CSE/ECE 343/543: Machine Learning
Assignment-2

Max Marks: 90+25(UG), 90+35(PG) (35-Extra Credit)
Due Date: 11:59 PM Mar. 24, 2021

Instructions

- Keep collaborations at high level discussions. Copying/Plagiarism will be dealt with strictly.
- Start early, solve the problems yourself. Some of these questions may be asked in Quiz/Exams.
- **Q1 and all theory questions must be attempted individually.**
- **Q2 and Q3 must be attempted with your group.**
- **Q4 may be attempted individually for extra credit.**
- **Please submit solutions of all questions (group and individual). For coding questions, both Notebook and Python files should be submitted.**
- Use Google Colab to train and save your models. You can follow this tutorial to get started with Google Colab. You can upload your data on Google Drive and access it from there.
- **Save your models for every question (You might be asked to run the saved models during the demo)**
- Late submission penalty: As per course policy.

PROGRAMMING QUESTIONS

1. (60 points) **Backpropagation - To be attempted individually**
   You have to implement a general algorithm for Neural Networks. You can only use the numpy library. Use the attached **Q1.py** for implementing the algorithm.

   1. (2) The network should have the following parameters
      - **n_layers:** Number of Layers (int)
      - **layer_sizes:** array of size n_layers which contains the number of nodes in each layer. (array of int)
      - **activation:** activation function to be used (string)
      - **learning_rate:** the learning rate to be used (float)
      - **weight_init:** initialization function to be used
      - **batch_size:** batch size to be used (int)

- **num_epochs:** number of epochs to be used (int)

2. (2+2+2+2+2 = 10) Implement the following activation functions with their gradient calculation too: **ReLU, sigmoid, linear, tanh, softmax**.
3. (1+1+1 = 3) Implement the following weight initialization techniques for the hidden layers:
   - **zero:** Zero initialization
   - **random:** Random initialization with a scaling factor of 0.01
   - **normal:** Normal(0,1) initialization with a scaling factor of 0.01
4. (9+2+2+2 = 15) Implement the following functions with bias=0 and **cross entropy loss** as the loss function. You can create other helper functions too.
   - **fit():** accepts input data & input labels and trains a new model.
   - **predict_proba():** accepts input data and returns class wise probability.
   - **predict():** accepts input data and returns the prediction using the trained model.
   - **score():** accepts input data and their labels and returns accuracy of the model.
5. (24) Use the **MNIST** dataset for training and testing the neural network model created above.
   (a) (3) Use the following architecture [#input, 256, 128, 64, #output], learning rate=0.1, and number of epochs=100. Use normal weight initialisation as defined in the first question. Save the weights of the trained model separately for each activation function defined above and report the test accuracy.
   (b) (12) Plot training loss vs epoch curve and validation loss vs epoch curve for ReLU, sigmoid, linear and tanh activation function. Finally, you should have 4 graphs for the 4 activation functions.
   (c) (3) In every case, what should be the activation function for the output layer? Give reasons to support your answer.
   (d) (1) What is the total number of layers and number of hidden layers in this case?
   (e) (5) Visualise the features of the final hidden layer by creating tSNE plots for the model with highest test accuracy. You can use sklearn/ matplotlib for visualization.
6. (6) Implement the same architecture using sklearn's MLP classifier. Use sklearn's fit, predict and score methods to get the accuracy on test data. Try all the 4 activation functions - Relu, Sigmoid, Linear and tanh. Compare the accuracy achieved with sklearn's MLP vs your own implementation of MLP and comment on differences in accuracies if any.

2. (15 points) **Convolutional Neural Networks** Note: You are only allowed to use the PyTorch framework in Python. Explicitly mention each parameter considered (Number of filters, filter size, type of pooling).

- Train a CNN consisting of convolutional layers, pooling layers and FC layers on the FashionMNIST dataset.
- (3) Use the following architecture Conv->Relu->MaxPool->Conv->Relu->Maxpool->FC->FC->FC. Choice of dimensions is left to you.
- (3+3+6 = 12) Report the accuracy, confusion matrix and loss-per-epoch plots on the training and test sets.

3. (15 points) **Text Classification**
   Use this data to classify a review as positive or negative.

   - (1+1+1=3) Preprocess the data (You can use nltk library for this step):
     - Remove punctuation signs
     - Lower case the words
     - Remove stop words
   - (5) Do word vectorization using TF-IDF (sklearn).
   - (5) Then, Train a SVM using sklearn. Perform a grid search using GridSearchCV on the hyperparameters.
   - (2) Report Accuracy, F1 score, and confusion matrix for the best hyperparameters.

4. (35 points) **Extra credit question - To be attempted individually**

   - Use the existing VGG16 model in Pytorch (pretrained on ImageNet) for this problem. Fine tune the pretrained model on CIFAR10 dataset using the following steps:-
     - Divide the training batch of the dataset into train set and validation set (80:20 split)
     - Freeze all the weights of the lower convolutional layers.
     - Replace the upper dense layers of the network with custom fully connected layers of your own. Please note that the number of outputs of your model would be equal to the number of classes of the CIFAR10 dataset.
     - Train only the custom added layers of this model using the training set, therefore, optimizing the VGG16 model for your dataset.
   - (15) Report the accuracy and confusion matrix on the test set using the fine tuned model.
   - (10) Report class wise accuracy and confusion matrices and analyse your findings.
   - (10) Add loss plots of training and validation set for convergence obtained while training. Plot the losses for training and validation set on the same plot. Justify the choice of preprocessing and hyperparameters used, if any.

THEORY QUESTIONS

1. (5 points) The KL divergence between two discrete probability distributions P and Q is given by $KL(P|Q) = \sum_z P(z) \log \frac{P(z)}{Q(z)}$. Show that the cross-entropy loss used for multi-class classification is the same as the KL divergence under certain assumptions of the posterior distribution $P(y|x)$, where y are the class labels, while x are the data samples. What are these assumptions?

2. (10 points) Consider the following neural network and the two activation functions:

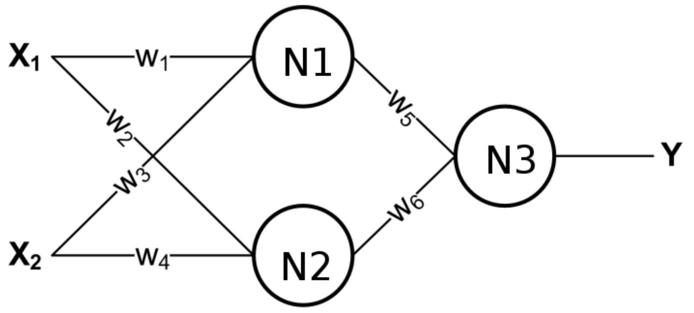

Figure 1

- A1: Signed Sigmoid function $S(x) = \text{sign}[\sigma(x) - 0.5] = \text{sign}[\frac{1}{1+\exp(-x)} - 0.5]$
- A2: Linear activation function $L(x) = Cx$

a) (5 points) What would be the activation function (A1 or A2) for each neuron (N1, N2 and N3) so that the output of the neural network is the binary logistic linear regression classifier $Y = \arg\max_y P(Y = y|X)$, where $Y = P(Y = 1|X) = \frac{\exp(\beta_1 X_1 + \beta_2 X_2)}{1+\exp(\beta_1 X_1+\beta_2 X_2)}$, $Y = P(Y = -1|X) = \frac{1}{1+\exp(\beta_1 X_1+\beta_2 X_2)}$

b) (5 points) Derive the values of β_1 and β_2 in terms of weights $w_0, ...w_6$.

3. (10 points) Consider the set of binary function overs d boolean (binary) variables:

$$F := \{f : \{0,1\}^d \to \{0,1\}\} \tag{1}$$

Prove that any function in F can be represented by a neural network with single hidden layer.

4. (10 points) (**For PG Students only**) Consider a CNN architecture with input size of 24x24:

   1. Convolution Layer with 64 filters and filter size of 3x3
   2. MaxPooling Layer of size 2x2
   3. Convolution layer with 32 filters and filter size of 3x3
   4. MaxPooling Layer of size 3x3

5. Fully Connected Layer of size 100
6. Fully Connected Layer of size 10

Determine the following for each layer of the given architecture:

1. Size of the feature map.
2. Trainable parameters

Show your calculations properly and also mention the total number of parameters for the whole architecture along with output size. (State your assumptions clearly if any).