CS57800 Statistical Machine Learning
Homework 3

Penghao Wang

wang1128@purdue.edu

November 10, 2015

1 Foundations

1.1

VC dimension of C is $2n+1$. The VC dimension of C is $2n+1$, because it is the largest number of subset of instances that can be shattered. If we put all the $2n+1$ points on a circle, and label n of them as negative and n+1 of them as positive.(We could change the labels, but there is no difference). We could shatter the points(instance) by using the polygons with n vertices in the plane. (Just connect the negative points, it will be a polygons with n vertices.) It is clear that for all sets of size $2n+2$ points, there is a labeling of these points that cannot be captured by polygons with polygons with n vertices. If we put a point with different label inside of the polygons, the polygons won't shatter the 2n+2 points. On the other hand, if we put it outside of the polygons, it is the same. We can change the labels to opposite, it result will be same, the origin ploygons can not shatter the $2n+2$ points. (Indeed, it need polygons with n+1 vertices) As a conclusion, VC dimension of C is $2n+1$.

1.2

$f(\hat{y}) = \frac{log(1+e^{-y\hat{y}})}{log2}$

$f(\hat{y})' = -\frac{y}{log(2)(e^{y\hat{y}}+1)}$

$f(\hat{y})'' = \frac{y^2e^{y\hat{y}}}{log(2)(e^{y\hat{y}}+1)^2} = \frac{y^2e^{y\hat{y}}}{log(2)}\frac{1}{(e^{y\hat{y}}+1)^2}$

Since we know, $y^2 > 0$ for $y \in \pm1$ and $e^{y\hat{y}} > 0$, $\frac{y^2e^{y\hat{y}}}{log(2)} > 0$. We know $(e^{y\hat{y}}+1)^2 > 0$, so $\frac{1}{(e^{y\hat{y}}+1)^2} > 0$ In that case, $f(\hat{y})'' > 0$. This function is convex for a fixed value of $y \in \pm1$

1.3

When there is one example occur, the training error $=\frac{1}{n}$. In that case, we need to guarantee that the training error is $< \frac{1}{n}$. Because training error of the t weak hypothesis is at most γ where $0 < \gamma < 0.5$, in the equation, $\gamma = \frac{1}{2} - \gamma$ We know that :

$Traningerror(H_{final}) \leq e^{(-2T(\frac{1}{2}-\gamma)^2)}$. The upper bound of that is $< \frac{1}{n}$
$e^{(-2T(\frac{1}{2}-\gamma)^2)} < \frac{1}{n}$

$-2T(\frac{1}{2}-\gamma)^2 < ln(n^{-1})$

$2T(\frac{1}{2}-\gamma)^2 > ln(n)$

$T > \frac{ln(n)}{2(\frac{1}{2}-\gamma)^2}$

1.4

1. The weak hypothesis used at each round:

First Round: $x_1 > 5$ then y is positive, else y is negative. Error: $e_1 = 0.2$

Second Round $x_1 > 8$ then y is positive. Otherwise y will be negative. Error: $e_2 = 0.25$
2. The distribution D over the data points for each round:

First round: $D_1 =$ [0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1]

Second round $D_2 =$ [0.0625,0.0625,0.0625,0.0625,0.0625,0.0625,0.0625,0.0625,0.25,0.25]

3. The final hypothesis:

$H_{final} = sign(0.6931 sign(x_1 - 5) + 0.5493 sign(x_2 - 8))$

1.5

$K_1(\overrightarrow{x}, \overrightarrow{y})$ is a valid kernel. We could build a new kernel, $\alpha K_1(\overrightarrow{x}, \overrightarrow{y}), \alpha > 0$

$K_2(\overrightarrow{x}, \overrightarrow{y})$ is a valid kernel. We could build a new kernel, $\beta K_2(\overrightarrow{x}, \overrightarrow{y}), \beta > 0$

We could build a new kernel by adding two valid kernel. In that case, the new kernel is $K(\overrightarrow{x}, \overrightarrow{y}) = \alpha K_1(\overrightarrow{x}, \overrightarrow{y}) + \beta K_2(\overrightarrow{x}, \overrightarrow{y})$
Then $K(\overrightarrow{x}, \overrightarrow{y})$ is also a kernel function

1.6

From the definition, we know $y_i(w \cdot x_i + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$. if $0 \leq \xi_i < 1$ The prediction is right. In that case, the trainerror of that instance is equal to zero $\leq \xi_i$. If $\xi_i \geq 1$, Some mistake may happen, $y_i(w \cdot x_i + b) < 0$ In that case, the $trainerror_i$ is $1 < \xi_i$ As a conclusion, in the both case, if we combine two case together, $\sum \xi_i \geq$ total training error. In that case $\sum \xi_i$ is the upper bound of the training error.

2 Programming Report

2.1 Introduction

In the homework, GD algorithm are implemented. The observation of their performances in practice by running the algorithms over the movie review datasets.

2.2 Feature sets

In the algorithm, feature set is a vector that contains 0 and 1. In the beginning, the list of words are created. Then, an array that contains zeros are created. Then If the word in the first row of the review is in the word list, then find the index of the position of the word and change value of that position from 0 to 1. For example, the first word in the first review is "message". The length of the list of the word is 5000. And the message appears in the first position, the the array will be (1,0,0,0,0...,0)

2.3 Functions in the project

1. The numpy and csv function are imported. Numpy is used to calculate the GD algorithm.

2. $cal\_reviewlistlabel()$ function is used to read the csv file. The parameter is filename. It return $review_list$, $review_label$. For the label part, if the label is '+', the review label is 1. Otherwise, the review label will be -1.

3. $calListuniWord(review\_list)$ function is used to generate the list of uni-gram words and save them to the $list\_word$. When read the unigram word to the list, the frequency of the words are calculated. I delete the words that appears less then five time or more than 300 times. I do this because the most frequent words are the words like : the, a , film, an... These words are meaningless. Also, the words that appear in low frequency are deleted because they won't affect the weight too much.

4. $calListbiWord(review\_list)$: function is used to generate the list of bi-gram words. It is pretty like $calListuniWord(review\_list)$. It is also remove the most common words and the words that appear only few times.

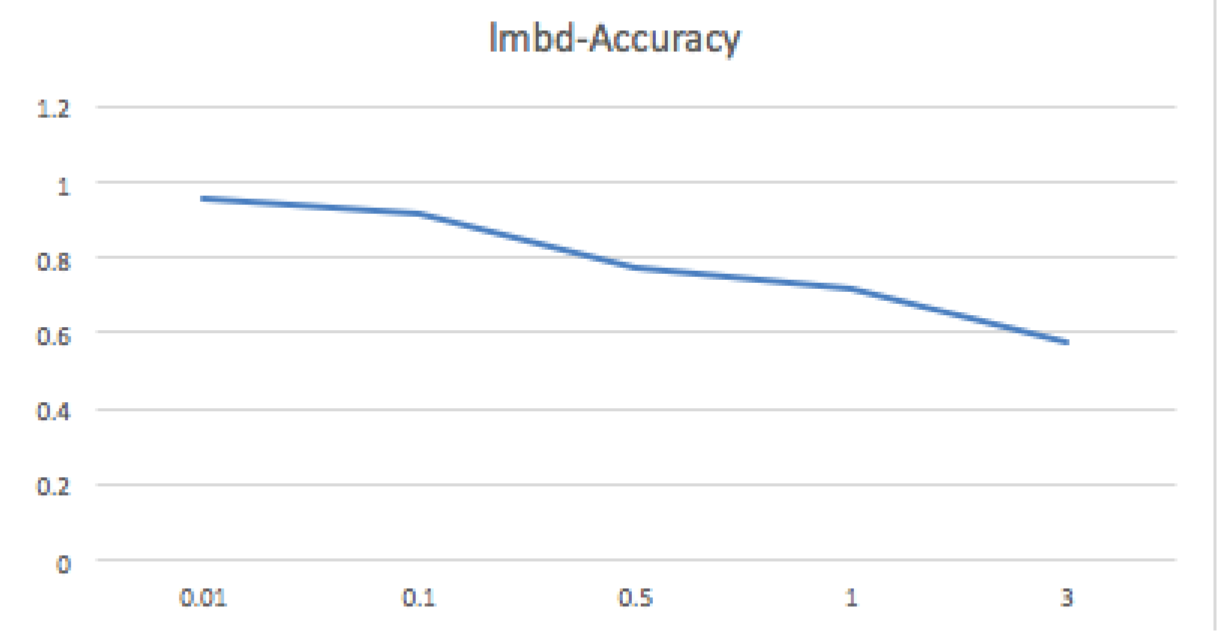
5. $cal\_feature\_array(review\_list, list\_word)$ $cal\_bifeature\_array$ and $cal\_both\_feature\_array$: is calculate the unigram,bigram and both feature set.

6. GD() function is calculate w by using GD algorithm and feature sets. The arguments are maxIterations,review_list,review_label,list_word,regularization,stepSize,lmbd,and feature Set. The regularization has two value, 1 means l1 and 2 means l2
7. *calPrecsion_p*() is to calculate precision.
8. *calRecall_p*() is to calculate recall.
9. *calFscore*() is to calculate F-score
10. *calTrainError_p*() is to calculate accuracy
11. calaprf() is to print out the precision, recall , Fscore and accuracy

2.4 Performance

2.4.1 Selection of Step size and lmbd

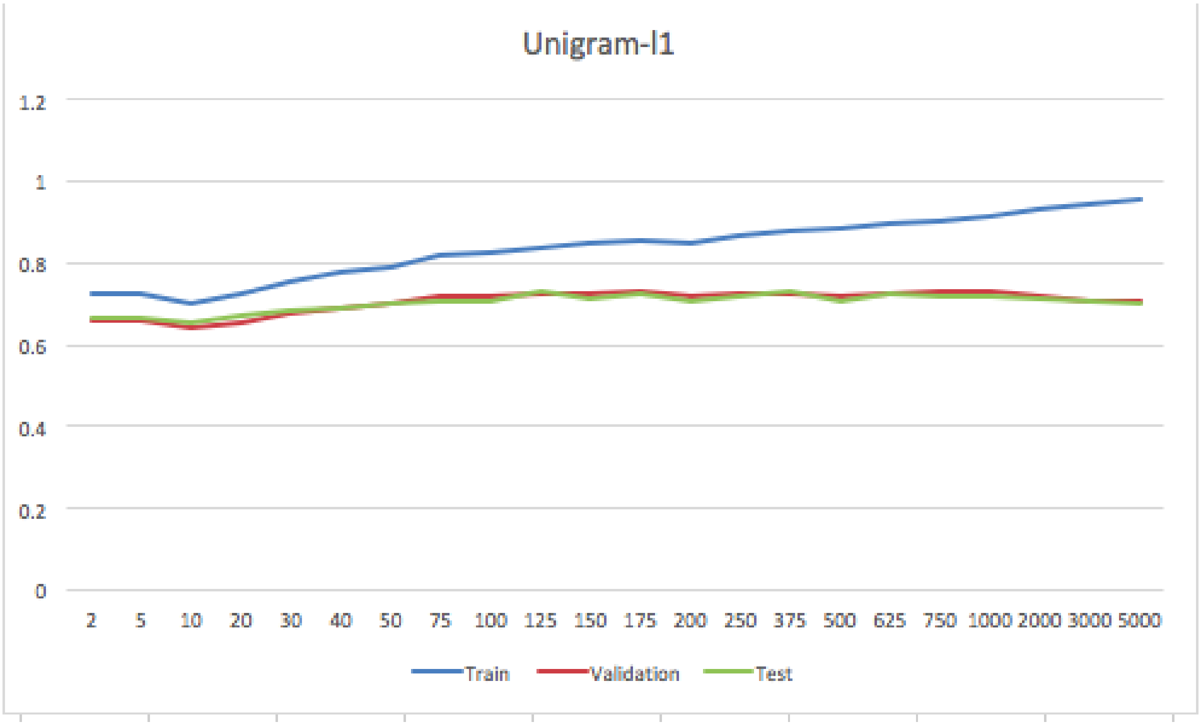
There are max Iteration, step size and lmbd in the GD function. The way to choose these variables are setting other fixed and change the other one that note the performance when the variable changes. Through the testing stage, I found that the step size effects the accuracy very little if it is small enough. If I choose a small number like 0.005, and 0.003. The main differences between them are speed to converge. In that case, step size affects the accuracy a little if it is small enough. To determine the lmbd, I test several values: 0.01, 0.1, 0.5, 1, and 3. Also, I use the max iteration as 1000. The reason why I do this is that I want to see how the performance in a relative small iteration. Then I will decide what the lmbd should be choose. The following figure represent the accuracy of the training by using different value.



As a conclusion, I choose the lmbd as 0.01 based on the performance. The step size is 0.005. If the max iteration is large enough, the value lmbd may not affect a lot, it will finally converge. However, when consider my calculation performance of my laptop, I choose a relative small number.

2.4.2 Unigram-l1

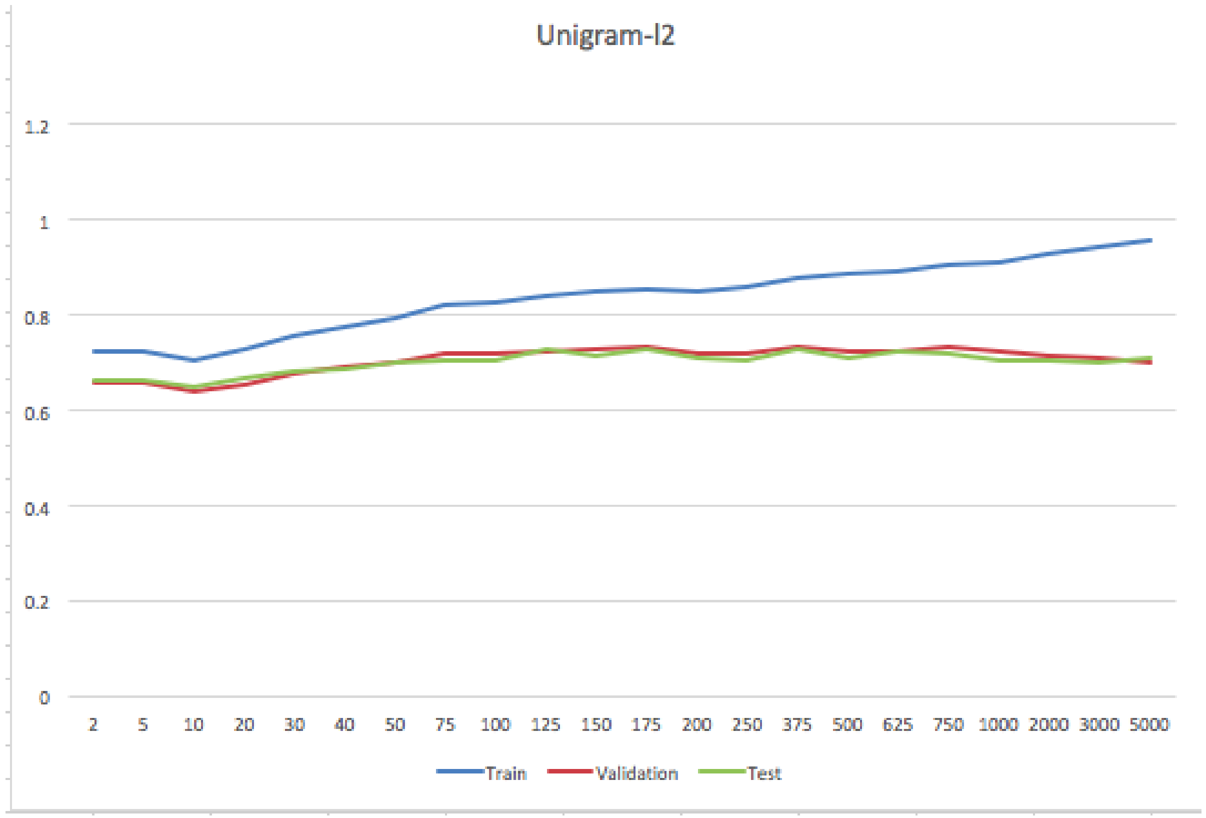
The following figure shows the accuracy by using l1. The word list is unigram words. We could know that, the accuracy of the train data is relative high when compare with the validation accuracy.



We could notice that, after 750 iteration, the validation and test accuracy change not that much. However, the train accuracy continue increase. It is overfitting finally. Through the validation performance, The maxIteration is 750. For the training, the accuracy is 0.901531728665. Precision is 0.925461741425. Recall is 0.874143302181. F-score is 0.899070810638. For the validation, the accuracy is 0.728424015009. Precision is 0.747219413549. Recall is 0.691947565543. F-score is 0.718522119592. For the test, the accuracy is 0.724202626642. Precision is 0.737487231869. Recall is 0.685660018993. F-score is 0.71062992126.

2.4.3 Unigram-l2

The following figure shows the accuracy by using l2. The word list is unigram words. We could know that, the accuracy of the train data is relative high when compare with the validation accuracy.

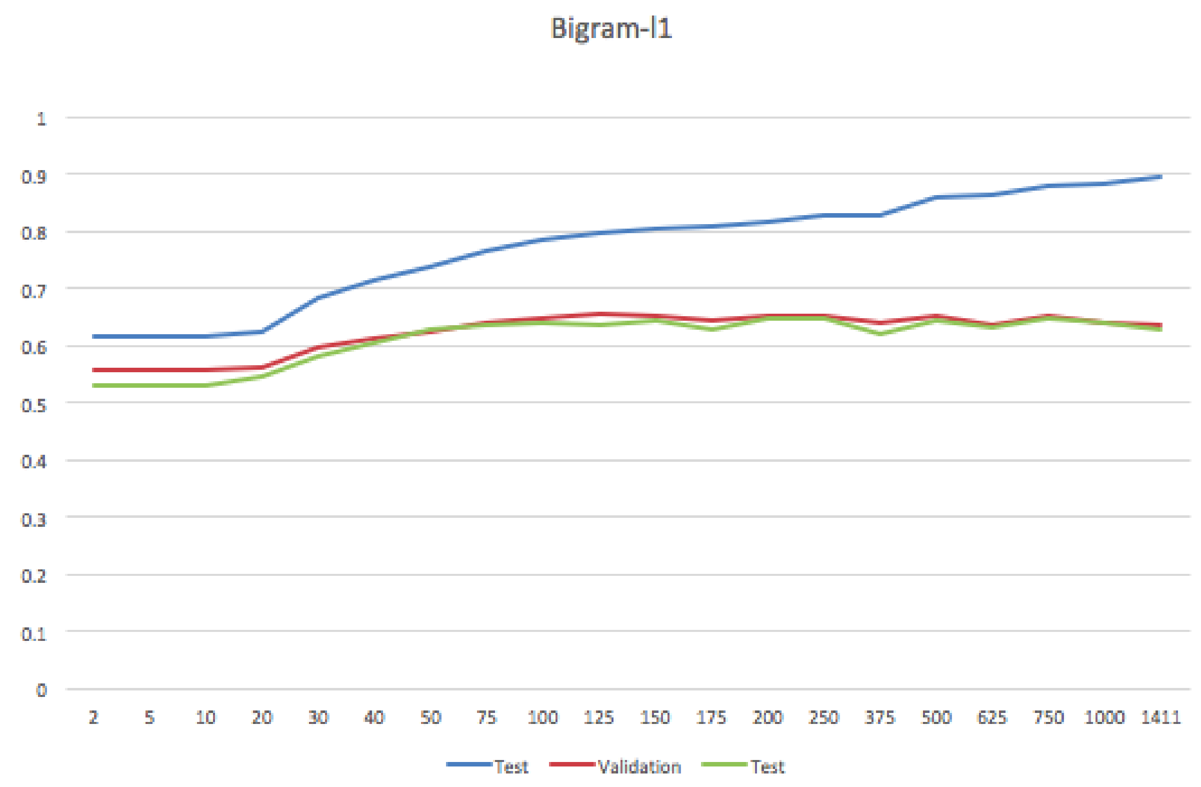


We could notice that, after 375 iteration, the validation and test accuracy change not that much. However, the train accuracy continue increase. It is overfitting finally. Through the validation performance, The maxIteration is 375. For the training, the accuracy is 0.879806189434. Precision is 0.872671755725. Recall is 0.890342679128. F-score is 0.881418658443. For the validation, the accuracy is 0.724202626642. Precision is 0.713143872114. Recall is 0.751872659176. F-score is 0.731996353692. For the test, the accuracy is 0.722795497186. Precision is 0.703345070423. Recall is 0.758784425451. F-score is 0.730013704888.

As a conclusion, if we compare based on the max iteration, l1 performance is a little bit better then l2.

2.4.4 bigram-l1

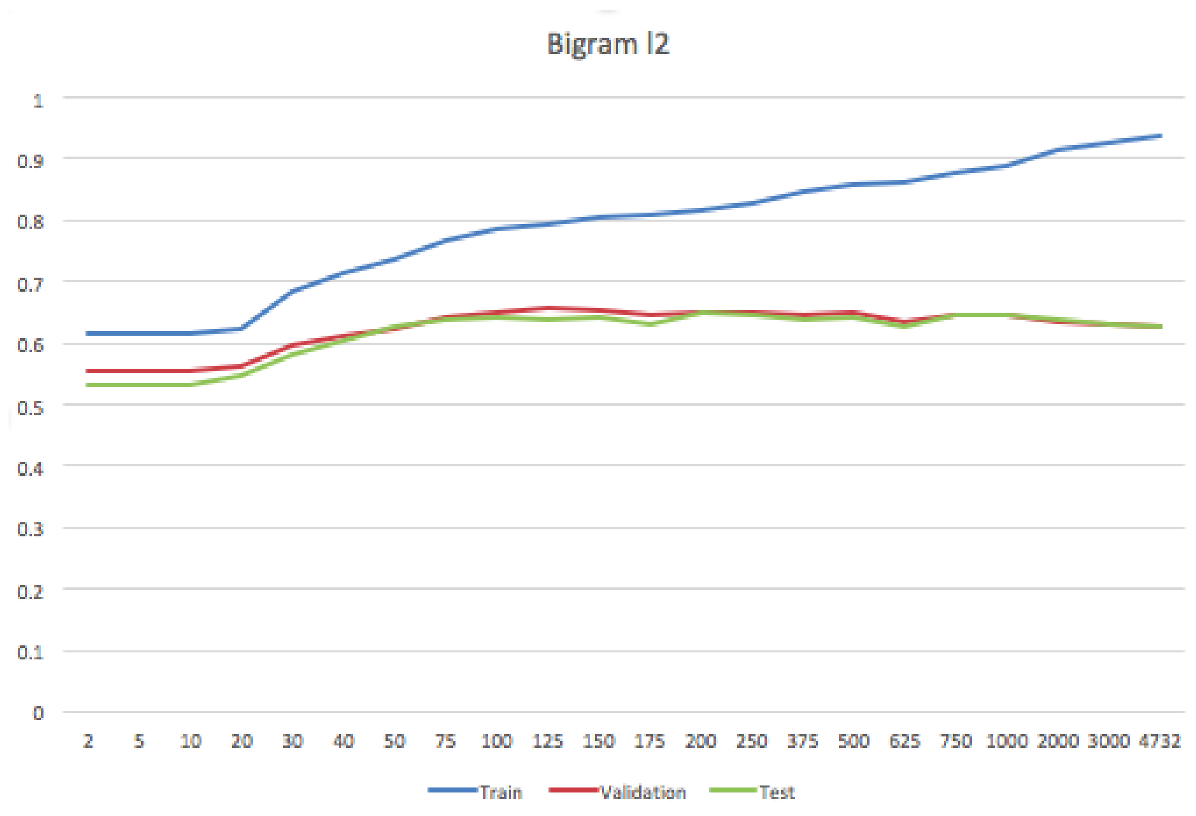
The following figure shows the accuracy by using l1. The word list is bigram words. We could know that, the accuracy of the train data is relative high when compare with the validation accuracy.



We could notice that, after 125 iteration, the validation and test accuracy change not that much. However, the train accuracy continue increase. It is overfitting finally. Through the validation performance, The maxIteration is 125. For training, the accuracy is 0.796342607065. Precision: 0.775657704539. Recall: 0.835825545171. F-score: 0.804618383566. For validation: the accuracy is 0.648686679174. Precision: 0.633027522936. Recall: 0.710674157303. F-score: 0.669607410675. For test: Accuracy: 0.640243902439. Precision: 0.618573797678. Recall: 0.708452041785. F-score: 0.660469234174.

2.4.5 bigram-l2

The following figure shows the accuracy by using l2. The word list is bigram words. We could know that, the accuracy of the train data is relative high when compare with the validation accuracy.

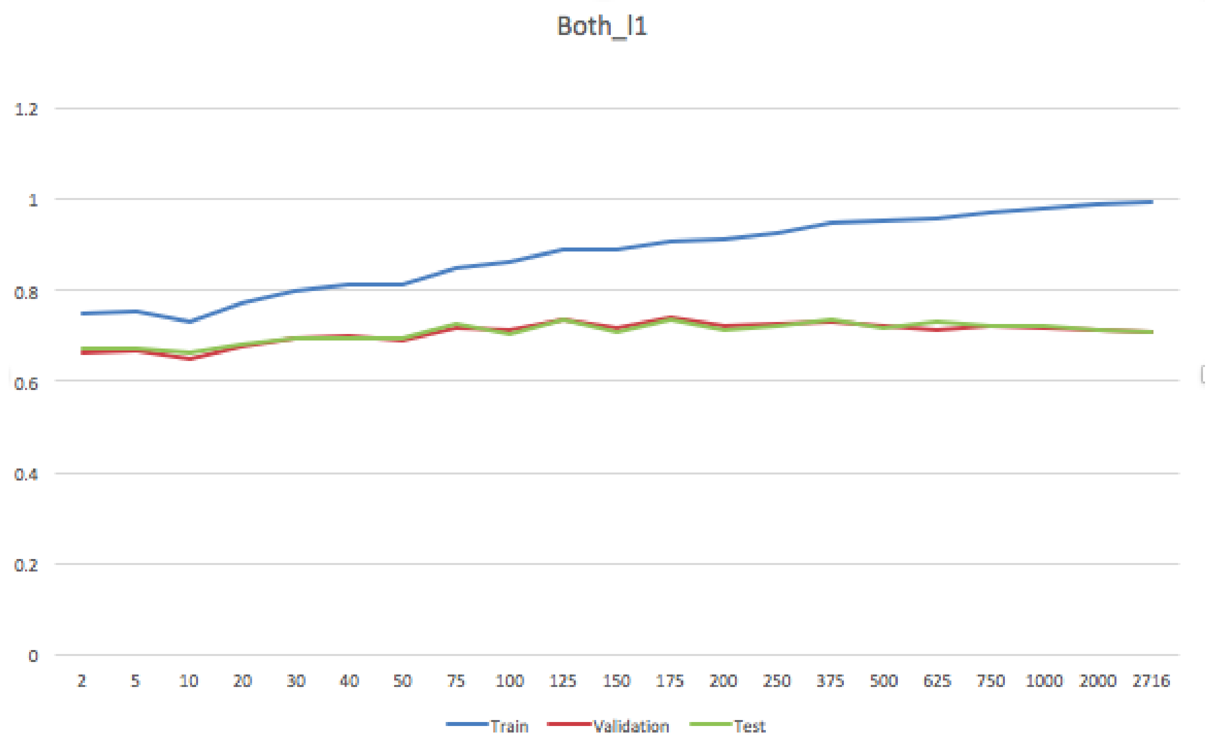


We could notice that, after 125 iteration, the validation and test accuracy change not that much. However, the train accuracy continue increase. It is overfitting finally. Through the validation performance, The maxIteration is 125. For train: Accuracy is 0.796655204751. Precision: 0.775946805435; Recall: 0.836137071651; F-score: 0.804918278602. For validation: Accuracy is 0.648686679174. Precision: 0.633027522936; Recall: 0.710674157303; F-score: 0.669607410675: For test, accuracy is 0.640243902439; Precision: 0.618573797678; Recall; 0.708452041785; F-score: 0.660469234174;

We could notice that, the steps for l1 to converge are less than the l2. L1 is a little bit better than l2.

2.4.6 Both-l1

The following figure shows the accuracy by using l1. The word list is both unigram and bigram words. We could know that, the accuracy of the train data is relative high when compare with the validation accuracy.

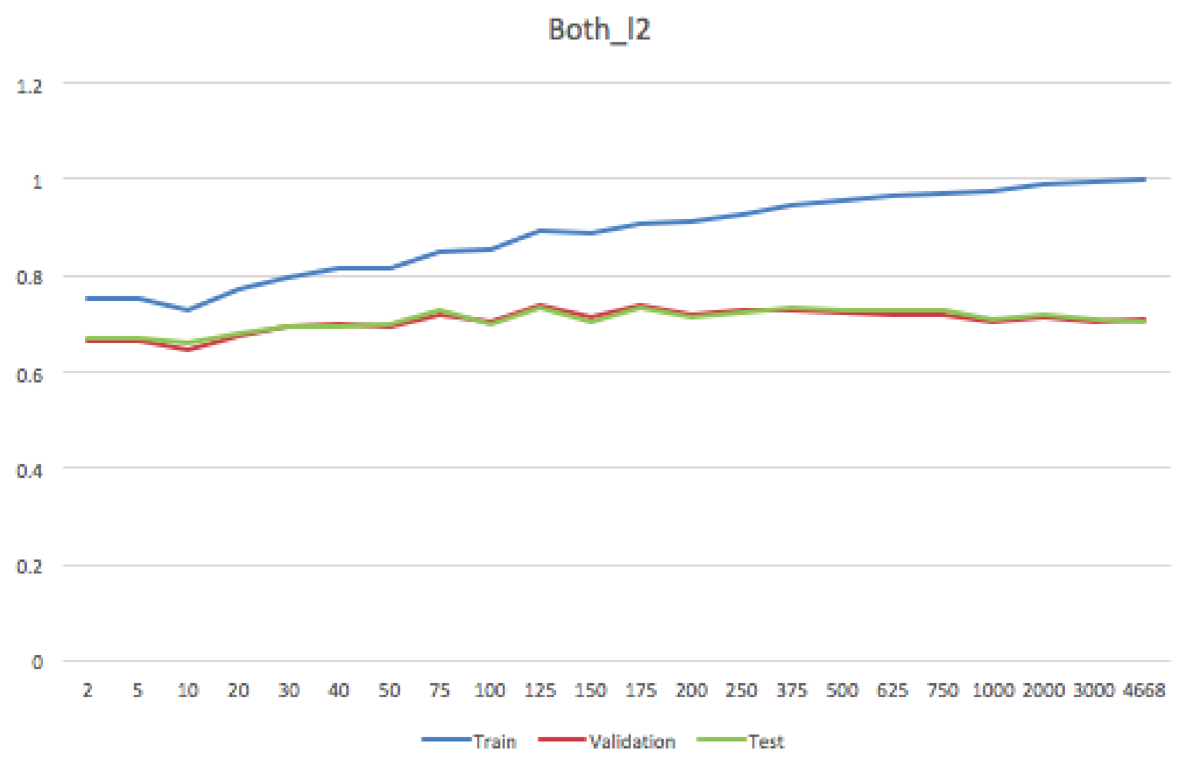


We could notice that, after 175 iteration, the validation and test accuracy change not that much. However, the train accuracy continue increase. It is overfitting finally. Through the validation performance, The maxIteration is 175.

For train: Accuracy: 0.907002188184; Precision: 0.892524767337; Recall: 0.926168224299; F-score: 0.909035315701; For validation: Accuracy is 0.729831144465. Precision: 0.713913043478: Recall: 0.76872659176; F-score: 0.740306582507. For test: Accuracy: 0.72138836773; Precision: 0.697334479794; Recall: 0.770180436847; F-score: 0.731949458484.

2.4.7 Both-l2

The following figure shows the accuracy by using l2. The word list is both unigram and bigram words. We could know that, the accuracy of the train data is relative high when compare with the validation accuracy.



We could notice that, after 175 iteration, the validation and test accuracy change not that much. However, the train accuracy continue increase. It is overfitting finally. Through the validation performance, The maxIteration is 175. For train: Accuracy is 0.907002188184. Precision: 0.893469756244; Recall: 0.92492211838; F-score: 0.90892392469; For validation: Accuracy: 0.729831144465; Precision: 0.714285714286; Recall: 0.767790262172; F-score: 0.740072202166; For test Accuracy: 0.720919324578; Precision: 0.697413793103; Recall: 0.768281101614; F-score: 0.731134206959; We could notice that, the steps for l1 to converge are less than the l2. L1 is a little bit better than l2.

2.4.8 Conclusion

The project use GD algorithm. The observation of their performances in practice by running the algorithms over the movie review datasets is shown above.