**Name: Leonardo Neves**

**Andrew ID: lneves**

**Nickname: soclose**

**Machine Learning for Text Mining**

**Homework 5**

1. **Statement of Assurance**

All the code and work for this assignment is original work done only by me.

1. **Data Preprocessing**
2. **[5 pts]** After your finishing the data preprocessing, report the top 9 frequent tokens and corresponding counts in the report.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Rank | Token | Count | Rank | Token | Count | Rank | Token | Count |
| No. 1 | good | 742812 | No. 2 | place | 716149 | No. 3 | food | 691505 |
| No. 4 | great | 585143 | No. 5 | like | 546150 | No. 6 | just | 521349 |
| No. 7 | time | 442453 | No. 8 | service | 431306 | No. 9 | really | 387324 |

1. **[5 pts]** Before continuing to the next step, another interesting problem is to check the star distribution of training samples. Report the count of training samples for each star (i.e., 1 to 5).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Star | 1 | 2 | 3 | 4 | 5 |
| # of training data | 128038 | 112547 | 178215 | 373469 | 463084 |
| Percentage | 10.2% | 9.0% | 14.2% | 29.7% | 36.9% |

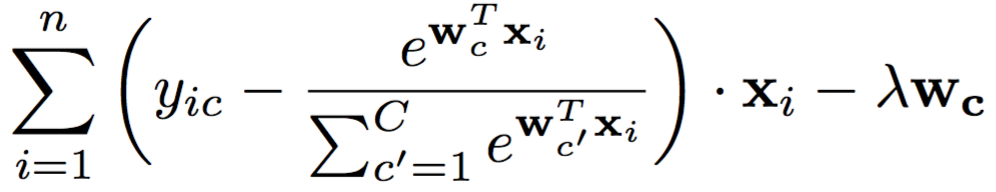
Do you find something unexpected from the distribution (e.g., whether the dataset is balanced)?

Yes, the dataset is unbalanced because it has much more good reviews (4,5) than bad reviews, (1,2).

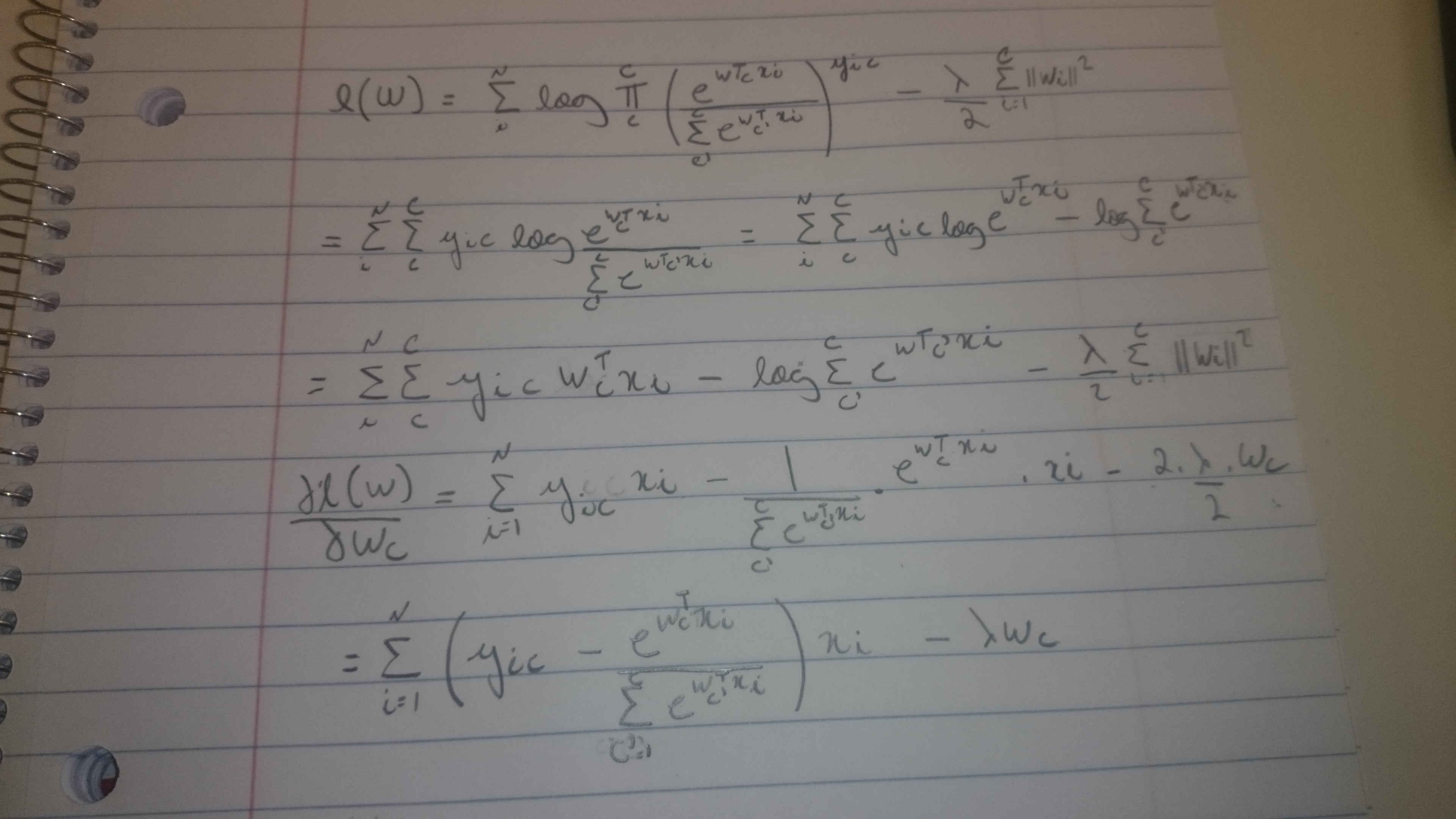
**[5 bonus pts]** Will this be a problem in training the model? If so, could you give some idea about how to address it and explain why your idea should work?

It could be a problem, but I don’t think it will be. If our dataset is representative of the real Yelp reviews (ie it has approximately the same distribution of ratings as all Yelp reviews) we should not change the imbalance. Otherwise, if it was a bad sample of the real world data, we might have to balance it. To do so, we could either undersampling, by deleting some reviews from the largest classes (this would be bad because we would be wasting data) or, a better approach, would be to give more weight to the smaller classes, balancing the contribution of each class to the overall prediction.

1. **Model Design**
2. **[5 pts]** Show that the gradient of regularized conditional log-likelihood function with respect to the weight vector of class (i.e., ) is equal to



Notice that the gradient of log-likelihood function with respect to a vector is itself a vector, whose -th element is defined as , where is the -th element of vector .



1. **[5 pts + 5 bonus pts]** Let the learning rate be , outline the algorithm you choose (SGD or Batched-SGD) for implementation. You should cover how would you like to update the weights in each iteration, how to check the convergence and stop the algorithm and so on.

I have implemented the batched-SGD. Since the SGD is a particular case of BSGD, where I set my batch size to 1, I have also tried this and the training time seems to be longer. I am computing the likelihood on every epoch and checking if it is always growing, since I am maximizing the likelihood on my code. Also, my stopping criteria would be when I reach 2000 epochs or the change on the likelihood value is less than 1 in one epoch. In addition, I held 10% of the training data out for validation and would check accuracy and RMSE on both training and validation on each epoch. I ended up using the batch size of 1000 for converging fast. My update for the weights would then be W = W + learning\_rate\*(currX.T\*(currY – pred\_prob) - lambda\_value\*W) where currY and currX would be the one hot encoding for the labels in this batch and the instances of this batch respectively and pred\_prob would be the prediction probability per class.

1. **[10 pts]** After implementing your model, please use these two types of prediction to calculate and report the Accuracy and RMSE (See definition in Evaluation part) on the entire training set with the two features designed in Task 2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Feature | CTF | | DF | |
| Dataset | Training | Development | Training | Development |
| Accuracy | 60.03% | 59.61% | 59.96% | 59.62% |
| RMSE | 0.9 | 0.79 | 0.9 | 0.79 |
| Parameters Setting | Learning Rate alpha=10-4 Regularization Parameter lambda=0.1  How many iterations used? 981 for ctf, 951 for df | | | |

**[10 pts] Multi-class Support Vector Machine**

After you figure them out, report only the accuracy on the training and development set using the two features designed in Task 2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Feature | CTF | | DF | |
| Dataset | Training | Development | Training | Development |
| Accuracy | 58.06% | 57.72% | 58.1% | 57.8% |
| Parameters Setting | Penalty = L2 , Loss = hinge squared C = 1, dual = True | | | |

1. **Feature Engineering (Continued)**

**[10 pts + 10 bonus pts]** Describe in details your most satisfying design and the corresponding considerations, use formula to illustrate your idea if necessary. Besides, report the evaluation results on training and development set here (The reported result here should match the record on the leaderboard).

For creating the features, the first thing I did was to compute the tfidf of each term. Also, I have stemmed the terms so that we would not be using two features to represent the same thing. The last and most important step was to use the mutual information formula to choose the features.

I have computed the mutual information value for each word/class pair. From this value, I have ranked, for each class, the top words on that class. Getting the top 400 words from each class list, then, created my feature dictionary. This way, I would have the most informative words from each class as features, making it easier to classify each class.

As my results, I got, for the training accuracy, 61.02% and the training RMSE was 0.87. For the development accuracy, I got 60.5% and dev RMSE was 0.76.

1. **One sentence of your feeling for this homework**

Is that good or not? Why?

I enjoyed the homework a lot. I think it gave me a better understanding of the whole pipeline from data extraction to prediction and the leaderboard was interesting so that I knew I was near the right result and could compare to other people in class. I would suggest using the same leaderboard system for the Netflix homework, where we were a bit lost of how good was our system and got caught by surprise when the grades came.