CS6375 Assignment 1

FULL_GITHUB_REPOSITORY_URL

Dhyey Rajesh Shah
DXS240020

1 Introduction and Data (5pt)

In this project we mainly focus on exploring different neural network models, Recurrent Neural Network and Feed Forward Neural Network to be precise. We use them on a Yelp Review Dataset to train the model for sentiment analysis. We use a bag of word vectorization for the FFNN whereas we use the word vectors from the provided word embedding file

After a quick analysis of data using pandas we can conclude that

Dataset	Total Samples	Class Distribution
training.json	8000	{1.0: 3200, 2.0: 3200, 3.0: 1600}
validation.json	800	{1.0: 320, 2.0: 320, 3.0: 160}
test.json	800	{4.0: 320, 5.0: 320, 3.0: 160}

2 Implementations (45pt)

2.1 FFNN (20pt)

The FFNN.py program builds a feedforward neural network for sentiment classification. There are two linear layers: the hidden layer (W1) with ReLU activation and the output layer (W2) with LogSoftmax for the probability distribution. The model is trained on Negative Log-Likelihood Loss (NLLLoss).

Data processing entails reading JSON files (load_data()), building a vocabulary (make_vocab()), word-to-index mapping (make_indices()), and text to bag-of-words vectors (convert_to_vector_representation()).

Training is done using SGD with momentum (0.9) and mini-batches of size 16. The model is evaluated at the end of each epoch by computing training and validation accuracy to check for correct learning and generalization.

2.2 RNN (25pt)

RNN.py employs a Recurrent Neural Network (RNN) for sentiment classification, working with sequential text data rather than a bag-of-words vector of fixed size as in FFNN.py.

The RNN model works by using an RNN layer with a hidden state (h) to process input sequences, passing the last hidden state through a linear layer and LogSoftmax activation to get class probabilities. It is trained with Negative Log-Likelihood Loss (NLLLoss).

Data processing involves review loading, tokenizing text, and using pre-trained word embeddings provided to us in the word embeddings pkl file.

Training of the model is done using Adam optimizer and mini-batches and same goes for validation and find the accuracy of the model using correct outputs over total outputs.

3 Experiments and Results (45pt)

Evaluations (15pt)

Both FFNN and RNN models use accuracy as the measurement parameter, calculated as correct predictions divided by total predictions.

Accuracy is achieved by comparing the maximum probability predicted label and the true label, then computing the ratio:

Accuracy = Correct/ Total

The loss function used here is Negative Log-Likelihood Loss (NLLLoss), a common function for classification problem with softmax output.

Results (30pt)

FFNN

Hidden Dimensions	Epoch	Highest Training Accuracy	Highest Validation Accuracy
8	1	52.2	54.88
32	1	53.36	52.25
128	1	53.01	54.25

RNN

Hidden Dimensions	Epoch	Highest Training Accuracy	Highest Validation Accuracy
8	10	44.55	44
64	10	40.75	45.13
128	1	40.23	41.75

4 Conclusion and Others (5pt)

The overall code implementation for this assignment was easier to fill in based on the comments provided. This assignment allowed us to explore the implementation of RNN and FFNN using PyTorch and in general how to use PyTorch based on the documentation. I spent 5-6 hours to go thought the PyTorch documentation. The low accuracy of these models is due to lack of training data as described in the data analysis but another probable reason is the use of simple Linear Algorithm in the Neural Network which can be replaced with other algorithms for better performance.