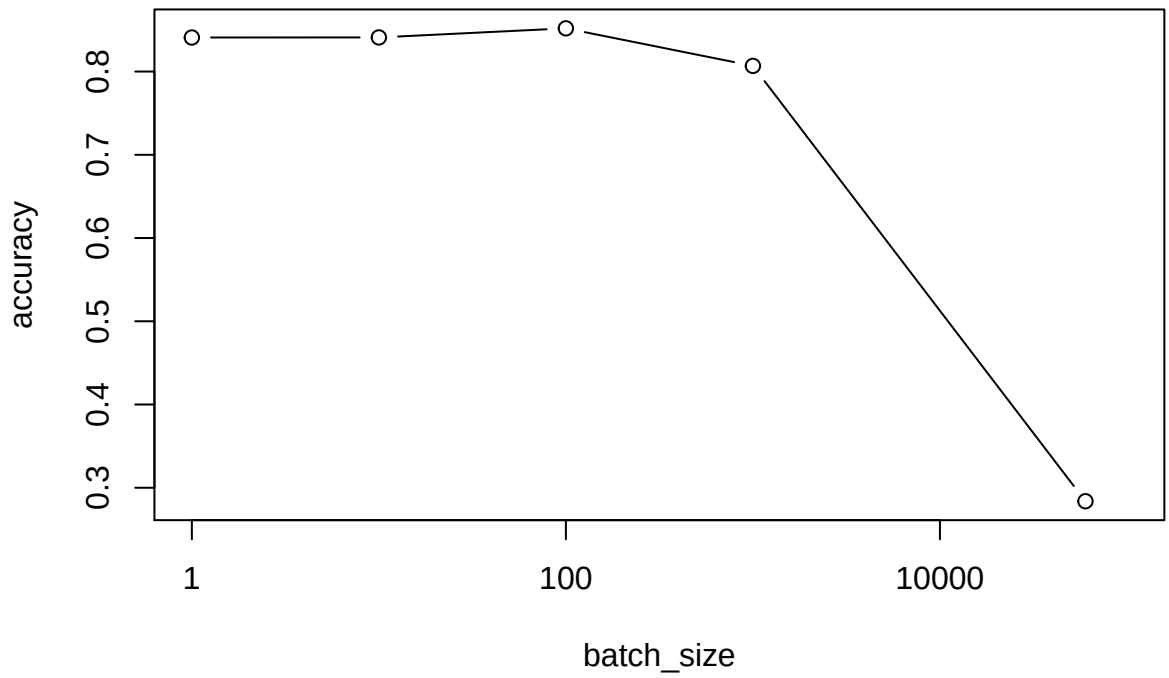
neurons * (input_size + 1)

Question 3 Here you will evaluate different mini-bach sizes for stochastic gradient descent (see the deep learning lecture). Please separately run the training code above with batch sizes of 1, 10, 100, 1000 and 60000. Write down the training times (you can use the first number in seconds, not the per sample time) and the training set accuracy reached, both in the first line of the output. This can randomly vary a bit between runs but it should give you an idea. In your lab report, plot both curves and reason about which batch size produced the most accuracy given the time spent, i.e. which batch size would be best to start the training with? You have to run the Reset All Parameters code above this one between your runs to always start over.

Answer:

batch_size	time	accuracy
1	79s	0.8409
10	9s	0.8410
100	2s	0.8519
1000	1s	0.8068
60000	0s	0.2838


time
0
20
40
60
80
1
100
10000
batch_size



We see in the graph for accuracy that it stays similar up until a batch size of 1000 were the accuracy starts to drop off a little and at 60000 the accuracy nosedives. We also see in the graph for time that the time decreases drastically in the beginning but the time for the batch size of 1000 is not that much faster than the batch size 100. Therefore we would use a batch size of 100 as that gives us a good accuracy and a relatively good time.