MSML641 Assignment 3 - RNN Sentiment Analysis

By Ashish Kumar Singh

Compute Configuration =>
CPU: Intel Core i9 12900K
RAM: 64 GB DDR4
GPU: Nvidia RTX 4080

Core ML library: Jax/Flax (Flax is a high level pytorch like library built on top of jax. Jax is a high performance mathematical/scientific library built around the idea of purely functional programming paradigm. I LOVE jax and I prefer using it for every project I do, hence the assignment uses Flax instead of pytorch. Also, it always runs 30% faster than the same code in pytorch)

1. Dataset Summary

IMDb Movie review dataset contains 50k reviews, and is hosted on kaggle, with 25k train and 25k test split. The data exploration has been done and recorded in the 'playground.ipynb' notebook, with the following statistics:

- **Total Vocabulary**: 144024 unique words
- **Average review length**: 226 tokens (median of 170)
- **Review length range**: 4 to 2450 tokens

2. Model Configuration

For the assignment, I chose a baseline configuration as follows:

- **Model**: LSTM
- **Activation**: tanh
- **Optimizer**: Adam
- **Sequence** Length: 50
- **Gradient** Clipping: No

Apart from these, the other configurations have been as per the assignment:

- **Embedding dims**: 100
- **Hidden** Layers: 2 with 64 units each
- **Dropout**: 0.4
- **Output Layer**: FNN sigmoid
- **Loss**: Binary Cross entropy
- **Batch size**: 32

- **Max training epoch**: 30 (with early stopping)

I define an experiment config adhering to the assignment requirements as follows:

```
   {'name': 'MODEL ARCHITECTURE', 'variable': 'model_type', 'values':
['rnn', 'lstm', 'bilstm']},
   {'name': 'ACTIVATION FUNCTIONS', 'variable': 'activation', 'values':
['sigmoid', 'relu', 'tanh']},
   {'name': 'OPTIMIZERS', 'variable': 'optimizer', 'values': ['adam',
'sgd', 'rmsprop']},
   {'name': 'SEQUENCE LENGTHS', 'variable': 'seq_length', 'values': [25,
50, 100]},
   {'name': 'GRADIENT CLIPPING', 'variable': 'grad_clip', 'values':
[False, True]}
```

A simple early stopping mechanism was implemented with patience = 5 to save computation time and best test f1 scores were tracked for it, which were then reported for every experiment. Baseline was trained and evaluated first, and then other models were trained in comparison by changing a single variable. Although this is not the best hyperparameter search, this is what the assignment required as per my understanding.

3. Comparative Analysis:

For each experiment, only a single variable was changed, keeping all the other variables fixed (equal to baseline). For each experiment run, I tracked and plotted the f1 scores, and we can see from the final experiment logs that the best configurations were:
Model: BiLSTM
Activation: tanh
Optimizer: rmsprop
Sequence Length: 100
Gradient Clipping: Yes

```
================================================================================
Best Configurations per Experiment
================================================================================

MODEL ARCHITECTURE:
  BILSTM | Act=tanh | Opt=adam | Seq=50 | Clip=No
  F1: 0.7676 | Acc: 0.7692

ACTIVATION FUNCTIONS:
  LSTM | Act=sigmoid | Opt=adam | Seq=50 | Clip=No
  F1: 0.7753 | Acc: 0.7753

OPTIMIZERS:
  LSTM | Act=tanh | Opt=rmsprop | Seq=50 | Clip=No
  F1: 0.7821 | Acc: 0.7821
```

SEQUENCE LENGTHS:
LSTM | Act=tanh | Opt=adam | Seq=**100** | Clip=No
F1: 0.8187 | Acc: 0.8190

GRADIENT CLIPPING:
LSTM | Act=tanh | Opt=adam | Seq=50 | Clip=**Yes**
F1: 0.7625 | Acc: 0.7644

model_type	activation	optimizer	seq_length	grad_clip	accuracy	f1_score	best_epoch	avg_epoch_time
lstm	tanh	adam	50	FALSE	0.76308	0.7611573938	2	4.251046555
rnn	tanh	adam	50	FALSE	0.72416	0.7240195388	9	2.393057585
bilstm	tanh	adam	50	FALSE	0.7692	0.7675693274	2	8.099146264
lstm	sigmoid	adam	50	FALSE	0.77532	0.7752672547	3	4.117018759
lstm	relu	adam	50	FALSE	0.69228	0.6893376603	4	4.019469182
lstm	tanh	sgd	50	FALSE	0.52044	0.5009133132	2	4.095594781
lstm	tanh	rmsprop	50	FALSE	0.78212	0.782110043	5	3.990185809
lstm	tanh	adam	25	FALSE	0.72852	0.7281900701	2	2.505736487
lstm	tanh	adam	100	FALSE	0.819	0.8187354916	5	7.332429218
lstm	tanh	adam	50	TRUE	0.76436	0.7624785929	2	4.15446966

Using the best configuration found, the model trained had by far the best f1 score:

Results: Accuracy=**0.8277**, F1=**0.8277**, Best Epoch=15, Avg Epoch Time=14.50s
Confusion Matrix:
[[10437 2037]
[2271 10255]]
Classification Report:

	precision	recall	f1-score	support
Negative	0.82	0.84	0.83	12474
Positive	0.83	0.82	0.83	12526
accuracy			0.83	25000
macro avg	0.83	0.83	0.83	25000
weighted avg	0.83	0.83	0.83	25000

4. Discussion

4.1 Model Architecture Impact

BiLSTM (F1: 0.7676) outperformed RNN (F1: 0.7240) and showed modest improvement over standard LSTM (F1: 0.7612). The bidirectional architecture captures both forward and backward context, providing richer representations for sentiment classification. However, BiLSTM requires approximately 2x the training time (8.10s vs 4.25s per epoch).

4.2 Activation Function Analysis

Sigmoid (F1: 0.7753) achieved the best performance among activation functions, compared to tanh (F1: 0.7612) and ReLU (F1: 0.6893). ReLU showed worse performance with training instability. For LSTM cells, both sigmoid and tanh are good choices, with sigmoid being slightly better for this sentiment classification task.

4.3 Optimizer Comparison

RMSprop (F1: 0.7821) slightly outperformed Adam (F1: 0.7612), while SGD showed poor convergence (F1: 0.5009). Adaptive learning rate methods (Adam, RMSprop) are essential for effective RNN training, with SGD struggling on this task.

4.4 Sequence Length Effects

Most significant factor: Sequence length of 100 tokens (F1: 0.8187) outperformed 50 tokens (F1: 0.7612) and 25 tokens (F1: 0.7282). Longer sequences capture more context for sentiment analysis

4.5 Gradient Clipping

Gradient clipping showed minimal impact (F1: 0.7625 with clipping vs 0.7612 without), a difference of only 0.13 percentage points. This suggests the model training is already stable, and explicit gradient constraints provide negligible benefit for this dataset.

5. Conclusion

Optimal Configuration

- **Model:** BiLSTM
- **Activation:** Sigmoid
- **Optimizer:** Rmsprop
- **Sequence length:** 100
- **Gradient clipping:** Yes

- **Performance:** F1 = 0.8277, Accuracy = 0.8277

Justifications

1. BiLSTM are marginally better in performance compared to LSTM, but at 2x the compute cost (not a problem for my PC though).
2. Adaptive optimizers like Adam, RMSProp are essential; SGD fails to converge effectively
3. Gradient clipping provides minimal benefit as the gradients are already fairly stable on the given dataset