CSCI 5996 Special Topics: Fundamentals of Deep Learning

Assignment 2

Gradient Descent and Backpropagation

1. Objective

- Understand how backpropagation applies to regression problems.
- Derive gradients for a neural network with a continuous-valued output.
- Implement a neural network from scratch (without deep learning libraries).
- Train and evaluate a model on a simple regression dataset.

This is an individual assignment.

You can consider using Generative AI (GenAI) (e.g., Copilot with Visual Studio Code or Gemini in Google Colab) to help you write code faster. Please double-check the generated code carefully to make sure it produces what you want. If you use GenAI for a specific question, please acknowledge that at the beginning of your code for that question and specific the GenAI tool used, its version, and a brief description of how it was utilized to generate the code (e.g., # Use Copilot with GPT 4o to generate initial code for the question).

2. Requirements

2.1 Conceptual Questions

1) (5 pts) Forward Pass for Regression. Consider a neural network with:
    - Input: 1 feature
    - Hidden layer: 5 neurons (sigmoid activation)
    - Output layer: 1 neuron (linear activation, no sigmoid)

    Write the mathematical expressions for the hidden activations and the output.

2) (15 pts) Loss Function.
    For regression, the sum-of-squares error function is used:
    $$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2$$
    Derive the gradient of $E(\boldsymbol{w})$ with respect to the weights of both the hidden and output layers.
3) (10 pts) Explain step by step how the chain rule is used to propagate errors from the output back to the hidden layer in this regression setup.

2.2 Implementation Task

1) (5 pts) Generate 100 data points uniformly spaced in [-1,1]. Target values: $y = \sin(2\pi x) + \epsilon$, where $\epsilon \sim N(0,0.01)$ is small Gaussian noise. Split the dataset into 80% training and 20% testing.
2) Implement the neural network in 2.1.1 and with the loss in 2.1.2:

a) (5 pts) Initialize weights and biases randomly.
b) (10 pts) Implement the **forward pass**.
c) (10 pts) Compute the error using the sum-of-squares error function.
d) (10 pts) Derive and implement **backpropagation** manually.
e) (10 pts) Update weights and biases using **gradient descent**.
f) (5 pts) Train the model for at least 3000 epochs.
g) Plot the following:
    a. (7.5 pts) Training loss vs epochs.
    b. (7.5 pts) Predictions vs true values on the test set.

2.3 Other requirements

- Your Python code should be written for Python version 3.5.2 or higher.
- Please write proper comments in your code to help the instructor and teaching assistants to understand it.
- Please properly organize your Python code (e.g., create proper classes, modules).
- You can put your code to Jupyter Notebook or a .py file.

3. Submission instructions

Put all your files (Python code, readme file, report, etc.) to a zip file named hw.zip and upload it to Canvas.

4. Grading criteria

- **ZERO** point will be given if your code does not work. Please do not submit code that you did not test and make sure it works.
- FIVE points will be deducted if files are not submitted in the required format.
- If the total points are more than 100. Your grades will be scaled to the range of [0,100].
- Make sure that you test your code thoroughly by considering all possible test cases. Your code may be tested using more datasets.