8. Fully-Connected Neural Networks

First, we will employ the most basic form of a deep neural network, in which the neurons in adjacent layers are fully connected to one another.

You will be working in the files`part2-mnist/nnet_fc.py`**in this problem**

Training and Testing Accuracy Over Time

1.0/1.0 point (graded)

We have provided a toy example **nnet_fc.py** in which we have implemented for you a simple neural network. This network has one hidden layer of 10 neurons with a rectified linear unit (ReLU) nonlinearity, as well as an output layer of 10 neurons (one for each digit class). Finally, a softmax function normalizes the activations of the output neurons so that they specify a probability distribution. Reference the PyTorch Documentation and read through it in order to gain a better understanding of the code. Then, try running the code on your computer with the command `python3 nnet_fc.py`. This will train the network with 10 epochs, where an epoch is a complete pass through the training dataset. Total training time of your network should take no more than a couple of minutes. At the end of training, your model should have an accuracy of more than $\%85$ on test data.

Note: We are not using a softmax layer because it is already present in the loss: PyTorch's `nn.CrossEntropyLoss` combines `nn.LogSoftMax` with `nn.NLLLoss`.

Report the test accuracy below.

Test Accuracy = 0.9204727564102564 ✓ **Answer:** 0.9204727564102564

Submit You have used 1 of 3 attempts

Answers are displayed within the problem

Improving Accuracy

5.0/5.0 points (graded)

We would like to try to improve the performance of the model by performing a mini grid search over hyper parameters (note that a full grid search should include more values and combinations). To this end, we will use our **baseline model (batch size 32, hidden size 10, learning rate 0.1, momentum 0 and the ReLU activation function)** and modify one parameter each time while keeping all others to the baseline. We will use the validation accuracy of the model after training for 10 epochs. For the LeakyReLU activation function, use the default parameters from pyTorch (negative_slope=0.01).

Note: If you run the model multiple times from the same script, make sure to initialize the **numpy and pytorch random seeds to 12321 before each run**.

Which of the following modifications achieved the highest validation accuracy?

- baseline (no modifications)
- [✓ batch size 64] ✓
- learning rate 0.01

- momentum 0.9
- LeakyReLU activation

What is the validation accuracy of the best performing model?

Validation Accuracy = 0.940020 ✓ **Answer:** 0.9400201612903226

Does the model variation that achieved the highest validation accuracy achieved also the highest test accuracy?

[✓ Yes] ✓

No

Submit You have used 2 of 3 attempts

Answers are displayed within the problem

Improving Accuracy - Hidden 128

3.0/3.0 points (graded)
Modifying the model's architecture is also worth considering. Increase the hidden representation size from 10 to 128 and repeat the grid search over the hyper parameters. This time, what modification achieved the highest validation accuracy?

- baseline (no modifications)
- batch size 64
- learning rate 0.01
- momentum 0.9
- [✓ LeakyReLU activation] ✓

Submit You have used 1 of 3 attempts

Answers are displayed within the problem

Discussion

Show Discussion

Topic: Unit 3 Neural networks (2.5 weeks):Project 3: Digit recognition (Part 2) / 8. Fully-Connected Neural Networks

© All Rights Reserved