Kernel Regression

Yanhui Zhu* Sandesh Thapa*

April 14, 2021

1 Introduction

Kernel regression is a non-parametric estimator that estimates the conditional expectation of two variables which is random. The goal of a kernel regression is to discover the non-linear relationship between two random variables. To discover the non-linear relationship, kernel estimator or kernel smoothing is the main method to estimate the curve for non-parametric statistics. In kernel estimator, weight function is known as kernel function [8]. With the help of kernel estimators, we calculate the local average or a moving average for response [8]. In 1964, the Nadaraya-Watson kernel estimator was invented. Since then, in statistics, the subject of non-parametric kernel regression estimation has been extensively investigated [5].

Kernel estimator has made a significant impact in the past. In 1964, with the development of the Watson estimator, it is applied for few applications such as to develop graphical plots between blood pressure and age, and regression between the temperature and previous day [5]. In 1979, Schuster and Yakowitz used kernel estimator on noisy measurement to classify time-varying linear systems [5]. In 1985, Miller addressed the principle of using higher-order kernels for derivative estimation and used it in the analysis of the hypophyseal hormone regression curve toward the puberty era [5]. In 1984, the analysis of human growth features proposed by Gasser et al. is a noteworthy use of kernel regression in the field of biostatistics [5].

In recent years, kernel regression finds a non-linear relation between a pair of random variables and response variables. Support vector ma-

*Equal contribution.

chines(SVM) outperform other classification and regression methods for various data. However, the performance of SVM is greatly affected by the choice of a kernel function among other factors. Thus, it is essential to fine tune the kernel parameters for a specific application scenarios. SVM was first proposed to solve the binary classification tasks [6, 13]. Afterward, researchers have paid much attention on it[3, 16]. Now, it has become a well established supervised learning method both for classification and regression. Kernel methods of SVMs are regarded as the most prominent feature. Kernel SVMs have achieved good performance on high dimensional data sets [12].

SVM models a classification problem as a maximum margin optimization problem, where the decision boundary that has the largest margin to separate the training points of different classes is searched. The values of the kernel hyperparameters of SVMs are typically determined by cross validation on a grid of candidate values [17]. This is a rough process, because there is no way to fine-tune the results, so we can only roughly approximate the optimal value. In other words, this is the case of blind search, because the previous possible kernel hyperparameter test information was not used to facilitate the search. RBF kernels can be regarded as a restricted version of general Gaussian ones, where the Gaussian matrix is constrained to be the unit matrix multiplied by a scale factor. In particular, the correct tuning of the hyperparameter γ of the RBF kernel is essential for SVM classification/regression performance [18, 11].

This report presents a comparative study of different kernel functions for a commonly used two data sets Iris species and Wine quality from UCI Machine Learning Data Repository. We illustrate the optimal parameter choice by designing experiments. In addition, we also compare the results with linear regression models. Consequently, kernel SVM has a better performance on this data. In this report, we fine tune the parameters of kerel functions and SVM heperparameter using R programming language on Iris species and wine quality data set. The structure of this paper is as follows. First, the proposed methodology for SVM and kernel functions is detailed in Section 2. Then the experimental design and the results and comparisons are reported in Section 3.

2 Methodology

2.1 Nadarayan-Watson Kernel estimator

The common non-parametric regression model is $Y_i = m(X_i) + \epsilon_i$, where Y_i can be defined as the sum of the regression function value $m(x)$ for X_i. Here m(x) is unknown and some errors where the predicted error value is zero [10]. With the help of this definition, we can create the estimation for local averaging i.e. m(x) can be estimated with the product of Y_i average and X_i is near to x [10]. In other words, this means that we are discovering the line through the data points with the help of surrounding data points. The estimation formula is printed below[10]

$$M_n(x) = \sum_{i+1}^{n} W_{n,i}\,(X).Y_i \tag{1}$$

$W_{n,i}\,(x)$ is the sum of weights that belongs to all real numbers. Weights are positive numbers and small if X_i is far from x [10]. In the Nadaraya-Watson kernel estimator formula [10]

$$M_n(x) = \frac{\sum_{i=1}^{n} k(\frac{x-X_i}{h})Y_i}{\sum_{i=1}^{n} K(\frac{x-X_i}{h})} \tag{2}$$

$k(\cdot)$ is the kernel function that uses the weighted average of Y_i [8]. The average weight of y_i is calculated from the distance between X_i and x [8]. 'h' is a bandwidth which is h > 0. As a result, the formula of weight is [10]

$$W_{n,i}\,(x) = \frac{K(\frac{x-X_i}{h})}{\sum_{i=1}^{n} K(\frac{x-X_i}{h})} \tag{3}$$

The weight $W_{n,i}$ can be modeled by the simple approximation of the kernel smoothing process. With the help of Nadaraya-Watson kernel estimator, we are discovering the smooth curve in a scatter plot for the response variable and predictors. The estimation of the smoothness for the curve is regulated by the bandwidth. Bandwidth controls the number of data points to consider for the weighted average.

2.2 Support Vector Machines

Given a set of N training samples denoted as $\mathbf{X} = \{\mathbf{x_i} \in \mathbf{R}^{n \times n}, i = 1, ..., N\}$, and the ground-truth vector $\mathbf{y}$. In general, the regression/ classification model is formulated as

$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b \tag{4}$$

where ϕ is the feature transformation function and b is the bias of the model. The objective function for maximize the margin between different kinds of labels of SVM models is:

$$\arg\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2, \quad s.t. : y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b > 1 \tag{5}$$

Then, the Lagrange multipliers can convert the equation 5 to a maximization problem with respect to α [2]. In that case, the optimization problem can be represented by the formula:

$$\max_{\alpha} L_\gamma(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \tag{6}$$

This equation 6 is subject to $\sum_{i=1}^{N} \alpha_i y_i = 1$ and α_i is non-negative and no greater than a parameter C. Note that the selection of kernel functions $k(x_i, x_j)$ can influence the performance. Thus, in the next section, we demonstrate four kind of commonly used kernel functions in R.

2.3 Kernel Functions

The kernel function returns the inner product between two points in the appropriate feature space. Therefore, by defining the concept of similarity, even in a very high space, only a small amount of calculation is required [1]. There are four kind if kernels built in R, e.g., linear kernel, polynomial kernel, gaussian kernel and radial basis function (RBF) kernel, and sigmoid kernel. The most used type of kernel function is RBF. Because it has localized and finite response along the entire x-axis. There corresponding mathematical equations are illustrated below.

2.3.1 Linear Kernel

Linear Kernel is the inner product of two vectors.

$$k(x_i, x_j) = x_i \cdot x_j + 1 \tag{7}$$

There is no parameter to optimize.

2.3.2 Polynomial Kernel

Polynomial Kernel is widely used for image processing tasks.

$$k(x_i, x_j) = (x_i \cdot x_j + 1)^d \tag{8}$$

where d is the degree of the polynomial.

2.3.3 Gaussian Kernel and RBF Kernel

Gaussian kernel is a general-purpose kernel, which is used when there is no prior knowledge about the data.

$$k(x_i, x_j) = \exp(-\frac{||x_i - x_j||^2}{2\sigma^2}) \tag{9}$$

where $||x_i - x_j||^2$ is can be recognized as the squared Euclidean distance between the two feature vectors. The free parameter is σ and if we generalize the $\gamma = \frac{1}{2\sigma^2}$, then the gaussian kernel becomes the RBF kernel form:

$$k(x_i, x_j) = \exp(-\gamma||x_i - x_j||^2) \tag{10}$$

Here, γ is the essential parameter when we are optimizing kernel SVM.

2.3.4 Sigmoid Kernel

$$k(x_i, x_j) = \tanh(\gamma \cdot x_i x_j + r) \tag{11}$$

where γ and r are parameters to optimize.

3 Experiments

3.1 Experiment Settings and Ideas

For Nadarayan-Watson Kernel estimator, the selection of parameters is vital as the estimator relies on smoothing parameters [10]. So, the question may arise to determine the smoothness of the curve. In the bias-variance trade-off, when the value of bandwidth is greater, the bias increases by including more data points for weighted average. This gives smoother curve

and decrease in variance. The bandwidth is only dependent on the sample size N. The optimal value of bandwidth gets smaller as sample size increases. One of the method use in this paper to obtain optimal bandwidth is k-fold cross validation. It is mostly used method. The steps for k-fold cross validation is describe in next section while obtaining the optimal bandwidth for stimulation data.

For kernel SVM, we design experiments on kernel SVM models using R programming language, where the linear kernel, polynomial kernel, radial basis function (RBF) kernel, and sigmoid kernel are all tested on some popular data sets. Next, we will find out the optimal parameter for each kernel method excluding the linear kernel since there is no parameter for this method. Afterward, we compare them with the accuracy of the testing set and conclude the best kernel method and its corresponding parameter for each data set. To show the advantage of kernel SVM, we also compute the traditional linear regression method results.

3.2 Data sets preparation

We plan to test the kernel methods on four data sets. The first two datasets are for Nadaraya-watson estimator and last two datasets are for SVM method. The descriptions for them are listed as follows.

- Simulation data [4]. The simulation data consists of Nadaraya-Watson formula [15] from scratch, simulation data. For the simulation, the 200 data are generated randomly following the example of author Yen-Chi Chen [4]. The data has one predictor and one response variable. The regression function is $m(X) = sin(x)$.

- Old Faithful Geyser [14]. The predictor X is the length of eruption time in minutes and the response variable Y is the waiting time until the next eruption occurs [14]. The assumption is that the response variable depends on predictors X. The `ksmooth()` function in R helps to perform Nadaraya-Watson estimator.

- Iris Data Set: The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. There are four features for this data set, sepal length, sepal width, petal length, petal width, respectively. Our goal is to predict/classify the iris species with the given feature vectors. [9].

- Wine Quality Data Set: To predict whether a particular wine is "good quality" or not, each wine in this data set is given a "quality" type between 0 and 3. There are 13 features for this data, including Fixed acidity, Volatile acidity, Citric acid, Residual sugar, Chlorides, Free sulfur dioxide, Total sulfur dioxide, Density, pH, Sulfates, Alcohol. [7].

4 Examples and Results

For each data set, we will first create a model. This step includes how to split the data into training data and testing data. For Iris data, we select the first 100 data to train a model and the remaining 50 to test it. For Wine data, 80% of the raw data is used for training the model and the rest 20% to test it. After that, we start tuning parameters for each kernel methods.

4.1 The simulation data

The formula for the non-parametric regression model is $Y_i = f(x_i) + \epsilon_i$. ϵ_i is just a random variable with zero mean and variance. ϵ_i can be considered as Gaussian noisy [4] data which adds a variation in our simulation data. The value of Y is produced by function m(x) = $sin(x)$ by adding noise. We need noise in our data because we are estimating the value of Y when the value of X is equal to variations. The goal is to find non-linear relation between X and Y. The weight for kernel regression changes with the value of variation for X. When we plot the graph, we obtained the following graph [4], Figure 1, which is display below. The data looks non-linear. So, we applied kernel regression estimator that helps us to know the pattern of the graph.

Before applying the kernel function we must select the optimal bandwidth for kernel regression. Bandwidth is an important factor to determine the smoothness for kernel regression. The reason we are trying to have optimal bandwidth is to minimize error for the function m(x). So, I have used the k-fold cross-validation technique which I obtained from the following source [4]. In this k-fold cross-validation procedure, the value of k is 5. The data is randomly split into 5 sub-samples. Then, the procedure is run 5 times with one sub-sample is chosen for the validation set and other data as a sample set (training data set). Once all the sub-sample is used, then the average estimation of the prediction error is obtained. Then, such estimation is performed several times on validation data and gives minimum cross-validation

error. With k-fold cross validation technique, our optimal bandwidth is at h = 0.9 for the simulation data.

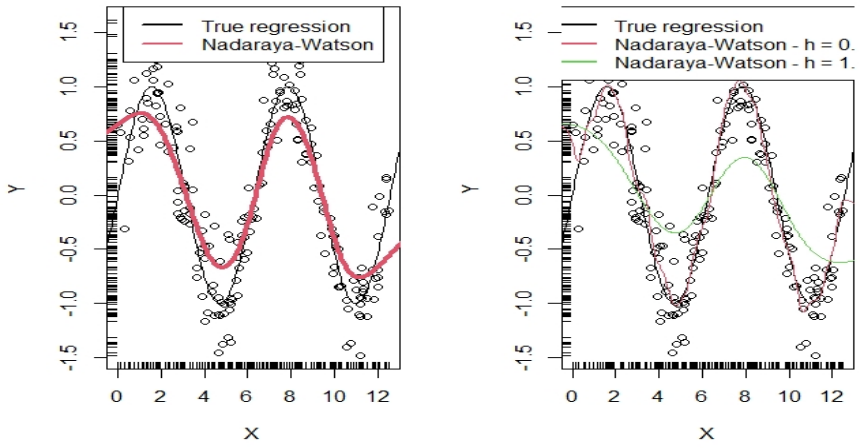

Figure 1: The graph of the data with different bandwidth [4]

The left-side figure is obtained after obtaining the optimal bandwidth. The red line is obtained from the Nadaraya-Watson kernel estimator which is close to the true regression line which is denoted by black line. The black line is a true regression line which we will never obtain because there would be some prediction error. As a result, our goal is to reach near to true regression line. The line looks smooth and near to the true regression line.

Another plot was obtained with the value of bandwidth $h = 0.2$ and 1.5. When the value of bandwidth is 0.2 which is represented by the red line, it looks bumpy. It does not look smooth. As a result, there is more variance and less bias. However, when the value of bandwidth is 1 which is represented by the green line, it looks smooth. In this case, the bias is increase and variance is decrease. However, the curve doesn't fit the data well enough.

4.2 Old Faithful Geyser Example

There are four plots in Figure 4.2. The top left figure is the plot between response variable and predictor variable where the smooth line hasn't obtain. The figure on the top right is obtained with the bandwidth = 0.5 which is

the default bandwidth in `ksmooth()` function. The curve for the data looks smooth and good. The bottom left figure is obtained with bandwidth = 0.1. So, as the bandwidth is small, we can see the high variance in the curve. The bandwidth of bottom right plot is 2 which gives smooth curve but doesn't fit well in the data.

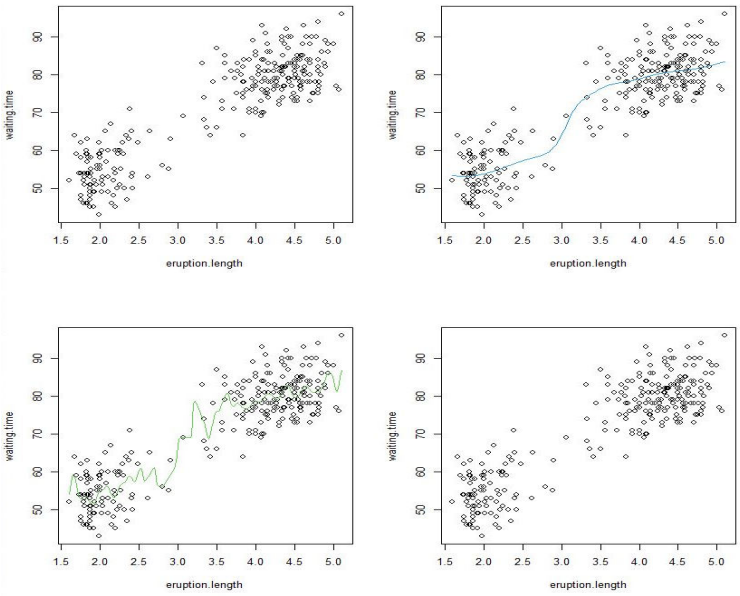

Figure 2: The graph of the data with different bandwidth [14]

So, again we used a cross-validation technique to obtain optimal bandwidth which will have small prediction error and gives a smooth curve. The optimal bandwidth with the cross validation is 0.8. So, one of the smoothest curve we can get is displayed below in a red line.

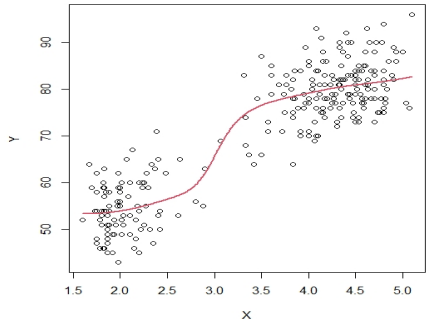

Figure 3: The graph with optimal bandwidth [14]

So, for both experiment (simulation and real world data), we obtain the optimal value for bandwidth with k-fold cross validation technique where the value of k is 5. We test the data with different bandwidth. At the beginning we didn't know the curve for the data. We didn't have any assumption about the data. So, as the value of bandwidth gets smaller, the line is more wiggly and as the value of bandwidth gets bigger, the line is smooth. More smooth would not cover all the data. We must have a optimal bandwidth. However, there is no any strict rule to obtain optimal bandwidth. There were various method that I came across while doing research for kernel regression such as dpill function in R, k-fold and leave one out cross validation. Among all of them, cross validation technique is highly used to obtain the optimal value for bandwidth. So, the value of bandwidth is important to approximate the regression curve.

4.3 Iris Species classification Example

4.3.1 Model Creation

```
library(tidyverse)
library(e1071)
data(iris)
index <- c(1:nrow(iris))
# split data into training and testing
test.index <- sample(index, size = (length(index)/3))
train <- iris[-test.index ,]
test <- iris[test.index ,]

svm.model.linear <- svm(Species ~ ., data = train, kernel = 'linear')
svm.model.linear
x1 = table(Prediction = predict(svm.model.linear, train),Truth = train$Species)
x2 = table(Prediction = predict(svm.model.linear, test),Truth = test$Species)
xtable(x1)
xtable(x2)
```

Table 1: Prediction vs Truth for Training data

	setosa	versicolor	virginica
setosa	32	0	0
versicolor	0	28	0
virginica	0	1	39

Table 2: Prediction vs Truth for Testing data

	setosa	versicolor	virginica
setosa	18	0	0
versicolor	0	18	0
virginica	0	3	11

For the linear model here, there is only one sample in the training data is misclassified, that is, a sample should belong to the versicolor class but classified as virfinica. However, there are three sample misclassified for the test data.

We calculate that the training accuracy is $1 - 1/100 = 99\%$ and the testing accuracy is $1 - 3/50 = 94\%$.

4.3.2 Kernel Tuning

In this section, we test on three kernel methods and calculate the training accuracy and testing accuracy. We will focus on the testing accuracy to determine the optimal kernel function and parameters.

Polynomial Kernel

```
svm.model.poly <- svm(Species ~ ., data = train, kernel = 'polynomial')
svm.model.poly
p1 = table(Prediction = predict(svm.model.poly, train),Truth = train$Species)
p2 = table(Prediction = predict(svm.model.poly, test),Truth = test$Species)
xtable(p1)
xtable(p2)
```

Table 3: Polynomial Kernel for Training data

	setosa	versicolor	virginica
setosa	32	0	0
versicolor	0	29	7
virginica	0	0	32

Table 4: Polynomial Kernel for Testing data

	setosa	versicolor	virginica
setosa	18	0	0
versicolor	0	21	0
virginica	0	0	11

Training accuracy is $1 - 7/100 = 93\%$ and the testing accuracy is $1 - 0/50 = 100\%$.

RBF Kernel

```
svm.model.radial <- svm(Species ~ ., data = train, kernel = 'radial')
svm.model.radial
r1 = table(Prediction = predict(svm.model.radial, train),Truth = train$Species)
r2 = table(Prediction = predict(svm.model.radial, test),Truth = test$Species)
xtable(r1)
xtable(r2)
```

Table 5: RBF Kernel for Training data

	setosa	versicolor	virginica
setosa	32	0	0
versicolor	0	28	0
virginica	0	1	39

Table 6: RBF Kernel for Testing data

	setosa	versicolor	virginica
setosa	18	0	0
versicolor	0	18	1
virginica	0	3	10

Training accuracy is $1 - 1/100 = 99\%$ and the testing accuracy is $1 - 4/50 = 92\%$.

Sigmoid Kernel

```
svm.model.sig <- svm(Species ~ ., data = train, kernel = 'sigmoid')
svm.model.sig
s1 = table(Prediction = predict(svm.model.sig, train),Truth = train$Species)
s2 = table(Prediction = predict(svm.model.sig, test),Truth = test$Species)
xtable(s1)
xtable(s2)
```

Table 7: Sigmoid Kernel for Training data

	setosa	versicolor	virginica
setosa	32	0	0
versicolor	0	26	1
virginica	0	3	38

Table 8: Sigmoid Kernel for Testing data

	setosa	versicolor	virginica
setosa	17	0	0
versicolor	1	17	0
virginica	0	4	11

Training accuracy is $1 - 4/100 = 96\%$ and the testing accuracy is $1 - 4/50 = 92\%$.

4.3.3 Parameter Tuning

Next, we are tuning the parameters for each kernel method. There are two parameters to tuning including γ and cost.

```
tune.svm <- tune(svm, Species~., data = train, kernel = 'radial',
                 ranges = list(gamma = 2^(-10:5), cost = 2^(0:5)))
tune.svm
```

The codes above tune parameter γ ranging from 2^{-10} to 2^5 and the cost from 1 to 32. From this code, we get the output that computes the optimal γ and cost for a specific kernel. For example, for RBF, the outputs are $\gamma = 0.25$ and cost=1. Therefore, we will set these two values as the following.

```
tuned.svm <- svm(Species ~ . , data = train, kernel = 'radial', gamma = 0.25, cost = 1)
t1 = table(Prediction = predict(tuned.svm, train),Truth = train$Species)
t2 = table(Prediction = predict(tuned.svm, test),Truth = test$Species)
```

We report the best parameters for each kernel and the corresponding training and testing accuracy in a table below.

Table 9: Comparison Table

No.	kernel	γ	cost	training acc	testing acc
1	Linear	0.00098	1	0.99	0.94
2	Polynomial	1	2	0.99	0.96
3	RBF	0.25	1	0.99	0.92
4	Sigmoid	0.125	1	0.98	0.92

4.4 Wine quality classification

4.4.1 Model Creation

```
wine.url <- "F:\\Desktop\\Spring 2021\\Adv. Stat Modeling\\wine.data"
wine <- read.csv(wine.url, header=FALSE)
colnames(wine) <- c('Type', 'Alcohol', 'Malic', 'Ash',
                    'Alcalinity', 'Magnesium', 'Phenols',
                    'Flavanoids', 'Nonflavanoids',
                    'Proanthocyanins', 'Color', 'Hue',
                    'Dilution', 'Proline')
wine$Type <- as.factor(wine$Type)
save(wine, file="wine.Rdata", compress=TRUE)
```

The wine data set is from a file downloaded from `http://archive.ics.uci.edu/ml/machine-learning-databases/wine/`. The codes above are to prepare the data in R and store it.

```
index <- c(1:nrow(wine))
test.index <- sample(index, size = (length(index)/5))
train <- wine[-test.index ,]
test <- wine[test.index ,]

svm.model.linear <- svm(Type ~ ., data = train, kernel = 'linear')
svm.model.linear
x1 = table(Prediction = predict(svm.model.linear, train),Truth = train$Type)
x2 = table(Prediction = predict(svm.model.linear, test),Truth = test$Type)
xtable(x1)
xtable(x2)
```

Similar to the Iris data process in Section 4.1, "Type" is the response variable and it has values of 1, 2 and 3.

Table 10: Prediction vs Truth for Training data

	1	2	3
1	46	0	0
2	0	59	0
3	0	0	38

Table 11: Prediction vs Truth for Testing data

	1	2	3
1	13	0	0
2	0	11	0
3	0	1	10

For the linear model here, there is only one sample in the testing data is misclassified, that is, a sample should belong to the class 2 but classified as class 3 for the test data.

We calculate that the training accuracy is $1 - 0/143 = 100\%$ and the testing accuracy is $1 - 1/35 = 97.143\%$.

4.4.2 Kernel Tuning

In this section, we test on three kernel methods and calculate the training accuracy and testing accuracy. We will focus on the testing accuracy to determine the optimal kernel function and parameters.

Polynomial Kernel

```
svm.model.poly <- svm(Type ~ ., data = train, kernel = 'polynomial')
svm.model.poly
p1 = table(Prediction = predict(svm.model.poly, train),Truth = train$Type)
p2 = table(Prediction = predict(svm.model.poly, test),Truth = test$Type)
xtable(p1)
xtable(p2)
```

Table 12: Polynomial Kernel for Training data

	1	2	3
1	43	0	0
2	3	59	0
3	0	0	38

Table 13: Polynomial Kernel for Testing data

	1	2	3
1	11	1	0
2	2	11	0
3	0	0	10

Training accuracy is $1 - 3/143 = 97.9\%$ and the testing accuracy is $1 - 2/35 = 94.286\%$.

RBF Kernel

```
svm.model.radial <- svm(Type ~ ., data = train, kernel = 'radial')
svm.model.radial
r1 = table(Prediction = predict(svm.model.radial, train),Truth = train$Type)
r2 = table(Prediction = predict(svm.model.radial, test),Truth = test$Type)
xtable(r1)
xtable(r2)
```

Table 14: RBF Kernel for Training data

	1	2	3
1	46	0	0
2	0	59	0
3	0	0	38

Table 15: RBF Kernel for Testing data

	1	2	3
1	13	0	0
2	0	12	0
3	0	0	10

Training accuracy is $1 - 0/143 = 100\%$ and the testing accuracy is $1 - 0/35 = 100\%$.

Sigmoid Kernel

```
svm.model.sig <- svm(Type ~ ., data = train, kernel = 'sigmoid')
svm.model.sig
s1 = table(Prediction = predict(svm.model.sig, train),Truth = train$Type)
s2 = table(Prediction = predict(svm.model.sig, test),Truth = test$Type)
xtable(s1)
xtable(s2)
```

Table 16: Sigmoid Kernel for Training data

	1	2	3
1	45	0	0
2	1	58	0
3	0	1	38

Table 17: Sigmoid Kernel for Testing data

	1	2	3
1	13	0	0
2	0	12	0
3	0	0	10

Training accuracy is $1 - 2/143 = 98.6\%$ and the testing accuracy is $1 - 0/35 = 100\%$.

4.4.3 Parameter Tuning

Next, we are tuning the parameters for each kernel method. There are two parameters to tuning including γ and cost.

```
tune.svm <- tune(svm, Type~., data = train, kernel = 'radial',
                 ranges = list(gamma = 2^(-10:5), cost = 2^(0:5)))
tune.svm
```

The codes above tune parameter γ ranging from 2^{-10} to 2^5 and the cost from 1 to 32. From this code, we get the output that computes the optimal γ and cost for a specific kernel. For example, for RBF, the outputs are $\gamma = 0.0625$ and cost=1. Therefore, we will set these two values as the following.

```
tuned.svm <- svm(Type ~ . , data = train, kernel = 'radial', gamma = 0.0625, cost = 1)
t1 = table(Prediction = predict(tuned.svm, train),Truth = train$Type)
t2 = table(Prediction = predict(tuned.svm, test),Truth = test$Type)
```

We report the best parameters for each kernel and the corresponding training and testing accuracy in a table below.

Table 18: Comparison Table

No.	kernel	γ	cost	training acc	testing acc
1	Linear	0.00098	1	1	0.97143
2	Polynomial	0.625	4	1	0.97143
3	RBF	0.25	1	1	1
4	Sigmoid	0.03125	4	0.993	0.97143

5 Conclusion

In this report, we first discussed the Nadaraya-Watson kernel estimator with real-world data. The optimal bandwidth for kernel regression is chosen with the help of the k-fold cross-validation technique. The rough estimation of the curve is given by the small bandwidth value and smooth estimation of the curve is given by the larger bandwidth value. Next, we tested the kernel SVM models on the Iris species and wine quality classification data and compared the training accuracy and testing accuracy on different kernel selections. Two key parameters σ and cost are tuned further to get the optimal ones. We find that the results are not sensitive to the parameter cost while we have to carefully determine γ. The polynomial kernel for SVM with $\gamma = 1$ and cost set to be 2 is the best for Iris data, $\gamma = 0.25$ and cost=1 for RBF kernel for wine quality data, respectively. which achieved a testing accuracy of 96% and 100%. For future study, we will test on various data sets.

References

[1] S. Bergman. *The kernel function and conformal mapping*, volume 5. American Mathematical Soc., 1970.

[2] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

[3] R. G. Brereton and G. R. Lloyd. Support vector machines for classification and regression. *Analyst*, 135(2):230–267, 2010.

[4] Y.-C. Chen. Nonparametric regression and cross-validation. `http://faculty.washington.edu/yenchic/17Sp_302/R12.pdf`. Accessed: 2021-03-31.

[5] P. E. Cheng. Applications of kernel regression estimation: survey. *Communications in statistics-theory and methods*, 19(11):4103–4134, 1990.

[6] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[7] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision support systems*, 47(4):547–553, 2009.

[8] S. Efromovich. *Nonparametric curve estimation: methods, theory, and applications*. Springer Science & Business Media, 2008.

[9] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[10] L. Györfi, M. Kohler, A. Krzyzak, and H. Walk. *A distribution-free theory of nonparametric regression*. Springer Science & Business Media, 2006.

[11] S. Han, C. Qubo, and H. Meng. Parameter selection in svm with rbf kernel function. In *World Automation Congress 2012*, pages 1–4. IEEE, 2012.

[12] T. Hofmann, B. Schölkopf, and A. J. Smola. Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220, 2008.

[13] T. Joachims. Making large-scale svm learning practical. Technical report, Technical report, 1998.

[14] U. of South Carolina. R code to analyze the simulated (x,y) data using nonparametric kernel regression. `https://people.stat.sc.edu/hitchcock/kernelregressionRexample704.txt`. Accessed: 2021-04-01.

[15] E. G. Portugués. 6.2.1 nadaraya-watson estimator. `https://bookdown.org/egarpor/PM-UC3M/npreg-kre.html#npreg-kre-nw`. Accessed: 2021-03-21.

[16] K. Thurnhofer-Hemsi, E. López-Rubio, M. A. Molina-Cabello, and K. Najarian. Radial basis function kernel optimization for support vector machine classifiers. *arXiv preprint arXiv:2007.08233*, 2020.

[17] T. Van Gestel, J. A. Suykens, B. Baesens, S. Viaene, J. Vanthienen, G. Dedene, B. De Moor, and J. Vandewalle. Benchmarking least squares support vector machine classifiers. *Machine learning*, 54(1):5–32, 2004.

[18] W. Wang, Z. Xu, W. Lu, and X. Zhang. Determination of the spread parameter in the gaussian kernel for classification and regression. *Neurocomputing*, 55(3-4):643–663, 2003.