Assignment 2 Instruction

Overview

The goal of this assignment is to consolidate your understanding of fundamental deep learning models. In particular, this task focuses on a classical neural network architecture known as the **Multi-Layer Perceptron (MLP)** for predicting diabetes using the provided dataset. You will be required to perform data preprocessing, model training, and evaluation on a real-world dataset, applying the key concepts learned in the first four modules.

Background

Deep learning is built upon the foundation of the **perceptron**, a simple computational model of a biological neuron. A perceptron takes multiple input features, applies weights and a bias, computes a linear combination, and then passes it through an activation function to produce an output. Mathematically, it can be expressed as:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

where x_i are the inputs, w_i the weights, b the bias term, and $f(\cdot)$ an activation function (e.g., sigmoid or ReLU).

An **MLP** extends this idea by stacking multiple layers of perceptrons, typically consisting of an input layer, one or more hidden layers, and an output layer. This structure allows the model to capture complex, nonlinear relationships in data. Figure 1 below illustrates a simple MLP architecture.

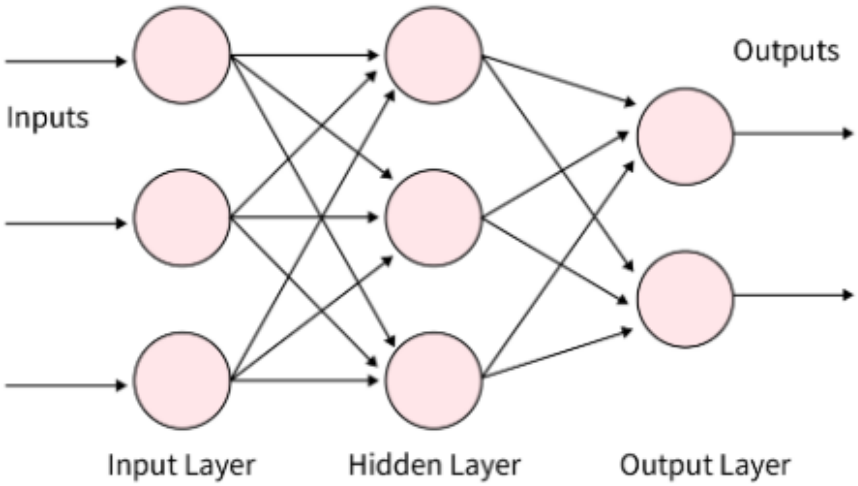

Figure 1: A simple Multi-Layer Perceptron (MLP) architecture with one hidden layer.

You will be asked to prepare a technical report that demonstrates the task, model architectures, and experimental analysis, limited to **three pages excluding references**. This report should clearly demonstrate your understanding of the dataset, model, and experimental results without additional materials.

In addition, this assignment also evaluates your programming skills in terms of reproducibility, code readability, and maintainability. You will submit your implementation in a Jupyter Notebook that contains all relevant results, including plots and analyses related to data and model evaluation. All results reported in the paper should be traceable to your code.

Resources

- A folder of dataset with train and test set
- Jupyter Notebook Template for model implementation.

Preliminary Knowledge

This assignment builds upon the materials covered in the first four weeks of lectures and workshops, including:

- **Module 1:** Introduction
- **Module 2:** Supervised Learning
- **Module 3:** Convolutional Neural Networks
- **Module 4:** Training Deep Networks

Students are expected to have a basic understanding of these topics before starting the assignment, as they provide the essential theoretical and practical foundation for implementing and analyzing deep learning models.

Task 1: Technical Report (15%)

We highly recommend you to present your report using conference LaTeX toolkits such as CVPR (two-column style) or ICLR (one-column style). You may also want to use Overleaf, a free online LaTeX editor, to edit and compile your LaTeX documents. It is completely fine to use Microsoft Word or other editors that may work for you. There is no specific format required in this assignment.

Suggested Structure. Although no strict format is required, we recommend organizing your report in a clear and coherent structure similar to standard research papers. A suggested outline is as follows:

- **Introduction:** Clearly state the problem, motivation, and the goal of your work. Summarize what is being done and why it matters.
- **Related Work (optional):** Briefly mention any existing methods, references, or concepts that are relevant to your approach.
- **Method:** Describe your model design, algorithm, and implementation details. Focus on explaining your reasoning and model structure rather than generic deep learning theory.

- **Experiments:** Present your setup, dataset usage, and results. Include comparisons between different hyperparameter settings or model variants, and interpret your findings.
- **Conclusion:** Summarize your main findings, insights, and any possible improvements or limitations.

An abstract is **not required** for this assignment. The report should read as a concise, technical summary of your implementation and analysis, demonstrating both your understanding and ability to communicate experimental findings effectively.

Your paper must be all your own work, with no text copied from any other document. All non-original works and statements must be cited by any widely used reference format (e.g. APA or Harvard). The paper is limited up to **3-pages long**, including figures and tables and excluding references. No appendix is allowed. Marks deduction will be applied for any exceeding page (1 mark per page). You will submit only one pdf for your report part to GradeScope, link provided in Assignment 2 page.

Introduction (3%)

Provide a description of the problem and the proposed method or algorithm, citing relevant sources (papers or reputable websites) where appropriate. Include a clear overview of the background and motivation for the task, along with a concise summary of the methodology and experimental analysis. Conclude with the key insights gained from the analysis. All sources should be properly cited using any widely recognized reference format (e.g., APA, Harvard).

Method (6%)

Provide a detailed description of the network. This should include an explanation of the algorithm (or relevant components), with examples that illustrate its effects and limitations. If you propose an improvement, clearly explain how your method works, with sufficient detail for a reasonably skilled person to be able to implement it.

Experimental Analysis (6%)

Briefly describe the experiments you conducted and explain the motivation behind them. **You should design and compare at least three different experimental settings based on varying hyperparameters of your model**, such as the number of hidden layers, the number of neurons per layer, learning rate, or feature selection strategies. The goal is to analyze how these variations affect model performance and to provide insights into the strengths and limitations of your chosen configuration.

The objective is not necessarily to achieve the highest accuracy, but to demonstrate a clear understanding of how different hyperparameter choices influence the learning process and outcomes. This section should also include your interpretation of the results and justification for the final configuration you selected.

Task 2: Code (5%)

In this section, students will be assessed on their ability to prepare data, implement a deep learning model from scratch, and produce reproducible experiments, while ensuring that their code is well-structured and of high quality. Readability and clarity of the code are essential. All results reported must be traceable to the code implementation, but it is not necessary to include additional results or plots within the code itself, as these will not contribute to the report. All analysis and discussion should be presented in the report rather than embedded in the code.

Preprocessing (2%)

You will be asked to implement appropriate techniques to visualize and preprocessing techniques such as data cleaning / feature selection, data split. Brief visualization is encouraged to help on understanding of dataset, showing the insights of data (statistics, feature correlation and so on).

Model Implementation (2%)

The proposed method should be implemented using Python principles with a clear and modular structure. You are expected to construct your model using deep learning frameworks such as **PyTorch** or **TensorFlow**. Re-implementing low-level operations (e.g., backward propagation or gradient computation) manually is **not required**.

You may freely use standard **building blocks** from these libraries, such as:

```
nn.Linear, nn.ReLU, nn.Sigmoid, nn.Sequential,
torch.optim.Adam, torch.utils.data.DataLoader
```

However, your model architecture should be defined by yourself (e.g., creating a subclass of `nn.Module`), and you **must not** load or fine-tune a pre-trained model. The following illustrates the distinction:

```
# Allowed: using library layers and optimizers
nn.Linear(...), nn.ReLU(), torch.optim.Adam(...)

# Not allowed: using pretrained models
model = torchvision.models.resnet18(pretrained=True)
```

The quality of your implementation will be evaluated based on **clarity, readability, and maintainability**. The code structure should align with the descriptions in your report, and all reported results must be reproducible from the provided implementation.

Experiments (1%)

This section requires to align with report. The results in the code should be able to reproduce and show evidence that the result of report is obtained from actual program. You should not have additional comments that explain this your results, all results in experiments should be displayed in report. The experiment setting and clear experiments structural is encouraged to show.