COMP 551 Applied Machine Learning - Assignment 1 Report

Alexander Harris, Abbas Yadollahi, Bao Nguyen

January 31st, 2019

1 Abstract

In this project, we utilized a linear regression model to predict popularity of comments on Reddit based on features provided either from the raw text or basic information about the comment. Linear regression was implemented using two different approaches: a closed-form solution as well as using gradient descent with a fine-tuned learning rate. The dataset being examined consists of 12,000 comments from the AskReddit subreddit (Reddit's form of subforum). After careful experimentation, it was found that the closed-form approach was multiple orders of magnitude faster than the gradient descent approach, however it is slightly more unstable. We found adding features like the number of words in a comment, number of curse words and number of capitals improved our models performance while features such as number of URLs or sentiment analysis had a marginal effect on performance. Our final model performed fairly well on the test set with a mean square error (MSE) of 1.2937.

2 Introduction

In this project, we tackled the task of predicting the popularity of Reddit comments. This is realized by training a linear regression model on the dataset provided, using two different approaches, namely closed-form method and gradient descent. The features chosen for predicting the popularity of such comments consist of a mix between Reddit's proprietary statistics associated to each comment and custom features we extracted from the raw comment's text. Python is the chosen programming language for this project because of its wide range of available big data frameworks/libraries, of which the library NumPy is extensively used for the purpose of computation and representation of data.

To predict the popularity of a Reddit comment given its features, we implemented a linear regression algorithm which trains on the 12,000 comments from the dataset and then outputs a weight vector. The latter is then used to create a linear model which represents our estimates for the popularity of the comments.

The linear regression model can be implemented in two separate ways: the closed-form solution or the gradient descent solution, which can be modeled by the following equations respectively.

$$\hat{w} = (X^T X)^{-1} X^T y$$

Figure 1: Closed-form Linear Regression

i = 1;
do
  $\alpha = \frac{\eta_0}{1+\beta i}$;
  $\mathbf{w}_i = \mathbf{w}_{i-1} - 2\alpha\left(\mathbf{X}^\top \mathbf{X}\mathbf{w}_{i-1} - \mathbf{X}^\top \mathbf{y}\right)$;
while $\|\mathbf{w}_i - \mathbf{w}_{i-1}\|_2 > \epsilon$;

Figure 2: Gradient Descent Linear Regression

After implementing both these algorithms, we were able to fine tune our model by comparing our predicted popularity scores to true scores of the validation set. Our goal was to create a model with a minimal mean-squared error (MSE) when compared to the validation set, which was computed as follows.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

Figure 3: Mean-squared Error

By the end of our experiments, we found that most of our custom features greatly improve the accuracy of the constructed model compared to only using Reddit's proprietary statistics. The feature which contributes the most to reducing the MSE is the number of words in each comment, which came as no surprise. On the other hand, our custom sentiment analysis feature negatively impacts the MSE which was a shock seeing as we expected it to greatly improve the prediction accuracy. As a result, it was one of the features we did not use in our final model for predicting the popularity scores of the test set. The results of the latter are discussed in more depth in the following sections.

3 Dataset

The dataset consists of 12,000 real world comments taken from the subreddit AskReddit. The data was presented as a JSON array, with each entry consisting of a dictionary containing the raw text as well as the following information: `popularity_score`, `children`, `controversiality` and `is_root`. The data was divided into 3 sets: a training set containing the first 10,000 comments, a validation set containing the next 1,000 comments, and a test set containing the last 1,000 comments. We performed some pre-processing on the text itself, namely lowercasing and splitting on whitespace, as well as creating an additional text field in the dictionary where we removed punctuation using a regex filter.

When it came to coming up with additional features, we decided on the following:

- `num_words`: Simply the total number of words contained in the comment, including punctuation.
- `num_capitals`: The number of capitals letters contained within the comment.
- `num_curse_words`: The number of curse or swear words contained within a comment. We check each word in a comment against a database of curse words.
- `links`: The number of URLs present within a comment.
- `sentiment`: The overall sentiment of a comment (positive/neutral/negative). Computed using NLTK and the VADER[1] lexicon.

We also took the logarithm of `num_words` and added that as a feature as well.

Ethically, one could make the argument that we obtained and used this data without every individual users consent, however since these are comments posted on a public website and do not have any personally identifiable information such as usernames, we do not believe that this constitutes a significant issue.

4 Results and Findings

4.1 Comparison of runtime and performance between closed-form and gradient descent implementations

We compared the runtime and performance of our two linear regression methods using the most basic feature vector containing only the `children`, `controversiality` and `is_root` features as

well as the bias term. This was done in order to speed up computation time to allow us to measure average runtime over multiple iterations. We first measured the average closed-form runtime, over 20 iterations we found it to be 0.0039 seconds. Comparatively, the gradient descent average runtime using a learn rate of 10^{-6}, decay speed of 10^{-12} and epsilon of 10^{-7} was 0.34 seconds for a random initial weight vector and 0.30 seconds for a zero initial weight vector. We believe the reason for the difference between the initial weight vectors used is due to the additional computation required to generate a random vector compared to a vector of zeroes. In any case, both methods gave almost identical MSE on both the training and validation set. We can also observe that gradient descent is much slower than the closed form solution as expected, however it does give more or less identical MSE on both the training and validation set of 1.08468 and 1.02032 respectively.

In order to test the stability of our gradient descent approach, we tested multiple combinations of learn rates and decay speed. The following figures show the MSE we obtained using these different combinations and an epsilon of 10^{-7}.

Learn rate/decay speed	1e-05	1e-06	1e-07	1e-08	1e-09	1e-10
1e-05	1.08468	1.08468	1.08468	1.08468	1.08468	1.08468
1e-06	1.08468	1.08468	1.08468	1.08468	1.08468	1.08468
1e-07	1.08469	1.08468	1.08468	1.08468	1.08468	1.08468
1e-08	1.08883	1.08484	1.08471	1.08471	1.0847	1.0847
1e-09	1.10789	1.09433	1.08849	1.08702	1.08687	1.08685
1e-10	1.31064	1.13619	1.1141	1.10975	1.10927	1.10922

Table 1: MSE for Gradient Descent on Training Set

Learn rate/decay speed	1e-05	1e-06	1e-07	1e-08	1e-09	1e-10
1e-05	1.02033	1.02033	1.02033	1.02033	1.02033	1.02033
1e-06	1.02033	1.02033	1.02033	1.02033	1.02033	1.02033
1e-07	1.02043	1.02035	1.02035	1.02034	1.02034	1.02034
1e-08	1.02567	1.02091	1.02055	1.02052	1.02052	1.02052
1e-09	1.04722	1.03106	1.02533	1.02381	1.02364	1.02362
1e-10	1.25706	1.07942	1.0544	1.04938	1.04882	1.04876

Table 2: MSE for Gradient Descent on Validation Set

Learn rate/decay speed	1e-05	1e-06	1e-07	1e-08	1e-09	1e-10
1e-05	0.0489182	0.0330427	0.0382464	0.0456505	0.0340171	0.0448442
1e-06	0.350077	0.273121	0.30565	0.313023	0.254367	0.254817
1e-07	3.12573	1.91857	1.72181	2.00232	1.65269	1.8404
1e-08	9.08036	12.3187	11.5575	10.9516	10.3267	10.2265
1e-09	6.65009	9.04114	23.757	27.4603	29.5138	29.0245
1e-10	6.76558	9.11673	13.4172	14.5175	15.9303	14.4766

Table 3: Runtime for Gradient Descent

As we can see from Table 1 and Table 2, the MSE actually increases as we lower the learning rate, while lowering the decay speed decreases the MSE, the effect being more pronounced at lower learning rates. As for the effect on runtime, we can see in Table 3 that decreasing the learning rate can significantly increase the runtime, with higher learning rates having less of an effect. Decreasing decay speed seems to have an inconsistent effect on runtime at lower learning rates, however higher ones appear to produce the expected response. Since the closed form method appears to be fairly stable with our features and performs much better than the gradient descent solution, we will be using it for the remainder of our testing.

4.2 Comparison between different size text features

For this experiment, we used the closed-form linear regression algorithm to compute linear models for different amounts of top words features. Seeing as we had already processed the Reddit com-

ments to create a list of the top 160 occurring words, we could compare the effectiveness of each word on the prediction model. To do so, we created 80 models that use anywhere from 0 to 160 of the top words as features, as well as the bias and Reddit's proprietary features. To evaluate the accuracy of each model, we computed the MSE against the training set and the validation set, which can be seen in the following plot, where the x-axis represents the number of top words and the y-axis represents the MSE.

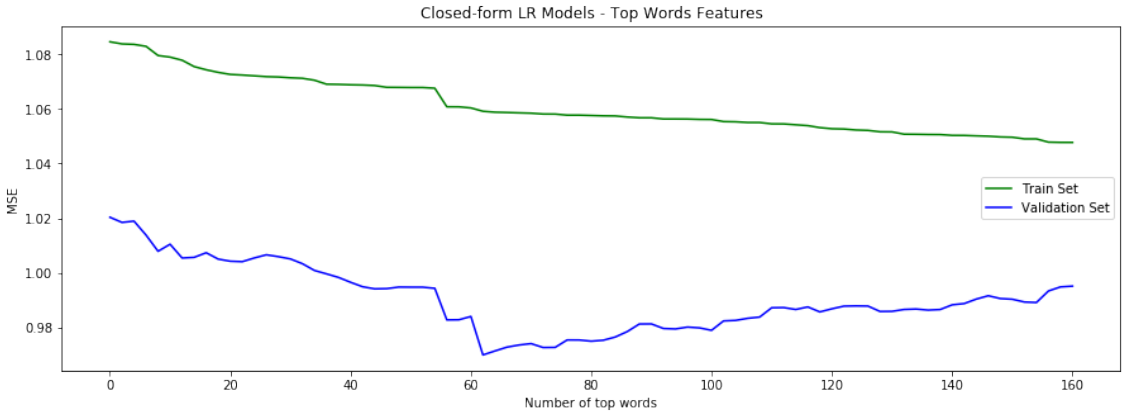

Figure 4: Closed-form Linear Regression Models for Different Number of Top Words Features

From the Figure 4 above, we can see that the accuracy peaks when using the top 62 words. Using too little top words leads to underfitting as can be seen for the first 50 top words. This is expected as we will have high training error and low validation error. On the other hand, as we increase the number of top words used, our model slowly begins overfitting. At this point, we will have low training error and a higher validation error.

4.3 Performance of additional features

For this section we used a text feature size of 60 and the closed-form solution for linear regression. First we tested the improvement from just `num_words`, which gave us fairly insignificant improvement on both the training and validation sets. However when using the logarithm of `num_words`, we obtain improvements of 0.004878 and 0.009882 on the training and validation sets respectively. We believe this is due to the fact that `num_words` can vary quite a lot between comments, so taking the logarithm helps to normalize that data.

We then tested `num_capitals`, which produces a small improvement on its own, however combining it with our previous features gives us an improvement of 0.005008 on the training set and 0.01013 on the validation set. On its own, `num_curse_words` gives a small improvement of 0.001844 and 0.001685, which yields improvements of 0.006259 and 0.01080 when combined with the previous features. As for `links`, it gives an insignificant improvement on its own and 0.006480 and 0.008753 when combined with the other features. Finally `sentiment` provides an insignificant improvement on its own, which was unexpected since we thought it would provide the best performance out of all our features. When combined with the other features, we see improvements of 0.006318 and 0.01019 for the training and validation sets respectively.

Considering these results, we have decided the leave `links` and `sentiment` out of our final, as the seem to provide little to no benefit, and in the case of `sentiment` significantly increase computation time for the features. Therefore our final model will include all the basic features, as well as a text feature of size 60, `num_words`, `num_capitals` and `num_curse_words`. This model gives MSEs of 1.0606 on the training set and 0.98697 on the validation set, with improvements over the baseline of 0.006259 on the training set and 0.01080 on the validation set.

4.4 Performance of best model on test set

Testing our final model on the test set, we obtained an MSE of 1.2937. This is around 30% higher than our MSE for the validation set, which while higher than we hoped for is within our expectations.

5 Discussion and Conclusion

Our main takeways from this project are twofold. First, we learned how to apply two different forms of linear regression to a practical problem involving messy, real-world data. Also, we learned that choosing good features to include in our model is not always straightforward or obvious. We thought that sentiment analysis would be a useful feature for our model, but in the end it did not have much of an impact. However, our model performs fairly well with the features we have and computation time for our features is much faster as a result of not including the sentiment analysis. Our MSE on the test set is also within 30% of the validation set.

In the future we could apply a version of this model to predicting which Reddit comments are likely to be gilded, or even use it for other websites such as YouTube or Twitter.

6 Contribution Statement

All team members have made significant personal contribution towards this project. The amount of work for each is described as follows:

- Alexander Harris: Dataset loading and preprocessing, implementation of linear regression algorithms, model training, validation and testing, write-up contribution
- Abbas Yadollahi: Dataset loading and preprocessing, implementation of algorithms, model training, validation and testing, write-up contribution
- Bao Nguyen: Implementation of algorithms, write-up contribution.

References

[1] C. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," *Eighth International Conference on Weblogs and Social Media (ICWSM-14)*, 2014.