Deep Learning Course – Assignment 1
Feedforward Neural Networks, Optimizations and Regularizations

Submission date: 28/04/21, 23:55
Please submit the assignment via Moodle

Assignment Instruction:
· Solitary submissions or in pairs.
· The code must be written in Python 3.7.1 and run on GPU in google colab
· You are allowed to use only Pytorch, numpy and matplotlib as external libraries
· The code must be reasonably documented
· You are not allowed to change function declarations

In class we've learned about MNIST data set, which can be loaded from torchvision.datasets. We built a feedforward network for MNIST classification with accuracy of ~96%.
In this assignment we will use Fashion-MNIST, which is also a built-in dataset in Pytorch. We'll try different networks and find which one fits better.

Please implement the following modules and functions:

1. Function 1: load_dataset()
   Load the Fashion-MNIST training and test set from Pytorch. Split the training set to training and validation set (80%-20% ratio). Define a DataLoader for each set. In the DataLoader parameters use "batch_size"=64.
2. Function 2: one_hidden_layer_no_activation(number_of_neurons)
   For this function, you should implement the following steps:
   - Define the model: feed-forward network with one hidden layer, no activation functions are applied on the hidden layer (linear). The number of neurons in the hidden layer is given by the parameter "number_of_neurons". The output layer activation function is log softmax (recall what should be the dim of the output layer as we learned in class).
   - Train the network on the Fashion-MNIST training set for 50 epochs
   - Use cross entropy as a cost function
   - Use the optimizer:
     learning_rate = 0.01
     optimizer = torch.optim.SGD(model.parameters(),lr=learning_rate)
   - Plot a figure with the accuracy on the training set and validation set for each epoch
   - Plot the accuracy on the test set of the final model
3. Function 3: two_hidden_layers_sigmoid(number_of_neurons)
   - Same as Function 2, with 2 hidden layers
   - Use sigmoid as activation function
   - Train for 20 epochs with lr = 0.1
   - Plot a figure with the accuracy on the training set and validation set for each epoch
   - Plot the accuracy on the test set of the final model
4. Function 4: two_hidden_layers_relu(number_of_neurons)
   - Same as Function 2, with 2 hidden layers
   - Use relu as activation function
   - Train for 20 epochs

- Train the network with different learning rates, plot a figure with the validation accuracy for each learning rate you tried
- Use the learning rate that yields the best accuracy on the validation set and report the accuracy on the test set
- Plot a figure with the accuracy on the training set and validation set for each epoch
- Plot the accuracy on the test set of the final model

5. Function 5: two_hidden_layers_relu_SGD_decreasing_lr (number_of_neurons)
    - Same as Function 4
    - Use lr = 0.01
    - Use decreasing learning rate
    - Plot a figure with the accuracy on the training set and validation set for each epoch
    - Plot the accuracy on the test set of the final model
6. Function 6: two_hidden_layers_relu_adam (number_of_neurons)
    - Same as Function 4
    - Use Adam as optimizer with lr=0.001
    - Train for 30 epochs
    - Plot a figure with the accuracy on the training set and validation set for each epoch
    - Plot the accuracy on the test set of the final model
7. Function 7: four_hidden_layers_adam (number_of_neurons)
    - Same as Function 6, but use 4 hidden layers instead of 2
    - Train for 30 epochs
    - Use Adam as optimizer
    - Plot a figure with the accuracy on the training set and validation set for each epoch
    - Plot the accuracy on the test set of the final model
8. Function 8: four_hidden_layers_adam_weight_decay (number_of_neurons)
    - Same as Function 6, but use 4 hidden layers instead of 2
    - Use only 10% of the training set for training
    - Train for 250 epochs
    - Use Adam as optimizer with weight decay as regularization method
    - Plot a figure of the training and validation loss function during the training process. Write in your report what's the difference between training with and without weight decay? Based your answer on the plots.
    - Plot a figure with the accuracy on the training set and validation set for each epoch
    - Plot the accuracy on the test set of the final model
9. Function 9: four_hidden_layers_adam_early_stopping (number_of_neurons)
    - Same as Function 6, but use 4 hidden layers instead of 2
    - Use Adam as optimizer
    - Use only 10% of the training set for training
    - Train for 250 epochs
    - Use early stopping regularization (on the validation set) to prevent overfitting with patience=20.
    - Plot a figure of the training and validation loss function during the training process. Show on your figure the early stopping point.

- Plot a figure with the accuracy on the training set and validation set for each epoch
- Plot the accuracy on the test set of the final model

Run the following commands:

load_dataset()

# 4 neurons per layer
number_of_neurons = 4
one_hidden_layer_no_activation(number_of_neurons)
two_hidden_layers_sigmoid(number_of_neurons)
two_hidden_layers_relu(number_of_neurons)
two_hidden_layers_relu_SGD_decreasing_lr (number_of_neurons)
two_hidden_layers_relu_adam (number_of_neurons)

# 32 neurons per layer
number_of_neurons = 32
one_hidden_layer_no_activation(number_of_neurons)
two_hidden_layers_sigmoid(number_of_neurons)
two_hidden_layers_relu(number_of_neurons)
two_hidden_layers_relu_SGD_decreasing_lr (number_of_neurons)
two_hidden_layers_relu_adam (number_of_neurons)
four_hidden_layers_adam (number_of_neurons)
four_hidden_layers_adam_weight_decay (number_of_neurons)
four_hidden_layers_adam_early_stopping (number_of_neurons)

After implementing, running and getting the results write the following to a "doc/docx" file format:

1. The networks sorted according to their test results from the worst to the best.
2. Give intuitive explanation why we got such order after sorting.
3. Is there any difference between the test and the training accuracy? Why?
4. Don't forget to add all the function plots to your report.