# AMATH 482/582 HOMEWORK 3 - QUALIFYING RED WINE

PHILIP WESTPHAL

*Department of Applied Mathematics, University of Washington, Seattle, WA*
*philw16@uw.edu*

Abstract. In this report, the performance of Kernel Ridge Regression (KRR) models versus a linear regression model was explored using a real world wine quality dataset. Two KRR models were built using Laplacian and Gaussian (RBF) kernels, and hyperparameter tuning was performed to optimize their input parameters. The Mean Square Error (MSE) on the training and test datasets were compared, as well as the predictions of each model on unlabeled data.

## 1. Introduction and Overview

The goal of this problem was to measure the performance of Kernel Ridge Regression (KRR) methods on a Portuguese "Vinho Verde" red wine quality dataset against a least squares linear regression models. A training dataset $X_{train} \in \mathbb{R}^{1115,12}$ and a test set $X_{test} \in \mathbb{R}^{479,12}$ were provided for model training and validation. The first eleven columns represent values for physical features of the wine such as residual sugars, chlorides, density, and alcohol percentage. The quality rating of the wine was the final column of each dataset, which were broken off to give $y_{train}$ and $y_{test}$. A third dataset was also provided that contained only the eleven physical features of five wine samples for the models to predict the quality of. This simulates production data being input for predictions as would be done in a real world setting.

Using kernel methods to solve optimization problems is advantageous as they allow feature maps to be calculated that are of a higher dimensional space than the problem [1]. This is helpful as data that cannot be linearly separated in the problem space may be linearly separable in a higher dimensional space [2]. There also exists a "kernel trick" utilizing the reproducing property of equation 7 [1] which allows the higher dimensional kernel values to be obtained in the problem space without actually transforming the data [2]. As a result, non-linear problems in can be made linear in higher dimensions without applying computationally expensive spatial transformations.

## 2. Theoretical Background

### 2.1. Kernel Theory

A kernel is defined as a mapping function $K: \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ that maps higher dimensional vectors to a single value [3]. For a set of input data $X \in \mathbb{R}^n \times \mathbb{R}^n$, a kernel matrix $\Theta$ can be written with elements representing the kernel evaluated at all combinations of columns

$$\Theta \in \mathbb{R}^n \times \mathbb{R}^n, \ \Theta_{ij} = K(X_i, X_j) \tag{1}$$

A kernel is said to be non-negative definite (NDS) and symmetric if

$$K(X, X') = K(X', X) \ \ \forall \ X, X' \in \mathbb{R}^n$$

For NDS kernels with a bounded integral over $\mathbb{R}^n$, Mercer's Theorem tells us that there exists an orthonormal basis of a linear operator related to $K$ in $L^2$ hilbert space [4] such that

$$K(X, X') = \sum_{j=0}^{\infty} \lambda_j \psi_j(X) \psi_j(X') \tag{2}$$

where $\lambda_j$ represent the eigenvalues and $\psi_j$ represent the eigenfunctions of the linear operator. This is similar to eigendecomposition of NDS matrices, however a major difference is that there can be infinitely many

*Date*: February 26, 2022.

terms $j$ in the kernel representation versus a finite number from matrix decomposition [1]. From equation 2, a feature map representation $F(X)$ of the kernel $K$ can be defined on space $l^2$ as

$$F(X) = (F_0(X), F_1(X), \dots, F_\infty(X)) \in l^2,\ F_j(X) = \sqrt{\lambda_j}\psi_j(X) \tag{3}$$

$$l^2 := \left\{ \{c_j\}_{j=0}^{\infty} \;\middle|\; c_j \in \mathbb{R}\ \&\ \sum_{j=0}^{\infty} |c_j|^2 < \infty \right\},\ < c_j, b_j >_{l^2} = \sum_{j=0}^{\infty} c_j b_j$$

The kernel can be thus be written using the $l^2$ inner product of the feature maps

$$K(X, X') = \langle F(X), F(X') \rangle_{l^2} = \sum_{j=0}^{\infty} F_j(X) F_j(X') \tag{4}$$

For an NDS kernel $K$, the Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}_k$ is a set of functions defined as

$$\mathcal{H}_k : \left\{ f : \mathbb{R}^n \to \mathbb{R} \;\middle|\; f(X) = \sum_{j=0}^{\infty} c_j F_j(X),\ \sum_{j=0}^{\infty} c_j^2 < \infty \right\} \tag{5}$$

which have a corresponding $\mathcal{H}_k$ norm of

$$\|f\|_{\mathcal{H}_k} = \sqrt{\sum_{j=0}^{\infty} c_j^2} \tag{6}$$

For every RKHS a reproducing kernel $K$ is guaranteed to exist [5], for which at point $X^* \in \mathbb{R}^n$ and function $f \in \mathcal{H}_k$

$$f(X^*) = \langle f(X), K(X^*, \cdot) \rangle \tag{7}$$

Every function $f \in \mathcal{H}_k$ can also be written in kernel form with coefficients $a \in \mathbb{R}$ and $X_j \in \mathbb{R}^d$ [3]

$$f(X) = \sum_{j=0}^{\infty} a_j K(X_j, X) \tag{8}$$

### 2.2. **Kernel Ridge Regression**

The formula for linear ridge regression was derived in [6] as

$$\beta_{MLE} = \underset{\beta}{\operatorname{argmin}} \frac{1}{2\sigma^2} \left\| A\beta - y \right\|^2 + \frac{\lambda}{2} \|\beta\|^2 \tag{9}$$

Considering $A$ as a matrix of feature maps $F$, a model of the following form can be written

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

where $f$ is in the RKHS space of a kernel $K$ by equation 5. The squared RKHS norm of $f$ will be given as

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

By comparison, the ridge regression problem of equation 9 can thus be written as the following for a function $f \in \mathcal{H}_k$

$$f_{opt} = \underset{f \in \mathcal{H}_k}{\operatorname{argmin}} \|f(X) - y\|^2 + \lambda \|f\|_{\mathcal{H}_k}^2 \tag{10}$$

The solutions to 10 are expressed using a representation of $f$ with a finite number of features $J$ [1]. The kernel form of 10 can also be considered, which is advantageous due to having the increased modeling flexibility of potentially infinite features. By equation 8 a function $f$ can be written as an infinite sum of kernel terms, but this is not something that can be explicitly computed. However, a fundamental result of a family of theorems known as the Representer Theorems says the solution to this problem exists with a finite number

of dimensions [3]. The Representer Theorems say that a function $\hat{f} \in \mathcal{H}_k$ is a solution of equation 10 if and only if $\hat{f}$ is of the form

$$\hat{f}(X) = \sum_{n=0}^{N-1} \hat{a} K(X_n, X) \tag{11}$$

with the coefficient vector $\hat{a} \in \mathbb{R}^N$ being a minimizer of the following [3]

$$\hat{a} = \underset{a \in \mathbb{R}^N}{\operatorname{argmin}} \|\Theta a - y\|^2 + \lambda a^T \Theta a \tag{12}$$

Equation 12 is the Kernel Ridge Regression (KRR) problem, where the matrix $\Theta$ is the kernel matrix from equation 1. Two common kernels that are used in KRR are the Gaussian (RBF) and Laplacian kernels

$$K_{RBF}(X, X') = \exp\left(\frac{-\|X - X'\|_2^2}{2\sigma^2}\right) \qquad K_{lap}(X, X') = \exp\left(\frac{-\|X - X'\|_1}{\sigma}\right) \tag{13}$$

The mean square error formula defined in [6] can be modified to use the kernel matrix $\Theta$ and coefficients $\hat{a}$ in order to measure the performance of KRR models

$$MSE = \frac{1}{\operatorname{length} y} \|\Theta \hat{a} - y\|_2^2 \tag{14}$$

The KRR problem is very sensitive to changes in the hyperparameter values $\sigma$ from equation 13 and $\lambda$ from equation 12. As a result, cross validation as described in [6] will be required to find optimal values. Cross validation breaks the training dataset into a specified number of pieces, then trains the model repeatedly with each piece as the test set once. The optimal parameters are those for which the total $MSE$ is lowest when aggregated over all runs. In this case there are two hyperparameters that need to be optimized, and thus a two dimensional search will need to be performed

$$\lambda_{opt}, \sigma_{opt} = \underset{\lambda, \sigma \in \mathbb{R}}{\operatorname{argmin}} \sum_{k=0}^{K-1} MSE(X_{-k}, \lambda, \sigma) \tag{15}$$

Given computational limits the true optimum values of the hyperparameters may not be reached, however smart searching strategies can find values yielding high performing models.

## 3. Algorithm Implementation and Development

The KRR problem of equation 12 was constructed using `KernelRidge()` models from the `sklearn` package with both the RBF and Laplacian kernels [7]. A `LinearRegression()` model from `sklearn` was also built to compare the KRR models against. Algorithm 1 outlines the main function used for model fitting and evaluation.

The `GridSearchCV()` and `RandomSearchCV()` functions from `sklearn` use cross validation to optimize parameters over a certain range. The random search randomly samples from each parameter range provided for a given number of iterations, while the grid search will try every possible combination of the parameters. For optimizing $\lambda$ and $\sigma$ in this problem, a random search was performed across a wide parameter range (200 iterations) before a grid search (20x20 grid) was performed within $\pm 30\%$ of the optimal random search values. Ten-fold cross validation was used for the random and grid searches to evaluate each parameter combination, which is taking $K = 10$ in equation 15. Algorithm 2 outlines the workflow used for loading the models and performing the cross validation to optimize parameters. The KRR problem is very sensitive to the scale of both the inputs values and the targets, so a `StandardScaler()` object was used to scale the data to mean zero and standard deviation one [7].

## 4. Computational Results

Table 1 shows the training and test set MSE for the initial base parameter models outlined in algorithm 2. Both KRR models had lower MSE than the base linear regression model on the training and testing set, with the Laplacian kernel performing slightly better that the Gaussian kernel. Given that the KRR models performed better than the linear model, it suggests the the wine data may not be linearly separable in the original feature space but is with a higher dimensional feature map.

**Algorithm 1** Run Regression Model

**function** RUNWINEMODEL(train, target, scaler, model, search, param grid, prefit, cv params)
  $X_{train}, y_{train}$ = train data
  $X_{test}, y_{test}$ = scaler transform(target) ▷ Scale the target
  **if** param grid **then** ▷ Run search if we are performing parameter tuning
    **if** search = grid **then**
      model = `GridSearchCV`(param grid, cv params)
    **else if** search = random **then**
      model = `RandomSearchCV`(param grid, cv params)
    **end if**
  **else**
    model = base model
  **end if**
  **if** not prefit **then** ▷ Fit model if it isn't prefit
    model fit($X_{train}, y_{train}$)
  **end if**
  $y_{train\ pred},\ y_{test\ pred}$ = model predict($X_{train}, y_{train}$), model predict($X_{train}, y_{train}$)
  $MSE_{train},\ MSE_{test}$ = MSE($y_{train}, y_{train\ pred}$), MSE($y_{test}, y_{test\ pred}$)
  **return** $MSE_{train},\ MSE_{test}$, model
**end function**

**Algorithm 2** Run & Evaluate Regression Models

$X_{train}, y_{train}$ = `StandardScaler()` fit and transform of training data
linear, gauss, lap = `LinearRegression()`, `KernelRidge('rbf')`, `KernelRidge('laplacian')`
Evaluate base models for initial results using algorithm 1
lambdas, sigmas = $2^n,\ n = [-10, ..., 10] \in \mathbb{R}^{1000}$ ▷ Initial parameter range
gamma gauss = 1 / 2∗sigmas$^2$, gamma lap = 1 / sigmas ▷ `sklearn` syntax
param grid = {lambdas, gammas}, search = random
rand gauss, rand lap = run algorithm 1 with random search parameters
param grid = ±30% of optimal random search results, search = grid
grid gauss, grid lap = run algorithm 1 with grid search parameters
Predict quality of new wine data using linear, best gaussian, and best laplacian model

| MSE | Linear Regression | Gaussian KRR | Laplacian KRR |
|---|---|---|---|
| Traning Set | 0.6278 | 0.4475 | 0.3586 |
| Test Set | 0.7472 | 0.6666 | 0.6480 |

TABLE 1. Training & Test MSE For Base Models

Figure 1 visualizes the results of the random search performed with the KRR models as described in algorithm 2. The red stars on the plot signify the best performing model (by average of equation 15) found during cross validation. The ±30% range for the best random search parameters allows searching through the optimal contours in figure 1 with the grid search. Table 2 shows the training and test MSE for the optimal models (which can be compared against table 1), along with the corresponding $\lambda$ and $\sigma$ values.

| KRR Kernel Used | Parameter Values | Training MSE | Test MSE |
|---|---|---|---|
| Gaussian | $\lambda = 0.7118,\ \sigma = 2.6233$ | 0.4491 | 0.6669 |
| Laplacian | $\lambda = 0.2888,\ \sigma = 5.9487$ | 0.1076 | 0.6147 |

TABLE 2. Training & Test MSE Performance of Optimal Random Search Models

It can be seen that while the training and test MSE improved for the Laplacian KRR model, the Gaussian KRR model performed worse after the search. An additional random search was thus performed for the

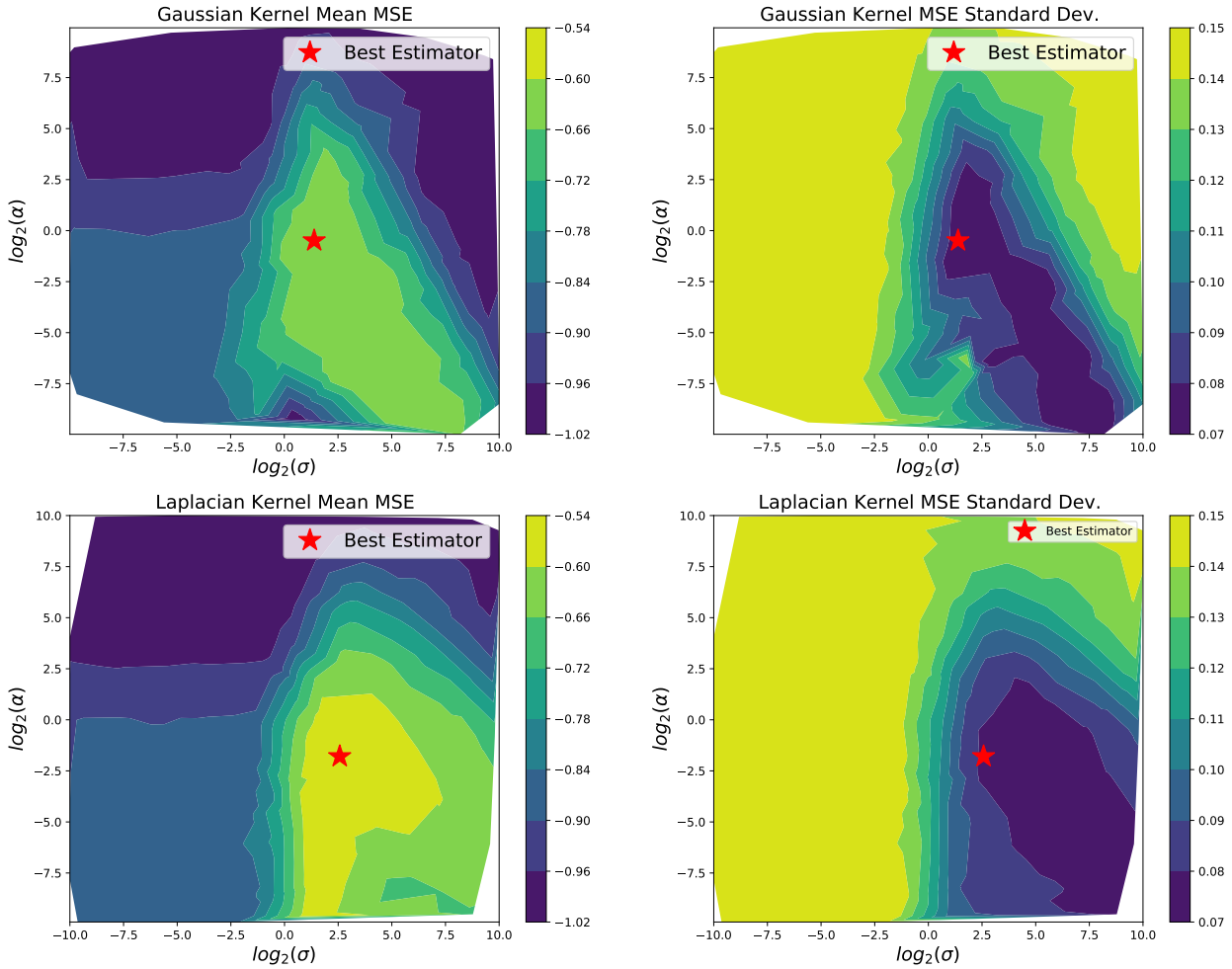


FIGURE 1. MSE Contours of KRR Random Search Models

Gaussian model using the default $\sigma$ value and searching over $\alpha$. The better performer between this and the initial random search were the center parameters used for the Gaussian KRR grid search. Table 3 shows the MSE results of the additional random search for the Gaussian kernel and the grid searches for each kernel.

The data from table 3 show that the best estimator from the one dimensional random search outperformed

| KRR Kernel Used | Parameter Values | Training MSE | Test MSE |
|---|---|---|---|
| Random 1D-Gaussian | $\alpha = 0.5603,\ \sigma = 2.2361$ | 0.4068 | 0.6639 |
| Gaussian KRR | $\lambda = 0.4630,\ \sigma = 2.6726$ | 0.4266 | 0.6653 |
| Laplacian KRR | $\lambda = 0.2387,\ \sigma = 4.5759$ | 0.06448 | 0.6088 |

TABLE 3. Training & Test MSE of 1D Random Search/Grid Search Models

that of the two dimensional random search for the Gaussian KRR model. In fact, the subsequent grid search produced a slightly worse model then the one dimensional random search as well. Considering only the test set MSE of the best performing models, the Laplacian KRR is the best performing, followed by the Guassian KRR, and finally the Linear Regression. These results are shown in table 4. However, the data also show a very large discrepancy between the training MSE and test MSE for the Laplacian KRR model. The fact that the test MSE is significantly higher than the training MSE suggests that the model might be overfitting the training data.

| Model | Parameter Values | Training MSE | Test MSE |
|---|---|---|---|
| Linear Regression | | 0.6278 | 0.7472 |
| Gaussian KRR | $\alpha = 0.5603,\ \sigma = 2.2361$ | 0.4068 | 0.6639 |
| Laplacian KRR | $\lambda = 0.2387,\ \sigma = 4.5759$ | 0.06448 | 0.6088 |

TABLE 4. Training & Test MSE of Best Performing Models

Table 5 shows the raw and rounded quality predictions for the unlabeled wine samples using best performing models. The data show all three models agree on the rounded predictions, although have differing raw predictions.

| Model | Parameter Values | Raw Pred | Rounded Pred |
|---|---|---|---|
| Linear Regression | | $6.005, 5.288, 5.564, 6.067, 5.942$ | $6, 5, 6, 6, 6$ |
| Gaussian KRR | $\lambda = 0.5603,\ \sigma = 2.2361$ | $6.114, 5.498, 5.538, 6.195, 6.195$ | $6, 5, 6, 6, 6$ |
| Laplacian KRR | $\lambda = 0.2387,\ \sigma = 4.5759$ | $6.06, 5.462, 5.627, 5.992, 6.024$ | $6, 5, 6, 6, 6$ |

TABLE 5. New Data Predictions for Best Performing Models

## 5. Summary and Conclusions

In this problem, KRR models were trained using Gaussian and Laplacian kernels to predict wine quality given input characteristics. The KRR models were optimized by using ten-fold cross validation to tune the hyperparameters $\lambda$ and $\sigma$. The training set MSE, test set MSE, and quality predictions for five unlabeled wines were calculated using the optimized KRR models and a standard linear regression model. While all three models predicted the same rounded quality for the unlabeled wine, the KRR models had lower train and test set MSE than the linear regression model. Additionally, while the Laplacian KRR model outperformed the Gaussian KRR model in test MSE, it showed signs of overfitting the training data and thus may not be the optimal model to use on production data. In future experiments, it would be interesting to try different kernels on the data to see the impact on the MSE values. It could also be seen if other kernels used overfit the data as well, and methods for reducing the overfitting could be explored.

## Acknowledgements

The author would like to thank Professor Bamdad Hosseini for covering the background concepts of Kernel theory and Kernel Ridge Regression in lecture, as well as the other classmates in Discord who assisted with troubleshooting cross validation results. In addition to sklearn, the numpy [8] and matplotlib [9] packages in python were also extremely useful for processing data and generating visualizations.

## References

[1] B. Hosseini, "Introduction to kernel methods," University of Washington, Feb 2022, aMATH 482/582.
[2] V. Srikumar, "Kernels and the kernel trick." [Online]. Available: https://svivek.com/teaching/lectures/slides/svm/kernels.pdf
[3] B. Hosseini, "Kernel ridge regression," University of Washington, Feb 2022, aMATH 482/582.
[4] [Online]. Available: https://mathworld.wolfram.com/L2-Space.html
[5] D. Sejdinovic and A. Gretton, "What is an rkhs?" Mar 2014. [Online]. Available: https://www.stats.ox.ac.uk/~sejdinov/teaching/atml14/Theory_2014.pdf
[6] P. Westphal, "Amath 482/582 homework 2 - classifying digits," Feb 2022.
[7] 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*, vol. 12, pp. 2825–2830, 2011.
[8] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2
[9] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.