ZZEN9444 Hexamester 2 2022

Assessment 3 Written Report

Sze Chan

Z5061978

Part 1: Selection of architecture, algorithms, and enhancements:

The chosen architecture was a recurring bidirectional long short-term memory (LSTM) neural network after assessing its main advantages over others in the task of learning to read business reviews in text format and predicting a rating. LSTMs in general are better than a simple recurrent network as, firstly, it sidesteps vanishing and exploding gradients that is otherwise associated with simple recurrent networks. Also, in the course we learned that LTSM networks can learn long range dependences, which simple recurrent networks struggle on, by utilising a combination of 3 internal neural networks, being the forget, input and output gates. In particular, the bidirectional variation of the LSTM was chosen after realising its ability to provide additional context to a sequence thus improving the performance of a model in classification situations. This is owing to its mechanism to present an input forward and backwards to two different networks, both connected to same output layer.

The algorithm used is the Global Vectors for World Representation (GloVe 6B) unsupervised learning algorithm to get words represented as a vector. This was chosen after further reading that it embeds words for text, resulting in similar words to be represented in a similar manner.

As part of enhancements to the model's efficiency and performance, weight initialisation was used to get to a set of weights that can make optimal predictions, that is minimising a loss function. As discussed in the course and deployed here, the initial values were assigned to the weights at the start of the process.

Part 2: Selection of cost function and optimiser:

For rating prediction, the appropriate cost function was determined to be the binary cross-entropy loss function, as we know that theoretically (and indeed in practice here) it approximates the likelihood of a forecasted label belonging to a target class label and yields quicker learning with better generalisation during classification tasks.

The optimiser chosen was adaptive moment estimator (Adam) as the pace and efficacy of learning with Adam was better than stochastic gradient descent with momentum. To illustrate this, the network that experimented with stochastic gradient descent required the learning rate (LR) to be 0.07 and momentum (M) to be 0.75. However, Adam required a LR = 0.001, far lower than that of stochastic gradient descent.

I also realised that the optimal learning rate for Adam was 0.001 in conjunction with weight decay of 1e-6 after trialling many different values.

Part 3: Dimensionality of Word Vectors, metaparameter values, and any data preprocessing used (if any):

From above, it was discussed that the model used Global Vectors for Word Representation (GloVe 6B) unsupervised learning algorithm was used and I have determined that the best dimensionality of the word vector was 300 for this task as it allows vector to get more information, and after experimentation, performed best compared 50, 150, 200 and 250 dimensions. Although one drawback would be greater complexity in computation with increased dimensions.

In terms of metaparameters, for the ratings prediction task I ensured values of 0 or 1 were output:

1. Prediction values more than 0.5 were categorised positive with 1.
2. Prediction values lower than 0.5 were categorised negative with 0.

For classifying businesses into categories, the target class labels [0,1,2,3,4], reflect restaurants, shopping, home services, health and medical, and automotive respectively. The model employs the Cross EntropyLoss() loss function, and gives a probability distribution [0,1]. Thus, the highest probability predicted label was allocated to target class after using this conversion process (from code):

- categoryOutput = tnn.Softmax(categoryOutput, dim=1)
- categoryOutput = torch.argmax(categoryOutput, dim=1)

Before deep diving into the work, I made sure to do a bit of cleaning with the data to make sure to it was ready to use later. The preprocessing() defined in student.py performs additional preprocessing of data using regular expression package, including the removal of html mark tags, removal of unwanted characters and removal of non-ascii and digits.

To perform the text cleansing task, the function has been called after the tokenising (break up and separating sequencing into discrete elements) but before numericalising.

Part 4: Validation set and dealing with overfitting:

The validation set was employed such that 20% of input data was assigned to validation. Approximation of the network's performance was obtained afterwards through approximations for the validation set. This allowed for the experimentation of different hyperparameters and choosing the best one(s) based on performance on validation set. Once these parameters were found the network was run on the complete training dataset. This helps increase the accuracy.

To deal with overfitting, dropout and weight decay were used. With the first, the dropout layer momentarily disconnected a few of the units' inputs and output from the preceding layer. Once batch was completed, such units and connections were reinstated. Once the following batch starts, a new random units are momentarily removed and so on for each epoch. It thus prevents overfitting as it stops any unit from focusing too much on the training data.

Weight decay involved adding a penalty term to loss function, encouraging the network weights to remain small. Weight decay specifies Gaussian prior over the model parameters and results in regularisation of the network's intricacy.