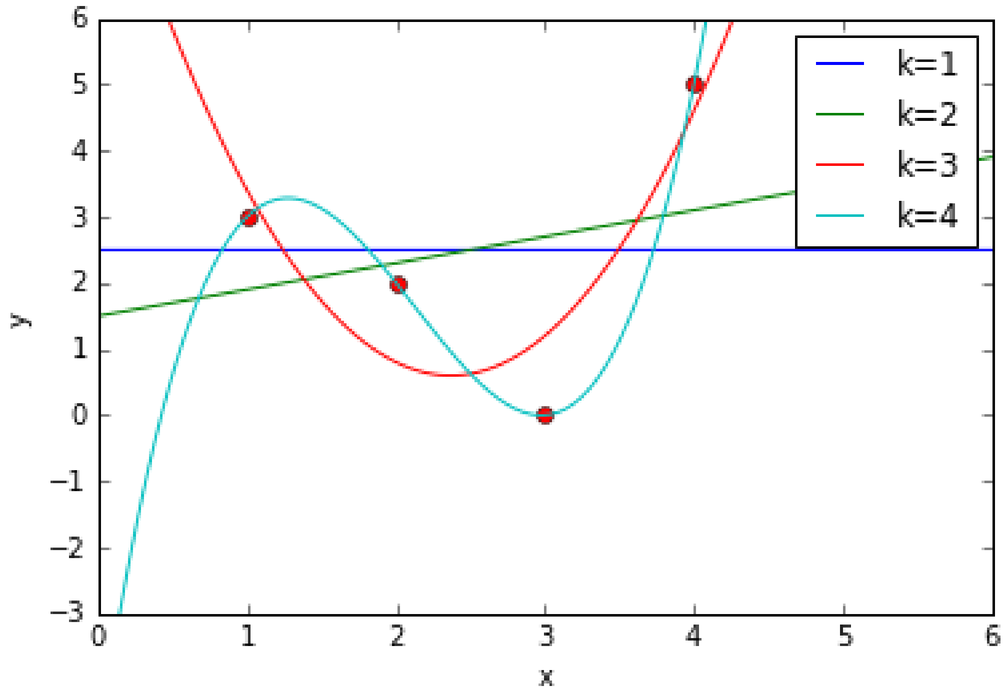
Supervised Learning: Coursework 1

Matineh Akhlaghinia, Abdul Faiz Punakkath

November 14, 2018

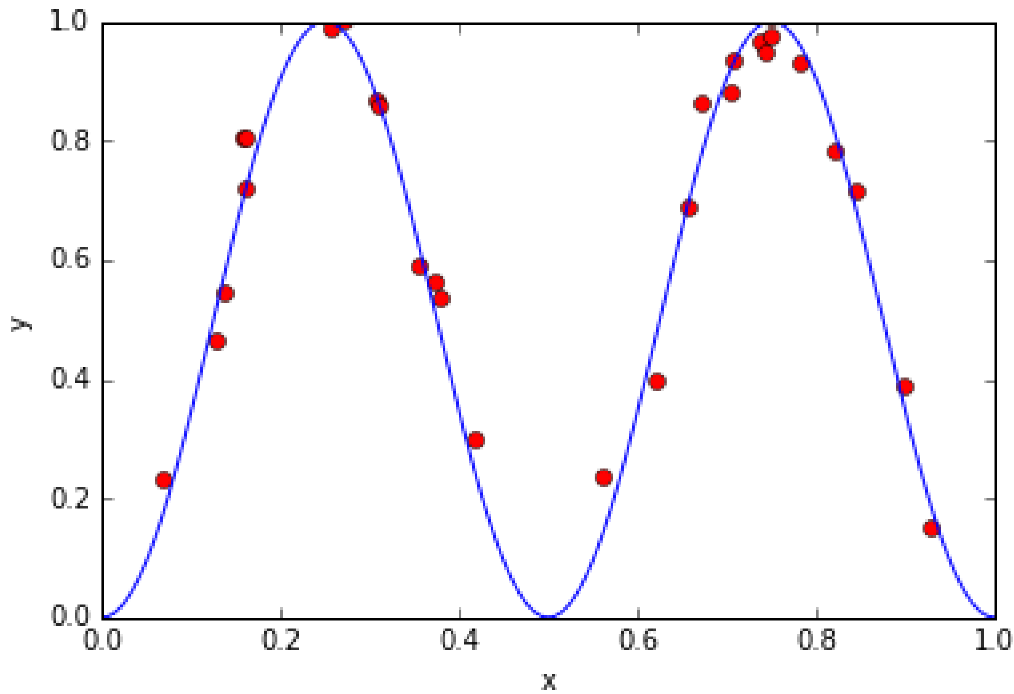
1. (a)



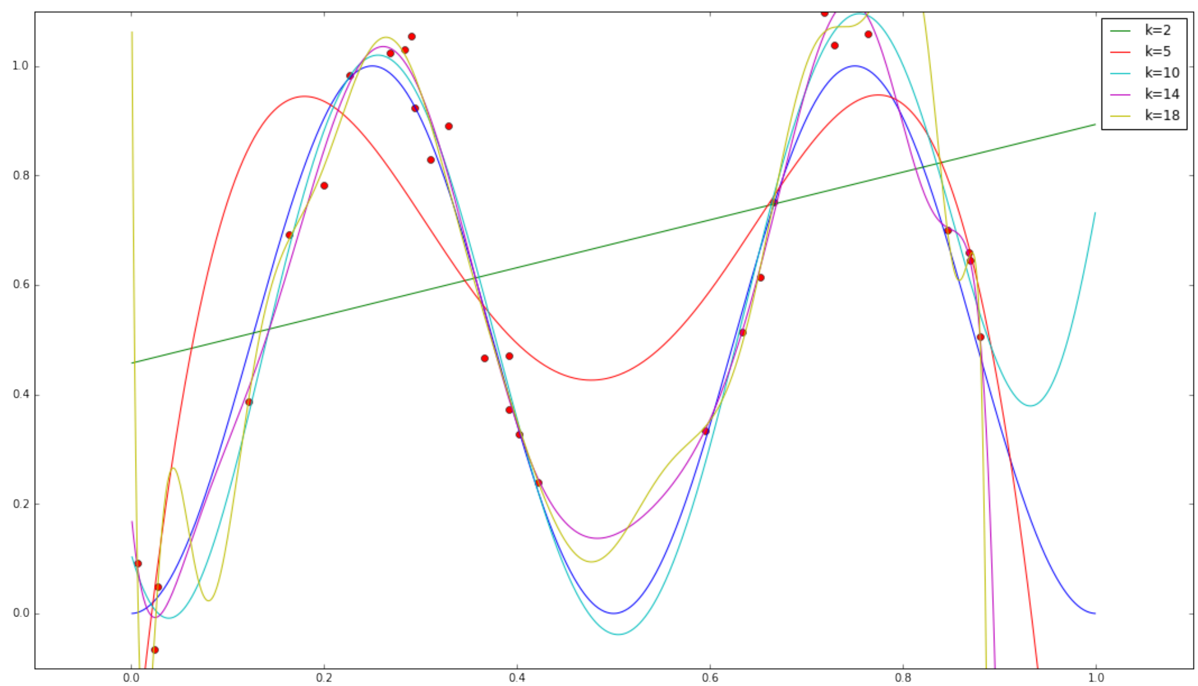
   (b) $k = 1 : 2.5$
   $k = 2 : 1.5 + 0.4 * x$
   $k = 3 : 9 - 7.1 * x + 1.5 * x^2$

   (c) $k = 1 : 3.25$
   $k = 2 : 3.0499999999999994$
   $k = 3 : 0.7999999999999998$
   $k = 4 : 6.589355141311112 * 10^-27$

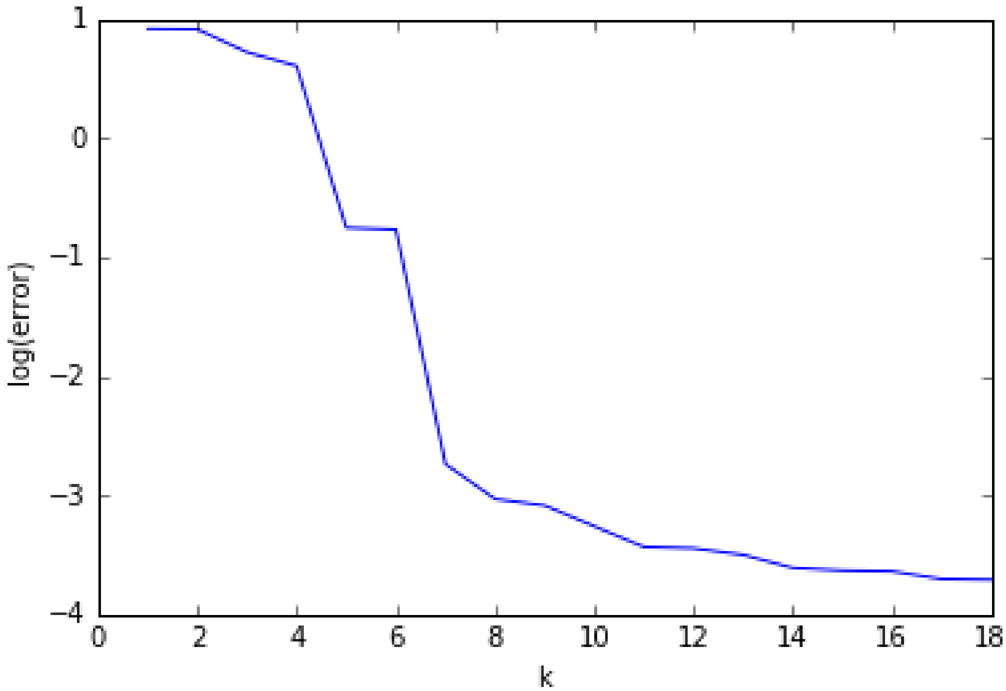
2. (a) i.



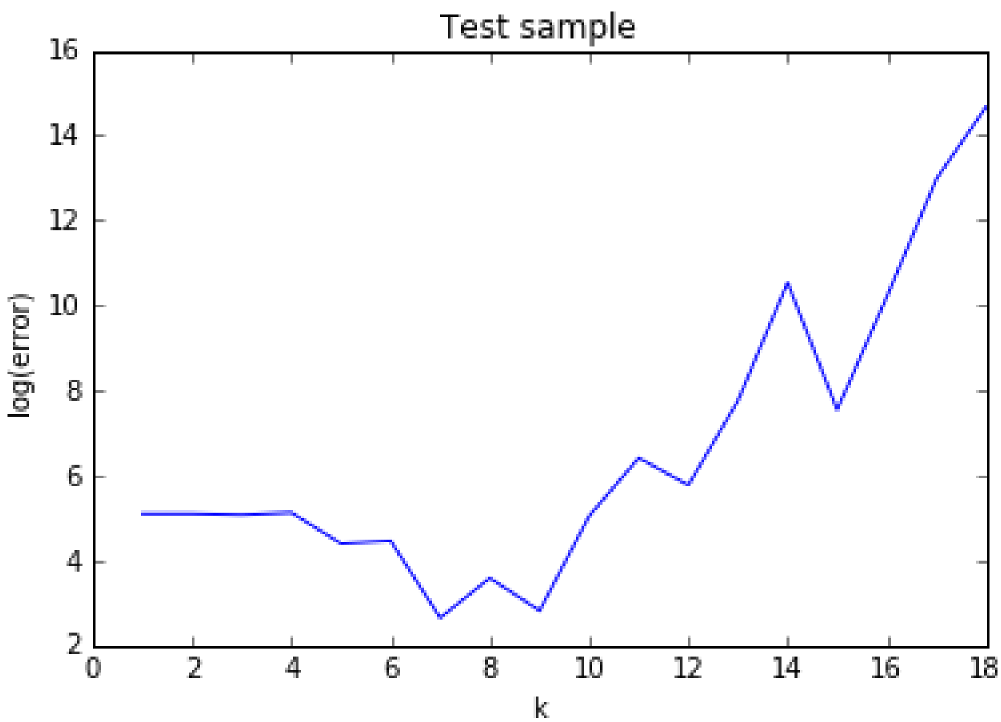
ii.



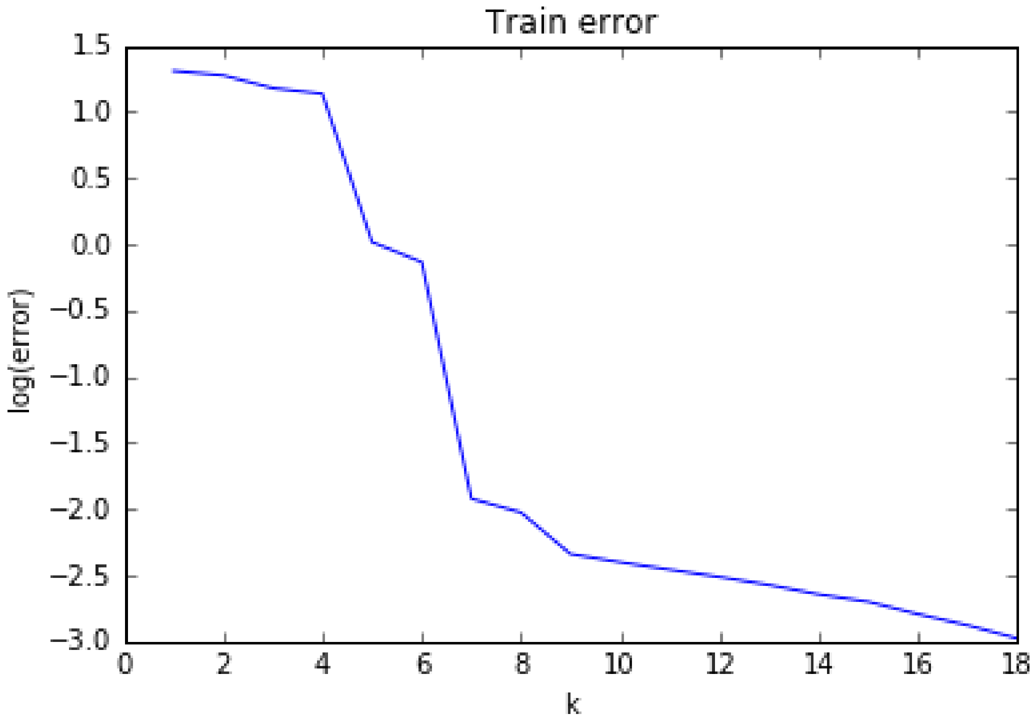
(b)



(c)

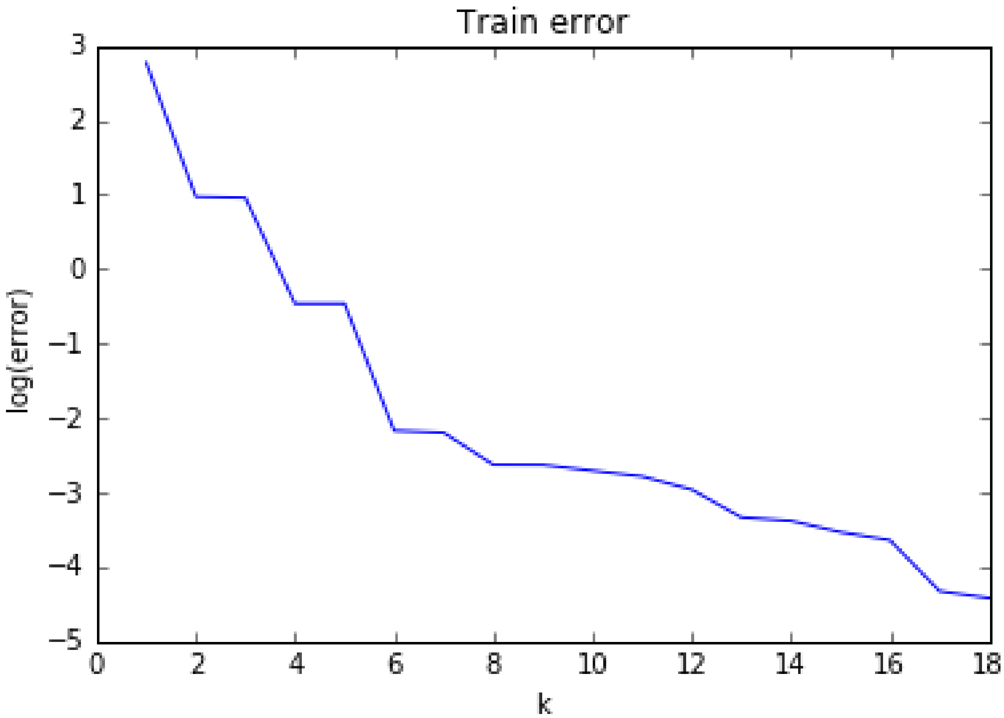


(d)

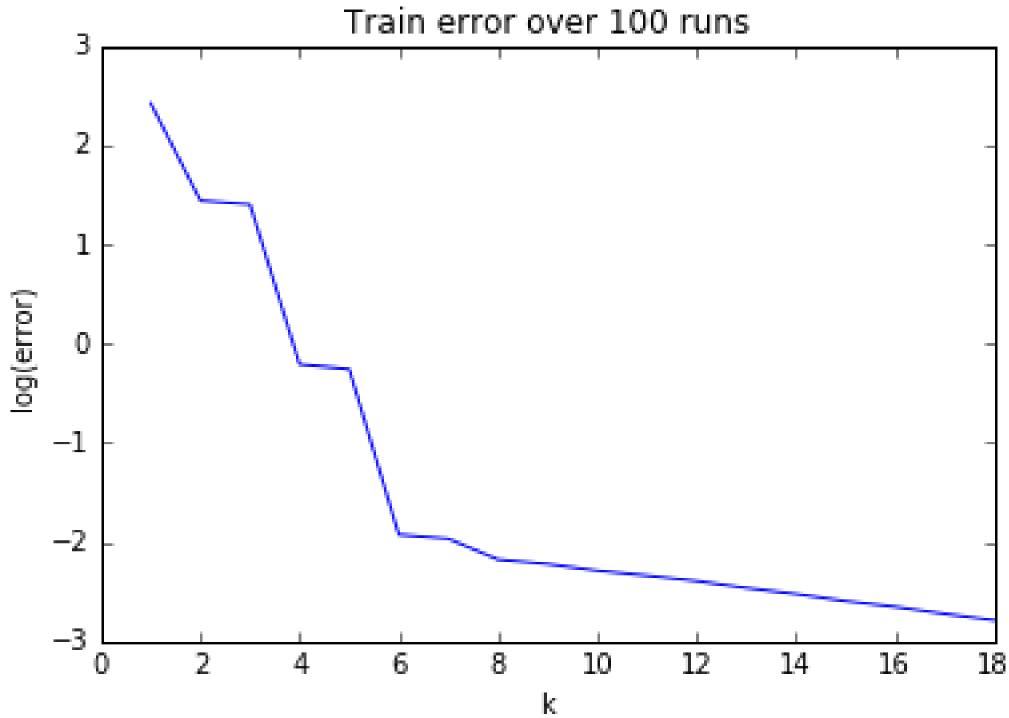

Train error
log(error)
k
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
0
2
4
6
8
10
12
14
16
18


Test error
log(error)
k
12
10
8
6
4
2
0
2
4
6
8
10
12
14
16
18

3.


Train error
log(error)
3
2
1
0
−1
−2
−3
−4
−5
0 2 4 6 8 10 12 14 16 18
k


Test error
log(error)
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
0 2 4 6 8 10 12 14 16 18
k



4. (a) $Train\ MSE = 74.94491097922847$
$Test\ MSE = 130.8907648716782$

(b) Constant function in the above question will be the average of all the outputs. Since $f(x) = (average of all the house prices)$ will achieve the minimum MSE for a constant function.

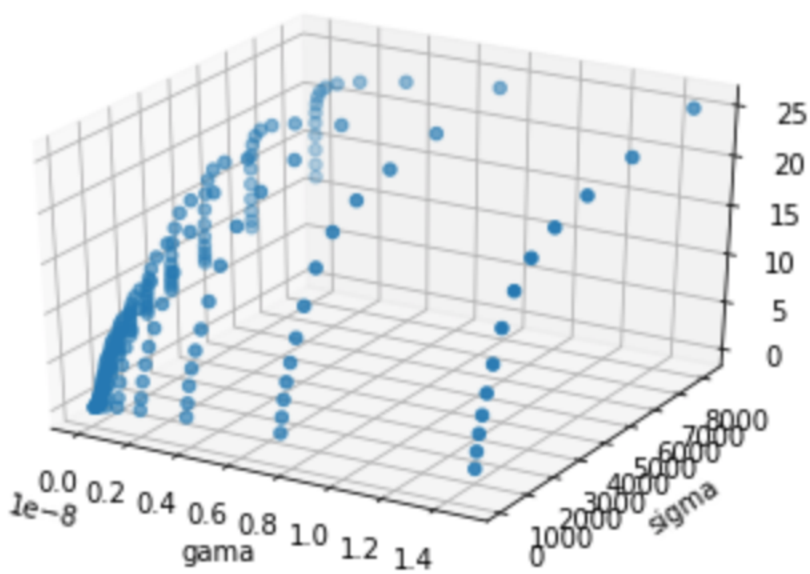
Attributes	Train MSE	Test MSE
1: CRIM	71.13968241182597	73.85021859717854
2: ZN	73.11604863223906	74.76428861359675
3: INDUS	64.47064841447379	65.39468687518753
4: CHAS	81.16463035498536	83.8633011280258
5: NOX	68.81917327442122	69.78392360937791
6: RM	43.78222803786089	43.740061995511425
7: AGE	73.88134430479434	70.01600700106209
8: DIS	78.85716721087809	80.20163857248215
9: RAD	72.34832512917372	72.08181955662278
10: TAX	65.5366146679282	67.04818858708646
11: PTRATIO	62.73194016874686	62.89616046358227
12: BLACK	73.68884487908194	78.03805991786986
13: LSTAT	38.44191281643897	38.89767608243805

(c)

(d) $Train\ MSE = 21.11943099906709 // Test\ MSE = 25.155269280832382$

5. (a) The best sigma and gama are 128.0 and 2^{-40}, respectively.

(b)



(c) Train MSE = 0.029
Test MSE = $2.23 * 10^{-6}$

(d)

Method	MSETrain	MSETest
Naive Regression	84.88 ± 4.7	82.90 ± 9.4
Linear Regression (attribute 1 (CRIM))	66.02 ± 4.1	64.12 ± 7.1
Linear Regression (attribute 2 (ZN))	69.72 ± 4.8	68.47 ± 9.1
Linear Regression (attribute 3 (INDUS)	59.83 ± 3.4	61.52 ± 6.8
Linear Regression (attribute 4 (CHAS))	80.37 ± 3.2	83.49 ± 6.2
Linear Regression (attribute 5 (NOX))	66.94 ± 4.3	71.82 ± 9.3
Linear Regression (attribute 6 (RM))	36.75 ± 2.8	39.19 ± 5.3
Linear Regression (attribute 7 (AGE))	69.32 ± 5.7	71.55 ± 11.7
Linear Regression (attribute 8 (DIS))	73.31 ± 6.4	78.22 ± 12.8
Linear Regression (attribute 9 (RAD))	69.27 ± 4.8	67.87 ± 9.8
Linear Regression (attribute 10 (TAX))	57.52 ± 2.6	58.17 ± 5.8
Linear Regression (attribute 11 (PTRATIO))	61.03 ± 3.8	62.43 ± 7.8
Linear Regression (attribute 12 (BLACK))	67.79 ± 4.2	71.18 ± 8.6
Linear Regression (attribute 13 (LSTAT))	27.69 ± 1.8	26.45 ± 4.8
Linear Regression (all attributes)	0.0074 ± 0.041	$3.41 * 10^{-5} \pm 6.1 * 10^{-4}$
Kernel Ridge Regression (all attributes)	0.012 ± 0.34	$1.21 * 10^{-5} \pm 4.3 * 10^{-4}$

6. (a)

$$\mathcal{E}(f) = E[L_c(y,\hat{y})] = E[[y \neq \hat{y}]c_y] = \sum_{x\in X}\sum_{y\in Y}[y \neq f(x)]c_y p(x,y)$$

Applying Bayes rule, we can write:

$$\mathcal{E}(f) = \sum_{x\in X}\left[\sum_{y\in Y}[y \neq f(x)]c_y p(y|x)\right]p(x)$$

Now let's find the value of the Bayes Estimator at a specific point x = x':

$$\mathcal{E}(f(x')) = \left[\sum_{y\in Y}[y \neq f(x')]c_y p(y|x')\right]p(x')$$

As we can see if for any y, $y \neq f(x)$ the cost would be c_y, hence we want to choose $f(x')$ in a way that would increase the number of times we get $y = f(x')$ to have more 0 costs, which would give us the minimum error. As a result, the Bayes Estimator $f(x')$ is mode of the probability distribution.

(b)

$$\mathcal{E}(f) = E[L(y,\hat{y})] = E[|y-\hat{y}|] = \int |y - f(x)|\ dP(x,y)$$

In order to derive Bayes Estimator, we need to minimize the expected error. First we need to apply Bayes rule:

$$\mathcal{E}(f) = \int_{x\in X}\left\{\int_{y\in Y}|y - f(x)|\ dP(y|x)\right\}dP(x)$$

Then we need to find the value of $f(x')$ at a fixed point x=x':

$$\mathcal{E}(f(x')) = \int_{y \in Y} \{|y - f(x')| \; dP(y|x')\} \, dP(x')$$

$$= \int_{y<f(x')} (f(x') - y) \; dP(y|x') \; dP(x') + \int_{y \geq f(x')} (y - f(x')) \; dP(y|x') \; dP(x')$$

Now let's assume $z = f(x')$, we need to differentiate $\mathcal{E}(f(x')$ w.r.t z in order to find the Bayes Estimator:

$$\frac{\partial e}{\partial z} = \int_{y<z} dP(y|x') - \int_{y \geq z} dP(y|x') = 0$$

$$\int_{y<z} dP(y|x') = \int_{y \geq z} dP(y|x')$$

As $P(y|x')$ is a distribution,

$$\int_{y \in Y} dP(y|x') = 1$$

Hence we can conclude that:

$$\int_{y<z} dP(y|x') = \int_{y \geq z} dP(y|x') = 1/2$$

So the Bayes Estimator is the median of the probability distribution $P(y|x')$.

7. (a) We have:

$$K_c(\mathbf{x}, \mathbf{z}) = c + \sum_{i=1}^{n} x_i z_i$$

where $x, z \in \mathbb{R}^n$

K_c would be positive semidefinite if and only if:

$$K_c(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$$

if $\phi(x) = (x_{i_1} x_{i_2} x_{i_{13}} ... x_{i_n})$ then $\sum_{i=1}^{n} x_i z_i = (x^T z)$ which is already a positive semidefinite by definition. So for any $c >= 0$, K_c is positive semidefinite.

(b) c will act as a bias term if K_c is used as kernel for linear regression.
$\epsilon(f) = \sum_{i=1}^{l} (\sum_{j=1}^{l} \alpha_j (x_i, x_j) + c\alpha_j - y_i)^2$ where f is predictor.
As you can see, in the expected error, c acts as a bias term when we use K_c for linear regression.

8. As $\beta \to \infty$, Guassian kernel K_β starts to simulate 1-Nearest Neighbour. We illustrate this below by defining the classifier as follows,

$$f(x) = \begin{cases} -1 & \sum_{i=1}^{m} \alpha_i K_\beta(x_i, x) < 0 \\ 1 & otherwise \end{cases}$$

Since $K_\beta(x_i, x_j) = \exp(-\beta||x_i - x_j||^2)$, as $\beta \to \infty$, $K_\beta = 0$ if $i \neq j$ and $K_\beta = 1$ if $i = j$. Which means, if there are no duplicate points, all the other points except $i = j$ becomes negligible.

From the notes,

$$\alpha_i = \frac{1}{2\lambda} V' \left(y_i, \sum_{j=1}^{m} \alpha_j K_\beta(\mathbf{x}_i, \mathbf{x}_j) \right)$$

so α_i becomes $\alpha_i = \frac{1}{2\lambda} V'(y_i, \alpha_i)$
$\alpha_i = \frac{1}{\lambda}(\alpha_i - y_i)$

hence $\alpha_i = k y_i$

We also know that $\forall ij, \alpha_i = \alpha_j$

Now we can modify our classifier to be,

$$f(x) = \begin{cases} -1 & \sum_{i=1}^{m} k y_i K_\beta(x_i, x) < 0 \\ 1 & otherwise \end{cases}$$

Our classifier needs to predict y_p if x is closer to x_p. Which means,

$$|y_p K_\beta(\mathbf{x}_p, \mathbf{x})| > \sum_{i \in \{1,..,m\} \setminus \{p\}} |y_i K_\beta(\mathbf{x}_i, \mathbf{x})|$$

Since as $\beta \to \infty$, $|y_p K_\beta(\mathbf{x}_p, \mathbf{x}_p)| \to \infty$ and $|y_i K_\beta(\mathbf{x}_i, \mathbf{x}_j)| \to 0$, our classifier $f(x)$ will predict y_p when x is closer to x_p. Hence as $\beta \to \infty$, our Guassian kernel K_β simulate a 1-NN.

9. Let's represent the initial configuration of a Whack-A-Mole as a vector of 1s and 0s, which 1 represents a mole being present in the square and 0 represents an empty square without a mole. For example, for the configuration given in the question the initial representation of the grid would look like: INITIAL = (0, 1, 0, 0, 1, 1, 1, 0, 0, 0 , 0, 1, 1, 1, 0, 1)

   As we want to empty the grid from any moles, we want our final configuration to be: FINAL = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 , 0 , 0, 0)

   We will also represent the moves that need to be made to solve the Whack-A-Mole problem as a vector of 1s and 0s, where 1s would represent whacking a mole and 0 not whacking a mole. This way, we can represent all the possible moves and its affects in a matrix where each row (i) would represent the affect of a hit in the ith square in the grid and the columns of the matrix would represent the squares that are affected by that move.

For example: $moves = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}$

Now we can solve Whack-A-Mole by solving the below equation:

$$MOVES * X + INITIAL = FINAL \tag{1}$$

Where X represents the moves that we need to actually make to reach to the final configuration of all squares being 0. Which means that if there's a solution to this equation, there exist a series of moves that would give us the final configuration.

$$X = MOVES^{-1} * (FINAL - INITIAL) \tag{2}$$

Computational complexity of this equation is polynomial. Computing the inverse of a $n * n$ matrix is $O(n^3)$ and matrix multiplication is $O(n^2)$. So the overall complexity would be: $O(n^3 + n^2) \approx O(n^3)$