CS6375 Assignment 1

<FULL_GITHUB_REPOSITORY_URL>

Sri Kruthi Madugula

sxm220429

# Introduction and Data (5pt)

**TODO:** Briefly describe the project and your main experiments and results, including mentioning the data you use.

Given is a Yelp reviews dataset having 5 output classes(target classes).The main task lied in predicting the y i.e sentiment score of the reviews and hypertuning different parameters to optimize the accuracies while observing and making note of how the model is learning,generalizing and performing on test data.

Two Neural network architectures i.e FFNN which used a Bag-of-words approach and RNN which used word-embeddings given in a .pkl file are employed to perform this.

TODO: Briefly describe task and data (e.g. how many examples are in the training, develop-ment, and test sets.), it is best to report all the statistics, including counts, in a table.

Both these models are evaluated based on accuracy and loss metrics by varying different hyperparameters.

|  |  |
| --- | --- |
| Train data | 16000 |
| Test data | 800 |
| Validation data | 800 |

# Implementations (45pt)

## FFNN (20pt)

**TODO:** Explain briefly how you implemented filled in the incomplete code for FFNN.py in the form of screenshot (and explanations) in the report. Provide any other libraries/tools that are used; Try to understand what other part of the code is doing, and write your understandings here (e.g. optimizers, initializations, stopping, etc.)

A screen shot of a computer program

Description automatically generated

* In the **forward() method**,a linear transformation is applied to the input vector whereafter a ReLU activation function is applied.A dropout layer(0.25) is added to prevent overfitting.The final output vector is obtained by applying another linear transformation on the hidden layer and a LogSoftmax function is applied on the output vector to covert it into target classes.
* The model uses **Negative Log Likelihood Loss (NLLLoss**), which is appropriate for the log probabilities produced by LogSoftmax.
* **Training data is shuffled** to avoid model Learning the order of data rather than generalizing it.There is a variable mini\_batch size which is used to process data in batches to improve efficiency.
* FFNN does use **SGD optimizer** which is Stochastic Gradient Descent with a learning rate of 0.0001 and a momentum of 0.9.
* The code calculates **accuracy for each epoch** during training,testing and validation.Training accuracies help in understandinghow good the model is being trained,validation accuracies help in understanding how model performs on data not seen and adjusting hyper parameters accordingly.After the training and validation there is a testing accuracy calculation inorder to check model’s performance on new data.
* While doing validation **torch.nograd()** is used to make sure the model does not learn while doing validation and Testing.model.eval() is also used to make sure the model is set to evaluation mode while testing and validation is performed.
* The **libraries** used are pytorch,numpy,matplotlib,json,tqdm to run ffnn and plot graphs.The different hyperparameters used for tuning are learning rate,hidden dimensions,drop out,batch size.

## RNN (25pt)

**TODO:** Explain briefly how you implemented filled in the incomplete code for RNN.py in the form of code-snippet screenshot (and explanations) in the report. Try to understand what other part of the code is doing, and write your understandings here (especially parts that is functioning differently as compared to FFNN).

A screen shot of a computer program

Description automatically generated

* The inputs are passed via the RNN layer where data is processed step by step. A dropout layer is used to provide help to the model in regularization.The output of the last time step in the sequence is passed through a linear transformation(W) to get a tensor of shape(batch\_size,no of classes).An Addition of logits of the outputs are performed in sumop .The output logits are passed into softmax layer to get log probabilities of each class resulting in the predicted\_vector which is returned .
* Data is tokenized and punctuations are removed before converting them into word embeddings.The program uses a **pretrained word embeddings** from a .pkl file.
* The **training data** is shuffled and processed using a batch size of 32.The optimizer used is **Adam** with a learning rate 0.0005 and weight decay 1e-4. To prevent exploding gradients ,gradients are clipped using torch.nn.utils.clip\_grad\_norm\_().
* This script included **early stopping** mechanism if the validation accuracy did not improve for 3 cycles then the model is stopped.

**Key differences :**

**Architecture :** RNN deals with sequential data capturing the temporal connection between words whereas FFNN treats data as independent features losing order of data

**Gradient Flow :** Rnn has a problem of exploding gradients which is dealt with gradient clipping ffnn does not face this problem because it does not deal with sequential data

**Word Embeddings** : RNN uses word embeddings capturing semantic information whereas FFNN uses a Bag-of-Words approach processing data as a fixed-size vector losing context of data.

**Optimizer :** RNN uses Adam optimizer adapting learning rate and includes weight decay for regularization helping with vanishing gradients.FFNN uses SGD optimimzer with momentum which is simpler.

# Experiments and Results (45pt)

**Evaluations (15pt) TODO:** Explain how you evaluate the models. What metric is used – you can refer to the current implementation.

The metrics used are **Accuracy and loss** for various sets of data,where accuracy measures the amount of right predictions,loss quantifies the confidence of those predictions.

Accuracy = No. of correct Predictions/Total No. of Predictions.

As this is a muti-class classification task,accuracy is a primary indicator of how the model is performing.

The **test accuracy** is a metric used to analyze on how the model is performing on totally unseen data.**The validation accuracy** is used to analyze the need of early stopping and convergence criteria.The loss function used is **Negative Log-Likelihood Loss (NLLLoss)**,where in the lower the loss the better the performance as loss measures the difference between predicted and actual distribution.

A computer screen with text

Description automatically generated

**Results (30pt) TODO:** Apart from the default hyperparameters, try multiple variations (be- tween 1-2 for FFNN and RNN each) of models by changing hidden unit sizes.

**TODO:** Summarize the performance of your system and Put the results into tables or diagrams and include your observations and analysis.

FFNN :

The below is a default one.This is with hidden dimension 10 and epoch 1

A screenshot of a computer

Description automatically generated

Different combinations of parameters of tried of which 3 of them are presented here.results of other combinations are present in results/test1.out

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Observations :**

* Case 3 comes out with best accuracy 0.5813 where better generalization occurred.This suggests that a balance occurred between fitting the training data and regularization with the chosen hyperparameters.
* Case 1 had showed consistent improvements over epochs.Comparitively it showed better stability and steady improvement but did not generalize too well.
* Case 2 presented a higher training accuracy but lower training accuracy indicating potential overfitting of the model.Increased hidden dimensions and epochs might have contributed to a larger gap netween training and validation accuracies.

**RNN :**

Different combinations of parameters of tried of which 3 of them are presented here.results of other combinations are present in results/test2.out

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Observations :**

* Case 1 has the highest accuracy 0.4288 despite its early stopping at 7 epochs.The higher hidden dimensions might have heped model in performing better on unseen data by learning better
* Case 2 : The lower testing accuracy when compared to case 1 implies that it did not generalize well but in general higher dimensions resulted in better performance
* Case 3: It had a very low test accuracy which means that it did not perform well on unseen data.Lower learning rate regularized the model too much ,that early stopping occurred,the model neither learned better nor predicted better.

# Analysis (bonus: 10pt)

## TODO:

* (5pt) Plot the learning curve of your best system. The curve should include the training loss and development set accuracy by epoch.

**FNN** : Graphs of case 3 and case 2 are as follows

A graph with blue and orange lines

Description automatically generated

A graph with blue and orange lines

Description automatically generated

RNN :

Case 1 :

A graph with blue and orange lines

Description automatically generated

* (5pt) Error analysis. List some (one or more than one) error examples and provide some analysis. How might you improve the system?

# Conclusion and Others (5pt)

## TODO:

* Individual member contribution.

I made significant contributions to the project's different components, including, model implementations of RNN and FFNN,hyperparameter and analysis of performance. Actively engaged in model evaluations and tackled issues that arose during the training and optimization phases.

* Feedback for the assignment. e.g., time spent, difficulty, and how we can improve.

The assignment demanded a significant time commitment, but the learning outcomes didn't fully justify the time invested. A considerable portion of the work was spent waiting for model runs to complete before conducting analysis. To enhance the learning experience, minimizing technical difficulties or offering more detailed guidance could help with the learning experience.