Project Assignment 2

CZ4042: Neural Networks

Deadline: 4th November, 2016

- ✓ The project is to be done in a group of not more than two.
- ✓ Complete both parts 1 and 2. Data files and example codes for both parts can be found under Project Assignment 2.
- ✓ You will need Matlab 2016 and machines with CUDA-enabled NVIDIA GPU with compute capability 3.0 or higher for the project. You have free access to computers with necessary tools at the Software Projects Lab at N4-B1b-10 (Mon – Thurs: 8.30 A.M. to 5.45 P.M., Fri: 8.30 A.M. to 5.15 P.M.).
- ✓ Prepare one project report for both parts including sections on
  - o Introduction
  - o Methods describing the architectures and learning algorithms of the neural networks implemented.
  - o Results including testing on different parameters used for training, plots of convergence of learning, training and test error rates, supporting tables and graphs.
  - o Discussion on the results and challenges.

  The cover page of the report should carry the names of all the members of the group.

- ✓ Each member of the group should submit the project report in .pdf format with file name your_name_P2_report.pdf and all the source codes in a zip file named: your_name_P2_codes.zip
- ✓ Submit both your report and source code on line via NTU Learn before the deadline.
- ✓ Assessment is based on correctness(30%), execution(30%), report(35%), and bonus(5%, this is for the extra effort).
- ✓ TA Mr. Sukrit Gupta (SUKRIT001@e.ntu.edu.sg) is in charge of the course projects. Please see him at the Biomedical Informatics Graduate Lab (NS4-04-33) during his office hours: Friday 3:30 P.M. – 5:30 P.M., in case you face issues.

The project uses hand-written digit images provided by the MNIST database. 5,000 training images and 1,000 test images have been selected for the project. The data and sample codes of the project are given in Project2.zip file.

You are advised to select a subset of train and test images for debugging your routines, but the final results should be based on experiments on complete dataset provided.

Before training and testing, the train and test data should be shuffled.

Part 1: Deep convolutional neural network

This part of the assignment is to give you some exposure to the use of convolutional neural networks for object recognition in images.

You are given a subroutine 'cnnPreprocess.m' which can be used to create imageDatastore objects for both training and test data in Matlab. Make sure that the path to the image folder in 'cnnPreprocess.m' routine is correct.

You can, then, load data from 'dataTrainstore.mat' (imageDatastore object for training data) and 'dataTest.mat' (imageDatastore object for testing data).

a) Using the images in the training set, build a convolution neural network with one hidden layer to recognize the digits in images as follows:
   - Input layer I_0 with dimensions 28x28
   - Convolution layer C_1 consists of 20 filters of 9x9 receptive fields
   - Mean pooling layer S_1 has a pooling dimension of 2x2
   - A fully connected layer connected to a softmax regression layer.
   - An output classification layer.

   Experiment with the following parameters:
   - Different learning rates.
   - Momentum and decay term in gradient descent learning.
   - Different batch sizes in the stochastic gradient descent.

   Decide on the best parameters and report the accuracies on train and test images. You can set the L_2 regularisation coefficient to 0.001 and take the maximum number of epochs as 25. A sample code for your reference is given in file 'cnnSample.m'.

b) Build a deep multi-layer feedforward neural network with the following configuration to recognize the handwritten digits:
   - Input layer I0 with dimensions 28x28.
   - First hidden layer consists of a convolution layer C_1 consisting of 30 filters of 5x5 receptive fields and a mean pooling layer S_1 with a pooling dimension 2x2.
   - Second hidden layer consists of convolution layer C_2 consists of 50 filters of 5x5 receptive fields and a mean pooling layer S_2 with a pooling dimension 2x2.
   - A fully connected layer connected to a softmax regression layer.
   - An output classification layer.

   Use the learning parameters obtained in Part 1(a).

   Experiment with the number of filters in the layers (in the range of 20-70) and determine the best configuration. Report the accuracy of the final model on test images. Maximum number of epochs should be 20.

Part 2: Autoencoders

This part of assignment aims to provide you with some exposure to the use of neural networks as content addressable memories.

You are given a subroutine ‘autoencoderPreprocess.m’ which can be used to create matlab objects for both training and test data and labels in Matlab . You can change the path of your image folder in the ‘cnnPreprocess.m’ subroutine.

You can, then, load data from ‘dataTrain.mat’ (~5,000 training samples), ‘labelsTrain.mat’ (labels for training samples), ‘dataTest.mat’ (~1,000 test samples) and ‘labelsTest.mat’ (labels for test samples).

a) Design an autoencoder having one hidden layer made of 100 neurons to store handwritten digits in the training set.

   Experiment with the following parameters:
   - Number of epochs (max: 1000).
   - The sparsity proportion and the sparsity regularization coefficient.

   Reconstruct the inputs using the trained encoder and determine the best parameters for the model by finding the lowest mean squared error on the test patterns.

   Compare the results for the minimum error with different transfer functions (using the same parameters calculated before) both by visual inspection and by using mean squared error and comment on any peculiarities.

Remember, you don't have to use test/train labels for this task, simply use the train/test patterns. A sample code for your reference is given in file 'autoencoderSample.m'.

b) Train a second autoencoder to predict outputs of the encoder in Part 2(a) by using 50 hidden neurons. You can extract the second set of features by passing the outputs of the first encoder through the second encoder.

   Compare the reconstruction errors with those in Part 2(a).

   Show sample of features learned by the first hidden layer and the second hidden layer.

c) Train a softmax layer after the second autoencoder to recognize the images. To train the softmax layer, use the output of the second auto encoder and corresponding labels from the test images.

   Experiment with the number of neurons in the hidden layers and find the configuration that gives the best classification results (in terms of percentage) as well as the best reconstruction error.

   You can vary the Sparsity Regularization Coefficient from 1 to 10, Sparsity Proportion from 0.1 to 0.3, use logistic sigmoidal transfer function for both encoders, you can scale the data for both encoders and do not consider more than 200 epochs for any of the layers.