SMAI Assignment 2

Prof. Vineet Gandhi

Deadline: 12th March, 2025

1 General Instructions

- Your assignment must be implemented in Python.
- Submit your assignment as a folder with a Jupyter Notebook files (.ipynb)
- The Deadline will not be extended.
- There is no need to make a separate report. All the observations, graphs etc. should be included as markdown in the notebooks itself.
- You have to implement MLP, GMM and PCA from scratch whereas you are allowed to use only pytorch framework for the autoencoder and variational autoencoder.

2 Multi Layer Perceptron

(60 Marks)

Meet Dr. Sam, a passionate data scientist who faces diverse challenges in his day-to-day work. He needs innovative solutions to:

1. **Digitize and classify handwritten symbols** from historical documents.
2. **Predict housing prices** in the booming tech hub of Bangalore to aid urban planners and real estate investors.
3. **Automatically categorize news articles** to help a renowned media house filter content for various topics.

Your task is to step into Dr. Sam's shoes by implementing Multi-Layer Perceptron (MLP) models from scratch. You will work on three distinct sub-problems using different datasets.
Details for each part is given below

2.1 MLP Multi-Class Classifier

(20 Marks)

Dr. Sam has acquired the **SYMBOL** dataset—a collection of handwritten symbols from historical manuscripts. His goal is to develop a digital archive where every symbol is automatically identified and categorized, preserving the cultural heritage for future generations.

2.1.1 Dataset

- The **images** folder contains all the images.
- The **classification-task** folder consists of 10 subfolders named:

  `fold-1`, `fold-2`, . . . , `fold-10`

  Each of these contains:
  - `train.csv` - Training dataset.
  - `test.csv` - Testing dataset.
- Each row in the CSV files includes:
  - Path to the image.
  - `symbol_id` (Unique but not continuous).
  - LaTeX representation of the symbol.
- `symbols.csv` contains a mapping of `symbol_id` to its corresponding LaTeX representation.

SOURCE: **Dataset Link**
Note: The `symbol_ids` are not continuous, so careful handling is required.

2.1.2 Model Development from scratch

- Develop a configurable MLP class that allows you to adjust the learning rate, activation function, optimizer, and the number/size of hidden layers.
- Implement methods from scratch for forward propagation, backpropagation, and the training process.
- Code the Sigmoid, Tanh, and ReLU activation functions so that they can be easily interchanged in your MLP framework.
- Build optimization routines from scratch for Stochastic Gradient Descent (SGD), Batch Gradient Descent, and Mini-Batch Gradient Descent, ensuring these can be integrated into your MLP.

2.1.3 Hyperparameter Tuning & Evaluation with 10-Fold Validation

- **Hyperparameters:**
  - **Learning Rate & Epochs:** Experiment to optimize convergence and prevent under/overfitting.
  - **Hidden Layers:** Try different numbers and sizes.
  - **Activation Functions:** Compare Sigmoid, Tanh, and ReLU (all implemented from scratch).
  - **Optimizers:** Evaluate SGD, Batch GD, and Mini-Batch GD (all implemented from scratch).
- **Result Aggregation & Visualization:**
  - Calculate the mean and standard deviation of evaluation metrics across all folds to find the best set of hyperparameters.
  - Report ordered scores for each activation-optimizer combination.
  - Plot accuracy and other metrics trends; include training curves and final test performance.
- **Report Questions:**
  - What do the mean and standard deviation tell you about model performance and consistency?
  - How does a high vs. low standard deviation impact confidence in the model's generalization?
  - If one configuration has a slightly higher mean accuracy but a significantly higher standard deviation compared to another with marginally lower mean accuracy, which would you choose and why?

2.2 MLP Regressor for Price Prediction in Bangalore

(20 Marks)

In Bangalore—a fast-growing city and the heart of India's tech revolution—a real estate agent named Rajesh is struggling to set competitive prices. Dr. Sam is called upon to create an MLP regressor that can accurately predict housing prices using various features such as location, size, and amenities.

2.2.1 Dataset

- **SOURCE:** Link

2.2.2 Data Preprocessing

- Clean up the dataset by addressing missing values, eliminating outliers, and removing features with a high percentage of missing data. You will need to manually review and fix the data because it is really unclean. Record your preprocessing.
- Describe each attribute using summary statistics (mean, standard deviation, minimum, and maximum).
- Visualize label distribution across the dataset using a graph (e.g., with Matplotlib).
- Partition the dataset into training, validation, and test sets. Also normalize and standardize the data.

2.2.3 Model Development from Scratch

- Develop a configurable MLP class that allows you to adjust the learning rate, activation function, optimizer, and the number/size of hidden layers.
- Implement methods from scratch for forward propagation, backpropagation, and the training process.
- Code the Sigmoid, Tanh, and ReLU activation functions so that they can be easily interchanged in your MLP framework.
- Build optimization routines from scratch for Stochastic Gradient Descent (SGD), Batch Gradient Descent, and Mini-Batch Gradient Descent, ensuring these can be integrated into your MLP.

2.2.4 Hyperparameter Tuning & Evaluation

- **Metrics:** Use MSE, RMSE, and R-squared.
- **Experimentation:** Vary hyperparameters (learning rate, epochs, architecture) and compare activation functions and optimizers. Record evaluation metrics for each configuration.
- **Reporting:** Plot training curves (loss vs. epoch) and final evaluation metrics on the test set. Report the ordered scores for each combination of activation function and optimizer and identify the best configuration.

2.3 Multi-Label News Article Classification

(20 Marks)

Develop an MLP from scratch to tag news articles with multiple topics using the provided CSV files (columns: `document` and `category`).
SOURCE: **Dataset Link**.

2.3.1 Data Preprocessing

- **Cleaning:** Parse CSV files, split comma-separated labels, and handle missing/malformed entries.
- **Feature Extraction:** Compute TF-IDF from scratch, limiting to ∼5000 features.
- **Label Transformation:** Apply multi-label binarization to convert topics into a binary matrix.
- **Splitting:** Divide the training set into training and validation subsets.

2.3.2 Model Development

- Develop a configurable MLP class that allows you to adjust the learning rate, activation function, optimizer, and the number/size of hidden layers.
- Implement methods from scratch for forward propagation, backpropagation, and the training process.
- Code the Sigmoid, Tanh, and ReLU activation functions so that they can be easily interchanged in your MLP framework.
- Build optimization routines from scratch for Stochastic Gradient Descent (SGD), Batch Gradient Descent, and Mini-Batch Gradient Descent, ensuring these can be integrated into your MLP.

2.3.3 Hyperparameter Tuning & Evaluation

- **Metrics:** Use Accuracy, Hamming Loss, etc.
- **Experimentation:** Vary hyperparameters (learning rate, epochs, architecture) and compare activation functions and optimizers. Record evaluation metrics for each configuration.
- **Reporting:** Plot training curves (loss vs. epoch) and final evaluation metrics on the test set. Report the ordered scores for each combination of activation function and optimizer and identify the best configuration.

3 Gaussian Mixture Model

(20 Marks)

- Implement a GMM from scratch. (10 Marks)

- Using the GMM above segment the gray matter, white matter and CSF from the image named sald_031764_img.nii. The image can be found here. Visualize the GMM segmentation and original segmentation using itksnap. The probability masks for the three labels can also be found in the above link. Report the pointwise accuracy of the segmentation(7 Marks)
- Visualize the Frequency of points vs intensity graph for all three labels. Also Visualize the GMMs distribution generated. Looking at these can you explain where is the highest misclassification and why? (3 Marks)

4 PCA

(30 Marks)
In this part of the assignment, you will implement Principal Component Analysis (PCA) from scratch and apply it to the MNIST dataset to analyze its effectiveness in dimensionality reduction. You will also investigate how reducing the dimensionality of image data affects classification performance using a Multi-Layer Perceptron (MLP) classifier. You will work on the following subtasks:

4.1 Explained Variance and Lossy Reconstruction

(10 Marks)

- Load the MNIST dataset (train and test sets) using Torchvision and flatten each sample.
- Randomly sample 1000 images from either of them (ensure uniform class distribution across these samples).
- Implement PCA from scratch (having all steps like computation of covariance matrix, eigenvectors and eigenvalues, etc., you can use numpy) and project this new dataset (of 1000 samples) to (500, 300, 150 and 30) dimensions separately.
- Plot the explained variance vs. the number of principal components graph.
- Visualize the samples using the first 2 PCs (using a scatter plot) and write your observations.
- Now, select any 5 images from these samples. Plot them before dimensionality reduction, and after projecting them back to the original space (do this for every type of final dimensions value). Write your observations.

4.2 Classification Performance with vs without dimensionality reduction

(15 Marks)

- Pick 40K random samples from the train set and the entire test set (10K samples). Now, train a MLP classifier model (any simple model, with 2-3 fully connected layers of appropriate size, using Sklearn) on the train set. Find the accuracy, precision and recall by running inference on the test set.
- Now, perform dimensionality reduction on the train and test sets. Train a new MLP model for classification with the new train set and report the above metrics for the new test set.
- Perform PCA for the above by taking dimensions = 500, 300, 150 and 30 (each separately). Write your observations from the performance in each case.

4.3 Report

(5 Marks)
Report the following:

- Plots and images from the above tasks, with your observations.
- How does PCA help mitigate the curse of dimensionality? Can you think of cases where PCA might not be effective in high-dimensional spaces?
- PCA assumes that the directions of maximum variance are the most informative. Is this always true? Provide an example where this assumption might fail.

5 AutoEncoder

(30 Marks)

1. Implement the encoder and decoder networks using PyTorch. (10 Marks)
2. Use the MNIST data set and select the last digit of your roll number as the normal digit and other digits as anomalous data. Train only on the normal data present in the train dataset and test on the whole test dataset (mix of normal and anomaly digits). (5 Marks)
3. Compute and plot histogram of reconstruction error for both normal and anomalous digits. (5 Marks)

4. Choose a threshold for anomaly detection based on reconstruction error distribution.

5. Evaluate model performance using: Precision, Recall, F1-score (5 Marks)

6. Hyperparameter Tuning: Choose 3 different bottleneck dimensions and compare their performance using the AUC-ROC score and plotting the ROC curve. You can use scikit-learn for the auc-roc score. Explain your observations. (5 Marks)

6 Variational AutoEncoder

(30 Marks)
A Variational Autoencoder (VAE) is a type of generative model that learns to encode input data into a structured, continuous latent space and then decode from that space to generate new data. Unlike traditional autoencoders, which map inputs to fixed latent representations, VAEs introduce a probabilistic approach by modeling the latent space as a distribution — typically a Gaussian. The key innovation of VAEs is the use of two loss components:

- Reconstruction Loss: Ensures the decoded output is close to the original input.

- KL Divergence Loss: Regularizes the latent space by keeping it close to a known prior distribution (like a standard normal distribution).

- Train a VaE on the MNIST dataset. Use binary cross entropy loss for the reconstruction loss. Visualize the latent space of the means. (10 Marks)

- What happens when we remove the reconstruction loss? Train the VaE without the reconstruction loss and visualize the latent space. Report the importance of the reconstruction loss (5 Marks)

- Repeact the above step with KL Divergence Loss (5 Marks)

- Keeping the latent space dimension as 2, sample points from a 2D Gaussian Grid, and visualize the reconstructions generated from the VaE on this Grid. Report your observations (5 Marks)

- Replace the binary cross entropy loss with a MSE loss and visualize a grid similar to above question. (5 Marks)