Assignment 3

Total Marks: 11

Deadline: 31 October

1. Next-Word Prediction using MLP [5 marks]

In this question, you will extend the next-character prediction notebook (discussed in class) to a next-word prediction problem.That is you will create a MLP based text generator. You will train the model, visualize learned word embeddings, and finally deploy a Streamlit app for interactive text generation. It is recommended to refer to Andrej Karpathy's blog post on the Effectiveness of RNNs.

You must complete this task for two datasets: one from Category I (Natural Language) and one from Category II (Structured/Domain Text).

1.1 Preprocessing and Vocabulary Construction [0.5 mark]

For text-based datasets, you can remove special characters except "full stop (.)" so that it can be used to split sentences. However, you cannot ignore special characters for other datasets like for C++ code. You will have to treat text between newlines as a statement. To remove special characters from a line, you can use the following code snippet:

```
import re

line = re.sub('[^a-zA-Z0-9 \.]', '', line)
```

It will remove everything except alphanumeric characters, space and full-stop.
Convert the text to lowercase and use unique words to create the vocabulary.

• Report:
  - Vocabulary size
  - 10 most frequent and 10 least frequent words

To create X, and y pairs for training, you can use a similar approach used for next-character prediction. For example:

```
. . . . . ---> to
. . . . to ---> sherlock
. . . to sherlock ---> holmes
. . to sherlock holmes ---> she
. to sherlock holmes she ---> is
to sherlock holmes she is ---> always
sherlock holmes she is always ---> the
holmes she is always the ---> woman
she is always the woman ---> .
```

You will get something like *". . . . . ---> to"* whenever there is a paragraph change.

1.2 Model Design and Training [1 marks]

Build an MLP-based text generator with the following structure:

- Embedding dimension: 32 or 64
- Hidden layers: 1–2 (1024 neurons each)
- Activation: ReLU or Tanh
- Output: Softmax over vocabulary

Use Google Colab or Kaggle for training (use maximum 500-1000 epochs). Start the assignment early, as training takes time.

Report in notebook:

- Training vs validation loss plot
- Final validation loss/accuracy
- Example predictions and commentary on learning behavior.

1.3 Embedding Visualization and Interpretation [1 mark]

Visualize the embeddings using t-SNE if using more than 2 dimensions or using a scatter plot if using 2 dimensions and write your observations. For visualizations, you may have to select words with relations like synonyms, antonyms, names and pronouns, verbs and adverbs, words with no relations, and so on. Discuss your observations on clustering patterns and semantic relationships.

1.4 Streamlit Application [1.5 marks]

Write a streamlit application that asks users for an input text, and it then predicts the next k words or lines. In the streamlit app, you should have controls for modifying context length, embedding dimension, activation function, random seed, etc. You can use any one of the datasets mentioned. Incorporate temperature control in your streamlit app to control the randomness of predicted words. Refer to this article.

Think how you would handle the case where words provided by the user in the streamlit app are not in the vocabulary. There is no need to re-train the model based on the user input. Train two to three variants and accordingly give options to the user.

1.5 Comparative Analysis [1 mark]

• Compare your two trained models (Category I vs Category II):

- Dataset size, vocabulary, context predictability
- Model performance (loss curves, qualitative generations)
- Embedding visualizations

• Summarize insights on how natural vs structured language differs in learnability.

Datasets:

a. Category I
   i. Paul Graham essays
   ii. Wikipedia (English)
   iii. Shakespeare
   iv. Leo Tolstoy's War and Peace
   v. The Adventures of Sherlock Holmes, by Arthur Conan Doyle

b. Category II
   i. Maths texbook

ii. Python or C++ code (Linux Kernel Code)
iii. IITGN advisory generation
iv. IITGN website generation
v. Generate sklearn docs
vi. Notes generation
vii. Image generation (ascii art, 0-255)
viii. Music Generation
ix. Something comparable in spirit but of your choice (do confirm with TA Neerja)

2. Moons Dataset & Regularization [3 marks]

Generate Make-Moons dataset without using sklearn make_moons. Use default noise 0.2, also create two extra test sets with noise 0.1 and 0.3 for robustness reporting. Make training set and test set with 500 points each. Standardize x after the split using train statistics only. Create a validation split of the train set with 20 percent for model selection. Use random seed 1337.
Train the following models:
1. MLP with hidden layer - early stopping (patience=50)
2. MLP with L1 regularization . L1 gird $\lambda \in \{1e{-}6, 3e{-}6, 1e{-}5, 3e{-}5, 1e{-}4, 3e{-}4\}$. Report layerwise sparsity and validation AUROC vs. λ
3. MLP with L2 regularization (you may vary the penalty coefficient by choose the best one using a validation dataset)

4. Logistic regression with polynomial features (x_1x_2, x_1^2, etc.)

Evaluation and Analysis

• Evaluate test accuracy on noise = 0.20, and robustness accuracy on 0.10 & 0.30.
• Create a table with test accuracy for the four models on the three test noise levels. Include parameter count.

• Plot decision boundaries side by side for all 4 models with default noise 0.2.
• Discuss:
  - Effect of L1 on sparsity and boundary jaggedness
  - Effect of L2 on smoothness and margin
• Add class imbalance (70:30) in the trainset while keeping the testset balanced. Report accuracy and AUROC and discuss the effect of imbalance.

3. MNIST and CNN Experiments [3 marks]

This section explores deep learning for images. You will train MLPs and CNNs on MNIST, compare performance against baseline models, visualize embeddings using t-SNE, and test cross-domain generalization on Fashion-MNIST.

3.1 Using MLP [1.5 marks]

Train on MNIST dataset using an MLP. The original training dataset contains 60,000 images and the test contains 10,000 images. If you are short on compute, use a stratified subset of a smaller number of images but keep the same test set. Your MLP has 30 neurons in the first layer, 20 in the second layer and then 10 finally for the output layer (corresponding to 10 classes)

Report the following:
• Compare against Random Forest and Logistic Regression. The metrics can be: accuracy, F1-score, confusion matrix. Write your observations and discuss misclassifications.
• Visualize t-SNE of the 20-neuron layer for the 10 digits for the trained and untrained model and compare the two.
• Test the trained MLP on Fashion-MNIST dataset. What do you observe? Compare t-SNE plots for MNIST and Fashion-MNIST embeddings for the layer with 20 neurons.

3.2 Using CNN [1.5 marks]

• Implement a simple CNN with a convolutional layer having 32 filters of size 3x3, a maxpool layer, a fully connected layer with 128 neurons and an output layer with 10 neurons (for the 10 classes) and ReLU activation. Train on MNIST dataset.
• Additionally, use two any pretrained CNNs of your choice (e.g. AlexNet, MobileNet, or EfficientNet) for inference.
• Compare all three models:
    - Accuracy, F1-score, confusion matrix
    - Model size (number of parameters)
    - Inference time on test set

Submission Format: Share a GitHub repo with your training notebooks named "question<number>.ipynb". Include textual answers in the notebook itself. For Question 1, put the link to the streamlit app at the top of the notebook.