USING MACHINE LEARNING FOR SENTIMENT ANALYSIS

A report on training different machine learning models and evaluating final results

Abstract

This report outlines an almost complete machine learning workflow that uses a publicly available dataset. A companion Jupyter notebook is also available for anyone wishing to review the underlying Python code.

Prepared by: Babak Eslamieh (July 12, 2025)

Contents

Deep Learning and Reinforcement Learning

This is the final assessment project for course 5 of IBM Machine Learning Specialization.

About the data

We'll be working with Sentiment Labelled Sentences dataset provided by UCI Machine Learning Repository. This dataset was originally created for the following paper, published in Knowledge Discovery and Data Mining (KDD):

From Group to Individual Labels Using Deep Features (2015), by:

- Dimitrios Kotzias
- Misha Denil
- Nando de Freitas
- Padhraic Smyth

This dataset contains 3000 reviews labelled with positive (1) or negative (0) sentiment. These reviews are collected from three different sources:

- Movie reviews from IMDB
- Cell phone reviews from Amazon
- Food reviews from Yelp

There are 500 positive and 500 negative reviews from each site (collected and labelled in separate text files). These reviews are randomly collected from a larger dataset to contain strictly positive or negative items (i.e., no neutral sentiments). Each text file contains 1000 reviews and each line contains a single review (English text) and its corresponding label (0 or 1), separated by a single tab character.

Objectives

We want to conduct Sentiment Analysis on this dataset. The main objective is to predict a person's sentiment on a movie, cell phone or food by training different models on existing reviews in our dataset.

We're going to use a simple approach for final model evaluation and selection. Using the performance of an initial model as baseline, we'll train several artificial neural networks (ANNs) and try to come up with an optimal model (i.e., ANN architecture and parameters) that outperforms the baseline model.

Data processing

Original dataset contains three text files, each file containing 1000 reviews. The following steps were performed to transform raw data to a form that is suitable and ready for modeling:

- Collecting reviews from all three text files and making an initial DataFrame.
- Preprocessing all reviews by converting to lowercase and removing numbers, punctuations and special characters.
- Carefully detecting emoticons and appending them to the end of each document.
- Removing all stop words using NLTK.
- Building feature vectors using Tf-Idf encoding.

Model training

The modeling process can be summarized as below:

- Training a default Logistic Regression model and using its performance as baseline
- Training several sequential neural networks with different architectures and hyperparameters
- Aggregating train/test metrics from each model

- Deciding if any trained network can beat our baseline performance

Sequential neural networks

Several sequential neural networks were trained using Keras Sequential model. For each model, the following steps were performed:

- Create, compile and train the model, using predefined hyperparameters
- Evaluate the model using both train and test sets and view aggregated results
- Plot train/test accuracy and loss to see if the model is overfitting

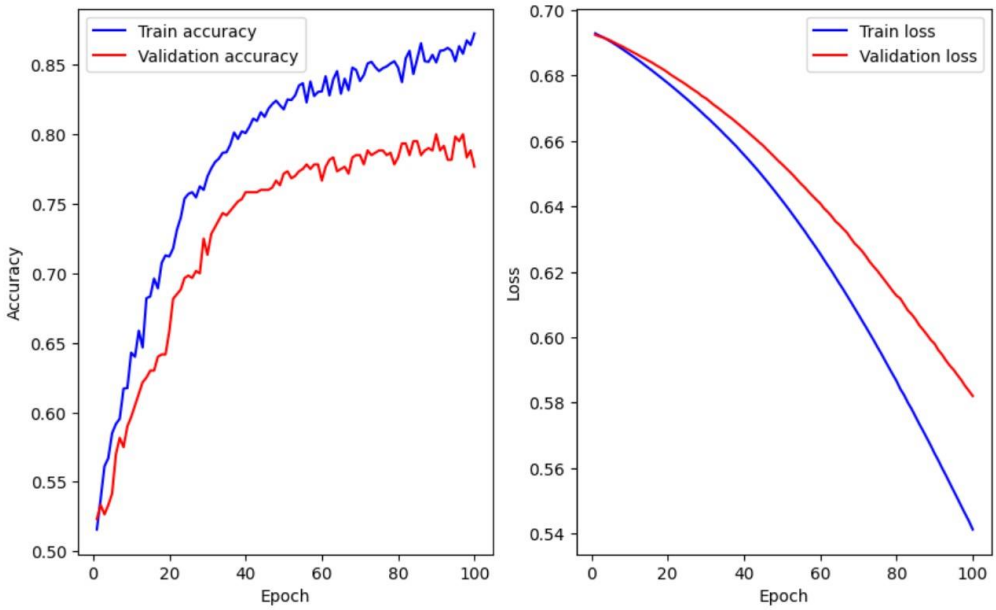
Below is the summary of trained networks:

- Model 1: Single layer, 8 nodes, SGD with lr = 0.01, 100 epochs
- Model 2: Single layer, 16 nodes, SGD with lr = 0.01, 100 epochs
- Model 3: Single layer, 8 nodes, Adam with lr = 0.001, 20 epochs
- Model 4: Single layer, 64 nodes, Adam with lr = 0.0001, 20 epochs
- Model 5: Two layers (64, 16), SGD with lr = 0.001, 300 epochs
- Model 6: Two layers (64, 16), SGD with lr = 0.01, 150 epochs
- Model 7: Four layers (128, 64, 64, 8), SGD with lr = 0.005, 100 epochs
- Model 8: Two layers (64, 16), SGD with lr = 0.005, 200 epochs, leaky relu activation (0.15)
- Model 9: Two layers (64, 16), SGD with lr = 0.005, 150 epochs, relu activation

Note: in the above model configurations, 'lr' stands for 'learning rate'

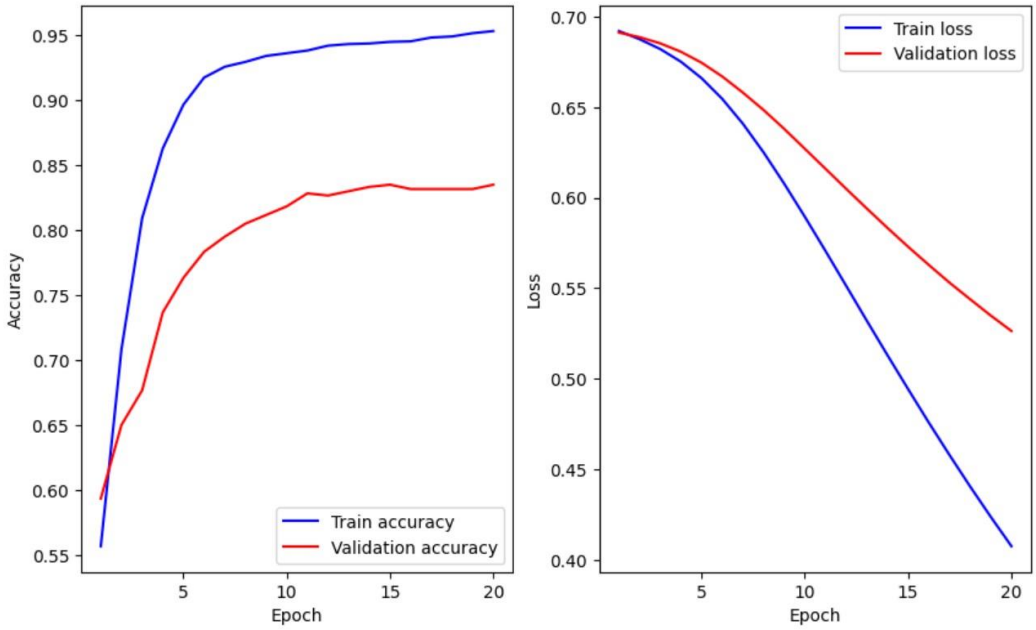
Most models displayed rather similar behavior in terms of performance level and overfitting. Some sample results will be presented next.

Model 2 (performance summary)



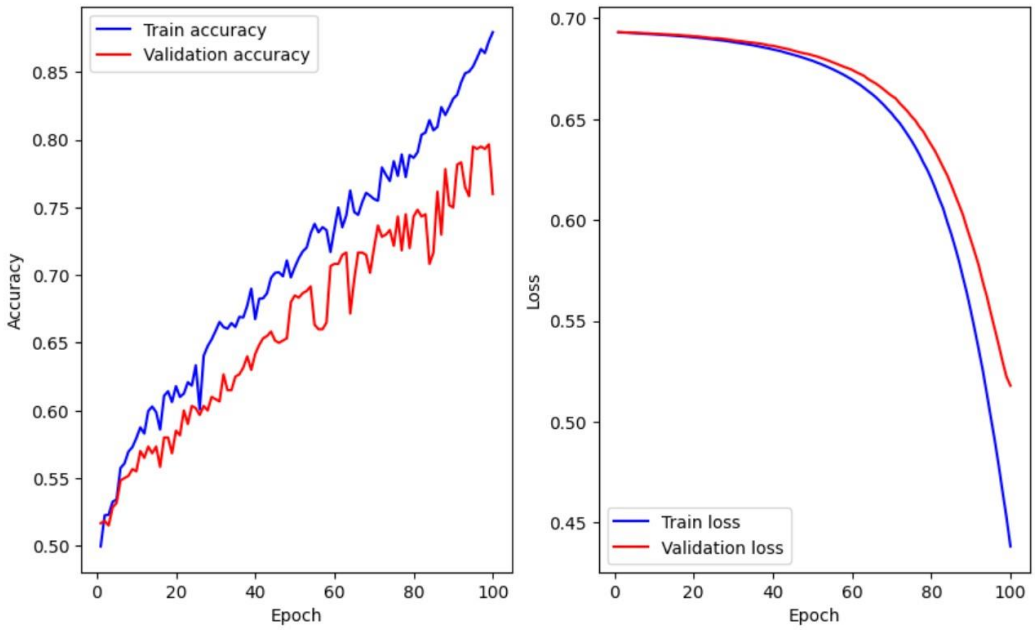
From the above plots, one can verify that this model starts overfitting around epoch 40 (train accuracy = 80%, validation accuracy = 76%)

Model 4 (performance summary)



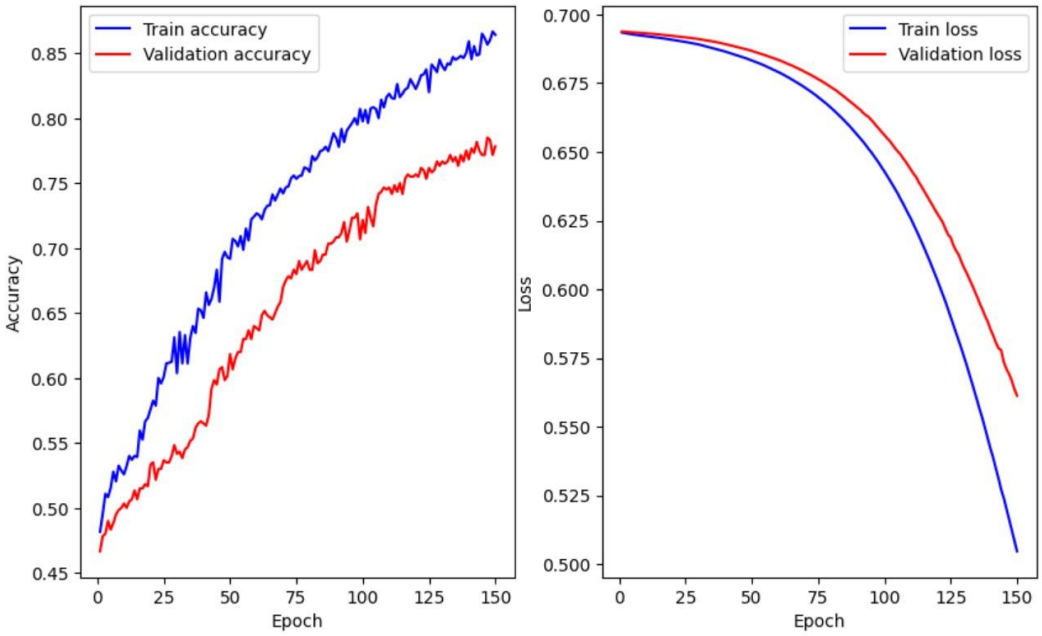
This model starts overfitting around epoch 5 (train accuracy = 90%, val. accuracy = 76%)

Model 7 (performance summary)



This model starts overfitting around epoch 80 (train accuracy = 80%, val. accuracy = 72%)

Model 9 (performance summary)



This model starts overfitting around epoch 25 (train accuracy = 60%, val. accuracy = 54%)

Best model

Before starting with all the above neural network architectures and hyperparameters, we found a baseline accuracy of around 83% using default Logistic Regression model. Although relatively good, we expected to go much higher with our sequential network models.

Unfortunately, none of our trained models achieved a satisfactory performance with this dataset. Finally, our **Baseline Logistic Regression** model proved to be the best model. However, it's going to need careful tuning to be really useful for normal sentiment analysis.

Insights and key findings

After careful study of the above performance summaries, we can make the following remarks about training sequential neural networks for our dataset:

- Training sequential networks on medium-sized datasets is relatively fast. With 2400 training samples and approximately 5000 bag of words features, all network architectures were trained rather effortlessly (1 second per epoch). No parallelization was required.
- Adam optimizer is normally the preferred choice in most use cases, but it didn't work well in our dataset.
- The Stochastic Gradient Descent (SGD) optimizer worked much better than Adam, but still didn't provide acceptable model performance.

- With all hyperparameters available for tuning neural network (number of hidden layers, number of units per layer, optimizer, etc.), it's a little difficult to come up with a suitable configuration.

Next steps

To achieve better results, several improvement ideas and future plans can be considered, as summarized in the following sections.

Possible flaws

- From a semantic viewpoint, our dataset contains reviews about 3 different and unrelated subjects (movies, cell phones and restaurants). Although neural networks are generally famous for fast learning, variety of subjects may have confused all of our models.
- Sentiment analysis might actually be more effective when existing reviews and sentiments are all about one or more related subjects.

Action plan

- Regarding subject variety, we can retry all models using only one set of reviews (e.g., movies, cell phones or restaurants only)
- Due to large size of our bag of words feature space, we didn't perform any correlation analysis. Many correlated features (e.g., similar words) may have caused problems in learning process of our models.
- Applying Principal Component Analysis (PCA) or Non-negative Matrix Factorization (NMF) on our processed dataset may provide better results.

Acknowledgements

Thank you so much for taking the time to review and grade this project.

I'd also like to thank our brilliant instructors (especially Dr. Joseph Santarcangelo) for their excellent learning materials. It's been a real pleasure taking this course.