Predicting Wine Quality Using Linear Regression

Bibek Karki
Department of Computer Science,
bkarki3@huskers.unl.edu

Biquan Zhao
School of Natural Resources, University of Nebraska - Lincoln
bzhao10@huskers.unl.edu

Tyler Zinsmaster
Department of Computer Engineering, University of Nebraska
tzinsmaster@huskers.unl.edu

***Abstract*—We implemented and evaluated the Linear Regression for the red-wine dataset to predict the quality of a red wine, given its physicochemical properties. Exploratory Data Analysis and extensive feature engineering methods are used to produce best features to feed into our model. Moreover, Model selection is used to find the best parameter for Linear Regression model, we have engineered. We have used Batch Gradient Descent and Stochastic Gradient Descent to find the optimal weights for our Model. We studied over-fitting and under-fitting by implementing polynomial regression for various degrees.**

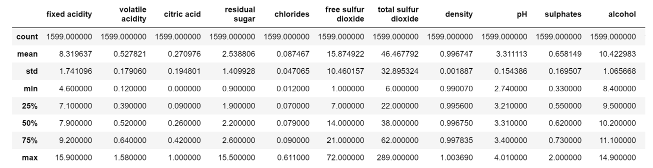
I. Introduction

We are predicting the quality of red-wine based on its physicochemical properties. The target variable of our Linear Regression model is quality of wine that ranges from zero to ten. We are also implementing Polynomial Regression by feeding polynomial features to our linear model to learn polynomial decision boundary. We are using iterative method(Batch Gradient Descent and Stochastic Gradient descent) instead of closed form solution. The number of features explodes when we increase the degree. We are using iterative method due to computational complexity. For Closed form solution, we need to compute inverse of a matrix of size (d + 1) x (d + 1) matrix, where d is number of features. The computational complexity of computing the inverse is: $\mathcal{O}(d^3)$. Our data set is large. We need to store the whole data set in our memory which is impossible with our hardware. Also, there are numerical complexities that results in instability and inconsistency. These are all situation when inverse of a matrix exits. For some cases, matrix becomes singular. Pseudo inverse is not the best solution.

II. Data Summary

A. Description Of Data Set

The data set is red-wine quality pulled from UCI repository for Machine Learning. It contains 12 features and 1599 samples.The target variable is quality, renamed as y. Data set contains m(1599) examples, one example (x^i, y^i) per row. In particular, the i-th row contains columns $x^i \in \mathbb{R}^{11}$ and $y^{(i)} \in [0, 10]$ The size of training set is $\mathbb{R}^{1599\times 12}$.The eleven features are physicochemical tests. These tests value is used to determine the quality of a wine.



	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747	3.311113	0.658149	10.422983
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	0.154386	0.169507	1.065668
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.400000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.550000	9.500000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	0.620000	10.200000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	0.730000	11.100000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2.000000	14.900000

Fig. 1. Summary of each of the variables in the dataset in terms of mean, standard deviation, and quartiles

B. Descriptive Statistics

C. Feature Selection and Scaling

The data is standardized by subtracting its mean and dividing by standard deviation. Data has zero mean and unit variance after standardization. This will avoid over representation of some of the features. Standardization makes all features to contribute equally to the dissimilarity measures. It always provide better result for classification.

We notice there are redundant features. Wine is acidic, So fixed acidity and ph provides similar information. Also, we could see pair plot to notice that these features have non zero Co-variance. Similarly for Sulphates, Total sulphur dioxide and free sulphur dioxide as all are just sulphates. Moreover, we notice same phenomenon for residual sugar and density. those features are dependent as adding sugar increases density.SO, more residual sugar means more density.

Using intution and pair plot we found some of redundant features. We then used Feature Selection using Backward search to find the optimal features. This is possible since features and data set is small enough. In backward search, we keep removing feature until and unless our accuracy stops increasing in all possible subsets of features. Complexity of backward search is O(n^2). There are 2^n possible Linear Regression models to compare for feature selection. Our optimal features is 'volatile acidity', 'citric acid', 'residual sugar', 'pH', 'sulphates' and 'alcohol'.

III. Methods

A. Polynomial Regression with Batch Gradient Descent

For, Linear Regression, we firstly approximate y as linear function of x. For the bias weight(w_0) we write it as w_0 x_0, where x_0 = 1 such that we could write linear relationship between y and x as:

$$y(x) = \sum_{i=0}^{d} w_i x_i = \vec{w}^T \vec{x}$$

Fig. 2. Linear Relationship of x and y

Given a basic training set, we can fit a linear function onto it using the weights(θ). This linear function is used to calculate y for new values of x. Linear Regression finds optimal parameters by comparing the error between the predicted and actual outcomes of the model, which can be referred to as Mean Squared Error (MSE). In the closed-form Linear Regression, this is done via the "normal equation", calculating based on all of the training set at once, and in this case, the loss/error is a function called L(w), the "cost function", as it tells how much the algorithm has to pay for incorrect predictions. But this is a limited approach to Linear Regression, and for our dataset, we utilized a more complex approach. To find these weights, or rather, optimal parameters, we use an approach to Linear Regression known as the Gradient Descent. This is a method of Linear Regression which gradually tweaks parameters to minimize the "cost" function over our training set. (The cost being directly based on the MSE). To implement Gradient Descent for Linear Regression, we call the cost function J(θ), and θ is our new weight parameter. We calculated the MSE by dividing the sum of squared error by the size of the training set, and divided that by 2 for mathematical convenience.

$$\boxed{J(\vec{\theta}) = \frac{1}{2m}\sum_{i=1}^{m}\left(y_i - \vec{\theta}^T\vec{x}_i\right)^2}$$

Fig. 3. Cost function

Gradient Descent starts with a "search algorithm", with an initial guess for θ. Derivative of cost function with respect to θ gives the direction to the steepest descent. We then control the magnitude of the descent using η, so as to descent to the minimum of out cost space. To begin, θ is filled with zeros or random values, in a process called random initialization. This θ is then updated repeatedly with the update function:

$$\boxed{\theta_j := \theta_j - \eta \frac{\partial}{\partial \theta_j} J(\theta)}$$

Fig. 4. Update rule for Gradient Descent

For MSE cost function, we compute the gradient for all θ. So now the update rule becomes(Fig. 5):

$$\boxed{\vec{\theta} := \vec{\theta} - \frac{\eta}{m}\vec{X}^T.(\vec{X}.\vec{\theta} - \vec{y})}$$

Fig. 5. Update rule for Gradient Descent with MSE cost

In order to penalize the flexibility of our model that may result in over-fitting, we add magnitude of the weights controlled by hyper parameter λ in our cost function. Otherwise, our model would always pick a high degree polynomial resulting in great performance in train set and suffers heavily in test set.

So our cost function would become(Fig.6):

$$\boxed{J(\vec{\theta}) = \frac{1}{2m}\sum_{i=1}^{n}\left(y_i - \vec{\theta}^T\vec{x}_i\right)^2 + \frac{\lambda}{2m}\|\vec{\theta}\|_2^2}$$

Fig. 6. l2 Regularized Cost function

And, our update rule is(Fig.7):

$$\boxed{\theta := \theta - \frac{\eta}{m}X^T.(X\theta - Y) - \frac{\eta\lambda\theta}{m}}$$

Fig. 7. l2 Regularized Update rule

If we use l1 norm for weights instead of l2 norm. Our regularized cost and update rule is(Fig.8):

$$\boxed{J(\vec{\theta}) = \frac{1}{2m}\sum_{i=1}^{n}\left(y_i - \vec{\theta}^T\vec{x}_i\right)^2 + \frac{\lambda}{2m}\|\vec{\theta}\|}$$

$$\boxed{\theta := \theta - \frac{\eta}{m}X^T.(X\theta - Y) - \frac{\eta\lambda}{m}sign(\theta)}$$

Fig. 8. l1 Regularized Update rule

B. Linear Regression Arguments

The major arguments for Linear Regression are the number of features and the degree. Given the number of the features and the degree of polynomial, the Linear Regression above is modified and presented as:

$$\boxed{\hat{y} = w^T\phi(x)}$$

Fig. 9. Polynomial Regression

For example, considering the number of features is three, and the degree of polynomial is two, we will get the arguments following(Fig.10):

C. Model Selection and Complexity

We determined the best model based on the overall performance(lowest average error) to find the hyper parameters lambd, learning rate and regularizer. We used MSE to compute the error of our model.

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\phi(\vec{x}) = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_1^2 \\ x_2^2 \\ x_3^2 \\ x_1x_2 \\ x_2x_3 \\ x_3x_1 \end{bmatrix}$$

Fig. 10. Polynomial Features

IV. Results

A. Cross-validation

We randomly split $S_{dataset}$ into S_{train} (say, 80% of the data) and S_test (the remaining 20%). Here, S_{test} is called the hold-out cross validation set.We then, train each model $M_i \in$ M(set of models) on S_{train} only, to get some score.We then Select the model with greatest score on the hold out cross validation set.

B. S-fold cross-validation

We randomly split $S_{dataset}$ into k disjoint subsets of m/k training examples each. Lets call these subsets S_1, . . . ,S_k. For each model M_i, we evaluate it as follows: For j = 1, . . . , k Train the model M_i on $S_1 \cup \cdot\cdot\cdot S_{k-1} \cup S_k$ to get some model.Test the model on S_j, to get a score. The estimated generalization error of model $M_i$is then calculated as the average of the scores of the model(averaged over j).We then pick the model M_i with the greatest average score, and retrain that model on the entire training set S. It is described visually on fig7 below.

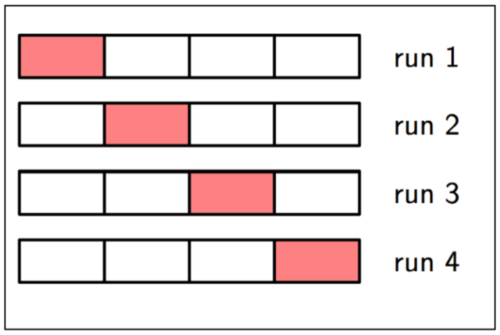


Fig. 11. S-fold cross-validation

C. Mean Squared Error(MSE)

The mean squared error between the true label(y) and predicted label($\hat{y}$) by Linear Regression model is:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Fig. 12. MSE

D. Root Mean Squared Error(RMSE)

The root mean squared error between the true label(y) and predicted label($\hat{y}$) by Linear Regression model is:

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

Fig. 13. RMSE

E. Learning Curves

We have plotted the root mean squared error(RMSE)against training set size for both Degree 1 and 3 polynomial regression.

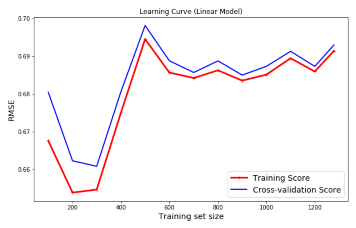


Fig. 14. Degree 1 Learning Curve

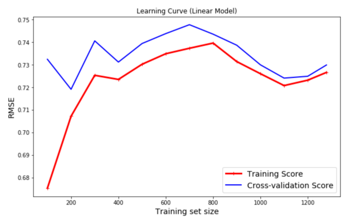


Fig. 15. Degress 3 Learning Curve

We could see that when the training size is low, our model fails to generalize. So from Learning curves we could observe for both cases our train RMSE score is very low while test RMSE score is large. As we increase the train size, the degree 1 polynomial Regression improves its generalization error which is evident from the learning curve. The RMSE for both train and test score gets almost similar. This shows we are doing good in train size and test size. So, generalization error is low. We could minimize the score if we use a more complex model.Hence, the model is under fitting. While in the case of Degree 3 polynomial Regression, even when we increase the train size the gap between train and test RMSE score is significant. We are doing great in training but fail to generalize. Hence our generalization error is large. We are over fitting the data.

F. Polynomial Complexity

To observe the RMSE score as we increase the polynomial degree of polynomial regression, We plot the degree against the RMSE.

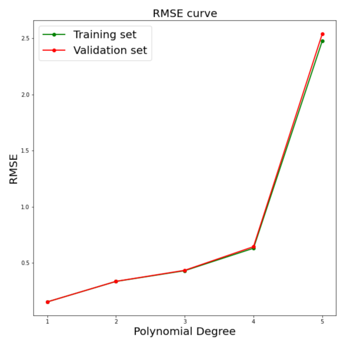


Fig. 16. Polynomial Complexity

G. Weights Stochastic vs Batch Gradient Descent

As we change the learning rate, we experience little effect in weights of our model. So, weights that minimizes the cost function seem to be almost similar. i.e in batch we descent to the minimum local minimum each time. In stochastic gradient descent we get close to minimum but doesnot fully converge to theta that minimizes the cost function. So changing learning rate changes the theta. When learning rate is large when we get close to the minimum theta at each training example we update the theta that slight differs(keeps rounding around the minimum point). If we reduce the learning it almost converge to the minimum(i.e theta almost same as batch) since there is less space to wander around at each training example. SGD can also be useful to avoid local minimums to get global.

H. Figures and Tables

Table I shows the MSE for train and test error for degree 1 using polynomial regression.

TABLE I
TEST SET PERFORMANCE

Polynomial Regression Gradient Descent	**MSE**		
	train	***test***	
Batch	0.50	0.58	
Stochastic	0.46	0.51	

ACKNOWLEDGMENT

Acknowledgment is given to the UNL CSCE478 professor, Dr. M. R. Hasan, and the class GA, Atharva Tendle, for their contributions in our learning of the subject matter, and in organizing the assignment and course materials.

REFERENCES

[1] Bibek Karki, Biquan Zhao, Tyler Zinsmaster, GitHub repository for the assignment, https://github.com/CSCE478-ML/Assignment2

[2] Dr. M. R. Hasan slides regarding Linear Regressioncan be found at $https : //canvas.unl.edu/courses/97606/pages/lecture - slides - and - reading - guide?module_item_id = 2130394$.

[3] Dr. M. R. Hasan github repo on Linear Regression $https : //github.com/rhasanbd/Linear - Regression - Extensive - Adventure$