CS6375 Assignment 1

https://github.com/BobMaa/CS6375-Assignment1.git

Zhao Ma
ZXM240001

March 2, 2025

Instructions

TODO: Fill in your name and details above. If not filled correctly, we will subtract 2pt.

TODO: It is highly suggested to use the template for reports. You are free to use other software (e.g., Microsoft Word, Google Docs), but we prefer you follow the structure below.

1 Introduction and Data (5pt)

TODO: Briefly describe the project and your main experiments and results, including mentioning the data you use.

ANSWER: This project aims to predict the star ratings according to the comment customers make. It involves implementing and training two neural network models for text classification tasks: a Feedforward Neural Network (FFNN) and a Recurrent Neural Network (RNN). The models are trained using a dataset of textual reviews and their corresponding star ratings. The experiments involve training the models on a training dataset, validating them on a validation dataset, and evaluating their performance using accuracy as the metric.

TODO: Briefly describe the task and data (e.g., how many examples are in the training, development, and test sets). It is best to report all the statistics, including counts, in a table.

ANSWER:

Dataset	Number of Examples
Training	16000
Training for Each Star	3200
Validation	800
Test	800

2 Implementations (45pt)

2.1 FFNN (20pt)

TODO: Explain briefly how you implemented and filled in the incomplete code for `FFNN.py` in the form of a screenshot (and explanations) in the report. Provide any other libraries/tools that are used. Try to understand what other parts of the code are doing, and write your understanding here (e.g., optimizers, initializations, stopping, etc.).

ANSWER: The Feedforward Neural Network (FFNN) consists of the following components:

```
def forward(self, input_vector):
    # [to fill] obtain first hidden layer representation
    hidden_layer = self.activation(self.W1(input_vector))
    # [to fill] obtain output layer representation
    output_layer = self.W2(hidden_layer)
    # [to fill] obtain probability dist.
    predicted_vector = self.softmax(output_layer)
    return predicted_vector
```

Figure 1: **forward()** in FFNN

1. Layers:
   (a) Input layer
   (b) A hidden layer with h neurons with ReLU activation.
   (c) An output layer with 5 units which correspond to the star ratings. LogSoftmax activation is applied.
2. Loss Function: Negative Log Likelihood Loss (NLLLoss) is adopted.
3. Optimization: Stochastic Gradient Descent (SGD) with a learning rate of 0.01 and momentum 0.9.
4. Training Process:
   (a) Text data is vectorized.
   (b) Mini-batch training is performed with a batch size of 16.
   (c) The training loop iterates over mini-batches of the training data, computes the loss, and updates the model parameters using back-propagation.

2.2 RNN (25pt)

TODO: Explain briefly how you implemented and filled in the incomplete code for `RNN.py` in the form of a code-snippet screenshot (and explanations) in the report. Try to understand what other parts of the code are doing, and write your understanding here (especially parts that function differently compared to FFNN).

ANSWER: The Recurrent Neural Network (RNN) implementation differs from FFNN by incorporating sequential dependencies in text. It consists of:

```
def forward(self, inputs):
    # [to fill] obtain hidden layer representation (https://pytorch.org/docs/stable/generated/torch.nn.RNN.html)
    outputs, hidden = self.rnn(inputs)
    # [to fill] obtain output layer representations
    output_layer = self.W(outputs)
    # [to fill] sum over output
    sumed_outputs = torch.sum(output_layer, dim=0)
    # [to fill] obtain probability dist.
    predicted_vector = self.softmax(sumed_outputs)
    return predicted_vector
```

Figure 2: **forward()** in RNN

1. Architecture:
   (a) The input is a sequence of word embeddings.
   (b) A single hidden layer with h units and tanh activation.
   (c) A linear output layer.
   (d) The hidden states are summed and passed through the final layer.
   (e) A LogSoftmax activation for probability distribution.
2. Loss Function: Negative Log-Likelihood Loss.
3. Optimization: Adam optimizer (lr=0.01).
4. Training Process:
   (a) Text is converted into word embeddings.
   (b) iterate over mini-batches and update the model parameters.
   (c) Training stops early if validation accuracy decreases while training accuracy increases.

3 Experiments and Results (45pt)

3.1 Evaluations (15pt)

TODO: Explain how you evaluate the models. What metric is used—you can refer to the current implementation.

ANSWER: The models are evaluated using accuracy (the proportion of correct predictions) as the metric. The trained model is evaluated on the validation set.

3.2 Results (30pt)

TODO: Apart from the default hyperparameters, try multiple variations (between 1-2 for FFNN and RNN each) of models by changing hidden unit sizes.

TODO: Summarize the performance of your system and put the results into tables or diagrams. Include your observations and analysis.

ANSWER: We adopt 5 epochs and calculate the mean of validation accuracy, and take the training accuracy in last epoch for FFNN.

Model	Number of Hidden units	Training Accuracy	Validation Accuracy
FFNN	50	0.597	0.518
FFNN	100	0.615	0.550
RNN	50	0.3348	0.38875
RNN	100	0.3381	0.3

1. The FFNN generally performs better than RNN in this task.
2. With the increase of the number of hidden units in FFNN, training accuracy increase a bit but the validation accuracy improves a lot. This shows that the dimension of hdden layer affects the performance of FFNN.
3. When we increase the number of hidden units in RNN, the training accuracy does not improve, and the validation accuracy even decreases.
4. In FFNN, with the increasing of training accuracy, the validation accuracy does not improve simultaneously. In contrast, the validation even decreases in RNN when training accuracy increases which may be led by over-fitting.

4 Analysis (bonus: 10pt)

TODO:

- **(5pt)** Plot the learning curve of your best system. The curve should include the training loss and development set accuracy by epoch.
- **(5pt)** Error analysis. List some (one or more) error examples and provide some analysis. How might you improve the system?

  ANSWER:

  1. Error: "Nothing special... Side items were good, but ribs were blah! I wouldn't make this a regular place.." $\rightarrow$ Predicted: 3, Actual: 2.
  2. Analysis: The words are both good and bad, so it may cause some misunderstandings.
  3. Fine-tuning word embeddings for better representation.

5 Conclusion and Others (5pt)

TODO:

- Individual member contribution: Zhao Ma independently finished this report.
- Feedback for the assignment (e.g., time spent, difficulty, and how we can improve): About 8 hours. Medium Difficulty. If there is a Jupyter Notebook version, that will be easier for debugging.