1)

for degree 1 the mean square error for test error is 10669.73627867426

for degree 2 the mean square error for test error is 226.03678502947685

for degree 3 the mean square error for test error is 1.1148186421253086

for degree 4 the mean square error for test error is 1.0835919472260966

for degree 5 the mean square error for test error is 1.055942391195368

for degree 6 the mean square error for test error is 1.052832803413073

for degree 7 the mean square error for test error is 1.0497692253381574

for degree 8 the mean square error for test error is 1.0096300009101462

for degree 9 the mean square error for test error is 0.9991985572231676

for degree 10 the mean square error for test error is 1.1888350894539637

----

for degree 1 the mean square error for train error is 13098.316982179042

for degree 2 the mean square error for train error is 382.5514347582968

for degree 3 the mean square error for train error is 0.9169436488781806

for degree 4 the mean square error for train error is 0.9158162265403782

for degree 5 the mean square error for train error is 0.9119222674745413

for degree 6 the mean square error for train error is 0.8851638909059292

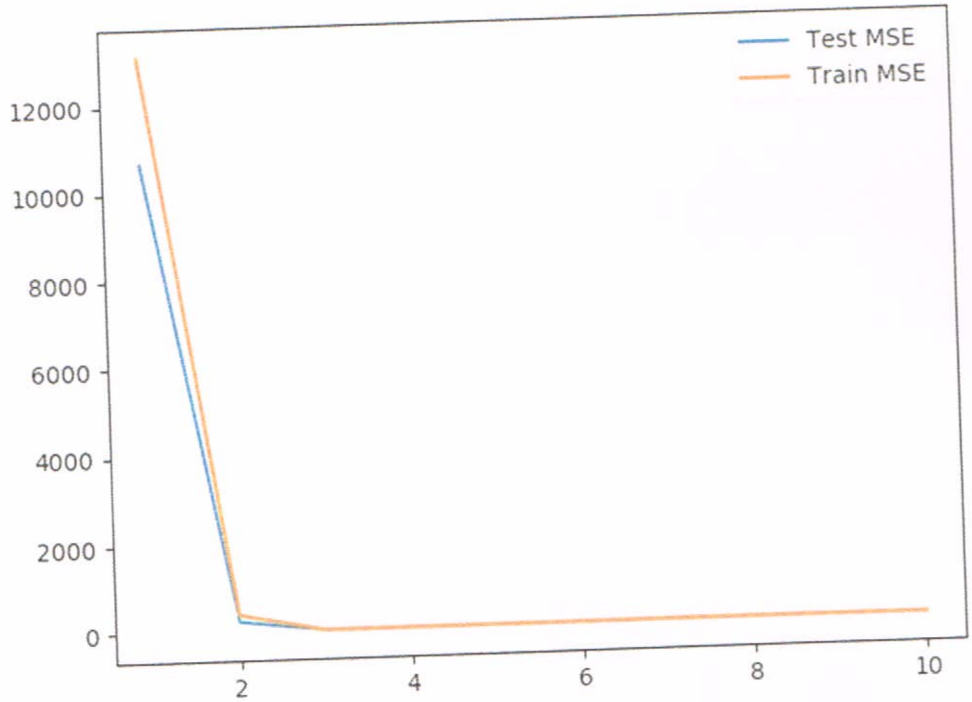
for degree 7 the mean square error for train error is 0.8375178292342661

for degree 8 the mean square error for train error is 0.8272856463221285

for degree 9 the mean square error for train error is 0.8322388494881415

for degree 10 the mean square error for train error is 0.8432616296374058

9) continued



Pick degree 3, as in d=3 as from the graph above, and the values, it drops significantly to d=3 then stays steady throughout.

2)

A) $y = f(x) = \sum_{r=0}^{R} w_r b_r(x)$

$$= \sum_{r=0}^{R} w_r \underbrace{\exp\left(\frac{-(x-c_r)^2}{2\sigma^2}\right)}_{b_r(x)}$$

For 1D inputs, $b(x) = [b_0(x), b_1(x), \ldots b_r(x)]^T$

& $w = [w_0, \ldots, w_R]$

$\hookrightarrow y = f(x) = w \cdot b(x)$

there is column of 1's in the first or last column, scanner cut it off

$\underline{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$, $\underline{w} = \begin{bmatrix} w_0 \\ \vdots \\ w_R \end{bmatrix}$, $B = \begin{bmatrix} b_1(x_1) & b_2(x_1) & \cdots & b_R(x_1) \\ b_1(x_2) & b_2(x_2) & \cdots & b_R(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ b_1(x_N) & b_2(x_N) & \cdots & b_R(x_N) \end{bmatrix}$

Similar to polynomial basis,

$$E(w) = \sum_{i=1}^{N} \Big(y_i - \big(\sum_{r=0}^{R} w_r b_r(x)\big)\Big)^2$$

$y_i \in \underline{y}$ $w_r \in \underline{w}$ $b_r(x)$ represented by

$$E(w) = \| \underline{y} - B\underline{w} \|^2$$

$$E(w) = \begin{bmatrix} y_1 - \sum_{r=0}^{R} w_r \exp\left(\frac{-(x_1 - c_r)^2}{2\sigma^2}\right) \\ \vdots \\ y_N - \sum_{r=0}^{R} w_r \exp\left(\frac{-(x_N - c_r)^2}{2\sigma^2}\right) \end{bmatrix}$$

$$= \begin{bmatrix} y_1 - \left(w_0 \exp\left(\frac{-(x_1 - c_0)}{2\sigma^2}\right) + \cdots + w_R \exp\left(\frac{-(x_1 - c_R)}{2\sigma^2}\right)\right) \\ \vdots \\ y_N - \left(w_0 \exp\left(\frac{-(x_N - c_0)}{2\sigma^2}\right) + \cdots + w_R \exp\left(\frac{-(x_N - c_R)}{2\sigma^2}\right)\right) \end{bmatrix}$$

if we use linear algebra for $\| \underline{y} - B\underline{w} \|^2 \stackrel{?}{=} E(w)$

$(\underline{y} - B\underline{w})^T (\underline{y} - B\underline{w}) = E(w)$

$(\underline{y}^T - \underline{w}^T B^T)(\underline{y} - B\underline{w}) = E(w)$

$$\boxed{E(w) = \underline{y}^T \underline{y} - \underline{y}^T B\underline{w} - \underline{w}^T B^T \underline{y} + \underline{w}^T B^T B\underline{w}}$$

2.

B)

without ridge regression error is 27284.000000000004

without ridge regression error is 20327.070225222527

w/ ridge

better

3.

alpha = 0.01 with T = 0.0001 has iterations and fscore respectively as 0 and 0.6554621848739496

alpha = 0.01 with T = 0.9 has iterations and fscore respectively as 0 and 0.6554621848739496

```
C:/Users/Aydin Baradaran/Desktop/C11 new version/Q3_code.py:38: RuntimeWarning: divide by zero encountered in log
  loss = -np.mean(y_train*np.log(sig) + (1-y_train)*np.log(1-sig));
C:/Users/Aydin Baradaran/Desktop/C11 new version/Q3_code.py:38: RuntimeWarning: invalid value encountered in multiply
  loss = -np.mean(y_train*np.log(sig) + (1-y_train)*np.log(1-sig));
```

alpha = 0.5 with T = 0.0001 has iterations and fscore respectively as 0 and 0.6379310344827587

alpha = 0.5 with T = 0.9 has iterations and fscore respectively as 0 and 0.6379310344827587

alpha = 0.99 with T = 0.0001 has iterations and fscore respectively as 0 and 0.6379310344827587

alpha = 0.99 with T = 0.9 has iterations and fscore respectively as 0 and 0.6379310344827587

I would pick $\alpha = 0.01$ & $T = 0.0001$.
or $\alpha = 0.01$ & $T = 0.9$
depending on the #
of iterations (that
I couldn't get to work).