R Package: KernelRegression

```
library(ggplot2)
library(tidyverse)
library(magrittr)
source("../R/kernel_functions.R")
source("../R/kernel_regression_class.R")
```

Link to package GitHub: https://github.com/edwarddavis1/KernelRegression

The Kernel Regression

The most simple form of regression is the least squares (LS) regression. This finds a regression model, $f(\mathbf{x}, \mathbf{w})$ by finding an optimal value for a weights vector, w_{LS}, which weights feature vectors in some input space feature transform, $\phi(\mathbf{X})$, such that the difference between the model and the output, $\mathbf{y}$ is minimised. This optimal value has the solution,

$$\mathbf{w}_{LS} = (\phi(\mathbf{X})\phi(\mathbf{X})^T)^{-1}\phi(\mathbf{X})\mathbf{y}^T.$$

To reduce the effect of overfitting, a regularisation parameter, λ, can be introduced, which modifies the optimal weights vector to

$$\mathbf{w}_{LS-R} = (\phi(\mathbf{X})\phi(\mathbf{X})^T + \lambda I)^{-1}\phi(\mathbf{X})\mathbf{y}^T.$$

Through the use of the Woodbury identity, this solution can be re-written such that the feature space, $\phi(\mathbf{X})$ only appears in the form of an inner product. This allows for the use of kernel methods in the regression. Using this, the kernel regression model can be written as

$$\mathbf{f}(\mathbf{x}; \mathbf{w_{LS-R}}) = \mathbf{k}(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y}^T,$$

where $\mathbf{K}$ is a matrix of kernel function values from pairs of $\mathbf{X}$ data points, and $\mathbf{k}$ is a vector of kernel function combinations of a new data point, x_0, with $\mathbf{X}$. In the case of kernel regression, different feature spaces can be induced with different kernel functions.

The linear kernel function,

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle,$$

induces a linear feature space, which is the same as the one found in a linear least squares regression.

The polynomial kernel function,

$$k(\mathbf{x}_i, \mathbf{x}_j) = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle + 1)^b,$$

induces a polynomial feature transform of degree b. This kernel is useful for modelling polynomial data with a large b; while this is possible with the least squares regression, it would require the construction of very large $(b \times b)$ matrices and would therefore be very computationally inefficient.

This idea of complex induced feature spaces can go further, however. As the feature space term is always in an inner product, this means that there is no longer a limit on the feature space dimensionality, and

therefore it is possible to induce an infinite feature space, allowing for greatly enhanced model flexibility. This is made possible through the use of the radial basis function (RBF),

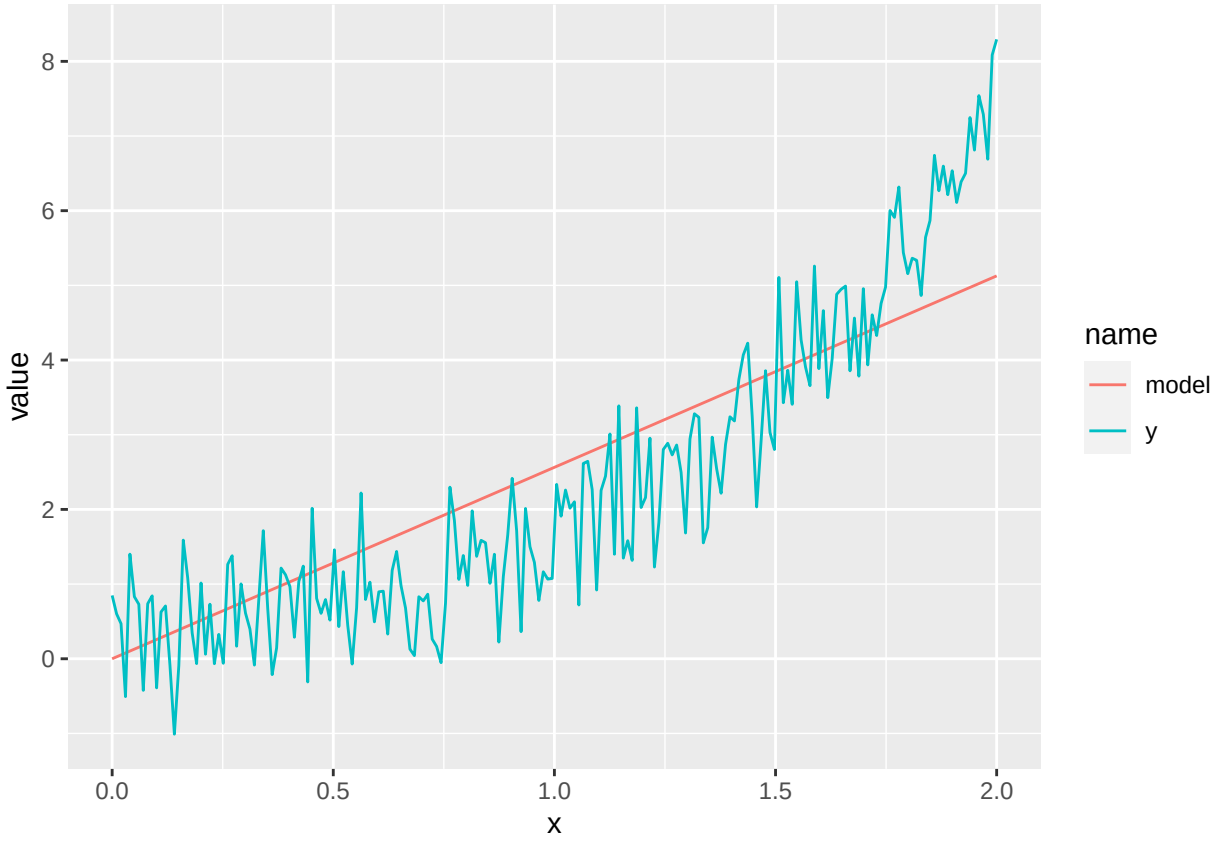
$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}\right).$$

Example Package Usage

The package contains a reference class for a `kernel_regression` object. The default constructor for this object selects a linear kernel, but through the use of the `set_kernel()` member function, this can easily be changed to either a polynomial or a radial basis function kernel. Additionally, this function allows the user to update any parameters associated with these kernels.

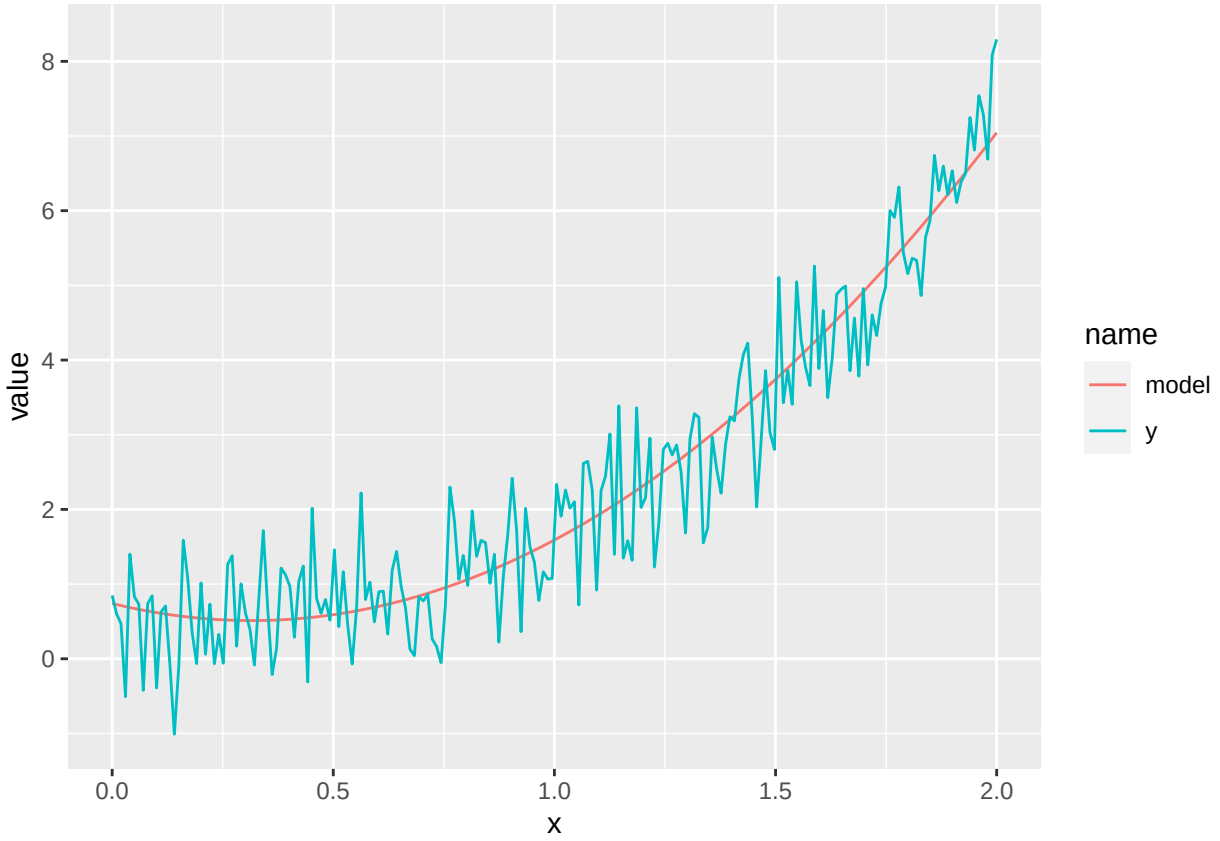
The example below demonstrates the default object which regresses on noisy exponential data. The performance of the regression can illustrated by using the `plot()` member function, which plots the original data with the kernel regression model.

```
# Exponential data
x = as.matrix(seq(0, 2, length=200))
get_exp_data <- function(x) exp(1.5*x-1)
exp_data = get_exp_data(x)

# Noise
noise = rnorm(x, mean=0, sd = 0.64)

y = exp_data + noise

x = data.frame(x)
y = data.frame(y)

kr1 = kernel_regression(x, y)
kr1$plot()
```



In this example, the linear kernel is not appropriate. To construct a more accurate regression, the object's kernel can be swapped to the polynomial kernel, with a degree parameter of 2.

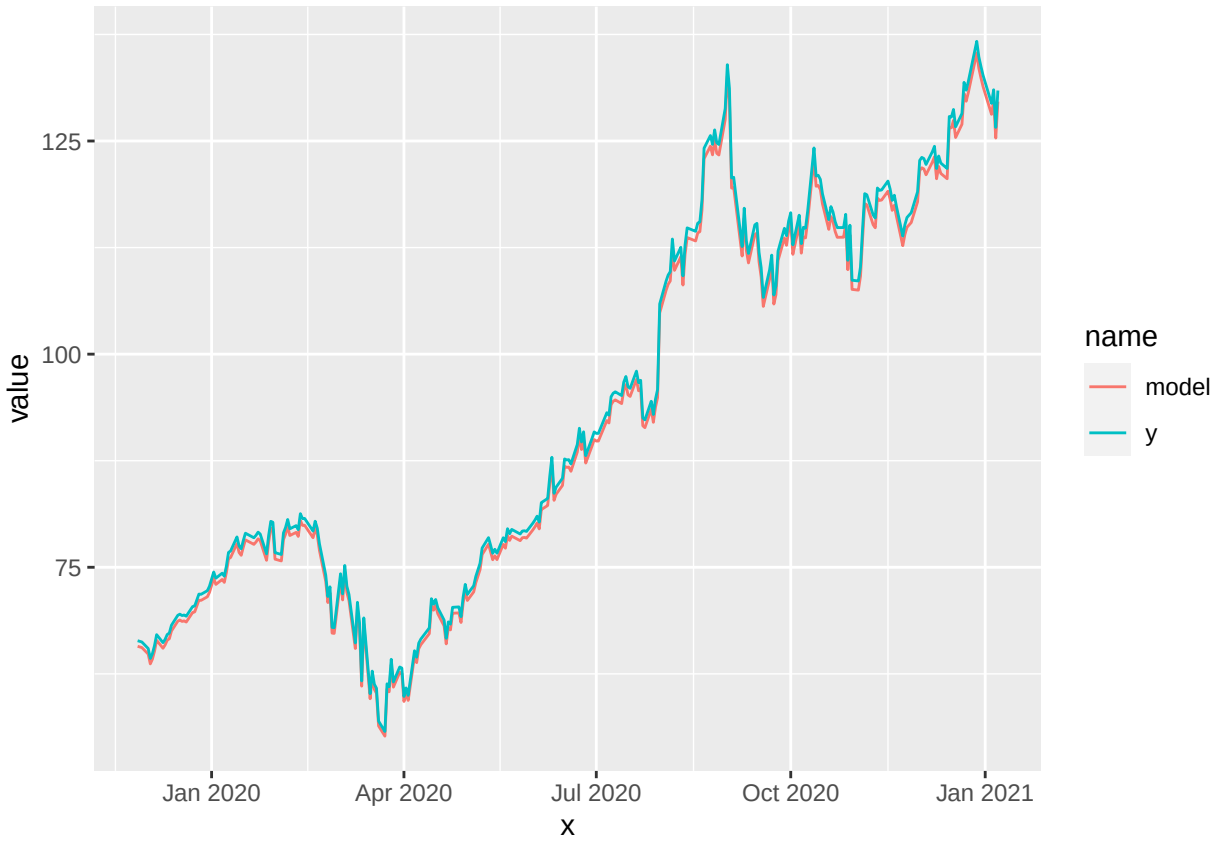
```
kr1$set_kernel("poly", b=2)
kr1$plot()
```



This regression is much more appropriate.

Where the kernel regression shines, is when modelling highly complex trends. In the example below, the kernel function object is created using data from the stock market. Here, the RBF kernel can be used which can capture highly complex relationships as it can induce an infinite feature space. Further, as we are now modelling a time series, the `x_date` parameter in the constructor is set to `TRUE`, which allows for the timestamps to be preserved for plotting.

```
# Apple stock data
aapl_data = read.csv("../data/apple_stock_data.csv")
aapl_data %<>% select(c(date, adjusted))

x_df = data.frame(aapl_data$date)
y_df = data.frame(aapl_data$adjusted)
kr2 = kernel_regression(x_df, y_df, kernel="RBF", x_date=TRUE)
kr2$plot()
```


125
100
75
value
Jan 2020
Apr 2020
Jul 2020
Oct 2020
Jan 2021
x
name
model
y