Lab Course Machine Learning

Exercise Sheet 7

Prof. Dr. Dr. Lars Schmidt-Thieme
Kiran Madhusudhanan

HiWi: Basharat Mubashir Ahmed
Sophia Damilola Lawal

Submission deadline : January 7, 2023

General Instructions

1. Perform a data analysis, deal with missing values and any outliers.
2. Unless explicitly noted, you are not allowed to use scikit, sklearn or any other library for solving any part.
3. Data should be normalized.
4. Convert any non-numeric values to numeric values. For example you can replace a country name with an integer value or more appropriately use one-hot encoding.

1 Perceptron Algorithm. (5 points)

A perceptron is a simple one node of a deeper neural network which processes weighted inputs and performs binary classification. We use a toy dataset for the problem. Set aside 50 examples in each for testing. Using an OOP code up the perceptron algorithm. The task also involves to generate an **animation** showing how the decision boundary varies in each **iteration** for both training and testing. Load the following datasets in your notebook:

1. Xlin_sep.npy and ylin_sep.npy. This dataset is linearly separable. Run your algorithm for this data and you should achieve 100% train and test accuracies!
2. Xlinnoise_sep.npy and ylinnoise_sep.npy. This dataset is not linearly separable and contains noise. Run your algorithm for this data and observe what happens to the decision boundary in the animation. You should get a test accuracy close to 88%.
3. circles_x and circles_y. This dataset is non-linear. Devise a strategy to make the dataset separable linearly. Hint: **Polynomial Features**. Plot the decision boundary showing how the two classes are separated.

2 Feed-Forward Neural Network. (10 points)

This problem intends to teach how a feed-forward neural network functions. Implement a general neural network with forward propagation, backpropagation for computing derivatives and the optimizer Stochastic Gradient Descent. Do not fix the number of layers in advance. Your implementation should work with any number of layers. Also, your code should be flexible for choice of the activation functions. Limit these to: i) ReLU ii) tanh. Dataset to be used for this problem is MNIST classification data. Load this data using sklearn.

3 MLP using sklearn. (5 points)

In this problem, you will use the same dataset from Question 2 and implement a multi-layer perceptron using sklearn. Set aside 20% of the image for testing. Import the necessary classes and do a 5-fold cross-validation by defining a hyperparameter grid for the MLP classifier. Read about the hyperparameters supported and define a grid for them. Perform a random search on the grid that you have chosen. Report a single test accuracy with the best found hyperparameters.

4 **Bonus: Dropout. (2 points)

Neural networks are prone to overfitting and thus we have to regularize them. Dropout is one way of doing so. Implement a dropout layer to your implementation from Question 2.

5 **Bonus: Convolutional Neural Networks. (8 points)

1. Sign up on `www.kaggle.com` and visit the page `https://www.kaggle.com/asrsaiteja/car-steering-angle-predict` to see the dataset for this exercise
2. The task is that given the camera view (image) from the car, predict the steering angle.
3. Read the images in your code.
4. Divide these resulting arrays into corresponding train/validation/test splits. Leave the last 10k images for testing (images are id'ed). You are free to define the length of the validation split.
5. Implement the Convolutional Neural Network Architecture proposed in the paper titled, "End to End Learning for Self-Driving Cars". The paper can be accessed here: `https://arxiv.org/abs/1604.07316`.
6. Report one test RMSE for the test set of images.

You are free to use Tensorflow/Pytorch for this exercise. Google Colab is preferred here as it offers free GPU for faster computation.