IMDB Reviews Analysis

CS 273 Final Project report

Chiyu Cheng
77749072

Yi Zhou
18656445

Xiaomi Liu
68122189

I. INTRODUCTION

Sentiment analysis, as a classic natural language processing (NLP) problem, refers to using the text analysis, computational linguistics to systematically identify and quantify affective states of subjective information. It's widely used for customer material such as survey and feedback. Basic sentiment analysis includes classifying the polarity of a sentence to figure out whether it's positive or negative or its emotional states such as "happy", "sad", "angry".

In fact, "What other people think" has always been an important piece of information for most of us during the decision-making process. Long before the invention of the World Wide Web, It's quite common for us to ask our friends to recommend a good restaurant, a fantastic tourist attraction or a good job opportunity. That is the base of the recommendation system. Nowadays, lots of companies such as Yelp, Netflix, IMDB are trying to use a large amount of data in order to provide their customers with a more accurate recommendation. However, it's not quite that common when those data are regular and easy to be understood by a computer. In fact, most of the data those companies collect from social media like Twitter, are just sentences like "Jane Austen's books madden me so much that I can't conceal frenzy from reader" which is quite difficult for the computer to understand whether this person like Jane Austen or not. The early study used keywords flagged by volunteers manually to determine the emotional-affective of a sentence. Whereas, such method is unlikely to cover all the keywords when datasets are becoming larger and sentences become more complex. During recent years, with the improvement of machine learning especially deep neural network, more and more advanced techniques have been proved useful to teach a computer to learn what a natural sentence means.

In this final project, several states of art techniques including RNN, CNN, and random forest are used to analyze and predict the IMDB reviews dataset provided by Andrew Maas [1]. To gain a better understanding of this topic, each person of this group uses a different method to evaluate the IMDB reviews dataset. Chiyu practices the CNN technique, Xiaomi is responsible for the random forest method, and Yi explores the LSTM method and he is also responsible for the introduction of this project. The remaining of this paper continues as follows. First, it describes and analyzes the IMDB reviews dataset from different perspectives. Then, the detailed introductions of the techniques mentioned above are presented along with the evaluations of each method with varieties parameter options. Finally, the paper ends with potential future directions and a conclusion.

II. DATA ANLYSIS

According to the readme file [1], 50,000 labeled movie reviews, which splits equally into a 25k training dataset and 25k test dataset, are included in the IMDB reviews dataset. The negative reviews and positive reviews are also distributed evenly. Under the [train/] and [test/] directories, each review is stored in a file with the name format "index-score." For the purpose of analyzing, a python script is used to extract the content and score from the text files under the [train/] and [test/] directories. For each review, the script converts each vocabulary to one number by a vocabulary dictionary. The script also pads the length of review number arrays to certain number depends on the requirements. The detailed explanation about how to apply the given dataset to a specific method is presented in the following sections.

III. CNN

A. Introduction

The convolutional neural network is known as one sub-class of deep learning. It consists of multiple layers of perceptions to forward and backward data. The name of CNN is inspired by the animal biology process where the connection between real neurons is similar to the connection between each node in CNN. There are several applications of CNN such as image reorganization and classification, medical image analysis, and NLP (natural language processing). In this project, a basic CNN model is used to analyze and predict the IMDB reviews dataset. In the rest of the section, it first shows the detailed introduction of different types of layers in CNN. Then, it describes the CNN model used in this section, evaluates its performance based on different parameter options.

1.1 Convolutional Layer

The main function of the convolutional layer is applying the convolution operation on the input layer and forward the convoluted result to the next layer. In general, the input layer is treated as a two-dimension array that consists of different numbers. If the input is a picture, then each number presents the color of the pixel in the picture. In this project, the input is a review that concludes a sequence of vectorized vocabularies. Figure 1[2] shows that the convolution operation is using the “feature” or “kernel,” which is a small receptive field, to scan the input 2D number array. Each kernel is a small 2d number array, and each element presents the weight. For each scan, it times the original value of the scanned part of the input and the weight values of each kernel, and the result is kept as the output of the convolutional layer. Padding strategy can be used to avoid losing input by padding 0s on the boundary of input. After repeating the convolution process entirely in the input 2d array, a filtered 2d array is created to the output.

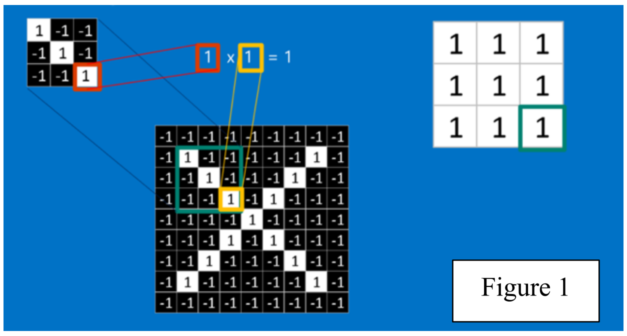


Figure 1

1.2 Pooling Layer

In case of large input, pooling layer is used to shrink it while still be able to keep the essential information of the input. There are two types of pooling strategies. The max pooling strategy in figure 2[2] is using a sliding window to scan the input 2d number array and extract the maximum number from it as the output. After applying the sliding windows operation on the entire input array, a new smaller array is constructed to be the output. Another strategy is average pooling, which is similar to max pooling except it calculates the average of the values in the sliding window as the output instead of keeping the maximum value.

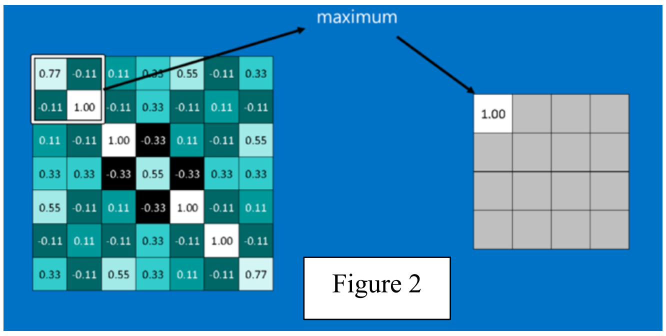


Figure 2

1.3 Rectified Linear Layer

This is an activation layer that converts all the negative value of the input array to 0. With this method, the non-linear properties of the CNN increase without changing the kernels of the convolutional layer.

1.4 Fully Connected Layer

Fully connected layer translates the input array to votes by converting the 2d input array into a 1d array. In the IMDB review dataset, there are only two categories, thus each element only needs to vote whether it is positive or negative. And there are different weights for the vote of each element. Initially, the weights are randomized and change later during the back-propagation process.

1.5 Back Propagation

A completed CNN can be stacked by the layers mentioned above. The input array is passed linearly through different layers and converted to the wanted format based on the weights of each layer. Then, a decision is generated by CNN. However, this is not the end. The essential part of learning ability is provided by the back-propagation operation, which helps CNN to learn how to adjust the kernels and weights of different layers. Based on the provided labels, CNN can know how good the prediction is. Then it backward the output back for the same layer to adjust the features and weight to see if the results are better. After performing the operation to every element of kernels, CNN can be improved slightly better.

B. Evaluation of CNN

The basic idea of applying the CNN technique is converting the reviews to an image-like 2D number array. As mentioned in the dataset analysis section, the raw data with vocabularies review is not suitable for CNN. It is necessary to convert it to the number version by a vocabulary dictionary. In addition, converting the number of version reviews to vectors improves the usability of CNN during the analyzing process. This project uses the embedding function provided by TensorFlow to embed the review to a vector with shape (256,16,1), where 256 is the fixed length and 16 is the vector size for each word. The structure of the CNN is, one ConvLayer(8 kernels(4x4), stride1), ReluLayer, MaxPoolingLayer(2x2, stride 2), another ConvLayer(16 kernels(4x4x8), stride1), ReluLayer, MaxPoolingLayer(3x3, stride 3), FullyConnectedLayer(1x1x2). Based on this simple model, several experiments are designed to test the effectiveness of different parameters including learning rate, momentum, batch size, and weight decay. The measurement is based on the prediction accuracy of each test case on the fixed size of the training data.

A. Learning rate

Learning rate, a hyperparameter of CNN that controls how much the updating step should adjust the weights of features. Lower the learning rate may help CNN to take care of every local minimum. However, the slow learning rate may delay the converging speed as well. In figure 3, the validation accuracy reaches the peak when learning rate = 0.01 and starts to fall down when learning rate increases. The reason is high learning rate makes the CNN skips much more information needed[3].

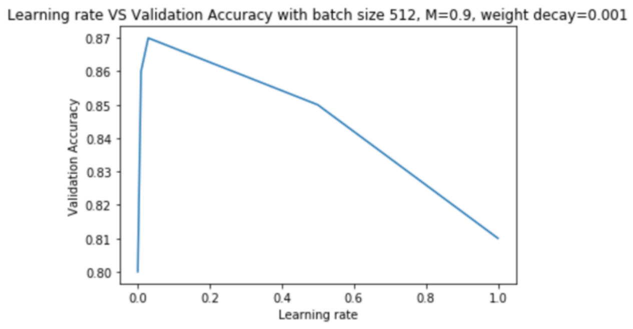


Figure 3

B. Momentum

Similar to the learning rate, momentum is another hyperparameter of CNN to avoid stuck on the local minima. It is a value that increases the size of the step for each epoch. Large momentum means it can reach the global minimum quicker. However, over large momentum may result in skipping the g Figure 4 mum in an unexpected large step, which may delay the time to converge. In figure 4, it shows that the validation accuracy rate increases when momentum increase. It could happen if the minimum is not skipped by large momentum step[3].

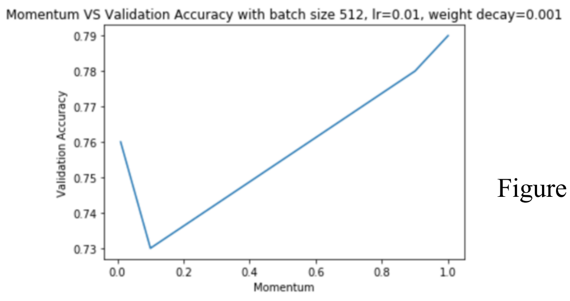


Figure 4

C. Weight decay

Unlike other two, weight decay is the hyperparameter that prevents the oversized weights of CNN. It reduces the weights to decay to zero if there are no other updates. In figure 5, it shows that the validation accuracy rate gets a peak when the weight decay = 0.003 and keeps decreasing when the weight decay increases. The high weight decay may lead to over-reduce the weights of features, which loses the meaning of updates [3].

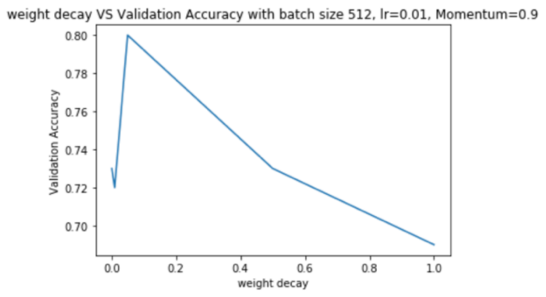


Figure 5

C. Conclusion

In this section, it presents the detailed definition of different types of layers in CNN, describes the CNN model used in this final, and analyze the performance of the CNN model by modifying different hyperparameters. According to the collected data, the CNN model performs best when the learning rate = 0.01, momentum = 0.99, and weight decay = 0.01. In this next section, the random forest method is used to interpret the dataset in a different way

IV. Random Forest

A. Introduction

The ensemble is a kind of a machine learning technique that solves the defects inherent in a model or a certain set of parameters by combining many predictors together. These predictors can learn from each other and avoid limitations. Random forests are a kind of ensemble, which integrates many decision trees into forests and combine them to predict the result.

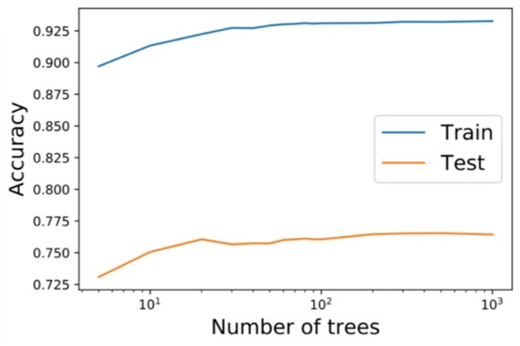
Random forests have many distinct features. It is unexcelled in accuracy among algorithms. It runs efficiently on large databases. It can handle thousands of input variables without variable deletion. It generates an internal unbiased estimate of the generalization error as the forest building progresses. It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing [3].

We use Scikit-learn to build a random forest classifier. For the split criterion, we use Gini impurity instead of entropy. Because Gini impurity is easier to compute. For a large dataset like IMDB, we prefer Gini impurity.

B. Hyperparameters Optimization

1) Number of trees

The number of trees has a great influence on the performance of random forest. If the number is too small, the error of classification would be very high. If the number is too big, the forest can become too complex and take too much time to build. To decide on the number, we computed the accuracy of both training and test data with different numbers of trees.

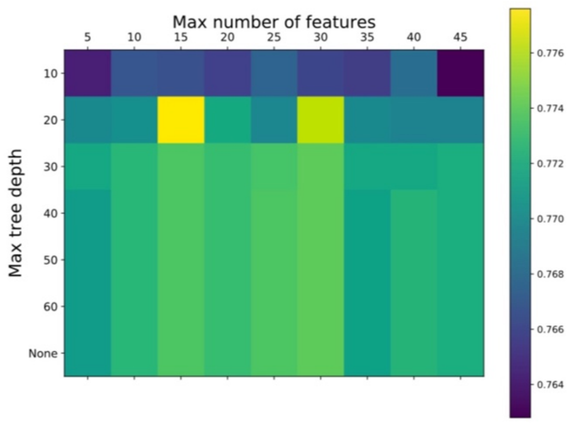


We can see that the improvement on the accuracy is smaller and smaller once we have more than 100 trees. At last, we chose 200 as the number of trees.

2) Max-depth & max-features

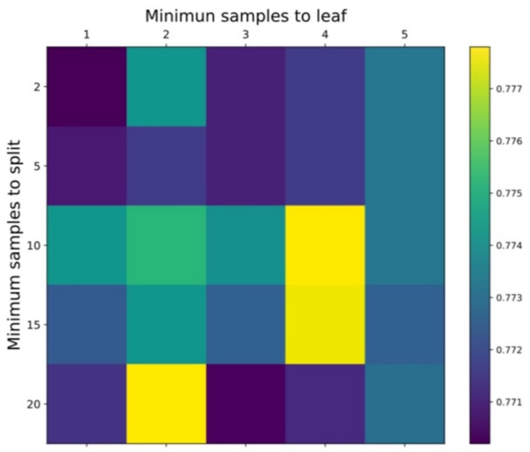
The default value of max-depth is “None”, which means no limit on the depth of trees. The default value of max-features is also “None”, which means to consider all features. But for a dataset with large data size and large feature size, we’d better set a max number of depth and a max number of features to control the time needed for building the random forests.

We evaluated a wide range of values for each hyperparameter to narrow our search by using Scikit-Learn’s RandomizedSearchCV method[4]. Based on the results, we try to find the optimal max-depth around 10, and the optimal max-features around 25. There is no significant improvement on accuracy when max-depth is bigger than 20.



3) Min-parent & min-leaf

Min-parent is a condition on whether to split further. If the sample number of a node is smaller than min-parent, it won’t be split. The default value is 2. Min-leaf is the minimum sample number of leaf nodes. The default value is 1. For a dataset with large data size and large feature size, we’d better increase these two hyperparameters.We also used Scikit-Learn’s RandomizedSearchCV method to narrow our search for optimal min-parent and min-leaf. Based on the results, we try to find the optimal min-parent around 15, and the optimal min-leaf around 4. This shows that the model reaches the highest accuracy rate when min-leaf equals to 4 and min-parent equals to 10.



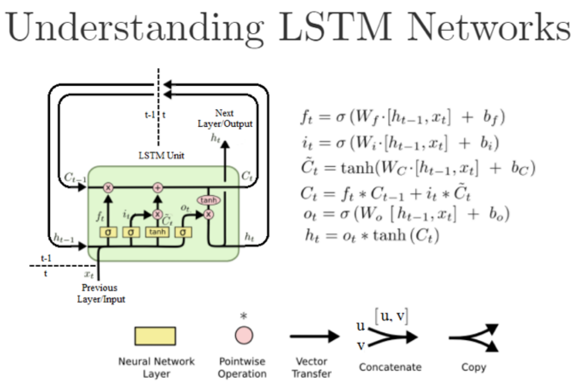
C. *Conclusion*

We set each hyperparameter to its optimal value: number of trees = 200, split criterion = Gini impurity, Bootstrap = False, max-depth = 20, max-features = 15, min-parent = 10, and min-leaf = 4. The optimal test accuracy of our random forest classifier is 0.763. During this process, we found that tuning the hyperparameters of random forests cannot bring a big improvement in accuracy compared to other classifiers.

V. LSTM

A. *Introduction*

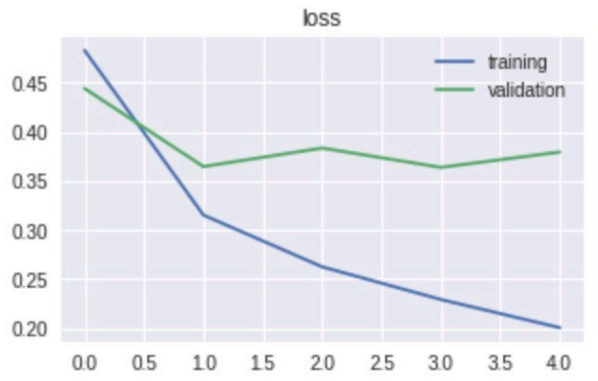
LSTM is a traditional way to do NLP task such as sentiment analysis. In theory, classic (or "vanilla") RNNs can keep track of arbitrary long-term dependencies in the input sequences. The problem of vanilla RNNs is computational (or practical) in nature: when training a vanilla RNN using back-propagation, the gradients which are back-propagated can "vanish" or "explode" , because of the computations involved in the process, which use finite-precision numbers. RNNs using LSTM units partially solve the vanishing gradient problem, because LSTM units allow gradients to also flow unchanged. A traditional LSTM unit looks like the right figure.

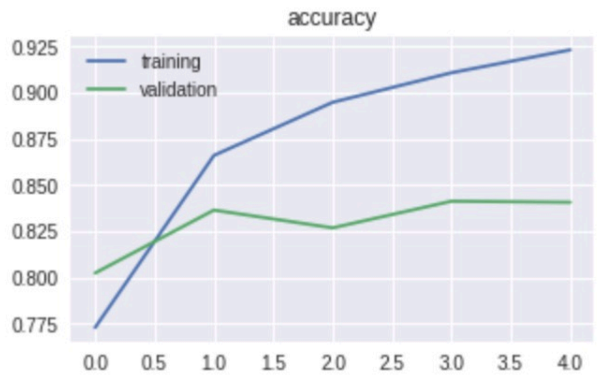

Understanding LSTM Networks
t-1 | t
Next Layer/Output
LSTM Unit
C_{t-1}
C_t
h_{t-1}
h_t
x_t
Previous Layer/Input
$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$
$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$
$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$
$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$
$o_t = \sigma(W_o [h_{t-1}, x_t] + b_o)$
$h_t = o_t * \tanh(C_t)$
Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy

B. *Evaluation*

At first, for the LSTM model, the parameters we use is embedding-size = 200, max-words = 200, dropout = 0.5 and epochs = 5. The architecture of model is an embedding layer, followed by a

dropout layer, then the LSTM layer with 32 units and finaly a Dense layer to output the results.
We choose 5000 of training examples as our validation data and here right according to the epochs shows the accuracy and the loss of my model. We finally get an accuracy of 83.89% on our test data set.





C. Conclusion

Obviously, LSTM model can gain the best accuracy in all of the three models, and that's because it can save the previous state of a sentence and combine it with the current word and then generate a better understanding of the meaning of a sentence. This experiment shows the great power of LSTM on NLP tasks.

VI. Conclusion And Future Direction

In this paper, three different machine learning techniques, CNN, random forest, and CNN are used to analyze the IMDB dataset. For each method, we provide the introduction and definition of the basic concept and evaluate it under different hyperparameters. Due to the limited time, we are not able to compare the performance of each method under the same environment. In the future, we will develop the ensemble method to take the prediction of all three methods, and research the optimized weights of each method. With the help of ensemble methods, the prediction accuracy and speed may be improved.

References

[1] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). Learning Word Vectors for Sentiment Analysis. *The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).*

[2] http://brohrer.github.io/how_convolutional_neural_networks_work.html

[3] O'Shea, Keiron & Nash, Ryan. (2015). An Introduction to Convolutional Neural Networks. ArXiv e-prints.

[4] Information Resources ManagemAent Association, Artificial Intelligence: Concepts, Methodologies, Tools, and applications, Information Science Reference, 2016

[5] https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74

[6] Gers F A, Schmidhuber J, Cummins F. Learning to forget: Continual prediction with LSTM[J]. 1999.

[7] Xingjian S H I, Chen Z, Wang H, et al. Convolutional LSTM network: A machine learning approach for precipitation nowcasting[C]//Advances in neural information processing systems. 2015: 802-810.