EEE 443/543: Neural Networks

Assignment 3

(Due 10 December 2019, 17:00PM)

Instructions:

1. Prepare a report (including your answers/plots) to be uploaded on Moodle.
2. The report should be typeset (no handwriting allowed except for lengthy derivations, which may be scanned and embedded into the report).
3. Show all steps of your work clearly.
4. Unclear presentation of results will be penalized heavily.
5. No partial credits for unjustified answers.
6. **Use of any toolbox or library for neural networks is prohibited**.
7. Return all Matlab/Python code that you wrote in a single `.m/.py` file.
8. Code should be commented, code for different HW questions should be clearly separated.
9. The code file should NOT return an error during runtime.
10. If the code returns an error at any point, the remaining part of your code will not be evaluated (i.e., 0 points).

Question	Points	Your Score
Q1	50	
Q2	50	
TOTAL	100	

Question 1. [50 points]

In this question you will implement an autoencoder neural network with a single hidden layer for unsupervised feature extraction from natural images. The following cost function will be minimized:

$$J_{ae} = \frac{1}{2N}\sum_{i=1}^{N}\|d(m) - o(m)\|^2 + \frac{\lambda}{2}\left[\sum_{b=1}^{L_{hid}}\sum_{a=1}^{L_{in}}\left(W_{a,b}^{(1)}\right)^2 + \sum_{c=1}^{L_{out}}\sum_{b=1}^{L_{hid}}\left(W_{b,c}^{(2)}\right)^2\right] + \beta\sum_{b=1}^{L_{hid}} KL(\rho|\hat{\rho}_b) \tag{1}$$

The first term is the average squared-error between the desired response and the network output across training samples. Note that the desired output is the same as the input. The second term enforces Tykhonov regularization on the connection weights with parameter λ. The last term enforces that the hidden unit activations are sparse with parameter β for controlling the relative weighting of this term. The level of sparsity is tuned via ρ in the KL term (Kullback-Leibler divergence) between a Bernouilli variable with mean ρ and another with mean $\hat{\rho}_b$. $\hat{\rho}_b$ is the average activation of hidden unit b across training samples.

a) The file `assign3_data1.mat` contains a collection of 16×16 RGB patches extracted from various natural images in `data`. Preprocess the data by first converting the images to graysale using a luminosity model: $Y = 0.2126 * R + 0.7152 * G + 0.0722 * B$. To normalize the data, first remove the mean pixel intensity of each image from itself, and then clip the data range at ±3 standard deviations (measured across all pixels in the data). To prevent saturation of the activation function, map the ±3 std. data range to [0.1 0.9]. Display 200 random sample patches in RGB format, and separately display the normalized versions of the same patches. Comment on your results.

b) Prior to training, initialize the weights and the bias terms as uniform random numbers from the interval $[-w_o, w_o]$, where $w_o = sqrt(\frac{6}{Lpre+L_{post}})$ and $L_{pre,post}$ are the number of neurons on either side of the connection weights. Write a cost function for the network $[J, J_{grad}] = aeCost(W_e, data, params)$ that calculates the cost and its partial derivatives. $W_e = [W_1\ W_2\ b_1\ b_2]$, a vector containing the weights for the first and second layers followed by the bias terms; *data* is of size $L_{in} \times N$; *params* is a structure with the following fields *Lin* (L_{in}), *Lhid* (L_{hid}), *lambda* (λ), *beta* (β), *rho* (ρ). Use J and J_{grad} as inputs to a gradient-descent solver to minimize the cost. Assuming $L_{hid} = 64$, $\lambda = 5 \times 10^{-4}$, experiment with β, ρ to find parameters that work well. Note that performance here is defined based on the 'quality' of the features extracted by the network.

c) The solver will return the trained network parameters. Display the first layer of connection weights as a separate image for each neuron in the hidden layer. What do the hidden-layer features look like? Are these features representative of natural images?

d) Retrain the network for 3 different values (low, medium, high) of $L_{hid} \in [10\ 100]$, of $\lambda \in [0\ 10^{-3}]$, while keeping β, ρ fixed. Display the hidden-layer features as separate images. Comparatively discuss the results you obtained for different combinations of training parameters.

Question 2. [50 points]

The goal of this question is to introduce you CNN models. You will be experimenting with two demos, one on a CNN model in Python, and a second on a CNN model in one of two popular frameworks (PyTorch or TensorFlow). Download `demo_cnn.zip` from Moodle and unzip it. The demos are given as Jupyter Notebooks along with relevant code and data. The easiest way to install Jupyter with all Python and related dependencies is to install Anaconda. After that you should be able to run through demos in your browser easily. The point of these demos is that they take you through the training algorithms step by step, and you need to inspect the relevant snippets of code for each step to learn about implementation details.

a) The notebook `Convolutional_Networks.ipynb` contains demonstrations on a CNN model. You need to run the demo till the end without any errors. You are supposed to convert the outputs of the completed demo to a PDF file, and attach it to the project report. You should also comment on your results.

b) The notebooks `PyTorch.ipynb` and `TensorFlow.ipynb` contain demonstrations on a CNN model in deep learning frameworks. Please pick a single framework to work with (PyTorch has a Python like feeling but might have limited visualization options, and TensorFlow might have a steeper learning curve but is better equipped with supporting tools). You need to run the selected demo till the end without any errors. You are supposed to convert the outputs of the completed demo to a PDF file, and attach it to the project report. You should also comment on your results.