# CSCI 5521: Machine Learning Fundamentals (Spring 2025)

## Homework 3

## Due date: Apr 11, 2025 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, 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 $y^t$ =Sigmoid$(\sum_{h=1}^{H} v_h z_h^t + v_0)$ and $z_h^t$ =reLU$(w_h^T x^t + w_{h0})$. Derive the update equations of the regularized MLP using the given activation functions. Note that you need to show the update equations for all trainable parameters.

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

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.

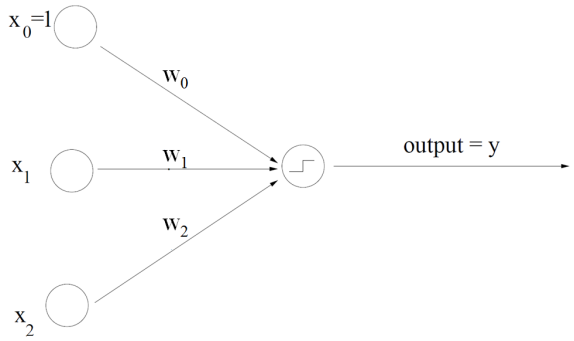


Figure 1: Single-layer perceptron. The non-linearity is a step function yielding a discrete value 1/0.

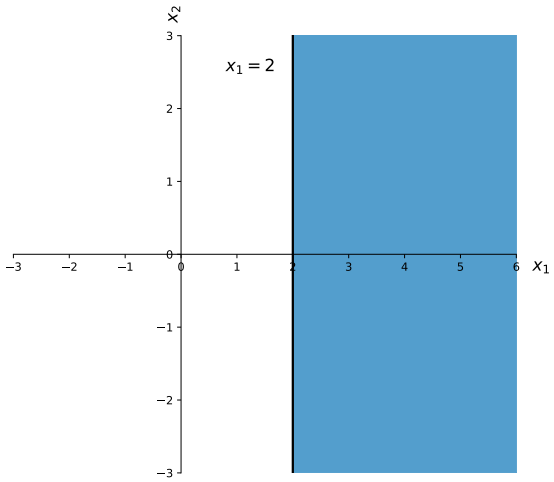


Figure 2: Accept shaded region.

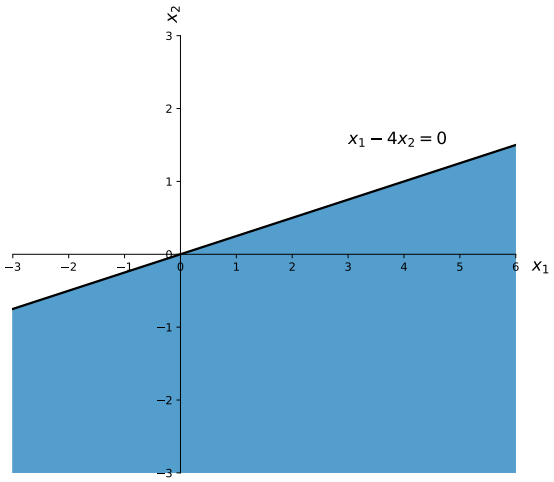


Figure 3: Accept shaded region.

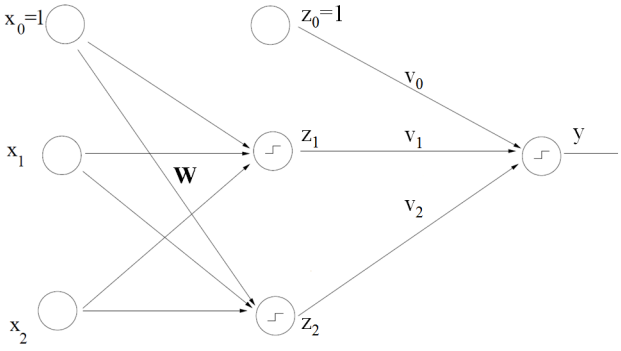


Figure 4: Multi-layer perceptron.

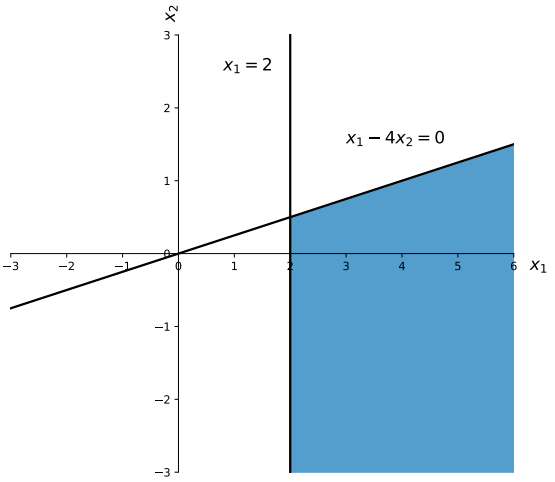


Figure 5: Accept shaded region (intersection of Figures 2 & 3)

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 $\mu$ and $\sigma$, the mean and the standard deviation, are always calculated from the training set (even when normalizing the validation and test set).

   **Task**: Implement a MLP with 1 hidden layer for classifying the 10 digits (read section 11.7.3 in the textbook, note that each update should consider all training samples and the equation we use may be different from those in the book), 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.
   We provide the skeleton code in class `MyMLP`. `MyMLP` 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. Please follow the comments to fill in the functions and derive the update equation for the MLP. Table 1 provides a brief overview

of the functions to be implemented.

| functions | descriptions |
|---|---|
| predict(x) | implement predict function of MLP (from input x to the classification results) |
| dEdv(*) | derive the partial derivative of dE/dv |
| dEdv0(*) | derive the partial derivative of dE/dv0 |
| dEdw(*) | derive the partial derivative of dE/dw |
| dEdw0(*) | derive the partial derivative of dE/dw0 |
| update(*) | update all parameters by leveraging the derivatives |

Table 1: List of functions to be implemented. *Please refer to the code comments for detailed implementations.

**Hint 1**: Given $y_i$ =softmax$(\alpha_i) = \frac{e^{\alpha_i}}{\sum_k e^{\alpha_k}}$, the derivative is $\frac{\partial y_i}{\partial \alpha_j} = y_i(\delta_{ij} - y_j)$, where $\delta_{ij} = 1$ if $i = j$ and 0 otherwise.
**Hint 2**: Instead of manually checking the values of $i$ and $j$, you could write down the whole update equation and simplify it to cancel out $\delta_{ij}$.

# Submission

- **Things to submit:**
  1. hw3_written.pdf: a document containing all your answers for the written questions.
  2. `hw3.ipynb`: a Jupyter notebook containing solutions and answers for Problem 3. Use the skeleton code and fill in the missing parts and answer the questions in markdowns.
- **Submit**: All material must be submitted electronically via Canvas. Please **Do NOT** submit `zip` files. Instead, submit two separate files, hw3_written.pdf and `hw3.ipynb`.