Assignment 3

CS 5824: Advanced Machine Learning
Spring 2022

Due Date: April 12, 2022

- Feel free to talk to other members of the class when doing the homework. We are more concerned that you learn how to solve the problem than that you demonstrate that you solved it entirely on your own. You should, however, write down your solution only yourself. Please try to keep the solution brief and clear.
- Please use the discussion section on Canvas (`https://canvas.vt.edu/courses/145285/discussion_topics`) first if you have questions about the homework. Also feel free to send us e-mails and come to office hours.
- The homework is due at 11:59 PM on the due date. We will be using Canvas (`https://canvas.vt.edu/courses/145285`) for collecting homework assignments. Please do NOT hand in a scan of your handwritten solution, only the typed solution (e.g., Microsoft Word, Latex, etc) will be accepted for grading. Contact the TAs if you are having technical difficulties in submitting the assignment. We do NOT accept late homework!
- The homework submission should include two files: (1) a PDF using the name convention `yourFirstName-yourLastName.pdf`, and (2) a Python file (or a Jupyter Notebook) using the name convention `yourFirstName-yourLastName.py` (for notebook the name should be `yourFirstName-yourLastName.ipynb`).
- For each question, you will NOT get full credit if you only give out a final result. Necessary calculation steps are required. If the result is not an integer, round your result to 3 decimal places.

Problem 1. Clustering Basics (**15** points total)
The following table summarizes the clustering results of a newly designed algorithm where C_1, C_2, and C_3 denote the clusters, while T_1, T_2, and T_3 are ground truth.

(a) (5 points) Based on the Table 1, calculate the purity of the clustering algorithm.

(b) (5 points) Calculate the maximum matching score of the clustering algorithm.

(c) (5 points) Calculate F-measure of the clustering algorithm.

C/T	T_1	T_2	T_3
C_1	20	30	10
C_2	30	40	10
C_3	0	0	60

Table 1: Clustering Result.

Problem 2. K-Means (**25** points total)

(a) (10 points) Using the data in Table 2, compute the clustering result of K-Means for K = 2 with P3 and P9 as the initial centroids and using Euclidean distances as the distance metric.

(b) (10 points) Do the same as in part 1, but with different initial centroids: P1 and P5.

(c) (3 points) Are the results same or different? Based on the results, in what conditions, k-means will not give optimal results?

(d) (2 points) Output of the K-Means is sensitive to the initial centroids. Suggest a general method (independent of specific datasets) to choose the best initial centroids for K-Means.

Point	x	y
P1	1	1
P2	1	3
P3	1	4
P4	1	5
P5	0	4
P6	2	4
P7	3	4
P8	4	4
P9	5	4
P10	4	3
P11	4	5

Table 2: Dataset

Problem 3. Deep Learning Basics (**50** points total)

(a) Consider a basic neural network with the following settings; For this sub-problem, we only consider the case of $n = 1$ (i.e., only one sample per batch) for simplicity.

i. It has an input layer. The input layer takes a vector $\mathbf{x}$ that has a dimension of two as its input, i.e., $\mathbf{x} = \{x_1, x_2\}$.

ii. It has a hidden layer $\mathbf{H}$ with two hidden nodes (i.e., with a dimension of two) along with bias b_1; for this hidden layer, it uses ReLU function $ReLU$ as its activation function; the weight of this layer is represented as $\mathbf{W}_1$.

iii. This neural network is a binary classification network, so the output layer $\mathbf{Z}$ (with bias b_2) has only one node and its output is treated as the predicted value $\hat{y}$. It uses sigmoid function σ as its activation function, so the predicted value should be a number between 0 and 1. The weight of this layer is denoted as $\mathbf{W}_2$.

iv. The optimization object is a binary cross-entropy loss function $\mathcal{L}$. For n samples, it is given by:

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \tag{1}$$

where $y^{(i)}$ is the label of the i-th training sample $x^{(i)}$, and $\hat{y}^{(i)}$ is the predicted value for the same sample produced by the neural network.

For each layer, we use superscript i to denote the input of the corresponding layer and use superscript o to denote the output of the same layer. Take the hidden layer $\mathbf{H}$ for example, its input is represented as $\mathbf{H}^i$ and the output is represented as $\mathbf{H}^o$. In this case, we have:

$$\mathbf{H}^i = \mathbf{x}$$
$$\mathbf{H}^o = ReLU(\mathbf{W}\mathbf{x} + b_1)$$

Then, the output the current layer is sent into the next layer as its input.

(1) (5 points) What is the size of weight $\mathbf{W}_1$ and $\mathbf{W}_2$? What is the sizes of the input and output for each layer? (Skip the input layer for this question)

(2) (5 points) Write an expression for the output $\mathbf{Z}^o$ (which is equivalent to $\hat{y}$) as a function of the input $\mathbf{x}$ and the weights of each layer. Feel free to break down your expression into several parts for ease of understanding.

(3) (10 points) Consider the weight parameter that connects x_1 with the first node in the hidden layer $\mathbf{H}$, called w_{12}. Derive an expression for the partial derivative of the loss with respect to w_{12}, that is, $\frac{\partial \mathcal{L}}{\partial w_{12}}$.

(4) (5 points) List at least three different modifications to this model to (potentially) achieve better performance.

(b) (25 points) In this sub-problem, you are required to implement a slightly more complicated neural network classifier to identify digit numbers. You will work on implementing the model using the MNIST dataset, which includes a large amount of handwritten digits. MNIST is very popular in the machine learning and data mining community for verifying learning models. It is accessible at this website `http://yann.lecun.com/exdb/mnist/`. By default, it is split into the training set and testing set, where the training set has 60k images of size 28×28 pixels, and the testing set has 10k images of the same image size. The total number of categories is 10 in the original MNIST dataset, i.e., the digits

from 0 to 9. Please note that these images are stored in grayscale, so you will have to treat them as images with **one color channel**. For this problem, you are allowed to use off-the-shelf deep learning packages, i.e., PyTorch. Here are some basic requirements:

- It has a similar architecture as the neural network introduced in the previous sub-problem, which consists of a hidden layer **H** sandwiched by an input and an output layer **Z**. For the hidden layer, either ReLU or sigmoid can be used as the activation function. For the output layer, you need to use softmax function as your activation function. You will use $d = 784 = 28 \times 28$ as your input vector dimension, one for each of the image's pixels. The recommended dimension h for the hidden layer is $h = 32$.
- To train the neural network you need to use cross entropy introduced in Eq. 1 as your loss function. You do not have to implement these functions yourself. Instead, you can use built-in functions such as `torch.nn.CrossEntropyLoss()`. Note that `torch.nn.CrossEntropyLoss()` incorporates a softmax function, so you do not need to explicitly include an activation function in the last layer of your network.[1]
- The recommended batch size would be 256 for training, and 1000 for testing. The optimizer could be SGD or Adam. The suggested learning rate is 0.01 for SGD and 0.001 for Adam.
- For the results, please (1) plot the training and testing loss curve w.r.t. the training epochs (40 epochs at least); (2) plot the training and testing accuracy w.r.t. the training epochs (40 epochs at least); You are required to have multiple runs (at least 3 runs) with different random network initialization, and report their mean and std for both curves; (3) Please report the hyperparameters you use, including the hidden dimension h, the batch size, the optimizer (SGD/Adam), and the learning rate.

Problem 4. K-Nearest Neighbor Classifier (**10** points total)
Considering the online learning situation where besides the observed training samples, we have new samples becoming available as time goes on, some classifiers have to be retrained from scratch.

(a) (5 points) When a new training sample becomes available, among SVM, Naive Bayes and KNN, which classifier(s) have to be retrained from scratch? Please justify your answer.

(b) (5 points) When a new test sample becomes available, among SVM, Naive Bayes and KNN, which classifier needs the most computation to infer the class label for this sample? What is the time complexity for this classifier (using $O(\cdot)$ notation) assuming we have n training samples? Please justify your answer.

[1] Check the document: `https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html`.