AMATH 482: Home Work 3

Avinash Joshi
bwaseous@uw.edu
AMATH 482 Wi 22
25 February, 2022

Abstract

Assessing the quality of a wine is a complex and specialized task that critics, vintners, and the wine industry as a whole have extensively studied and mastered. This mastery, however, can become more accurate, objective, and efficient through mathematical techniques. Using kernel methods, cross-validation (CV), and regression, three mathematical critics are developed in order to classify red wines from the Vinho Verde winery in Portugal. The mathematical critics are prepared using three regression methods: least squares, Gaussian and Laplacian kernel ridge. By using ten-fold CV on the training data, the Gaussian and Laplacian kernel hyperparameters, found in Table 1 and graphically represented in Figure 2, are optimized. These three different critics were then tested against known training and test data with their respective mean-squared error (MSE) reported in Table 2. Judging by MSE, the least accurate sommelier was developed using the least squares method while the best critic was trained by the Laplacian kernel. These critics are then employed upon new wine samples with their judgment found in Table 3.

1 Introduction and Overview

The purpose of this assignment is to train three separate critics in classifying 1,115 training and 479 testing red wines from the Vinho Verde winery based on a list of eleven different features. All the wines samples, in both training and testing sets, are scored on a scale from 0-10. Before the critics are trained, however, all the data (features and scores) is standardized by the mean and standard deviations of the training data set of 1,115 wines. This ensures that all the data has mean zero and standard deviation one as this improves the accuracy of all methods, especially kernel methods. The graphical representation of the standardized features can be found in Figure 1. To have a control critic to test against the kernel methods, linear regression is trained on the standardized training data and tested on the 479 testing wine samples.

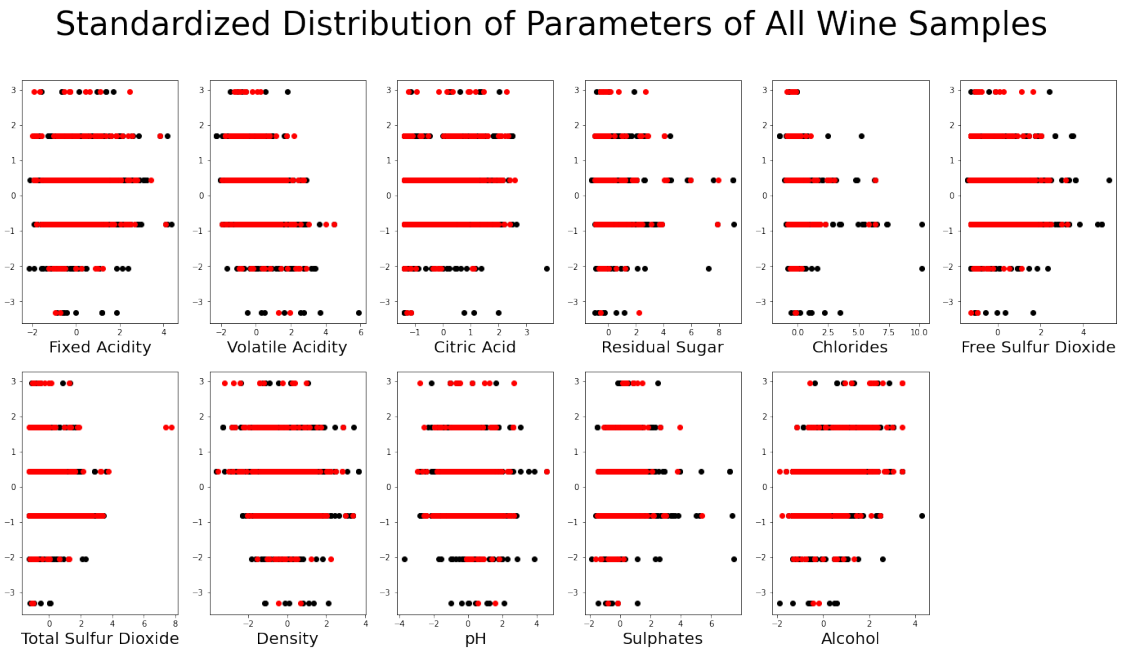

Figure 1: Standardized distribution of wine scores by features with black representing the training samples and red for testing samples.

Moving away from the linear critic, two kernel critics are developed, a Gaussian and a Laplacian, using kernel ridge regression. Kernel methods are employed in order to find nonlinear, exact solutions to the ridge regression problem outlined in the Theoretical Background section. In order to tune the hyperparameters of both the kernel, σ, and ridge regression problem, λ, ten-fold CV optimization is employed. CV is employed twice, first with a broad scale of hyperparameters, and then again on a finer scale.

With the three critics - linear, Gaussian, and Laplacian - trained and evaluted using MSE, they are then tested upon five unseen wine samples. There scores are then converted from the standardized scales back to the original 0-10 scale.

2 Theoretical Background

Linear regression and kernel ridge regression are two similar yet separate ways to train a classifier given a list of features, N data points of said features, and a label or score for each set of features in a given data point. Linear regression relies upon a linear approximation, $A\beta = Y$, where β is the classifier which weighs each of the different parameters of a given column of A, x_i to achieve the desired label, y_i. For this problem, the data matrix X is first standardized so that each feature, or wine parameter, has mean zero and standard deviation one. This "shrinks" the space that the data resides in centering it around the origin. Standardization has many benefits to supervised learning as the linear regression problem no longer relies upon an intercept for β, simplifying the problem from $A\beta = Y$ to

$\overline{X}\beta = \overline{Y}$, where $\overline{X}$ represents the standardized data. This linear, least squares problem is solved by

$$\hat{\beta} = (\overline{X}^T\overline{X})^{-1}\overline{X}^T\overline{Y}.$$

Like linear regression, kernel ridge regression and kernel methods also use standardized data in order to approximate some desired parameter, e.g., the score of a wine sample. Kernel methods, however, go about this problem not by defining features that reproduce the "shape" of the data, but rather produce a kernel that produces necessary features to define the "shape." Using kernels is akin to taking an infinitely paged book and using its pages to make a reproduction of a desired image versus taking a single page from a book and using multiples of that singular page to reproduce an image, as in linear regression and other feature-based analysis. Formally, the kernel, K, is a non-negative, definite, symmetric (NDS) function such that $K : R^N \times R^N \to R$ and has the unique property that $K(x, x') = [F(x), F(x')]_{l^2}$ where $F(x) = \sqrt{\lambda_j}\Psi_j(x)$ from the Eigenvalue Decomposition of K and l^2 is the bounded inner product between $F(x)$ and $F(x')$. Using this and the fact that every kernel has a Reproducing Kernel Hilbert Space (RKHS) or H_K for a given kernel K, the function f which approximates the data, is given by $f(x) = \sum a_j K(x_j, x)$ for some points $x_j \in R^d$. f also has a defined norm over the RKHS, $\|f\|_{H_K} := \sqrt{(\sum a_j^2)}$.

The useful properties of kernels can be fully utilized in kernel ridge regression, where instead of trying to solve $\hat{\beta} = \min_{\beta \in R^n} \|\overline{X}\beta - \overline{Y}\|^2 + \frac{\lambda}{2}\|\beta\|^2$, this can be solved using kernels as

$$\hat{f} = \min_{f \in H_K} \frac{1}{2}\|f(\overline{X}) - \overline{Y}\|^2 + \frac{\lambda}{2}\|f\|_{H_K}^2.$$

This problem can be simplified further using properties of the kernels, specifically the reproducing property of kernels, so that $\hat{a} = (\theta + \lambda I)^{-1}\overline{Y}$ where $\theta = K(x_j, x_j) \in R^{N \times N}$ which by definition is invertible.

Using the kernel perspective for solving ridge regression problems is helpful as the kernel is comprised of infinitely many features which can best describe the data. From here, individual kernels can be descrbied, such as the Gaussian (RBF) kernel and Laplacian kernels:

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

$$K_{lap}(x, x') = \exp(-\frac{\|x - x'\|_1}{\sigma}). \tag{2}$$

To tune the hyperparameters for the kernels, σ, and $\hat{a}$, λ, n-fold cross-validation is preformed. In CV, the training data is partitioned into n-fold partitions, where the classifier is trained on $n-1$ partitions and then tested and evaluated on partition n for varying values of the hyperparameters as defined by $CV(\hat{a}, \lambda) = \frac{1}{N}\sum \|\hat{f}_K(\tilde{X}_k, \hat{a}, \lambda) - \tilde{Y}_k\|^2$ with the optimal $(\hat{a}^*, \lambda^*) = \underset{\hat{a} \in R^N, \lambda \in R}{\operatorname{argmin}} CV(\hat{a}, \lambda)$. $\tilde{Y}_k$ represents the n^{th} partition or testing partition which the classifier is tested upon for accuracy.

The tuned classifiers can now be used to approximate the N scores, $\overline{Y}$, for both the training and testing data sets. To determine the error or accuracy of the classifier, mean squared error (MSE) is employed, where

$$MSE(\hat{f}) = \frac{1}{N}\|\hat{f}(\overline{X}) - \overline{Y}\|^2.$$

3 Algorithm Implementation and Development

Before any analysis is done, the data is loaded using *np.loadtext*[1] with *delimiter = ','*. The data is then split into the appropriate training and testing sets with the score taken as the final column of the data. The data is then normalized around the mean and standard deviation of the training set features and score using *np.mean* and *np.std* along *axis = 0*. Using *np.matlib.repmat* on the number of rows of the data matrix with mean and standard deviation allows the mean to be subtracted from data and then that is divided by the standard deviation.

Following standardization, the hyperparameters for the Gaussian and Laplacian kernel are optimized. Both kernels are initialized without σ or λ using *skl.kernel_ridge.KernelRidge* with *rbf* or *laplacian*. CV is then implemented through scikit-learn's [2] *skl.model_selection.cross_val_score* in a nested for loop that inputs the σ and λ values into the model and evaluates their negative MSE, e.g., a lower MSE correlates to a more accurate performance. The coarser hyperparameter search is generated by *np.linspace* on a scale from $-20 \rightarrow 20$ with ten values for each parameter. These values are representative of powers of two, i.e., 2^n for $n \in [-20, 20]$. This is done for the hyperparameters for both the Gaussian and Laplacian models separately. The mean of each score for each parameter set is then found and plotted on a meshgrid using *plt.contour* to locate a finer scale to rerun the CV on. The lowest CV scores are found using *np.where* and *np.max* which provides indices to pull out the best values of σ and λ. This finds the indices of the lowest value of the MSE which corresponds to the best hyperparameters. Once the finer CV search has concluded, the critics can be trained.

The kernel ridge regression is reinitialized with the finer hyperparameters and then trained using *.fit()* parameter which requires both $\overline{X}_{\text{train}}$ and $\overline{Y}_{\text{train}}$ for both kernels. The critics are then tested using *.predict()* with the $\overline{X}$ - training, testing, and new samples. The MSE is evaluted using *skl.metrics.mean_squared_error()* given the true $\overline{Y}$ then $\overline{Y}_{\text{predict}}$. The least squares critic is trained similarly using *skl.linear_model.LinearRegression.fit()* and scored just the same as the kernel methods. To reconstruct the scores of the wine on the original score scale, the predicted scores are multiplied by the training score STD and then the mean is added back. The scores can the be rounded to whole numbers using *np.round*.

[1] Numpy will be referred to as *np* for convenience

[2] Scikit-learn will be referred to as *skl* for convenience

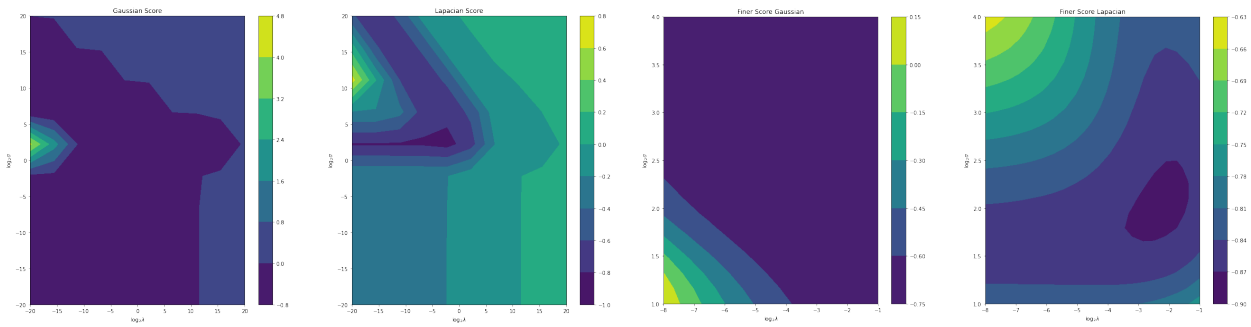

Figure 2: The first set of pictures shows the coarse CV search for the RBF, first, and Laplacian, second. The bluer the areas correlate to better performance. The second figure for both sets of two show the distinguishable local minimum for the Laplacian kernel hyperparameters.

4 Computational Results

After standardizing the data, ten-fold 2D CV was employed to optimize the hyperparameters, σ and λ, for the Gaussian and Laplacian kernels. Their values are reported in Table 1. These hyperparameters are then used in Equation 1 and Equation 2, respectively, to generate the optimal kernels for kernel ridge regression. These parameters, however, are not the globally optimal value, but are simply the locally optimal and most computational efficient values. The Laplacian kernel has a better distinguished minimum compared to the expansive minimum region for the Gaussian kernel. The different methods are then used to predict the training and testing scores and evaluated using MSE. The MSE for each method is provided in Table 2. The new wine samples are then predicted using each method and re-scaled between 0-10, like the original scale. The predicting wine scores of the five samples are reported in Table 3.

Method	σ	λ
RBF	$2^{1.9474}$	$2^{-2.4737}$
LAP	$2^{2.1053}$	$2^{-2.1053}$

Table 1: Fine σ and λ values for the Gaussian (RBF) and Laplacian Kernels

Method	**Training MSE**	**Testing MSE**
Linear	0.6278	0.7472
RBF	0.4589	0.6815
LAP	0.0579	0.6077

Table 2: MSE for all three methods for both training and testing data sets. The Laplacian kernel preformed the best, followed by the Gaussian (RBF), and then least squares.

5 Summary and Conclusions

After the mathematical critics have been initialized and their scores tested, unsurprisingly, the least squares critic preformed the worst. Linear regression only fits a line based upon the wine features which, based upon the individual standardized features in Figure 1, doesn't seem to be the best way of score differentiation. The Gaussian kernel preformed the second worst but still better in both training and testing MSE. This is due to the nonlinear nature of kernels allowing a better curve of best fit to be generated for the nonlinear critics to score with; however, its 2D CV produced ambiguous results with no definitive local minimum, like the Laplacian, resulting in reduced testing accuracy. The Laplacian, on the other hand, had a distinguishable local minimum for its hyperparameters which resulted in an incredibly low training MSE and the best testing MSE. A training MSE of 0.0579 implies that the Laplacian almost exactly fit the training data. All mathematical critics produced similar results for the wine sample, but, based upon the performance given by the MSE, the Laplacian Kernel will produce the best results for feature-based wine scoring, including this new batch. This means that the new batch, starting from the first sample and increasing, were likely a 6, 5, 6, 6, and 6 out of 10. The Gaussian (RBF) critic reports the third sample a 5/10, but, like many of the sample, is because the score hovers between 5.4-5.6 which, when rounded, produces differing results.

Wine #	**Linear**	**RBF**	**LAP**
First	6	6	6
Second	5	5	5
Third	6	5	6
Fourth	6	6	6
Fifth	6	6	6

Table 3: Predicted wine scores of new batch by each method.

To better approximate the scores of the wines, more kernels should be tested in order to find the best method, most accurate results. Also, a more comprehensive and efficient cross-validation scheme should be implemented as 2D cross validation with hundreds of values is computationally intensive and inefficient. For improving overall wine scoring, a more robust grading system, such as a score out of 100, will produce better, more accurate results. This grading system should also include a range of values given by multiple human and mathematical critics to better represent the wine's quality, too.

6 Acknowledgements

The author would like to graciously thank his classmates, especially Nalu Zou, for allowing him to discuss the problem with them as a means to help them and test his own understanding of the material.

The author would also like to thank Professor Hosseini for his demonstration of CV and kernel methods which served as the foundations for their implementation in this assignment.