CS388 Mini 2: Neural Networks for Sentiment Analysis

Yixuan Ni, yn2782

1 Introduction

The goal of this project is to classify the sentences from Rotten Tomatoes as either positive or negative sentiment. I have implemented a simple feedforward network and a CNN network, which yields a best accuracy of 0.7636 and 0.8002, respectively. This report includes the experiments I have done to search for the best hyper-parameters for each model and a comparison of the results.

2 Method

The training set used in this project includes 8530 sentences, and the validation set includes 1066 sentences. I train each model for 20 epochs, shuffling the data at each epoch. I use the provided relativized embedding of both sizes and keep the embedding fixed during training. I apply early stopping mechanism to avoid overfitting.

For the experiments, I first design a benchmark model and measure the performance. Then I change the value of one of the hyper-parameters and keep others unchanged to test the effect on the model performance.

2.1 Feedforward Neural Network

I follow the implementation in (Iyyer et al., 2015). The benchmark model has 1 hidden layer with 64 hidden units, embedding size of 300 and dropout rate of 0. It uses Adam optimizer and the tanh function as activation function. The hyper-parameters I have tested on are: number of layers, embedding size, activation function and optimizer.

2.2 Convolutional Neural Network

I implement the model following the architecture defined in (Kim, 2014). The benchmark model has filter region size of $(3, 4, 5)$, each with 100 feature maps, embedding size of 300, dropout rate of 0.5 and uses SGD optimizer with learning rate 0.1. It uses the ReLU function as activation function. To facilitate the training process, I use mini-batches with size 32. The hyper-parameters I have tested on are: filter region sizes, feature map size, embedding size and activation function.

3 Results

Table 1 includes the experimental results for the feedforward neural network model. There are some interesting observations:

hyper-parameter	**train acc**	**dev acc**
benchmark	0.7570	0.7495
number of layers		
0	0.7311	0.7373
1	**0.7570**	**0.7495**
2	0.4994	0.5047
embedding size		
50	0.6989	0.6885
300	**0.7570**	**0.7495**
activation		
Tanh	**0.7570**	**0.7495**
ReLU	0.4994	0.5047
LogSigmoid	0.6147	0.6023
Softplus	0.6508	0.6492
RReLU	0.7478	0.7448
optimizer		
Adam	0.7570	0.7495
SGD	**0.7774**	0.7523
AdaDelta	0.7685	0.7570
Adagrad	0.7645	**0.7636**

Table 1: Experimental results on feed-forward neural network models. All other hyper-parameters are fixed except for the one listed in the table.

hyper-parameter	**train acc**	**dev acc**
benchmark	1.000	0.7862
filter region size		
(2,3,4)	0.9864	**0.8110**
(3,4,5)	1.000	0.7862
(4,5,6)	0.9961	0.7853
(3,3,3)	1.000	0.7899
feature map size		
100	1.000	0.7862
200	1.000	**0.7908**
300	1.000	0.7871
activation		
Tanh	1.000	**0.7954**
ReLU	1.000	0.7862
optimizer		
Adam	0.5885	0.5770
SGD	1.000	0.7862
AdaDelta	1.000	0.7825
Adagrad	0.9820	**0.7882**

Table 2: Experimental results on CNN models.

- It is not necessary that deeper network performs better than shallower network. For this task, a simple linear model (no hidden layer) performs very close to the model with 1 hidden layer.

- The choice of activation function is essential to the model. The best accuracy is 0.25 higher than the worst accuracy.
- The choice of optimizer does not affect too much on the performance. However, I have not fine-tune the hyper-parameters of the optimizers, which may affect the performance of some optimizers.

In overall, a simple model such as the benchmark model works well on this task, yielding an accuracy of 0.7495 on the validation dataset. The best model uses the Adagrad optimizer and achieves an accuracy of 0.7636.

Table 2 includes the experimental results for the CNN model. From these results, I notice that:

- Although according to (Zhang and Wallace, 2015), using single value as the filter region size usually performs better than using a combination of region sizes, in this case, using region size of $(2, 3, 4)$ performs the best.
- a larger feature map size does not necessary give the best performance. Using larger size increases the training time significantly. The average training time for one epoch increases from 35 seconds for size of 100 to 45 seconds for size of 300.
- Adagrad, again, is the optimizer that gives the best performance. The reason might be that Adagrad requires the least amount of fine-tuning to provide the best results.

The best CNN model uses filter region size of $(2, 3, 4)$ and gives an accuracy of 0.8110. Comparing these two models, feedforward networks in general perform worse than CNN and the performance varies significantly depends on the hyper-parameter choices; CNN on the other hand perform better and is more stable. CNN, however, takes longer to train. It takes 35-45 seconds to train one epoch while feedforward network takes about 15-20 seconds to train one epoch. For a small dataset we do not need to consider much about the overall training time, but the difference will be more noticeable for larger dataset.

4 Conclusion

In this project I implement feedforward network models and CNN models to classify sentiment of the sentences from Rotten Tomatoes. I experiment on different hyper-parameter settings for both models. The best accuracy achieved are 0.7636 and 0.8002 for feedforward and CNN, respectively.

References

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691, Beijing, China. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882.

Ye Zhang and Byron C. Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *CoRR*, abs/1510.03820.