Statistical Methods in AI

Instructor : Prof Ravi Kiran Sarvadevabhatla

Deadline : 21 October 2023 11:55 P.M

Assignment - 3

General Instructions

- Your assignment must be implemented in Python.
- While you're allowed to use ChatGPT for assistance, you must explicitly declare in comments the prompts you used and indicate which parts of the code were generated with the help of ChatGPT.
- Plagiarism will only be taken into consideration for code that is not generated by ChatGPT. Any code generated with the assistance of ChatGPT should be considered as a resource, similar to using a textbook or online tutorial.
- The difficulty of your viva or assessment will be determined by the percentage of code in your assignment that is not attributed to ChatGPT. If during the viva if you are unable to explain any part of the code, that code will be considered as plagiarized.
- Clearly label and organize your code, including comments that explain the purpose of each section and key steps in your implementation.
- Properly document your code and include explanations for any non-trivial algorithms or techniques you employ.
- Ensure that your Jupyter Notebook is well-structured, with headings, subheadings, and explanations as necessary.
- Your assignment will be evaluated not only based on correctness but also on the quality of code, the clarity of explanations, and the extent to which you've understood and applied the concepts covered in the course.
- Make sure to test your code thoroughly before submission to avoid any runtime errors or unexpected behavior.
- The Deadline will not be extended.

- Moss will be run on all submissions along with checking against online resources.
- We are aware how easy it is to write code now in the presence of ChatGPT and Github Co-Pilot, but we strongly encourage you to write the code yourself.
- We are aware of the possibility of submitting the assignment late in github classrooms using various hacks. Note that we will have measures in place for that and anyone caught attempting to do the same would be give zero in the assignment.

In this assignment, you are required to work with W&B library. Here are some resources on W&B logging and reporting :

1. Experiment management with Weights and Biases platform
2. Reports by W&B

Datasets for Task 1, 2, 3 are uploaded on moodle.

You can use PyTorch for Task 4 & Task 5.

1 Multinomial Logistic Regression
[Marks : 40, Estimated Time : 2-3 days]

Implement a Multinomial logistic regression model from scratch using **numpy** and **pandas**. You have to train this model on Wine Quality Dataset to classify a wine's quality based on the values of its various contents.

1.1 Dataset Analysis and Preprocessing [5 marks]

1. Describe the dataset using mean, standard deviation, min, and max values for all attributes.
2. Draw a graph that shows the distribution of the various labels across the entire dataset. You are allowed to use standard libraries like Matplotlib.
3. Partition the dataset into train, validation, and test sets. You can use sklearn for this.
4. Normalise and standarize the data. Make sure to handle the missing or inconsistent data values if necessary. You can use sklearn for this.

1.2 Model Building from Scratch [20 marks]

1. Create a Multinomial Logistic Regression model from scratch and Use cross entropy loss as loss function and Gradient descent as the optimization algorithm (write seperate methods for these).
2. Train the model, use sklearn classification report and print metrics on the validation set while training. Also, report loss and accuracy on train set.

1.3 Hyperparameter Tuning and Evaluation [15 marks]

1. Use your validation set and W&B logging to fine-tune the hyperparameters (learning rate , epochs) for optimal results.
2. Evaluate your model on test dataset and print sklearn classification report.

2 Multi Layer Perceptron Classification

[Marks : 60, Estimated Time : 4-5 days]

In this part, you are required to implement MLP classification from **scratch** using numpy, pandas and experiment with various activation functions and optimization techniques, evaluate the model's performance, and draw comparisons with the previously implemented multinomial logistic regression.

Use the same dataset as Task 1

2.1 Model Building from Scratch [20 marks]

Build an MLP classifier class with the following specifications:

1. Create a class where you can modify and access the learning rate, activation function, optimisers, number of hidden layers and neurons.
2. Implement methods for forward propagation, backpropagation, and training.
3. Different activation functions introduce non-linearity to the model and affect the learning process. Implement the Sigmoid, Tanh, and ReLU activation functions and make them easily interchangeable within your MLP framework.
4. Optimization techniques dictate how the neural network updates its weights during training. Implement methods for the Stochastic Gradient Descent (SGD), Batch Gradient Descent, and Mini-Batch Gradient Descent algorithms from scratch, ensuring that they can be employed within your MLP architecture.

2.2 Model Training & Hyperparameter Tuning using W&B [10 marks]

Effective tuning can vastly improve a model's performance. Integrate Weights & Biases (W&B) to log and track your model's metrics. Using W&B and your validation set, experiment with hyperparameters such as learning rate, epochs, hidden layer neurons, activation functions, and optimization techniques. You have to use W&B for loss tracking during training and to log effects of different activation functions and optimizers on model performance.

1. Log your scores - loss and accuracy on validation set and train set using W&B.
2. Report metrics: accuracy, f-1 score, precision, and recall. You are allowed to use sklearn metrics for this part.
3. You have to report the scores(ordered) for all the combinations of :
    - Activation functions : sigmoid, tanh and ReLU (implemented from scratch)
    - Optimizers : SGD, batch gradient descent, and mini-batch gradient descent (implemented from scratch).
4. Tune your model on various hyperparameters, such as learning rate, epochs, and hidden layer neurons.
    - Plot the trend of accuracy scores with change in these hyperparameters.
    - Report the parameters for the best model that you get (for the various values you trained the model on).
    - Report the scores mentioned in 2.2.2 for all values of hyperparameters in a table.

2.3 Evaluating Model [10 marks]

1. Test and print the classification report on the test set. (use sklearn)
2. Compare the results with the results of the logistic regression model.

2.4 Multi-Label Classification [20 marks]

For this part, you will be training and testing your model on Multilabel dataset: "advertisement.csv" as provided in Assignment 1.

1. Modify your model accordingly to classify multilabel data.
2. (a) Log your scores - loss and accuracy on validation set and train set using W&B.

(b) Report metrics: accuracy, f-1 score, precision, and recall.

(c) You have to report the scores(ordered) for all the combinations of :
- Activation functions : sigmoid, tanh and ReLU (implemented from scratch)
- Optimizers : SGD, batch gradient descent and mini-batch gradient descent (implemented from scratch).

(d) Tune your model on various hyperparameters, such as learning rate, epochs, and hidden layer neurons.
- Plot the trend of accuracy scores with change in these hyperparameters.
- Report the parameters for the best model that you get (for the various values you trained the model on).
- Report the scores mentioned in Point b for all values of hyperparameters in a table.

3. Evaluate your model on the test set and report accuracy, f1 score, precision, and recall.

3 Multilayer Perceptron Regression
[Marks : 50, Estimated Time : 3-4 days]

In this task, you will implement a Multi-layer Perceptron (MLP) for regression from scratch, and integrate Weights & Biases (W&B) for tracking and tuning. Using the **Boston Housing dataset**, you have to predict housing prices while following standard machine learning practices. In this dataset, the column MEDV gives the median value of owner-occupied homes in $1000's.

3.1 Data Preprocessing [5 marks]

1. Describe the dataset using mean, standard deviation, min, and max values for all attributes.
2. Draw a graph that shows the distribution of the various labels across the entire dataset. You are allowed to use standard libraries like Matplotlib.
3. Partition the dataset into train, validation, and test sets.
4. Normalise and standarize the data. Make sure to handle the missing or inconsistent data values if necessary.

3.2 MLP Regression Implementation from Scratch [20 marks]

In this part, you are required to implement MLP regression from scratch using numpy, pandas and experiment with various activation functions and optimization techniques, and evaluate the model's performance.

1. Create a class where you can modify and access the learning rate, activation function, optimisers, number of hidden layers and neurons.
2. Implement methods for forward propagation, backpropagation, and training.
3. Implement the Sigmoid, Tanh, and ReLU activation functions and make them easily interchangeable within your MLP framework.
4. Implement methods for the Stochastic Gradient Descent (SGD), Batch Gradient Descent, and Mini-Batch Gradient Descent algorithms from scratch, ensuring that they can be employed within your MLP architecture.

3.3 Model Training & Hyperparameter Tuning using W&B [20 marks]

1. Log your scores - loss (Mean Squared Error) on the validation set using W&B.
2. Report metrics: MSE, RMSE, R-squared.
3. You have to report the scores(ordered) for all the combinations of :
   - Activation functions : sigmoid, tanh and ReLU (implemented from scratch)
   - Optimizers : SGD, batch gradient descent and mini-batch gradient descent (implemented from scratch).
4. Tune your model on various hyperparameters, such as learning rate, epochs, and hidden layer neurons.
   - Report the parameters for the best model that you get (for the various values you trained the model on).
   - Report the scores mentioned in 3.3.2 for all values of hyperparameters in a table.

3.4 Evaluating Model [5 marks]

1. Test your model on the test set and report loss score (MSE, RMSE, R-squared).

4 CNN and AutoEncoders

[Marks : 100, Estimated Time : 5-6 days]

Welcome to the **Hello World** of image classification - a CNN trained on MNIST dataset. You can use Pytorch for this Task. You can load the MNIST dataset using **PyTorch's torchvision**.

4.1 Data visualization and Preprocessing [10 marks]

1. Draw a graph that shows the distribution of the various labels across the entire dataset. You are allowed to use standard libraries like Matplotlib.
2. Visualize several samples (say 5) of images from each class.
3. Check for any class imbalance and report.
4. Partition the dataset into train, validation, and test sets.
5. Write a function to visualize the feature maps. Your code should be able to visualize feature maps of a trained model for any layer of the given image.

4.2 Model Building [20 marks]

1. Construct a CNN model for Image classification using pytorch.
2. Your network should include convolutional layers, pooling layers, dropout layers, and fully connected layers.
3. Construct and train a baseline CNN using the following architecture: 2 convolutional layers each with ReLU activation and subsequent max pooling, followed by a dropout and a fully-connected layer with softmax activation, optimized using the Adam optimizer and trained with the cross-entropy loss function.
4. Display feature maps after applying convolution and pooling layers for any one class and provide a brief analysis.
5. Report the training and validation loss and accuracy at each epoch.

4.3 Hyperparameter Tuning and Evaluation [20 marks]

1. Use W&B to facilitate hyperparameter tuning. Experiment with various architectures and hyperparameters: learning rate, batch size, kernel sizes (filter size), strides, number of epochs, and dropout rates.
2. Compare the effect of using and not using dropout layers.
3. Log training/validation loss and accuracy, confusion matrices, and class-specific metrics using W&B.

4.4 Model Evaluation and Analysis [10 marks]

1. Evaluate your best model on the test set and report accuracy, per-class accuracy, and classification report.
2. Provide a clear visualization of the model's performance, e.g., confusion matrix.

3. Identify a few instances where the model makes incorrect predictions and analyze possible reasons behind these misclassifications.

4.5 Train on Noisy Dataset [10 marks]

In the subsequent parts, you have to work with a noisy mnist dataset. Download the **mnist-with-awgn.mat** from here. You can load .mat file using scipy.io.

1. Train your best model from the previous parts of Task 4 on the noisy mnist dataset which contains noise in it (additive white gaussian noise, don't worry it's just a fancy name).
2. Report validation losses, validation scores, training losses, training scores.
3. Evaluate your model on test data and print the classification report.

4.6 AutoEncoders to Save the Day [30 marks]

1. Implement an Autoencoder class which will help you de-noise the noisy mnist dataset from Part 4.5.
2. Visualise the classes and feature space before and after de-noising.
3. Now using the de-noised dataset, train your best model from the previous parts.
4. Report validation losses, validation scores, training losses, training scores.
5. Evaluate your model on test data and print the classification report.
6. Analyse and compare the results/accuracy scores as obtained in Part 4.5 and 4.6.

5 Some Other Variants

[Marks : 50, Estimated Time : 2-3 days]

You can use PyTorch for this Task.

5.1 Multi-digit Recognition on Multi-MNIST Dataset [25 marks]

Download the **DoubleMNIST** dataset from here. The DoubleMNIST dataset contains images with two handwritten digits, and the task is to correctly identify and classify each digit within the image.

Build and train models that can simultaneously recognize and predict the two digits from a single image.This is basically another version of a multilabel classification.

- Display several images from the filtered dataset to familiarize yourself with the dual-digit nature of the images.
- From the Multi-MNIST dataset, filter out and exclude images where the same digit appears twice. Your model will be trained only on images that have distinct digits
- Ensure you split the datasets into training and validation sets to evaluate the model's performance during hyperparameter tuning.

5.1.1 MLP on Multi-MNIST

1. Implement and train an MLP model on the MultiMNIST dataset.
2. Hyperparameter Tuning: Adjust the number of hidden layers and the number of neurons within each layer to optimize performance and find the best model.
3. Report the accuracies on the train and validation set.
4. Evaluate your trained model on test set and report the accuracy.

5.1.2 CNN on Multi-MNIST

1. Design and train a CNN model on the MultiMNIST dataset.
2. Hyperparameter Tuning: Experiment with different learning rates, kernel sizes, and dropout rates to determine the optimal configuration.
3. Report the accuracies on the train and validation set.
4. Evaluate your trained model on test set and report the accuracy.

5.1.3 Testing on Single digit MNIST (regular MNIST)

Evaluate your trained model on the regular MNIST dataset with single-digit images. See how the model, initially trained on images with two digits, performs on these single-digit images. Report the accuracies.

5.2 Permuted MNIST [15 marks]

The Permuted MNIST dataset is a variation of the original MNIST dataset where the pixels of each image are randomly permuted. This permutation makes the task significantly more challenging because it destroys the spatial structure of the images. In Permuted MNIST, the goal is still to recognize the digits but now without relying on the spatial relationships between pixels.

Ensure you split the datasets into training and validation sets to evaluate the model's performance during hyperparameter tuning.

5.2.1 MLP on Permuted-MNIST

1. Implement and train an MLP model on the Permuted-MNIST dataset.
2. Hyperparameter Tuning: Adjust the number of hidden layers and the number of neurons within each layer to optimize performance and find the best model.
3. Report the accuracies on the train and validation set.
4. Evaluate your trained model on test set and report the accuracy.

5.2.2 CNN on Permuted-MNIST

1. Design and train a CNN model on the Permuted-MNIST dataset.
2. Hyperparameter Tuning: Experiment with different learning rates, kernel sizes, and dropout rates to determine the optimal configuration.
3. Report the accuracies on the train and validation set.
4. Evaluate your trained model on test set and report the accuracy.

5.3 Analysis [10 marks]

1. Contrast the performances of MLP vs. CNN for both datasets.
2. Discuss the observed differences and any challenges faced during training and evaluation.
3. Compare the potential for overfitting between a CNN and an MLP in the context of datasets in Task 5. Use training vs. validation loss/accuracy plots to support your observations.

6 Report

1. The submission should consist of two separate files one for Task 1,2 and 3, one for Task 4 and 5.
2. Ensure you submit a clear and organized report using W&B. The quality of your report is essential for this assignment and holds significant weightage. Proper documentation and presentation of results align with good ML practices, so take the time to structure your W&B report effectively. You can submit 5 different W&B reports for all the 5 tasks.