In the name of God

University of Tehran

School of Electrical and Computer Engineering

Neural Networks and Deep Learning Course

Assignment 1

Contents

1 Rules

Before answering the questions, please read the following items carefully:

- Prepare a report of your answers in the format provided on the course page in the Elearn system, named `REPORTS_TEMPLATE.docx`.
- It is recommended to do the exercises in groups of two. (More than two people are not allowed, and individual submissions do not receive extra credit). Note that there is no requirement for group members to remain the same throughout the semester. (i.e., you can do the first assignment with person A and the second assignment with person B, etc.).
- The quality of your report is of great importance in the grading process; therefore, please mention all the points and assumptions you considered in your implementations and calculations in the report.
- In your report, according to what is provided in the sample template, consider captions for figures and supercaptions for tables.
- It is not necessary to provide detailed explanations of the code in the report, but the results obtained from it must be reported and analyzed.
- Analysis of the results is mandatory, even if not explicitly asked for in the question.
- The teaching assistants are not obliged to run your codes; therefore, any results or analyses requested from you in the questions should be clearly and completely stated in the report. In case of non-compliance with this point, it is obvious that points will be deducted from your grade.
- The codes must be prepared in a notebook with the `.ipynb` extension. At the end of the work, all the code must be run, and the output of each cell must be saved in your submitted file. Therefore, for example, if the output of a cell is a plot that you have included in the report, this plot must also be present in the report and in the code notebook.
- In case of cheating, the score of all participants involved will be marked as -100.
- The only authorized programming language is **Python**.
- The use of pre-written codes for the exercises is by no means permissible. If two groups use a common source and submit similar codes, it will be considered cheating.
- The method of calculating lateness is as follows: After the submission deadline, it is possible to submit with a delay for up to one week. After this one week, the score for that assignment will be zero for you.
  - First three days: No penalty
  - Fourth day: 5% penalty
  - Fifth day: 10% penalty
  - Sixth day: 15% penalty

  - Seventh day: 20% penalty

- The maximum score that can be obtained for each question is 100, and if the total points for a question are more than 100, if a score higher than 100 is obtained, it will not be applied.
  - For example, if the score obtained from question 1 is 105 and the score for question 2 is 95, the final score for the assignment will be 97.5 and not 100.
- Please place the report, codes, and other attachments in a folder with the following name, compress it, and then upload it to the Elearn system: `HW[Number]_[Lastname]_[StudentNu` (Example: `HW1_Ahmadi_810199101_Bagheri_810199102.zip`)
- For groups of two, it is sufficient for one member to upload the assignment, but it is recommended that both members upload it.

2 Question 1: Credit Card Fraud Detection using Multilayer Perceptron (MLP)

2.1 Introduction

In this exercise, you will design, implement, and evaluate a Multilayer Perceptron (MLP) neural network for detecting fraudulent transactions. First, the transaction data will be examined and preprocessed, then different MLP models will be designed and their performance analyzed. In addition, the confusion matrix and evaluation metrics such as accuracy, recall, F1-score, and AUC-ROC will be calculated and examined to measure the model's effectiveness. Finally, by searching for hyperparameters, the optimal settings for the model will be selected, and the performance difference between the models will be compared and analyzed.

2.2 Preprocessing and Data Exploration (10 points)

1. Download the Credit Card Fraud Detection dataset from Kaggle and load it in Google Colab.
   Dataset Link: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud
2. Display a summary of the data.
3. Plot a bar chart of the class distribution to identify the data imbalance.
4. Explain why class imbalance is challenging for modeling.
5. Normalize the features using StandardScaler or MinMaxScaler.
6. Split the dataset into 70% training and 30% testing sets (while preserving the class balance).

2.3 Designing and Implementing a Simple MLP Network (20 points)

1. Design a neural network with one hidden layer:
   - Input: The number of features should be equal to the input data.
   - Hidden Layer: With 64 neurons and ReLU activation function.
   - Dropout: After the hidden layer, once without dropout and once with a dropout rate of 30%.
   - L2 Regularization: With $\lambda = 0.0001$, once using regularization and once without.
   - Output Layer: One neuron (you can use the sigmoid activation function).
2. Model Training:
   - Loss Function: Your choice (you can use Binary Cross-Entropy).
   - Optimizer: Adam.
   - Batch Size: 32.

- Number of Epochs: 40.
- Plot the changes in accuracy and loss during training.

3. Evaluation:
   - Display and analyze the confusion matrix.
   - Calculate and analyze Accuracy, Precision, Recall, and F1-Score.
   - Plot the ROC curve and the AUC value.

2.4 Designing a Deeper MLP Neural Network (20 points)

1. Design an MLP network with two hidden layers:
   - First Hidden Layer: With 128 neurons and ReLU.
   - Second Hidden Layer: With 64 neurons and ReLU.
   - Dropout: 20% after each hidden layer.
   - L2 Regularization: With $\lambda = 0.0001$.
2. Model Training and Evaluation: Same as the previous section.
3. Compare the results of this model with the model from the previous section.

2.5 Analysis of Confusion Matrix and Evaluation Metrics (15 points)

- **Accuracy:** What does it measure? Why is it not a suitable metric for imbalanced data?
- **Precision:** Why is it important in fraud detection?
- **Recall:** What does a high or low value mean?
- **F1-score:** Why is it better than simple accuracy?
- **ROC Curve and AUC:** What does the AUC value indicate?
- In the confusion matrix, which class is more often misclassified? (State for both models from the previous sections).
- What is the trade-off between precision and recall?

2.6 Grid Search for Hyperparameter Tuning (20 points)

Perform a Grid Search or Random Search for the following settings of a single hidden layer network:

- Number of neurons in the hidden layer: 64, 128, and 256.
- Dropout rate: 0.2, 0.3, and 0.4.
- Regularization: 0.001 and 0.0001.
- Batch size: 16, 32, and 64.

Find the best combination, train it, and evaluate the model.

2.7 Comparison of MLP with Logistic Regression (10 points - Bonus)

1. Train a logistic regression model on the same data.
2. Compare the results of this model with the best MLP model.
3. Analyze your findings:
    - Is deep learning significantly better than logistic regression?
    - Under what conditions might logistic regression be better?

2.8 Conclusion (15 points)

- Which model had the best performance and why?
- What was the effect of adding more layers on the model's performance?
- How did hyperparameter optimization affect the model's outcome?
- What was the model's error rate and what was its main cause?
- Based on the confusion matrix, which class did the model misclassify more?
- What is the relationship between precision and recall, and why is it important in fraud detection?
- Model Comparison: Did the more complex model perform better, or did it just increase computations?
- What methods are suggested for improving the model's performance?
- What are the most important challenges in real-world fraud detection data?
- If you were to design the model again, what changes would you make?

3 Question 2: Designing a Multilayer Perceptron for Concrete Compressive Strength Regression

3.1 Introduction

In this problem, students will design a Multilayer Perceptron (MLP) to predict the compressive strength of concrete and investigate the effect of changing various settings such as the number of epochs, optimizer, and loss function on the model's performance. In addition, they will analyze the statistical features of the data to gain a better understanding of the relationships between inputs and outputs.

3.2 Data Preparation and Statistical Analysis (30 points)

1. Obtain the Concrete Strength Dataset from the link below.
    - Dataset Link
2. Data Exploration:
    - Display a statistical summary (mean, standard deviation, max, and min for each feature).
    - Plot a histogram for each feature.
3. Investigate the correlation of features with concrete strength:
    - Plot the correlation matrix.
    - Plot a scatter plot for key features.
4. Which feature has the highest correlation with concrete strength?
5. Are there features that have a strong correlation with each other?
6. Can some features be removed based on this analysis? Why?
7. What happens if features with low or high correlation are removed?

3.3 Implementation of the Multilayer Perceptron Model (25 points)

Define a fixed MLP model with the following structure:

- **Input Layer:** Proportional to the number of data features.
- **One Hidden Layer:** Train with 16 and 32 neurons and report the better model at the end of this section.
- **Output Layer:** One neuron with a numerical value.

Train the model with the following settings:

- **Loss Function:** Mean Squared Error (MSE).

- **Optimizer:** Adam.
- **Number of Epochs:** 50.

Model Evaluation:

- Display MSE and MAE on the test data.
- Plot the graph of changes in error and accuracy.

3.4 Investigating Changes in Model Settings (30 points)

In this section, first select the better model from the previous section. Then, investigate the following items.

1. **Effect of the number of epochs:**
    - Train the model for 20, 50, and 100 epochs.
    - Does increasing the number of epochs always lead to better results?
2. **Comparison of loss functions:**
    - Train the model with MSE, MAE, and Huber Loss functions.
    - Which one performed better? Why?
    - Briefly explain each one.
3. **Comparison of optimizers:**
    - Train the model with SGD, Adam, and RMSprop optimizers.
    - Show the performance difference of each and analyze it using the error and accuracy plot.

3.5 Conclusion (15 points)

Analyze the obtained results, discuss the reasons for the differences one by one, and answer the following questions.

- Which feature has the most impact on concrete strength? Is this feature logical?
- What were the best settings for this model?
- Did increasing the number of epochs always improve the model?
- Which loss function had better accuracy? Why?
- Among the optimizers, which one converged faster?
- What were the main challenges of regression modeling with a neural network?

4 Question 3: Implementing Adaline for the IRIS Dataset

4.1 Introduction

The purpose of this exercise is to become familiar with the Adaline (Adaptive Linear Neuron) model and Madaline as one of the early structures of neural networks and to understand how it works in classification problems. In this exercise, you will implement the Adaline model and test it on a subset of the Iris dataset for classifying two classes, Setosa and Versicolor. Additionally, an analysis of the model's performance will be provided.

4.2 Introduction to Adaline (15 points)

Briefly answer the following questions:

1. Explain the Adaline and Madaline algorithms.
2. Explain the main difference between Madaline and MLP.

4.3 Data Preparation (10 points)

In this section, prepare the Iris dataset for binary classification:

1. **Download Dataset:**
    - Load the Iris dataset from `sklearn.datasets.load_iris`.
2. **Select Subset:**
    - Keep only the two classes Setosa and Versicolor (remove the Virginica class).
    - Use the features `petal length` and `petal width`.
3. **Data Preparation and Splitting:**
    - Normalize the data (to the range [0, 1]).
    - Split the data into two parts: training (70%) and testing (30%).

4.4 Implementation and Training of the Adaline Model (40 points)

In this section, implement the Adaline algorithm and train it with the training data:

1. **Adaline Algorithm Implementation:**
    - Write a function in Python that implements the Adaline algorithm.
    - **Function Inputs:**
        - Network dimensions (number of neurons in the input and output layers).
        - Learning rate.
        - Number of epochs.

- **Function Outputs:**
  - Weights and biases at each epoch.
  - Error at each epoch.
  - Accuracy on the training data at each epoch.

2. **Training the model with different learning rates:**
   - Train the algorithm implemented in the previous section with three learning rates: 0.02, 0.005, and 0.001. Initialize the weights randomly and train each of the models for up to 10 epochs.
3. **Saving Results:**
   - Save the error and accuracy for all three models and for each epoch.

4.5 Display and Analysis of Results (35 points)

Present the results visually and analytically:

1. **Display the separating line:**
   - Plot the separating lines for each of the three models on the training and test data in the petal length and petal width space. (In total, one plot for the training data and one plot for the test data).
2. **Error and Accuracy Plot:**
   - Plot two graphs:
     - Error vs. epoch for each of the models.
     - Accuracy vs. epoch for each of the models.
3. **Results Analysis:**
   - Answer the following questions:
     - Did all three models converge? If not, what were the contributing factors?
     - What is the effect of linear separability of data on Adaline's performance? How will this algorithm perform if the data is not linearly separable?
     - Analyze the effect of the learning rate on the model's performance.

5 Question 4: Autoencoder Training and Classification with the MNIST Dataset

5.1 Introduction

In this exercise, you will train an Autoencoder to learn a compressed representation of handwritten digit images from the MNIST dataset. Then, you will use its encoder part along with a feed-forward layer to classify the digits (0 to 9). The goal is to investigate the effect of the latent space size on image reconstruction and classification accuracy.
The MNIST dataset consists of 70,000 handwritten digit images (60,000 training and 10,000 test) of digits from 0 to 9. Each image is 28x28 pixels (784 features) and is black and white.

5.2 Downloading and Preprocessing Data (5 points)

Load the training and test data and preprocess them. This dataset is accessible via TensorFlow (`tf.keras.datasets.mnist`) or PyTorch (`torchvision.datasets.MNIST`).

- Normalize the pixel values from [0, 255] to [0, 1].
- Convert the images to 784-dimensional vectors.

5.3 Model Design and Implementation (60 points)

In this section, two autoencoder models with different latent space sizes are designed and trained.

Part One: Autoencoders Implement and train both autoencoder models on the training data.

- **Model One:** Autoencoder with an encoder output of 8 neurons.
  - **Architecture** (number of neurons and activation function in each layer are specified in order):
    - **Encoder:** 784, 128 (ReLU), 8 (ReLU).
    - **Decoder:** 8, 128 (ReLU), 784 (Linear).
  - **Training Settings:**
    - Loss Function: MSE.
    - Number of training epochs: at least 20.
- **Model Two:** Autoencoder with an encoder output of 4 neurons.
  - **Architecture:**
    - **Encoder:** 784, 128 (ReLU), 4 (ReLU).
    - **Decoder:** 4, 128 (ReLU), 784 (Linear).
  - **Training Settings:** Same as model one.

Part Two: Classification with the Encoder

- Combine the encoders trained in the previous section with a feed-forward layer and train on the training data with labels.
  - The encoder trained in the previous section should be frozen.
  - **Feed-forward layer:**
    - For the 8-neuron model: 8, 4 (ReLU), 2 (Softmax).
    - For the 4-neuron model: 4, 2 (Softmax).
  - **Training Settings:**
    - Loss Function: Cross-Entropy Loss.
    - Number of training epochs: at least 20.

Note that in the second part, the encoder is frozen, and its weights do not change.

5.4 Results and Analysis (40 points)

1. **Results:**
   - Plot the reconstruction error (MSE) of the autoencoders on the training data versus the epoch.
   - Calculate the accuracy of each model (autoencoder + classifier) on the training and test data and plot its graph versus the epoch.
   - Report the number of parameters of each model (encoder + decoder and encoder + classifier).
2. **Analysis and Comparison:**
   - **Reconstruction Error Comparison:** Did the model with 8 neurons reconstruct better or the one with 4 neurons? Why?
   - **Classification Accuracy Comparison:** Which latent space size (8 or 4) had better performance? Did reducing the dimension too much harm the accuracy?
   - **Parameter Count Comparison:** What is the effect of the number of parameters on performance?
   - Did you observe any signs of overfitting? How can it be reduced?
   - **(5 points bonus)** Based on the obtained results, propose a new model to improve performance and train it. For example, you can change the autoencoder architecture.