CSCI 5521: Introduction to Machine Learning (Fall 2018)[1]

Homework 4

1. (**30 points**) Consider the Multilayer Perceptron (MLP) for binary classification described in section 11.7.2 in the textbook, we have the following error function:

$$E(W, v|X) = -\sum_t r^t \log y^t + (1 - r^t)\log(1 - y^t),$$

where $y^t = \text{sigmoid}(\sum_{h=1}^{H} v_h z_h^t + v_0)$ and $z_h^t = \text{sigmoid}(w_h^T x^t)$.

Now, let's look at a regularized version of MLP, in which the activation function of each hidden unit becomes the leaky rectified linear unit $\text{LReLU}(x)$, instead of the sigmoid function, where:

$$\text{LReLU}(x) = \begin{cases} 0.01x, & \text{for } x < 0 \\ x, & \text{otherwise} \end{cases}$$

and the output unit becomes a hyperbolic tangent unit $\tanh(x)$. The error function becomes the following:

$$E(W, v|X) = -\sum_t (r^t \log y^t + (1 - r^t)\log(1 - y^t)) + \sum_h ||w_h||_2^2,$$

where $y^t = \tanh(\sum_{h=1}^{H} v_h z_h^t + v_0)$ and $z_h^t = \text{LReLU}(w_h^T x^t)$. Derive the update equations of the regularized MLP with one hidden layer.
Hint: Read Section 11.7.2 to see how Equations 11.23 and 11.24 are derived from Equation 11.22
Hint 2: $\tanh'(x) = 1 - \tanh^2(x)$.
Hint 3: $\text{LReLU}'(x) = \begin{cases} 0.01, & \text{for } x < 0 \\ 1, & \text{otherwise} \end{cases}$.

[1]Instructor: Rui Kuang (kuang@cs.umn.edu). TA: Zhuliu Li (lixx3617@umn.edu).

2. (**40 points**) Implement a Multilayer Perceptron (MLP) **with stochastic gradient descent** to classify the optical-digit data. Train your MLPs on the "optdigits_train.txt" data, tune the number of hidden units using the "optdigits_valid.txt" data, and test the prediction performance using the "optdigits_test.txt" data. (Read the submission instruction carefully to prepare your submission files.)

   (a) Implement a MLP with 1 hidden layer **using the ReLU activation function**:

$$\mathrm{ReLU}(x) = \begin{cases} 0, & \text{for } x < 0 \\ x, & \text{otherwise} \end{cases}$$

   Use the MLP for classifying the 10 digits. Read the algorithm in Figure 11.11 and section 11.7.3 in the textbook. When using the ReLU activation function, Equation 11.29 becomes:

$$\Delta w_{hj} = \begin{cases} 0 & \text{for } w_h^T x < 0 \\ \eta \sum_t \left[\sum_i (r_i^t - y_i^t) v_{ih} \right] x_j^t & \text{otherwise} \end{cases}$$

   Try MLPs with $\{3, 6, 9, 12, 15, 18\}$ hidden units. Report and plot the training and validation error rates by the number of hidden units. How many hidden units should you use? Report the error rate on the test set using this number of hidden units.
   Hint: When choosing the best stepsize η (between 0 and 1 such as 10^{-5}), you might need to start with some value and, after a certain number of iterations, decrease your η to improve the convergence. Alternatively, you can implement Momentum or Adaptive Learning Rate (section 11.8.1 in the textbook).

   (b) Train your MLP with the best number of hidden units obtained. Combine the training set and the validation set as one (training+validation) dataset to run the trained MLP from problem 2(a) with the data. Apply PCA to the values obtained from the hidden units (you can use the Matlab *pca()* function). Using the projection to the first 2 principal components, make a plot of the training+validation dataset (similar to Figure 11.18 in the textbook). Use different colors for different digits and label each sample with its corresponding digits (the same as you did in HW3). Repeat the same projecting the datasets to the first 3 principal components and do the

visualization using 3-D plot. (Hint: you can use the MATLAB function *plot3()* to visualize the 3-D data). Compare the 2-D and 3-D plots and explain the results in the report.
Note: Change the x-axis and y-axis to log scale in order to better visualize the datapoints.

3. (**30 points**) MATLAB provides the Deep Learning Toolbox for designing and implementing deep neural networks. In this homework question you will learn how to create simple convolutional neural networks (CNNs) for optdigits classificaion.

   (a) Read the MATLAB documentation[2] to get familiar with how to

      i. Load and explore image data.
      ii. Define the network architecture.
      iii. Specify training/validation options.
      iv. Train the network.
      v. Predict the labels of testing data and calculate the classification accuracy.

   Read another MATLAB documentation[3] to learn how to define your own customized layer.

   (b) Run the `dataPreparation.m` script to convert the three `optdigits.txt` files into the required input formats. Modify the `examplePreluLayer.m` (as described in the **Completed Layer** section of 3) file to define a class called myLReLULayer, which creates the leaky ReLU layer as defined in Question 1.

   (c) Modify the **Define Network Architecture** section in the `main.m` file to test the following two CNN structures.

      i. Input layer $\rightarrow$ 2D convolution layer (1 filter with size 4) $\rightarrow$ Batch normalization layer $\rightarrow$ LReLU layer (use your own customized myLReLULayer class) $\rightarrow$ Fully connected layer $\rightarrow$ Softmax layer $\rightarrow$ Classification layer

[2] `https://www.mathworks.com/help/deeplearning/examples/create-simple-deep-learning-network-for-cla`
`html`

[3] `https://www.mathworks.com/help/releases/R2018a/nnet/ug/`
`define-custom-deep-learning-layer.html`

ii. Input layer $\rightarrow$ 2D convolution layer (20 filter with size 3) $\rightarrow$ Batch normalization layer $\rightarrow$ LReLU layer (use your own customized myLReLULayer class) $\rightarrow$ Pooling layer (use max pooling with poolsize 3 and stride size 2) $\rightarrow$ 2D convolution layer (32 filter with size 3) $\rightarrow$ Batch normalization layer $\rightarrow$ LReLU layer (use your own customized myLReLULayer class) $\rightarrow$ Fully connected layer $\rightarrow$ Softmax layer $\rightarrow$ Classification layer

For both network structures, take a screen shot of the Training Process images generated by MATLAB, and report the accuracies on the testing data.

Instructions

- Solutions to all questions must be presented in a report which includes result explanations, and all images and plots.

- All programming questions must be written in Matlab, no other programming languages will be accepted. The code must be able to be executed from the Matlab command window on the cselabs machines. Each function must take the inputs in the order specified and print/display the required output to the Matlab command window. For each part, you can submit additional files/functions (as needed) which will be used by the main functions specified below. Put comments in your code so that one can follow the key parts and steps. **Please follow the rules strictly. If we cannot run your code, you will receive no credit.**

- **Question 2:**

  – Train a MLP: mlptrain(*train_data.txt*: path to training data file, *val_data.txt*: path to validation data, m: number of hidden units, k: number of output units). The function must return in variables the outputs (z: a $n \times m$ matrix of hidden unit values, w: a $m \times (d+1)$ matrix of input unit weights, and v: a $k \times (m+1)$ matrix of hidden unit weights). The function must also print the training and validation error rates for the given function parameters.

  – Test a MLP: mlptest(*test_data.txt*: path to test data file, w: a $m \times (d+1)$ matrix of input unit weights, v: a $k \times (m+1)$ matrix of hidden unit weights). The function must return in variables the outputs (z: a $n \times m$

matrix of hidden unit values), where n is the number of training samples. The function must also print the test set error rate for the given function parameters.

  - mlptrain will implement an MLP with d inputs and one input bias unit, m hidden units and one hidden bias unit, and k outputs.
  - `problem2a.m` and `problem2b.m`: scripts to solve the problems 2 (a) and (b), respectively, calling the appropriate functions.

- You may find the following built-in Matlab functions: *repmat()* and *reshape()*.
- For the optdigits data, the first 64 columns are the data and the last column is the label.

Submission

- **Things to submit:**

  1. hw4_sol.pdf: A PDF document which contains the report with solutions to all questions.
  2. mlptrain.m: The Matlab code of the *mlptrain* function.
  3. mlptest.m: The Matlab code of the *mlptest* function.
  4. problem2a.m: Code to solve problem 2 (a).
  5. problem2b.m: Code to solve problem 2 (b).
  6. myLReLULayer.m: Your own customized leaky ReLU layer in problem 3(b).
  7. main.m: The modified script for the neural structure in problem 3(c)(ii).
  8. Any other files, except the data, which are necessary for your code.

- **Submit**: All material must be submitted electronically via Canvas.