Deep Learning models for Sentence Classification, Assingment 1, course CE7455

Adam Barla
n2308836j@e.ntu.edu.sg
NTU, Singapore

1 Configuration Optimization

1.a

Implement the `pack_padded_sequence` function in PyTorch's RNN library. Report results under the default setting and discuss the benefits of this function.

The function `pack_padded_sequence` takes input sequences of varying lengths and packs them into a compact form, before they are fed into the RNN. By using pack_padded_sequence, the model can skip the padded areas, focusing only on the actual data, which can lead to more accurate and faster training.

I created a class `PackedRNN` that used this function. Then, I conducted 10 training runs of 100 epochs each for model using the function and for the baseline model. I compared the models on validation accuracy so the test accuracy can still serve as an indicator of the performace on unseen data in the end. Average valildation accuracy of the baseline model was 67.17% while model with `pack_padded_sequence` reached 68.53%, which is a minor improvement. The average duration of one epoch went up by ~3.5 seconds from 17.5s to 21s. This can be improved by packing the sequences only once and not every time `forward` method is called.

1.b

Experiment with different configurations (optimizers, learning rates, batch sizes, sizes of hidden embedding) and report the best configuration's performance on the validation and test sets.

For this purpouse I decomposed the notebook to a project. Using `hydra` and `wandb` I created a sweep over hyperparameters. I chose to test using grid search over these parameters based on some empirical test runs:

```
lr:
  values: [ 0.00001, 0.0001, 0.001, 0.01, 0.1 ]
batch_size:
  values: [ 32, 64, 128 ]
optimizer:
  values: [ sgd, adam, adadelta, adagrad, rmsprop ]
model.hidden_dim:
  values: [ 50, 100, 200, 400 ]
```

I tested each combination of parameters with `PackedRNN` class from Section 1.a for 100 epochs. The best performance on validation accuracy was achieved when using these parameters:

```
hidden_dim: 200
optimizer: adadelta
batch_size: 32
lr: 0.1
```

The validation and test accuracy reached was 77.71% and 83.8% respectively.

Parameter interaction analysis.

Figure 1 shows an average validation accuracy when two parameters are kept at a specific value. We can observe trends that help us better understand which hyperparameters are optimal. For example we can see general correlation between hidden embedding size and validation accuracy, which is not suprising. Adam and RMSprop optimizers seem to generally dominate, even though Adadelta reached the highest validation accuracy. Each optimizer has a learning rate at which it works best as it seems. Trend towards smaller batch sizes seems curious. Given more time I would explore it further.

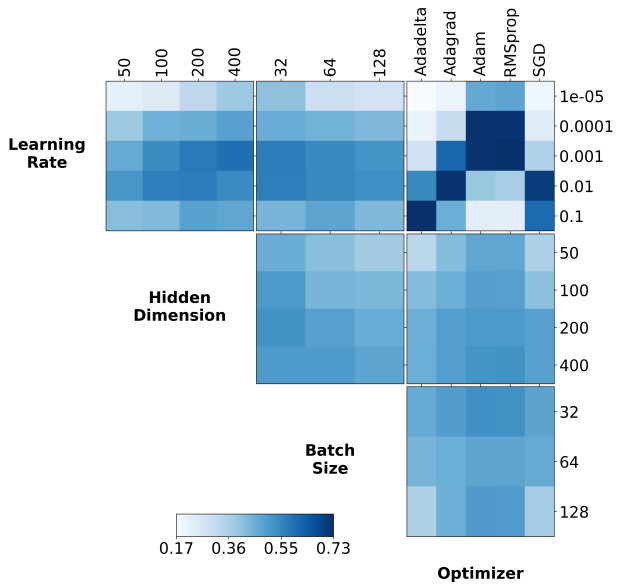

Figure 1: Each square represents a mean validation accuracy of runs with two parameters set to particular value.

For the next experiments I will use Adam with hidden dimension of 200, batch size of 32. Learning rate may vary in the future but a good value for this configuration seems 0.001.

1.c

Implement regularization techniques, describe them, and report accuracy results after application.

Dropout. During training, randomly zeroes some of the elements of the input tensor with probability p given by a parameter, which greatly reduces overfitting. This has proven to be an effective technique for regularization and preventing the co-adaptation of neurons as described in the [1]. Dropout has an effect of training and using an ensamble of models and promotes learning of a sparse representation.

L1/2 Regularization.

We can include a regularization parameter to the loss, which is computed based on the parameters of the model.

For L1 it is

$$\alpha \sum_{\omega \in \Omega} |\omega|$$

and for L2 it is

$$\alpha \sum_{\omega \in \Omega} \omega^2$$

where Ω is a set of all parameters of the model and α is a weight of the regularization parameter.

This regularization results in sparsity.

Gradient clipping.

To counteract exploding gradient we can employ gradient clipping. It scales the gradient $\boldsymbol{g}$ down if it's norm $\|\boldsymbol{g}\|$ is larger then some treshold t.

$$\text{if } \|\boldsymbol{g}\| \geq t : \boldsymbol{g} \leftarrow t * \left(\frac{\boldsymbol{g}}{\|\boldsymbol{g}\|}\right)$$

For this I used function `torch.nn.utils.clip_grad_norm`.

Early stoping. Up to a point, training improves the learner's performance on the validation set. Past that point, however, improving on the training data comes at the expense of increased generalization error.

Batch normalization.

Batch normalization makes training of NNs faster and more stable through normalization of the layers' inputs by re-centering and re-scaling. It can mitigate the problem of internal covariate shift, where parameter initialization and changes in the distribution of the inputs of each layer affect the learning rate of the network.

It isnt' common to use batch norm with RNNs but it can be used later with more complex classifiers in later sections.

2 Input Embedding

Switch from randomly initialized input word embeddings to pretrained word2vec embeddings. Report accuracy on the validation set and compare performance.

Gensim installation and pretrained word2vec models: Gensim, Pretrained models.

3 Output Embedding

Explore options for computing sentence embedding beyond the final hidden representation. Implement the best option(s) and report accuracy on the validation set, comparing it to the performance in Task 2.

4 Architecture Optimization

Experiment with more complex RNN architectures (GRU, LSTM, Bidirectional simple RNN, simple RNN with 2 hidden layers) and report accuracy on the validation set.

5 Critical Thinking

Propose and implement a modification to further improve performance. Conduct experiments and report accuracy on the validation set.

References

[1] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *Corr*, 2012. [Online]. Available: http://arxiv.org/abs/1207.0580