Assignment # 02 (Part 02)

Introduction to Image Analysis and Machine Learning
(Spring 2017)

April 11, 2017

Purpose: In this part of assignment you will use Neural Networks to classify the handwritten digits from the MNIST dataset. You will use the trained classifier to classify the handwritten digits in given videos. Once you have classified the digits successfully, you will integrate the gaze estimation module from part#01 of assignment#02.

Task: Please download exercise resources from learnIT before starting the exercises.

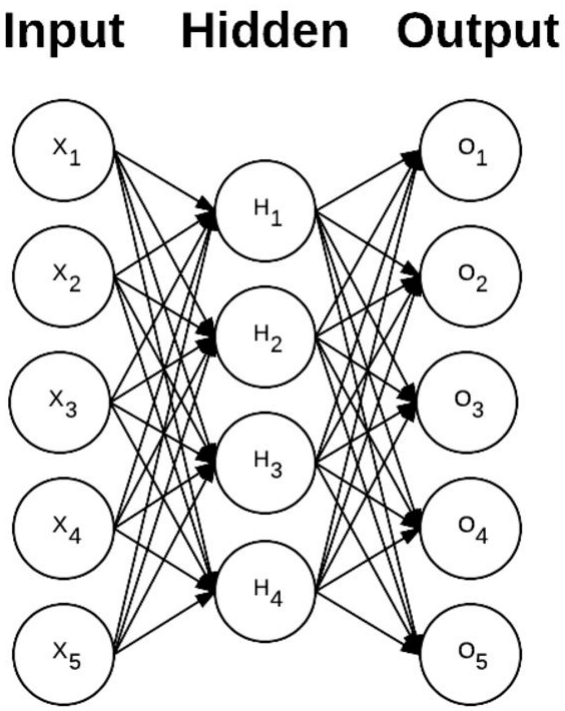

Figure 1: Example architecture of a neural network with an input layer, one hidden layer, and an output layer.

Submission Deadline: May 07, 2017 13:00

Handwritten Digit Classification using Neural Networks

Introduction to Neural Networks

Artificial neural networks are also referred to as "neural networks" or "artificial neural systems". It is common to abbreviate "Artificial Neural Networks" and refer to them as "ANN" or simply "NN".
For a system to be considered an NN, it must contain a labeled directed graph structure where each node in the graph performs some simple computation. From graph theory, we know that a directed graph consists of a set of nodes (i.e., vertices) and a set of connections (i.e., edges) that link together pairs of nodes. Below, we can see an example of such an NN graph, see figure 2:

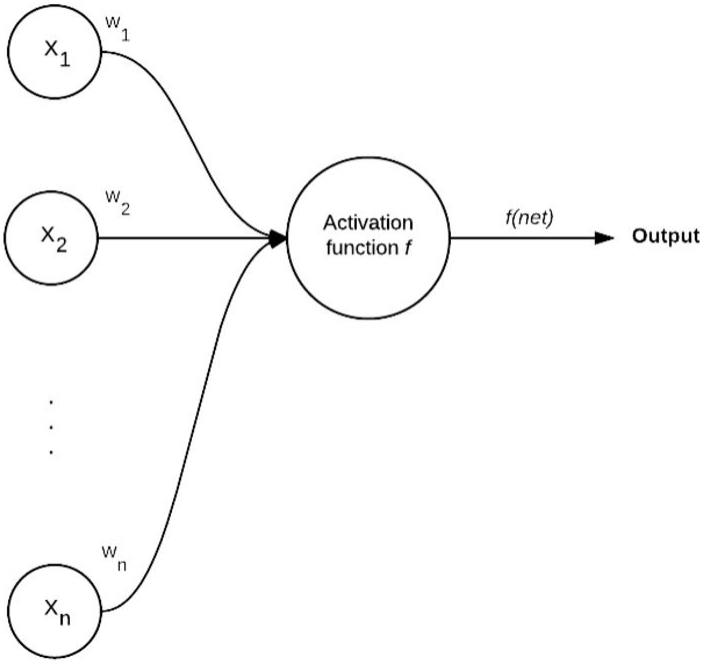

Figure 2: A simple Neural Network architecture. Starting from left, inputs are presented in the network, each connection carries a signal to the output node, where a final function "f " is computed to determine if the neuron "fires" or not.

Each node of the network performs a simple computation. Each connection then carries a "signal" (i.e., the output of the computation) from one node to another, labeled by a "weight" indicating the extent to which the signal is amplified or diminished.
Some connections have large, positive weights that amplify the signal, indicating that the signal is very important in making a classification. Others have negative weights, diminishing the strength of the signal, thus specifying that the output of the node is less important in the final classification.
We can call a system an Artificial Neural Network if it consists of a graph structure (like the one detailed in Figure 2 above) with connection weights that are modifiable using a learning algorithm.

Deep Belief Networks & Digit Classification

The Deep Belief Networks (DBN) is a set of Restricted Boltzmann Machines (RBMs) stacked on top of each other, allowing to learn more complex features in the higher-level layers of the network. Unlike traditional feedforward networks, the connections between visible and hidden layers of the RBM are undirected, implying that "information" can travel in both the visible-to-hidden and hidden-to-visible directions.
In this part of assignment you will use DBN to train a classifier on the MNIST handwritten digit dataset and evaluate the accuracy of the network. Later in the assignment you will classify handwritten digits using the same classifier.
After completing this part of the assignment you should be able to:

- Train and evaluate a DBN on the MNIST dataset.
- Classify handwritten digits using the classifier which is trained during the first step.

Required Output: You are required to include the following in your final assignment report:

- Error plots and classification reports (generated during the exercises) for all configurations of the neural network.
- Discuss on the basis of error plots and classification reports that how different configurations of network affect the accuracy of the classifier.
- Submit videos (scenario 1 and 2) where you show classified digits bounded inside the boxes as shown in figure 1 of assignment#02 part#01.
- Show the gaze location in the scene videos. Based on the gaze location the bounding box of the respective digits must change it's color and show the digit on top of the bounding box.

Training a DBN on MNIST

From assignment#02, you should be familiar with the MNIST handwritten digit dataset. This dataset consists of `70,000 (28 x 28)` gray-scale images representing the digits 0-9. Each digit appears as white on a black background. You will use a small subset of 5000 digits out of the dataset. The goal is to train a DBN to automatically recognize these handwritten digits directly from the pixel intensity values. The file named `mnist_dbn.py` is provided in the resources. **Task:** Open the file and read the code. To construct the DBN, you will be using the `nolearn` Python package. **Task:** Open command prompt and use `pip install nolearn` command to install `nolearn` package. Load the original MNIST dataset and split it into the training and testing sets, respectively. Normalize pixel intensities of the image to the range [0, 1], a requirement when training DBNs. **Lines 25-40** in `mmist_dbn.py` implements this functionality.
Now define the architecture of the DBN. See **Lines 45-50** for the network definition. The first argument to the DBN class is a list indicating the architecture of the network. The input layer has the same number of nodes as the dimensionality of the feature vector (in this

case, `28 x 28 = 784`). There is a hidden layer with 300 nodes. Finally, the resulting output layer has `10` nodes one for each of the possible `0-9` class labels.

Rules for DBNs: First, you need to ensure that the number of nodes in each layer progressively declines. This can be verified since `784 >= 300 >= 10`. Second, that no hidden layer is less than `1/4th` of the input layer's neuron count. This can also be verified since $784 \times 0.25 = 196 < 300$. Third, supply the `learn_rates` and `learn_rate_decays` to the DBN. The `learn_rates` define the learning rate of the neural network. It's important to set this value properly to ensure that the network is able to learn something. Next `epochs=20` indicate that the DBN will see the full training data a total of twenty times. Increasing the number of epochs normally allows the network to learn more, but at the risk of overfitting to the training data. It's important to keep a balance between the number of epochs and the generalizability of the network. There is another function named `plot_error()` in `mnist_dbn.py` which shows the decline in the error as the the training of the model progresses.
Finally, the call to `fit()` method trains the DBN. Once the DBN has been trained, it can be evaluated on the testing set. Lines **65-77** of the code in `mnist_dbn.py` randomly selects MNIST digits from the testing set, classify them, and display them to the screen. Copy-/Create the script file `mnist_dbn.py` to the base directory of the existing project structure from Assignment#02 part#01. **Task:** Run the script to see the results.

Exercise 2.2.1:

1. While keeping rest of the parameters constant, modify the number of nodes in hidden layer to 100, 200, and 300 respectively in DBN definition. Train and evaluate the resulting classifiers. Discuss the error plot and the results of the classification report in your assignment report. (Estimated time: 10-15 minutes)

2. Modify the number of epochs to 10, 30, and 50 respectively in DBN definition. Train and evaluate the resulting classifiers. Discuss the error plot and the results of the classification report in your assignment report. (Estimated time: 10-15 minutes)

3. Modify the learning rate to 0.1, 0.01, and 1.0 respectively in DBN definition. Train and evaluate the resulting classifiers. Discuss the error plot and the results of the classification report in your assignment report. (Estimated time: 10-15 minutes)

4. Modify train-test splits to 75-25, 66-33, and 50-50 proportions. Train and evaluate the resulting classifiers. Discuss the error plot and the results of the classification report in your assignment report. (Estimated time: 10-15 minutes)

5. Modify `mnist_dbn.py` so that you can get the trained classifier in `classify.py`. (Estimated time: 30 minutes)

6. Now when you have trained classifier using DBN in `classify.py`, test the classifier for the handwritten digits segmented from the frames of provided scene videos. You will have to modify the relevant part of `mnist_dbn.py` where the trained classifier is tested for the test set of MNIST dataset. (Estimated time: 45 minutes)

7. *Optional* Collect all the segmented digits from the scene video and use them as test dataset. In this case use whole MNIST dataset as training dataset. Discuss the error plot and the results of the classification report in your assignment report. (Estimated time: 30 minutes)

Gaze Point Estimation

So far, you have segmented and classified the digits in the scene video using Deep Belief Networks. You might have noticed that these videos (eye and scene videos) are recorded using Head Mounted Eye Tracker (HMET). The subject was looking (Gazing) at different digits while videos was being recorded. Next you will estimate subject's gaze position on different digits in the scene video. You will be using 1) the eye-video for tracking and localization of `pupil center` which will be used as an eye-feature during the gaze estimation stage, 2) the calibration matrix for mapping pupil center to the scene image and 3) the scene video where you already have the classified digits. The function `DetectPupil()` in `detect_pupil.py` implements this functionality and returns pupil center coordinates `(PupilX, PupilY)`. In the next stage, you will use estimated pupil center and given calibration matrix to estimate gaze position the scene video. You will use `pickle.load(fileStream)` function to load the calibration matrix from file. You need to convert the pupil center to homogeneous coordinates before multiplying it to the calibration matrix. The returned gaze-point in the scene video

is in normalized form. You will have to multiply it with image resolution to get the image coordinates in the scene video frames. The function `calculate_gaze_point()` implements this functionality in `classify.py` and returns the estimated gaze point.

Exercise 2.2.2:

1. Use your previously implemented `DetectPupil()` function in `detect_pupil.py` during assignment#02 part#01.(Estimated time: 10-15 minutes)

2. Once you have gaze location in the scene image, display only those digits, on top of the ROI, where user is looking at. Change the color of the bounding box as well. (Estimated time: 10 minutes)

3. Record the final video for your report. (Estimated time: 10 minutes)