Iran University of Science & Technology

School of Computer Engineering

Assignment #2

Neural Networks

BY: DR. Naser Mozayani, Fall 2024

Teaching Assistants:

Mehdi Feghhi

Due: 1403/08/18

Contents

Notes

1. Submit the answers in a complete PDF file and the code for the questions in the .ipynb format (including the notebook cell outputs) in a compressed file named HW2_StudentID.zip by the specified deadline.
2. A total of 72 hours of delay in submitting the answers is allowed across all projects. After that, for each additional day of delay, 10% of the score will be deducted.
3. If a student submits the project earlier than the deadline and achieves 75% of the score, up to 24 hours will be added to their allowable delay time.
4. The maximum delay for submitting each assignment is 4 days, and after 4 days, submission will not be accepted.
5. It is important to note that the explanation of the code and the obtained results must be included in the PDF file. Code without a report will result in a score deduction.
6. The evaluation of the assignment will be based on the correctness of the solution and the completeness and accuracy of the report.
7. Assignments must be completed individually, and group work on assignments is not allowed.
8. Please allocate sufficient time for the assignment and avoid leaving it until the last days.
9. You can ask your questions in the relevant group.

good luck.

Problem 1

Understanding Overfitting and Underfitting in Neural Networks. (15 points)

Based on the knowledge acquired from the course, answer the following questions:

1. What is Overfitting in Neural Networks?

Explain the concept of overfitting in neural networks, including how it impacts the model's ability to generalize to new data.

2. Identifying Overfitting in Neural Networks

When can we say that a neural network is overfitting? Discuss this in the context of the different networks you have studied and provide a general analysis.

3. Methods to Prevent Overfitting

What are some techniques used to prevent overfitting in neural networks? Offer detailed explanations of each method.

4. What is Underfitting in Neural Networks?

Explain the concept of underfitting and how it occurs in neural networks.

5. Methods to Address Underfitting

What are the suggested approaches to handle underfitting in neural networks? Explain these methods in detail.

Problem 2

Explore how to manipulate various parameters in neural networks to achieve **overfitting** and **underfitting**. **(15 points)**

Instructions:

1. Visit the following link and use the TensorFlow Playground tool to conduct experiments:

Playground TensorFlow

2. Experiment with different datasets and **increase the noise** to challenge the classification task.
3. Adjust the following parameters to explore conditions that lead to **overfitting** and **underfitting**:
    - **Number of layers** and **neurons** (increase complexity to explore overfitting)
    - **Activation functions**
    - **Learning rate**
    - **Number of epochs**
    - **Regularization rate** (experiment with low and high values)
4. Pay close attention to how your changes affect the model's performance:
    - Identify and explain instances of **overfitting** (where the model performs well on training data but poorly on test data).
    - Identify and explain instances of **underfitting** (where the model fails to capture the underlying pattern in the data).
5. **Observe the effect of regularization** (e.g., L2 regularization) in controlling overfitting and how it affects the model's performance.
6. **Take screenshots** of your results to illustrate the impact of different configurations on model performance.
7. Prepare a report summarizing your findings, including the screenshots to support your observations.

Deliverables:

- A comprehensive report detailing your experiments and results, accompanied by visual evidence in the form of screenshots.

Problem 3

In this assignment, you will delve into the fundamentals of machine learning by training a simple linear neural network on the Fashion MNIST dataset. Your task is to not only implement the network but also to analyze its performance through a series of training metrics, including accuracy, loss, and confusion matrix. **(30 points)**

1. **Data Preparation:** Start by exploring the Fashion MNIST dataset. Provide a brief description of the dataset and visualize some of the samples. Ensure you preprocess the data appropriately for training.
2. **Model Implementation:** Design and implement a simple linear neural network. Explain your choice of architecture and any relevant activation functions.
3. **Training and Evaluation:** Train the model and evaluate its performance using metrics such as accuracy, precision, recall, and F1-score. Discuss the significance of each metric in the context of your model's performance.
4. **Hyperparameter Tuning:** Explore different hyperparameters, including learning rate, batch size, and number of epochs. Utilize techniques such as grid search or random search to systematically find the optimal parameters. Document the impact of these changes on your model’s performance.
5. **Reflection:** Reflect on the results obtained through hyperparameter tuning. Discuss how different hyperparameters affected the training process and the final model performance. What insights can you draw from this experiment regarding the importance of hyperparameter tuning in machine learning?

Provide a comprehensive report summarizing your process, findings, and any challenges you faced during the assignment. Use visual aids where appropriate to enhance the clarity of your explanations.

Problem 4

Color Clustering Using SOFM. (20 points)

In this problem, the goal is to cluster four different colors using the Self-Organizing Map (SOFM) algorithm. The input colors are defined as follows:

1. Red (255, 0, 0)
2. Green (0, 255, 0)
3. Blue (0, 0, 255)
4. Yellow (255, 255, 0)

Steps to Solve the Problem

1. **Initialization:**
    - Create a SOFM with a size of 2x2. This map has 4 nodes.
    - Choose the initial weights for each node as follows:
        - Node 1: (100, 100, 100)
        - Node 2: (150, 150, 150)
        - Node 3: (50, 50, 50)
        - Node 4: (200, 200, 200)
2. **Competition:**
    - For each input color, calculate the Euclidean distance from the input color to each node and identify the winning node.
3. **Cooperation:**
    - After identifying the winning node, update the weights of the winning node and the weights of the neighboring nodes.
    - Assume the learning rate (η) = 0.1 and the neighborhood radius = 1.
4. **Synaptic Adaptation:**
    - Repeat steps 2 and 3 for all input colors.

Questions:

1. What are the final weights of each node after clustering the colors using the SOFM algorithm?
2. Did the weights reach a stable state? If so, what are the characteristics of this stable state?
3. Based on the results obtained, which nodes represent specific colors, and how can they be used in color clustering?

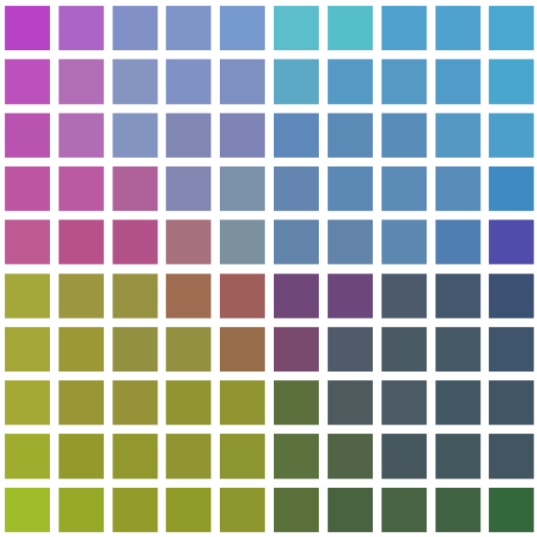
Problem 5

Implement a Self-Organizing Map (SOM) in Python to cluster and visualize a dataset of random colors. This task aims to demonstrate how the SOM organizes colors based on their RGB values and captures their underlying relationships. **(30 points)**

Requirements:

1. **Dataset Creation:**
    - Generate a dataset of exactly 256 random RGB colors. Each color should be represented as an array of three values corresponding to the red, green, and blue channels, with each value ranging from 0 to 1.
2. **SOM Initialization:**
    - Create a Self-Organizing Map with a fixed number of neurons: 100 neurons, arranged in a 10x10 grid.
    - Initialize the weights of the SOM randomly within the range [0, 1] for each RGB channel.
3. **Training Parameters:**
    - Set the learning rate to 0.1.
    - Train the SOM for exactly 500 epochs.
4. **Training Process:**
    - **For each color in your dataset, implement the following steps:**
        - Find the Best Matching Unit (BMU): Calculate the Euclidean distance between the input color and each neuron's weight, identifying the neuron with the smallest distance.
        - Update the BMU and its neighbors: For the identified BMU, update its weights and the weights of neighboring neurons. Use a fixed neighborhood size, considering a neighborhood radius of 5.
        - Use a weight update formula that incorporates the learning rate and the distance to adjust the weights of the BMU and its neighbors.

5. **Visualization:**
    - After training, visualize the learned SOM by creating a grid of colors. Each neuron should display its learned color in a 10*10 subplot layout using Matplotlib, similar to the example below:



Analysis:

- **Once the SOM is visualized, analyze the resulting color patterns:**
    - Discuss the organization of colors and how they relate to one another.
    - Identify any noticeable groupings or gradients that indicate relationships among the colors.
    - Reflect on how effectively the SOM has captured the structure of the color dataset based on the final visualization. Consider aspects like color clustering and distribution.

Sample Libraries to Use:

- **NumPy:** For numerical operations and array handling.
- **Matplotlib:** For creating visualizations of the SOM.

Expected Output:

- A grid of colors representing the learned weights of the SOM after training, along with a brief analysis of the visualized results.