AMATH 582: HOMEWORK 3

CASSIA CAI

School of Oceanography, University of Washington, Seattle, WA
fmc2855@uw.edu

Abstract. We develop an algorithm to predict the quality of wine based on a series of physicochemical attributes. We apply this algorithm to a batch of new wines. Data obtained from Portuguese "Vinho Verde" variants are used. This data set is widely used to test regression and classification methods. The data set is split into training and testing data. We apply linear regression to fit a linear model to the training set. We apply Gaussian and Laplacian kernel ridge regression to fit a nonlinear model to the training set, using 10-fold cross validation to tune σ and λ. The Laplacian kernel regression MSE (training MSE: 0.040, test MSE: 0.606) is lowest while the linear regression MSE is highest (training MSE: 0.628, test MSE: 0.747). Gaussian kernel ridge regression resulted in a training MSE of 0.438 and test MSE of 0.667. An extension is to develop an outlier detection algorithm and to test feature selection methods.

1. Introduction

Wine quality is determined by its physicochemical attributes, such as fixed acidity, volatile acidity, citric acid, residual sugars, and pH among many others. Moreover, wine quality is also assessed by sensory tests, in particular taste. Taste is not a well understood. In this study, we develop an algorithm that predicts the quality of wine and to learn valuable information about trends. We use data related to red variants of the Portuguese "Vinho Verde" wine, described in greater detail in [1] and publicly available Portuguese "Vinho Verde" here.To develop our algorithm, we use three methods: linear regression, Gaussian kernel ridge regression, and Laplacian kernel ridge regression. For the Guassian and Laplacian kernel ridge regressions, we use 10-fold cross validation (CV) to tune the length scale of σ and the regularization parameter λ. We then compare the training and test mean square errors (MSEs) of all three methods as well as discuss which method is most effective and why that might be the case.

The wine dataset we used is widely used to test regression and classification methods as well as develop outlier detection algorithm and to test feature selection methods [2]. We used only data related to red wine variants. The methods of this study can be applied to other marketing problems, such as modeling consumer taste for specific products.

2. Theoretical Background

We apply three different models: linear regression, Gaussian kernel ridge regression, and Laplacian kernel ridge regression. A kernel is often referred to as the "kernel trick," where a linear classifier is used to solve some non-linear problem through transforming linearly inseparable data into linearly separable data in a new space. We use the kernel function $\kappa(\mathbf{x}, \mathbf{x}')$ to replace inner products $< \mathbf{x}, \mathbf{x}' >$ given that we have defined our feature vector as $\phi(\mathbf{x}) = [\kappa(\mathbf{x}, \mathbf{x_1}), ..., \kappa(\mathbf{x}, \mathbf{x_N})]$.

Our previous approach to ridge was to solve problems of the form

$$\underset{\beta}{\text{minimize}} \|A\beta - Y\|^2 + \lambda\|\beta\|^2$$

Date: February 21 2022.

where A was our feature matrix. Now, we can consider that underneath this model, we have the kernel:

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

with features $\Psi_j : \mathbb{R}^d \to \mathbb{R}$ and whose RKHS is

$$\|f\|_{H_K}^2 = \sum_{j=0}^{J-1} \beta_j^2 = \|\beta\|^2$$

So we have:

$$\underset{f \in H_K}{\text{minimize}} \|f(X) - Y\|^2 + \lambda \|f\|_{H_K}^2$$

We consider the Gaussian and Laplacian kernels in this study.

Linear regression: We use a simple linear regression [3].

$$\hat{f}(\underline{x}) = \hat{\beta}_0 + \sum_{j=0}^{d-1} \hat{\beta}_j x_j$$

$$\hat{\beta} = argmin \|\hat{f}(x) - \underline{y}\|^2$$

We solve to find the regression solution:

$$\hat{\underline{\beta}} = argmin \frac{1}{2\sigma^2} \|A\underline{\beta} - \underline{Y}\|^2 + \frac{\lambda}{2} \|\underline{\beta}\|^2$$

$\hat{\beta}$ is the predicting coefficient vectors. λ is the regularization/penalization parameter.

Gaussian (RBF) kernel ridge regression: The Gaussian kernel is defined as:

$$\kappa(\mathbf{x}, \mathbf{x}') = \exp(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T \Sigma^{-1})(\mathbf{x} - \mathbf{x}')$$

We can rewrite the above equation if Σ is diagonal:

$$\kappa(\mathbf{x}, \mathbf{x}') = \exp(-\frac{1}{2} \sum_{j-1}^{D} \frac{1}{\sigma_j^2} (x_j - x_j^{'})^2)$$

Where σ_j is the characteristic length scale of j. In this paper, Σ is spherical, so we get this kernel:

$$\kappa(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right) for \mathbf{x}, \mathbf{x}' \in \mathbb{R}^{11}$$

Where σ^2 is the bandwidth [4]. The above equation is a function of $\|\mathbf{x} - \mathbf{x}'\|$ This is a radial basis function (RBF).

Laplacian kernel ridge regression: The Laplacian kernel is defined as:

$$\kappa(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|_1}{\sigma}\right) for \mathbf{x}, \mathbf{x}' \in \mathbb{R}^{11}$$

We have 11 attributes. The Laplacian kernel is a RBF kernel and is less sensitive to changes in σ.

Mean Squared Error: Defining the trained regression model as:

$$\hat{f}(\underline{x}) = \sum_{j=1}^{J} \hat{\beta}_j \psi_j(\underline{x})$$

We define the training MSE as:

$$MSE_{\text{train}}(\hat{f}, \underline{Y}) = \frac{1}{N} \sum_{n=0}^{N-1} |\hat{f}(\underline{x}_n) - y_n|^2 \quad for \underline{x}_n \in \mathcal{X}, y_n \in \mathcal{Y}$$

We calculate the test MSE similarly. With both training and test MSEs, we can assess the performance of the classifier.

Cross Validation: Training and testing errors are dependent on the choice of λ, which is a the regularization parameter. The choice of λ can result in variations in MSE between the test and training datasets [3]. A high λ results in a too simple model. A low λ results in a too complicated model. Different λ values can be tested by calculating corresponding training and testing MSEs. We could also use K-fold CV, which splits the training data into K-subsets (some as training and others as testing). K-fold CV then uses these different parts of the training data to test and train the model iteratively. K-fold CV is used in this paper.

3. Algorithm Implementation and Development

The provided wine dataset is already split into training and test data. The training data consists of 1115 instances of different wines. Each instance has 11 attributes (1. fixed acidity, 2. volatile acidity, 3. citric acid, 4. residual sugar, 5. chlorides, 6. free sulfur dioxide, 7. total sulfur dioxide, 8. density, 9. pH, 10. sulphates, and 11. alcohol). Each instance also has a corresponding quality score, which ranges from 0 to 10. We are also provided with 5 instances of wines without a quality output, which we aim to predict with our algorithm. To assess wine quality, we first normalize the training, test, and new batch data. Then, we use linear regression (least squares) to fit a linear model to the training set as well as kernel ridge regression to fit a nonlinear model to the training set, using Gaussian and Laplacian kernels respectively. To apply linear regression, we use scikit-learn's LinearRegression package. For the Gaussian and Laplacian kernels, we use scikit-learn's KernelRidge package. To tune the length scale of σ and the regularization parameter λ, we use two methods: (1) scikit-learn's GridSearchCV and (2) a loop within which cross validation is implemented using scikit-learn's crossvalscore and visual inspection. The training and test MSEs from these two methods are comparable.

The steps undertaken to predict the quality of wine are outlined in the pseudo-code below. I coded Python and used NumPy and scikit-learn for mathematical calculations and Matplotlib for visualizations [5, 6].

Pseudo-code: Predict the quality of wine from a series of physicochemical measurements

---

Step 1: Normalize training, testing, and new batch data

a. Define training and testing matrices consisting of the 11 physicochemical attributes
b. Define labels (last column of quality values of the data)
c. Normalize training, testing, and new batch data using the training set mean and STD

Note: shift and scale testing and new batch data according to the training set

Step 2: Use linear regression by calling scikit-learn's linear regression package and make label predictions

Step 3: Use Gaussian kernel ridge regression

a. Method 1: Call scikit-learn's GridSearchCV and set K-fold CV = 10

b. Method 2: loop through a range of σs and corresponding γ values, call scikit-learn's crossvalscore to determine optimal hyperparameters, and use visual inspection

Note. Define $\gamma = \frac{1}{2\sigma^2}$

Step 4: Use Laplacian kernel ridge regression

Note. Repeat steps from Step 3 but specify the Laplacian kernel and define γ as $\frac{1}{\sigma}$

Step 5: Calculate the training and test MSEs

Step 6: Predict the quality of wines of the new batch data using the three methods

---

4. Computational Results

The linear regression method resulted in the highest training MSE (0.628) and test MSE (0.747) compared to the other two methods. From Figure 1, we observe that the attributes with the largest positive coefficients are alcohol followed by sulphates. The attributes with the largest negative coefficients are volatile acidity and chlorides. The fact that the linear regression method resulted in the highest training and test MSEs is not unexpected because, as its foundation, linear regression assumes that linear relationship between the input variables and the output variable. The relationship between wine attributes and prediction, or generally human taste, is described better with a nonlinear model.

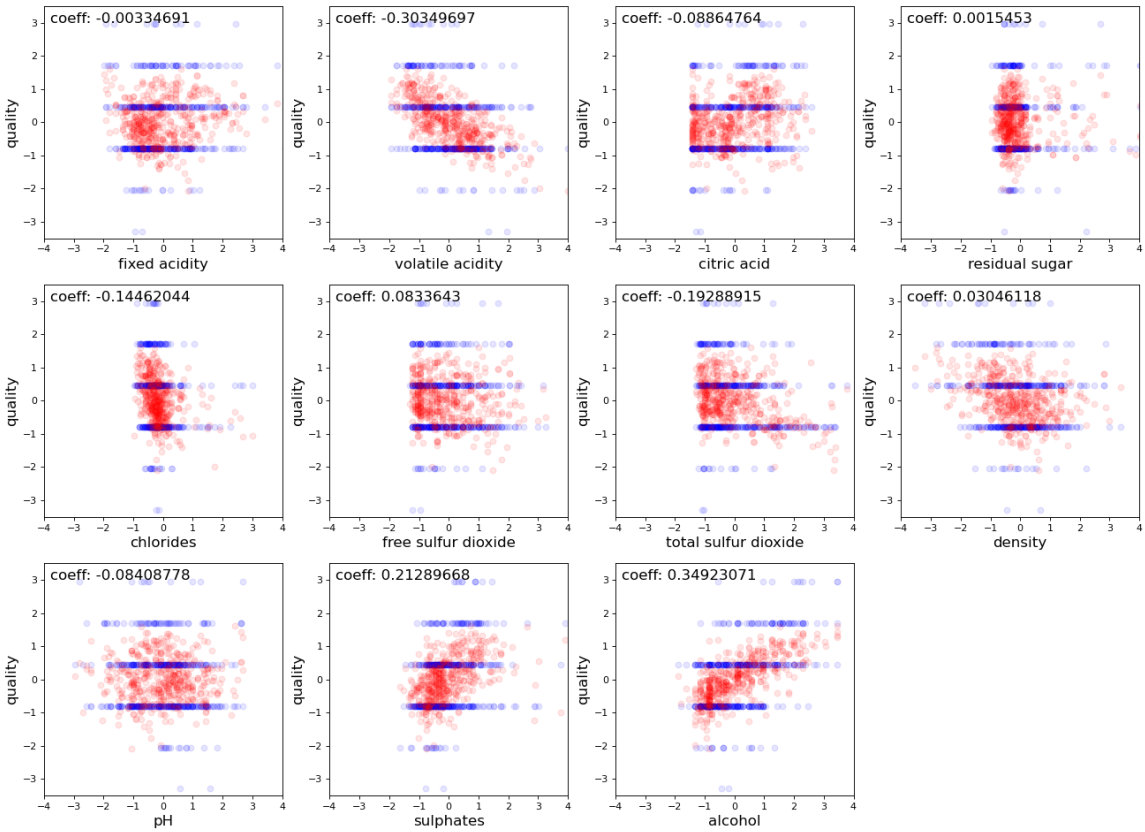

Figure 1. Blue circles indicate the the features of the test data with the labels of the test data. Red circles indicate the features of the test data plotted against the predicted labels of the test data using linear regression.

For the Guassian and Laplacian kernels, we determined the hyperparameters using two methods: (1) scikit-learn's GridSearchCV and (2) looping, visual inspection, and using crossvalscore. In both these methods, we use a K-fold CV value of 10 (CV is described in the Section 2. Theoretical Background). The tables show the results from method 2. However, we discuss the results from GridSearchCV as well. Using method 2, we define a range of σ and λ values to loop through and use scikit-learn's model selection package. We visually inspect to narrow our range of σ and λ values to loop through (Figures 2 and 3). We use 10-fold CV to tune the scale of σ and λ for each kernel. The reason why we picked the values (Table 1) is because after trying a large range of values and narrowing our range, we found that any additional adjustments and fine-tuning did not alter the algorithm's σ and λ outputs. Moreover, from cross-checking with GridSearchCV, we found that the α and γ values were comparable. Calculating the MSEs from looping through and

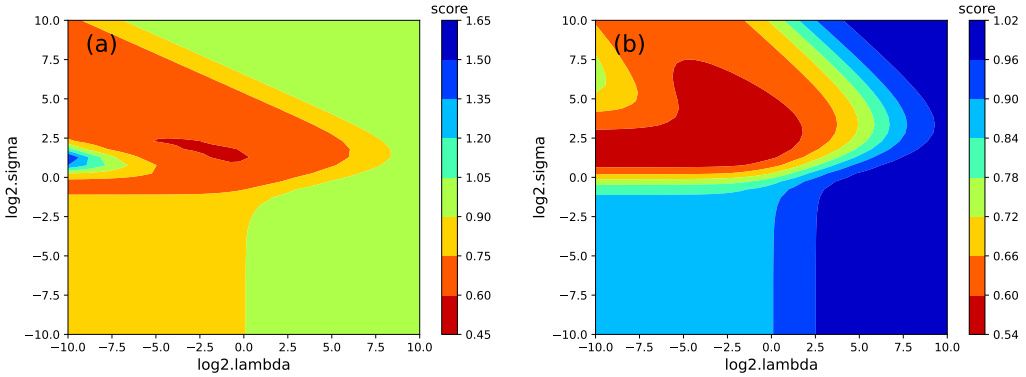

FIGURE 2. (a) Scores for Gaussian kernel. (b) Scores for Laplacian kernel. Note that the range for σ and λ are different for these two figures. The negative MSE is used for scoring. An alternative is to use the negative root MSE.

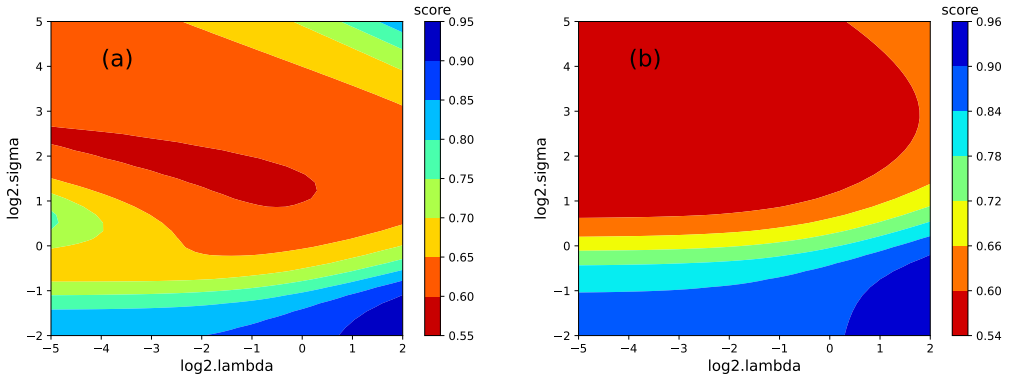

FIGURE 3. Same as Figure 2 but with narrower σ and λ ranges.

using crossvalscore and comparing those MSEs from the MSEs from GridSearchCV, we found that the MSEs were also comparable. The Gaussian kernel training MSE from GridSearchCV is 0.438 while the test MSE is 0.667. This is similar to the training and test MSEs from looping through and visual inspection (Table 2). Similarly, the Laplacian kernel training MSE from GridSearchCV is 0.040, while the test MSE is 0.606. Our hyperparameters are not necessary the 'true' optimal values, however, this is a good choice given our computation budget and the fact range of σ and λ values, there is some convergence to values that the algorithm outputs.

Method	sigma	lambda	alpha	gamma
Gaussian kernel	3.704	0.157	0.157	0.036
Laplacian kernel	3.470	0.202	0.202	0.288

TABLE 1. Hyperparameters using looping, visual inspection, and crossvalscore

From Table 2, we see that linear regression performed the worst while the Gaussian kernel ridge regression and Laplacian kernel ridge regression performed similarly (looking at test MSEs). The Gaussian kernel training MSE is significantly higher than the Laplacian kernel training MSE. That linear regression performed the worst is not unsurprising because it fits a line based on the wine attributes, assuming a linear relationship. However, this is not the best way to predict wine quality (Figure 1). The Gaussian and Laplacian kernels performed better due to the nonlinear nature of the kernels. The curve of best fit generated is thus better. However, 2D CV did not result in the "true" optimal parameters so there is some ambiguity. The test Laplacian MSE is lowest in comparison.

The predicted values range between approximately 5 and 6 (Table 3). If we consider these scores to be for different batches of the same kind of wine, then this would be like quality checking, which is important in food manufacturing.

Method	Training MSE	Test MSE
linear regression	0.628	0.747
Gaussian kernel	0.444	0.678
Laplacian kernel	0.037	0.605

TABLE 2. Training and test MSEs of each method.

Method	Wine 1	Wine 2	Wine 3	Wine 4	Wine 5
linear regression	$6.005 \to 6$	$5.288 \to 5$	$5.564 \to 6$	$6.067 \to 6$	$5.942 \to 6$
Gaussian kernel	$5.988 \to 6$	$5.477 \to 5$	$5.355 \to 5$	$6.163 \to 6$	$6.082 \to 6$
Laplacian kernel	$5.992 \to 6$	$5.524 \to 6$	$5.615 \to 6$	$5.899 \to 6$	$5.941 \to 6$

TABLE 3. The predicted quality of the new batch of wines using the three methods. The rounded values are shown after the arrow.

5. Summary and Conclusions

We developed an algorithm, using linear regression, Gaussian kernel ridge regression, and Laplacian kernel ridge regression, to predict the quality of wine based on physicochemical attributes. For the Gaussian and Laplacian kernels, we used 10-fold CV to determine optimal hyperparameters. Comparing MSEs, the linear regression method performed most poorly, suggesting that assuming a linear relationship between wine attributes and perceived quality is not a good assumption. This is unsurprising given the nonlinear nature of the problem. The Laplacian kernel performed the best, with a training MSE of 0.037 and a test MSE of 0.605. This training MSE is significantly lower than the training MSE from the Gaussian kernel. The test MSEs from the Gaussian and Laplacian kernels are similar. Using predicted wine qualities from the new batch approximately range between 5 and 6. An extension to this study is to use different kernels, such as the polynomial and sigmoid kernels, as well as to use this dataset to test feature selection methods.

Acknowledgements and Code Availability Statement

The author is thankful for the algorithm-implementation suggestions from peers in AMATH 482-582. The code is publicly available on this Github repository after the submission date.

References

[1] Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553, 2009.

[2] Àngela Nebot, Francisco Mugica, and Antoni Escobet. Modeling wine preferences from physicochemical properties using fuzzy techniques. *Proceedings of the 5th International Conference on Simulation and Modeling Methodologies, Technologies and Applications*, 2015.

[3] Bamdad Hosseini. Evaluating sl models. University of Washington-Seattle (LOW 216), Feb 2022. AMATH 482/582.

[4] Kevin P. Murphy. *Machine learning: A probabilistic perspective*. MIT Press, 2021.

[5] 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.

[6] 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.