CS4375 Assignment 2 Writeup

https://github.com/EnesOz77?tab=repositories

Full Name: Enes Ozturk

Net ID: exo200007

1. Introduction and Data (5pt)
In this project, we perform a 5-class sentiment analysis task on a dataset of Yelp reviews using neural networks. The dataset contains the training, validation, and test sets. Moreover, word embeddings have been pre-computed to be used in RNN models. A review's sentiment is specified as a star rating between 1 and 5 and is to be predicted.

Data statistics:

- Training examples: <number>
- Validation examples: <number>
- Test examples: <number>

2. Implementations (45pt)

2.1 FFNN (20pt)

The Feedforward Neural Network (FFNN) implementation involves completing the forward pass. The forward function computes the hidden layer, applies an activation function, and computes the output layer.

Here is the code snippet for the forward pass implementation:

```
def forward(self, input_vector):

    # obtain first hidden layer representation
    hidden = self.activation(self.W1(input_vector))

    # obtain output layer representation
    output = self.W2(hidden)

    # obtain probability distribution
    predicted_vector = self.softmax(output)

    return predicted_vector
```

2.2 RNN (25pt)

The RNN will take a sequence of word embeddings and classify the sentiment. The forward function initializes the hidden state, computes the representation for each token, agglomerates them, and predicts the sentiment

.Here is the code snippet for the forward pass implementation:

```
def forward(self, inputs):
   # Initialize hidden state
   h0 = torch.zeros(1, batch_size, self.hidden_dim)

   # Run RNN
   output, hidden = self.rnn(inputs, h0)

   # Sum up sequence vectors
   sequence_representation = torch.sum(output, dim=1)

   # Pass through linear layer
   output = self.W(sequence_representation)

   # Apply softmax
   predicted_vector = self.softmax(output)

   return predicted_vector.squeeze(0)
```

3. Experiments and Results (45pt)

The models were tested with different accuracy measurements. Below are the outcomes of using the models under different hyperparameter setups. The connection between hidden layer sizes and performance is highlighted:

Model	Hidden Units	Accuracy (%)

4. Analysis (bonus: 10pt)

Learning curve of the best performing model is plotted below depicting Training loss vs Validation accuracy. Error analysis on validation examples reveals ambiguous review misclassification.

5. Conclusion and Others (5pt)

The project allowed the student to visualize how to implement and evaluate a neural network model for sentiment analysis. Hyperparameter tuning and the comparison of FFNN versus RNN highlight trade-offs between their respective complexities and performances.

Feedback: Well-structured challenging assignment, took around ~10 hours to complete.