UNIVERSITY OF VICTORIA

Department of Electrical and Computer Engineering

ECE 503 Optimization for Machine Learning

PROJECT REPORT

Title: Sentiment Analysis on Movie Reviews

Date of Experiment: 16 Dec 2023

Report Submitted on: 17 Dec 2023

To: Dr. Lei Zhao

Name: Lepeng Zhou

V-number: V01045967

Abstract

Sentiment analysis is a relatively simple and common task in natural language processing(NLP), where the goal of such a project is to classify the polarity of a text as positive or negative. In this project, various machine learning methods have been applied to perform sentiment analysis on a corpus of movie reviews from IMDb. However, due to the nature of text data, before proceeding to a traditional machine learning task, I am also required to perform some preprocessing tasks. In this project, many methods such as cleaning useless components in text, embedding text to vector form, and dimension reductions are applied. Later I treated this as a binary classification problem, where reviews with ratings of 7, 8, 9, or 10 were labeled as positive, and reviews with ratings of 1, 2, or 3 were labeled as negative. and then fed them into different classifiers, such as linear, linear with L2 regularization, polynomial with degree 2, neural network, support vector machine (SVM), and decision tree. I then evaluated the performance of each classifier using various benchmarks, ie accuracy, confusion matrix. I discussed the possible reasons for these outcomes and limitations, also I suggested some directions for future work.

1 Motivation

In this section I will talk about the motives and real life applications behind this project and some background information about the origin of the dataset used.

1.1 Background

Since its public debut in late 2022, ChatGPT 3.5 has captured widespread attention, marking a significant shift in the perception of artificial intelligence (AI). This transition from viewing AI as a futuristic concept to recognizing its present-day capabilities has been remarkable. ChatGPT, with its proficiency in understanding human language, has greatly accelerated various daily tasks.

This rapid evolution sparked my interest in AI, particularly in the field of Natural Language Processing (NLP). The research, understanding, and implementation of Large Language Models (LLMs) like ChatGPT involve complexities far beyond conventional tasks such as training a basic neural network to regress a trend or developing reinforcement learning agents for solving mazes or playing chess. These LLMs are built upon foundational abstract research, which adds layers of complexity and sophistication to their development.

My enthusiasm for NLP was further fueled by undertaking some basic courses in the subject. Now, I have the opportunity to apply and demonstrate my knowledge through this project. While this project does not parallel the scale of work done by organizations like Uvic researchers or even OpenAI, it represents a crucial step in my academic journey as a master student.

Looking ahead, my goal is to delve into more complex models such as Long Short-Term Memory (LSTM) networks and transformers, to address some of the limitations I've encountered in this project. However, due to the scope of this course, I should have not incorporated LSTM and transformer models. Nevertheless, these advanced models remain a key area of interest for my future studying endeavors.

1.2 Real life application of sentiment analysis

The practical applications of this project extend into many areas of everyday life and professional domains. Some typical areas I will talk about here, to indicate the importance of my project topic choice. [1]

Sentiment analysis aids in adaptive customer service by providing real-time insights into customer mindsets, thereby reducing churn and improving service quality. With the vast amount of data generated online, sentiment analysis facilitates the efficient sorting of this data, revealing critical insights about human sentiment quickly and cost-effectively.
Furthermore, by analyzing sentiments in real-time, especially on social media, businesses can promptly respond to public opinions, adjusting strategies to prevent negative repercussions.

2 Preparations

In this section I will talk about the dataset I have selected and used, and the transformation of text data(unstructured) to a vectorized data(structured) that is suitable for a machine learning task.

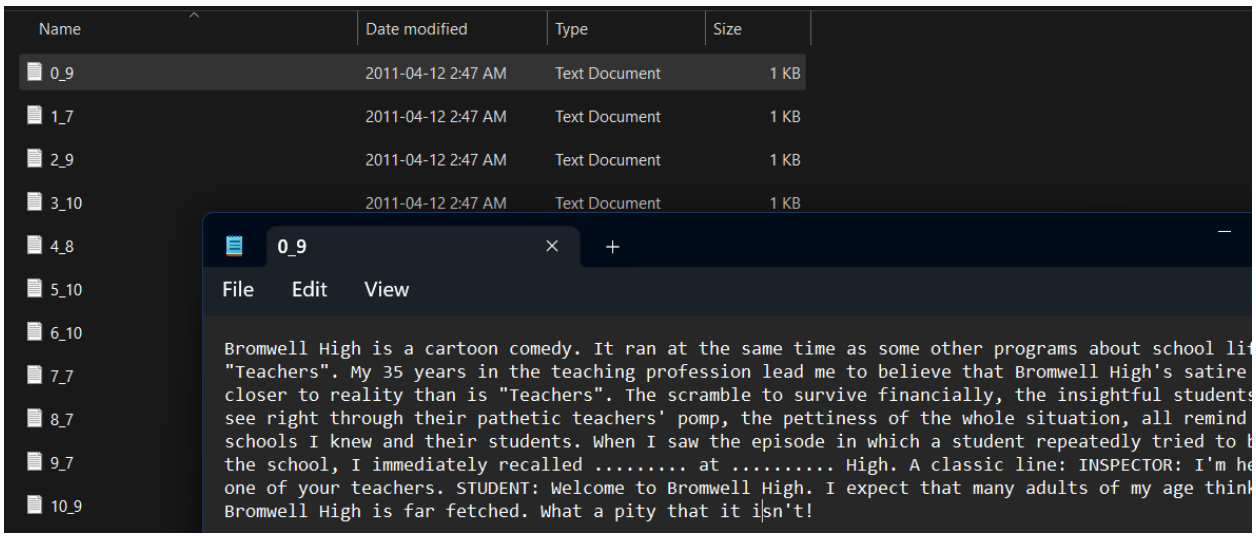
2.1 Unstructured data

Unstructured data, such as text, fundamentally differs from structured datasets like the Iris, Breast Cancer, Boston Housing datasets, or image datasets like MNIST. Unlike structured data, which is organized in a tabular form with defined rows and columns, unstructured data lacks this predefined format. Text data, in particular, is more complex due to its variability in length,

syntax, and semantic meanings, making it challenging yet essential to transform for effective machine learning applications. [2]

2.2 Raw Text Dataset

The dataset used in this project is the Large Movie Review Dataset from Stanford University, specifically designed for binary sentiment classification. This dataset provides a substantial volume of data, including 50000 highly polar movie reviews for training and testing, along with additional unlabeled data for unsupervised learning tasks. For this project, I focus solely on the positive and negative reviews, later combining them into a single comprehensive pandas DataFrame for further processing. The data mainly has 2 important fields, first is the “content”, which contains an entire typical movie review we can easily find on imdb, it looks something like “I have gone to this movie @9’oclock with my ……. I will definitely recommend it to my friends!!!!! :)”. As you may already noticed there are many content that are not helpful and suitable for the task of NLP, like internet emojis, unconventional use of punctuations and marks, etc. therefore in the next step what we will need to do is to remove informations that are not useful in the text and separate words by space. [3]



How the dataset looks like in its original form

As you can see from figure 1, when unzipping the package downloaded you have in total 50000 text documents, each named in “index_rating” format. For example, if text document number 86 has a rating of 7, it will have a name of “86_7”. Inside each text document is just the plain text of comment.

2.3 Processing and Cleaning

The transformation, also known as processing and cleaning of raw text data into a structured format suitable for machine learning involves several steps, as outlined in the provided Python code. Here's an overview of the data processing pipeline:

First, I need to concatenate all text documents together into a pandas dataframe. Each row has a content and rating field.

index	content	rating
0	Bromwell High is a cartoon comedy. It ran at the same tin	9
0	Story of a man who has unnatural feelings for a pig. Start	3
1	If you like adult comedy cartoons, like South Park, then th	7
1	Robert DeNiro plays the most unbelievably intelligent illite	1
2	Bromwell High is nothing short of brilliant. Expertly scripte	9
2	I saw the capsule comment said "great acting." In my opir	1
3	"All the world's a stage and its people actors in it"--or so	10
3	If I had not read Pat Barker's 'Union Street' before seeing	4
4	FUTZ is the only show preserved from the experimental th	8
4	This fanciful horror flick has Vincent Price playing a mad r	4
5	I came in in the middle of this film so I had no idea about	10
5	I basically skimmed through the movie but just enough to	3

Data in the form of pandas dataframe

Second, a cleaning of the raw text data is required, as you can see the raw text contains many punctuations and stop words, which are not necessarily useful for the NLP task. All letters should be in lowercase, and each word should be tokenized, or separated simply by a space.

cleaned_content
went saw movie last night coaxed friends mine ill admit reluctant see knew ashton kutcher able comedy
mr costner dragged movie far longer necessary aside terrific sea rescue sequences care characters us gh
boyfriend went watch guardianat first didnt want watch loved movie definitely best movie seen sometir
pale imitation officer gentleman chemistry kutcher unknown woman plays love interest dialog wooden
yardstick measuring movies watchability get squirmy start shifting positions noticing butt sore film long
seems ever since every two three years get movie claims next officer gentleman yet one movie lived clai
many movies think see movie like cant count sure seemed like movie makers trying give hint reminded
wow another kevin costner hero movie postman tin cup waterworld bodyguard wyatt earp robin hood
movie sadly underpromoted proved truly exceptional entering theatre knew nothing film except friend

Data after being cleaned and tokenized

Then the TfidfVectorizer from Scikit-learn is used to convert the raw text data into a TF-IDF matrix. TF-IDF stands for “Term Frequency-Inverse Document Frequency”, This step translates each row of text into a vector of numerical values, each entry in the vector represents a word, and each entry’s numerical value is calculated according to how often that word appears in that row of text and all texts, reflecting the importance of each word in the context of the entire dataset. TF-IDF is also a feature engineering technique, aimed to extract information from the data that are hard for machine learning algorithms to extract. The math behind the TF-IDF is intuitive. For each word(term) in our vocabulary(all words that appear in all the documents) in a document(one row of data), we calculate a TF and IDF score for it. If a word is not present in one document, then its value in that document is simply zero.

$$TF = \frac{\text{number of times the term appears in the document}}{\text{total number of terms in the document}}$$

$$IDF = log(\frac{\text{number of the documents in the corpus}}{\text{number of documents in the corpus contain the term}})$$

Then, the TF-IDF is simply the product of TF and IDF. Namely, TF-IDF = TF * IDF.

The advantage of such method that better than just simple count how many times a term appears is that It takes account into how important a term is within a document relative to the entire collection of documents. The importance of a term is high when it occurs a lot in a given document and rarely in others. In other words, if a word “bad” appears 10 times in one document but one time each in 10 other documents, that one document will have a very high score of “bad” but others will have a very low score. This addresses some of the issues that when occasionally saying “not bad” means “good”. But if “bad” appears so many times in one document, the author probably really means “bad”.[4] https://www.learndatasci.com/glossary/tf-idf-term-frequency-inverse-document-frequency/

The choice of vocabulary size is set to 3000, this is the average amount of words in English that are commonly used in daily life. However a 200 word review is impossible to have all 3000 words each to appear once. Thus the TF-IDF results in an extremely sparse matrix.

BZS	BZT	BZU	BZV	BZW	BZX	BZY	BZZ	CAA	CAB	CAC	CAD
acttrojans	acttwo	actual	actualities	actualities	actuality	actualitypi	actualityth	actualizati	actualize	actuall	actually
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0.07996
0	0	0	0	0	0	0	0	0	0	0	0.046493
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0.063664	0	0	0	0	0	0	0	0	0.041292
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0.06438	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0.096357	0	0	0	0	0	0	0	0	0.031248

Each row represents a review, each column represent a TF-IDF score, the size of such matrix is 50000 by 3000

Therefore, dimensionality reduction is obviously needed. Given the high dimensionality of the TF-IDF matrix, a TruncatedSVD method is incorporated to reduce the number of features while retaining essential information. This step is crucial for improving the efficiency and performance of the subsequent machine learning models. The math behind such a method is the same as PCA, we take the eigenvalue and vectors and rank them from high to low, then we choose only to keep some of them, then map each row of data to its new space represented by those selected eigenvectors. As for my experiment of resulting vector size, I can have either a 1000 dimensions vector explaining 62% of variance or a 600 dimensions vector explaining 49% of variance. I choose to use the 600 dimension vector as the trading of 13% of variance does not worth that 400 more dimension of data.

After all the processing, now we have a 50000 by 600 matrix and ready for machine learning.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	
-0.04563	-0.08729	0.004498	0.02842	0.031552	-0.04873	-0.01478	-0.01769	-0.04107	0.013823	0.020607	0.006746	-0.01729	-0.00758	0.011192	-0.01865	-0.0102	-0.00584	-0.02816	-
-0.04894	-0.04071	0.071969	-0.02265	-0.06492	-0.0801	-0.02111	-0.06446	0.022926	0.018285	-0.02738	-0.00983	-0.06458	0.037901	0.008854	-0.00339	0.013338	0.007433	-0.00607	(
-0.01736	-0.07708	0.014076	-0.0178	0.00943	0.023192	0.081317	-0.03676	0.07665	-0.01007	0.071764	-0.00475	0.048037	0.000672	0.007403	-0.00583	-0.01853	-0.03911	0.06926	(
-0.08315	-0.00998	-0.04388	-0.00089	0.038359	-0.00162	-0.03841	-0.00789	0.007102	0.030959	-0.01113	-0.05037	-0.00542	-0.00451	0.023769	-0.01128	-0.01861	0.004548	-0.02572	-
0.043814	0.020781	-0.07069	0.00969	0.033695	0.109571	-0.00412	0.00914	0.009421	-0.01507	0.035889	-0.01309	-0.01752	0.0356	-0.06903	-0.06305	-0.04903	0.022931	-0.01405	(
-0.01712	-0.06949	-0.018	0.024765	-0.01431	0.024716	-0.02961	-0.03358	-0.021	0.006591	0.016058	-0.03388	0.012544	0.003435	0.002536	0.001701	-0.01084	-0.02709	-0.00563	
-0.0312	0.039957	-0.03725	0.003716	0.006469	0.053188	-0.05809	0.022159	-0.02805	0.006519	-0.02005	-0.03713	0.03494	-0.04721	-0.00926	-0.01313	-0.09599	0.049422	0.044292	-
-0.07409	-0.02092	0.007417	-0.00296	-0.03583	0.002308	-0.03527	0.022631	0.016454	0.017159	-0.00349	0.007619	0.023095	0.022641	-0.03021	-0.00119	-0.02281	0.004315	-0.02371	(
-0.09156	-0.05439	-0.03674	-0.00135	-0.0629	0.009754	0.095491	0.00015	0.027526	0.01871	0.006271	0.006611	0.029033	0.029811	0.027892	-0.02302	0.002471	0.013691	0.028135	(
-0.01914	0.021042	-0.04094	-0.05181	-0.033	0.031975	0.077257	0.034901	0.008512	0.019526	0.034863	0.006617	0.009687	0.077541	-0.00023	-0.03989	-0.01556	0.025316	-0.02	(
0.002589	0.016718	-0.03975	-0.04342	-0.02704	-0.05845	-0.01963	0.053822	0.041028	0.059345	-0.00604	0.017265	-0.02303	0.058191	0.011629	-0.03042	0.056708	0.048268	0.007933	
-0.09503	0.070911	-0.04516	-0.00389	0.01524	0.109316	-0.03547	-0.01989	-0.02711	-0.06147	0.044934	-0.04134	-0.01238	-0.03732	-0.00287	-0.0118	-0.00909	-0.11855	0.101864	-
-0.07884	-0.07789	-0.019	-0.0159	-0.03984	-0.03854	-0.01255	-0.01083	0.053549	-0.0151	-0.00861	0.021177	0.033079	0.021256	0.007228	-0.01805	-0.0016	0.002582	-0.01883	-
0.025331	-0.00028	-0.05468	-0.09327	-0.04584	0.092005	0.042469	0.063855	-0.02685	0.061472	-0.01308	-0.00949	0.025129	0.010725	-0.03769	-0.00823	-0.06386	0.07408	0.003926	-
0.339883	0.006207	-0.18291	-0.15022	0.033041	0.044569	0.066355	0.007842	0.069915	-0.10509	-0.00259	-0.0185	-0.04357	0.015493	-0.03325	0.010178	0.056629	0.015995	0.029825	(
0.106462	0.037578	-0.00474	-0.03429	-0.04029	-0.00273	-0.03918	0.005195	-0.01055	0.002734	-0.01576	0.035302	-0.04156	0.009526	-0.01753	0.059611	0.004692	0.01514	-0.01969	(
-0.04966	-0.06623	-0.04783	0.00251	-0.006	-0.0025	-0.02633	-0.02462	-0.01821	-0.00157	0.017157	0.006416	0.027822	0.024657	0.017137	-0.01266	0.016935	-0.02784	-0.01502	(
-0.03077	-0.00823	-0.03186	-0.03301	-0.06638	0.072738	-0.01675	0.016273	-0.00018	0.043295	-0.01948	0.012935	0.040062	0.013397	0.007199	0.014783	0.023361	0.057435	0.040173	-
-0.07146	0.049998	-0.00299	-0.00327	-0.0294	0.02893	-0.02439	0.074292	-0.04111	0.014589	0.030654	-0.05815	0.002807	0.09447	0.011954	-0.02815	-0.00807	0.001252	0.019917	(
0.041045	0.053203	-0.06226	-0.05437	0.02553	-0.0222	-0.05144	0.107461	0.001988	-0.00645	0.014052	0.007151	0.052801	-0.01332	0.02596	0.026177	-0.04935	0.019637	-0.06461	-
0.0752	-0.0119	-0.04152	-0.02952	-0.07349	0.004857	0.031222	0.01919	0.031817	-0.00879	-0.03968	-0.05775	0.006703	0.004212	-0.00636	0.001335	-0.0226	0.034802	-0.00436	(
0.086597	0.04925	0.076162	-0.09702	-0.04225	0.00282	0.01593	0.057147	0.00177	0.022769	-0.03561	-0.04924	0.004963	0.036573	-0.05328	0.041289	0.033468	0.045247	-0.06271	-
-0.0545	-0.02076	-0.03034	0.026277	-0.0459	0.060462	0.004324	-0.01217	0.012501	-0.06708	0.060863	0.010111	-0.02336	-0.02605	0.000744	-0.03605	-0.01419	-0.01259	0.06307	-
-0.06162	-0.06616	-0.03107	0.114649	-0.03583	0.023545	-0.00914	-0.01114	-0.00713	-0.03879	-0.03533	-0.02683	0.041621	0.032314	-0.02128	0.012733	0.033065	0.010094	-0.03151	(
-0.07439	-0.05911	-0.02429	0.04705	-0.03958	-0.01845	0.029457	-0.06351	-0.05217	-0.05645	-0.04619	0.030966	0.064619	0.028028	0.055328	-0.01392	-0.00487	0.014427	-0.0395	-

Data after dimension reduction.

3 Machine learning Methodologies

In this section, I will discuss the machine learning techniques used to classify the data in my sentiment analysis project. All models are trained on a random selection of 40000 documents and test on 10000 documents.

3.1 Optimization

Also known as the solver, it's used to find the minimum of objective function. The Limited-memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) solver is chosen for this project's all machine learning methods, is an optimization algorithm from the family of quasi-Newton methods. It's designed for bound-constrained optimization, typically for problems where constraints are limited to specific bounds.

LBFGS approximates the original BFGS algorithm, which utilizes a full inverse Hessian matrix to guide the search in the variable space. However, unlike BFGS that stores a dense n×n approximation to the inverse Hessian (n being the number of variables in the problem), LBFGS stores only a few vectors that represent this approximation implicitly. This approach significantly reduces memory requirements, making LBFGS particularly well-suited for optimization problems with many variables. [5]

$$d_k = -S_k \nabla f(x_k) = -S_k g_k$$

2) BFGS:

a) Initial $S_0 = I$,

b) compute:

$$\delta_k = x_{k+1} - x_k$$

$$\gamma_k = \nabla f(x_{k+1}) - \nabla f(x_k)$$

c) Update S_k

$$S_{k+1} = S_k + \left(1 + \frac{\gamma_k^T S_k \gamma_k}{\gamma_k^T \delta_k}\right) \frac{\delta_k \delta_k^T}{\gamma_k^T \delta_k} - \frac{\delta_k \gamma_k^T S_k + S_k \gamma_k \delta_k^T}{\gamma_k^T \delta_k}$$

Figure 6. BFGS algorithm updating rules[6]

1) Reconstruct the updating $S_k \to S_{k+1}$ by extracting S_k:

$$S_{k+1} = S_k + \left(1 + \frac{\gamma_k^T S_k \gamma_k}{\gamma_k^T \delta_k}\right) \frac{\delta_k \delta_k^T}{\gamma_k^T \delta_k} - \frac{\delta_k \gamma_k^T S_k + S_k \gamma_k \delta_k^T}{\gamma_k^T \delta_k}$$

$$= S_k + \frac{\delta_k \delta_k^T}{\gamma_k^T \delta_k} + \frac{\gamma_k^T S_k \gamma_k \delta_k \delta_k^T}{(\gamma_k^T \delta_k)^2} - \frac{\delta_k \gamma_k^T}{\gamma_k^T \delta_k} \cdot S_k \cdot I - I \cdot S_k \cdot \frac{\gamma_k \delta_k^T}{\gamma_k^T \delta_k}$$

$$= I \cdot S_k \cdot I + \frac{\delta_k \delta_k^T}{\gamma_k^T \delta_k} + \frac{\delta_k \gamma_k^T}{\gamma_k^T \delta_k} \cdot S_k \cdot \frac{\gamma_k \delta_k^T}{\gamma_k^T \delta_k} - \frac{\delta_k \gamma_k^T}{\gamma_k^T \delta_k} \cdot S_k \cdot I - I \cdot S_k \cdot \frac{\gamma_k \delta_k^T}{\gamma_k^T \delta_k}$$

$$= \left(I - \frac{\delta_k \gamma_k^T}{\gamma_k^T \delta_k}\right) \cdot S_k \cdot \left(I - \frac{\gamma_k \delta_k^T}{\gamma_k^T \delta_k}\right) + \frac{\delta_k \delta_k^T}{\gamma_k^T \delta_k}$$

2) Replace S_k by I:

$$S_{k+1} = \left(I - \frac{\delta_k \gamma_k^T}{\gamma_k^T \delta_k}\right) \left(I - \frac{\gamma_k \delta_k^T}{\gamma_k^T \delta_k}\right) + \frac{\delta_k \delta_k^T}{\gamma_k^T \delta_k}$$

Figure 7. LBFGS algorithm is just did not choose to keep the S matrix [6]

3.2 logistic classification

Logistic regression serves as a benchmark in our analysis. The mathematical foundation of logistic regression revolves around the logistic function, which models the probability of a binary outcome. Given the feature dimensionality of 600, a linear model seems adequate for initial experimentation.

$$\underset{\hat{\boldsymbol{w}}}{\textbf{minimize}} \quad E_L(\hat{\boldsymbol{w}}) = -\frac{1}{P}\sum_{p=1}^{P}\log\left(\frac{1}{1+e^{-y(\hat{\boldsymbol{w}}^T\hat{\boldsymbol{x}})}}\right) = \frac{1}{P}\sum_{p=1}^{P}\log\left(1+e^{-y(\hat{\boldsymbol{w}}^T\hat{\boldsymbol{x}})}\right)$$

[7]

With this setup, the model demonstrates commendable performance, with close training and test accuracies, indicating a well-fitted model. Introducing an L2 regularization term, aiming to prevent overfitting, yielded similar results. This suggests that the model is not overfitting, and we have maximized the potential of the linear approach.

$$\hat{f}(\boldsymbol{x}) = f(\boldsymbol{x}) + \frac{\mu}{2}\|\boldsymbol{x}\|_2^2$$

Loss function with l2 regularization term

```
Time: 10.521372556686401
[[4311  720]
 [ 568 4401]]
              precision    recall  f1-score   support

           0       0.88      0.86      0.87      5031
           1       0.86      0.89      0.87      4969

    accuracy                           0.87     10000
   macro avg       0.87      0.87      0.87     10000
weighted avg       0.87      0.87      0.87     10000

Accuracy_test: 0.8712
Accuracy_train: 0.881375
```

Logistic classification without l2 regularization term

```
Time: 10.321167945861816
[[4233  674]
 [ 598 4495]]
              precision    recall  f1-score   support

           0       0.88      0.86      0.87      4907
           1       0.87      0.88      0.88      5093

    accuracy                           0.87     10000
   macro avg       0.87      0.87      0.87     10000
weighted avg       0.87      0.87      0.87     10000

Accuracy_test: 0.8728
Accuracy_train: 0.88175
```

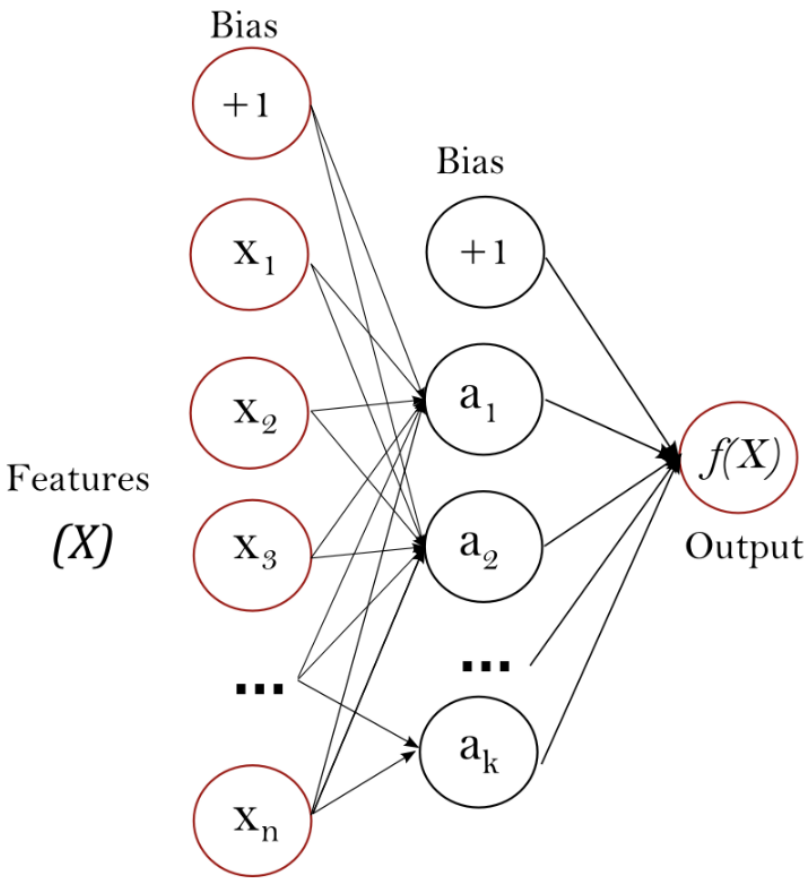
Logistic classification with l2 regularization term

3.3 Polynomial Features

Exploring higher dimensions, such as a polynomial transformation of degree 2, theoretically increases the feature space to an exponential number of 600. For example, if an input sample is two dimensional and of the form [a, b], the degree-2 polynomial features are [1, a, b, a^2, ab, b^2]. However, this approach led to computational limitations, including a system crash, indicating its impracticality for our dataset. [8]

3.4 Neural network

For the neural network implementation, we delve into the complexities of multi-layer perceptrons (MLPs). The MLPClassifier from the sklearn package was chosen for its compatibility with our pipeline. Selecting the appropriate hyperparameters for neural networks is an intricate process, blending art and science.



One inner layer of Neural Network [9]

Neural network essentially shares the same loss function with logistic methods. Here is a generalized loss function called softmax, if classes equal to 2, softmax is the same as logistic loss function.

$$\text{softmax}(z)_i = \frac{\exp(z_i)}{\sum_{l=1}^{k} \exp(z_l)}$$

[9]

Neural networks, known for their ability to model complex, non-linear relationships, require careful balancing of model complexity and dataset size. Initially, a two-layer neural network with dimensions 400x200 was tested but showed signs of overfitting, with a significant disparity between training and testing accuracies. Besides, the model is too complex so even the max iterations have been achieved still it does not converge.

```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
Time: 442.0388627052307
[[4395  660]
 [ 735 4210]]
              precision    recall  f1-score   support

           0       0.86      0.87      0.86      5055
           1       0.86      0.85      0.86      4945

    accuracy                           0.86     10000
   macro avg       0.86      0.86      0.86     10000
weighted avg       0.86      0.86      0.86     10000

Accuracy_test: 0.8605
Accuracy_train: 0.9143
```

Neural network of inner layer size of 400 by 200

Subsequent experiments involved using a single hidden layer of 400 neurons. While the time and training set performance improved, testset performance is even worse. Finally, introducing an L2 regularization term (alpha = 10) enhanced the test accuracy, underscoring the effectiveness of regularization in this context.

```
Time: 220.58008790016174
[[4183  729]
 [ 797 4291]]
              precision    recall  f1-score   support

           0       0.84      0.85      0.85      4912
           1       0.85      0.84      0.85      5088

    accuracy                           0.85     10000
   macro avg       0.85      0.85      0.85     10000
weighted avg       0.85      0.85      0.85     10000

Accuracy_test: 0.8474
Accuracy_train: 1.0
```

Neural network of inner layer size of 400 with no l2 regularization

```
Time: 151.0697159767151
[[4440  592]
 [ 625 4343]]
              precision    recall  f1-score   support

           0       0.88      0.88      0.88      5032
           1       0.88      0.87      0.88      4968

    accuracy                           0.88     10000
   macro avg       0.88      0.88      0.88     10000
weighted avg       0.88      0.88      0.88     10000

Accuracy_test: 0.8783
Accuracy_train: 0.9037
```

Neural network of inner layer size of 400 with l2 regularization, alpha = 10

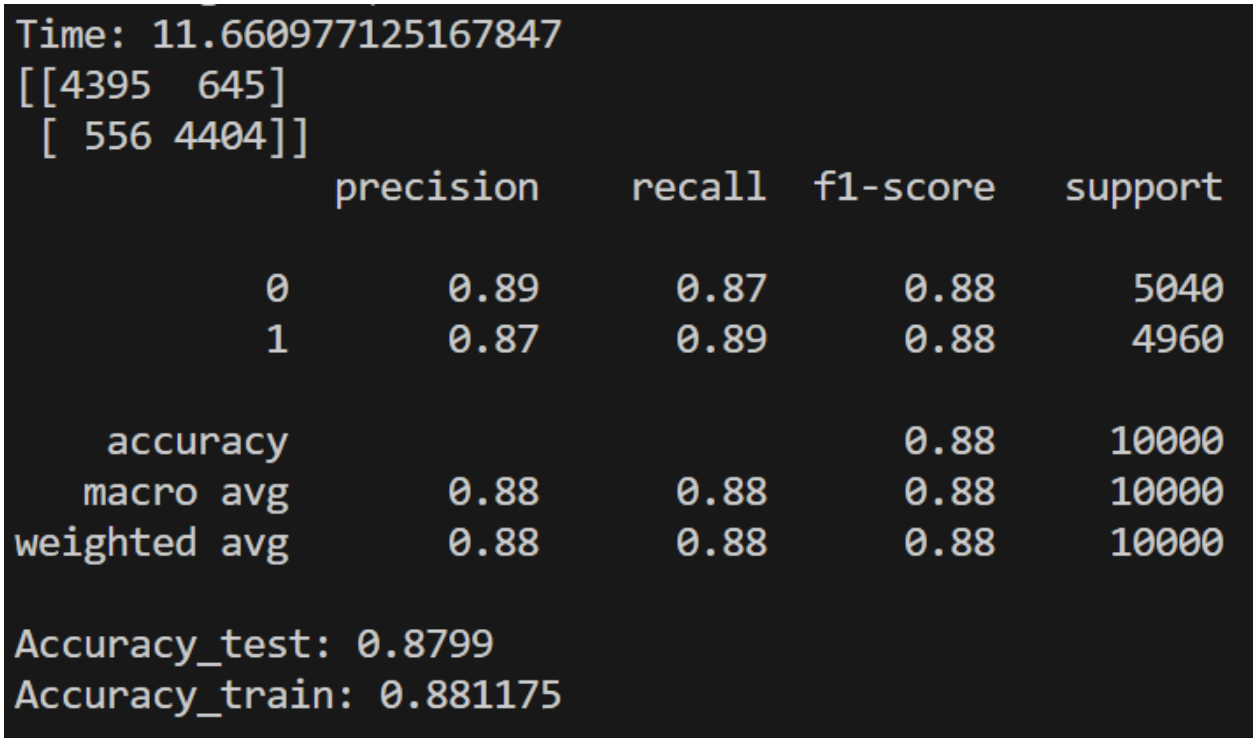
3.5 Support Vector Machine

In this section, I will apply Support Vector Machines (SVM) to my dataset. SVM is a powerful and versatile supervised machine learning algorithm, suitable for both classification and regression tasks. It is particularly effective in high-dimensional spaces like this dataset. The loss function used by SVM is essentially the hinge loss. Hinge loss is different from log loss as it can be zero for those correctly classified.

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \max(0, 1 - y_i(w^T \phi(x_i) + b)),$$

[10]

For the initial SVM model, we opted for a linear kernel. This choice is based on its simplicity and effectiveness in dealing with linearly separable data. The scikit-learn library's SVM implementation was utilized for this task.



SVM with linear kernel

As we see SVM performs as fast as other methods and reaches similar accuracy, and no sign of overfitting. And requires no tuning on hyperparameters.

4 Results

This section discusses the results obtained from different machine learning models applied to our dataset, comparing their performances and drawing conclusions from their differences.

4.1 Performance and conclusion

Logistic Regression: The application of logistic regression demonstrated that regularization wasn't necessary, as there was no significant overfitting observed. The model achieved an accuracy of around 87%, which seems to be the upper limit for this method on our dataset. The training process was relatively fast (approximately 10 seconds), and the model's interpretability and ease of mapping for visualization are notable advantages.

Neural Network: In contrast, the neural network model appeared to be overly complex for our problem, leading to overfitting. Additionally, the training time was considerably longer (about 200 seconds), making it less efficient for our task. The complexity and computational demands of neural networks did not translate into improved performance for our specific dataset.

Support Vector Machine: The SVM, fundamentally different in its loss function from logistic regression, also performed quickly and efficiently, similar to logistic regression, with no extensive need for hyperparameter tuning. The SVM's robustness in handling high-dimensional data was evident, yet it did not substantially outperform logistic regression in terms of accuracy.

Based on these observations, logistic regression emerges as the ideal choice for this project due to its balance of speed, interpretability, and satisfactory performance.

4.2 Discussion and future work

The consistent accuracy cap of around 87% across different models, despite their varying complexities, suggests that the bottleneck might lie in the preprocessing of text data rather than the models themselves. Our current feature engineering approach primarily relies on the bag-of-words method, which overlooks the sequential or temporal relationships inherent in natural language. This limitation leads to potential biases; for instance, sentences like "this is good!" and "is this good?" are treated similarly in our vectorization approach. Moreover, the subjectivity in document ratings introduces inherent biases, as different individuals may rate the same document differently. This aspect of the data highlights the limitations of our labels.

Many other techniques like RNNs could be a strategic move for future models. RNNs have the ability to take into account the temporal sequence of texts, which is a crucial aspect in natural language processing. Unlike bag-of-words, RNNs can understand the relationships between words and flow of sentences, potentially leading to more accurate predictions. Also, exploring word or sentence embeddings could offer a more nuanced representation of text data. Embeddings capture not just the presence of words but also the contextual relationships between them, which could address some of the current limitations. Integrating such models might provide a more sophisticated understanding of the text data, moving beyond the bag-of-words approach.

5 References

[1] P. Mishra, “What is sentiment analysis: Concepts, use cases & applications,” Gramener Blog, https://blog.gramener.com/sentiment-analysis-guide/ (accessed Dec. 17, 2023).
[2] “Unstructured Data,” Wikipedia, https://en.wikipedia.org/wiki/Unstructured_data (accessed Dec. 17, 2023).
[3] A. Maas, “Large Movie Review Dataset,” Sentiment Analysis, https://ai.stanford.edu/~amaas/data/sentiment/ (accessed Dec. 17, 2023).
[4] Author: Fatih Karabiber Ph.D. in Computer Engineering, Fatih Karabiber Ph.D. in Computer Engineering, E. R. Psychometrician, and E. B. F. of LearnDataSci, “TF-idf - term frequency-inverse document frequency,” Learn Data Science - Tutorials, Books, Courses, and More,

https://www.learndatasci.com/glossary/tf-idf-term-frequency-inverse-document-frequency/ (accessed Dec. 17, 2023).
[5] “Limited-memory BFGS,” Wikipedia, https://en.wikipedia.org/wiki/Limited-memory_BFGS (accessed Dec. 17, 2023).
[6] 1 quasi-newton methods, https://studentweb.uvic.ca/~leizhao/403-2022-fall/BFGS.pdf (accessed Dec. 18, 2023).
[7] 1 lecture note on 2023/10/13, https://studentweb.uvic.ca/~leizhao/403-2022-fall/note12.pdf (accessed Dec. 18, 2023).
[8] “Sklearn.preprocessing.PolynomialFeatures,” scikit, https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html (accessed Dec. 17, 2023).
[9] “1.17. neural network models (supervised),” scikit, https://scikit-learn.org/stable/modules/neural_networks_supervised.html (accessed Dec. 17, 2023).
[10] “1.4. Support Vector Machines,” scikit, https://scikit-learn.org/stable/modules/svm.html#svm-kernels (accessed Dec. 17, 2023).