ECBM E4040 Neural Networks and Deep Learning, 2016 Fall
HOMEWORK #2

INSTRUCTIONS: This homework contains two parts - theoretical and programming. Submission for this homework will be via bitbucket repositories created for each student and should contain the following

1. A file called hw2 writeup.pdf that contains solutions to the theoretical questions
2. Put all figures and discussions, and document all parameters you used in the programming question in the IPython notebook file, hw2b.ipynb, which is already included in the homework 2 repository. All the discussions should also be included in the notebook file.
3. **Please do not push the dataset to the repository.** Just commit the source files.

Please be advised that training time of all parts within programming problem may add up to more than a day, so start early!

Theoretical

You will need the definition of the PDF of a matrix normal distribution to complete this part. The following provides the PDF of a simplified special case that will be used in this assignment.
Matrix Normal Distribution: A $n \times n$ square matrix valued random variable $\mathbf{A}$ is said to follow a matrix normal distribution ($\mathcal{MN}$) with parameters ($\mathbf{M}$, $\lambda^{-\frac{1}{2}}\mathbf{I}, \lambda^{-\frac{1}{2}}\mathbf{I}$) if it has the following probability density function

$$\mathbb{P}(\mathbf{A};(\mathbf{M}, \lambda^{-\frac{1}{2}}\mathbf{I}, \lambda^{-\frac{1}{2}}\mathbf{I})) = \frac{1}{(2\pi)^{\frac{n^2}{2}}} exp(-\frac{1}{2}\mathrm{Tr}[\lambda(\mathbf{A}\text{-}\mathbf{M})^T(\mathbf{A}\text{-}\mathbf{M})])$$

Given the observed data $(\mathbf{x}^1,\mathbf{y}^1), \ldots,(\mathbf{x}^m,\mathbf{y}^m)$, where $\mathbf{x}^i \in \mathbf{R}^n$ and $\mathbf{y}^i \in \mathbf{R}^n$, $\forall\ i \in [1,m]$, we are interested in finding an $\mathbf{A} \in \mathbf{R}^{n \times n}$ such that *in some sense* (to be defined below) $\mathbf{y}^i \approx \mathbf{A}\mathbf{x}^i$.

For simplicity, we use the following notation

$$\mathbf{Y} = [\mathbf{y}^1\ \ \mathbf{y}^2\ \ \ldots\ \ \mathbf{y}^m\ ] \qquad \mathbf{X} = [\mathbf{x}^1\ \ \mathbf{x}^2\ \ \ldots\ \ \mathbf{x}^m\ ]$$

and assume that $m > n$, and $\mathbf{X}\mathbf{X}^T$ is invertible.

(i) For the least squares loss function

$$L_{\mathrm{ls}} = \sum_{i=1}^{m} (\mathbf{y}^i\text{-}\mathbf{A}\mathbf{x}^i)^T(\mathbf{y}^i\text{-}\mathbf{A}\mathbf{x}^i)$$

Find $\mathbf{A}_{\mathrm{ls}} = \mathrm{argmin}_{\mathbf{A}}\ L_{\mathrm{ls}}$

ECBM E4040 Neural Networks and Deep Learning, 2016 Fall
HOMEWORK #2

(ii) For the least squares loss function with Frobenius norm regularization term

$$L_{\mathrm{r}} = \lambda \| \mathbf{A} \|_F^2 + \sum_{i=1}^{m} (\mathbf{y}^i - \mathbf{A}\mathbf{x}^i)^T (\mathbf{y}^i - \mathbf{A}\mathbf{x}^i)$$

Find $\mathbf{A}_{\mathrm{r}} = \mathrm{argmin}_{\mathbf{A}} L_{\mathrm{r}}$

Note: $\| \mathbf{A} \|_F^2 = \mathrm{Tr}(\mathbf{A}^T\mathbf{A})$

(iii) Assume that the errors $\varepsilon_i = \mathbf{y}^i - \mathbf{A}\mathbf{x}^i$ are normally distributed with mean $\mathbf{0}$ under the ideal $\mathbf{A}$, i.e., $\varepsilon_i \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$.
Find $\mathbf{A}_{\mathrm{ML}}$, the maximum likelihood solution of $\mathbf{A}$.

(iv) Under the same assumption of normal error distribution, consider a prior on $\mathbf{A}$ of the the form $\mathbf{A} \sim \mathcal{MN}(\mathbf{M}, \lambda^{-\frac{1}{2}}\mathbf{I}, \lambda^{-\frac{1}{2}}\mathbf{I})$.
Find $\mathbf{A}_{\mathrm{MAP}}$, the maximum a posteriori estimate of $\mathbf{A}$. What will be $\mathbf{A}_{\mathrm{MAP}}$ if we assume $\mathbf{M}$ to be a zero matrix?

(v) Comment on the relation between the expression derived in (i) and (iii), and of those derived in (ii) and (iv).

ECBM E4040 Neural Networks and Deep Learning, 2016 Fall
HOMEWORK #2

Programming:

For this part, you will experiment with different Multilayer Perceptron configurations, and empirically study various relationships among number of layers and number of parameters. You should start by going through the Deep Learning Tutorials Project. In particular, the source code provided in the Homework 2 repository is excerpted from logistic_sgd.py and mlp.py.

You are asked to partially reproduce the phenomena shown in Figure 6.9 and Figure 6.10 of the textbook. The original work for these two figures implemented an advanced framework of deep network [2], which is beyond the material covered in the course till now. Instead of reimplementing the original work, you should simply use the multilayer perceptron described in the tutorial.

You will be using the street view house numbers (SVHN) dataset [1]. The dataset is similar in flavor to MNIST, but contains substantially more labeled data, and comes from a significantly harder, real world problem (recognizing digits and numbers in natural scene images). You will use the **Format 2** of the SVHN dataset. Each sample in this format is a MNIST-like 32-by-32 RGB image centered around a single character. Many of the images do contain some distractors on the sides, which of course makes the problem interesting.

The task is to implement an MLP to classify the images of the SVHN dataset. The input to the MLP is a color image, and the output is a digit between 0 and 9.

A python routine called **`load_data`** is provided to you for downloading and preprocessing the dataset. You should use it, unless you have absolute reason not to. The first time you call **`load_data`**, it will take some time to download the dataset (about 180 MB). Please be careful NOT TO commit the dataset files into the repository.

Note that all the results, figures, and parameters should be placed inside the IPython notebook file hw2b.ipynb. To remain jupyter notebook running even if you close the ssh access to your instance, you can use *tmux*, *nohup* or *screen*.

PROBLEM b:

1. First enable the construction of MLPs with multiple hidden layers. Implement your MLP in the skeleton **`myMLP`** class in **`hw2b.py`**.

2. Implement an MLP with 2 hidden layers. Compare the effect of two activation functions, *tanh* and *softmax*, on neurons in hidden layers with other parameters fixed. Note that the output neuron always uses *softmax*. Document your choice of parameters explicitly, and discuss your test accuracy results in both cases.

3. Experiment with the number of hidden layers. In particular, generate a plot similar to the one in Figure.1. Note that Figure.1 is similar in spirit to Figure 6.9 of the textbook. Each hidden layer should contain the same amount of neurons. You might want to start with a network with 3 hidden layers, and

experiment with parameters (e.g., activation function, learning rate, number of hidden neurons, early-stopping patience etc.). After finding a set of parameters, run your MLP 4 times with the number of hidden layers = {1,3,5,7} (the total number of layers thus ranges from 3 to 9). Document your choice of parameters explicitly, and discuss your test accuracy results. (Tips: Don't worry about reproducing the exact figure, exploration and rationale matter most. This rule is applied to whole programming homework.)

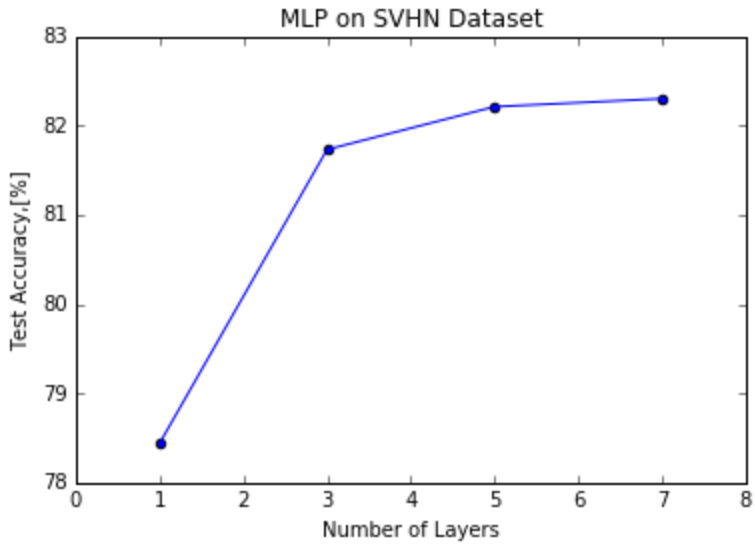

Figure 1

4. Experiment with the number of hidden layers, but fix the total number of neurons in all hidden layers. In particular, generate a plot similar to Figure.2. In Figure.2, the total amount of hidden neurons is fixed at 840. You may chose another number. Each hidden layer should contain the same amount of neurons (that is, $floor\ (\frac{total\ number}{number\ of\ layers})$ neurons in each layer). Run your MLP 6 times with number of hidden layers varying from 1 to 6. Document your choice of parameters and their number explicitly, and discuss your test accuracy results.

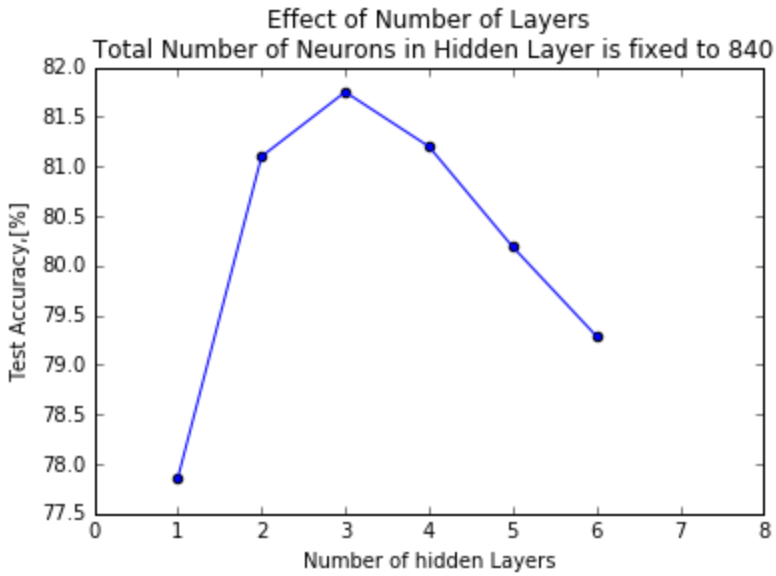

Figure 2

ECBM E4040 Neural Networks and Deep Learning, 2016 Fall
HOMEWORK #2

5. For a fixed number of hidden layers experiment with the number of neurons in hidden layers. In particular, generate a plot similar to the one in Figure.3. Note that Figure.3 is similar in spirit to Figure 6.10 in the textbook. Run your MLP with 1 hidden layer 5 times with 5 different numbers of hidden neurons. Document your choice of parameters and their number explicitly, and discuss your test accuracy results.

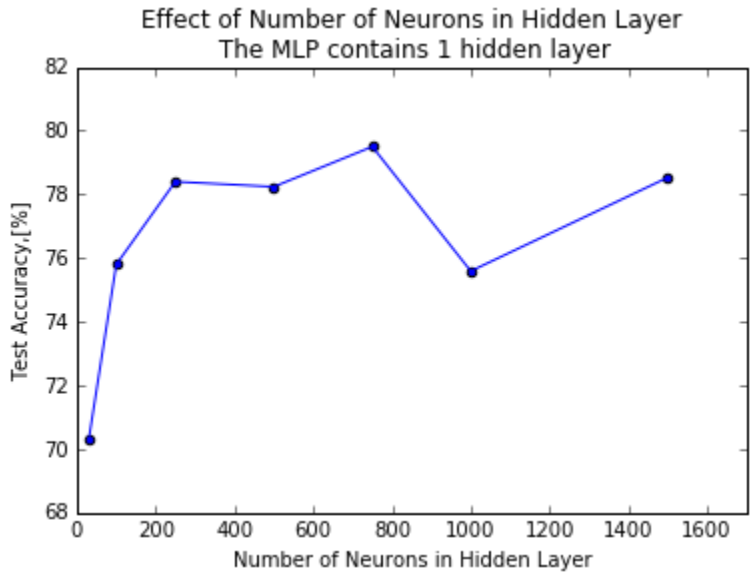

Figure 3

PROBLEM c:

For this part, you will be using (a) DROPOUT Regularization technique for an MLP with the parameters given below. (b) You are also required to implement a method called Momentum which helps accelerate SGD in the relevant direction.

Number of Hidden Layers = 2
Number of Neurons in each layer = 500
L1 reg = 0
L2 reg = 0.0001
Batch Size = 20
Learning Rate = 0.01
Number of Epochs = 500
p = 0.7
momentum = 0.5

1. The `DropoutHiddenLayer` class is already defined for you in hw2c.py. Implement an MLP with 2 Dropout Hidden Layers with momentum enabled in the `test_mlp_dropout` method in hw2c.py. Run the method with the parameters as specified above and discuss the test accuracy results.

2. Implement an MLP with 2 Hidden Layers **without dropout** for the same set of parameters as specified above. Compare the test accuracy results with those obtained in part 1 and discuss.

ECBM E4040 Neural Networks and Deep Learning, 2016 Fall
HOMEWORK #2

NEED HELP:
If you have any questions you are advised to use Piazza forum which is accessible through courseworks.
GOOD LUCK!

References

[1] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, Andrew Y. Ng, "Reading Digits in Natural Images with Unsupervised Feature Learning," *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*.
[2] Ian Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, Vinay Shet, "Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks," *ICLR 2014*.