Lab Course: distributed data analytics
Exercise Sheet 9

Mofassir ul Islam Arif
Information Systems and Machine Learning Lab
University of Hildesheim
Submission deadline: **July 13^{th}, Monday 10:00AM** (on LearnWeb, course code: 3116)

Instructions

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

1. You should submit two items. a) a zip or a tar 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 the form of graphs and tables.
3. The submission should be made before the deadline, only through learnweb.

In this lab we will be using PyTorch to create out neural networks in order compare and contrast against the examples of tensorflow that have been posted on learnweb. HINT: Please create your neural networks as classes and write you own forward passes for the networks. You can find the basics network design in Pytorch here `https://pytorch.org/tutorials/beginner/blitz/neural_networks_tutorial.html`

Exercise 1: Neural Networks on the Olivetti faces dataset (3 points)

First of all, let's look at some neural networks tutorials to understand how it works:

1. Deep Learning Tutorials `http://deeplearning.net/tutorial/`
2. Introduction to Neural Networks
   `https://pythonprogramming.net/neural-networks-machine-learning-tutorial/`
3. Multi-Layer Neural Network
   `http://ufldl.stanford.edu/tutorial/supervised/MultiLayerNeuralNetworks/`

Along with your solution, you might have to follow the proposed procedure.

1. Load the Olivetti faces dataset, randomly split it into training set 90% and test set 10%.
2. Define a learning model using cross entropy cost function. Explain how you come up with the learning model. In this exercise, your model should contain [Input Layer - Hidden Layer - Output Layer]. You are free to design the number of nodes in the hidden layer.
3. Train the model on the training set and make prediction on the test set.
4. Report and plot accuracy on both training set and test set.
5. Create a Tensorboard that presents basic information such as scalars, graphs, distributions, histograms and Images. You might check your Tensorboard at `localhost:6006`.

Exercise 2: Deep Neural Networks on the Olivetti faces dataset (3 points)

Basically, a deep neural network is a neural network with more than 2 hidden layers. In this exercise, you are going to extend the previous neural network model by adding the 2nd hidden layer.

Along with your solution, you might have to follow the proposed procedure.

1. Load the Olivetti faces dataset, randomly split it into training set 90% and test set 10%.
2. Define a learning model using cross entropy cost function. Explain how you come up with the learning model. In this exercise, your model should contain [Input Layer - Hidden Layer 1 - Hidden Layer 2 - Output Layer]. You are free to design the number of nodes in the two hidden layers.
3. Train the model on the training set and make prediction on the test set.
4. Report and plot accuracy on both training set and test set.
5. Create a Tensorboard that presents basic information such as scalars, graphs, distributions, histograms and Images. You might check your Tensorboard at `localhost:6006`.

For Exercise 1 and Exercise 2 please compare the runtime complexity (time taken), model complexity (number of trainable parameters) and percentage accuracy gain by using a deeper network.

Exercise 3: Convolutional Neural Networks on the Olivetti faces dataset (4 points)

A good tutorial on convolutional neural networks can be found in
`http://cs231n.github.io/convolutional-networks/`.

Along with your solution, you might have to follow the proposed procedure.

1. Load the Olivetti faces dataset, randomly split it into training set 90% and test set 10%.
2. Define a learning model using cross entropy cost function. Explain how you come up with the learning model. In this exercise, your model should contain [Input Layer - Convolutional Layer - Max Pooling Layer - Fully Connected Layer - Output Layer]. You are free to design the number of channels in the convolutional layer and the number of nodes in the fully connected layer.
3. Train the model on the training set and make prediction on the test set.
4. Report and plot accuracy on both training set and test set.
5. Create a Tensorboard that presents basic information such as scalars, graphs, distributions, histograms and Images. You might check your Tensorboard at `localhost:6006`.

As done previously, please compare the time taken, model complexity and percentage accuracy gain over the Deep Neural Network from Exercise 2

Exercise 4: Custom Losses in Convolutional Neural Networks on the MNIST dataset (10 points)

So far you have basically just reproduced the networks in different tutorials. In this exercise we will extend a neural network for a custom loss. This exercise will involve some additional reading on your part and learning how to implement a custom loss operation in a multi-task setting using CNNs.

The aim of multi-task learning is to use the learned features of a machine learning task to solve for more than one problem at a time. Object detection is a classic example of multi-task setting where we need to classify and detect objects in images using the same network.

For this task we want to do a simpler version of object detection which I will call Object Regression. We will be using the MNIST dataset which contain 50000 28x28 images of numbers between 0-9. The aim here is to formulate a multi-task solution to

- Classification: Classify the images using a convolutional neural network and a cross entropy loss.
- Regression: solve a regression problem to regress to the numerical value of the number in the image using a regression loss (your choice)

The input space that we have encountered in exercise 1 and 2 always had $X \in \mathbb{R}^{N \times W}$ where N is the number of images and W is the dimension of the images. Each instance $x \in X$ has a corresponding label $y \in Y$ where $Y = \{1, \ldots, N\}$ contains the N possible labels. These labels are catergorical variables and therefore are used in our cross entropy loss. The output of our model is always $\hat{y}_n = f(x_n)$ where (f) is our neural network.

In this exercise our input space during training will include the image, the label (y_n^C) and the ordinal value (y_n^R) of the number in that image. So if we have an image of a "0", we have its image, its label "0" and the ordinal value of 0 as inputs to our network. Furthermore, your network now needs to have a classification head and a regression head so the standard network you have used in the previous exercises needs to be modified as well. You need to include a regression head to the classifier that you have created in the exercise 3. This can be simple dense layer followed by the correct activation function for the terminal output. Your model's final out will also be 2 values i-e the classification label $\hat{y}_n^C$ and the regression value of the number in the image $\hat{y}_n^R$.

In short your neural network can be thought of as having three set of parameters.

- Shared parameters: ϕ
- Classification parameters: ψ
- Regression parameters: η

Your task is to design a neural network that minimizes the following objective function:

$$\arg\min_{\phi,\psi,\eta} \sum_{n=1}^{N} \mathcal{L}\Big(y_n^C, \hat{y}_n^C\big(\psi(\phi(x_n))\big)\Big) + \mathcal{L}\Big(y_n^R, \hat{y}_n^R\big(\psi(\eta(x_n))\big)\Big) \tag{1}$$

Equation 1. showcases a multi-task loss for the classification and regression heads which needs to be jointly minimized. The first term is the standard cross entropy loss where the shared parameters ϕ and fed into the classifier head parameterized by ψ. The second term is the regression head which also takes the shared parameters ϕ but now passes it to the regression head parameterized by η.

This loss needs to be minimized using an optimizer of your choice. Please refer to examples in the appendix to get familiar with the different choices that are possible. Present you final results using the plots for the accuracy for classification, loss for regression , loss of classification and the total loss.

Related reading material

1. Tensorboard tutorial: `https://pytorch.org/docs/stable/tensorboard.html`
2. Multi-Task learning: `https://towardsdatascience.com/multitask-learning-teach-your-ai-more-to-make-`
3. PyTorch Deep learning: `https://pytorch.org/tutorials/beginner/nlp/deep_learning_tutorial.html`
4. PyTorch Classifier: `https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html`