Predicting the Score of a Reddit Text Post
Mini-project #1

Casper Liu, Eric Quinn, Howard Huang

September 30, 2015

Data : `https://raw.githubusercontent.com/imcpr/linear_regression_project/master/reddit_full_data_76.csv`

1 Part 1

1.1 Using Gradient Descent

Given training examples $\boldsymbol{X}$ and $\boldsymbol{Y}$, and a random initial weight vector $\boldsymbol{w}$, the weight vector used for approximating $\boldsymbol{Xw} = \hat{\boldsymbol{Y}}$ can be computed by repeatedly applying the following equation:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha[2(\boldsymbol{X}^T\boldsymbol{X})\boldsymbol{w} - \boldsymbol{X}^T\boldsymbol{Y}]$$

This is repeated until the the sum of the changes to each element of the weight vector is smaller than some number ϵ.

At first, an arbitrary value of 0.001 was chosen to be α, which drove the weights to infinity within the first few iterations. The lack of convergence is attributed to the widge range of values in the dataset, making it easy to mis-tune the weight vector and overshoot the gradient. After normalizing the dataset, a converging set of weights were achieved with the same α value. By running k-cross-validation with k=5, α=0.1 gave the optimal total error, out of the arbitrary set of 0.5, 0.3, 0.2, 0.1, 0.09, 0.07, 0.05, 0.01, 0.005.

The other hyperparameter is ϵ. The value of ϵ was chosen by hand to be a very small number, but large enough that the algorithm does not need to run for too before terminating. The value used was $\epsilon = 10^{-6}$.

The performance on gradient descent further improved when L2 regularization is added, resulting in the equation:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha[2(\boldsymbol{X}^T\boldsymbol{X})\boldsymbol{w} - \boldsymbol{X}^T\boldsymbol{Y} + 2\boldsymbol{\lambda w}]$$

Again, this newly introduced hyperparameter is tuned using k-cross-validation, where the optimal value is $\lambda = 10$. The data was split between a train-

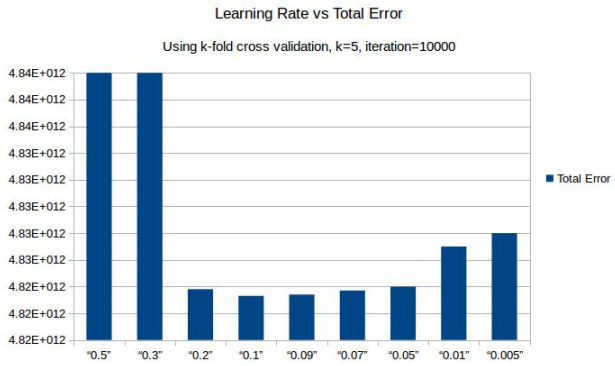

Figure 1: Learning rate (α) vs Total Error in 5-fold-cross-validation

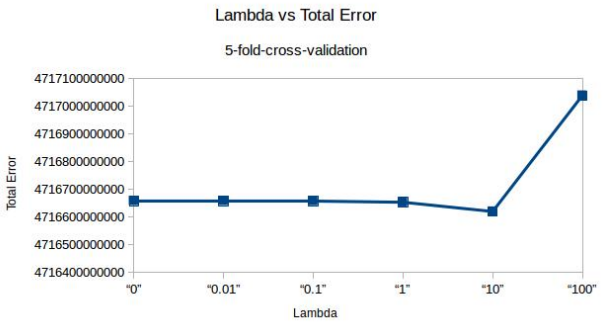

Figure 2: λ vs Total Error in 5-fold-cross-validation

ing set and a test set. The training set consists of the first 30000 training examples in the CSV file, and the test set consists of the remainder of the data.

1.1.1 Analysis

A mean squared error of 1.3×10^8 is achieved by the gradient descent algorithm using the hyperparameters described above. The MSE is very close to the one obtained using the closed form solution, which represents the global optimum. This demonstrates that the gradient descent method did not get trapped in a local minimum and is indeed moving towards the optimal point. However, to run the iterative algorithm until the difference between successive $\boldsymbol{w}$ is less than ϵ required many iterations and would take hours to converge.

The learning curve displayed below is obtained by shuffling the data and obtaining a 75% training set, and 25% test set. The training set has an overall lower MSE, and the testing set higher. However, since the error value seems to stabilise in further iterations for both sets, there does not seem to be an overfitting problem and the error is mostly attributed to the limitations of the feature and the model.

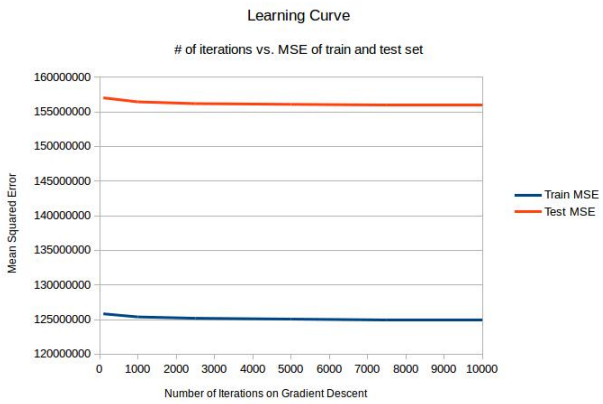

Figure 3: Learning curve for Gradient descent

1.2 Using the Matrix Equation

Given training examples in the form of a matrix $\boldsymbol{X}$ and $\boldsymbol{Y}$, where $\boldsymbol{X}$ contains the features and $\boldsymbol{Y}$ contains the expected output, there is a closed form solution for finding a weight vector $\boldsymbol{w}$ such that $\boldsymbol{X}\boldsymbol{w} \approx Y$. This vector can be computed as follows:

$$\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T Y$$

A problem encountered here is that the matrix $(\boldsymbol{X}^T\boldsymbol{X})$ is very close to being singular, which led to inaccuracies in the inverse, and thus the resulting weight vector. With the use of ridge regression instead, not only does it make the resulting vector non-singular, it also penalizes the use of large weights, so the weights are kept smaller, and (by Occam's Razor) would likely generalize better.

$$\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T Y$$

The data was used as is without any preprocessing, although the error rate may have been lower if that were not the case.

The only hyperparameters in this case is λ. This was chosen by comparing various values ranging from 10^{-10} to 10^{10}, computing the weights with that λ and the training set, and checking the mean squares error on the validation set (See figure 4). This ensures that the chosen λ works well in generalizing to unseen data. The training set chosen for this procedure was a uniformly random subset of 80% of the available data, and the rest was split evenly between the validation and test set.

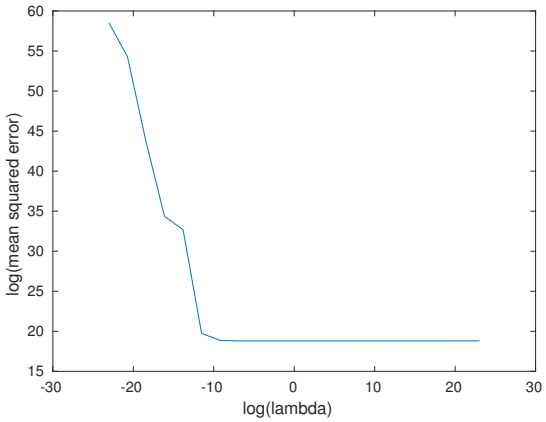

Figure 4: Error curve as a function of λ

1.2.1 Results

The optimal λ was found to be 0.01. This was used along with the training and validation set together to compute the weight vector. The test set was then used to check the error of the predictions. Averaged over 30 runs, the average mean squares error of the testing and training set were both approximately 1.3×10^8.

1.2.2 Analysis

A mean squares error of 1.3×10^8 means that the average difference between the actual number of shares

and the predicted number of shares is $\sqrt{1.3 \times 10^8} = 1.1 \times 10^4$. Using the least squares error to evaluate the predictions, this approach is only marginally better than guessing 0 for everything (Which leads to a 1.46×10^8 mean squares error). The least squares error function applied to linear regression is convex, and has a closed form solution which returns the global optimum, so this is not a problem of falling into a non-optimal local minimum. Additionally, the error is equally high both on the training set and on the test set, so it is not due to overfitting. Therefore, the most likely explanation for this error is the high bias in the model, due to the restriction to linear models and being unable to recode features. It is likely that better features could give a better analysis. Another possible source of errors lie in the fact that we cannot see where else an article is being posted. With such a large difference in possible values (1,000 vs 100,000) it seems likely that an article receiving many shares will be seen by more people and shared even more times. These kind of exponentialy sharing rates are not easy to predict in a linear function.

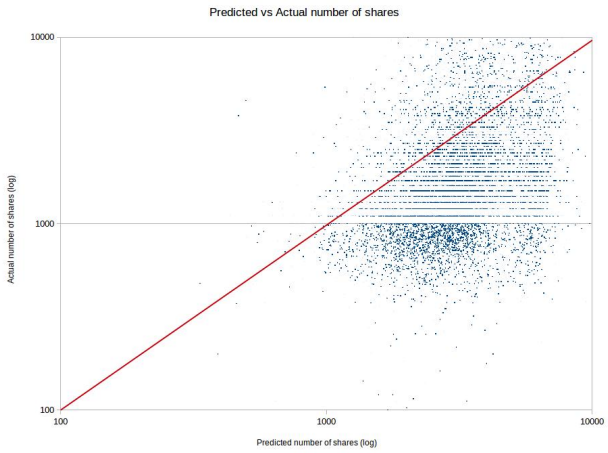

Figure 5: Actual vs Predicted shares

2 Part 2

2.1 Topic Choice

Reddit.com is a popular website, 34th in popularity worldwide [4], which means content posted on this website have a very large and varied potential audience. Studying which posts will receive the most Karma, and consequently the most views, can provide a lot of information regarding on how, what, when, and where to post in order to maximize exposure. This "Karma score" is determined by readers up-voting and down-voting posts, and defines how far up on the site they will be pushed for other people to view.

Because anyone can post on the website there is a very low barrier to entry, which means that posts will be of varying quality compared to articles on a news site. Therefore, instead of the topic of a post being the largest influence on the exposure it receives, it is plausible that the writing style of a post will have a greater impact on the reach of the submission.

The chosen data is from four specific subforums because they are all popular on the website, leading to a better distribution of the resulting scores. In addition, these subforums only allow text posts, which keeps all the content constrained to just this one website. It also means that all the content of the post is available within reddit.com itself.

2.2 Scraping techniques

In order to collect the posts, the authors used a python scraper to browse through the "new" pages of the subforums in order and collect the links to all of the posts. We then used these links to access the permanent link to each post, which can be used to collect the actual information about the submission, not just the metadata that appears on the search page.

The authors used the Python Reddit API Wrapper [1] to access reddit.com in order to collect data about each submission to the four chosen subforums of the main website. This API scrapes the information about the post when given the permanent link to a submission.

The posts used are from only as recent as two days old because at that point the submission will have fallen off the main pages for most users and the karma score should change less. This means that the data we are using is to predict the final score of a post before publishing or just after, without regard for how well it does in the first few minutes of its time on the website proper.

2.3 Feature Selection

The features selected are those that give a character of the submission. While not being an analysis of the actual topic of the post, the features include all the metadata of the post as well as some indication of the writing style. More interesting writing styles may be responded to more positively, leading to a higher score. Some analysis of the topic of the post was done however. Reddit's userbase skew younger and therefore more liberal than the general population, so more liberal or libertarian posts can be expected to

do well. In order to understand the political leanings of the post, the authors utilized Indico.io, which does feature extraction on text and images. Here Indico was used to get the rate of similarity to certain political viewpoints. It was also used to receive sentiment analysis about the post, whether the text has a positive or negative attitude. More controversial posts may have a negative attitude and not be responded to well.

The intent of the prediction is that it can be used before submitting a post, as all the information included in this study is information that can be understood before the post is created. The authors did not use characteristics such as the number of comments, which could possibly give a better result but are unavailable before the post has matured.

```
0: num words in title
1: num words in text
2: time_of_day
3: time_of_day_squared
4: day0
5: day1
6: day2
7: day3
8: day4
9: day5
10: day6
11: sub_reddit_0 (change_my_view)
12: sub_reddit_1 (relationships)
13: sub_reddit_2 (self)
14: sub_reddit_3 (tifu)
15: sentiment_analysis
16: liberal
17: libertarian
18: conservative
19: green
20-69: top 50 words
70: link rate
71: imgur.com link rate
72: youtube.com link rate
73: bold rate
74: italic rate
75: unique word rate
76: karma score
```

- 2: The time of day when the contents were posted. The value is the number of seconds elapsed since the day started.
- 3: Same as above, but squared.
- 4-10: A sparse vector representing the day of the week when the contents were posted. The feature corresponding to the posting day has a value of 1, and the rest are 0.
- 11-14: A sparse vector representing the subreddit to which the contents were posted. The feature corresponding to the post's subreddit has a value of 1, and the rest are 0.
- 15: Sentiment analysis using the Indico library [2]. The value represents how positive or negative the contents of a post are. The value's proximity to 1 indicates more positive content, while a value closer to 0 represents more negative content.
- 16-19: Political analysis using the Indico library [2]. The value represents the political alignment of the post. The closer a post is to a certain political ideology, the higher the value. The value is bounded between $[0, 1]$.
- 20-69: The top 50 words were found by aggregating the text from every reddit post in the data set. Using the Natural Language Tool Kit [3] the authors could exclude a list of common English stop words and aggregate the most common words, taking only those not in the stop list. These words should be able to represent the submissions that are most and least popular.
- 71-73: These items give extra information about the post. 70 checks the rate of links according to the reddit format. 73 and 74 check for links to the two most linked sites from reddit, in the belief that links to them may make a post more popular, as with the rest of reddit. This is given as a rate of links to the length of the post, so a post of only a link will be rated highly.
- 74-75: Checking for the rate of marked up text in a document. The hypothesis is that more marked up text will be more interesting to readers and therefore be given a higher score. This counts the rate of appearances compared to the overall text, so the entire post being in italics will only give a small weight.
- 76: The rate of unique non-stop words in the submission text. A higher number indicates that the vocabulary is higher. However, a shorter post may have less time to repeat itself in the number of words used.

Multiple approaches were attempted for features 20-69, the 50 most popular words. Instead of simply taking the 50 most popular non-stop words, for each word in the corpus of text the total score of all the submissions containing that word were added and then divided by the number of occurrences of

the word, giving the average score given to a post per instance of that word. This approach can give an idea of the topics that are popular on the website. However, this data is skewed heavily by a handful of popular posts with the inclusion of an uncommon word. In addition, a misspelled word in a popular post might only appear once in the corpus and be given a large score. The authors went over the data found and removed words expected to be outliers, including misspellings. However, in the end this data gave a worse result than simply using the most popular 50 words in the corpus, so it was removed as a feature.

2.4 Challenges

Karma is not necessarily static in the time frame used.

People can raise or lower the score of a post, so it is not as simple as the number of people that see a post. This is clear in some of the data that has many comments but a score of 0, indicating it was very controversial but not agreed with enough to become popular.

Karma is hard to predict because when something becomes more popular, even more people will see it, it can grow exponentially. Posts vary substantially in popularity based on the first few minutes of their time on Reddit.com. If they gain popularity at the start, then they will appear to more people and grow in popularity even more. This is doubled by the fact that old posts will not show up to people browsing the website in the normal manner, so an old post is unlikely to be found and increase in popularity later.

Reddit makes it difficult to collect posts over a long time frame because their API does not allow collecting all posts in a given date range. The "new" page that was used for scraping only collects the most recent thousand posts or so, which makes collecting a large sample size difficult. it is possible but more difficult to get older posts than that.

2.5 Results

Using the same procedure as part 1, the optimal value of λ was found to be 10^{-4} (See error curve in figure 6). The Mean Squared Errors on both the training and test set were 5.4×10^4. The Mean Squared Percent Error was 1155. Then the Root Mean Squared Percent Error is around 34. This is heavily weighted by the fact that many posts receive a score of 0, so low predicted values will match that more often. However, when replacing all negative values with 0, we still get an MSE of 5.4×10^4, a Mean Squared Percent Error of 1127, and a Root Mean Squared Percent Error of 33.5, which is not lower by a significant degree.

The number of upvotes in the training examples range from no upvotes and some number of downvotes (resulting in a final score of 0), up to $4,668$.

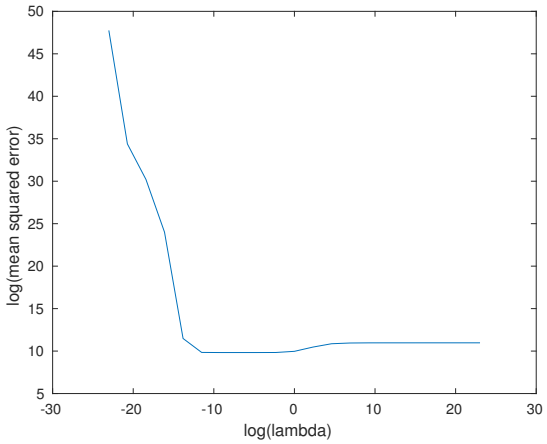

Figure 6: Error curve as a function of λ

2.6 Analysis

As with the results from the Mashable data set, the predicted values did not correspond with the actual values very frequently. Despite tracking the writing style, sentiment, and political leanings of the post, a good estimate was not being reached. It is possible that future research would be made better suited to looking at the topics themselves as well as the person posting, which might provide more data than we were able to extract. It is also likely that another model would be better than the linear regression, which is limited in what it can predict.

In addition, when we capped our minimum predicted value, to 0, we barely saw a change in the predicted error. This comes from the fact that the negative values did not go far into the negative section, but posts with a large score were often not predicted well and were weighted much more highly.

3 Statement of Contributions

Everyone wrote question 1 separately. The version submitted came mostly from Howard and Casper. Casper and Eric did all the scraping and data collection, as well as implementing the collection of features, including the use of the indico.io analysis and Natural Language Tool Kit. Howard wrote the sections of the paper on using the closed form matrix so-

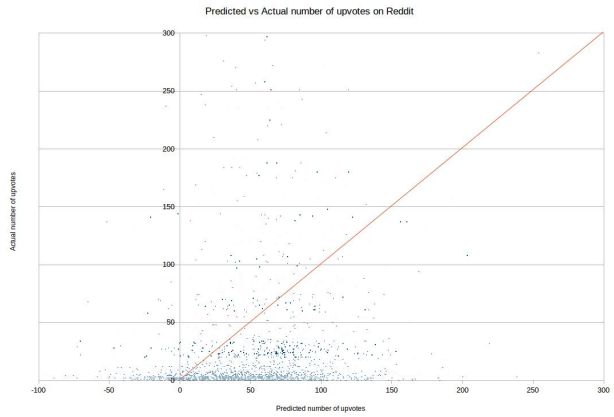

Figure 7: Score predictions with negative values

lution. Casper wrote the section on gradient descent. Eric wrote Topic Choice, Scraping Techniques, Feature Selection (with help from Howard), and Challenges.

We hereby state that all the work presented in this report is that of the authors.

References

[1] Boe, Bryce. (2015) Python Reddit API Wrapper [Python Code Library]. Available at `https://github.com/praw-dev/praw` (Accessed 26 Sept. 2015) - See more at: `https://praw.readthedocs.org/en/stable/`.

[2] Indico.io API team. (2015) [Python Code Library] Available at `indico.io` (accessed 26 Sept. 2015)

[3] NLTK Project. (2015) [Python Code Library] Available at `nltk.org` (accessed 26 Sept. 2015)

[4] "Reddit.com Site Overview." Alexa. Alexa Internet Inc., 27 Sept. 2015. Web. 28 Sept. 2015. `http://www.alexa.com/siteinfo/reddit.com`.