QUALIFYING RED WINE

XUECHENG LIU

Department of Applied Mathematics, University of Washington, Seattle, WA
xl0306@uw.edu

Abstract. Kernel methods is a useful technique for feature extraction over the training data. Generally speaking, kernel methods maps the original data into higher dimension to help the learning algorithm better understands the data in a more accurate representation through the Kernel function. Cross validation is an essential tool for hyper parameter tuning by splitting the training data into K folds and figure out the best hyper parameter.

1. Introduction and Overview

This project develops a supervised learning model about how to evaluate the quality of red wines on a scale of 0 to 10 based on 11 chemical features contained in the wine. The model applies linear regression and combines kernel tricks with ridge regression during the training stage. 2D cross-validation is applied to the kernel trick for hyper parameter tuning. The predictions on the new batch of red wine are compared across 3 models and mechanics behind each model is discussed.

2. Theoretical Background

A function $\mathbf{K} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$is called a Kernel. We also define K is non-negative definite & symmetric (NDS) if

$$\mathbf{K}(\mathbf{x}, \mathbf{x}^{'}) = \mathbf{K}(\mathbf{x}^{'}, \mathbf{x}) \ \forall \mathbf{x}, \mathbf{x}^{'} \in \mathbb{R}^{\mathbf{n}} \ and$$
$$(K_{ij}) = \mathbf{K}(\mathbf{x_i}, \mathbf{x_j}) \in \mathbb{R}^{M \times M} \ is \ NDS$$

In other words, we maps the n-dimensional data into M dimensional in order to get a more accurate representation of the features.

For example, Gaussian Kernel function is defined as $\mathbf{K}_{rbf}(x, x^{'}) = exp(-\frac{\|x - x^{'}\|^2}{2\sigma^2})$.
According to [Brunton and Kutz, 2019], by Mercer Theorem,

$$\mathbf{K}(\mathbf{x_i}, \mathbf{x_j}) = \sum_{j=0}^{\infty} F_j(x) F_j(x^{'})$$

where $F_j : \mathbb{R}^n \to \mathbb{R}$ are features of K. Given an NDS Kernel K, we also define its reproducing Kernel Hilbert space(RKHS) as the space of functions $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}$. We denote this space with $\mathcal{H}_f$.

According to [Brunton and Kutz, 2019], in ridge regression, $\hat{\beta}$ is defined as

$$\hat{\beta} = \underset{\beta \in \mathbb{R}^J}{\operatorname{argmin}} \frac{1}{2\sigma^2} \|A\beta - Y\|^2 - \frac{\lambda}{2} \|\beta\|_p^p$$

λ is called the penalty parameter and $\hat{\beta}$ can be solved as

$$\hat{\beta} = (A^T A + \sigma^2 \lambda I)^{-1} A^T Y$$

Date: February 25, 2022.

Incorporating the idea of kernel, substituting the design matrix A as feature matrix of the form

$$\begin{bmatrix} F_0(x_0) & F_1(x_1) & ... \\ F_0(x_1 & F_1(x_1) & ... \\ ... & & \\ F_0(x_N) & F_1(x_N), & ... \end{bmatrix},$$

.

We could also define kernel ridge regression to be

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

and solve accordingly.

K-fold cross validation is a powerful tool for hyper parameter tuning. The intuitive idea is to split the training set into K subset and pick the first K-1 subsets to train K-1 models where each model is equipped with a different hyper parameter, such as λ in ridge regression. We then define the CV prediction error as $\frac{1}{N}\sum_{k=0}^{K-1}\|f_K(X_K) - Y_K\|^2$ and use the Kth subset as the validation set to test. K-fold cross validation can be extended to two dimensions using double for loops and the next section will illustrate how to implement this idea.

3. Algorithm Implementation and Development

The main packages used in this project is **Numpy** and **Sklearn**.The following algorithm illustrate how to normalize the data.

Algorithm 1 Feature Normalization

```
import train_data
import test_data
X_train ← train_data[:, 0 : 11]
X_test ← test_data[:, 0 : 11]
X_train_mean ← mean(X_train)
X_train_std ← std(X_train)
X_train_normal ← (X_train − X_train_mean) / std(X_train)
X_test_normal ← (X_test − X_train_mean) / std(X_train)
```

The response variable Y can be normalized in the same way as illustrated above.The following algorithm illustrate how to do 2D cross validation.

Algorithm 2 2D Cross Validation

```
step ← K
scores ← zeros(step, step)
σ ← linsapce(a, b)
λ ← linsapce(c, d)
for i in range(K) do
    for j in range(K) do
        score ← cv_score(σ[i], λ[j], X_train_normal, Y_train_normal)
        scores[i, j] ← score
    end for
end for
return scores = scores.max()
```

4. Computational Results

Due to the limitation in computation power, finding the best hyper parameter is through brute force search over a large range will take too much time. In order to save computation power, I first searched over a wide range with a few values (small step size) and plot the contour plot v.s the corresponding hyper parameters. Based on the contour plot, we shrink the range and do the search again. This process is iterate 3 times to find a relatively good pair of hyper parameter. Each time we refine the range of the search and decrease the step size successively as well as including the best pair from the last step.
Figure 1 shows the contour plots of successively narrowed hyper parameter ranges for Gaussian Kernel Ridge Regression.

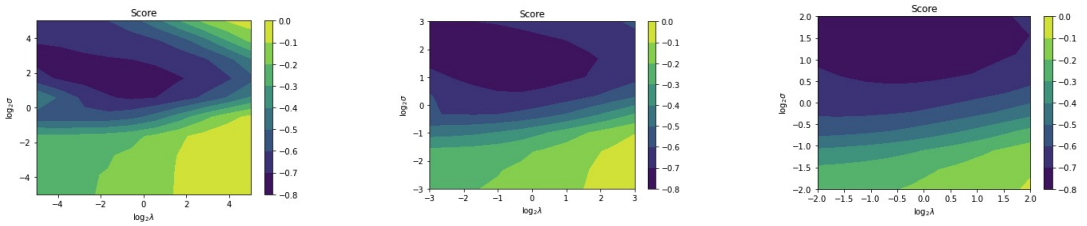

Figure 1. Contour plots for Gaussian Kernel Ridge Regression

The optimal value for λ is $2^{-1.556}$ and for σ is $2^{1.5554}$ based on refined search.
Figure 2 shows the contour plots of successively narrowed hyper parameter ranges for Laplacian Kernel Ridge Regression.

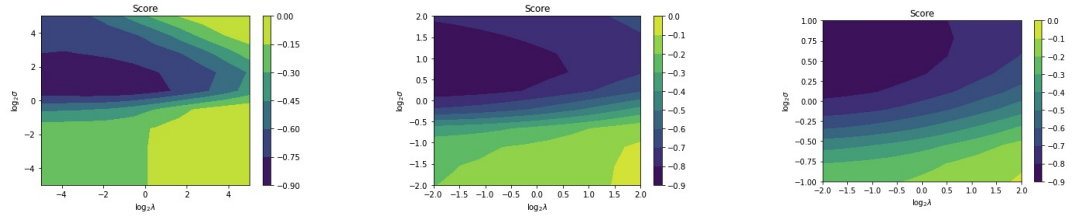

Figure 2. Contour plots for Laplacian Kernel Ridge Regression

The optimal value for λ is 2^{-2} and for σ is $2^{0.5554}$ based on refined search.

From the above results, we could fit the model using these hyper parameters. Table 1 shows the training and testing mean squared errors of all three models.

	Training MSE	Testing MSE
linear regression	0.6278	0.7472
RBF ridge	0.4309	0.6675
Laplacian ridge	0.0635	0.6076

Table 1. Training and Testing MSE of all three models

As we can see from the results above, Laplacian kernel did the best job in generalizing the data with the lowest testing MSE. However, it also has the lowest training MSE which seems like over fitting the data. Linear regression works the worst and it seems that linear model is not capable to capture the trend of the data.

Using the model fitted above, table 2 shows the prediction of the new batch of wines on the 0-10 scale with rounded values.

	Wine Quality
linear regression	[6,5,6,6,6]
RBF ridge	[6,5,5,6,6]
Laplacian ridge	[6,5,6,6,6]

TABLE 2. Predictions of new batch of wines

5. Summary and Conclusions

In summary, we could find that kernel trick have a significant impact in model fitting with a lower MSE in both training and testing phase compared with traditional linear model. However, in the practical sense, the differences are relatively not obvious compared with linear model. This leads us to the thinking of whether it is worth to devote huge amount computation power as a trade off to a tiny change in the real world results.

Acknowledgements

The author is thankful to Prof. Bamdad Hosseini for providing excellent lectures and well-formatted lecture notes as well as providing clean start code.

References

[Brunton and Kutz, 2019] Brunton, S. and Kutz, J. (2019). *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press.