Part One: Classifying Review Sentiment with Bag-of-Words Features

1. Background and Data Inspection

In this part of the project, we are given customer reviews collected from imdb, amazon and yelp with the goal of assigning correct sentiment to a given review where negative sentiment is being labeled as 0 and positive sentiment is being labeled as 1. x_train and x_test are provided with 2400 and 600 reviews respectively. For y data, y_train is provided while y_test is not. However, $predicted\ y\_test$ can be compared with y_test through leaderboard on Gradescope which will calculate the error rate as well as AUROC value.

The training data contains $1200\ (50\%)$ positive reviews and $1200\ (50\%)$ negative reviews. To better understand the classification of the data, 3 positive and 3 negative reviews are randomly chosen from x_train:

Positive (1):

- yelp, Just as good as when I had it more than a year ago!
- imdb, "I recommend this for EVERYONE who loves film, movies, anything...A Work of Art! "
- amazon, Keep up the good work Amazon!!

Negative (0):

- yelp, As a sushi lover avoid this place by all means.
- imdb, I'm still trying to get over how bad it was.
- amazon, Not enough volume.

2. Pipeline – BoW Feature Generation

The goal of this section is to first present the pipeline adopted for this project's bag of words pre-processing and feature generation and then justify some of the decisions made.

2.1 Pipeline

After loading the csv file of x data using numpy, the "text" column would be extracted and converted to list while the website name column would be left out for further processes.

Then, each review will undergo the following transformation in the tokenizing process:

1. Substitute any character that is not "A-Z", "a-z" or apostrophe with space
2. Convert all letters into lower case
3. Split the string into list of words base on whitespace
4. Remove word that is in standard English "stopword" and not "not or "no" or does not end in "n't"
5. Apply $PorterStemmer$ to simplify words to their stems when possible

For example, the output after each step described in the previous page for *"I recommend this for EVERYONE who loves film, movies, anything... Can't wait to use it"* would be:

1. I recommend this for EVERYONE who loves film movies anything   Can't wait to use it
2. i recommend this for everyone who loves film movies anything   can't wait to use it
3. ['i', 'recommend', 'this', 'for', 'everyone', 'who', 'loves', 'film', 'movies', 'anything', "can't", ' wait', 'to', 'use', 'it']
4. ['recommend', 'everyone', 'loves', 'film', 'movies', 'anything', "can't", 'wait', 'use']
5. ['recommend', 'everyon', 'love', 'film', 'movi', 'anyth', "can't", 'wait', 'use']

The declared TfidfVectorizer will then be fitted to the training x data and after fitting, the same transformation would be applied to x test. For the given x_train in this project, 3457 features are extracted from 2400 samples of reviews.

2.2 Justification

- *Tokenizer Step 1*
  Apostrophe is kept because if it is replaced by whitespace then words such as “can’t” would become “can t”. After splitting it would be interpreted as [“can”, “t”] which can no longer be distinguished from “can” and loses its negative connotation. On the other hand, dash is replaced because in reviews examined, dash not only serves to connect words but are also used by people to demonstrate the beginning of an explanation. Under this circumstance, dash itself does not carry any positive or negative meaning. In addition, for words such as “well-design”, with dash being replaced, “well” still carries the positive sentiment. Therefore, since reviews’ meanings are not lost with dash being replaced, dashes are not preserved during substitution. In addition, numbers are replaced because a single number without context does not help to determine the sentiment of a review and by disregarding numbers, feature size is also reduced.

- *Tokenizer Step 4*
  Adaption has been made to the standard English “stopword” imported from nltk because it also includes words such as “not”, “no”, as well as abbreviated words ending in “n’t”. If the original stopword is adopted, then after removing stopwords, “I like it” and “I don’t like it” would produce the exact same list of words and the negative sentiment from the latter sentence is lost. By whitelisting words including “not”, “no” and “n’t”, important negative sentiments are preserved while words that do not contribute to the analysis of sentiment such as “the” and “I” are effectively removed.

- *Tokenizer Step 5*
  Stemmer is applied to ensure that words originating from the same stem but have different tenses or singular/plural forms are unified. For instance, “loves”, “loved” and “love” all share the same stem of “love”. This reduces the number of features and increases the accuracy of distinguishing similarity among different reviews. For x_train, with the adoption of PorterStemmer, number of features is reduced from 4368 to 3457.

- *TfidfVectorizer*
  Inverse document frequency is used because not only does it normalizes data to be within the range of 0 to 1, it also takes frequency of a word's appearance into account where the more frequently a word is, the lower its relative value and less important it becomes.

3. Models

In this section, three models will be explored, including logistic regression, neural network (MLP) and SVM. For each model, one of its hyperparameters will be tuned and results will be analyzed.

3.1 Logistic Regression

Parameters

For the logistic regression, C-parameter is the chosen hyperparameter to be tuned. "liblinear" solver is adopted instead of the default "lbfgs" because scikit-learn documentation suggests that "liblinear" is a good choice when the dataset is small[1]. Since x_train for this part of the project only contains 2400 rows of data, it is suitable for "liblinear". All the other parameters have default values and max iteration is not increased because no convergence warning is raised by using the default value to construct the model.

Cross-Validation

For this project, 5-fold cross validation is adopted because 80% to 20% train-validate size is one of the most commonly used splitting ratio and has also been suggested in previous homework.

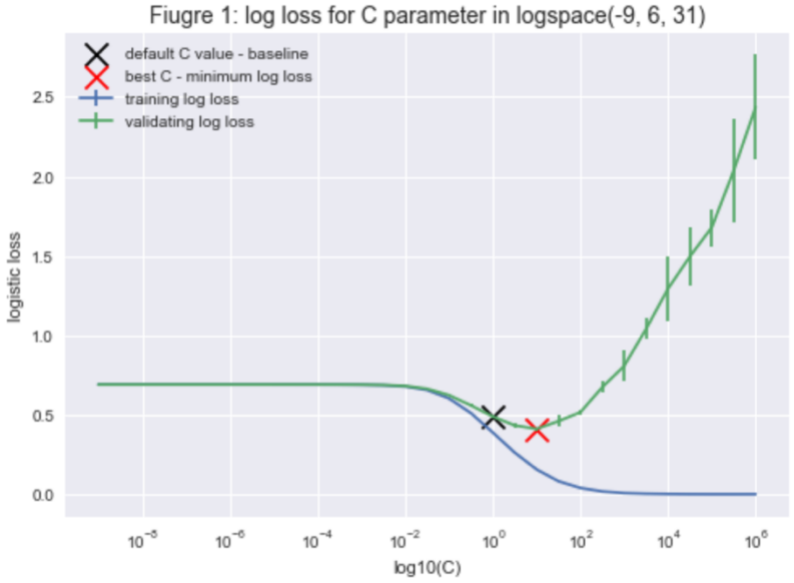
Before exploring any C value, a $KFold$ object (imported from $sklearn.model\_selection$) is declared with the number of fold equals 5 and shuffling turned on. The shuffle is turned on because when inspecting the input, positive and negative sentiment comes in clusters where a few hundred of positive reviews followed by a few hundred of negative reviews. By shuffling, this reduces the possibility of positive-negative training data imbalance after splitting.

For each C value, it would be fitted to the model 5 times (5-fold cross-validation) with the indices of testing and training data in one particular fold determined by the $split$ method[2] of the $KFold$ object declared. In each model, the model is fitted to the 80% training data and validated using the rest 20% and its training and validating log loss in each fold would be recorded. After completing the 5-fold cross-validation for a given C-value, training log loss average, validating log loss average, training log loss standard deviation and validating log loss standard deviation would be computed and recorded.

[1] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
[2] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html

C-Parameters

C is a float representing the inverse of regularization strength and smaller value means greater regularization strength[3]. In order to find the best C value, the original x_train is fitted to 31 logistic regression models each with a different C-parameter regularly spaced between log-space of -9 and 6. The range of log from -9 to 6 is adopted because it has been suggested in previous homework assignments as well as part one of the current project. In addition, by exploring this range, a minimum C value of 10 is identified, shown in figure 1 below. The convexity of the validation log loss produced also reassures that this is a reasonable range as log loss decreases before reaching 10 and increases afterwards, rendering 10 to be the minimum within the range.



Discussion

As shown by the figure above, uncertainty for validating data across different folds generally increases as C value increases. When C equals 10, its calculated standard deviation is 0.0105 and that for C equals 10^6 is 0.329, suggesting that models with higher C value is not as stable. On the other hand, the uncertainty for training log loss values across different folds remain so low throughout that they are not discernable in the figure. Even though training log loss uncertainty also increases with higher C values, its standard deviation for C equals 10^6 is only 0.00170.

For parameters examined, model performs the best when the C value is set to 10, as shown by the minimum point for the validating log loss (0.4142) plotted above where validating log loss decreases before C reaches 10 and increases afterwards. Within the chosen range of C-parameters, overfit is shown for C values larger than 10 as training log loss continues to decrease while testing log loss starts to increase. This may be due to the fact that regularization strength decreases as C value increases and models become more prone to overfit as regularization is relaxed.

[3] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

3.2 Neural Network (MLP)

Parameters

For MLP, the number of neurons in the hidden layer is the chosen hyperparameter to be tuned. The number of hidden layer is chosen to be one because the book "Introduction to Neural Networks for Java" by Jeff Heaton suggests that one hidden layer is sufficient to approximate any function that contains a continuous mapping from one finite space to another.[4] In addition, maximum iteration is increased to 400 because for some models tested, the default 200 default is not enough for models to converge. All the other parameters have default values and solver is not changed to "lbfgs" which according to the scikit-learn documentation performs well on smaller dataset[5] because after experimenting with both "adam", the default solver, and "lbfgs", "adam" is able to render much lower log loss.

Cross-Validation

A similar 5-fold cross-validation process is adopted as described in section 3.1 for logistic regression with the change that instead of exploring different C values, different number of neurons for the single hidden layer is being explored and MLP models are built instead of logistic regression models.

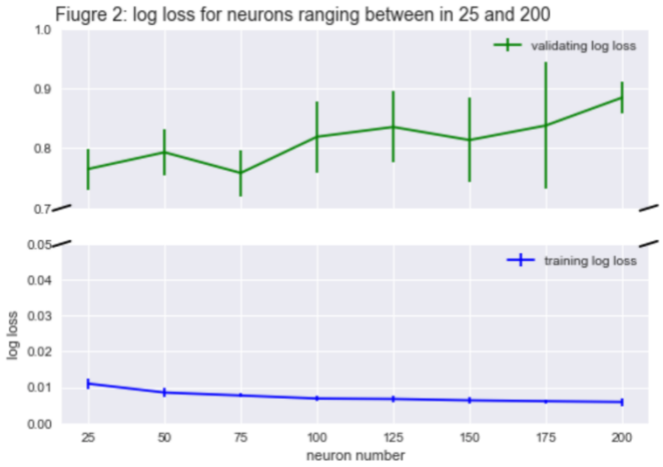
For details of the cross-validation process and data recorded, please view sub-section *Cross Validation* under *3.1 Logistic Regression*.

Hidden Layer Neuron Number

8 different neuron numbers are examined, ranging between 25 and 200 with a step size of 25. This is chosen because some articles suggest that the number of hidden neurons should be between the size of the input layer and the size of the output layer. In addition, because the size of input layer and output layer varies quite a lot for this dataset and the number of features exceeds the number of training data available, number inclining towards the smaller ones are chosen to be examined to avoid overfitting. Furthermore, the end range of the exploration stops with 200 because as shown in the figure below, a minimum validating log loss occurs when neuron number in the single hidden layer equals 75 and validating log loss increases afterwards, suggesting that this is a reasonable range.

[4] https://stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-neural-netw

[5] https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html



Discussion

As shown by the figure above, the standard deviation calculated for validating data log loss across different folds for all neuron numbers explored are all quite noticeable with the general trend that standard deviation is larger for higher neuron numbers. In addition, validating log loss has larger standard deviation when compared with training data of the same neuron number, suggesting they are less stable in result.

For the number of neurons examined, model performs the best when the neuron number equals 75 which has a validating log loss of 0.7586. This is significantly higher than the minimum validating log loss obtained by tuning C-parameter in logistic regression models which is 0.4142. The second lowest log loss occurs for neuron number equals 25, suggesting that MLP models for this dataset tends to perform better with smaller number of neurons.

Overfit is shown within the examined range of neuron numbers. As presented in figure 2, as number of neurons increases beyond 75, log loss for training data continue to decrease while that for validating data begins to increase, suggesting overfit in models. This may be partially due to the fact that the number of training data size is small, and the number of feature vector is larger than the number of training data available. On account of these factors, models would incline towards overfit with larger number of neurons.

3.3 Supporting Vector Machine (SVM)

Parameters

For SVM, C-parameter is the chosen hyperparameter to be tuned. Probability is turned on to be True because later on probability will be used to calculate log loss. Kernel is set to 'linear' instead of the default 'rbf' because after constructing model for both, log loss is generally lower for 'linear' than for 'rbf' with all other parameters being constant. Maximum iteration is changed to 2000 instead of the default -1 where no limit is set for maximum iteration because as C value increases, model becomes harder and harder to converge. In addition, the log loss calculated with maximum iteration set to 2000 are still reasonable and sufficient for analysis. Apart from the parameters discussed above, all the other parameters have default values.

Cross-Validation

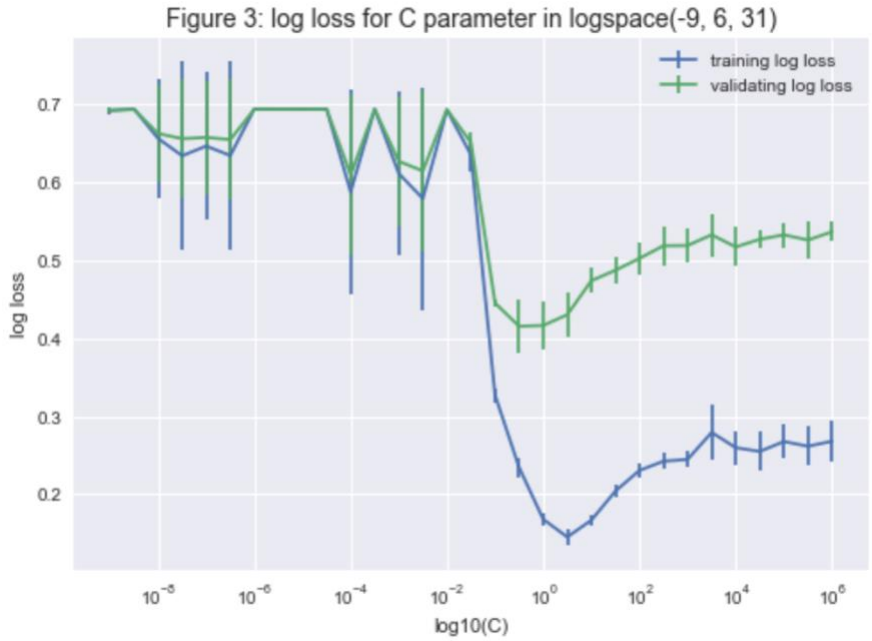
The same 5-fold cross-validation process is adopted as described in section 3.1 for logistic regression with the change SVM model is explored instead of logistic regression.

For details of the cross-validation process and data recorded, please view sub-section *Cross Validation* under *3.1 Logistic Regression*.

C-Parameters

C is a float representing the inverse of regularization strength and smaller value means greater regularization strength[6]. In order to find the best C value, the original x_train is fitted to 31 logistic regression models each with a different C-parameter regularly spaced between log-space of -9 and 6. The range is adopted because it has been suggested in previous homework assignment and projects. In addition, by observing the validating log loss in figure 3, a clear minimum occurs when C equals 0.316 since log loss generally decreases before reaching 0.316 and increases afterwards. The convexity of the graph suggest that this is a reasonable to be explored.

[6] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html



Discussion

As shown by the figure above, the standard deviation calculated for validating data log loss across different folds for all C values are all quite noticeable and is generally greater for smaller C values. In addition, for C values ranging between 0.01 and 100, validating log loss generally has larger standard deviation when compared with training data of the same C value, suggesting they are less stable in result under these C values explored.

For the range of C values explored, model performs the best when the C value equals 0.3162 which has a validating log loss of 0.4152. This is higher by a miniscule amount than the minimum validating log loss obtained by tuning C-parameter in logistic regression models which is 0.4142.

Within the chosen range of C-parameters, overfit is not an persisting problem as validating and training log loss both follow the same general trend. Scenarios where validating log loss increases and training log loss decreases occurs for two C values: 1 and 3.162. Therefore, overfit does occurs for C between 1 and 3.162 but does not persist for the entire latter range of C values explored.

4. Comparison

Among the three models explored, logistic regression has the lowest validating log loss of 0.4142 when its C parameter is tuned to 10, while the lowest validating log loss for MLP and SVM are 0.7586 and 0.4152 respectively. Although SVM and logistic regression render similar minimum validating log loss, in the following discussion, logistic regression will be chosen as the best classifier out of the consideration that it has the lowest validating log loss that is calculated based on 5-fold cross validation which ensures that it does not occur because of anomaly as well as the fact that its minimum log loss has lower uncertainty than SVM's, suggesting a more stable result.

Reasons that logistic regression perform better than SVM and MLP in this part of the project may include first, there are less outliers present as logistic regression's performance can easily be adversely affected by the presence of outliers when compared with SVM[7]. In addition, by using bag of words for feature generation, the data may be relatively more linearly separable, making it more suitable for logistic regression. Last but not least, the fact that logistic regression performs the best amongst the three classifiers explored may also be the result that more suitable parameters in other classifiers have not yet been found.

For the best logistic regression model where C is set to the optimum C value found of 10, 448 reviews are wrongly classified within 2400 validating data under 5-fold cross validation, rendering an error rate of 18.67%. Within the wrongly classified reviews, false negative (226) is slightly more than false positive (222) suggesting that more error is prone to occur when classifying reviews with a ground-truth of 1. When comparing across the three different sources, relatively larger amount wrongly classified reviews come from IMDB (168) than Yelp (144) than amazon (136). The result implies that classifying movie reviews are harder than classifying food and purchase reviews. This may be due to the fact that movie reviews are more likely to be longer and contains more sophisticated wordings and expression that makes it harder for models to discern the sentiment involved.

By looking into the reviews that logistic regression wrongly classifies, one noticeable characteristic for these reviews is the use of complex sentence structure such as metaphors and double negative involved. For instance, "Tasted like dirt" may not be easily associated with negative sentiment as a metaphor is used to link the taste of food with dirt. In addition, sentences such as "A film not easily forgotten" which uses double negative are also prone to wrong classification. The example mentioned in the previous sentence may be classified as negative sentiment review as "forgotten" appears in the sentence. Thus, these kind of reviews pose difficulty for the models built due to the complexity of the sentence.

[7] https://www.geeksforgeeks.org/differentiate-between-support-vector-machine-and-logistic-regression/

5. Leaderboard Result and Further Discussion

Model Name	**Error Rate**	**AUROC**
Logistic Regression (C = 10, solver = 'liblinear')	0.175	0.89767

The error rate is expected. As discussed in section 4, the 5-fold cross-validation error rate for the best model obtained is 18.67% (total of 448 out of 2400 reviews are wrongly classified), counted by summing the number of incorrectly classified reviews for validation data in each fold. This is similar to the error rate obtained by the testing data (17.5%) using the same model parameter as shown in the leaderboard. The similarity in error rate between validating data and testing data suggests that the 5-fold cross validation adopted is a reliable way of cross-checking the accuracy of a model under chosen parameters.

In addition, with a consistent error rate around 18%, it suggests that the model has room for further improvement as around one of five reviews is still being wrongly classified and at this stage, the predicted label can serve as supporting evidence but not conclusive evidence due to the relatively large error existing.

Potential areas of improvement include adding the website name into the feature vector, experimenting other models such as decision tree and KNN, and applying grid search to explore a more exhaustive combination of hyperparameters.

Part Two: Classifying Review Sentiment with Word Embeddings

1. Background and Data Inspection

In this part of the project, we are given customer reviews collected from imdb, amazon and yelp with the goal of assigning correct sentiment to a given review where negative sentiment is being labeled as 0 and positive sentiment is being labeled as 1 using pre-trained embedding vector generated by GloVe. x_train and x_test are provided with 2400 and 600 reviews respectively. For y data, y_train is provided while y_test is not. However, $predicted\ y\_test$ can be compared with y_test through leaderboard on Gradescope which will calculate the error rate as well as AUROC value.

The training data contains $1200\ (50\%)$ positive reviews and $1200\ (50\%)$ negative reviews. To better understand the classification of the data, 3 positive and 3 negative reviews are randomly chosen from x_train:

Positive (1):

- yelp, Just as good as when I had it more than a year ago!
- imdb, "I recommend this for EVERYONE who loves film, movies, anything...A Work of Art! "
- amazon, Keep up the good work Amazon!!

Negative (0):

- yelp, As a sushi lover avoid this place by all means.
- imdb, I'm still trying to get over how bad it was.
- amazon, Not enough volume.

2. Feature Generation Pipeline

The goal of this section is to first present the feature generation pipeline adopted and then justify some of the decisions made.

2.1 Pipeline

After loading the csv file of x data using numpy, the “text” column would be extracted and converted to list while the website name column would be left out for further processes.

Then, each review will undergo the following pro-processing processes:

1. Substitute any character that is not “A-Z”, “a-z” or apostrophe with space
2. Convert all letters into lower case
3. Split the string into list of words base on whitespace
4. Loop through all words and if a word ends in “n’t”, append “not” to the end of the list
5. Remove word that is in standard English “stopword” and not "not" or "no"

For example, the output after each step described in the previous page for *"I recommend this for EVERYONE who loves film, movies, anything... Don't miss it"* would be:

1. I recommend this for EVERYONE who loves film movies anything Don't miss it
2. i recommend this for everyone who loves film movies anything don't miss it
3. ['i', 'recommend', 'this', 'for', 'everyone', 'who', 'loves', 'film', 'movies', 'anything', "don't", miss, 'it']
4. ['i', 'recommend', 'this', 'for', 'everyone', 'who', 'loves', 'film', 'movies', 'anything', "don't", miss', 'it', 'not']
5. ['recommend', 'everyone', 'loves', 'film', 'movies', 'anything', 'miss', 'not']

Afterwards, for each review containing a list of words after processing, a feature vector will be calculated using the pre-trained embedding vector generated by GloVe. The details of feature vector generated for each review include:

1. Initialize an empty array and count to 0
2. Loop through all words in a review after preprocessing, if the word exists in the pre-trained embedding vector, add the embedding vector's value into the initialized array and increment count by one
3. After looping through all words in one line of review, divide the feature array's value by count number to obtain the average

When feature vector has been generated for all reviews, normalize each column using $MaxMinScalar$ to ensure that the value lies within 0 and 1 inclusive. For the given x_train in this project, the final feature vector would be of size 2400×50.

2.2 Justification

- *Pre-processing Step 1*
  Apostrophe is kept because if it is replaced by whitespace then words such as "can't" would become "can t". After splitting it would be interpreted as ["can", "t"] which can no longer be distinguished from "can" and loses its negative connotation. On the other hand, dash is replaced because in reviews examined, dash not only serves to connect words but are also used by people to demonstrate the beginning of an explanation. Under this circumstance, dash itself does not carry any positive or negative meaning. In addition, for words such as "well-design", with dash being replaced, "well" still carries the positive sentiment. Therefore, since reviews' meanings are not lost with dash being replaced, dashes are not preserved during substitution. In addition, numbers are replaced because a single number without context does not help to determine the sentiment of a review.

- *Pre-processing Step 4*
  "not" is appended to the list for a given review if a word ending in "n't" appears because for embedding vector generated by GloVe, it doesn't contain words such as "shouldn't", "don't", etc. Therefore, "not" is used in order to preserve the negated connotation of these word.

- *Pre-processing Step 5*
  Adaption has been made to the standard English "stopword" imported from nltk because it also includes words such as "not" and "no". If the original stopword is adopted, then after removing stopwords, "I do like it" and "I do not like it" would produce the exact same list of words and the negative sentiment from the latter sentence is lost. By whitelisting words including "not" and "no", important negative sentiments are preserved while words that do not contribute to the analysis of sentiment such as "the" and "I" are effectively removed.

- *Pre-processing without Stemmer*
  Stemmer is not applied because the embedding vector generated by GloVe does contain words of different forms from the same stem. For instance, it has "love", "loves" and "loved" even though they all share the same stem. In addition, even though PorterStemmer can effectively discern some stems, there are also cases where the stem it deduces is incorrect, for instance, "movies" would become "movi" and this would result not being able to find matches in the embedding vector dictionary.

- *Averaging feature vector and Normalization*
  An average value is taken because reviews vary in length and if a summation is taken instead of average, then longer reviews will be more likely to have higher values without a clear indication of positive or negative sediment. Normalization is adopted to ensure that values in each column are within the range of 0 to 1 inclusive, leading to faster convergence in models such as neural network.

- *Number of copies of embedding vector proportional to its occurrences*
  If a word appears twice in a given review, then its embedding vector would be added twice to result vector. This is done out of the consideration that certain sediment may be accentuated through the repetitive use of words. For instance, if a review contains phrases such as "Do not buy it! DO NOT", then the two occurrences of "not" emphasizes the negative sediment and by counting them proportional to the time they occur, the review is more likely to be classified as a negative review.

3. Models

In this section, three models will be explored, including logistic regression, neural network (MLP) and SVM. For each model, one of its hyperparameters will be tuned and results will be analyzed.

3.1 Logistic Regression

Parameters

For logistic regression, C-parameter is the chosen hyperparameter to be tuned. "liblinear" solver is adopted instead of the default "lbfgs" because scikit-learn documentation suggests that "liblinear" is a good choice when the dataset is small[8]. Since x_train for this part of the project only contains 2400 rows of data, its size is suitable for "liblinear". All the other parameters have default values and max iteration is not increased because no convergence warning is raised by using the default value to construct the model.

Cross-Validation

For this project, 5-fold cross validation is adopted because 80% to 20% train-validate size is one of the most commonly used splitting ratio and has also been suggested in previous homework.

Before exploring any C value, a $KFold$ object (imported from $sklearn.model\_selection$) is declared with the number of fold equals 5 and shuffling turned on. The shuffle is turned on because when inspecting the input, positive and negative sentiment comes in clusters where a few hundred of positive reviews followed by a few hundred of negative reviews. By shuffling, this reduces the possibility of positive-negative training data imbalance after splitting.

For each C value, it would be fitted to the model 5 times (5-fold cross-validation) with the indices of testing and training data in one particular fold determined by the $split$ method[9] of the $KFold$ object declared. In each model, the model is fitted to the 80% training data and validated using the rest 20% and its training and validating log loss in each fold would be recorded. After completing the 5-fold cross-validation for a given C-value, training log loss average, validating log loss average, training log loss standard deviation and validating log loss standard deviation would be computed and recorded.

C-Parameters

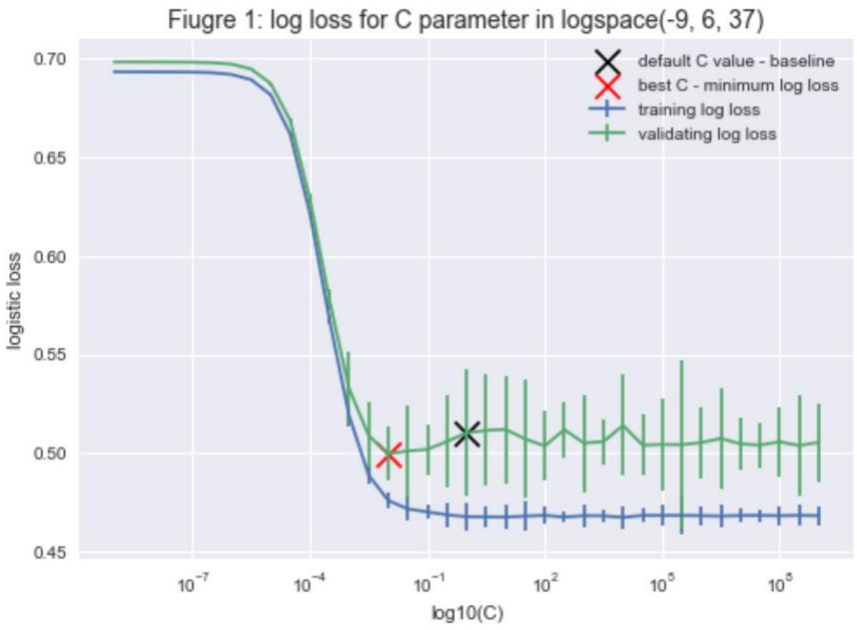
C is a float representing the inverse of regularization strength and smaller value means greater regularization strength[10]. In order to find the best C value, the original x_train is fitted to 37 logistic regression models each with a different C-parameter regularly spaced between log-space of -9 and 9. The end range of -9 is adopted because it has been suggested in previous homework assignment and projects. In addition, the log loss for both training and validating stays constant for smaller values of C implying that the lower bound of the range explored do not need to be further extended. The upper bound of 9 is chosen because no obvious minimum validating log loss can be identified when it is set to 6. Thus, a slightly higher than usual range is adopted.

---

[8] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[9] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html

[10] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html



Discussion

As shown by the figure above, uncertainty for validating and training data across different folds is negligible when C value is small and becomes greater for C values corresponding to validating log loss around 0.52. In addition, validating log loss has larger uncertainty in general when compared with training data, suggesting they are less stable in result.

For parameters examined, model performs the best when the C value is set to 0.01, as shown by the minimum point for the validating log loss plotted. However, although it has the lowest validating log loss among all C parameters explored, its uncertainty together with the slightly higher log loss for C values greater 0.01 suggest that it does not outperform other models as much and models with C values greater than and equal to 0 all have the chance of obtaining the lowest log loss when uncertainty is taken into consideration.

Within the chosen range of C-parameters, overfit is not an obvious problem. Even though training log loss remains low for C values greater than 0.01 and validating log loss is slightly higher for C values greater than 0.01, validating log loss varies within 0.01 for C values between 0.01 and 10^9. In addition, if uncertainty is taken into account, the slight increase in validating log loss becomes even less obvious. Therefore, no conclusive evidence is available to suggest overfit for the range of C values explored.

3.2 Neural Network (MLP)

Parameters

For MLP, the number of neurons in the hidden layer is the chosen hyperparameter to be tuned. The number of hidden layer is chosen to be one because the book "Introduction to Neural Networks for Java" by Jeff Heaton suggests that one hidden layer is sufficient to approximate any function that contains a continuous mapping from one finite space to another.[11] In addition, maximum iteration is increased to 400 because for some models tested, the default 200 default is not enough to converge. By increasing the maximum iteration, less non-convergence scenarios can be avoided. All other parameters have default values and solver is not changed to "Ibfgs" which according to the scikit-learn documentation performs well on smaller dataset[12] because after experimenting with both the default "adam" and "Ibfgs", "adam" renders much lower log loss.

Cross-Validation

A similar 5-fold cross-validation process is adopted as described in section 3.1 for logistic regression with the change that instead of exploring different C values, different number of neurons for the single hidden layer is being explored and MLP models are explored instead of logistic regression models.

For details of the cross-validation process and data recorded, please view sub-section *Cross Validation* under *3.1 Logistic Regression*.

Hidden Layer Neuron Number

13 different neuron numbers are examined, ranging between 2 and 26 with a step size of 2. This is chosen because some articles suggest that the number of hidden neurons should be between the size of the input layer, in this case 50, and the size of the output layer, in this case 2. The upper bound of the exploration stops with 26 because as shown in the figure below, validating log loss increases with greater number of neurons and it is logical to assume that if the trend follows, then for neuron number greater than 26 log loss will continue to increase which does not contribute to finding the best model. Therefore, because the lower bound is already as low as two neurons and the further extending the upper bound does not contribute to the discussion, the range explored is suitable at this stage.

[11] https://stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-neural-netw

[12] https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

Discussion

As shown by the figure above, the standard deviation calculated for validating data log loss across different folds for all neuron numbers explored are all quite noticeable with the general trend that standard deviation is larger for higher neuron numbers. In addition, validating log loss has larger standard deviation when compared with training data of the same neuron number, suggesting they are less stable in result.

For the number of neurons examined, model performs the best when the neuron number equals 2 which has a validating log loss of 0.5167. This is slightly higher than the minimum validating log loss obtained by tuning C-parameter in logistic regression models which is 0.5027.

Overfit is shown within the examined range of neuron numbers. As shown in figure 2, as number of neurons increases, log loss for training data continue to decrease while that for validating data begins to increase, suggesting overfit in models. This may be partially due to the fact that the number of training data size is small, and the number of feature vector is also relatively small. On account of the factors, models would incline towards overfit with larger number of neurons.

3.3 Supporting Vector Machine (SVM)

Parameters

For MLP, the number of neurons in the hidden layer is the chosen hyperparameter to be tuned. The number of hidden layer is chosen to be one because the book "Introduction to Neural Networks for Java" by Jeff Heaton suggests that one hidden layer is sufficient to approximate any function that contains a continuous mapping from one finite space to another.[13] In addition, maximum iteration is increased to 400 because for some models tested, the default 200 default is not enough for models to converge. All the other parameters have default values and solver is not changed to "Ibfgs" which according to the scikit-learn documentation performs well on smaller dataset[14] because after experimenting with both "adam", the default solver, and "Ibfgs", "adam" is able to render much lower log loss.

Cross-Validation

The same 5-fold cross-validation process is adopted as described in section 3.1 for logistic regression with the change SVM model is explored instead of logistic regression.

For details of the cross-validation process and data recorded, please view sub-section *Cross Validation* under *3.1 Logistic Regression*.

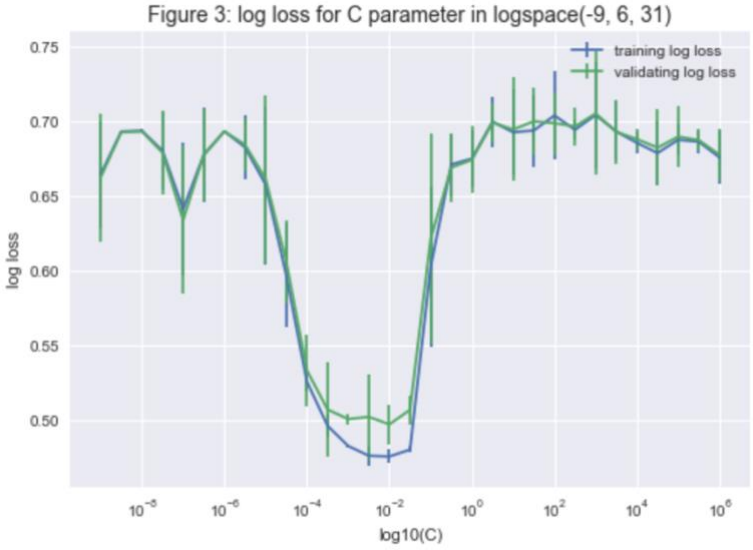
C-Parameters

C is a float representing the inverse of regularization strength and smaller value means greater regularization strength[15]. In order to find the best C value, the original x_train is fitted to 31 logistic regression models each with a different C-parameter regularly spaced between log-space of -9 and 6. The range is adopted because it has been suggested in previous homework assignment and projects. In addition, by observing the log loss in figure 3, a clear minimum occurs when C equals 0.01 since log loss decreases before reaching 0.01 and increases afterwards. The convexity of the graph suggest that this is a reasonable to be explored.

[13] https://stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-neural-netw

[14] https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

[15] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html



Discussion

As shown by the figure above, the standard deviation calculated for validating data log loss across different folds for all C values are all quite noticeable and no obvious change trend in standard deviation is discernable as C values increase or decrease. In addition, same as logistic regression models, validating log loss generally has larger standard deviation when compared with training data of the same C value, suggesting they are less stable in result.

For the range of C values explored, model performs the best when the C value equals 0.01 which has a validating log loss of 0.4975. This is slightly lower than the minimum validating log loss obtained by tuning C-parameter in logistic regression models which is 0.5027.

Within the chosen range of C-parameters, overfit is not an obvious problem as validating and training log loss both follow the same general trend as shown by the figure above and scenarios where validating log loss increases and training log loss decreases are not shown. Therefore, no supporting evidence is available to suggest overfit for the range of C values explored.

4. Comparison

Among the three models explored, SVM has the lowest validating log loss of 0.4975 when its C parameter is tuned to 0.01, while the lowest validating for logistic regression and MLP are 0.5027 and 0.5167 respectively. Although SVM and logistic regression render similar minimum validating log loss, in the following discussion, SVM will be chosen as the best classifier out of the consideration that it has the lowest validating log loss that is calculated based on 5-fold cross validation which increases credibility to the data obtained and avoid overfit for the range of C values explored.

Reasons that SVM perform better than logistic regression and MLP in this part of the project because may include first, it is less prone to outliers as its objective is to best margin that separates the data instead of the conditional likelihood of the training data[16]. In addition, as shown by Figure 3 in section 3.3, SVM is less likely to overfit where training log loss decreases and validating log loss increases. Therefore, with characteristics of being less prone to be outliers and overfit, SVM gives the best performance among the 3 classifiers on the labeled data.

For the best SVM model where C is set to the optimum C value found of 0.01, 529 reviews are wrongly classified within 2400 validating data using 5-fold cross validation, rendering an error rate of 22%. Within the wrongly classified reviews, false negative (268) is slightly more than false positive (261) suggesting that more error is prone to occur when classifying reviews with a ground-truth of 1. When comparing across the three different sources, slightly more wrongly classified reviews come from IMDB (181) than Yelp (178) than amazon (170). The result implies that classifying movie reviews are harder than classifying food and purchase reviews. This may be due to the fact that movie reviews are more likely to be longer and contains more sophisticated wordings and expression that makes it harder to models to discern the sediment involved.

By looking into the reviews that SVM wrongly classifies, one noticeable characteristic for these reviews is the use of complex sentence structure such as double negatives. For instance, “Kieslowski never ceases to amaze me” may be classified as negative sediment review as “never” appears in the sentence. However, “never ceases” act as a double negative that provides positive sediment for the sentence. Thus, this kind of reviews may pose difficulty for the models built due to the complexity of the sentence.

[16] https://www.geeksforgeeks.org/differentiate-between-support-vector-machine-and-logistic-regression/

5. Leaderboard Result and Further Discussion

Model Name	**Error Rate**	**AUROC**
SVM (C = 0.01, solver = 'linear')	0.21833	0.82157

The error rate is expected. As discussed in section 4, the 5-fold cross-validation error rate for the best model obtained is 22% (total of 529 out of 2400 reviews are wrongly classified), counted by summing the number of incorrectly classified reviews for validation data in each fold. This is similar to the error rate obtained by the testing data (21.833%) using the same model parameter as shown in the leaderboard. The similarity in error rate between validating data and testing data suggests that the 5-fold cross validation adopted is a reliable way of cross-checking the accuracy of a model under chosen parameters.

In addition, with a consistent error rate around 20%, it suggests that the model has room of improvement as one of five reviews are still being wrongly classified and the predicted label can serve as supporting evidence but not conclusive evidence due to the relatively large error existing.

Potential areas of improvement include adding the website name into the feature vector, experimenting other models such as decision tree and KNN, and applying grid search to explore a more exhaustive combination of hyperparameters.