University of Nebraska Lincoln

Introduction to Machine Learning
CSCE 478/878

Programming Assignment 2

Fall 2020

Linear Regression

Part A (Model Code): 478 (68 pts) & 878 (78 pts)
Part B (Data Processing): 478 & 878 (7 pts)
Part C (Model Evaluation): 478 (25 pts) & 878 (35 pts)
Part D (Written Report): 478 & 878 (25 pts)
Extra credit (BONUS) tasks for both 478 & 878: 30 pts

Total: 478 (125 pts) & 878 (145 pts)

Last Name 1: Zhang

First Name 1: Beichen

NUID 1: 39567681

Last Name 2: Hu

First Name 2: Qiao

NUID 2: 15142006

Last Name 3: Zhang

First Name 3: Ligang

NUID 3: 12509359

Obtained Score:

Using Linear Regression Model to Predict the Red Wine Quality

Beichen Zhang[1] and Ligang Zhang[1] and Qiao Hu[1]

***Abstract*— This electronic document is the final report of the programming assignment two of the class in 2020 fall–CSCE 478/878-Introduction to Machine Learning. In this assignment, we used the wine quality dataset from the University of California, Irvine (UCI) to predict red wine quality. Linear regression with an optional polynomial degree parameter was implemented in this assignment. Both batch gradient descent (BGD) and the stochastic gradient descent (SGD) were tested to learn the weights of the regression model. And the L1 and L2 regularization were employed to address the overfitting. The results showed that both L1 and L2 regularization would have similar effects on the models. When the polynomial degree is equal to 1, the model would have a better accuracy than the model with the polynomial degree at 3. The variance of the model would increase if we rise the polynomial degree. Moreover, with properly tuning, the BGD and SGD would produce similar results from the model. For the coding part, every team member constructed their own scripts from scratch which were eventually summarized into one code by checking with each other. For the written report, Beichen Zhang completed the data summary section; Qiao Hu completed the method section; Ligang Zhang completed the result section.**

I. Data Summary

The wine quality data set was downloaded from the Center for Machine Learning and Intelligent Systems at the UCI. It was developed and calculated by Cortez and his fellows in 2009 [1]. Two subsets of red and white Portuguese "Vinho Verde" wine were included. This assignment employed the data set of red wine with the sample size of 1599 and applied the linear regression to predict its quality. The quality was evaluated by scores in the range between 0 and 10. Table I shows the descriptive statistics of dependent and response variables.

We preformed the model selection by comparing the scatter plots of the Pearson correlation between every two variables shown in Fig. 1. Because the co-linearity issue is minor, we did not drop any features.

Additionally, because of the different units of each variable, we used feature scaling to reduce the skewness of variance in the predictors. The Gaussian standardization function is shown as (1), where Z is the z-scores that represent the standardized values from the data, x is the data point, $\bar{x}$ is the mean of the corresponding feature, and σ is the standard deviation.

$$Z = \frac{x - \bar{x}}{\sigma} \tag{1}$$

[1] School of Natural Resources, University of Nebraska-Lincoln, Lincoln, Nebraska, 68503, USA

TABLE I
DESCRIPTIVE STATISTICS OF VARIABLES

	fixed acidity	volatile acidity
Mean	8.320	0.528
Standard Deviation	1.741	0.179
25th Percentile	7.100	0.390
50th Percentile	7.900	0.520
75th Percentile	9.200	0.640
	citric acid	residual sugar
Mean	0.271	2.539
Standard Deviation	0.195	1.410
25th Percentile	0.090	1.900
50th Percentile	0.260	2.200
75th Percentile	0.420	2.600
	chlorides	free sulfur dioxide
Mean	0.087	15.875
Standard Deviation	0.047	10.460
25th Percentile	0.012	1.000
50th Percentile	0.070	7.000
75th Percentile	0.079	14.000
	total sulfur dioxide	density
Mean	46.468	0.997
Standard Deviation	32.895	0.002
25th Percentile	22.000	0.996
50th Percentile	38.000	0.997
75th Percentile	62.000	0.998
	pH	sulphates
Mean	3.311	0.658
Standard Deviation	0.154	0.170
25th Percentile	3.210	0.550
50th Percentile	3.310	0.620
75th Percentile	3.400	0.730
	alcohol	quality
Mean	10.423	5.636
Standard Deviation	1.066	0.808
25th Percentile	9.500	5.000
50th Percentile	10.200	6.000
75th Percentile	11.100	6.000

II. Methods

The model used in this study was linear regression, and we used two iterative optimization methods to assess the model performance with the dataset introduced above: gradient descent and stochastic gradient descent algorithm. First of all, both closed form solution and iterative optimization solution are aiming to minimize the cost function defined as (1), where y_i is the response of i_{th} observation and $\vec{W}^{\mathrm{T}}\vec{X}_i$ is the corresponding prediction, and W is the model parameter.

$$L(\vec{W}) = 1/n\sum_{i=1}^{n}(y_i - \vec{W}^{\mathrm{T}}\vec{X}_i)^2 \tag{2}$$

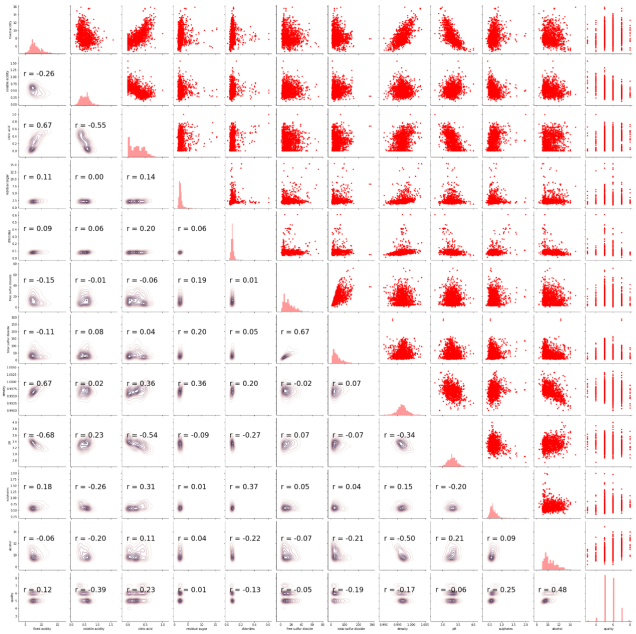

Fig. 1. The scatter plot of response and dependent variables

A. Closed form solution and iterative optimization

In closed-form solution, the normal equation is used to derive the weights, and ordinary linear system (OLS) regression is the best linear unbiased estimator (BLUE), which is defined as (3), where X is the feature matrix with dimension n by (d+1), W is the model parameter vector with the shape of (d+1) by 1, and Y is the response with the shape of n by 1. The d is the feature dimension and n the number of sample observations.

$$W_{OLS} = (X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}Y \tag{3}$$

However, the OLS solution requires the feature X as matrix without collinearity, which in most cases is not feasible, such as data in high dimension feature space or high-degree polynomial feature space where feature in different dimension may highly dependent. Thus, the regularized normal term need to be added on the $X^{\mathrm{T}}X$ term to ensure that it is inevitable. We can arrive at the least square method using the maximum likelihood estimation by assuming that the collected observations (x) follow a Gaussian distribution for the target variable (y).

$$W = (X^{\mathrm{T}}X + \lambda\mathbb{1})^{-1}X^{\mathrm{T}}Y \tag{4}$$

The iterative optimization solution employs gradient descent to gradually adjust the model parameters to minimize the cost function until the loss eventually converging to the similar weight patameter in close form solution. By derived the gradient of the loss function (the $L(\vec{W}$ term are changed to J(θ)) denoted in (2), we take multiple steps to update the weight parameter θ, where in every step, we subtract a small part of the derivative of J(θ) from the last weight parameter so that we can reach to the minimum point of J(θ). The update rule is defined as (5).

$$\theta_j := \theta_j - \eta\frac{\partial J(\theta)}{\partial\theta_j} \tag{5}$$

For a given instance, the Normal equation computes the inverse of $X^{\mathrm{T}}X$, which is a (d+1) by (d+1) matrix. The computational complexity of inverting such a matrix is typically about $O(d^3)$. Its complexity of the training dataset is linear to the number of instances (samples), typically denoted as O(n). Therefore, the total computational complexity given n instance is about $O(nd^3)$. The normal equation will become very slow when the number of features d or number of instances n get very large.

In every step (iteration), the iterative solution BGD algorithm uses the whole training data at every step, it scales well with the number of features. The time-complexity of the dot product between X (feature with d dimensional) and the weight parameter (d dimensional) is of d plus the product of the weight parameter θ and feature X, typically denoted as O(d). While compared with the OLS method with time-complexity of (computing the inverse of $X^{\mathrm{T}}X$) O(d3), gradient descent is much faster than closed form solution, especially when there are hundreds of thousands of features.

In this study, the total number of feature (d) is 11 and the total number of instance n is 1599, which resulting in a $1599 * 11^3$ computational complexity for close form solution while $1599 * 11$ for the iterative solution based batch gradient descent algorithm.

The close form solution is probably acceptable when only the 11 original data is used. While if we go high degree polynomial features is not applied, the memory pressure will increase. The iterative solution will be more suitable.

B. Iterative optimization

$$\vec{\theta} := \vec{\theta} - \frac{\eta}{m}\vec{W}^{\mathrm{T}}.(\vec{W}.\vec{\theta} - \vec{y}) \tag{6}$$

Two iterative optimizations will be tested in this study: batch gradient descent and the stochastic gradient descent algorithm. The Batch Gradient Descent algorithm uses the loss calculated by the whole batch of training data to update the weight vector based on (6) at every epoch, while the stochastic gradient descent algorithm uses the loss calculated by a randomly chosen instance to update the weight vector based on (6) at every iteration and the training will iterate n times to move into next epoch.

Theoretically, the two different methods will result in different training updates during each iteration. The batch gradient descent (BGD) updates the weight based on loss from the whole training dataset, while the stochastic gradient descent algorithm (SGD) updates the weight vector based on single and random instances. Thus, the loss and gradient update of stochastic gradient descent will be much unstable than batch gradient descent. When a large learning rate was set, the loss and gradient of stochastic gradient descent will likely explode after a few iterations and return inf or NaN

value. The statistic comparison was shown in the result section. When a suitable learning rate was set, both methods return similar weights.

C. Model assessment

The square root on MSE, calculated by (2), which ensures that RMSE is measured on the same scale (and in the same units) as the target variable, will be used to assess the model performance during each iteration. Five-fold cross-validation will be used to choose the optimal model parameter setting. The model that can return the lowest mean RMSE across all folders will be chosen as the optimal model to produce the final predictions.

The learning curve and polynomial feature were employed to assess the suitable training dataset and model complexity (polynomial feature degrees) and avoid over-fitting and under-fitting issues. The training size was set as 10 which means that the iteration will begin with 10 samples and every iteration will successively add 10 samples in each iteration. The polynomial feature degree will be set as 1, 2, 3, 4, and 5 which results in 11, 77, 363, 1364, 4367 feature dimensions respectively, to test the suitable polynomial degree.

III. Results

A. Comparision of BGD and SGD with different learning rate

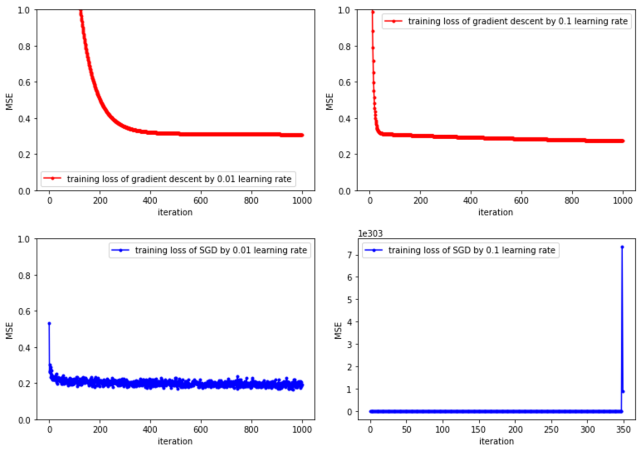

Fig. 2. Comparison of BGD and SGD by 0.1 and 0.01 learning rate

Fig. 2 show the comparison of BGD and SGD iterative solution based on the different learning rates. The learning rate was set as 0.1 and 0.01 but with all other parameter setting identical. The result shows that when the learning rate is set as 0.1, the BGD (plot on the left with the red curve) returns smooth model updates and the MSE plateau around 0.2-0.4. The SGD with a 0.01 learning rate (plot on the left with the blue curve) return similar MSE eventually but a very unstable learning curve. When the learning rate goes to 0.1 (higher), the BGD (plot on the right with the red curve) return similar MSE eventually but the learning curve reaches to plateau more quickly, but the SGD (plot on the right with the blue curve) go to an explosion (with inf loss) very quickly(if you zoom in to see the y limit) and never plateau.

This means that BGD and SGD can reach similar model weights update eventually but very different updating processes when the learning rate was set properly. The BGD always returns a smooth learning curve (model weight updates) while the SGD returns very unstable model weight updates. The SGD will easily to go explode if the learning rate was set too high.

TABLE II
Optimal Model Parameters (Degree = 1)

Measure	Value
Lambda	0.0
Learning Rate	0.1
Regularizer	L1
Training MSE	0.265923
Validation MSE	0.268132
Average MSE	0.2670275

B. Learning curve of the model with degree 1

The best model parameters for feature degree of 1 are listed in Table II. Fig. 3 shows the learning curve of the degree = 1 model. The training RMSE and validation RMSE curve show a relatively ideal condition of the learning curve. The decrease of the validation curve and increase of training curve seemed to balance in this model, which means that as the training set increases with feature degree 1, neither a high bias nor a high variance condition happens.

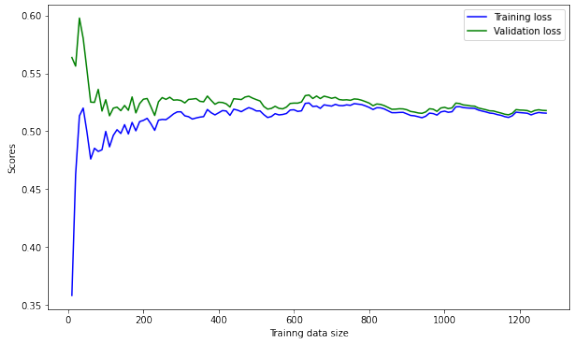

Fig. 3. Learning Curve Degree = 1

C. Learning curve of the 3rd degree polynomial data matrix

The best model parameters for feature degree of 3 are listed in Table III. This model shows a high variance trend, especially when the number of training data between around 100 and 300. With the increase of the training sample, the model eventually plateau around 1.35-1.40 average RMSE of both the training error and test error. The overfitting may more likely happen when applying this model when training data is small. Although the overfitting issue get mitigated, its error is still much higher than the degree 1 model (plateau around 0.50-0.55).

TABLE III

OPTIMAL MODEL PARAMETERS (DEGREE = 3)

Measure	Value
Lambda	0.0
Learning Rate	0.001
Regularizer	L1
Training MSE	1.852208
Validation MSE	1.871624
Average MSE	1.861916

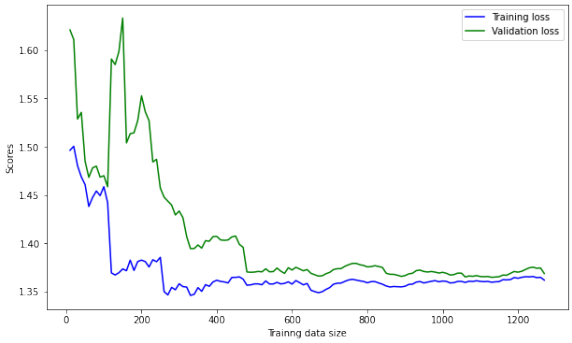

Fig. 4. Learning Curve Degree = 3

D. Performance of the chosen model

Consider the learning curve and overall RMSE of the degree = 1 and degree = 3 model. The degree = 1 model is the better model to apply in this project. Fig. 5 is the MSE curve of different degrees. The curve of training RMSE and validation RMSE are both increasing with the increase of degrees. The gap between two curves are also increasing with the increase of degrees, which means there is a high variance (overfitting) trend with when the degree of model increase from 1 to 5. This figure is also a proof of that the degree 2 is probably the optimal feature polynomial degree to apply on the data.

E. Conclusions

The study shows that linear regression is a fundamental and effective machine learning model to resolve the regression issues. The results show that the increasing of the model complexity (polynomial features degree) may lead to high variance issues. The results also show that the two different types of regularizer (l1 and l2) have little effect on the model performance. The comparison of BGD and SGD show that they can reach to similar model performance when proper parameter are set, but the SGD is more unstable than BGD.

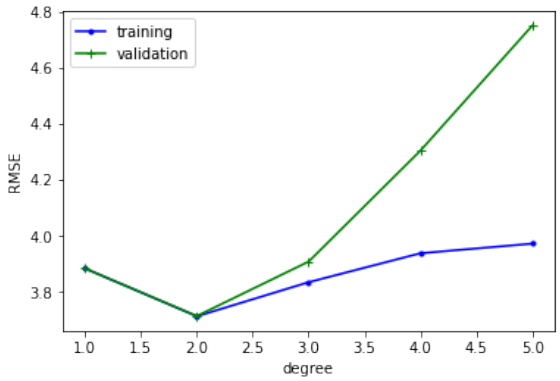

Fig. 5. RMSE value for different degrees

REFERENCES

[1] Cortez, P., Cerdeira, A., Almeida, F., Matos, T. and Reis, J., 2009. Modeling wine preferences by data mining from physicochemical properties. Decision Support Systems, 47(4), pp.547-553.