Project 2
Multilayer Perceptron (MLP) – MNIST Dataset

November 21, 2017

Due: December 5, 2017, 11:59 PM

Instructions

This project will be graded with a letter grade with respect to presentation (25%), methods (35%) and results (40%).

The project requires a report explaining the experimental procedures you followed and you must include data to support your conclusions (for example, figure and tables). Please use the format of an IEEE Transactions paper (limited to 7 double column pages). You can download the format from the IEEE website (https://www.ieee.org/conferences_events/conferences/publishing/templates.html). This means you have to write a brief intro to the theory, explain well the methods and present carefully the results (based on the questions below) and conclusions. Remember that any scientific paper should, by definition, contain sufficient information such others can replicate your results. A scientific paper must also contain ORIGINAL material only. If you happen to use equations from other source you have to reference what you cut and paste (this is not allowed in a normal publication, but here it is OK provide you reference). Of course, **the results should be done by the student alone**.

Your programs must be written in either MATLAB or Python. The relevant code to the project should be included as an attachment to your report.

Please add the following line, along with your signature, at the end of your report: **I confirm that this assignment is my own work, is not copied from any other person's work (published or unpublished), and has not been previously submitted for assessment either at University of Florida or elsewhere.**

Submit your report (including code and statement) as a PDF to the E-Learning at UF (http://elearning.ufl.edu/).

If you have any questions address them to:

- Catia Silva (TA) – catiaspsilva@ufl.edu
- Sheng Zou (TA) – shengzou@ufl.edu

In this project you should train a Multilayer Perceptron (MLP) on the MNIST dataset, which is a database of the 10 handwritten digits. The MNIST dataset (28×28 images) contains 60,000 images for training and 10,000 images for testing. The last 10,000 images in the training dataset should be used as validation dataset for cross-validation, which include early stopping and hyper parameter choices.

NOTE: If your computer is slow, you can use less exemplars for training, but please specify how large is the training set because this will affect the quality of the results. The size of the cross-validation and test sets should remain at 10,000.

You should address (at least) the following topics:

1. Download the MNIST dataset and code from the course website (courtesy of http://ufldl.stanford.edu/).

2. For python users, load images and labels using *LoadMNIST.py*. For MATLAB users, load images and labels using *loadMNISTImages* and *loadMNISTlabels* scripts. For example:

```
Training_images = loadMNISTImages('train-images.idx3-ubyte');
Training_labels = loadMNISTLabels('train-labels.idx1-ubyte');
```

3. Train multilayer perceptrons (MLPs) for classification. We suggest using one and two hidden layers only, otherwise training will be too long. You have to experiment with the number of units in each layer. You can also experiment with dropout (i.e. with a certain probability, set to zero weights during training, which may improve the generalization of the network).

4. We suggest to downsample the image to decrease the number of inputs or use PCA to project to a lower dimensional subspace. You should experiment with the level of downsampling and also with the dimensionality of the subspace.

5. Use samples 50,000 to 60,000 from the training dataset as the validation dataset for early stopping.

6. Report results on the test set as a *confusion matrix* (i.e. show the true labels and the assigned classes on a 10×10 matrix (one for each digit). If the classifier is perfect you should have only the diagonal populated, so classification errors will appear in the off-diagonal entries). Discuss the results you obtain and state if they make sense or not.

7. Show learning curves comparing training and validation cost function. Do the same for the accuracy.

8. Compare performance with Stochastic Gradient Descent (SGD) and SGD-momentum.

Hints: Use ReLU activation function, $f(net) = \begin{cases} 0 & net \leq 0 \\ net & net > 0 \end{cases}$, for better results and faster convergence.