CSE 4392 Special Topic: Natural Language Processing

Homework 6 - Spring 2024

Due Date: Mar 19th, 2024, 11:59 p.m. Central Time

Welcome to our exploration of Neural Network Basics. This assignment takes you on a journey from understanding the core principles of neural network architecture to hands-on experiences, such as manually calculating gradients for parameter optimization. We'll begin by solidifying your grasp of the basics and progress to the heart of the matter – computing gradients and formulating the update equation.

Problem 1 - 40%

In this task, you are provided with a simple fully connected neural network represented by figure 1. Your objective is to delve into the mathematical intricacies of backpropagation and calculate the gradients for each parameter in the network. The neural network consists of interconnected layers, and your task involves understanding the chain rule and applying it meticulously to determine how changes in each parameter affect the overall loss.

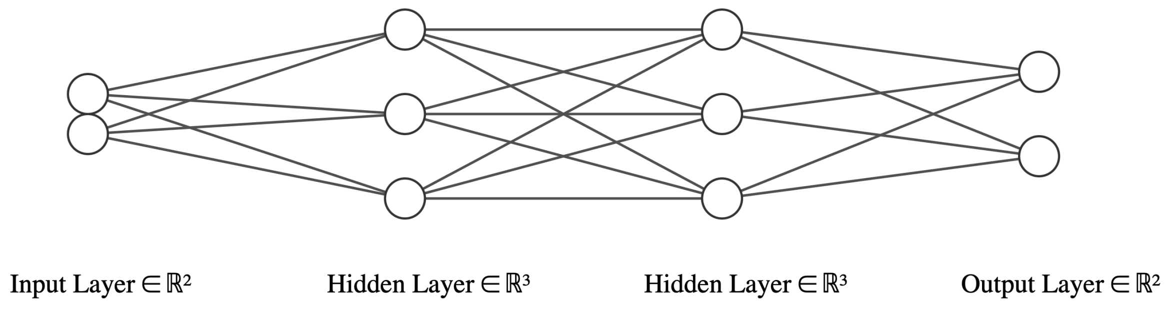
Input Layer $\in \mathbb{R}^2$
Hidden Layer $\in \mathbb{R}^3$
Hidden Layer $\in \mathbb{R}^3$
Output Layer $\in \mathbb{R}^2$

Figure 1: A Simple Neural Network Architecture

To guide your calculations, the feed forward procedure can be formulated as the equation shown below, consider the architecture of the neural network, paying special attention to **activation functions** $\tanh(\cdot)$, **weight matrices** $\mathbf{W}_1, \mathbf{W}_2, \mathbf{w}$, and **biases** $\mathbf{b}_1, \mathbf{b}_2, b$ and calculate these parameters' gradient on the cross entropy loss function. Once you've successfully computed the gradients, your next step is to **articulate the update equation** for each parameter. This process is fundamental for optimizing the network's performance through gradient descent.

$$
\begin{gathered}
\mathit{Input} : \mathbf{x} \\
\mathbf{h}_1 = \tanh(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) \\
\mathbf{h}_2 = \tanh(\mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2) \\
\mathbf{y} = \sigma(\mathbf{w}^{\mathbf{T}}\mathbf{h}_2 + b) \\
\mathcal{L}(\mathbf{y}, \mathbf{y}^*) = -y^* \log y - (1 - y^*) \log(1 - y)
\end{gathered}
$$

Problem 2 - 60%

In this problem, you will leverage your understanding of neural network fundamentals to build a simple fully connected neural network for classifying MBTI (Myers-Briggs Type Indicator) personality types based on preprocessed posts.

dataset: MBTI 500.csv
columns:

- **posts**: Equal-sized posts with 500 words per sample.
- **type**: MBTI personality types indicating differing psychological preferences.

instructions:

1. **Dataset Exploration:**
   - Load the provided CSV file (**MBTI500.csv**) into your preferred programming environment (e.g., Python with pandas).
2. **Data Preprocessing:**
   - Decide whether to perform a 4-way classification (based on dichotomies) or a 16-way classification (considering each personality type independently). Your decision will guide how you preprocess the type column.
3. **Feature Extraction:**
   - Utilize existing libraries (e.g., scikit-learn) to extract features from the posts. You may choose methods like TF-IDF or word embeddings for this purpose.
4. **Neural Network Construction:**
   - Build a simple fully connected neural network **from scratch** (please do not use exist neural network libraries), adhering to the specifications discussed in the lecture and utilizing the results in Problem 1.
5. **Training:**
   - Implement the backpropagation algorithm to train your neural network. Set hyperparameters such as learning rate, batch size, and epochs.

6. **Evaluation:**
   - Use k-fold cross-validation to evaluate the performance of your model. Analyze metrics such as accuracy, precision, recall, and F1 score.
7. **Documentation:**
   - Create a report detailing your approach, including decisions made during data preprocessing, feature extraction, and network architecture. Provide insights into your model's performance and discuss any challenges faced.
8. **Attach your codes and report.**