

Course: Deep Learning

Assignment 2

Deadline: 11:59 pm, 30th September, 2025

Task 1: Optimizer Performance on Non-Convex Functions

1. Non-Convex Function Optimization

a. Optimize the following functions:

i. $f(x, y) = (1 - x)^2 + 100(y - x^2)^2$

ii. $f(x) = \sin(1/x)$

Handle $(x = 0)$ by setting $(f(0) = 0)$ or slightly exceeding zero.

b. Apply the following optimizers at Learning rate $((\alpha))$: 0.01, 0.05, 0.1:

- - Gradient Descent
- - Stochastic Gradient Descent with momentum
- - Adam
- - RMSprop
- - Adagrad

Presentation of results:

- a) Convergence behaviour for each optimizer and time taken by each optimizer.
- b) Impact of hyperparameters on convergence speed and the final result, with plots showing convergence.

Note: We need the optimal x you got from the optimizer and the corresponding $f(x)$ for a fixed learning rate considering all the GD algorithms and whatever threshold criteria you have chosen to terminate your algorithm.

The question expects to implement the solution from scratch using Python programming language.

Task 2: Implementing Linear Regression Using a Multi-Layer Neural Network from Scratch

Dataset Description: Use the Boston Housing Dataset[[Link](#)] Focus on predicting the median value ['MEDV'] of homes based on two features

- Number of Rooms ('RM')
- Crime Rate ('CRIM')

Data Preprocessing:

- Normalize the features to ensure they are on a similar scale.
- Split the dataset into a training set (80%) and a test set (20%).

Model:

In this assignment, you will build a neural network from scratch to perform linear regression using Python. You will be implementing a multi-layer neural network with multiple hidden layers.

Details of the Architecture:

Input Layer:

- Number of Neurons: Equal to the number of features in the dataset (e.g., 2 input neurons if using 2 features from the Boston Housing Dataset).

Number of Hidden Layers: 2 hidden layers.

First Hidden Layer:

- Number of Neurons: 5 neurons.
- Activation Function: ReLU (Rectified Linear Unit) for non-linearity.

Second Hidden Layer:

- Number of Neurons: 3 neurons.
- Activation Function: ReLU (Rectified Linear Unit).

Output Layer:

- Number of Neurons: 1 neuron, since the task involves predicting a single continuous value (e.g., house price).

Training the Network:

- 1) Implement Gradient Descent for weight updates.
- 2) Experiment with different learning rates (e.g., 0.01, 0.001) to observe the effects on model training.
- 3) Train the network for a set number of epochs (e.g., 1000 epochs) and track the loss throughout the training process.

- 4) Implement and compare different optimizers:
 - a) Basic Gradient Descent
 - b) Momentum
 - c) Adam
- 5) Observe and compare how these optimizers affect the model's convergence and final performance.

Evaluation:

1. Evaluate the trained model on a test set.
2. Visualize the regression results by plotting the predicted vs. actual values.
3. Calculate evaluation metrics such as Mean Squared Error (MSE) on the test data to quantitatively assess the model's performance.

Bonus Questions:

1. Additional Hidden Layers:

- Add a third hidden layer with 2 neurons and observe the impact on model performance.
- Compare the performance of networks with different numbers of hidden layers.

2. Regularization:

- Implement L2 regularization (weight decay) and observe how it affects the model, especially with respect to overfitting.

Expectation of the question is to implement a solution from scratch using Python.

Task 3: Multi-class classification using Fully Connected Neural Network

Dataset Description

- 1) **Linearly separable classes dataset:** 3 class, 2-dimensional linearly separable data is given. Each class has 500 data points.
- 2) **Nonlinearly separable classes dataset:** 2-dimensional data of 2 or 3 classes that are nonlinearly separable. The number of examples in each class and their order is given at the beginning of each file.

Divide the data from each class into training, validation, and test data. From each class, train, validation, and test split should be **60%**, **20%** and **20%**, respectively.

Model: Fully connected neural network (FCNN) for each of the datasets. Try FCNN with one hidden layer for Dataset 1 (Linearly separable classes dataset) and 2 hidden layers for Dataset 2 (Nonlinearly separable classes dataset). Try different numbers of hidden nodes for both datasets and observe the results. Implement stochastic gradient descent (SGD) for the backpropagation algorithm. Use squared error as an instantaneous loss function.

Presentation of results:

- 1) Plot of average error (y-axis) vs epochs (x-axis): Give the plot only for the best architecture selected after cross-validation.
- 2) Decision region plot superimposed by training data for each of the datasets: Give the decision region plot for the best architecture selected after cross-validation.
- 3) **Confusion matrix and classification accuracy:** Give the confusion matrix and classification accuracy on the validation set for each of the different architectures, along with the best architecture selected after cross-validation. Also, for the best architecture, give the confusion matrix and classification accuracy on test data.
- 4) Plots of outputs for each of the hidden nodes and output nodes in FCNN for each of the datasets after the model is trained. Here, x and y axis are input variables of each example, z axis is output of hidden node/output node. Give the plots for training, validation, and test data. (Give the plots for the best architecture selected after cross-validation)

5) Comparison of performance with that of the single neuron model (Question-1) for each dataset.

6) Inferences on the plots and inferences on the results observed.

Expectation of the question is to implement perceptron from scratch using Python programming language.

Task 4: Multi-class classification using a Fully Connected Neural Network on the MNIST Dataset

Objective: The objective of this programming assignment is to deepen your understanding of the optimizers for backpropagation algorithms. The task is to train the fully connected neural network (FCNN) using different optimizers for the backpropagation algorithm and compare the number of epochs that it takes for convergence along with their classification performance.

Dataset Description: You are given the subset of the MNIST digit dataset for the same. Each group can choose any 5 classes. Divide the data into training and testing in 8:2. Every image is of the size 28 x 28. Flatten each image to represent it as a vector of 784-dimension (28 x 28).

The tasks for this assignment are as follows:

Develop an FCNN with different hidden layers (minimum number of hidden layers is 3 and maximum is 5).

- a) Use cross-entropy loss.
- b) Experiment with different a number of nodes in each of the layers.
- c) Train each of the architectures using:
 - i. Stochastic gradient descent (SGD) algorithm - (batch_size=1),
 - ii. Batch gradient descent algorithm (vanilla gradient descent) – (batch_size=total number of training examples),
 - iii. SGD with momentum (generalized delta rule) – (batch_size=1),
 - iv. SGD with momentum (NAG) – (batch_size=1),
 - v. RMSProp – (batch_size=1), and
 - vi. Adam optimizer – (batch_size=1).
- d) Use the same initial random values of weights for each architecture using each of the optimizers.
- e) Use the absolute difference between average error of successive epochs fall below a threshold $10e-4$ as stopping criteria. Do not use number of iterations as stopping criteria.

- f) Consider the learning rate (η) as 0.001 for all the optimizers.
- g) Consider momentum parameter as 0.9 for both generalized delta and NAG.
- h) Consider $\beta = 0.99$ and $\epsilon = 10^{-8}$ for RMSProp.
- i) Consider $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$ for Adam optimizer.

Presentation of Results:

1. Observe the number of epochs considered for convergence for each of the architectures. Tabulate and compare the number of epochs considered by each of the optimizers for each architecture.
2. Present the plots of average training error (y-axis) vs. epochs (x-axis) for each architecture. Superimpose the plots from each of the optimizers.
3. Give the training accuracy and validation accuracy for each of the optimizers in each of the architectures.
4. Choose the best architecture based on validation accuracy. Give the test confusion matrix and test classification accuracy along with training accuracy and confusion matrix for the chosen best architecture.

You can use deep learning APIs (PyTorch).

Note:

- You are not supposed to use pre-built functions for models from libraries like sklearn, and PyTorch except Task 4.
- Students will also be rewarded extra for solving bonus questions, thorough experimentation and insightful analysis based on the results and graphs they produce.

The report should be in PDF form, and the report by a team should also include the observations about the results of studies.

Instruction:

Upload all your codes(.ipynb file) and reports(.pdf file) in a single zip file.

- **Give the name of the folder as Group_Assignment2.**

Example: Group01_Assignment2

- Give the name of the zip file as Group_Assignment2.zip Example:
Group01_Assignment2.zip

We will not accept the submission if you don't follow the above instructions