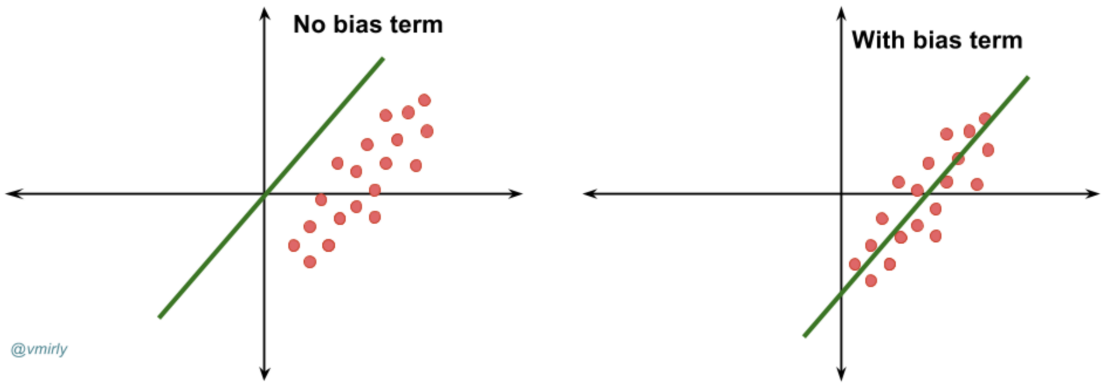
Assignment 1

Peter Park

260571481

Problem 1.

1 a)
In handling the data I added the bias term of 1 into the matrix $\mathbb{X}$. This is to deal with the fact in case the true model that describes the dataset does not pass through the origin. For reference please take a look at picture below



Now the L^2-norm regularization is given by

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

That is, the standard L^2-norm regularization does not penalize the bias term. This is because in the presence of adding arbitrary value c to all y_i, the bias term β_0 has to increase by c as well. This is not the case if β_0 is penalize and it will increase less than c. To achieve the most accurate model, ensuring L^2 regularization to not penalize the bias term is necessary. This was done by adding some constant c to output and using the same L^2-regularization to see if the value of the bias term had any change. Also to further deal with the scaling problem the input data was center by standardization, where all new input was given by

$$x_i^{new} = \frac{(x_i - \mu)}{\sigma}$$

The data was also shuffled to reduce bias error.

1 b)

It's clear that increasing the value of the regularization parameter has a significant effect: it reduces overfitting (as seen in the convergence of RMSE for training and test data in Figure 1), it dampens the overall magnitude of the predicted weight vectors, and decreases the magnitude of every individual weight. Looking at Figure 1) the ideal regularization parameter is 0 given that RMSE for both training and testing dataset is low and there is little difference between the two.

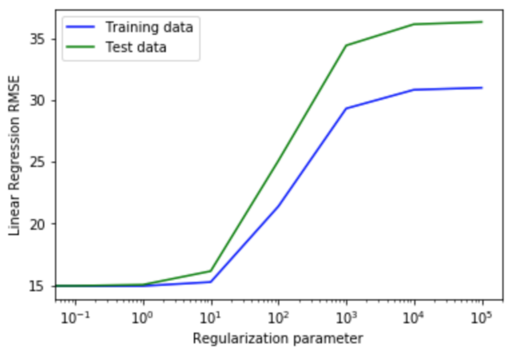


Figure 1: RMSE vs Regularization (not-scaled)

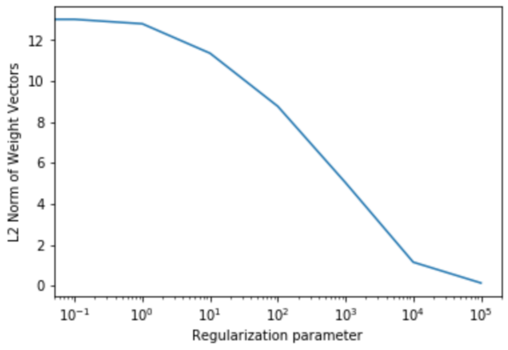


Figure 2: L_2 norm (not-scaled)

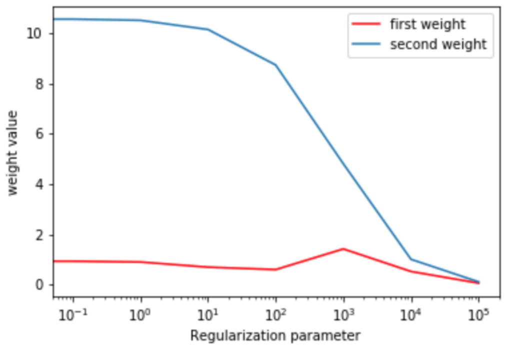


Figure 3: weights (not-scaled)

1 c)

The best value for λ I got using 5-fold cross-validation was 0.1 where as for 1(b) it was 0, although the difference in score is minimal. It seems that cross-validation gives a more rigorous measure for evaluating which regularization parameter is best and cross-validation is a good indicator of how well the model will likely generalize. Following is the detailed output of the experiment

```
For regularization parameter:  0.0
Training scores:  [ 13.53563629 15.59113941 14.70803841 16.47343471
15.71364099]
Average training score:  15.2043779614
Validation scores:  [ 21.28192818 14.15543063 17.56118625 9.48059086
13.64658025]
Average validation score:  15.2251432351

For regularization parameter:  0.1 Training scores:  [ 13.53569454
15.59119444 14.70810316 16.47347909 15.71369282]
Average training score:  15.2044328107
Validation scores:  [ 21.2905267 14.14509034 17.55693361 9.49091017
13.6405595 ]
Average validation score:  15.2248040636

For regularization parameter:  1.0
Training scores:  [ 13.54131016 15.59650369 14.71432828 16.47776941
15.71869727]
Average training score:  15.2097217601
Validation scores:  [ 21.37023137 14.05834316 17.52708441 9.58617926
13.59141503]
Average validation score:  15.226650645

For regularization parameter:  10.0
Training scores:  [ 13.97342616 16.00873959 15.18199425 16.81739565
16.11001425]
Average training score:  15.6183139824
Validation scores:  [ 22.31852874 13.7024454 17.85270779 10.68227485
13.50013578]
Average validation score:  15.6112185105

For regularization parameter:  100.0
Training scores:  [ 21.08866012 23.17513052 22.27428561 23.35176537
23.1495068 ]
Average training score:  22.6078696838
Validation scores:  [ 29.8593239 19.30711338 26.71799322 18.4583372
18.93193781]
Average validation score:  22.6549411011

For regularization parameter:  1000.0
Training scores:  [ 28.87775423 31.44282996 29.45797456 31.72551088
31.56960982]
Average training score:  30.6147358902
Validation scores:  [ 37.54490296 27.43982186 36.04815505 25.96041717
26.45561387]
Average validation score:  30.6897821802

For regularization parameter:  10000.0
Training scores:  [ 30.26757562 32.9472364 30.69398099 33.3202331
33.12769742]
Average training score:  32.071344706
Validation scores:  [ 38.92707253 28.94488366 37.65419042 27.31554457
27.86478792]
```

```
Average validation score:  32.1412958214

For regularization parameter:  100000.0
Training scores:  [ 30.41698235 33.10940756 30.82609636 33.4933384
33.29610169]
Average training score:  32.228385272
Validation scores:  [ 39.07589264 29.10731743 37.8258372 27.46180061
28.01722609]
Average validation score:  32.2976147965
```

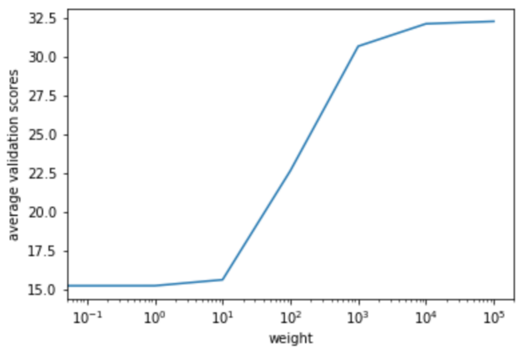
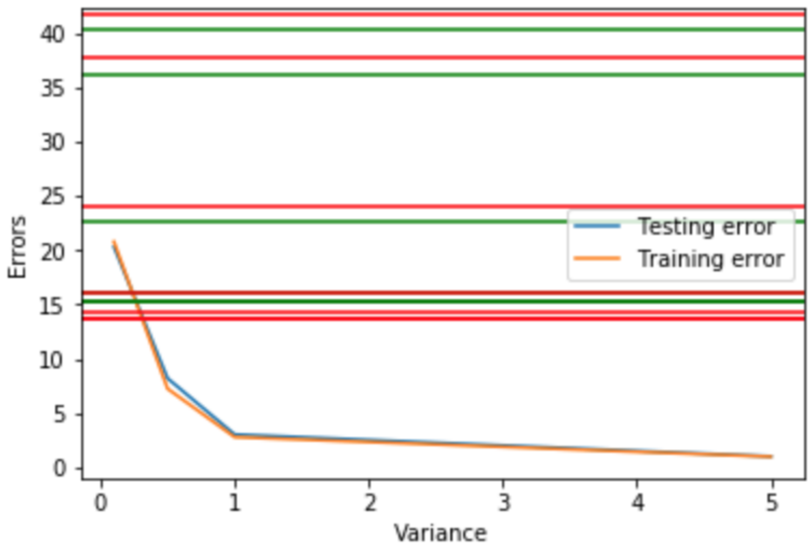


Figure 4: λ vs Average validation score

1 d)
Cross-validation would not be suitable such tasks. If we consider the scenario where the training folds are the 4 sets of smallest values for target variable $'y'$ and the validation fold is the set of largest values for $'y'$, Then the values of $'y'$ in the 80th+ percentile of the dataset are much bigger than the rest. Therefore, the model will be trained to perform well on a region of data where the values of $'y'$ are relatively low, and thus will not perform well on the validation region and the ridge regression estimator will have much greater variance.
1 e)

Please see Jupyter notebook for detail output.

1 f)
Increasing the value of σ^2 usually reduces the error of the model to certain point where it starts to level off. From running this experiment we get the following result.



Note that the green constant lines are the training errors obtained in 1(b), and the red constant lines are the testing errors obtained in 1(b).

1 g)

Let $X \in \mathbb{R}^{n\times m}$ be the design matrix and $y \in \mathbb{R}^n$ be the output variable. Then we have the following following cost function:

$$L(\omega,\mu) = \sum_{i=1}^{n}\{y_i - \sum_{j=1}^{m}\sum_{k=1}^{l} w_{jk}exp(-\frac{(x_i-\mu_i)^2}{2\sigma^2})\}^2 + \frac{\lambda}{2}\omega^T\omega$$

and

$$\frac{dL(\omega,\mu)}{d\mu_l} = 2\sum_{i=1}^{n}\{y_i - \sum_{j=1}^{m}\sum_{k=1}^{l} w_{jk}exp(-\frac{(x_i-\mu_i)^2}{2\sigma^2})\}(-\sum_{j=1}^{m}\omega_{jl}exp(\frac{-(x_i-\mu_l)^2}{2\sigma^2}))(\frac{2(x_i-\mu_l)}{2\sigma^2})$$

which simplifies to

$$2\sum_{i=1}^{n}(\{y_i - \sum_{j=1}^{m}\sum_{k=1}^{l} w_{jk}exp(-\frac{(x_i-\mu_i)^2}{2\sigma^2})\}(-\sum_{j=1}^{m}\omega_{jl}exp(-\frac{(x_i-\mu_l)^2}{\sigma^2}))(\frac{(x_i-\mu_l)}{\sigma^2}))$$

If we let $\mu = \begin{bmatrix}\mu_1\\ \mu_2\\ \mu_3\end{bmatrix}$, we can compute the placement of the basis functions by running gradient descent

$$\mu^{(k+1)} = \mu^{(k)} - \gamma\nabla H(\omega,\mu^{(k)})$$

To compute the weight vector ω from the data, we take the following steps:
Let $B \in \mathbb{R}^{n\times(m\times l)}$ represent the following transformed data matrix with $\mu_k's$ chose by the iterative algorithm described above. Then we have

$$H(\omega) = (Y - B\omega)^T(Y - B\omega) + \frac{\lambda}{2}\omega^T\omega$$

expanding this we get

$$H(\omega) = Y^TY - 2\omega^TB^TY\omega^TB^TB\omega + \frac{\lambda}{2}\omega^T\omega$$

The derivative of this is

$$\frac{dH(\omega)}{d\omega} = -2B^Ty + 2B^TB\omega + \lambda\omega$$

setting it to 0 and solving for ω gives

$$\omega = (B^TB + \lambda I)^{-1}B^Ty$$

1 h)

The algorithm converges to local minimum which is usually close to the global minimum. This is reasonably close to global minimum of the error function.

1 i)

Please look at the Jupyter file for detailed code. The test error using the optimal parameter found in cross-validation is : 7.729400375806702 for degree of 2. 9.300363042408447 for Ridge regression lambda 8.2620364776491 for Lasso regression lambda
The following is the result for the weights in L1 regression

```
lambda = 0.01
weights [-9.47828632e-01 1.90755341e+00 1.18212699e-01 -5.62033759e-
03 -1.61354591e-03 -1.25609800e-04 -2.67645988e-06 1.53536378e-06
3.83291223e-07]
intercept -3.6314658534209734

lambda =0.1
weights [-7.78358301e-01 1.83045509e+00 1.18405885e-01 -3.04004426e-
03 -1.78854610e-03 -1.31033461e-04 -3.19932607e-06 1.54450708e-06
3.89565036e-07]
intercept -3.5626280349081405

lambda = 1

weights [-0.00000000e+00 1.24189281e+00 1.84173131e-01 6.77278610e-
03 -2.96459088e-03 -1.79634575e-04 -5.54278880e-06 1.79259061e-06
4.54618042e-07]
intercept -2.82620211943647

lambda =10

weights [ 0.00000000e+00 0.00000000e+00 4.20630350e-01 3.47001378e-
02 -7.88900515e-03 -4.36624005e-04 -8.00800325e-06 3.65493281e-06
8.05819046e-07]
intercept -0.4879772934461464
```

The following is the result for the weights in L2 regression

```
lambda = 0.01
weights [-4.37826757e+00 2.89381010e+00 3.71093054e+00 -1.83670568e+00
-3.51270050e-01 3.99390939e-01 -9.81239921e-02 1.01469100e-02 -3.87854564e-
04]
intercept -2.778555601581484

lambda =0.1
weights [-4.17320740e+00 2.84650574e+00 3.57909492e+00 -1.77858389e+00
-3.40843890e-01 3.88513707e-01 -9.55927348e-02 9.89512013e-03 -3.78488583e-
04]
intercept -2.803086276668619

lambda = 1

weights [-2.78608552e+00 2.46179950e+00 2.69293759e+00 -1.36638376e+00
-2.74770108e-01 3.13792958e-01 -7.79413862e-02 8.12491687e-03 -3.12273576e-
04]
intercept -2.9339373413676917

lambda =10

weights [-3.51742367e-01 1.13806880e+00 1.13090421e+00 -4.23906681e-
01 -1.94208150e-01 1.63201177e-01 -3.97858783e-02 4.15691534e-03
-1.60233047e-04]
intercept -2.8100289866824646
```

We can see that for L2 regression as lambda gets higher the curve becomes more smooth with less apparent local minimum and maximum. Hence it generalizes to the data much better and reduces over fitting. For L1 we can see that less important feature (first order polynomial $\phi_1(x) = x$) reduces to zero. We can conclude that L1 undergoes variable selection to select out degree one polynomial as one of its features. The relationship between optimal degree and the weights is that the optimal degree tends to correlate the number of weights

present (that are not zero). This is due to the fact that Lasso selects out the polynomial that are not relevant in model fitting.
Below is the graph illustrating the fit of the models above

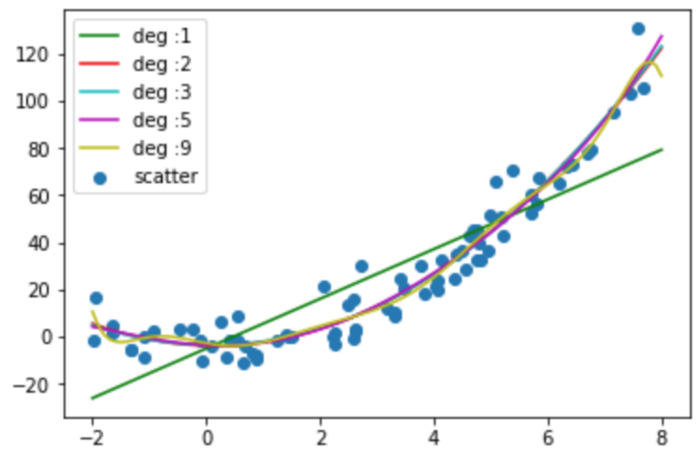


Figure 5:
degree vs RMSE

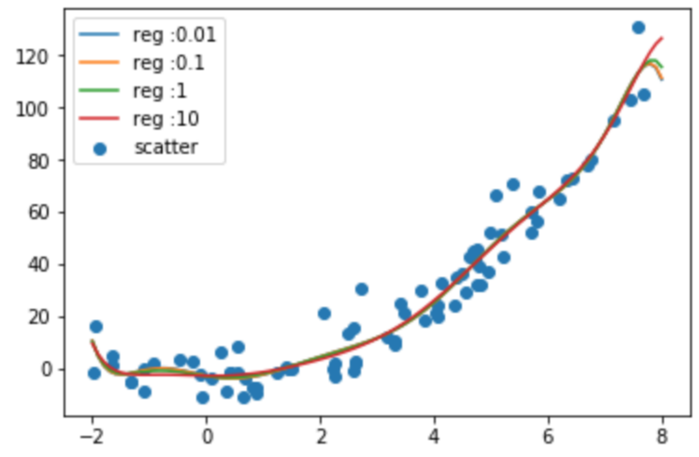


Figure 6: Regularization vs RMSE (Ridge)

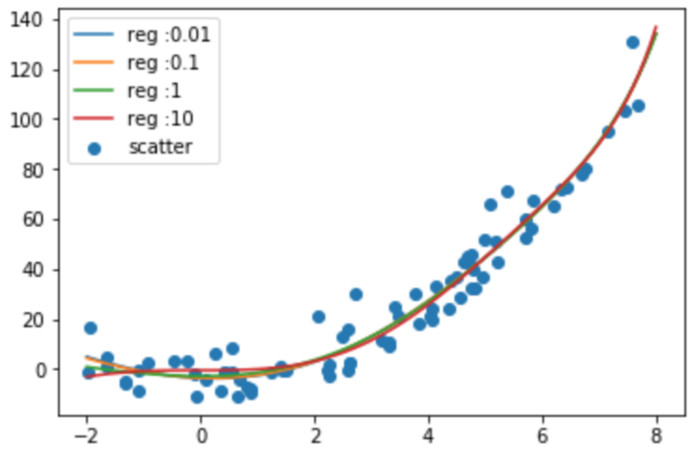


Figure 7: Regularization vs RMSE (lasso)

Problem 2.

We have

$$\begin{aligned} P(Y|X,\omega) &= \Pi_{i=1}^{m} P(y_i|x_i,\omega) \\ &= \Pi_{i=1}^{m} (\frac{1}{2\pi\sigma_i^2})^{\frac{1}{2}} exp(-\frac{1}{2\sigma_i^2}(y_i - \omega^T x_i)^2) \end{aligned}$$

hence

$$\begin{aligned} \log P &= \sum_{i=1}^{m} ((-\frac{1}{2})\log(2\pi\sigma_i^2) - \frac{1}{2\sigma_i^2}(y_i - \omega^T x_i)^2) \\ &= \frac{-m}{2}\log(2\pi\sigma_i^2) - \frac{1}{2}\sum_{i=1}^{m} \frac{(y_i - \omega^T x_i)^2}{\sigma_i^2} \end{aligned}$$

Taking the weight that would give the lowest value above is

$$\omega^* = \arg\max_{w} P(Y|X,\omega) = \arg\min_{\omega} \sum_{i=1}^{m} \frac{(y_i - \omega^T x_i)^2}{\sigma_i^2}$$

Taking the derivative of $P(Y|X,\omega)$ and setting to zero gives

$$0 = \sum_{i}^{m} \frac{x_i(y_i - \omega^T x_i)}{\sigma_i^2}$$

Let $X = \begin{bmatrix} X_1^T \\ X_2^T \\ \vdots \\ X_m^T \end{bmatrix}$ and $C = \begin{bmatrix} \frac{1}{\sigma_i^2} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \frac{1}{\sigma_m^2} \end{bmatrix}$

Then the above sum can be written as

$$X^T C(Y - X\omega) = 0$$

which is

$$X^T CY = X^T CX\omega$$

taking the inverse we get

$$\omega^* = (X^T CX)^{-1} X^T CY$$

Problem 3.

Let $X = (X_1, X_2, \cdots, X_N)$ which models N i.i.d coin flips where each $X_i = \begin{cases} 1 \text{ for } X_i = H \\ 0 \text{ for } X_i = T \end{cases}$
Then $P(X = x|\theta)$ becomes

$$\begin{aligned} &= \Pi_{i=1}^N P(X_i = x_i|\theta) \\ &= \Pi_{i=1}^N \theta^{x_i}(1-\theta)^{1-x_i} \end{aligned}$$

Then taking the logarithm, we have

$$\log P(X = x|\theta) = \sum_{i=1}^N (x_i \log(\theta) + (1 - x_i)\log(1-\theta))$$

Taking the derivative above we have

$$\frac{d \log P}{d\theta} = \sum_{i=1}^N \left[\frac{x_i}{\theta} - \frac{1-x_i}{1-\theta}\right]$$

setting to zero and solving for θ we have

$$\hat{\theta}_{ML} = \sum_{i=1}^N \frac{x_i}{N}$$

If $N = 3$ then $X = (HHH)$, which means we have $\hat{\theta}_{ML} = 1$, which seems overly optimistic. This does not seem like a good estimator and the prediction using this estimator seems to be unreasonable.
Now looking at the Bayesian analysis of θ, we have the following

$$posterior \propto prior \times likelihood$$

Now we know from looking at the situation that the beta distribution is

$$P(\theta|\alpha,\beta) = B(\theta;\alpha,\beta)$$

Then we have

$$P(\theta|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}$$

Now let R.V C model the number of observed heads. Then we have that the likelihood of C is

$$P(C=c|\theta) = \binom{N}{c}\theta^{c}(1-\theta)^{N-c}$$

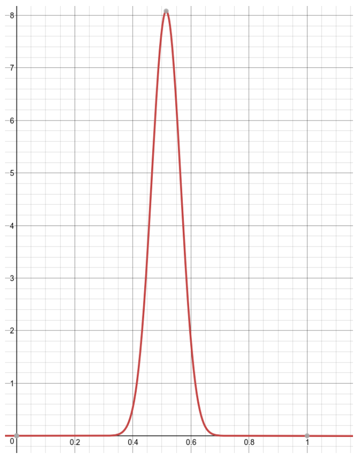
So then the posterior becomes

$$P(\theta|C,\alpha,\beta) \propto \theta^{\alpha+C-1}(1-\theta)^{\beta+N-C-1}$$

which after using the evidence we have

$$P(\theta|C,\alpha,\beta) = \frac{\Gamma(N+\alpha+\beta)}{\Gamma(C+\alpha)\Gamma(N-C+\beta)}\theta^{\alpha+C-1}(1-\theta)^{\beta+N-C-1}$$

Now the posterior is a Beta distribution, and the posterior mean is $\frac{C+\alpha}{N+\alpha+\beta}$ and the posterior mode is $\frac{C+\alpha-1}{N+\alpha+\beta-2}$ from looking at the table of distribution Now we consider $(\alpha,\beta) = (50,50)$. The posterior mean is given by $\frac{50+3}{3+50+50} = \frac{53}{103} \approx 0.5146$
This seems much more reasonable estimate for θ.
The posterior density if $P(\theta|C,\alpha,\beta) = \frac{\Gamma(103)}{\Gamma(53)\Gamma(50} * \theta^{52}(1-\theta)^{49}$
which gives the following density plot:



Problem 4.

a)

Now we have the following : $X = \begin{bmatrix} X_1^T \\ X_2^T \\ \vdots \\ X_m^T \end{bmatrix}$ and $Y = \begin{bmatrix} Y_1^T \\ Y_2^T \\ \vdots \\ Y_m^T \end{bmatrix}$ with $W = \left[w_1, w_2, \cdots, w_p\right]$

Then we have the following

$$\begin{aligned} J(\omega) &= \sum_{i=1}^{m} ||\omega x_i - y_i||_2^2 \\ &= \sum_{i=1}^{m} \sum_{j=1}^{(} p x_i \omega_j - y_{ij})^2 \end{aligned}$$

Which can be called as Frobenius norm of

$$= ||XW - Y||_F^2$$

Using the properties of the Frobenius norm we then have

$$\frac{dJ(W)}{dW} = 2X^T(XW - Y)$$

Setting this to zero and solving for W, we get

$$W^* = (X^T X)^{-1} X^T Y$$

b)

Let $Y = [Y_1, Y_2, \cdots, Y_p]$ Now again we have

$$\begin{aligned} J(W) &= ||W^T x_i - y_i||_2^2 \\ &= \sum_{i=1}^{m} \sum_{j=1}^{p} (x_i^T \omega_j - y_{ij})^2 && \text{from part (a)} \\ &= \sum_{j=1}^{p} \sum_{i=1}^{m} (x_i^T \omega_j - y_{ij})^2 && \text{interchanging the order of summation} \\ &= \sum_{j=1}^{p} ||XW_j - Y_j||_2^2 \end{aligned}$$

Now to minimize $J(W)$ w.r.t W, we differentiate $J(W)$ by W; equivalently we have

$$\begin{aligned} \frac{dJ(W)}{dW} &= [\frac{dJ(W)}{dW_1}, \frac{dJ(W)}{dW_2}, \cdots, \frac{dJ(W)}{dW_p}] \\ &= [\frac{d||XW_1 - Y_1||}{dW_1}, \frac{d||XW_2 - Y_2||}{dW_2}, \cdots, \frac{d||XW_p - Y_p||}{dW_p}] \end{aligned}$$

Now setting $\frac{dJ(W)}{dW}$ to zero we have

$$0_{d\times p} = [\frac{d||Xw_1 - Y_1||_2^2}{dW_1} \cdots \frac{d||Xw_p - Y_p||_2^2}{dW_p}]$$

We can see that then the solutions to these p equations above are just the solutions to p independent classical linear regression problems
Now performing independent regressions for each output variable is not the best thing to do when the output variables are highly correlated, (it would be a waste of time), since they would both affect the outcome similarly.
c) Let $W^T = U\Sigma V^T$ be the SVD of W^T, where $U \in \mathbb{R}^{p\times p}$, $V \in \mathbb{R}^{d\times d}$ where both these are orthogonal. Now, Σ is a diagonal matrix and its can be represented as

$$\Sigma = \begin{bmatrix} \theta_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \theta_R & \\ 0 & \cdots & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{p\times d}$$

and $\theta_1 \geq \theta_2 \geq \cdots \theta_R > \theta_{R+1} = \cdots = \theta_p = 0$ where we assume $(p < d)$

Now then we have $W^T = [u_1, u_2, \cdots, u_p] \begin{bmatrix} \theta_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \theta_R & \\ 0 & \cdots & \cdots & 0 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \\ \vdots \\ V_d^T \end{bmatrix}$ which becomes

$$\sum_{j=1}^{p} \theta_j u_j v_j^T$$

Now if $x \in \mathbb{R}^d$. Then we can write

$$x_i = \alpha_1 v_1 + \cdots \alpha_d v_d$$

where v_i's are columns of V. Since columns of V are an orthonormal basis of $\mathbb{R}^d$ we have

$$\begin{aligned} W^T x_i &= (\sum_{j=1}^{R} \theta_j u_j v_j^T)(\alpha_1 v_1 + \cdots + \alpha_d v_d) \\ &= \sum_{j=1}^{R} \alpha_j \theta_j u_j v_j^T v_j \quad v_j^T v_j \text{is 0 because orthogonal} \\ &= \sum_{j=1}^{R} \alpha_j \theta_j u_j \\ &= z_i \in \mathbb{R}^p \end{aligned}$$

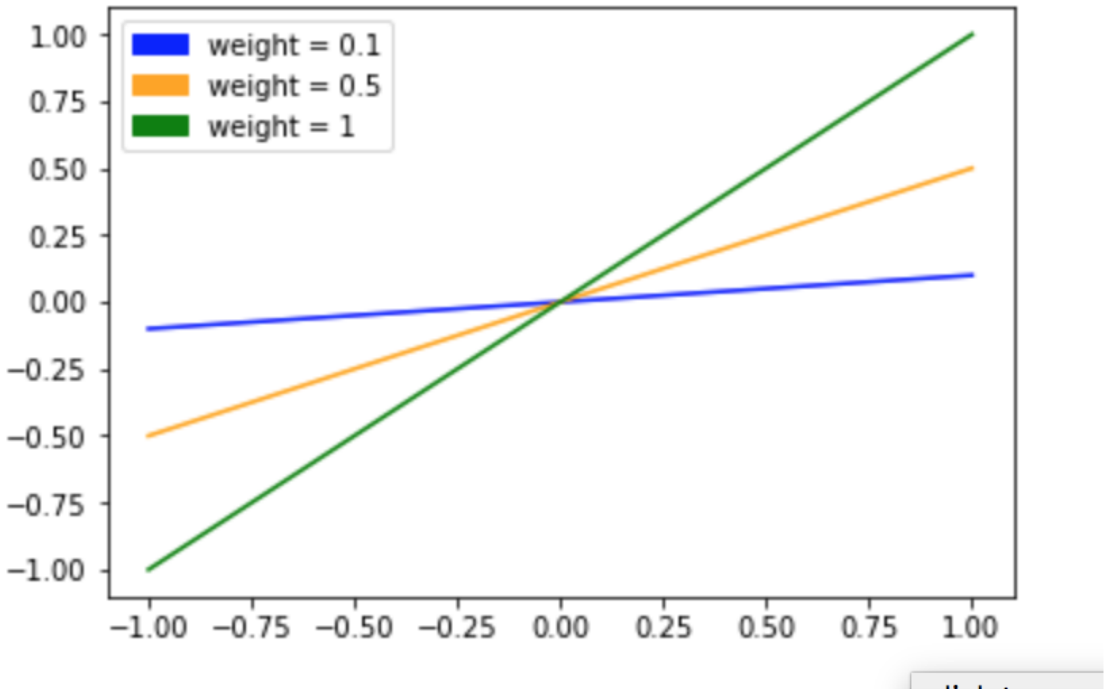
From this we deduce

$$\begin{aligned} z_i \in \text{Range}(\omega^T) &\Leftrightarrow \exists x_i \in \mathbb{R}^d s.t \quad \omega^T x_i = z_i \\ &\Leftrightarrow z_i \in span\{u_1, \cdots, u_R\} \end{aligned}$$

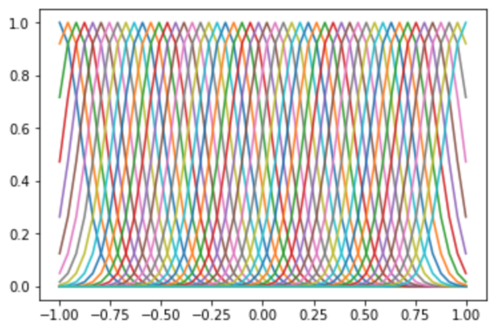
where $dim\text{Range}(\omega^T) = R$ and $\exists$ subspace of $\mathbb{R}^p$ with dimension $p-R$ that contains only noise

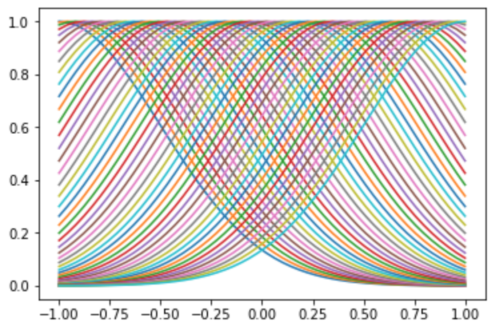
1 Problem 5.

a)
We then have the following norm with arbitrary weights
$w \in \{0.1, 0.5, 1\}$
and with $x \in [-1, 1]$, we generate the following function $f(x) = w^T x$ and the kernel is given by $k(x, x') = x^T x'$
Hence the following plot illustrates the above:

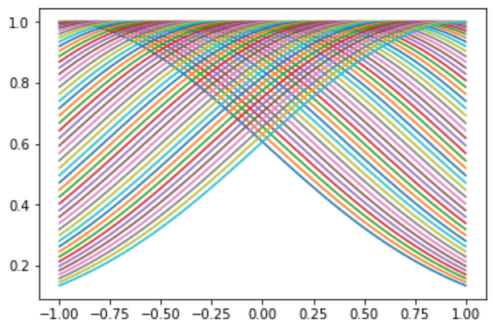


b)
We have the following plot for the Taylor expansion of the feature map up to $d = 150$.
For $\rho = 0.1$ For $\rho = 0.5$





For $\rho = 1$



From above we can say that taylor expansion of the feature map approximates the actual kernel very well, as these two sets of graphs seem almost identical. Taking the norm we get
c) Now we see the predictive formula with varying regularization parameters $\lambda \in \{0.1, 1, 10\}$

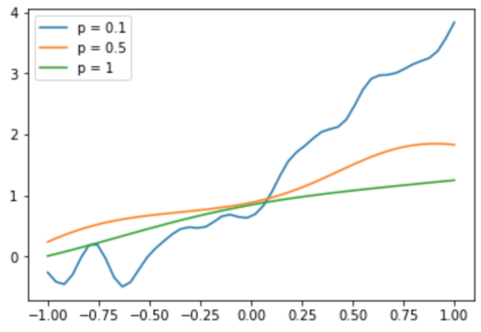


Figure 8: Norm using random weight vectors

With condition on N, we have

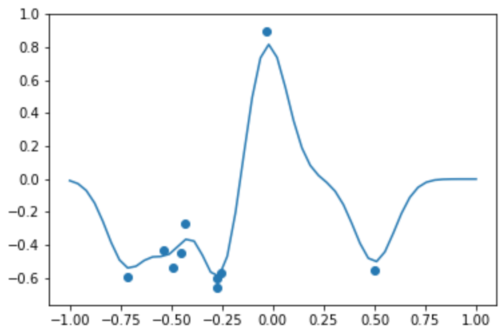


Figure 9: Conditioned on N data points, $\lambda = 0.1$

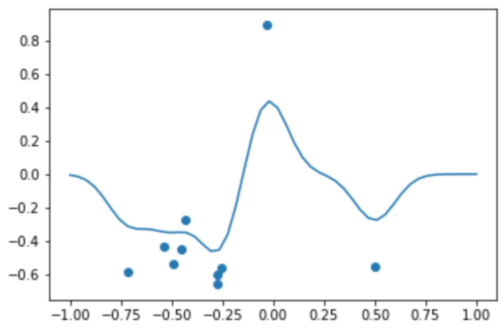


Figure 10: Conditioned on N data points, $\lambda = 1$

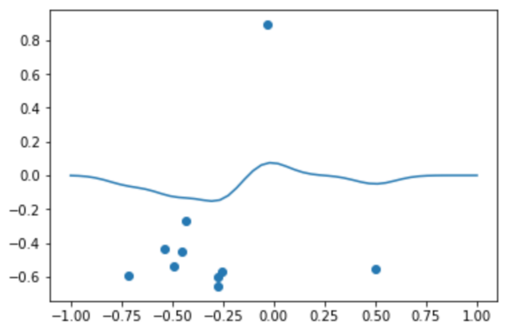


Figure 11: Conditioned on N data points, $\lambda = 10$

With condition on $N-1$, we have

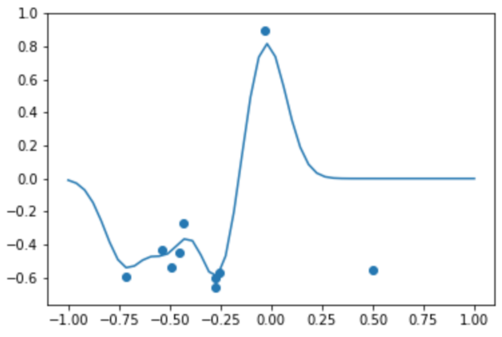


Figure 12: Conditioned on $N-1$ data points, $\lambda = 0.1$

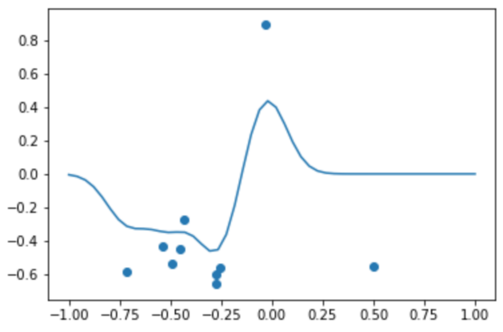


Figure 13: Conditioned on $N-1$ data points, $\lambda = 1$

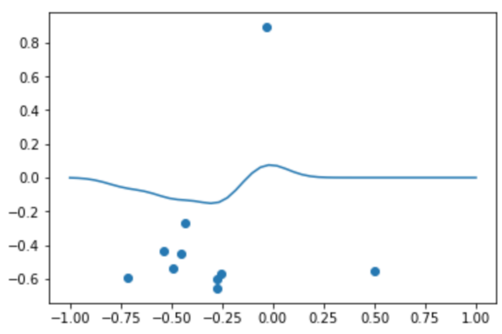


Figure 14: Conditioned on $N - 1$ data points, $\lambda = 10$

We can see from above that as we condition on the last point we have smaller variance around that point as we have gained some information, and the trajectory of the curve changes to address the last point. Also as we increase λ we have more regularization and the model tend to under fit the data.