Predicting the Number of Shares of a Webpage

Alexandre Piche
260478404

Philippe Nguyen
260482336

Yash Lakhani
260500612

***Abstract*—We implementated a linear regression using the Ordinary Least Square and the Maximum Likelihood methods. Trying to improve our prediction power, we also explored dimension reduction, and regularization methods**

I. Introduction

In the advertising industry, it is of interest to predict the popularity of an article to adequately set the price of the ads that will appear on its page.

II. Datasets

We first used a publicly available dataset available on the UCI Machine Learning repository. We also created our own dataset using articles from the Huffington Post.

A. Mashable Dataset

39,000 Mashable articles were collected from January 7 2013 to January 7 2015 by ???. Note that article with less of 3 weeks of existence were discarded since it is probable that the number of share have not converge.

It contains features such as word count, number of links

A comprehensive overview of the data is available on UCI Machine Learning repository [1].

Achieve the best results on this dataset by using Random Forest. [2]

B. Huffington Post Dataset

III. Implementation of Linear Regression

Ordinary least square is a simple and surprisingly powerful tool for prediction. Y is the target value, X is the covariates and β is the weights of those covariates.

$$Y = X\beta + \epsilon$$

A. Least Squares Estimate

With the traditional assumption of $E(X^T\epsilon) = 0$ [3], i.e. that the error is uncorrelated with the matrix X, it is easy to solve for the weights, the resulting equation is given by

$$\begin{aligned} Y &= X\beta + \epsilon \\ X^TY &= X^TX\beta + X^T\epsilon \\ \hat{\beta} &= (X^TX)^{-1}X^TY \end{aligned}$$

It is possible to weaken the assumption that $E(X^T\epsilon) = 0$, namely to deal with heteroskedasticity. The method is called *Generalized Least Squared* and is well explained in [3]. However due to it's computational complexity this method was not carried in this experiment.

B. Maximum Likelihood Estimate

Assuming that the error $\epsilon \sim \mathcal{N}(0, \sigma^2)$ it is possible to show that the maximum likelihood method yields the same solution than the closed solution.

$$f_t(y_t, \beta, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(y_t - X_t\beta)^2}{2\sigma^2}\right)$$

It can be easily show that by taking the log of the likelihood and summing over n, since the observations are assumed to be independent, gives the following log-likelihood:

$$l(y, \beta, \sigma) = -\frac{n}{2}\log 2\pi - \frac{n}{2}\log\sigma^2 - \frac{1}{2\sigma^2}(y - X\beta)^2$$

Differentiating the log likelihood with respect to σ yields

$$\frac{\partial l(y, \beta, \sigma)}{\partial \sigma} = -\frac{n}{\sigma} + \frac{1}{\sigma^3}(y - X\beta)^T(y - X\beta)$$

Setting the derivative equal to zero yields

$$\sigma^2(\beta) = \frac{1}{n}(y - X\beta)^T(y - X\beta)$$

which minimizes the variance. Substituting in the likelihood function yields

$$l(y, \beta) = -\frac{n}{2}\log 2\pi - \frac{n}{2}\log\left(\frac{1}{n}(y - X\beta)^T(y - X\beta)\right) - \frac{n}{2}$$

Since only the second term contains β and by the properties of the log the problem then boils up to minimize the $SSR(\beta) = (y - X\beta)^T(y - X\beta)$. We can take the derivative with respect to β (gradient).

$$\frac{\partial SSR(\beta)}{\partial \beta} = -2X^T(Y - X\beta)$$

A coverage in depth of this derivation can be found in [3]

To maximize the likelihood (minimize the SSR) we need to implement a gradient descent algorithm.

while $\epsilon > 0.1$ and $i <$ max_iterations **do**
  hypothesis $\leftarrow X^T\beta$
  loss $\leftarrow$ hypothesis - Y
  gradient $\leftarrow$ 2 X^T loss
  $\beta_{\text{new}} \leftarrow \beta - \frac{\alpha * \text{gradient}}{n}$
  $\epsilon \leftarrow \|\beta_{\text{new}} - \beta\|$
  $i \leftarrow i + 1$
  $\beta_{\text{new}} \leftarrow \beta$
end while

C. Model selection

It is sometimes useful to take into consideration the variance of our coefficients to determine whether or not they should be part of our model. Traditional statistics came up with multiple tools to assess the predictive power of a model based on the likelihood and the variance of the coefficients. We will cover some of the more useful in this section.

Only having the size of the weights is of little used if we cannot asses the fit of the model.

1) Significance of our Weights: It is possible that even if the weight is big that the variance is so high that it is not significantly different from zero.

$$\begin{aligned} Var(\hat{\beta}) &= s^2(X^TX)^{-1} \\ s^2 &= \frac{1}{n-k}\sum_{t=1}^{n}\hat{u}_t^2 \end{aligned}$$

[3]

2) Adjusted R^2: To asses the fit of the model, it is helpful to look at the percentage of the sum of square that is explained by our model.

$$\bar{R}^2 = 1 - \frac{\frac{1}{n-k}\sum_{t=1}^{n}\hat{u}_t^2}{\frac{1}{n-1}\sum_{t=1}^{n}(y_t-\bar{y})^2}$$

[3]

3) AIC and BIC: It is possible to select a model based on the information criteria. It is particularly useful to test the difference between two nonnested models.

Adding a variable cannot decrease the fit and almost always increase the prediction power of the model. However it might add noise to the prediction. AIC and BIC are a way to quantify the importance of adding a feature to our model by penalizing complexity.

Akaike information criterion is given by

$$\text{AIC}_i = l_i(\hat{\theta}_i) - k_i$$

Bayesian information criterion

$$\text{BIC}_i = l_i(\hat{\theta}_i) - \frac{1}{2}k_i \log n$$

[3]

IV. Model Complexity and Dimension Reduction

There is a total of 59 features, where some of them are correlated. Given the high number of features, it might be preferable to reduce the dimension to achieve better out of sample prediction. The intuition is that too many features will increase the variance of our weight estimates while too few features will lead our model to be bias.

It is also along the reasoning line of the widely accepted Occam's razor principle. The principle suggests that parsimonous models generalize better than more complex model.

There is multiple way to avoid an over complex model that will generalize well for prediction, we will explore the performance of some of the most known of them.

A. Principal Component Analysis

Given the large dimension of the dataset and that some of the feature are highly correlated we decided to the principal component analysis algorithm to reduce the dimension by using principal component analysis (PCA) algorithm. We noticed that over 95% of the variance can be explained by the first 22 dimensions. The idea behind the PCA algorithm is trying to reconstruct X, by the minimal set of component. Namely we want to find a W such that

L linear basis vector

X is $K \times N$, where K is the number of feature and N the number of examples.

$$J(W,Z) = \|X - WZ^T\|_F^2$$

Frobenius norm.

Where W is $K \times L$ orthonormal and Z is $N \times L$ matrix [4] [5]

B. Ridge or L2-Regularization

$$\hat{\beta}^{\text{ridge}} = \underset{\beta}{\text{argmin}} \left\{ \sum_{i=1}^{N}\left(y_i - \beta_0 \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2 \right\}$$

[6]

The gradient will then be

$$\frac{\partial SSR(\hat{\beta}^{\text{ridge}})}{\partial \hat{\beta}^{\text{ridge}}} = -2X^T(Y - X\beta) + 2\lambda\|\beta\|$$

We can then add the following condition to our gradient descent algorithm

if 'Ridge' is True **then**
    loss $+= 2 * \lambda\|\beta\|$
end if

Ridge regression can also be incorporated fairly easily to OLS

$$\hat{\beta} = (X^TX + \lambda I)^{-1}X^TY$$

Here, λ is a hyper-parameter that we can change and optimize using cross validation. A larger λ gives a greater amount of regularization. Having $\lambda = 0$ means no L2 regularization, giving the standard ordinary least squares solution.

C. Lasso or L1-Regularization

Lasso can be used for features selection by setting some of the coefficients to zero. Note that we should be careful in dropping features because it might increase the bias of our model, however it will reduce it's variance.

$$\hat{\beta}^{\text{lasso}} = \underset{\beta}{\text{argmin}} \left\{ \sum_{i=1}^{N}\left(y_i - \beta_0 \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| \right\}$$

There's no closed form solution for the lasso, since the solution is non-linear (due to the absolute value) [6]. We used the python library "Scikit Learn" [5] to estimate the lasso parameters.

D. Feature Normalization

It is numerically challenging to apply the gradient to feature that are of multiple order different from each other. We can normalize without changing the span of the columns (features), by applying a linear transformation to it.

V. Cross-Validation

To optimize the value of our hyperparameters, we train our model on $k-1$ fold using a particular choice of hyperparameter and then compute a metric of their predictive performance on the k-th fold. We repeated the procedure k times such as every fold has been used for prediction. We then average the error across the k experiments and compare the performance.

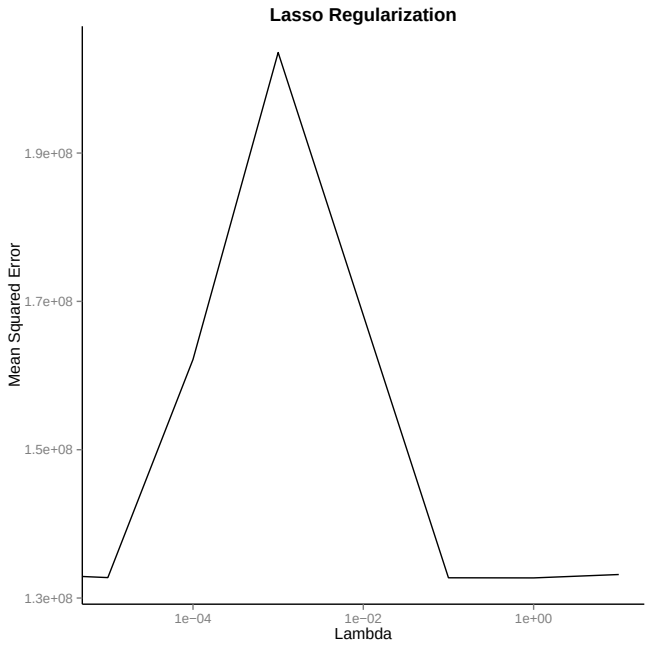
Formally, our cross validation is done by choosing a k, which is the number of cross validation folds. We split the entire data set into k different splits, such that each split has the same number of instances. If the data set is not divided equally by k, the remainder is added to the last split. We shall use $X[i]$ and $Y[i]$ to denote split i of the data set.

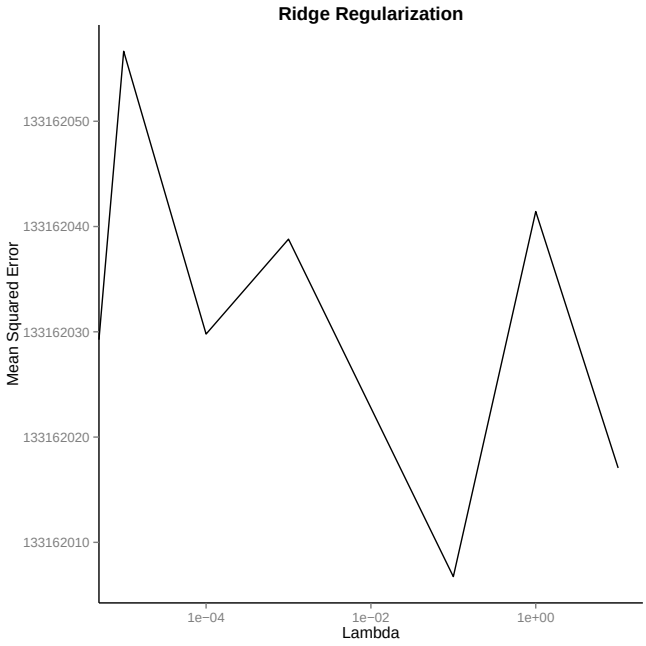
Shuffle the pair $\{X, Y\}$
for fold i **do**
  Set $X[i]$ to be the validation feature matrix
  Set $Y[i]$ to be the validation response vector
  Combine all other $X[-i]$ to be the training feature matrix
  Combine all other $X[-i]$ to be the training response vector
  for every hyper-parameter setting **do**
    Train the linear model with the current training set and given hyper-parameters
    Generate a prediction for Y given the validation feature matrix $X[i]$
    Record the error between the predicted $\hat{Y}$ and $Y[i]$
  end for
end for
Average the error for each hyper-parameter setting across every fold
Choose the hyper-parameter setting with the lowest average error

VI. Results

Also talk about the α parameter for the gradient descent.

Talk about the mean squared error (MSE) obtain when we varied the alpha of the lasso





Algorithm	Mean Squared Error
OLS	132736810
Lasso	**132712496**
Ridge	133162007
Principal component analysis	133592746

Lasso is the method that give us the best performance.

It is probably due to the fact that it is shrinking the coefficients of some features to zero which was bringing noise to the model instead of increasing it's predictive power.

Since it is the only method that we did not implemented, this might be explained by the fact that the scikit learn library use a better optimization algorithm than the simple gradient descent than we used.

Given the small difference it might be worth to take the difference in results with a pinch of salt, since it might not be significant.

VII. Conclusion

Given our large error linear regression might not be an appropriate tool to model the popularity of an article. It is possible that the features are not related in any linear fashion to the number of shares that an article will get.

References

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

[2] 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*. Springer, 2015, pp. 535–546.

[3] R. Davidson and J. G. MacKinnon, *Econometric theory and methods*. Oxford University Press New York, 2004, vol. 5.

[4] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[6] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin, "The elements of statistical learning: data mining, inference and prediction," *The Mathematical Intelligencer*, vol. 27, no. 2, pp. 83–85, 2005.