Supervised Learning (COMP0078) - Coursework 1

Toby Drane

16 November, 2020

1 PART I

We define our input vector $x = (x_1, ..., x_n)$ where $x \in \mathbb{R}^n$. Each input value has a corresponding y, such that $y \in \mathbb{R}$.

1.1 Linear Regression

For our input data set of $D = \{(1, 3), (2, 2), (3, 0), (4, 5)\}$. Our X matrix after applying the polynomial basis function, is defined as

$$X = \begin{bmatrix} \phi_1(x_1) & \cdots & \phi_k(x_1) \\ \vdots & \ddots & \\ \phi_1(x_m) & & \phi_k(x_m) \end{bmatrix} \tag{1}$$

With the y values being a vector

$$Y = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} \tag{2}$$

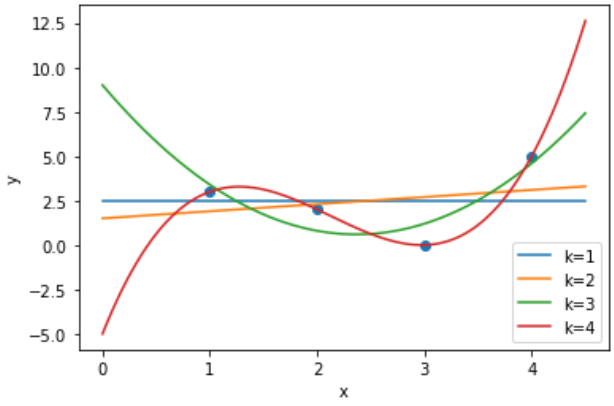
Linear regression then creates a weight vector to minimize the loss

$$w = (X^T X)^{-1} X^T Y \tag{3}$$

1. For each of the polynomial bases $k = 1, 2, 3, 4$, fit the data set D.

   (a) Produce a plot, superimposing the four different curves. The plot with k polynomial bases can be seen in figure 1.

Figure 1: Curves for $k = 1, 2, 3, 4$



(b) Give the equations that correspond to $k = 1, 2, 3$

- $k = 1$, $y = 2.5$
- $k = 2$, $y = 0.4x + 1.5$
- $k = 3$, $y = 9 - 7.1x + 1.5x^2$

(c) For every curve $k = 1, 2, 3, 4$, the mean square errors $MSE = \frac{SSE}{m}$ are as follows

- $k = 1$ $MSE = 3.25$
- $k = 2$, $MSE = 3.05$
- $k = 3$, $MSE = 0.80$
- $k = 4$, $MSE = 7.12E^{-25} \approx 0$

2. Here we illustrate the phenomena of *overfitting*.

$$g_\sigma(x) = sin^2(2\pi x) + \epsilon \tag{4}$$

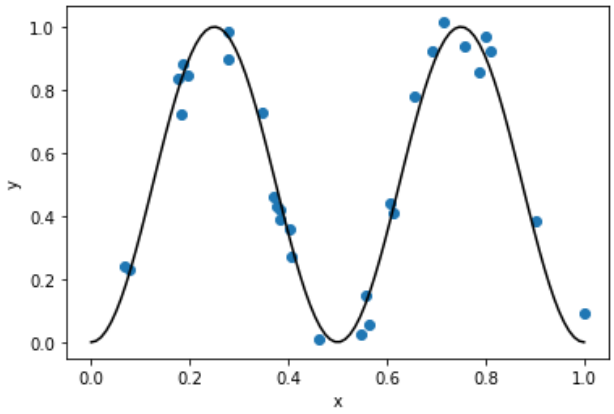
ϵ is our noise, a random variable distributed normally with mean 0 and variance σ^2.

$$S_{0.07,30} = \{(x_1, g_{0.07}(x_1), ..., (x_{30}, g_{0.07}(x_{30}))\} \tag{5}$$

Where x is a random uniform samples from 0 to 1, 30 times.

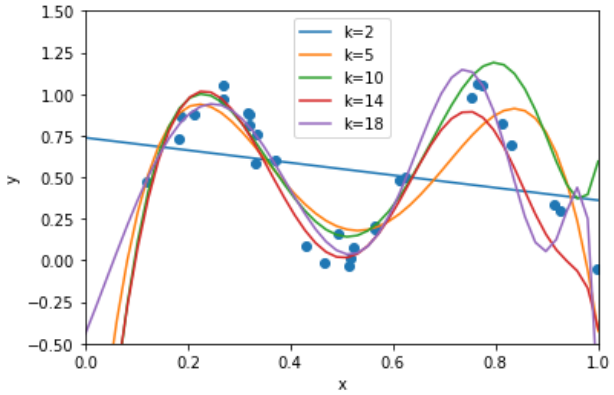
(a) i. The plot of function $sin^2(2\pi x)$ in the range $0 \leq x \leq 1$, can be seen in figure 2.

Figure 2: Curve of $sin^2(2\pi x)$



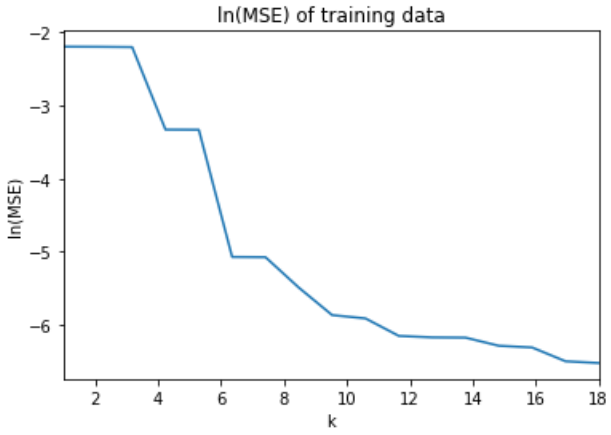
ii. Figure 3 is the plot for the different polynomial bases for data set S.

Figure 3: Curves for $k = 2, 5, 10, 14, 18$



(b) Figure 4 is the plot of, the natural log of the MSE for the training data S versus the polynomial k dimension.

Figure 4: Natural log of MSE for S data set, versus k dimension.

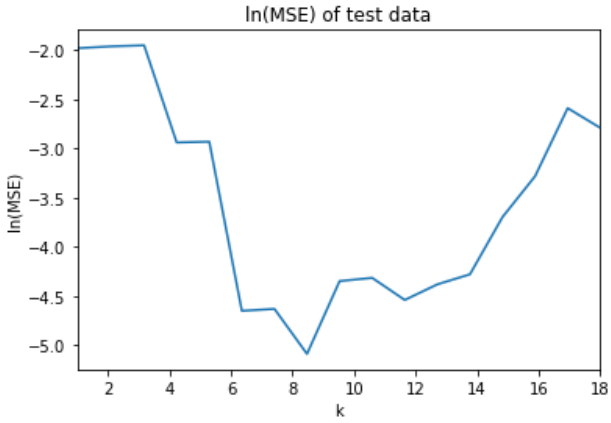


(c) For a test data set, T.

$$T_{0.07,1000} = \{(x_1, g_{0.07}(x_1), ..., (x_{1000}, g_{0.07}(x_{1000}))\} \tag{6}$$

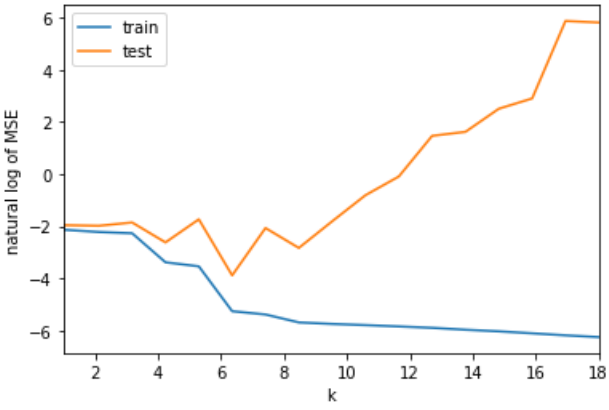
Using the model fit generated from the training set, we plot the natural log of the test MSE for the T data set, show in figure 5, it is good to note this is a **increasing function**, displaying the phenomena of *overfitting*.

Figure 5: Natural log of MSE for T data set, versus k dimension.



(d) Every run yields different results, figure 6, shows the plot of both the S and T natural log MSE errors, averaged over 100 runs.

Figure 6: Natural log of MSE for S & T, averaged over 100 runs.



3. We now use the basis function below, for b. c. d. experiments that are repeated with $k = (1, ..., 18)$.

$$sin(1\pi x), sin(2\pi x), ..., sin(k\pi x) \tag{7}$$

(a) Figure 7 shows the natural log MSE for the training data set S.

(b) Figure 8 shows the natural log MSE for the testing data set T.

(c) Natural log of MSE average values for both the S, T data set is show in figure 9.

Figure 7: Natural log of MSE for S, new basis function.

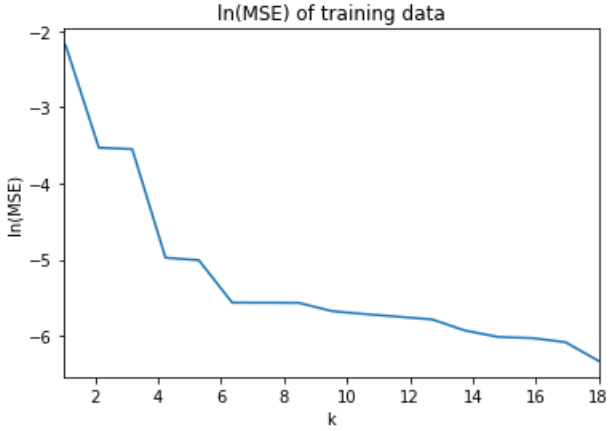


Figure 8: Natural log of MSE for T, new basis function.

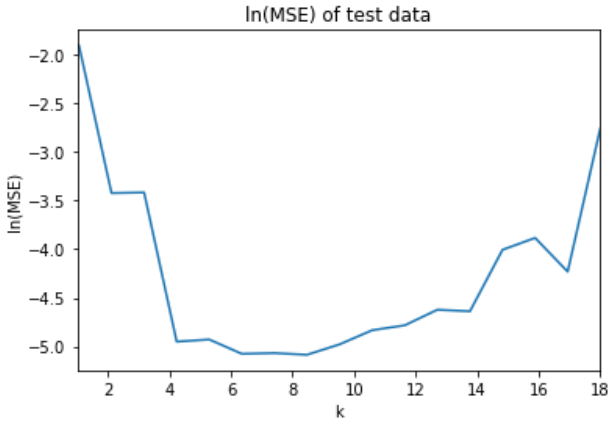
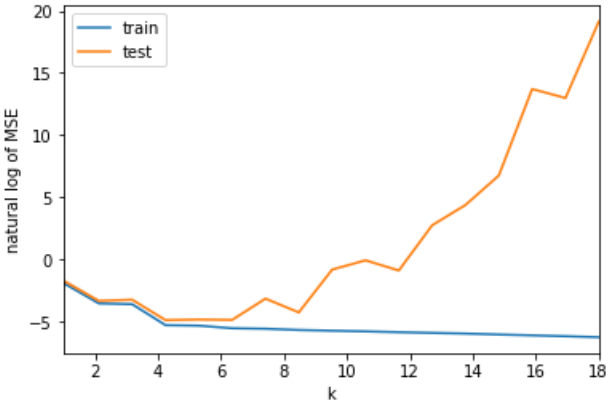


Figure 9: Natural log of MSE for S & T, averaged over 100 runs, new basis function.



1.2 Filtered Boston Housing and Kernels

1. *"Baseline versus full linear regression."*

   (a) Naive Regression: By using a vector of ones the length of the training and test sets respectively, fit the data with a constant function, and perform linear regression. The MSE of the training and test sets:

   - MSE training set: 83.98240826306767
   - MSE testing set: 84.87363555523683

   (b) Fitting data with a constant function is calculating the average y values across the data set.

   (c) Single Attribute Linear Regression: For each of the 12 separate attributes, perform linear regression, incorporating a bias term $(x_i, 1)$.

   - Attribute 1: Train MSE 72.59207798308606, Test MSE 68.40484721412149
   - Attribute 2: Train MSE 73.07417893128834, Test MSE 73.69521056232553
   - Attribute 3: Train MSE 65.98854803183018, Test MSE 60.83251802626501
   - Attribute 4: Train MSE 82.81252885490696, Test MSE 78.58546648415941
   - Attribute 5: Train MSE 69.19380243875086, Test MSE 67.96915982708865
   - Attribute 6: Train MSE 43.42398344621175, Test MSE 43.07703005789345
   - Attribute 7: Train MSE 76.06358332722291, Test MSE 64.48756263248931

- Attribute 8: Train MSE 79.7749581224216, Test MSE 76.75495343297662
- Attribute 9: Train MSE 71.95614175601358, Test MSE 71.76078020550806
- Attribute 10: Train MSE 66.9805613878754, Test MSE 62.61645252232605
- Attribute 11: Train MSE 62.68508362533773, Test MSE 61.781136055710945
- Attribute 12: Train MSE 37.81464584994674, Test MSE 39.15776899348431

(d) All Attribute Linear Regression: Perform linear regression across all 12 attributes. The MSE of the training and test sets:

- MSE training set: 22.040432795607025
- MSE testing set: 20.381950262563215

1.3 Kernelised Ridge Regression

Here we use the dual version with kernels to perform linear regression for nonlinear data sets. For a given function K we define the kernel matrix as

$$K_{i,j} := K(x_i, x_j) \tag{8}$$

The dual optimisation formulation with kernelization is

$$\alpha^* = argmin_{\alpha \in \mathbb{R}^l} \frac{1}{l} \sum_{i=1}^{l} (\sum_{j=1}^{l} \alpha_j K_{i,j} - y_i)^2 + \gamma \alpha^T K \alpha \tag{9}$$

Now for $y := (y_1, ..., y_l)^T$, we can solve the dual as

$$\alpha^* = (K + \gamma l I_l)^{-1} y \tag{10}$$

where I_l is the $l \times l$ identity matrix. The regression prediction function can be defined as

$$y_{test} = \sum_{i=1}^{l} \alpha_i^* K(x_i, x_{test}) \tag{11}$$

1. "Kernel Ridge Regression", we perform kernel ridge regression on the same Boston data set using the Gaussian kernel

$$K(x_i, x_j) = exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2}) \tag{12}$$

(a) We generate two vectors for γ and σ. Using five-fold cross-validation for all possible values of γ and σ we perform the ridge regression to generate the best combination.

- Best gamma: $\gamma = 2^{-33}$
- Best sigma: $\sigma = 2^{10}$

(b) The plot of the "cross-validation error" as a function of γ and σ can be seen in figure 10 and 11.

Figure 10: Cross-validation error as a function of *gamma* & σ

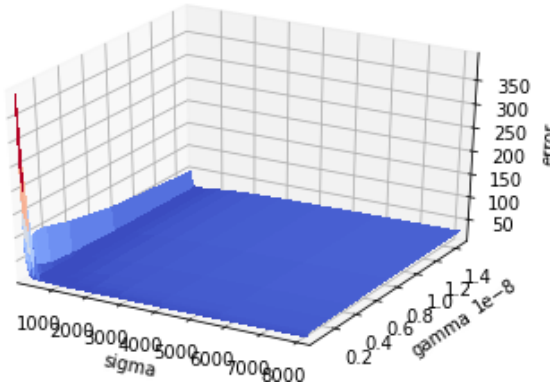
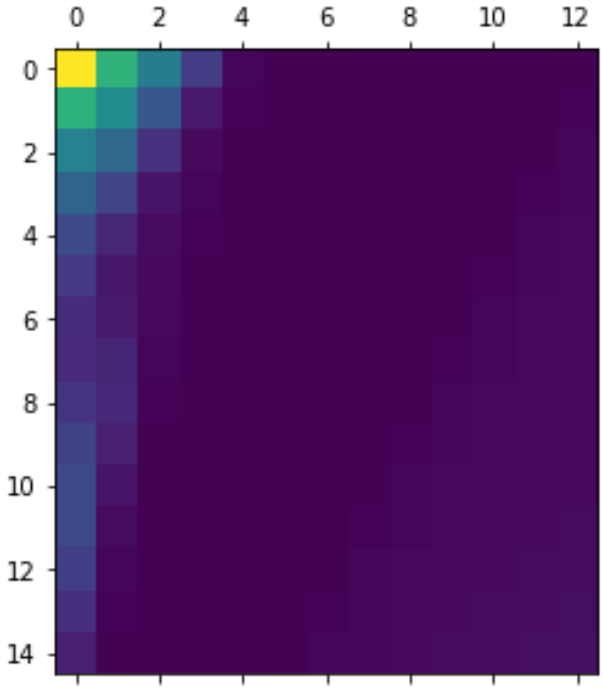


Figure 11: Cross-validation error as a function of *gamma* & σ, matrix form



(c) For the best values of gamma and sigma, the MSE were as follows

- MSE training set: 22.244272339393515
- MSE testing set: 20.13779340381408

(d) The previous a. c. d. and this c. were repeated 20 times, each with different data set splits. The results can be seen in table 1.

2 Part II

1. In the following questions we derive the Bayes estimator with respect to the probability mass function $p(x, y)$, over (X, Y) where X and Y are finite, such that $\sum_{x \in X} \sum_{y \in Y} p(x, y) = 1$.

Method	**MSE train**	**MSE test**
Naive Regression	$84.72 \pm \sigma^{'} 6.22$	$83.33 \pm \sigma^{'} 12.64$
Linear Regression (attribute 1)	$71.11 \pm \sigma^{'} 4.05$	$71.12 \pm \sigma^{'} 7.56$
Linear Regression (attribute 2)	$75.22 \pm \sigma^{'} 4.31$	$68.94 \pm \sigma^{'} 9.14$
Linear Regression (attribute 3)	$64.88 \pm \sigma^{'} 3.97$	$63.35 \pm \sigma^{'} 7.71$
Linear Regression (attribute 4)	$78.99 \pm \sigma^{'} 4.22$	$86.58 \pm \sigma^{'} 8.25$
Linear Regression (attribute 5)	$69.51 \pm \sigma^{'} 5.71$	$66.80 \pm \sigma^{'} 11.32$
Linear Regression (attribute 6)	$41.86 \pm \sigma^{'} 4.26$	$46.07 \pm \sigma^{'} 8.11$
Linear Regression (attribute 7)	$72.50 \pm \sigma^{'} 5.19$	$71.42 \pm \sigma^{'} 10.90$
Linear Regression (attribute 8)	$79.93 \pm \sigma^{'} 4.67$	$76.99 \pm \sigma^{'} 9.66$
Linear Regression (attribute 9)	$71.36 \pm \sigma^{'} 5.77$	$72.63 \pm \sigma^{'} 11.04$
Linear Regression (attribute 10)	$63.85 \pm \sigma^{'} 4.23$	$69.18 \pm \sigma^{'} 8.53$
Linear Regression (attribute 11)	$63.92 \pm \sigma^{'} 3.68$	$59.31 \pm \sigma^{'} 7.60$
Linear Regression (attribute 12)	$37.54 \pm \sigma^{'} 3.00$	$39.88 \pm \sigma^{'} 5.93$
Linear Regression (all attributes)	$22.76 \pm \sigma^{'} 1.30$	$19.34 \pm \sigma^{'} 1.52t$
Kernel Ridge Regression	$8.23 \pm \sigma^{'} 0.75$	$5.99 \pm \sigma^{'} 1.28$

Table 1: Table of errors, for all four different regression methods.

(a) Here $Y = [k]$, let $c \in [0, \infty)^k$ be a vector of k costs. Defining $L_c : [k] \times [k] \to [0, \infty)$ as,

$$L_c(y, \hat{y}) := [y \neq \hat{y}] c_y \tag{13}$$

This can be the imbalanced classification loss function, such that if we don't predict the correct outcome we suffer c_y loss. We can derive the **Bayes estimator** as:

$$\begin{aligned} L_c(y, \hat{y}) : &= [y \neq \hat{y}] c_y \\ &= [1 - (y = \hat{y}) c_y] \end{aligned} \tag{14}$$

$$\epsilon = argmin_{\hat{y}} \mathbb{E}[L_c(y, \hat{y})]$$

We condition on X to produce the following:

$$\begin{aligned} \epsilon &= \mathbb{E}_X \mathbb{E}_{Y|X}[L_c(Y, \hat{y})|X] \\ &= \mathbb{E}_X[\sum_{y \in Y} L_c(y, \hat{y}) p(y|X)] \\ &= argmin_{\hat{y}} \sum_{y \in Y} [1 - (y = \hat{y}) c_y] p(y|X) \\ &= argmax_{\hat{y}} \sum_{y \in Y} (y = \hat{y}) c_y p(y|X) \end{aligned} \tag{15}$$

We can say when $y = \hat{y}$, our loss function is 1, thus our Bayes estimator is,

$$\epsilon = argmax_{\hat{y}} p(y|X = x) c_y \tag{16}$$

(b) Given a strictly convex differentiable function $F : \mathbb{R} \to \mathbb{R}$ with define the F-Loss

$$L_F(y, \hat{y}) := F(\hat{y}) - F(y) + (\hat{y} - y) F^{'}(y) \tag{17}$$

i. If $F(x) = x^2$, we can define the F-Loss as

$$\begin{aligned} L_F(y, \hat{y}) &= \hat{y}^2 - y^2 + 2y^2 - 2y\hat{y} \\ &= (y - \hat{y})^2 \end{aligned} \tag{18}$$

this is better known as square loss.

ii. Prove that $y = \hat{y} \leftrightarrow L_f(y, \hat{y}) = 0$

$$\begin{aligned} L_F(y, y) &= y^2 - y^2 + (y - y)F^{'}(y) \\ &= y^2 - y^2 + (0)F^{'}(y) \\ &= y^2 - y^2 \\ &= 0 \end{aligned} \tag{19}$$

iii. There are three different scenarios that can occur for F-Loss; $y > \hat{y}$, $y = \hat{y}$, $y < \hat{y}$. It is instinctive to assume that if $y > \hat{y}$, the square loss will always be > 0. We have proven above that for the second case when $y = \hat{y}$, the F-Loss is 0. This finally leaves the scenario of when $y < \hat{y}$. A quick proof below shows under this circumstance F-Loss will be greater than 0
Let $y = z$ and $\hat{y} = 2z$

$$\begin{aligned} &= (z - 2z)^2 \\ &= (-z)^2 \\ &= -z * -z \\ &= z^2 \end{aligned} \tag{20}$$

The positive value of z is important to note here proving that for $y, \hat{y} \in \mathbb{R}$, F-Loss ≥ 0.

iv. Derive the Bayes estimator for F-Loss, for when F-Loss is $(y - \hat{y})^2$.
For the following derivations we assume $\hat{y}$, is the outcome of some learnt functions given a input data x. We also solve for a given point x".

$$\begin{aligned} \epsilon(f) &= \sum_{x \in X} \sum_{y \in Y} (y - f(x))^2 p(x, y) \\ &= \sum_{x \in X} [\sum_{y \in Y} (y - f(x))^2 p(y|x)] p(x) \\ \epsilon(f(x")) &= [\sum_{y \in Y} p(y - f(x"))^2 p(y|x")] p(x") \\ \frac{\partial \epsilon(f(x"))}{\partial f(x"))} &= \sum_{y \in Y} -2(y - f(x"))p(y|x") \end{aligned} \tag{21}$$

Set equal to 0 to find the minimum error, thus the optimal solution.

$$\begin{aligned} 0 &= \sum_{y \in Y} yp(y|x") - \sum_{y \in Y} f(x")p(y|x") \\ f(x") &= \sum_{y \in Y} yp(y|x") \\ &= \mathbb{E}[y|x] \end{aligned} \tag{22}$$

Now we set the function $F(x) = |x|^p$, and derive the Bayes estimator similar to that of above.

$$\begin{aligned} L_f(y, f(x)) &= |f(x)|^p - |y|^p + \frac{py^p(y - f(x))}{|y|} \\ \epsilon(f) &= \sum_{x \in X} [\sum_{y \in Y} \left(|f(x)|^p - |y|^p + \frac{py^p(y - f(x))}{|y|} \right) p(y|x)] p(x) \\ \epsilon(f(x")) &= \sum_{y \in Y} \left(|f(x")|^p - |y|^p + \frac{py^p(y - f(x"))}{|y|} \right) p(y|x")p(x") \end{aligned} \tag{23}$$

Let's make the algebra simpler, $e := \epsilon(f(x"))$, $z := f(x")$, replace $=$ with $\propto$ and remove the $p(x")$.

$$
\begin{aligned}
e &\propto \sum_{y\in Y}\left(|z|^p - |y|^p + \frac{py^p(y-z)}{|y|}\right)p(y|x") \\
\frac{\partial e}{\partial z} &= \sum y \in Y\left(pz|z|^{p-2} - \frac{py^p}{|y|}\right)p(y|x") \\
0 &= \sum_{y\in Y} pz|z|^{p-2}p(y|x") - \sum_{y\in Y}\frac{py^p}{|y|}p(y|x") \\
\sum_{y\in Y}\frac{py^p}{|y|}p(y|x") &= pz|z|^{p-2} \\
\mathbb{E}\left[\frac{py^p}{|y|}|x\right] &= pz|z|^{p-2} \\
\mathbb{E}\left[\frac{py^p}{|y|}|x\right] &= pf(x)|f(x)|^{p-2}
\end{aligned} \tag{24}
$$

The interesting case is when $p > 1$, the Bayes estimator suffices when $y = f(x)$, and vice versa with negative values for y and $f(x)$.

2. Consider the Kernel function $K_c(x,z) := c + \sum_{i=1}^n x_i z_i$, where $x, z \in \mathbb{R}$.

   (a) Prove for what values of $c \in \mathbb{R}$, K_c is a positive semidefinite kernel.
   We can prove a kernel is positive semidefinite by the inner product in a Hilbert space

$$
\begin{aligned}
\sum_{i=1}^n\sum_{j=1}^n a_i a_j k(x_i,x_j) &= \sum_{i=1}^n\sum_{j=1}^n \langle a_i\phi(x_i), a_j\phi(x_j)\rangle \\
&= \langle\sum_{i=1}^n a_i\phi x_i, \sum_{j=1}^n a_j\phi(x_j)\rangle \\
&= \|\sum_{i=1}^n a_i\phi(x_i)\|^2 \geq 0
\end{aligned} \tag{25}
$$

   We can define our kernel as an polynomial kernel $k_c(x,z) := (c + x^T z)^r$ where $r = 1$, and can prove the kernel is positive semidefinite when $c \geq 0$. Note: the change of c to t for the kernel.

$$
\begin{aligned}
\sum_{i,j}^m c_i c_j k_t(x_i,z_j) &= \sum_{i,j}^m c_i c_j (t + x_i^T z_j)^r \\
&= \sum_{i,j}^m c_i c_j (t + \sum_k^m x_{ik} z_{jk})^r \\
&= \sum_{i,j}^m c_i c_j t + \sum_{i,j}^m (\sum_k^m x_{ik} z_{ik}) c_i c_j \\
&= t\|c\|^2 + \|ck_t\|^2 \geq 0
\end{aligned} \tag{26}
$$

   (b) Suppose we perform linear regression with the kernel K_c. Our value of c includes a offset (bias) term to the regression. It is used to reimburse values that are not centered around the origin, and can help to transform single dimension linear regression into higher multi linear regression.

3. We perform a linear regression with a Gaussian Kernel $k_\beta(x,t) = exp(-\beta\|x-t\|^2)$ to train a classifier on a data set $(x_1,y_1)...(x_m,y_m) \in \mathbb{R}^n \times \{-1,1\}$. For which a function is in the form of $f(t) =$

$\sum_{i=1}^{m} \alpha_i K_\beta(x_i, t)$. In what scenario will a chosen β selected for the kernel cause the linear classifier to simulate that of 1-Nearest Neighbour classification.

We will assume the value xnn is the x parameter for the 1NN classification such that this is point closest point when a formula such as Euclidean distance. We can simulate 1NN by producing a value for the kernel regression using xnn, tnn.

$$\begin{aligned} \sum_{i=1}^{m} \alpha_i K_\beta(xnn, tnn) &= \sum_{i=1}^{m} \alpha_i K_\beta(x, t) \\ \sum_{i=1}^{m} \alpha_i exp(-\beta \|xnn - tnn\|^2) &= \sum_{i=1}^{m} \alpha_i exp(-\beta \|x - t\|^2) \end{aligned} \tag{27}$$

For the rest of the algebra we will omit $\sum_{i=1}^{m} \alpha_i$, and multiply all sides by log.

$$-\beta \|xnn - tnn\|^2 = -\beta \|x - t\|^2 \tag{28}$$

To find the value of β, this becomes a problem of the difference between the 1NN solution and the regular kernel solution. The kernel regression simulates that of 1NN when the difference is zero. We can define the value of β to be a function of $(x_1, ..., x_m, t)$ such that

$$f((x_1, ..., x_m), t) := \left[(\|xnn - tnn\|^2) - (\|x - t\|^2) = 0 \right] \tag{29}$$