Lab Course: Distributed Data Analytics
Exercise Sheet 7

Prof. Dr. Dr. Schmidt-Thieme, Daniela Thyssens
Information Systems and Machine Learning Lab
University of Hildesheim
Submission deadline: Sunday June 26, 23:59PM (on LearnWeb, course code: 3116)

Instructions

Please follow these instructions for solving and submitting the exercise sheet.

1. You should submit two items a) a zipped file containing python scripts and b) a pdf document.
2. In the pdf document you will explain your approach (i.e. how you solved a given problem), and present your results in form of graphs and tables.
3. The submission needs to be made before the deadline, only through learnweb.
4. Unless explicitly stated, you are not allowed to use scikit, sklearn or any other library for solving any part. All implementations must be done by yourself.

https://shonit2096.medium.com/cnn-on-cifar10-data-set-using-pytorch-34be87e09844

https://github.com/rasbt/deeplearning-models/blob/master/pytorch_ipynb/cnn/cnn-lenet5-cifar10.ipynb

Network Analysis: Image Classification - Part 2 (3 points)

In the previous lab, we have implemented CNNs for an image classification task without paying much attention to the training behavior. Unchecked neural networks may or may not work based on their initialization. In this lab, we will try to exercise a fine-grain control over the parameters of the network. For this lab, you will implement the network following architecture and use the CIFAR dataset:

1. conv1: convolution and rectified linear activation (RELU)
2. pool1: max pooling
3. conv2: convolution and rectified linear activation (RELU)
4. conv3: convolution and rectified linear activation (RELU)
5. pool2: max pooling
6. FC1: fully connected layer with rectified linear activation (RELU)
7. FC2: fully connected layer with rectified linear activation (RELU)
8. FC3: fully connected layer with rectified linear activation (RELU)
9. softmax_layer: final output predictions i.e. classify into one of the ten classes.

NOTE: the kernel size, FC layer hidden units etc. is left up to you to design (bigger FC layers would lead to additional model complexity).
You will notice that the network is a little bit poorly designed and that is intentional. We would like the network to be more complex than is needed to overfit to our data. In addition to the increased complexity of the network, we will also reduce the amount of training data to 1/2 and also remove all data augmentations/normalizations.
With this network design and data constraint, you are required to get a baseline result to showcase the model behavior. Please **plot both the training and test accuracy and loss for all minibatches** (not just at the end of training). This can be achieved by interleaving a testing step after every 50-100 minibatch during model training.
This baseline aims to serve as a straw-man example that we can compare our next steps against.

Exercise 1: Normalization Effect (CNN) (7 points)

Now that we have a weak baseline, we can start to investigate how to improve the model performance. We will first try to address the data aspect of model improvement.
Improving data:

1. Data Augmentation: the process of artificially 'increasing' our dataset by adding translation, scaling and flipping to the images to fabricate examples for training.
2. Normalization: Normalizing the input data helps remove the dataset artifacts that can cause poor model performance.

In this task, you are required to add data augmentation and normalization to the dataset and run the training. Compare the training performance (accuracy and loss) to the baseline first with only data augmentation, secondly with only normalization, and lastly with both. **Comment on the difference you observe in the training behavior and the final accuracy as seen on the test set**. NOTE: You should use tensorboard plots.

Exercise 2: Network Regularization (CNN) (5 points)

Having tried to fix the dataset problems, we can have a look at the network itself. Regularization techniques are useful in learning a generalizable solution and help in avoiding overfitting. For the Deep Neural Network regularization is generally achieved by using a dropout technique, L1,L2 regularization among many other techniques. Here we will look at the dropout technique.

- Dropout: Adding dropout layers to a network simply means we stochastically turn off a set number of neurons in our Fully connected layer to prevent the model from overfitting on training data.

Similar to the first task, we are interested in seeing how the addition of network regularization can improve model training behavior and overall performance. You are required to add a dropout layer and report the training and testing loss and accuracy similar to the exercise above. **Comment on the impact of this modification when compared to the baseline approach**.

BONUS (3 points):
Perform the same analysis as for the dropout technique above, also on the following regularization techniques:

- L1 regularization: Penalizing the weights of the network to encourage sparse weights. (1.5 points)
- L2 regularization: Penalizing the weights of the network to encourage them to fit on the unit circle. (1.5 points)

To get any bonus points here you will need to plot and comment on the training and testing loss as well as the accuracy for both the regularization techniques (L1 and L2).

Exercise 3: Optimizers (CNN) (5 points)

In the last part, you will experiment with two different optimizers i.e. SGD and AdamOptimizer, more specifically, how robust they are to the initial learning rate. The choices for your initial learning rate are left up to you. Please compare and contrast the behavior of these two optimizers specifically on how they react when presented with different learning rates. You should plot the training curves as have been requested in the exercises above.

Annex

1. Download original CIFAR-10 dataset from: `https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz`
2. L1 and L2 regularization: https://www.programmersought.com/article/52011513926/

3. CNNs: http://cs231n.github.io/convolutional-networks/
4. CNNs: http://cs231n.github.io/convolutional-networks-1/
5. Data preprocess http://cs231n.github.io/neural-networks-2/
6. http://cs231n.github.io/neural-networks-3/
7. CNN Lecture 1 https://www.youtube.com/watch?v=bNb2fEVKeEo&t=833s&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF index=6
8. CNN Lecture 2 mini-batch http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture6.pdf
9. ADAM vs SGD https://towardsdatascience.com/learning-parameters-part-5-65a2f3583f7d