CIS 520, Machine Learning, Fall 2020
Homework 4
Due: Monday, October 12th, 11:59pm
Submit to Gradescope

Instructions. Please write up your responses to the following problems clearly and concisely. We require you to write up your responses using LaTeX; we have provided a LaTeX template, available on Canvas, to make this easier. **Submit your answers in PDF form to Gradescope. We will not accept paper copies of the homework.**

Collaboration. You are allowed and encouraged to work together. You may discuss the **written homework** to understand the problem and reach a solution in groups. However, **it is recommended that each student also write down the solution independently and without referring to written notes from the joint session.** You must understand the solution well enough to reconstruct it by yourself. (This is for your own benefit: you have to take the exams alone.)

Homework Submission Instructions

Written Homeworks

All written homework **must** be submitted as a PDF to Gradescope. **Handwritten assignments (scanned or otherwise) will not be accepted.** We require the use of LaTeXto generate your final PDF. We will be posting the homeworks in both PDF and LaTeX source form, and we encourage you to use this source as a template for your submission. **We recommend using Overleaf,** a free online LaTeX editor, though you are welcome to edit assignments however you like.

Coding Homeworks

All coding assignments will be done in Jupyter Notebooks. We will provide a `.ipynb` template for each assignment as well as function stubs for you to implement. Your final submission will be a `.py` file compiled from your notebook submitted to Gradescope. All code you have written that is not an import or contained within the function definitions must be commented out to ensure the autograder works properly. Though you are free to use your own installation of Jupyter for the assignments, **we recommend using Google Colab**, which provides a Jupyter environment connected to Google Drive along with a hosted runtime containing a CPU and GPU.

Learning Objectives

After completing this assignment, you will be able to:

- Understand how backpropagation works.
- Compare different classification algorithms and understand how hyper-parameters work.
- Understand how Convolutional Neural Network works and use them for multi-classification task.
- Understand how AdaBoost works.
- Understand role of support vectors in SVM.

Deliverables

This homework can be completed individually or in groups of 2. You need to make one submission per group. Make sure to add your team member's name on Gradescope when submitting the homework's written and coding part. Please view the following link if you are not familiar with adding group member's name for Gradescope submission - https://help.gradescope.com/article/m5qz2xsnjy-student-add-group-members

1. **A PDF compilation of `hw4.tex`**
2. **A `.ipynb` file with the functions implemented**

1 Neural Networks: Backpropagation [15 points]

Compute the gradients and derivatives of the loss function of the given neural network as shown in figure 1 with respect to the parameters: W_1, W_2, b_1 and b_2. Where W_1, W_2 are the weight matrices and b_1 and b_2 are the bias vectors. Let $x \in \mathcal{R}^2$, $W_1 \in \mathcal{R}^{2*500}$, $b_1 \in \mathcal{R}^{500}$, $W_2 \in \mathcal{R}^{500*2}$ and $b_2 \in \mathcal{R}^2$.

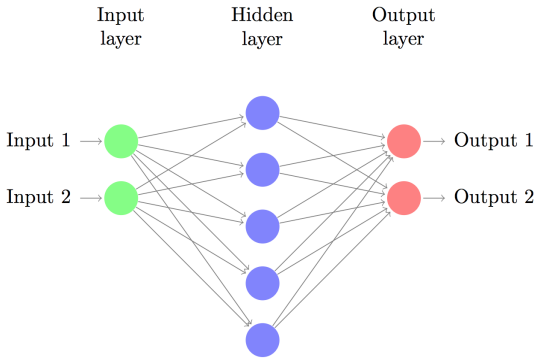


Figure 1: Neural Network

Note: This question serves as a good exercise for you to see how forward propagation works and then how the gradients are computed to implement the backpropagation algorithm. Also, you will get comfortable with the computation of vector, tensor derivatives and vector/matrix calculus. To ease things for you, you

can work this out on paper. Scan the solution and then upload that in your latex file. Also, you can relax the problem and just compute the derivatives for the scalar case.

Let us first compute the forward propagation. Let x be the input. The first hidden layer is computed as follows:

$$z_1 = xW_1 + b_1 \tag{1}$$

We then apply a non-linear activation function to equation 1

$$a_1 = tanh(z_1) \tag{2}$$

The output layers' activation are obtained using the following transformation

$$z_2 = a_1 W_2 + b_2 \tag{3}$$

Finally, a *sigmoid:* σ is applied to equation 2 to get:

$$a_2 = \widehat{y} = \sigma(z_2) \tag{4}$$

where $\widehat{y}$ is the predicted output by the feedforward network.

[15 points] Your task is to now compute the derivatives of the loss function given in 5 with respect to W_1, W_2, b_1 and b_2 by hand, i.e., $\frac{\partial Loss}{\partial W_1}$, $\frac{\partial Loss}{\partial b_1}$, $\frac{\partial Loss}{\partial W_2}$ and $\frac{\partial Loss}{\partial b_2}$.

$$Loss = y * ln(\sigma(z_2)) + (1 - y) * ln(1 - \sigma(z_2)) \tag{5}$$

Show all the intermediate derivative computation steps. You might benefit from making a rough schematic of the backpropagation process. Also recall the derivatives of the sigmoid function and the tanh function:

$$\frac{d\sigma(z_2)}{dz} = \sigma(z_2) * (1 - \sigma(z_2)) \tag{6}$$

$$\frac{\partial tanh(z_1)}{\partial z_1} = 1 - tanh^2(z_1) \tag{7}$$

2 Programming [20 points]

For this question, refer the Jupyter Notebook. You will be using functions from sklearn and call them in the notebook. We will be using the MNIST digits dataset for a classification task. Familiarize yourself with the dataset on the sklearn website, here.

2.1 Random Forests

[3 points]

Tabulate the prediction results on the test set in Table 1.

n_estimators	Accuracy (%)
1	—
5	—
10	—
50	—
100	—
500	—

Table 1: Accuracy for the Random Forests classification problem on the test set

2.2 Kernel SVM

[3 points]

Tabulate the prediction results on the test set in Table 2.

kernel	Accuracy (%)
Linear	—
Poly	—
RBF	—

Table 2: Accuracy for the kernel SVM classification problem on the test set

2.3 Multi Layer Perceptron

[3 points]

Tabulate the prediction results on the test set in Table 3.

Network Architecture	Accuracy (%)
(3)	—
(10)	—
(10,10,10)	—
(20,50,20)	—

Table 3: Accuracy for the MLP classification problem on the test set

2.4 AdaBoost

[3 points]

Tabulate the prediction results on the test set in Table 4.

n_estimators	Accuracy (%)
1	—
5	—
10	—
50	—
100	—
150	—

Table 4: Accuracy for the AdaBoost classification problem on the test set

2.5 Short Answer

[8 points] Write a short paragraph describing your results for each of the four models with regards to performance trends and explain the influence of hyperparameters on these results.

3 Convolutional Neural Networks [35 points]

3.1 Theory

Answer the following questions.

1. **[2 points]** You have an input volume that is 86 x 86 x 16, and convolve it with 64 filters that are each 6 x 6, using a stride of 3 and a padding of 2. What is the output volume? (hint : the third dimension of each filter spans across the whole third dimension of the input)

2. **[2 points]** Suppose your input is a 450 x 450 color (RGB) image, and you are not using a convolutional network. If the first hidden layer has 200 neurons, each one fully connected to the input, how many parameters does this hidden layer have (including the bias parameters)?

3. **[2 points]** Using the following toy example, let's compute by hand exactly how convolutional layer works.

$$\text{input image} = \begin{bmatrix} 5 & 7 & 6 & 0 & 1 \\ 9 & 2 & 6 & 3 & 1 \\ 2 & 8 & 2 & 6 & 9 \\ 0 & 3 & 2 & 8 & 1 \\ 8 & 5 & 6 & 1 & 8 \end{bmatrix} \quad \text{filter 1} = \begin{bmatrix} 1 & -4 & 3 \\ 3 & -4 & -3 \\ 2 & 0 & 3 \end{bmatrix} \quad \text{filter 2} = \begin{bmatrix} 0 & 2 & 1 \\ -3 & 4 & -1 \\ 3 & 1 & 4 \end{bmatrix}$$

   Here we have a $5 \times 5 \times 1$ input image, and we are going to use 2 different filters with size $3 \times 3 \times 1$ and stride 1 with no padding as our first convolutional layer. Compute the outputs and complete table 5. (Hint: the output dimension is $3 \times 3 \times 2$).

Row	Column	Filter	Value
1	1	1	—
1	1	2	—
1	2	1	—
2	1	1	—

Table 5: Output from convolution

4. **[2 points]** Let's train a fully-connected neural network with 7 hidden layers, each with 25 hidden units. The input is 40-dimensional and the output is a scalar. What is the total number of trainable parameters in your network?

5. **[2 points]** State two advantages of convolutional neural networks over fully connected networks.

3.2 Programming

For this question, refer to the Jupyter Notebook. You will be using PyTorch to implement a convolutional neural network – the notebook will have detailed instructions. We will be using the fashion MNIST dataset for a classification task.

3.2.1 Convolutional Neural Network

[10 points] Look for the code marked `#TODO` in the notebook and complete the code between the segments `#Begin Your Code` and `#End Your Code`.
Add the accuracy and the loss curve from tensorboard in this report.

3.2.2 Network Architecture and Implementation

[10 points] Table 6 describes a partial baseline architecture for the CNN. Please implement this architecture by tuning on the given sets of hyperparameters to find the best baseline network. You are, however, free to change the architecture even further as long as you beat the accuracy of this optimal baseline.

Layers	Hyperparameters
Convolution 1	Kernel =(5,5,32); Stride = 1; Padding = 2; followed by BatchNorm = ?
ReLU 1	—
Avgpool 1	Kernel size = ?; Stride = ?; Padding = ?
Convolution 2	Kernel =(5,5,32); Stride = 1; Padding = 2; followed by BatchNorm = ?
ReLU 2	—
Avgpool 2	Kernel size = ?; Stride = ?; Padding = ?
Convolution 3	Kernel =(5,5,64); Stride = 1; Padding = 2; followed by BatchNorm = ?
ReLU 3	—
Avgpool 3	Kernel size = ?; Stride = ?; Padding = ?
Dropout	Probability = ?
Fully Connected Layer	Output Channels = 64; followed by BatchNorm = ?
ReLU 4	—
Fully Connected Layer	Output Channels = 10; followed by Softmax

Table 6: Baseline Architecture for Convolutional Neural Network

Kernel, Stride, Padding
(2, 2, 0)
(4, 3, 1)

Dropout
0.1
0.3
0.5

Batch Normalization
True
False

Figure 2: Hyperparameter Options for Convolutional Neural Network

If you are using your own network architecture, give an explanation for your choice of network and how it may be better than the best baseline network.

3.2.3 Accuracy

[5 points] Report the overall accuracy and the per-class accuracy of the best model:

Overall Accuracy	—

Table 7: Overall Accuracy for Convolutional Neural Network

Class	**Accuracy**
T-shirt/top	—
Trouser	—
Pullover	—
Dress	—
Coat	—
Sandal	—
Shirt	—
Sneaker	—
Bag	—
Ankle boot	—

Table 8: Per Class Accuracy for Convolutional Neural Network

Identify the problematic classes and list the possible reasons as to why these classes may have significantly lower accuracy compared to other classes.

4 Multiclass Adaboost and Support Vector Machines [30 points]

4.1 Adaboost: Theory

In this problem you will analyze the AdaBoost.M1 algorithm, a multiclass extension of AdaBoost. Given a training sample $S = ((x_1, y_1), \ldots, (x_m, y_m))$, where x_i are instances in some instance space $\mathcal{X}$ and y_i are multiclass labels that take values in $\{1, \ldots, K\}$, the algorithm maintains weights $D_t(i)$ over the examples (x_i, y_i) as in AdaBoost, and on round t, gives the weighted sample (S, D_t) to the weak learner. The weak learner returns a multiclass classifier $h_t : \mathcal{X} \to \{1, \ldots, K\}$ with weighted error less than $\frac{1}{2}$; here the weighted error of h_t is measured as

$$\text{er}_t = \sum_{i=1}^{m} D_t(i) \cdot \mathbf{1}(h_t(x_i) \neq y_i)\,.$$

Note that the assumption on the weak classifiers is stronger here than in the binary case, since we require the weak classifiers to do more than simply improve upon random guessing (there are other multiclass boosting algorithms that allow for weaker classifiers; you will analyze the simplest case here). For convenience, we will encode the weak classifier h_t as $\widetilde{h}_t : \mathcal{X} \to \{\pm 1\}^K$, where

$$\widetilde{h}_{t,k}(x) = \begin{cases} +1 & \text{if } h_t(x) = k \\ -1 & \text{otherwise.} \end{cases}$$

In other words, $\widetilde{h}_t(x)$ is a K-dimensional vector that contains $+1$ in the position of the predicted class for x and -1 in all other $(K-1)$ positions. On each round, AdaBoost.M1 re-weights examples such that examples misclassified by the current weak classifier receive higher weight in the next round. At the end, the algorithm combines the weak classifiers h_t via a weighted majority vote to produce a final multiclass classifier H:

Algorithm **AdaBoost.M1**

Inputs: Training sample $S = ((x_1, y_1), \ldots, (x_m, y_m)) \in (\mathcal{X} \times \{1, \ldots, K\})^m$
Number of iterations T

Initialize: $D_1(i) = \frac{1}{m} \quad \forall i \in [m]$

For $t = 1, \ldots, T$:

– Train weak learner on weighted sample (S, D_t); get weak classifier $h_t : \mathcal{X} \to \{1, \ldots, K\}$

– Set $\alpha_t \leftarrow \dfrac{1}{2} \ln\left(\dfrac{1 - \mathrm{er}_t}{\mathrm{er}_t}\right)$

– Update:

$$D_{t+1}(i) \leftarrow \frac{D_t(i) \exp(-\alpha_t \widetilde{h}_{t,y_i}(x_i))}{Z_t}$$

where $Z_t = \sum_{j=1}^{m} D_t(j) \exp(-\alpha_t \widetilde{h}_{t,y_j}(x_j))$

Output final hypothesis:

$$H(x) \in \arg\max_{k \in \{1,\ldots,K\}} \underbrace{\sum_{t=1}^{T} \alpha_t \widetilde{h}_{t,k}(x)}_{F_{T,k}(x)}$$

You will show, in five parts below, that if all the weak classifiers have error er_t at most $\frac{1}{2} - \gamma$, then after T rounds, the training error of the final classifier H, given by

$$\mathrm{er}_S[H] = \frac{1}{m} \sum_{i=1}^{m} \mathbf{1}(H(x_i) \neq y_i)\,,$$

is at most $e^{-2T\gamma^2}$ (which means that for large enough T, the final error $\mathrm{er}_S[H]$ can be made as small as desired).

(a) **[5 points]** Show that

$$D_{T+1}(i) \;=\; \frac{\frac{1}{m}\, e^{-F_{T,y_i}(x_i)}}{\prod_{t=1}^{T} Z_t}\,.$$

(b) **[5 points]** Show that

$$\mathbf{1}(H(x_i) \neq y_i) \;\leq\; \mathbf{1}\big(F_{T,y_i}(x_i) < 0\big)\,.$$

Hint: Consider separately the two cases $H(x_i) \neq y_i$ and $H(x_i) = y_i$. For the $H(x_i) \neq y_i$ case, you need to show $F_{T,y_i}(x_i) \leq 0$. Note that $\sum_{k=1}^{K} F_{T,k}(x_i) = -(K-2) \sum_{t=1}^{T} \alpha_t$, which will be a useful equality.

(c) **[5 points]** Show that

$$\mathrm{er}_S[H] \;\leq\; \frac{1}{m} \sum_{i=1}^{m} e^{-F_{T,y_i}(x_i)} \;=\; \prod_{t=1}^{T} Z_t\,.$$

Hint: For the inequality, use the result of part (b) above, and the fact that $\mathbf{1}(u < 0) \leq e^{-u}$; for the equality, use the result of part (a) above.

(d) **[5 points]** Show that for the given choice of α_t, we have

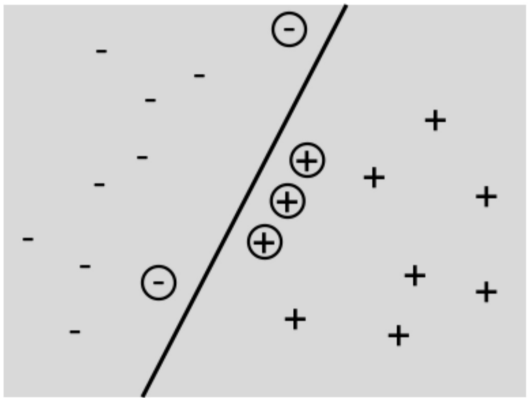
$$Z_t \;=\; 2\sqrt{\mathrm{er}_t(1-\mathrm{er}_t)}\,.$$

(e) **[5 points]** Suppose $\mathrm{er}_t \leq \frac{1}{2} - \gamma$ for all t (where $0 < \gamma \leq \frac{1}{2}$). Then show that

$$\mathrm{er}_S[H] \;\leq\; e^{-2T\gamma^2}\,.$$

4.2 Support Vector Machine

[5 points] Consider a binary classification problem in a 2-dimensional instance space $\mathcal{X} = \mathbb{R}^2$. You are given a linearly separable training set containing 10 positive and 10 negative training examples. You run the hard-margin SVM algorithm and obtain the separating hyperplane below (support vectors are circled):



What is the largest number of data points that can be removed from the training set without changing the hard margin SVM solution? Explain your solution.