11747 HW1 Report

Yikai Feng (yikaif)

1. Introduction

In this assignment, I built a text classifier with dataset from Topic Classification data. The dataset has three parts, training data, development data and test data. The classifier was trained on the training dataset and then validated on the development dataset. Both datasets compared the prediction result output from the model with the ground truth label and computed the accuracy rate as the evaluation metrics. To get statistics on different model performance, I experimented with different models and parameters on the same text classification task.

1. Implemented Model

I implemented an CNN model with pre-trained Fasttext word embedding based on the course code base. To add pre-trained embedding weight matrix, I inversed the w2i (a dictionary map from the words in the training set to its index) dictionary and map the word to the 300 size embedding vector obtained from the Fasttext pre-trained model so that the embedding weight is preloaded and in order. Then, I trained the dataset with different models and applied parameter tuning to it. The main tricks that I tried is multi-layer CNN with max pool, dropout, and parameter tuning on CNN parameters and Adam optimizer parameters. I used cross entropy loss to update the weight matrix in the neural network. After the training process was completed, I calculated the accuracy of the prediction and use this as the performance metrics for analysis. The trained model is then applied to the development dataset to calculate a validation accuracy for analysis. The final goal is to use the best model to predict the unlabeled data in the test dataset.

1. Result and Analysis
   1. Different layers of CNN

I built three models each with one, two and three CNN layers and experimented on the network architecture to see if the performance is varied with each other. I observed that the two layer model performs the best.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Single layer | Two layers | Three layers |
| Train | 0. 8117 | 0.7814 | 0.7728 |
| Dev | 0.7652 | 0.8103 | 0.7932 |

* 1. Parameter tune
     1. Filter size (8, 50, 100, 150)

At first, I found that small filter size will result in bad performance. Since the embedding dimension is 300 per word, the in channel of the convolutional layer is 300, more filter number can extract more features from the embedding. However, I found that if out channel = 150, the performance is not improving.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Filter size = 8 | Filter size = 50 | Filter size = 100 | Filter size = 150 |
| Train | 0. 6882 | 0.6704 | 0.8117 | 0.7053 |
| Dev | 0.7030 | 0.6998 | 0.7652 | 0.7527 |

* + 1. Learning rate (1e-4, 3e-4, 5e-4, 1e-3)

After fixing the filter size (100), I ran the one layer CNN to see how learning rate influenced the model performance. From the table, development dataset achieved best on learning rate 5e-5 and 1e-4.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | lr = 5e-5 | lr = 1e-4 | lr = 3e-4 | lr = 5e-4 | lr = 1e-3 |
| Train | 0.7928 | 0.8100 | 0.8228 | 0.8187 | 0.7039 |
| Dev | 0.0.8072 | 0.8056 | 0.7916 | 0.7652 | 0.7558 |

* 1. Overfitting

During the experiment, I observed an overfitting phenomenon on the training process in the single CNN layer model. From the chart, as the training epoch going, the training set loss reduced and the predict accuracy grew, however the validation accuracy dropped. Since higher the training accuracy is, the better the model is predicting the data in the training set, the model is approximating the training set distribution while ignore the data bias in the development dataset. I also observed that the overfitting is reduced in the two-layer CNN model as shown in the plot below, this also appeared in the three-layer model. So I assumed that the more complex the model is, the easier avoiding overfitting is.

* + 1. Dropout

To apply a dropout layer on the model could to some extent reduce the overfitting problem, however, the model took longer to converge. I found that the two-layer CNN with dropout (0.2) worked better, since it achieved a validation accuracy rate of 81.03% and converged within 5 epochs.

1. Conclusion

The model that produced the best result is the one with two-layer CNN architecture. The kernel size is 3, filter size is 100, learning rate is 5e-5. The two layers are:

* Conv1d
* Con1d, Dropout(0.2)