PSL432 Theoretical Physiology
Department of Physiology
University of Toronto

ASSIGNMENT 2

Assigned 25 February 2020
Due 10 March 2020

This assignment is worth 40% of your final mark in the course. Please do all parts of both questions. For each question you will submit computer code as m-files. Please email these m-files to douglas.tweed@utoronto.ca and ali.mojdeh@mail.utoronto.ca by 11:59 p.m. on 10 March. Your code will be graded partly on concision and clarity, so keep it brief and orderly.

In what follows, the symbol Ω stands for the string Yoursurname_Yourinitial, e.g. in my case, Tweed_D. And ω stands for yoursurname_yourinitial, e.g. tweed_d.

1. Learning with a retinotopic map

Use backpropagation with adam to train a network to recognize the handwritten digits in the MNIST data set. The network should be a 3-layer mapnet where the middle layer is a 32-by-32 map with a field-fraction of 0.25, and the connections from the middle to the output layer are dense. In the middle layer, use the *rectified tanh* activation function, i.e. set $\boldsymbol{y} = \max(0, \tanh(\boldsymbol{v}))$. And in the output layer use the *softmax* function, $y_i = \exp(v_i)/\Sigma_j \exp(v_j)$. Define the error and loss as usual, $\boldsymbol{e} = \boldsymbol{y} - \boldsymbol{y}^*$ and $L = 0.5\boldsymbol{e}^\mathsf{T}\boldsymbol{e}$. (In the literature, a different loss function is often used together with softmax, but here I want you to use the squared-error loss.).

To make the mapnet, write an m-file called ω_create_mapnet, analogous to the posted m-file create_net. In your file, initialize each cell's weights W by drawing them from a normal distribution with a mean of 0 and standard deviation of sqrt(2/N), where N is the number of the cell's inputs, i.e. the number of upstream neurons projecting to the cell. Initialize all biases b to 0.

For learning, write m-files ω_forward_rectanh and ω_backprop_rectanh, analogous to the posted m-files forward_relu and backprop_relu, and implement adam using the posted m-file adam. Set the adam η = 0.0005 and the other adam hyperparameters to their usual values.

You will find the MNIST data set in Quercus, in the file MNIST.mat, along with MNIST_key.m, which explains the format of those data. Load the data set as in MNIST_key.m. Feed the images to the network in minibatches of 100 each. If you can avoid it, you should not write any code, anywhere in your files, that uses a loop or list to go through the minibatch example by example; your code will run much faster if you write it so all operations are done on the whole minibatch array at once, as in the posted sample code.

You will need several epochs — several passes through all 60,000 training examples — to learn the task. At the start of each epoch, reshuffle the data, as shown in MNIST_key.m (this shuffling can aid learning, as it supplies a different series of 600 minibatches to the network in different epochs).

After each epoch, run your network on the entire test set (in one big "minibatch" of all 10,000 test examples), count the number of incorrect guesses, and display that number in the Command Window. If all goes well, then on most runs, the number of incorrect guesses should be ~300 or better after one epoch and 180 or better after 10 epochs, i.e. 98.2% correct.

Debugging hint. Once you have set up your network with its rectanh map and softmax output layer, you should be able to get good learning (driving the incorrect guesses down to ~180) by starting backprop with the error vector $\boldsymbol{e}$, rather than the proper $\partial L/\partial \boldsymbol{v}\{n_l\}$ that goes with the squared error loss. *To get full marks on this question, you must NOT start backprop with* $\boldsymbol{e}$ *in this way in your final submission,* but instead use the $\partial L/\partial \boldsymbol{v}\{n_l\}$ that goes with the squared error loss. *But for purposes of debugging*, if you haven't yet figured out $\partial L/\partial \boldsymbol{v}\{n_l\}$ and you want to check whether the *rest* of your network is correct, then you can test whether it learns well when you backpropagate $\boldsymbol{e}$.

Submit an m-file called Ω_A2_Q1, along with ω_create_mapnet, ω_forward_rectanh, and ω_backprop_rectanh.

2. Nesterov momentum

Here you will modify your code from Question 1 to learn the same task with the same map network, but using relu in place of rectanh, and Nesterov momentum in place of adam. You will create your map-net with ω_create_mapnet.m in the same way as before, using the same initialization, but for its forward and backward passes you will use the posted m-files forward_relu and backprop_relu.

In this question, please initiate backprop using the error vector $\boldsymbol{e} = \boldsymbol{y} - \boldsymbol{y}^*$, not the $\partial L/\partial \boldsymbol{v}\{n_l\}$ that goes with the squared error loss (this way, your success on this question won't depend on whether you figured out $\partial L/\partial \boldsymbol{v}\{n_l\}$ in Question 1).

To implement Nesterov momentum, you will write an m-file called ω_nesterov, which you will call immediately after backprop_relu, just as I did with adam in Compare_descent_complex.m. Your nesterov file should not require that you do any other calculations between forward_relu and backprop_relu apart from finding the error, and it should not compute any gradients itself, but should use the gradients computed beforehand by backprop_relu, just as adam.m did. Do not change forward_relu.m or backprop_relu.m, but use the posted versions as they are, and write your nesterov file to work with them. To implement Nesterov momentum in this way, you will have to store information in the net.W_, net.W__, net.b_, and net.b__ arrays, in a way somewhat analogous to what adam.m does, though the details will be quite different. Set your Nesterov $\eta = 0.05$ and $\beta = 0.9$.

If all goes well, the number of incorrect guesses should usually be ~400 or better after one epoch and 170 or better after 10 epochs. Submit your code in an m-file called Ω_A2_ Q2.m, along with ω_nesterov.m.