CSP 304: Machine Learning Lab (Spring 2023)

Indian Institute of Information Technology (IIIT), Kota
Instructor: Ankit Sharma

Posted in week: May 2- 5, 2023
Due in week: May 8-12, 2023

Homework 4

Topic **Multilayer Perceptron (MLP)** This homework illustrates the use of non-linear MLPs for supervised learning.

Data Train, validation and test datasets given in the data.zip file attached.

Problems

We will 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. (Please read the **Instructions** below before you start programming)

A1 **Method Implementation (50 points):** Implement a MLP with 1 hidden layer using the ReLU activation function:

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

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, the online version of Equation 11.29 becomes:

$$\triangle \mathbf{w}_{hj} = \begin{cases} 0, & \text{for } w_h^T x < 0 \\ \eta\left[\sum_i (r_i - y_i) v_{ih}\right] x_j, & \text{otherwise} \end{cases}$$

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).

A2 **Test and Plot (50 points):** 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 part A1 with the data. Apply PCA to the values obtained from the hidden units (you can use *PCA()* function, which is included in the module decomposition in *scikit-learn* package). 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 previous HWs). Repeat the same, projecting the datasets to the first 3 principal components and do the visualization using the 3-D plot. (**Hint**: you can use the function *scatter()* 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.

Instructions

- Solutions to all questions must be presented in a report which includes results, explanations, all images and plots.
- *numpy, scipy, skimage and matplotlib* can be relied on to implement the algorithm. You may need the following functions in your homework: *hstack, vstack, tile, argmax, argmin* of *numpy*. Each function must take the inputs in the order specified and print/display the required output to either terminal or PyCharm console or notebook. 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.
- *Function definition:* Train a MLP: **MLPTrain**($train\_data.txt$: path to training data, $val\_data.txt$: path to validation data, K: number of output units, H: number of hidden units). The function must return in variables the outputs (Z: a $N \times H$ matrix of hidden unit values, W : a $(D + 1) \times H$ matrix of input unit weights, and V : a $(H + 1) \times K$ matrix of hidden unit weights). The function must also print the training and validation error rates for the given function parameters.
- *Function definition:* Test a MLP: **MLPtest**($test\_data.txt$: path to test data file, W: a $(D + 1) \times H$ matrix of input unit weights, V: a $(H + 1) \times K$ matrix of hidden unit weights). The function must return in variables the outputs (Z: a $N \times H$ 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, H hidden units and one hidden bias unit, and K outputs.
- For the optdigits data, the first 64 columns are the data and the last column is the label.

Deliverables

Report — A brief report in PDF format describing the various experimental tasks mentioned above. Description should include the details of your experiments (process & setup), results and discussions/comments on the observations.

Code — Properly commented code file(s) or notebook(s) or any other setup and/or read-me files. All programming questions must be written in MATLAB/Python, no other programming languages will be accepted. For MATLAB you have to provide your own implementation using basic functionalities and no ready made libraries.

If there are several of them, please bind all the code files (NOT the report file) into a single ZIP file and upload as a response to Google classroom.

Notes — Although not mandatory, the report is encouraged to be written in LaTeX preferably via www.overleaf.com using the NIPS format available here: https://neurips.cc/Conferences/2021/PaperInformation/StyleFiles. Also, an example overleaf NIPS template can be readily found here: https://www.overleaf.com/latex/templates/neurips-2021/bfjnthbqvhgs.
