CSCI 5521: Introduction to Machine Learning (Fall 2021)[1]

Homework 3

Due date: Nov 17, 2021 11:59pm

1. (**30 points**) Consider the Multilayer Perceptron (MLP) for binary classification described in section 11.7.2 in the textbook. Let's look at the regularized version of MLP when the activation function of each hidden unit becomes the reLU function $\text{reLU}(x) = \max(0, x)$, and the activation function of the output unit is the sigmoid function. In the regularized version, the error function becomes the following:

$$E(W, v|X) = -\sum_{t=1}^{N} \left[r^t \log y^t + (1 - r^t) \log(1 - y^t) \right] + \sum_{h=1}^{H} ||w_h||_2^2,$$

where $y^t = \text{sigmoid}(\sum_{h=1}^{H} v_h z_h^t + v_0)$ and $z_h^t = \text{reLU}(w_h^T x^t + w_{h0})$. Derive the update equations of the regularized MLP using the given activation functions.

Hint 1: Given $y = \text{sigmoid}(\alpha) = 1/(1 + e^{-\alpha})$, the derivative $\frac{\partial y}{\partial \alpha} = y(1 - y)$.
Hint 2: Given $\text{reLU}(f(x)) = \max(0, f(x))$, the derivative of $\text{reLU}(f(x))$ is given by $f'(x)$ if $f(x) > 0$, and 0 otherwise.

2. (**30 points**) Build a (Multilayer) Perceptron to recognize a certain area of the plane. That is, the Perceptron should output a "1" if the input vector lies in the shaded region.

   (a) Determine the vector of coefficients $w = [w_0, w_1, w_2]^T$ for a single layer perceptron of the form in Figure 1 to recognize the area in Figure 2 and again for Figure 3 shaded blue. Use a step-function as the non-linear activation function at designated nodes:

$$s(a) = \begin{cases} 1, & \text{if } a > 0 \\ 0, & \text{otherwise} \end{cases}$$

   (b) Determine coefficients $\boldsymbol{W} = \begin{bmatrix} w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \end{bmatrix}^T$, $v = [v_0, v_1, v_2]^T$ in the 2-layer Perceptron of the form in Figure 4 to recognize the shaded region in Figure 5.
   Hint: The shaded region in Figure 5 equals to the intersection of the regions of Figures 2 & 3.

[1] Instructor: Catherine Qi Zhao. TA: Shi Chen, Xianyu Chen, Helena Shield, Jinhui Yang, Yifeng Zhang. Email: csci5521.f2021@gmail.com

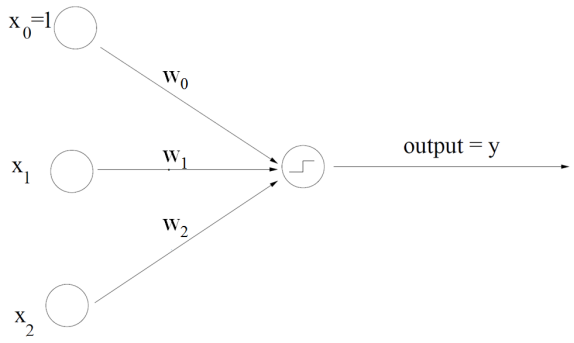

Figure 1: Single-layer perceptron. The non-linearity is a step function yielding a discrete value 1/0.

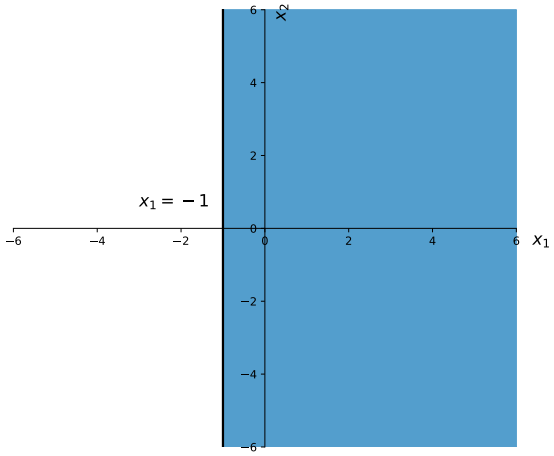

Figure 2: Accept shaded region.

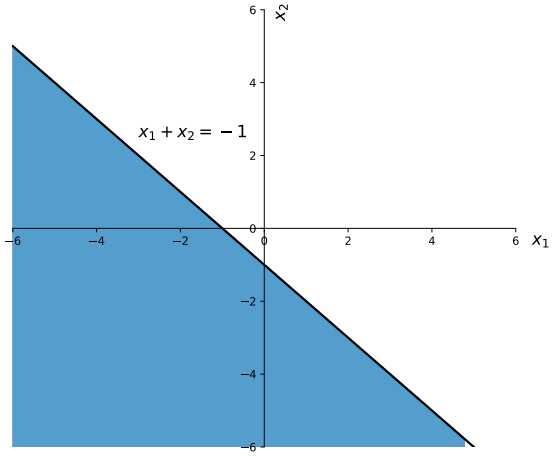

Figure 3: Accept shaded region.

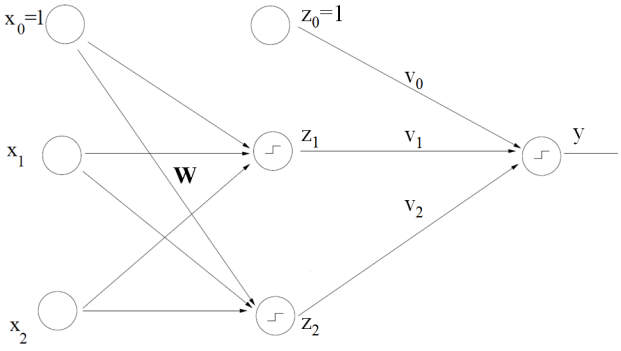

Figure 4: Multi-layer perceptron.

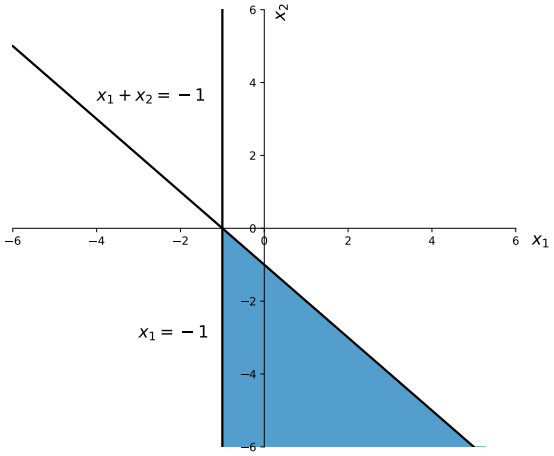

Figure 5: Accept shaded region (intersection of Figures 2 & 3)

3. (**40 points**) In this programming exercise you will implement a Multilayer Perceptron (MLP) for optical-digit classification. You will train your MLP using 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. For each file, the first 64 columns correspond to the features for different samples while the last one stores the labels. Features in each matrix should be normalized as $X_{norm} = \frac{(X-\mu_{trn})}{\sigma_{trn}}$. Notice that μ and σ, the mean and the standard deviation, are always calculated from the training set (even when normalizing the validation and test set).

   (a) Implement a MLP with 1 hidden layer for classifying the 10 digits (read the algorithm in Figure 11.11 and section 11.7.3 in the textbook), use the tanh activation function $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ for the hidden layer, and the softmax activation function $\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_k e^{x_k}}$ for the output layer. The error function is the cross-entropy loss:

$$E(r^t, y^t) = -\sum_i^C r_i^t \log y_i^t \tag{1}$$

   where y^t is the predicted probabilities for different classes, C is the number of candidate labels. r^t is the one-hot vector for ground truth label, $r_i^t = 1$ if the current sample belongs to the i_{th} class and 0 otherwise.

   Try MLPs with $H = 4, 8, 12, 16, 20$ and 24 hidden units. **Report the validation accuracy by the number of hidden units. How many hidden units should we use? Report the accuracy on the test set using this number of hidden units.** Hint: Given $y_i = \text{softmax}(\alpha_i) = \frac{e^{\alpha_i}}{\sum_k e^{\alpha_k}}$, the derivative $\frac{\partial y_i}{\partial \alpha_j} = y_i(\delta_{ij} - y_j)$, where $\delta_{ij} = 1$ if $i = j$ and 0 otherwise.

   (b) Train your MLP with 2 hidden units and visualize the data by the values of the hidden units (similar to Figure 11.18). **Make a plot for the training, validation, and test sets separately**. Use different colors for different digits.

   (c) Repeat previous question by training 3 hidden units and do the visualization using 3-D plot. **Compare the 2-D and 3-D plots and explain the results in the report.**

   We have provided the skeleton code `MyMLP.py` and `visualization.py` for implementing the algorithm. `MyMLP.py` is written in a *scikit-learn* convention, where you have a *fit* function for model training and a *predict* function for generating predictions on given samples. To verify your implementation, call the main function `hw3.py`.

Submission

- **Things to submit:**

  1. hw3_sol.pdf: a document containing all your answers for the written questions (including answers and plots for problem 4).
  2. `MyMLP.py`: a Python source file containing your implementation of the MLP algorithm (`class MLP`, `def tanh` and `def softmax`), and functions for data processing (`def process_data` and `def process_label`). Use the skeleton file `MyMLP.py` found with the data on the class web site, and fill in the missing parts. The `fit` function should take both training and validation samples as inputs, update the model parameters based on training samples and determine convergence based on validation accuracy. It should also return the best validation accuracy. The *predict* function should take features as inputs and return the predicted class labels. The *get_hidden* function should extract features computed at the hidden layers for given inputs. The *process_data* and *process_label* functions should normalize the features and convert class labels into one-hot vectors, respectively.
  3. `visualization.py`: a Python source file containing your code for visualizing the hidden features. Given the hidden features of different samples and their class labels as inputs, the *plot2d* and *plot3d* functions should generate 2D and 3D scatter plots for the features, respectively. Samples from different classes should be labeled with different colors.

- **Submit**: All material must be submitted electronically via Gradescope. **Note that There are two entries for the assignment, *i.e.,* Hw3-Written (for hw3_sol.pdf) and Hw3-Programming (for a zipped file containing the Python code), please submit your files accordingly.** We will grade the assignment with vanilla Python, and code submitted as iPython notebooks will not be graded.