COMPUTATIONAL METHODS FOR DATA ANALYSIS QUALIFYING RED WINE

AVERY LEE

Applied Mathematics Department, University of Washington, Seattle, WA
leeave22@uw.edu
02/25/2022

Abstract. Qualifying Red Wine creates a Machine Learning (ML) model that reads in qualities of different types of wine and their ratings from 0 to 10, and provides ratings of a new batch of wine given their qualities. Three ML models are created using linear regression, Gaussian (RBF) kernel regression, and Laplacian kernel regression. Cross validation will also be computed to find the best parameters for the kernel models. This paper will go through the mathematical theories behind the calculation methods, the algorithms used to create the ML model, the final computational results, and conclusions.

1. Introduction and Overview

Machine Learning is a field of study that has been expansively growing due to its endless applications; it can be used to predict, classify, approximate, or even cluster any type of dataset [1]. In this paper, three ML models will be created by studying given data about wines' qualities and their ratings from 0 to 10, to output a new wine's ratings given its qualities. The qualities, in the order provided in the dataset, are: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, and alcohol.

Numerous ML models exist that can possibly fit the dataset in question. Since it is unknown which model would provide the most accurate wine ratings, multiple models should be studied. For this example, linear, Gaussian (RBF) kernel, and Laplacian kernel regression models will be created [2].

Section 2 of this paper will go over the mathematical background necessary to understand the computations. Section 3 will cover the algorithms and Python software packages. Then Section 4 will summarize the results from the computations, and Section 5 will summarize the learnings.

2. Theoretical Background

2.1. **Machine Learning.**

In Machine Learning, it is common practice to create a model by reading in a training and testing dataset. Both datasets provide an X_{train} and X_{test} dataset that represent the features, and a Y_{train} and Y_{test} that represent the outcomes based on the features. The Ys are what are being predicted.

An important part of any modeling procedure is observing the error of the outputs. It can be done by studying the mean squared error, calculated by using this equation where n is the number of datapoints, Y_i is the observed value, and $\hat{Y}_i$ is the predicted value:

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

2.2. **Linear Regression.**

Linear regression (least squares) is one of the simplest ML models that can be created. It assumes a linear relationship between the input variables and output values. Then, it finds the coefficients b_i that give the smallest squared error for this equation where x_i are the independent variables, y is the dependent variable, and b_i are the coefficients:

$$y = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n$$

2.3. **Ridge Regression.**

Ridge regression is very similar to linear regression, but it solves this optimization problem:

$$\min_{\beta} ||A\beta - Y||^2 + \lambda ||\beta||^2$$

where A contains features in the form

$$A = \begin{pmatrix} F_0(x_0) & F_1(x_0) & \dots \\ F_0(x_1) & F_1(x_1) & \dots \\ \dots & & \\ F_0(x_{N-1}) & F_1(x_{N-1}) & \dots \end{pmatrix}$$

Now if $f(x)$ is defined as

$$f(x) = \sum_{j=0}^{J-1} \beta_j F_j(x)$$

a kernel can be defined as

$$K(x, x') = \sum_{j=0}^{J-1} F_j(x) F_j(x')$$

and ridge regression, in other words, can be defined as this optimization problem

$$\min_{f \in H_k} ||f(X) - Y||^2 + \lambda ||f||^2_{H_k}$$

where

$$H_k = \{f : \mathbb{R}^d \to \mathbb{R}\}$$
$$||f||^2_{H_k} = \sum_{j=0}^{J-1} \beta_j^2 = ||\beta||^2$$

Now, the model can be differed based on the kernel. The Gaussian (Radial Basis Function, RBF) Kernel is defined as

$$K_{rbf}(x, x') = exp(-\frac{||x - x'||_2^2}{2\sigma^2}) = exp(-\gamma ||x - x'||_2^2), \quad \gamma = \frac{1}{2\sigma^2}$$

The Laplacian Kernel can be defined as

$$K_{laplacian}(x, x') = exp(-\frac{||x - x'||_1}{\sigma}) = exp(-\gamma ||x - x'||_1), \quad \gamma = \frac{1}{\sigma}$$

2.4. **Cross Validation.**

From the formulas above, it can be seen that there are multiple parameters that determine the outcomes of the optimization problem. In fact, change in these parameters can greatly change the outcome, so it is important to choose appropriate values. The best regularization parameter λ and kernel length scale parameter σ can be found by testing out different values and seeing which gives the smallest error. The process of trial and error will be described in Section 3

3. Algorithm Implementation and Development

3.1. **Python Packages and Notable Functions.**

Python was used to compute the algorithms and produce plots. Below is a list of packages used.

Python Package or Function	Description
`google.colab, drive`	Gives access to the user's Google Drive account where they can access the data.
`scikit-learn [3]`	Provides tools for machine learning algorithms including linear_model, metrics, and kernel_ridge.
`numpy [4]`	Allows usage of multidimensional arrays and functions to manipulate those arrays.
`matplotlib.pyplot [5]`	Useful for graphics manipulation.
`fit()`	Fits the X and Y values into a model.
`KernelRidge()`	Helps create a kernel ridge regression model.
`predict()`	Predicts the output using the model created.
`mean_squared_error()`	Calculates the mean squared error given the true and predicted values.

Table 1. Python Packages and Functions

3.2. **Setup.**

The given data includes a training dataset X_{train} with 1115 rows each representing a wine, and 11 columns each representing a quality of the wine, of which all listed in Section 1. The Y_{train} represents the rating each wine gets from 0 to 10. X_{test} and Y_{test} are the same thing but with only 479 wines. An X_{new} is also provided, which represents a new batch of 5 wines that do not have a rating (Y_{new}) yet.

Before doing anything with the data, everything should be normalized and centered with mean of 0 and standard deviation of 1. For all datasets, the training dataset's mean and standard deviation will be used to normalize and unnormalize. It is important to remember that all of the datasets used from here on out is normalized.

3.3. **Linear Regression.**

Creating a linear regression model in Python is extremely simple, using the LinearRegression().fit() function. It will automatically provide the best least squares model given the X_{train} and Y_{train} as parameters, using the formulas explained in Section 2. The MSEs and predictions will be discussed in Section 4.

3.4. **Gaussian (RBF) Kernel Regression and Cross Validation.**

Testing out many λ and σ values can take a very long time to run on Python. However, accurate results are still desired. So a way to achieve both as much as possible with the computational budget available is to just test out values in a wider range, then narrow it down as much as possible.

This can be done by plotting the contour plot of the absolute value of the ”scores” that are determined by the errors, where higher the score, the better (less error). The x and y axes are $log_2\lambda$ and $log_2\sigma$ respectively. This will display which range of λ and σ are desired. This process is repeated until the contour graph displays the best possible parameter values.

The optimized λ and σ are then used to calculate the alpha (same as lambda) and gamma (equation provided in Section 2). The alpha and gamma parameters are used in the KernelRidge() and fit() functions provided in Python to create the RBF model using the X_{train} and Y_{train} dataset.

3.5. **Laplacian Kernel Regression and Cross Validation.**

The process for the Laplacian regression is very similar to the RBF, except for minor changes such as the gamma equation stated in Section 2. In this example, more iterations were needed to get to the optimal λ and σ.

3.6. **Mean Squared Error and Rating Prediction.**

The mean squared errors for the training and testing datasets using linear, RBF kernel, and Laplacian kernel regression can be calculated. Results are in Section 4

Now that the three models are created, the batch of new wines can be rated. Since all the data used so far have been normalized, this prediction will also be normalized. It can be unnormalized by multiplying the standard deviation of the Y_{train}, then adding the mean of the Y_{train}.

4. Computational Results

4.1. **Cross Validation and Optimal λ and σ Parameters.**

All intervals used are evenly spaced by 10 points. Starting with cross validation for Gaussian (RBF) kernel regression, the initial range of 0 to 4 and -6 to 1 were chosen for $log_2\sigma$ and $log_2\lambda$. The red dot represents the spot with the highest score.

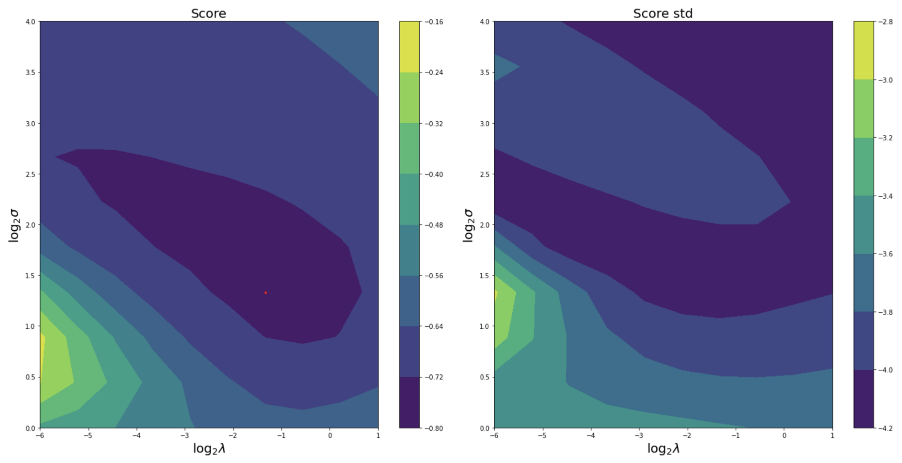

Figure 1. Contour Plot of Gaussian Cross Validation (1st Iteration)

Looking at the left graph, the highest scores are in the dark purple region, of around interval 0.75 to 2.75 and -5.75 to 0.75 for $log_2\sigma$ and $log_2\lambda$. This interval is used for the same process.

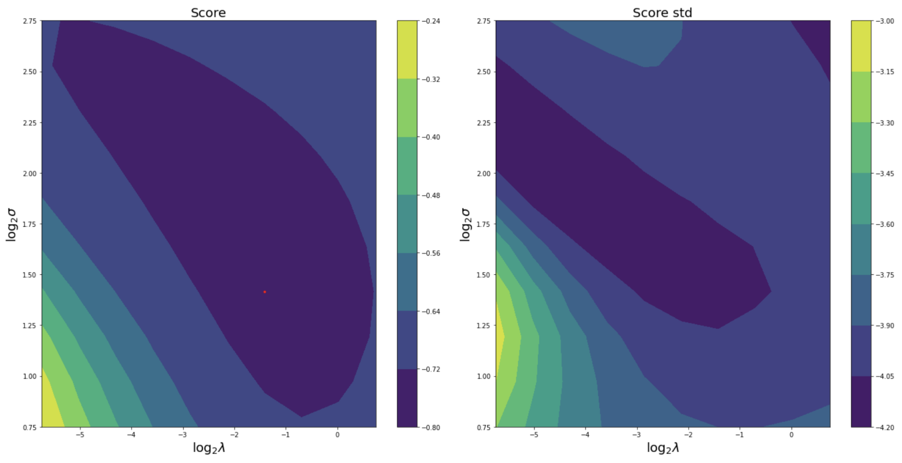

FIGURE 2. Contour Plot of Gaussian Cross Validation (2nd Iteration)

From the graph, there is not much room to close the interval so this is good enough with the existing computational budget. The red point is the optimal $log_2\sigma$ and $log_2\lambda$.

The same process is done for Laplacian regression. To simplify, after trimming the interval 5 times, the optimal point can be found on the 6th iteration shown below.

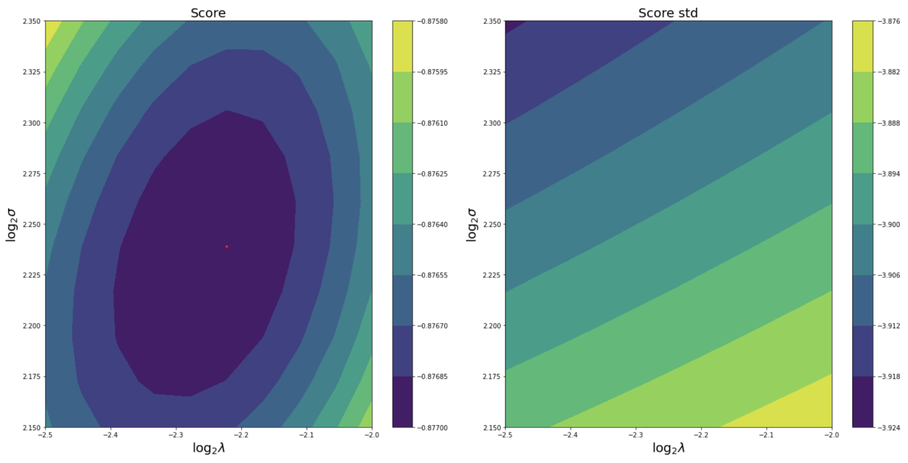

FIGURE 3. Contour Plot of Laplacian Cross Validation (6th Iteration)

From these processes, the optimal σ and λ are below, using $\alpha = \lambda, \sigma = \sqrt{\frac{1}{2\gamma}}$ for Gaussian and $\alpha = \lambda, \sigma = \frac{1}{\gamma}$ for Laplacian.

	σ	λ	γ
Gaussian (RBF)	2.67	0.37	0.07
Laplacian	4.72	0.21	0.21

TABLE 2. Optimal σ, λ, and γ

4.2. **Mean Squared Errors.**

	Linear	Gaussian (RBF)	Laplacian
Training	0.627	0.413	0.059
Testing	0.747	0.665	0.609

Table 3. Mean Squared Errors

The Laplacian had the lowest MSE compared to the other two methods, but the extremely low training MSE might indicate overfitting of the training dataset. A better λ and σ need to be found, which requires higher computational budget.

4.3. **Rating Predictions.**

Wine	1	2	3	4	5
Linear	6.00	5.28	5.56	6.06	5.94
Gaussian (RBF)	6.07	5.47	5.48	6.24	6.19
Laplacian	6.07	5.45	5.63	6.00	6.03

Table 4. Wine Rating Predictions for New Batch

The ratings for each wine for each ML method are all pretty similar when rounded to an integer.

5. Summary and Conclusions

In this paper, three different Machine Learning techniques, linear, Gaussian (RBF) kernel, and Laplacian kernel regression were used to predict the ratings of wine given the different features of the wine. For kernel regression, cross validation with contour graphs was used to find the optimal σ and λ to use as parameters. The MSE was found to see the effectiveness of the techniques, and rating predictions were made using each method. In the future, these calculations can be used not only with wine, but with other type of prediction needs with similar data characteristics.

Acknowledgements

The author is thankful to TA Katherine Owens for the general rundown and clarifications on calculations and code, as well as classmates such as Pranav Natarajan that reconfirmed my data results and Python syntax. Last but not least, the author is thankful to Professor Bamdad Hosseini for teaching most of the mathematical concepts mentioned in the paper.

References

[1] Bamdad Hosseini. Introduction to machine learning. University of Washington (LOW 216), Jan 2022.

[2] Bamdad Hosseini. Kernel ridge regression. University of Washington (LOW 216), Feb 2022.

[3] 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*, 12:2825–2830, 2011.

[4] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[5] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.