Analyzing the Popularity of News Articles Using Linear Regression

Jonathan Campbel
School of Computer Science
jonathan.campbell@mail.mcgill.ca

Yann Aublet-Longpr
Department of Biology
yann.aublet-longpre@mail.mcgill.ca

Jian (Ethan) Li
School of Computer Science
jian.li9@mail.mcgill.ca

***Abstract*—We apply linear regression to two datasets of news articles, one newly-created, to predict popularity of the articles and present our findings.**

1. Introduction

In this project, we build a machine learning (ML) system to predict the popularity of online news articles of two datasets, each with different features. Indeed, the online news platform is becoming a very important source of information. Its low cost and relative ease of distribution mean it can reach a vast audience, but it also means that there exists an intense competition for attention. The resulting asymmetry in the distribution of collective attention makes work like the present one extremely valuable to content providers, journalists and advertisers alike.

Two separate approaches have been taken to solve the problem at hand. The first method is to take data available immediately after the article publication to predict its final popularity. The second is to predict the popularity using only the information available prior to the publication. The features employed for the latter are metadata, such as topic, authors, date of publication, number of words, etc as well as semantic characteristic, like keywords or natural language processing.

Many different target variables can be used to express the popularity of content: the number of views, comments, shares and user votes all work well as metrics for popularity. In this project we explored two particular metrics. In Part 1, where we analyze an existing dataset of Mashable articles [1], the number of shares is our target variable, while in Part 2, where we analyze a new dataset of CBC News articles[1], we use the number of comments.

1.1. Related Work

In related work [2], Szabo and Huberman studied, using the platforms Digg and YouTube, how popularity of an online content could be predicted soon after submission. They found a linear correlation between the logarithm transform of the early and later popularity. The prediction made by their linear regression by least-squared error displayed a lot of variance, while minimizing over relative errors displayed considerably smaller dispersion. Consequently, they favored the relative error model.

Similarly, Tatar et al. [3] showed how a simple linear regression model could help to predict the volume of comments from an online news article based on the number of comments observed within a certain period after publication.

Next, Tsagkias et al. [4] presented an exploratory work on predicting the volume of comments in news articles prior to publication. They looked at datasets for many sources and addressed the prediction task as one of classification.

Bandari et al. [5] worked on predicting the popularity of news items on Twitter using features extracted prior to the article release. Their results show that these features might not be sufficient to predict the exact number of tweets, but can be effective at a similar classification task.

Finally, Fernandes et al. [1] explored the same dataset that we used for Part 1 and implemented five different classifier models to first predict which articles would become popular and then optimize a subset of the features to enhance the predicted popularity.

1.2. Outline

In the present work, we applied linear regression on a set of features to predict a target variable which represents popularity. We applied our algorithm to two separate datasets: the first one containing Mashable articles taken from the UCI Machine Learning repository articles, and the second containing articles from CBC News which were gathered by a web crawler.

This paper is divided as follows: first, we present our datasets, and describe how they were acquired and processed. Then, we introduce linear regression and show how the method of least-squares can be solved both in closed-form and using gradient descent. Next, we expose our experimental results. We conclude with a brief discussion of the usefulness of the project, offering future ideas for dataset acquisition.

1. The dataset is available for download at http://www.campbelljc.com/cbc.csv

TABLE 1. STATISTICAL MEASURE FOR THE TRAINING SET

Number of Articles	31,715
Average Number of Shares	3,424
Maximum Number of Shares	1
Minimum Number of Shares	843,300
Standard Deviation	12,323

2. Methods

2.1. Data Acquisition and Preparation

2.1.1. Part 1. For the first part of our project, we used a dataset consisting of Mashable articles prepared by Fernandes et al. [1] from the UCI Machine Learning repository [6]. The dataset consists of 39,797 articles published over a period of two years. There are 58 predictive features, all of which are obtained prior to publication of an article. These features can be broken down into seven categories: words, links, digital media, time, keyword and natural language processing. The target variable is the number of shares.

The only transformation that we did on the features was to normalize them, which must be done in order to apply ridge regularization. As we have seen in class, this linear transformation does not affect the decision boundary. All features are scaled and restricted to be between -1 and 1. Apart from this, no feature recoding was done since it was not allowed in this part of the project. Further, we chose not to drop any of the features present in the database, even though we noticed quite a bit of redundancy in the data.

Finally, training and testing data were split in preprocessing, that is, after the data was randomized to discard any latent pattern in the dataset. 80% of the original data was allocated to the training set while 20% was kept in the testing set for evaluation.

2.1.2. Part 2. The CBC News website was chosen for the acquisition of our news article dataset. We based our crawler code on an open source Python project [7], which has the ability to render crawled pages in a browser setting, so that the asynchronous JavaScript (AJAX) and other communication protocols (e.g. WebSocket) can execute and fetch various parts of the page. This feature is important since, on any particular CBC News article page, several parts of the page are only loaded after rendering has completed through the aforementioned methods.

The crawling code was subsequently modified from the original to conform specifically to the chosen website. In particular, the area of the Internet to crawl was limited to pages beginning with the URI "http://www.cbc.ca/news", the domain and directory in which the news articles reside. Furthermore, several subsections of said location were disallowed, including the Sports, Interactive, and Archives subsections of the site, due either to a lack of news articles in those directories or to non-conformity with established (regular) article HTML format.

When a news article is found by the crawler, metadata about the article will be extracted and placed into a new datafile. This metadata includes the URL, article category, sub-category, story flag (whether the piece is an Analysis piece, e.g.), title, description, author, post date, number of headlines within the article body, and the numbers of paragraphs, videos, links, inline images, shares, and comments. The approach to save each article's metadata within its own file was taken to preserve data in case of crawler instability (due to potential non-conforming news article HTML structure).

Any article where the crawler was unable to extract number of comments and/or number of shares was discarded. Some complete sections of the CBC site do not have comment sections (e.g. news local to Hamilton, Ontario), while other individual articles have no comment section, possibly done on purpose by the article author. Further, some articles were discarded because either comment or share sections of the site did not load in time for the crawler to save them. For purposes of time, the crawler devoted at most 10 seconds waiting for each page to load.

In total, approximately 3,000 news articles were crawled and their metadata parsed for our dataset. A further 3,500 articles were crawled and discarded due to the aforementioned reasons.

A further Python script takes in all article metadata, and does further processing on it to extract further relevant features. These features include number of words in title, number of authors, average length of words in title, number of numbers in title, whether the article was posted on a weekday or weekend, and whether it was posted in the morning, afternoon, or evening. The script also converts some numerical values to binary ones (e.g. a category of Aboriginal news would set the isAboriginal feature to 1). All features end up being represented by numerical or binary values (see Appendix 1 for a complete listing of features).

We also used a Natural Language Processing library to calculate a sentiment value for the title of each news article [8]. The sentiment value of an article could be *very negative* (assigned numeric value -2), *negative* (-1), *neutral* (0), *positive* (1) and *very positive* (2).

2.2. Linear Regression

Linear regression is a widely used statistical tool for modeling relationship between predictory variables X_i (features) and some outcome y (target variable). In particular, we assume that our target variable y can be estimated from a linear hypothesis of the form:

$$f_W(X) = w_0 + \sum_{j=1}^{m} w_j X_j \tag{1}$$

The goal of our learner is to find the parameter vector W that minimizes the discrepancy between the predicted value $\hat{y}$ and the actual outcome y. We define the loss function to minimize as the mean squared error:

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

In this section, we display two algorithms that we implemented to solve this problem. We show how this can be solved first in closed-form, and second using gradient descent. Finally, we present Ridge regression and an approximated version of Lasso regression.

2.2.1. Closed-Form. For simplicity, let us re-write the equation (1) in matrix form:

$$f_W(X) = XW \tag{3}$$

where: X, the design matrix, is $n \times m$, with:
n := number of training examples
m := number of features
W, the parameter vector, is $m \times 1$

Likewise for the loss function:

$$L_s(f) = \frac{1}{n}(Y - XW)^T(Y - XW) \tag{4}$$

Where Y, the target vector, is nx1

To minimize the loss funcion, we take its derivative with respect to W and equate it to zero:

$$\frac{\partial L_s(f)}{\partial W} = 0 = \frac{-2}{n}X^T(Y - XW) \tag{5}$$

Solving for W, we get:

$$W = (X^T X)^{-1} X^T Y \tag{6}$$

Given that the matrix $(X^T X)$ is invertible, an exact solution to this equation can be found in closed-form. In our implementation, we use the jblas linear algebra for Java library (cite) to perform the matrix multiplications. We also implemented an algorithm to do the matrix inversion that find a matrix X^{-1} that solves:

$$X^{-1}X = I \tag{7}$$

where I is the identity matrix.

The computation cost for this algorithm lies mostly with the matrix inversion, which can be done in polynomial time with respect to the size of the matrix $(X^T X)$. Although this can cause problem when handling very large datasets, the method performs very well for our present application.

2.2.2. Gradient Descent. A gradient descent approach is also implemented for this project. In the gradient descent approach, instead of looking at close form solution, we look at the gradient of the error function (equation 5), choose a set of initial weights, and iteratively find the solution. This approach is specially useful when the data set is too large where a direct solution would not be possible.

One of the key part in the gradient descent approach is deciding the learning rate α. A learning rate too big would result in the algorithm not descending to a minimum. In fact, in part 1, a learning rate too big will result in the weights going to infinity after a few iterations.

A simple checking mechanism was implemented to make sure that the learning rate is not too big. Before actually starting the gradient descent, we check the least square error at the starting point, and calculate the least square error after 1 step of gradient descent. If the error gets bigger, that would mean the learning rate α is too big. We then scale it to a smaller value, and repeat the process until the error gets smaller after 1 step of gradient descent.

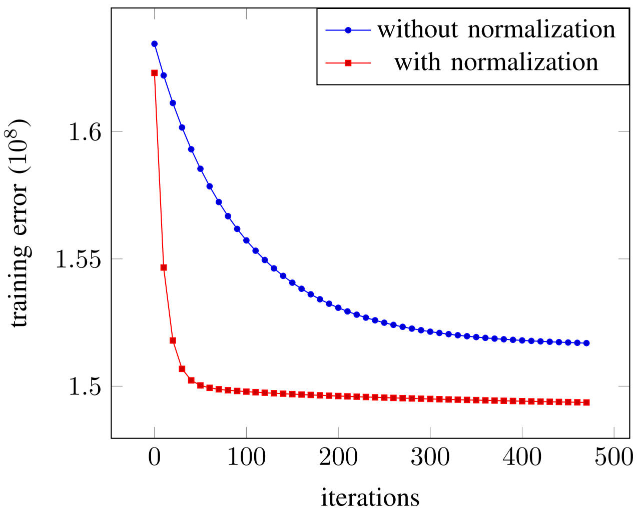

Figure 1. Gradient Descent: converging speed with and without regularization

We chose the 0 vector as the initial weights for the gradient descent.

One important reason we implemented data normalization is because of the gradient descent approach.

In our development process, we found that when the data is not scaled, it takes a very long time for the gradient descent to run. We used the above mentioned mechanism to find the largest possible learning rate α, and yet it has a very slow converging speed. See figure 1.

2.2.3. Ridge Regression. The solution given by the least-squares estimate is unbiased. In fact, we know from the Gauss-Markov Theorem that the least-squares estimate gives the best unbiased linear estimate, meaning that it has the smallest variance amongst all unbiased estimates. However, in some case it is better to add a little biasness in favor of a smaller variance. This can happen if the features are highly correlated. Ridge Regression or Tikhonov regularization can be used to alleviate multicollinearity amongst features. Mathematically, this is implemented by adding a second term to equation (2)

$$L_s(f) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda\sum_{j=1}^{m} w_j^2 \tag{8}$$

In essence, what this does is to prevent over-fitting by imposing a penalty on the model size, as characterized by the second term. The rest of the algorithm is similar to what we described in section 2.2.1, we minimize the loss function by taking its derivative with respect to W. Solving for W in $\frac{\partial L_s(f)}{\partial W} = 0$ we get:

$$W = (X^T X + \lambda I)^{-1} X^T Y \tag{9}$$

Which can be solved directly by the numerical machinery we set up in 2.2.1 or via gradient descent, as in 2.2.2. This solution shrinks the weight down, effectively reducing the model complexity, it is said to give a smooth solution. Note that the ridge regression is not invariant under scaling, which is why we normalized our data during pre-processing.

2.2.4. Lasso Regression. While Ridge regression gives a smooth solution, a more sparse solution could be preferred in certain scenarios. Lasso regression is more suitable for these situations [9]. Different from Ridge regression, the Lasso uses $L1$ regularization:

$$L_s(f) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{m} |w_j| \tag{10}$$

Ridge regression effectively shrinking the weights, but drives few weights to 0. The Lasso, on the other hand, has the property that if λ is sufficiently large, some of the coefficients w_j are driven to zero [10].

There is no close-form solution for Lasso regression and the implementation is harder than Ridge regression, and also more computationally expensive.

An approximated approach was presented in [11]. The approach was based on the following approximation:

$$|w_i|_1 \approx \frac{{w_i}^2}{|w_i|_1} \tag{11}$$

Plug it into equation 10, we can derive an iterative updating scheme:

$$w_{new} = (X^T X + \lambda diag(|w_{old}|)^{-1})^{-1} X^T Y \tag{12}$$

There are some problems with this approach, such as that once one feature is driven to 0, it will never be driven away from 0, and that it sometimes gets suboptimal results instead of the optimal result. However, we chose to implement this approach because implementation for this approach is very simple and it is not very computationally expensive [11].

Lasso regression can be used for feature selection.

3. Implementation

The linear regression algorithm is implemented through Java. The library *OpenCSV* [12] is used in this project to load and parse dataset files. The linear algebra library *jblas* [13] is used for matrix computation.

The web crawler was implemented with Python, and uses an open source Python project [7].

For part 2, we also used a Natural Language Processing library, Stanford CoreNLP [8], to calculate the sentiment of new titles.

4. Results

4.1. Part 1

The training and testing errors of different approaches tried in this project is listed in table 2. The epsilon value for gradient descent was set to 5.0.

TABLE 2. TRAINING AND TEST ERRORS FOR PART 1

	Training Error	Test Error
Close Form	1.48485×10^8	6.6537×10^7
Close Form (Ridge)	1.48488×10^8	6.6548×10^7
Gradient Descent	1.49116×10^8	6.6871×10^7
Gradient Descent (Ridge)	1.49117×10^8	6.6871×10^7
Lasso (Approximation)	1.48485×10^8	6.6537×10^7

TABLE 3. TRAINING AND TEST ERRORS FOR PART 2

	Training Error	Test Error
Close Form	3.81745×10^5	4.05639×10^5
Close Form (Ridge)	3.81666×10^5	4.05620×10^5
Gradient Descent	3.96606×10^5	4.13898×10^5
Gradient Descent (Ridge)	3.97207×10^5	4.14191×10^5
Lasso (Approximation)	3.81638×10^5	4.05578×10^5

As can be seen in the table, for our training and test data split, the test error is much smaller than the training error. However, in our k-Fold validation approach, the validation error could get much bigger than the corresponding training set.

And while the training error and test errors are similar across different approaches, the weights are different, especially between regularized version and regularized version. Before regularization, some weights could go to very large numbers. The bigger the λ gets, the less likely these large weights appear.

In our final data set, one outlier was removed (the outlier was reported on my courses). Before the removal of the outlier, approaches without regularization will regularly get weight vectors with big values, while the ones with regularization does not have this problem.

Finally, the approximated Lasso will get weights with more close to 0 values. For data from part 1, the feature ẅeekday_is_sundayänd the feature L̈DA_04g̈ets very close to 0 weights, while that is not the case in other approaches.

4.2. Part 2

Part 1 and Part 2 are using the same regression code, therefor the general program features stay the same. For our specific CBC news data, we found some interesting correlations between certain features and the result.

The results for part 2 is listed in table 3. The epsilon value for gradient descent for part 2 was set to 0.125.

On average, the features attributed the highest weights were the number of shares, followed by the number of videos and links present in the article. Of the categories, Politics had the highest correlation to the number of comments, followed less significantly by Canadian or Business news, with other categories having weights closer to zero.

5. Discussion

The number of shares was, as expected, the highest-weighted predictor of the number of comments: an article that gets more shares will get more people to go to the article page, and perhaps comment on it.

The number of videos was also a strong predictor as well as the number of links. This correlation could be because the author of an article will put more work into it if he/she believes it is a popular subject. Videos in particular are a time-intensive undertaking for a news organization to undertake, requiring manpower, planning, and financing. If a video is made about a topic or story, then presumably it is a very popular one, and thus the number of videos is a strong predictor. The number of links is also a strong predictor (though less so than videos) since a popular topic will have more related stories to refer to that happened in the past.

The fourth strongest predictor was if the news article was in the Politics category. This correlation makes sense since politics is one of the most divisive topics that can be reported on (and often political articles have a certain bias), leading to much heated discussion in the comment area, with each individual reader possibly commenting many times in reply to other commenters. Furthermore, Canadians have experienced 4 elections in the past 10 years (including the current one), showing that political issues and divisiveness are a constant issue in the country.

Other strong predictors were the number of images in the article, as well as number of paragraphs. One negative correlation was if the article had a story flag (such as Ánalysis,́ Ópinion,́ Ṕoint of View,́ etc.), then the number of comments would be predicted to decrease. It is possible that readers of CBC news prefer actual news articles to editorial pieces, or that such latter pieces do not get frequently discussed or shared by readers with other people.

6. Conclusion

We believe our dataset was quite useful in that it pulled a large amount of metadata from each news article. Further, both the number of shares and of comments were available for most articles, unlike similar news websites, making a (slightly) more diverse set of features.

One limitation with the particular website chosen for the dataset is that the crawling time is quite large, due to the need to render each page so that relevant information can be loaded from the server and saved. Through various networking errors, this delay resulted in low efficiency and high waste of articles which timed out and had to be discarded due to missing information.

Further potential features that could be extracted from the metadata include more processing of the actual article text, to determine keywords, sentiment, and related features extracted through textual analysis.

Also, it may be interesting to look more closely into the comment section to individual comments for further research. Temporal metadata of individual comments could be extracted, and the rate at which comments are made could be extracted and used as a feature.

...

7. Contributions

Jonathan Campbell was in charge of creating the web crawler, parsing the metadata to create the features, and loading the dataset into matrix form to be used by the algorithms.

Yann Aublet-Longpr was in charge of implementing the closed-form method, implementing the cross-validation as well as the different error measures.

Jian (Ethan) Li was in charge of implementing gradient descent algorithm, implementing the approximated Lasso regularization algorithm, integrating the Stanford CoreNLP library, and use it to analyze the sentiment of news titles.

All three team members worked on discussing and deciding the features to be used in the project, as well as writing the report.

8. Appendix 1: Attribute Information

Number of attributes: 31 (28 predictive attributes, 2 non-predictive, 1 goal field)

Attribute Information:

1) url: URL of the article (non-predictive)
2) title: Title of the article (non-predictive)
3) isAboriginal: Is article category Áboriginal?́
4) isArts: Is article category Árts?́
5) isBusiness: Is article category B́usiness?́
6) isCanada: Is article category Ćanada?́
7) isElections: Is article category Élections?́
8) isGoPublic: Is article category Ǵo Public?́
9) isHealth: Is article category H́ealth?́
10) isPolitics: Is article category Ṕolitics?́
11) isTech: Is article category T́echnology?́
12) isTrending: Is article category T́rending?́
13) isWorld: Is article category Ẃorld?́
14) hasSubCategory: Does the article have a sub-category?
15) hasStoryFlag: Does the article have a story flag (e.g. Ánalysis,́ Ńew,́ Úpdated,́ etc.)
16) numWordsTitle: Number of words in the title
17) numNumsTitle: Number of numbers in the title
18) avgLengthWordsTitle: Average length of words in the title
19) numAuthors: Number of authors
20) wasMorning: Was the article published in the morning?
21) wasAfternoon: Was the article published in the afternoon?
22) wasEvening: Was the article published in the evening?
23) wasWeekday: Was the article published on a weekday?
24) wasWeekend: Was the article published on a weekend?
25) numSubHeadlines: Number of headlines in the article body (indicating different sections)

26) numParagraphs: Number of paragraphs in the article body
27) numInlineFigures: Number of images in the article body
28) numVideos: Number of videos in the article body
29) numLinks: Number of links in the article body
30) numShares: Number of shares of the article
31) sentiment: Sentiment of the title
32) numComments: Number of comments on the article (target)
33) sentiment: The sentiment value of the title

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

References

[1] K. Fernandes, P. Vinagre, and P. Cortez, "A proactive intelligent decision support system for predicting the popularity of online news," in *Progress in Artificial Intelligence*, ser. Lecture Notes in Computer Science, F. Pereira, P. Machado, E. Costa, and A. Cardoso, Eds. Springer Intl. Publishing, 2015, vol. 9273, pp. 535–546.

[2] G. Szabo and B. A. Huberman, "Predicting the popularity of online content," *Communications of the ACM*, vol. 53, no. 8, pp. 80–88, 2010.

[3] A. Tatar, J. Leguay, P. Antoniadis, A. Limbourg, M. D. de Amorim, and S. Fdida, "Predicting the popularity of online articles based on user comments," in *Proceedings of the International Conference on Web Intelligence, Mining and Semantics*. ACM, 2011, p. 67.

[4] M. Tsagkias, W. Weerkamp, and M. De Rijke, "Predicting the volume of comments on online news stories," in *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, 2009, pp. 1765–1768.

[5] R. Bandari, S. Asur, and B. A. Huberman, "The pulse of news in social media: Forecasting popularity." in *ICWSM*, 2012, pp. 26–33.

[6] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[7] L. Invernizzi, "js-crawler," https://github.com/invernizzi/js-crawler, 2013.

[8] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2014, pp. 55–60. [Online]. Available: http://www.aclweb.org/anthology/P/P14/P14-5010

[9] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[10] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[11] J. Fan and R. Li, "Variable selection via nonconcave penalized likelihood and its oracle properties," *Journal of the American statistical Association*, vol. 96, no. 456, pp. 1348–1360, 2001.

[12] "Opencsv," http://opencsv.sourceforge.net/, accessed: 2015-09-29.

[13] "Jblas," http://jblas.org/, accessed: 2015-09-29.