Predicting the number of shares of a social network with linear regression

Gustavo Ciotto Pinton, RA117136*

Abstract

Relying on a Mashable social network's real database [1], this report uses linear regression techniques to predict the number of shares that an eventual publication would have based on its describing features. Improving tools such as regularization and variable selection are explored, and the settings of their important parameters are discussed as well. The proposed linear regression was obtained by two different methods, gradient descent and least squares, whose results are compared in the last sections of this document.

1. Introduction

Linear regression consists of a simple technique whose objective is to relate an input data set to a output data set through the use of a linear equation $Y = \theta_0 + \theta_1 X_1 + \theta_2 X_2 + \ldots + \theta_N X_N$, in which Y, X_i and θ_i represent, respectively, the output generated by the model from a set of describing attributes and the coefficients of the equation. The previous equation is used only for a singular sample, but, as seen in class, one of the main aspect that guarantees the quality of results obtained by machine learning methods is the large volume of data. Having that in mind, a set of M input samples described by N attributes is represented by the matrix $\boldsymbol{x}_{N+1 \times M}$. In the same way, the coefficients of the equation are represented by the vector $\boldsymbol{\theta}$ of $N + 1$ elements and the output by the vector $\boldsymbol{y}$ of size M, in order to obtain the vector equation 1, just below.

$$\boldsymbol{y} = \boldsymbol{\theta}^T \boldsymbol{x} \tag{1}$$

The determination of each component of $\boldsymbol{\theta}$ is performed through the minimization of the cost function $J(\boldsymbol{\theta}, \boldsymbol{x})$, described by the equation 2, in which $\boldsymbol{x}_i$ corresponds to the i-th sample and t_i to the *target* that we want to achieve.

$$J(\boldsymbol{\theta}, \boldsymbol{x}) = \frac{1}{2M} \sum_{i=1}^{M} \left(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)} - \boldsymbol{t}^{(i)} \right)^2 \tag{2}$$

The datasets used during the minimization process, $\boldsymbol{x}$ and $\boldsymbol{t}^{(i)}$, belong to a set called **training set**, meanwhile the parameters and model validation occurs in a set that receives this same name. A general rule is to use at least 80% of the available data for training and the rest, 20%, for testing and validation.

The cost function $J(\boldsymbol{\theta}, \boldsymbol{x})$ has the important property of being **convex** [2]. This fact ensures that it has no local minima, except for the global one. Thus, two minimization methods can be used. The first one, called **normal equations**, allows to obtain the optimal $\boldsymbol{\theta}$ from a single expression, represented in 3. It is quickly verified that the negative point of this approach is the calculation of the inverse matrix, whose asymptotic computing complexity is $\Theta(n^3)$, as seen in class. For the matrix $\boldsymbol{x}^T \boldsymbol{x}$ of dimension $N + 1 \times N + 1$, the number of attributes N may represent an impediment to the use of this technique.

$$\boldsymbol{\theta} = \left(\boldsymbol{x}^T \boldsymbol{x} \right)^{-1} \boldsymbol{x}^T \boldsymbol{t} \tag{3}$$

The second method, called **gradient descent**, is iterative and does not require the computing of any inverse matrix. By partially deriving the equation 2 in relation to $\boldsymbol{\theta}_j$, it is possible to obtain the cost function slope and therefore the direction we must take to approximate the model to the global minimum. By repeating this reasoning for all parameters $\boldsymbol{\theta}_j$, we obtain the operation 4 which must be applied to each of them at each iteration. It is emphasized that $\boldsymbol{x}_1 = 1$ for all i.

$$\boldsymbol{\theta}_j = \boldsymbol{\theta}_j - \alpha \frac{\partial J}{\partial \boldsymbol{\theta}_j} = \boldsymbol{\theta}_j - \frac{\alpha}{M} \sum_{i=1}^{M} \left(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)} - \boldsymbol{t}^{(i)} \right) \boldsymbol{x}_j^{(i)} \tag{4}$$

Equation 4 introduces the parameter α, called *learning rate*. This parameter controls the convergence speed to the global minimum, i.e, the higher its value, the faster it is reached. On the other hand, very large values will produce an effect called *overshooting*, that is, the found solution will always *swing* around the minimum without ever reaching it. In the following sections, the author explains the strategy he used to define this parameter for this problem.

Having seen all the theory behind linear regression, we can now apply it to the real datasets. In this report, it is used a real database [1] got from the *Mashable* social network, which relates a series of attributes of a given publication to the number of shares it has received. In other words, it is sought to find a linear relationship between the number of shares received with the attributes of a publication. In total, 31715 input samples were used to train and validate the models and 7929 for testing.

2. Activities

The next subsections explain the choices of the various parameters adopted by the author.

---

*Is with the Institute of Computing, University of Campinas (Unicamp). **Contact**: gustavociotto@gmail.com

2.1. Learning rate setting strategy

As mentioned in the 1 section, the learning rate parameter controls the speed of convergence to the cost function's global minimum. Considering the issues discussed in this same section, an initial α value of 0.5 is adopted, modifying it in only two situations:

- When the error at iteration k is greater than the one of the iteration $k - 1$. In this case, it is concluded that the solution found was worse and, therefore, we must decrease the speed of convergence to avoid the solution to keep *swinging* around the minimum.

- The error variation between iterations k and $k - 1$, i.e., $\frac{R_{k-1}-R_k}{R_k}$, was less than a percentage $\Delta = 0.5\%$. In this case, it is desired to eliminate the effects of *overshooting* and to bring the solution as close as possible to the global minimum. The root mean square error obtained at iteration k is denoted by R_k and given by the formula 5:

$$R_k = \sqrt{\frac{\sum_{i=1}^{M}\left(\boldsymbol{\theta}_k^T \boldsymbol{x}^{(i)} - \boldsymbol{t}^{(i)}\right)^2}{M}} \qquad (5)$$

In these two situations, the value of α is divided by half.

2.2. Adjustment of the discrete variables

The attributes description of each of the entries revealed the presence of some discrete variables. Two groups were identified: one with variables of type *publication occurred on Monday?*, *publication occurred on Tuesday?* and so on, and one with attributes similar to *is an entertainment channel?*. Before using them or not in the models proposed below, it is necessary to verify if such variables are **orthogonal** to each other, that is, if the distance between them considered two by two is always the same. Taking into account that the variables of both groups can never be set simultaneously (a publication can not occur on Monday and Tuesday and it can not belong to two channels at the same time, for example), we can verify the property of **orthogonality** and therefore we can use them without any modifications.

2.3. Input and output data sets normalization

The normalization of the dataset before training prevents the values $\boldsymbol{\theta}_j$ being extremely small or large, causing instabilities or interferences in the model's convergence speed. Among the many ways of normalization that can be applied, we choose for this activity the one of equation 6. In this case, the mean μ_j and the standard deviation σ_j of each feature are calculated over all training samples and then the expression 6 is applied.

$$\boldsymbol{x}'_j = \frac{\boldsymbol{x}_j - \mu_j}{\sigma_j} \qquad (6)$$

Note that the values μ_j and σ_j must be saved for later use in the test set or any other input to be used with the model.

2.4. Regularization

Regularization is a technique that allows to limit the negative effects of *overfitting* without removing any describing feature from the model. It consists of applying a penalty factor over each parameter $\boldsymbol{\theta}_j$. Equation 2 then becomes:

$$J(\boldsymbol{\theta}, \boldsymbol{x}) = \frac{1}{2M}\left[\sum_{i=1}^{M}\left(\boldsymbol{\theta}^T\boldsymbol{x}^{(i)} - \boldsymbol{t}^{(i)}\right)^2 + \lambda\sum_{j=2}^{N+1}\boldsymbol{\theta}_j^2\right] \qquad (7)$$

If λ is very large, then the minimization process will produce a small-norm vector $\boldsymbol{\theta}_j$, resulting in a biased model that would not succeed in fitting to the training data. In contrast, small λ could not counteract the effects of *overfitting* and the variation created by the test dataset would still be large.

Equation 6, in turn, becomes expression 8, where $\boldsymbol{I}_{N\times N}$ is the identity matrix:

$$\boldsymbol{\theta} = \left(\boldsymbol{x}^T\boldsymbol{x} + \lambda\begin{bmatrix}0 & 0 \\ 0 & \boldsymbol{I}_{N\times N}\end{bmatrix}\right)^{-1}\boldsymbol{x}^T\boldsymbol{t} \qquad (8)$$

Note that $\boldsymbol{\theta}_1$, the bias appended into the model, is not penalized.

3. Proposed solutions

This section is dedicated to discussing the proposed solutions. We performed tests with simple, regularized and containing second and third order variables models. For the gradient descent technique, the input dataset was randomly divided into two other sets, training (70%) and validation (30%). For the normal equations, no further separation was applied to the original training dataset.

3.1. Simple linear model

In this model, the linear regression method is used without any regularization factor and second or third order terms. For the gradient descent technique, the formula 4 is computed in a loop that iterates 100 times in total. Figure 1 represents the root mean square errors according to the equation 5 obtained for the training, validation and test datasets at each iteration. It is observed that both errors stabilize around 11357 and 14638 respectively and that there were no negative effects of overfitting. Curiously, the obtained validation error curve is below the curve corresponding to the training dataset during almost all the simulation.

Figure 3.1, in turn, represents the results found through the use of the expression 6. Two plots are shown: in the first one, we compare the errors obtained in the train and test datasets for many input set sizes ranging from 10000 to 31715, meanwhile in the second, we compare the parameter vectors $\boldsymbol{\theta}$ obtained by the two methods described in this section. As expected, the RMS found by the normal equation, around 10580, was lower than the one found by the *gradient descent* technique. The test error, in turn, also remained around 14570. It is also noted that
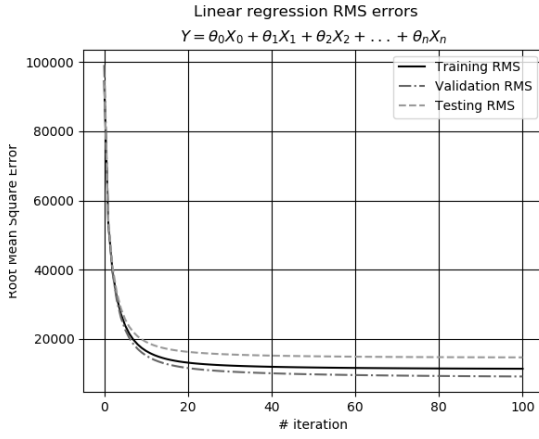
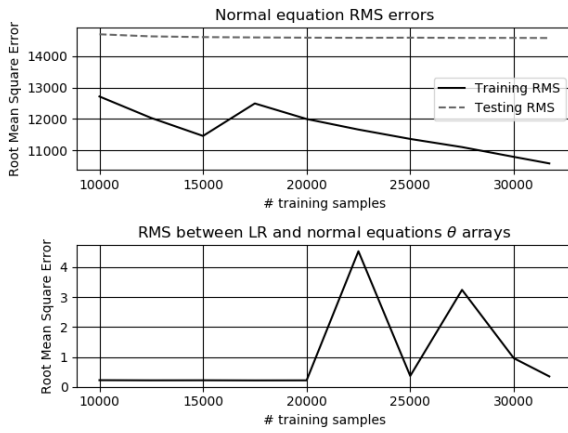Figure 1. RMS obtained for the training and test dataset at each iteration.

Figure 2. RMS obtained for many training dataset sizes.

Figure 3. RMS obtained for the regularized models.
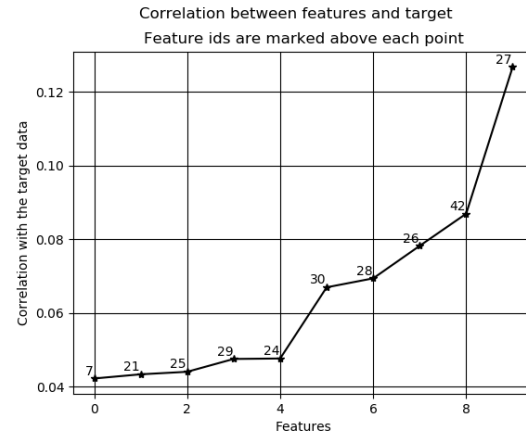
Figure 4. Correlation of the input features with the output.

there are some considerable differences between the vectors θ computed by each method, especially for intermediate sized data sets.

3.2. Regularized simple linear model

Although we have not verified the *overfitting* presence in the previous results, we propose the use of regularization, as explained in subsection 2.4. Two values of λ were tested, one equal to 1.0 and the other, twenty times greater, 20.0. Figure 3 compares the results found by the *gradient descent* technique between the regularized model and the one from the previous subsection. We verify that the regularization did not substantively modify the error found for both the training and the test sets. This indicates that the suggested values of λ are not big enough when compared to each of the factors θ_j^2 individually to produce changes on the model. These same models were also submitted to the normal equations, however, as in the *gradient descent*, no considerable changes were noted.
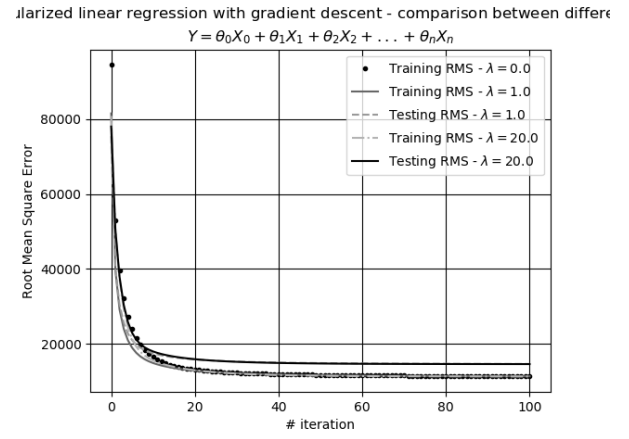
3.3. Reduction of features

Another attempt to minimize the error consisted of reducing the number of attributes used in the model. The attributes selection was done by computing the correlation of each of them to the output. Considering that the model is still linear, it is expected that more correlated variables follow the same pattern as the one observed at the output. The figure 4 represents only the 10 most correlated variables that were used for the proposed new model. It is worth mentioning that the highest correlation, corresponding to attribute 27 (*Avg. Keyword (avg. Shares)*), is still a very low value, i.e, 0.125, indicating that no variable strongly influences the number of shares received by a publication. Figure 5 compares the results found by the simple linear model and the model proposed in this subsection. A very slight increase in the testing error for the reduced model is observed at the iteration 10, indicating a slight overfitting presence. In addition, it is verified that both training and test errors of the reduced model converge faster than those computed by the simple model.
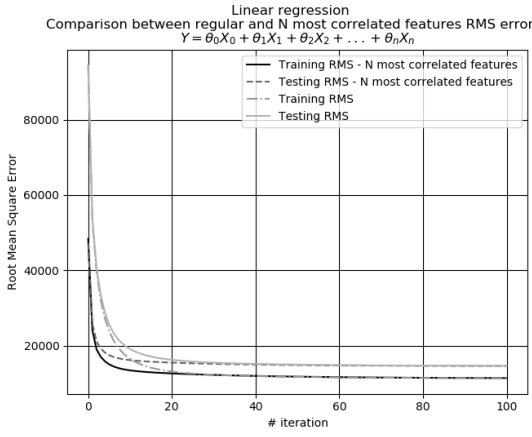
Figure 5. Comparison between simple linear model and the one with the 10 most correlated variables with the output.
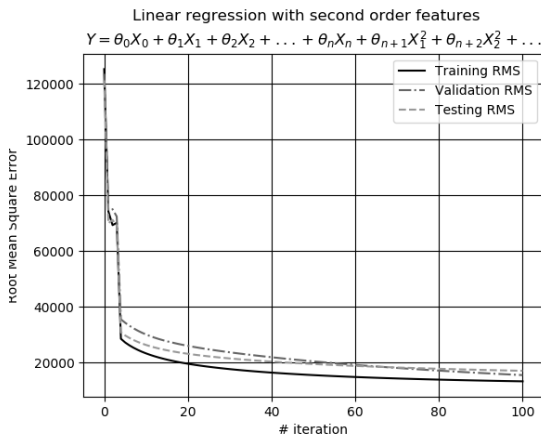

Figure 6. RMS errors obtained at each iteration by the model with second order attributes.

3.4. Linear model with second order attributes

Another technique that can be used to reduce training and test errors is the addition of new attributes based on the existing ones to the simple model. In this case, we append to the traning and test input datasets all existing variables to the power of 2, obtaining the new relation $Y_2 = Y + \theta_{N+1}X_1^2 + \theta_{N+2}X_2^2 + \ldots + \theta_{2N}X_N^2$. This new model is expected to be better adapted to the training and test data simultaneously, i.e, without overfitting. Note that the equations 3 and 4 do not need to be changed, given that the cost equation 2 remains linear in θ. Figure 6 has the result of the *gradient descent* technique for this new model. Note that at iteration 5, the value of α is modified, as described in section 2.1, since the error produced was higher than that found in the previous iteration. The training and test RMS errors after 100 iterations were, respectively, 13317 and 17089, higher than those verified for the simple linear model. The use of the 3 equation, in turn, resulted in errors around 10540 and 14580, very similar to those obtained by the same technique applied to the simple linear model.
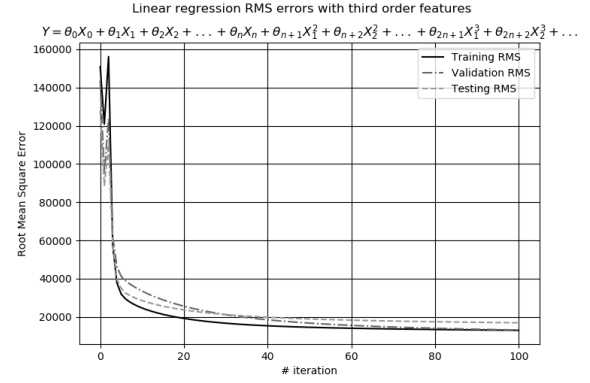

Figure 7. RMS errors obtained at each iteration by the model with second and third order attributes.

3.5. Linear model with third order attributes

The last model tested in this report used second and third order attributes, obtaining the expression $Y_3 = Y_2 + \theta_{2N+1}X_1^3 + \theta_{2N+2}X_2^3 + \ldots + \theta_{3N}X_N^3$. The figure 7 has the evolution of the errors found during the 100 iterations. As occurred in figure 6, the value of the parameter α - the learning rate - had to be decreased considering that the training error of iteration 3 was higher than the one of the iteration 2. The smallest RMS errors found for training and testing were, respectively, 12990 and 16934, higher than those obtained in the simple linear model. Finally, the use of the normal equations resulted in errors equal to 10458 and 15084. The first was slightly lower than the same of the simple model, while the second, bigger. This fact illustrates one of the main difficulties encountered when we increase the complexity of the model: *overfitting*. In this case, the error of the training set is reduced, but that one of test set is increased, indicating that the model has adapted in an exaggerated way to the training data. As already discussed, one of the possible solutions to this problem is regularization.

4. Conclusions

Five linear models were proposed to predict the number of shares of a given Mashable social network publication. In almost all of them, the train and test RMS errors were around 10500 and 14500, respectively, indicating that the linear model is not able to accurately model the behavior of the users of this social network.

References

[1] Kelwin Fernandes, Pedro Vinagre, and Paulo Cortez. A proactive intelligent decision support system for predicting the popularity of online news. In Francisco Pereira, Penousal Machado, Ernesto Costa, and Amílcar Cardoso, editors, *Progress in Artificial Intelligence*, pages 535–546, Cham, 2015. Springer International Publishing. 1

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. 1