CSE343/ECE343: Machine Learning
Assignment-3: MLP, SVM

Max Marks: 25 (Programming: 15, Theory: 10) **Due Date**: 05/11/2023, 11:59 PM

Instructions

- Keep collaborations at high-level discussions. Copying/Plagiarism will be dealt with strictly.
- Late submission penalty: As per course policy.
- Your submission should be a single zip file **202xxxx_HW3.zip** (Where *2020xxx* is your roll number). Include **all the files (code and report with theory questions)** arranged with proper names. A single **.pdf report** explaining your codes with results, relevant graphs, visualization and solution to theory questions should be there. The structure of submission should follow:

  202xxxx_HW3
  |− code_rollno.py/.ipynb
  |− report_rollno.pdf
  |− (All other files for submission)

- Anything not in the report will **not** be graded.
- Remember to **turn in** after uploading on Google Classroom. No excuses or issues would be taken regarding this after the deadline.
- Start the assignment early. Resolve all your doubts from TAs in their office hours at least **two days before the deadline.**
- Your code should be neat and well-commented.
- **You have to do either Section B or C.**
- **Section A is mandatory.**

---

1. (10 points) **Section A (Theoretical)**

   (a) (3 points) Consider a regression problem where you have an MLP with one hidden layer (ReLU activation) and a linear output. Train the network using mean squared error loss. Given a dataset with inputs [1.2, 0.8, 2.0] and corresponding targets [3.0, 2.5, 4.0], perform a single training iteration and update the weights. Assume appropriate initial weights and a learning rate of 0.01.

   OR

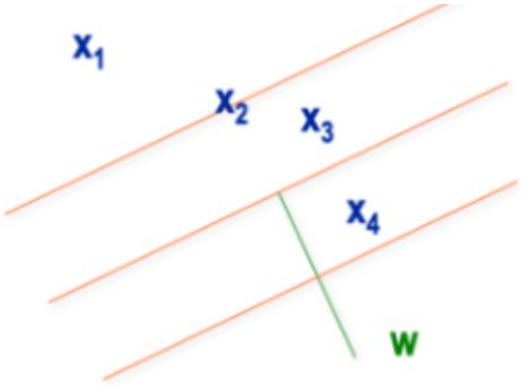
   (3 points) Implement the dropout regularization technique in an MLP with two hidden layers (ReLU activation). Apply dropout with a probability of 0.3 to the first hidden layer and a probability of 0.5 to the second hidden layer. Perform forward propagation for a given input [0.6, 0.9].

(b) (2 points) Suppose you have a nonlinear SVM trained with a Gaussian kernel. Write down the expression of the classification rule that you would apply on an unseen test sample.

Consider the SVM problem with the following formulation:

$$L(\mathbf{w}, b, \xi, \alpha, \lambda) = \frac{1}{2}\mathbf{w}^\top \mathbf{w} + C\sum_j \xi_i + \sum_i \alpha_i \left(1 - \xi_i - y_i \left(\mathbf{w}^\top \mathbf{x}_i + b\right)\right) - \sum_i \lambda_i \xi_i$$

Considering the figure given below, for each points what is the relation between ai and the hyperparameter C:



(a) Point X1 is on the correct side of the margin.

(b) Point X2 is on the margin.

(c) Point X3 is on the wrong side of the margin.

(d) Point X4 is on the wrong side of the decision hyperplane.

(c) (7 points) You are given the following training data points:

Data points	Label
(2,3)	+1
(6,7)	-1
(5,6)	-1
(4,1)	+1
(1,1)	+1
(7,8)	-1

(a) (1 point) Are the points linearly separable? Support your answer by plotting the points.

(b) (1 point) Find the equation of the decision boundary/optimal margin.

(c) (1 point) Determine the support vectors for the solution obtained in part b.

(d) (1 point) Calculate the margin of the decision boundary.

(e) (1 point) What is the effect on the optimal margin if we remove any one of the support vectors in this question?

2. (15 points) **Section B (Scratch Implementation)**

Neural Network Implementation

(1) (7.5 points) Implement a class named NeuralNetwork with the following parameters during initialization:

- N: Number of layers in the network
- A list of size N specifying the number of neurons in each layer
- lr: Learning rate
- Activation function (same activation function is used in all layers of the network except the last layer)
- Weight initialization function
- Number of epochs
- Batch size

The NeuralNetwork class should also implement the following functions:

- fit(X, Y): trains a model on input data X and labels Y
- predict(X): gives the prediction for input X
- predict_proba(X): gives the class-wise probability for input X
- score(X, Y): gives the accuracy of the trained model on input X and labels Y

(2) (2.5 points) Implement the following activation functions (along with their gradient functions): sigmoid, tanh, ReLU, Leaky ReLU, linear, and softmax (only used in the last layer).

(3) (2.5 points) Implement the following weight initialization functions: zero init, random init, and normal init (Normal(0, 1)). Choose appropriate scaling factors.

(4) (2.5 points) Train the implemented network on the MNIST dataset. Perform appropriate preprocessing. Use the following configurations for training the network:

- Number of hidden layers = 4
- Layer sizes = [256,128,64,32]
- Number of epochs = 100 (can be less if computation is taking too long)
- Batch size = 128 (or any other appropriate batch size)

Plot training loss vs. epochs and validation loss vs. epochs for each activation function. Also, save all the trained models as you might be asked to run them during the demo.

OR

3. (15 points) **Section C (Algorithm implementation using packages)**

Multi-Layer Perceptron (MLP) Implementation on SVHN Dataset

In this question, you'll implement a multi-layer perceptron using the built-in 'MLPClassifier' from scikit-learn and apply it to the SVHN dataset (format 2).

1. Data Preparation: (3 points)

1. Download train_32x32.mat from http://ufldl.stanford.edu/housenumbers/ (format 2)
2. Split the training dataset into validation (20%) and testing (10%) sets using `train_test_split` from scikit-learn. (1 point)
3. Visualize the distribution of class labels in the training, validation, and test sets. (1 point)
4. Visualize 5 unique samples from the training data. (1 point)

2. Model Training and Activation Functions: (9 points)

1. Construct a neural network with 2 hidden layers (excluding input and final layers). (1 point)
2. Use Grid-Search to find optimal hyperparameters for batch size and other model parameters. (2 points)
3. Train the model using various activation functions: sigmoid, ReLU, tanh, and linear. (2 points)
4. For each activation function, plot the training loss vs. epochs and validation loss vs. epochs. (1 point)
5. Analyze whether the model can learn effectively with all activation functions. If not, provide justifications. (2 points)
6. Determine the best accuracy achieved on the test set. Is your model able to achieve a decent accuracy? If yes, explain how you reached the set of hyperparameters. (1 point)

3. Visualization of Incorrect Predictions: (3 points)

1. For each class in the test set, visualize 3 misclassified images along with their predicted class labels. (1.5 points)
2. Analyze possible reasons for model misclassification. Consider whether the misclassified images resemble the predicted class more than the actual class, or if other factors are contributing. (1.5 points)