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.**

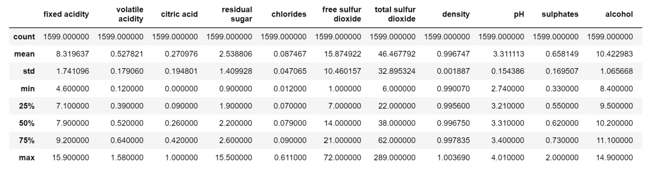
I. Introduction

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 4898 samples.The target variable is quality, renamed as y. Data set contains m(4898) 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 $\mathbb{R}^{4898 \times 12}$.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 any $x_i^{(i)} \in \mathbb{R}^{11}$, we calculate l1 or l2 norm with each $x_i^{(i)} \in (x, y)$, the train set. We store the value in a vector d. Then arg $\min\limits_{x}$ (d). We find k minimum arguments. We then use a threshold for conditional probability to assign class 'c' to the test point. The Probability is given by fig.2.

$$\boxed{p(y=c|\mathbf{x},\mathcal{D},K)=\frac{1}{K}\sum_{i\in N_K(\mathbf{x},\mathcal{D})}\mathbb{I}(y_i=c)}$$

Fig. 2. Conditional Probability for y

B. *KNN Arguments*

We use Minkowski distance metric or L_p norm to measure the dissimilarity between two vectors. Mathematically L_p norm is given by,

$$\boxed{L_p(\vec{x},\vec{z}) := d(\vec{x},\vec{z}) = \left[\sum_{i=1}^{d}|x_i - z_i|^p\right]^{1/p}}$$

Fig. 3. L_p norm

We use inverse distance for weighted KNN. We compute the inverse of 'd'. We then sum all positive and negative classes of k neighbors and compare them to assign class for the particular test point. Using inverse distance our performace increased significantly. The classes nearer to test point are assigned more weight. Instead having to tweak the threshold for finding class probability that the test point lies, we can decay weight as distance increase to get better result.Inverse distance kernel is given by:

$$k(\vec{x},\vec{x_n}) = \frac{1}{d(\vec{x},\vec{x_n})+\epsilon}$$

C. *Model Selection and Complexity*

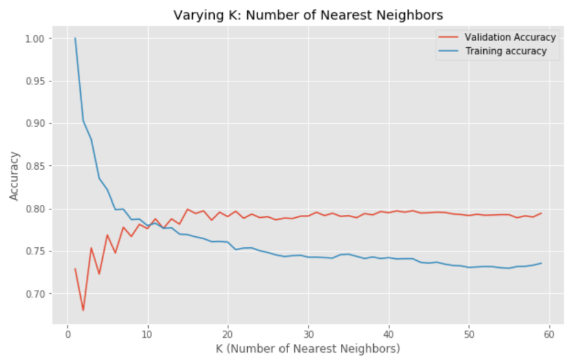
First, we tried to vary number of neighbors denoted by 'k' to find the best KNN model. When we use k equals 1 with l1 norm and uniform weights, our model seems to over fit. This produces great result in trainset but costly for testset as it fails to generalize. While, when k is large underfit and doesnot seem to produce any good result. We find that under this setting optimal value of k is 15.

Also, we perform same test using all the arguments of KNN and found that optimal value of k is 11.

IV. Results

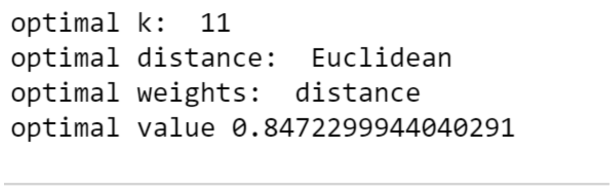
A. *Cross-validation*

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.



```
Optimal K:  15
```

Fig. 4. Varing value of Neighbors



```
optimal k:  11
optimal distance:  Euclidean
optimal weights:  distance
optimal value 0.8472299944040291
```

Fig. 5. Optimal Arguments for KNN

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 $S_1 \cup \cdots S_{k-1} \cup S_k$ 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.

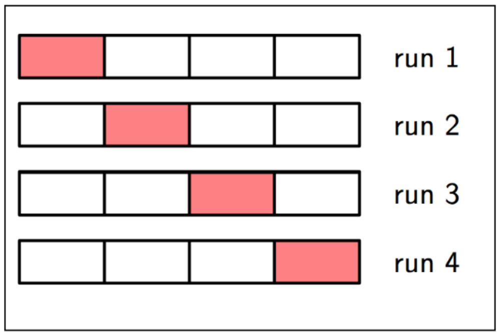


Fig. 6. S-fold cross-validation

C. *Accuracy and Generalization error*

We compare the true target y with the prediction('p') to find the accuracy. Accuracy simply computes the number of correct prediction divided by total prediction.

$$Accuracy = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = p^{(i)}\}}{\sum_{i=1}^{m} 1}$$

Generalization Error is simply given by,

Generalization Error = 1 - Accuracy

Since the data set might be skewed. Accuracy and Generalization error will give us incorrect presentation of out accuracy. To better understand what were being misclassified, we use confusion matrix.

D. Confusion Matrix

Confusion matrix overcomes the issue of Accuracy. It shows the actual and prediction target for both positive and negative targets. Since we are performing binary classification, confusion matrix can be represented as:

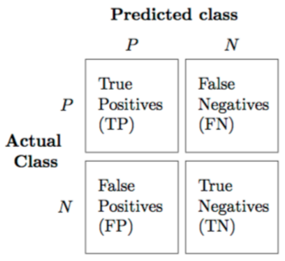
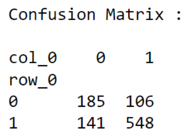


Fig. 7. Confusion matrix for Binary Classification

For our model with optimal parameters on Testset we get:



```
Confusion Matrix :

col_0    0    1
row_0
0      185  106
1      141  548
```

Fig. 8. Confusion matrix for wine quality classification

E. Precision, Recall and f1 score

Precision computes the accuracy of positive labels.

$$\boxed{\text{precision} = \frac{TP}{TP + FP}}$$

Fig. 9. Precision

Recall is the ratio of positive instances that are correctly detected by the classifier.

The F1 score is the harmonic mean of precision and recall.

The precision-recall plot was used to find the optimal threshold to increase f1 score. By observing the curve, 0.6 threshold is optimal for our model.

$$\text{recall} = \frac{TP}{TP + FN}$$

Fig. 10. Recall

$$\boxed{F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{TP}{TP + \frac{FN + FP}{2}}}$$

Fig. 11. F1 score

Another good metric to evaluate our classification and find optimal threshold is ROC curve.

Using the ROC curve we found the area under curve AUC to be 0.794195 where 1 represents perfect classifier and 0.5 represents totally random classifier.

F. Figures and Tables

Test performance metrics are roported in the Table I and Table II. Here k represents number of neighbor and p represents norms

TABLE I
TEST SET PERFORMANCE

KNN Parameters	**Metrics**		
	Accuracy	***F1 Score***	
k=5, p=2, uniform weight	0.7449	0.8117	
(Standardized) k=5, p=2, uniform weight	0.7548	0.8156	
(Optimal)k=11, p=2, inverse distance weight	0.8296	0.8745	
k=11, p=2, uniform weight	0.7673	0.8321	

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.

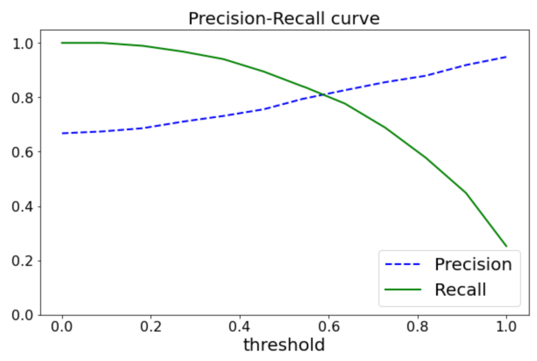


Fig. 12. Precision-Recall curve

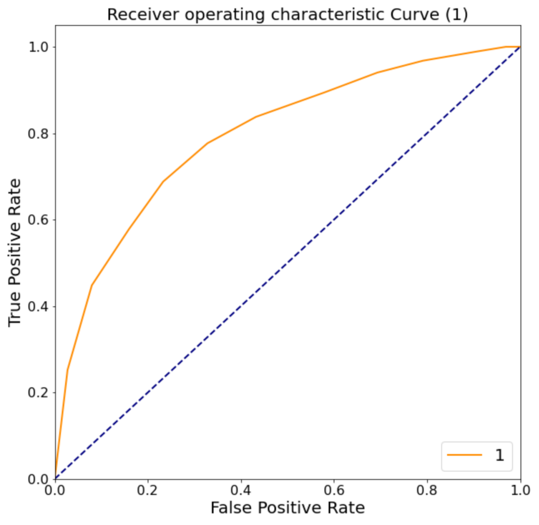


Fig. 13. ROC curve

TABLE II
PERFORMANCE WITH DIFFERENT METRICS

KNN	***Precision***	***Recall***	***F1***	***Accuracy***	***Gen Error***
Optimal	0.8595	0.7992	0.8252	0.7592	0.2408

REFERENCES

[1] Bibek Karki, Biquan Zhao, Tyler Zinsmaster, GitHub repository for the assignment, https://github.com/CSCE478-ML/Assignment1

[2] Andrew Ng, "CS229 Lecture notes", https://see.stanford.edu/materials/aimlcs229/cs229-notes5.pdf

[3] Dr. M. R. Hasan slides regarding KNN can be found at $https://canvas.unl.edu/courses/97606/pages/lecture-slides-and-readingguide?module_item_id=2130394$.

[4] Dr. M. R. Hasan github repo on KNN $https://github.com/rhasanbd/K-Nearest-Neighbors-Learning-WithoutLearning$