Rio Sonoyama
Homework 2

1. Prove that $X^TX$ is nonsingular if and only if $X$ has linearly independent columns.

Proof

Let $X = \begin{pmatrix} | & | & & | \\ X_1 & X_2 & \cdots & X_m \\ | & | & & | \end{pmatrix} \in M_{n\times m}(\mathbb{R})$, where $X_i$ is the $i$th column vector of $X$.

($\Rightarrow$) Suppose $X^TX$ is nonsingular. Then, $\det(X^TX) \neq 0$. That is,

$$\det(X^TX) = \det \begin{pmatrix} X_1X_1 & \cdots & X_1X_m \\ \vdots & \ddots & \vdots \\ X_mX_1 & \cdots & X_mX_m \end{pmatrix} \neq 0.$$

Since the determinant of a matrix is 0 if the matrix has at least one zero column, $X_1 \neq 0, \cdots, X_m \neq 0$ — ( * ).
Now consider the following equation:

$$X^TX \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_m \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Since $X^TX$ is nonsingular, the above equation has only the trivial solution (i.e. $\lambda_1 = \cdots = \lambda_m = 0$).
Also, the above equation implies $X_i(X_1\lambda_1 + \cdots + X_m\lambda_m) = 0$ for $1 \leq i \leq m$.
But from ( * ), we know that $X_i \neq 0$. So,

$$X_1\lambda_1 + \cdots + X_m\lambda_m = 0.$$

Now since $\lambda_1 = \cdots = \lambda_m = 0$, by definition of linear independence, $X_1, \ldots, X_m$ are linearly independent.
Hence, the columns of $X$ are linearly independent.

($\Leftarrow$) Suppose $X$ has linearly independent columns.
Consider the following equation:

$$X^TX \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_m \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}.$$

This implies $X_i(X_1\mu_1 + \cdots + X_m\mu_m) = 0$ for $1 \leq i \leq m$. But since $X_1, \ldots, X_m$ are linearly independent, $X_1 \neq 0, \cdots, X_m \neq 0$. Hence, $X_1\mu_1 + \cdots + X_m\mu_m = 0$. But because $X_1, \ldots, X_m$ are linearly independent , it immediately follows that $\mu_1 = \cdots = \mu_m = 0$. Hence, the above equation has only the trivial solutions; therefore, $X^TX$ is nonsingular.

Hence, $X^TX$ is nonsingular if and only if $X$ has linearly independent columns. ⧄

2.

(a) Proof

$$\begin{aligned} H^T &= (X(X^TX)^{-1}X^T)^T \\ &= (X^T)^T(X(X^TX)^{-1})^T \\ &= X((X^TX)^{-1})^TX^T \\ &= X((X^TX)^T)^{-1}X^T \\ &= X(X^TX)^{-1}X^T \\ &= H \end{aligned}$$

Hence, $H = H^T$; therefore, $H$ is symmetric. ▨

(b) Prove $H^k = H\ \forall k \in \mathbb{Z}^+$ by induction on $k$.

Proof

For $k = 1$, $H^1 = H$. So the statement is trivially true.

For $k = 2$,

$$\begin{aligned} H^2 &= HH \\ &= (X(X^TX)^{-1}X^T)(X(X^TX)^{-1}X^T) \\ &= X(X^TX)^{-1}X^TX(X^TX)^{-1}X^T \\ &= XI_m(X^TX)^{-1}X^T \\ &= X(X^TX)^{-1}X^T \\ &= H \end{aligned}$$

Hence, the statement is true for $k = 2$.

Suppose the statement is true for $k = n \geq 2$. That is, $H^n = H$.

Then,

$$\begin{aligned} H^{n+1} &= H^nH \\ &= HH \qquad \text{(by the inductive hypothesis)} \\ &= H \qquad \text{(the statement is true for k = 2)} \end{aligned}$$

Thus, $H^{n+1} = H$.

Hence, by induction, it has shown that $H^k = H\ \forall k \in \mathbb{Z}^+$. ▨

(c) Prove $(I - H)^k = I - H\ \forall k \in \mathbb{Z}^+$ by induction on $k$.

Proof

For $k = 1$, $(I - H)^1 = I - H$. So the statement is trivially true.

For $k = 2$,

$$\begin{aligned} (I - H)^2 &= (I - H)(I - H) \\ &= I - IH - HI + HH \\ &= I - 2H + HH \\ &= I - 2H + H \qquad \text{(from part (b))} \\ &= I - H \end{aligned}$$

Hence, the statement is true for $k = 2$.

Suppose the statement is true for $k = n \geq 2$. That is, $(I - H)^n = I - H$.

(c) Then,

$$\begin{aligned}(I-H)^{n+1} &= (I-H)^n(I-H) \\ &= (I-H)(I-H) \qquad \text{(by the inductive hypothesis)} \\ &= I-H \qquad \text{(the statement is true for k = 2)}\end{aligned}$$

Thus, $(I-H)^{n+1} = I-H$.

Hence, by induction, it has shown that $(I-H)^k = I-H \ \forall k \in \mathbb{Z}^+$. ⍁

(d) Proof

$$\begin{aligned}\text{tr}(H) &= \text{tr}(X(X^TX)^{-1}X^T) \\ &= \text{tr}(X^TX(X^TX)^{-1}) \qquad \because \text{tr}(AB) = \text{tr}(BA) \\ &= \text{tr}(I_M) \\ &= \sum_{i=1}^{M} 1 \\ &= M\end{aligned}$$

Hence, $\text{tr}(H) = M$. ⍁

3.

We're given that $J(\mathbf{w}) = \sum_{n=1}^{N} \alpha_n(\mathbf{w}^T\mathbf{x}_n - y_n)^2$, $\mathbf{w} = (w_0, w_1)^T$, and $\mathbf{x}_n = (1, x_{n,1})^T$ for all $1 \le n \le N$.

First, we compute the partial derivative of $J(\mathbf{w})$ with respect to $w_0$ and $w_1$.

$$\frac{\partial}{\partial w_0}J(\mathbf{w}) = \sum_{n=1}^{N} \alpha_n \cdot 2(\mathbf{w}^T\mathbf{x}_n - y_n)$$

$$\frac{\partial}{\partial w_1}J(\mathbf{w}) = \sum_{n=1}^{N} \alpha_n \cdot 2(\mathbf{w}^T\mathbf{x}_n - y_n)\mathbf{x}_n$$

So we have,

$$\nabla J(\mathbf{w}) = \begin{pmatrix} 2\sum_{n=1}^{N} \alpha_n(\mathbf{w}^T\mathbf{x}_n - y_n) \\ 2\sum_{n=1}^{N} \alpha_n(\mathbf{w}^T\mathbf{x}_n - y_n)\mathbf{x}_n \end{pmatrix}.$$

The gradient descent method on linear regression updates $\mathbf{w}$ as:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \frac{\nabla J(\mathbf{w})}{\|\nabla J(\mathbf{w})\|}.$$

So, the direction of update (i.e. gradient) is affected by the terms with larger $a_n$ in the summation. For example, suppose there is a data point $(\mathbf{x}_i, y_i)$ with $\alpha_i = 0$. Then, the prediction error on that datapoint is not taken into account when updating $\mathbf{w}$. So, even if the prediction error on $(\mathbf{x}_i, y_i)$ is extremely large with some $\mathbf{w}$, as long as the other data points are predicted well enough, the algorithm would converge and return $\mathbf{w}$. On the other hand, if there is a data point $(\mathbf{x}_j, y_j)$ with $\alpha_j \gg \alpha_m \forall m \ne j$, then $\mathbf{w}$ is updated only in favor of minimizing the error on $(\mathbf{x}_j, y_j)$. So as long as the prediction error on $(\mathbf{x}_j, y_j)$ is small enough, the algorithm would converge and return such $\mathbf{w}$ even though the prediction on the other data points are not desirable.

4. Our objective function is given:

$$J(\mathbf{w}) = -\sum_{n=1}^{N} \left( y_n \log \sigma(\mathbf{w}^T \mathbf{x}_n) + (1 - y_n) \log(1 - \sigma(\mathbf{w}^T \mathbf{x}_n)) \right) + \frac{1}{2} \sum_i w_i^2$$

Note that derivative of sigmoid function is as follows:

$$\frac{\partial \sigma(\mathbf{w}^T \mathbf{x}_n)}{\partial w_i} = \frac{\partial}{\partial w_i} \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}_n}} = \frac{1}{(1 + e^{-\mathbf{w}^T \mathbf{x}_n})^2} \cdot (-1) \cdot e^{-\mathbf{w}^T \mathbf{x}_n} \cdot (-x_{n,i}) = \sigma(\mathbf{w}^T \mathbf{x}_n)(1 - \sigma(\mathbf{w}^T \mathbf{x}_n)) x_{n,i}.$$

So,

$$\frac{\partial \sigma(\mathbf{w}^T \mathbf{x}_n)}{\partial \mathbf{w}} = \sigma(\mathbf{w}^T \mathbf{x}_n)(1 - \sigma(\mathbf{w}^T \mathbf{x}_n)) \mathbf{x}_n.$$

Now take partial derivative of $J(\mathbf{w})$ with respect to $w_i$:

$$\frac{\partial}{\partial w_i} \log \sigma(\mathbf{w}^T \mathbf{x}_n) = \frac{1}{\cancel{\sigma(\mathbf{w}^T \mathbf{x}_n)}} \cdot \cancel{\sigma(\mathbf{w}^T \mathbf{x}_n)} \cdot (1 - \sigma(\mathbf{w}^T \mathbf{x}_n)) \cdot x_{n,i} = (1 - \sigma(\mathbf{w}^T \mathbf{x}_n)) \cdot x_{n,i}$$

$$\frac{\partial}{\partial w_i} \log(1 - \sigma(\mathbf{w}^T \mathbf{x}_n)) = \frac{1}{\cancel{1 - \sigma(\mathbf{w}^T \mathbf{x}_n)}} \cdot (-1) \cdot \sigma(\mathbf{w}^T \mathbf{x}_n) \cdot \cancel{(1 - \sigma(\mathbf{w}^T \mathbf{x}_n))} \cdot x_{n,i} = -\sigma(\mathbf{w}^T \mathbf{x}_n) \cdot x_{n,i}$$

So, by combining the above partial derivatives, we have:

$$\begin{aligned}
\frac{\partial J(\mathbf{w})}{\partial w_i} &= -\sum_{n=1}^{N} \left( y_n \frac{\partial}{\partial w_i} \log \sigma(\mathbf{w}^T \mathbf{x}_n) + (1 - y_n) \frac{\partial}{\partial w_i} \log(1 - \sigma(\mathbf{w}^T \mathbf{x}_n)) \right) + \frac{1}{2} \frac{\partial}{\partial w_i} \sum_i w_i^2 \\
&= -\sum_{n=1}^{N} \left( y_n \cdot (1 - \sigma(\mathbf{w}^T \mathbf{x}_n)) \cdot x_{n,i} + (y_n - 1) \cdot \sigma(\mathbf{w}^T \mathbf{x}_n) \cdot x_{n,i} \right) + w_i \\
&= -\sum_{n=1}^{N} \left( y_n - \sigma(\mathbf{w}^T \mathbf{x}_n) \right) x_{n,i} + w_i \,.
\end{aligned}$$

So the update rules for $w_i$ is:

$$\begin{aligned}
w_i &= w_i - \eta \frac{\partial J(\mathbf{w})}{\partial w_i} \\
&= w_i - \eta \left( -\sum_{n=1}^{N} \left( y_n - \sigma(\mathbf{w}^T \mathbf{x}_n) \right) x_{n,i} + w_i \right) \\
&= (1 - \eta) w_i + \eta \sum_{n=1}^{N} \left( y_n - \sigma(\mathbf{w}^T \mathbf{x}_n) \right) x_{n,i} \,.
\end{aligned}$$

5.
MPA estimation:

$$w^* = \arg\max_w \prod_{n=1}^{N} P(y_i|x_i, w)(2\pi)^{m/2} e^{-\frac{1}{2}\sum_{i=1}^{m} w_i^2}$$

Since $y_i|x_i, w \sim \text{Bernoulli}(\sigma(w^T x))$, we can write the conditional probability as:

$$P(y_i|x_i, w) = \sigma(w^T x_i)^{y_i}(1 - \sigma(w^T x_i))^{1-y_i}.$$

So we can rewrite MPS estimation as:

$$w^* = \arg\max_w \prod_{n=1}^{N} \sigma(w^T x_i)^{y_i}(1 - \sigma(w^T x_i)^{1-y_i} \cdot (2\pi)^{m/2} e^{-\frac{1}{2}\sum_{i=1}^{m} w_i^2}.$$

Let $L(w) = \prod_{n=1}^{N} \sigma(w^T x_i)^{y_i}(1 - \sigma(w^T x_i)^{1-y_i} \cdot (2\pi)^{m/2} e^{-\frac{1}{2}\sum_{i=1}^{m} w_i^2}$.

Now instead of finding $w$ that maximizes $L(w)$, we can find $w$ that maximizes log of $L(w)$ because $\log L(w)$ is a strictly increasing function of $L(w)$.
So,

$$\begin{aligned}
\log L(w) &= \log \prod_{n=1}^{N} \sigma(w^T x_i)^{y_i}(1 - \sigma(w^T x_i)^{1-y_i} \cdot (2\pi)^{m/2} e^{-\frac{1}{2}\sum_{i=1}^{m} w_i^2} \\
&= \sum_{n=1}^{N} \left( y_n \log \sigma(\mathbf{w}^T \mathbf{x}_n) + (1 - y_n)\log(1 - \sigma(\mathbf{w}^T \mathbf{x}_n)) \right) - \frac{m}{2}\log(2\pi) - \frac{1}{2}\sum_{i=1}^{m} w_i^2 \\
&= -\left( -\sum_{n=1}^{N} \left( y_n \log \sigma(\mathbf{w}^T \mathbf{x}_n) + (1 - y_n)\log(1 - \sigma(\mathbf{w}^T \mathbf{x}_n)) \right) + \frac{1}{2}\sum_{i=1}^{m} w_i^2 \right) - \frac{m}{2}\log(2\pi) \\
&= -J(W) - \frac{m}{2}\log(2\pi).
\end{aligned}$$

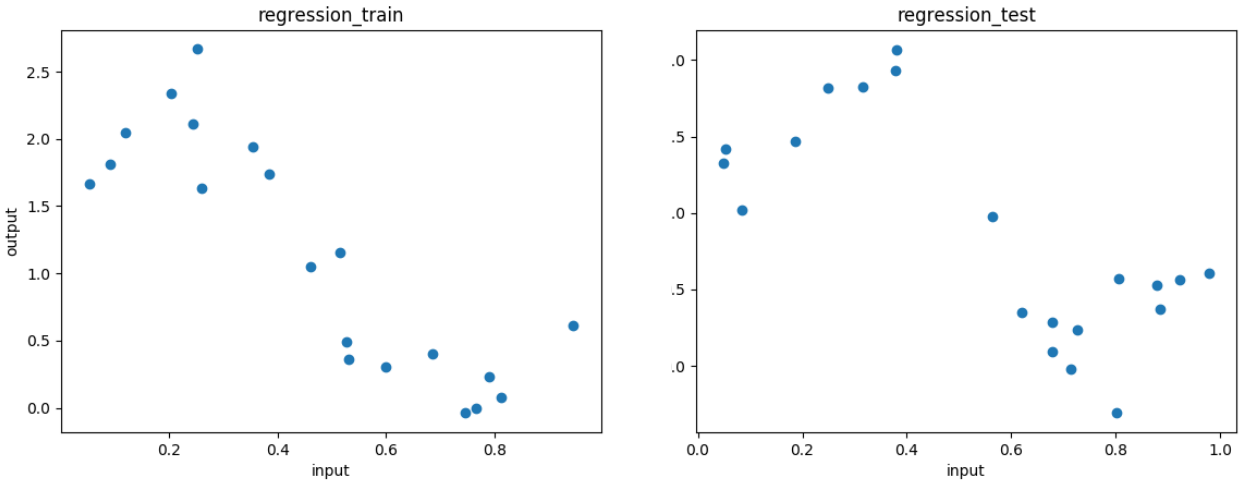
where $J(w)$ is the objective function in problem 4.

Notice that since $-(m/2)\log(2\pi)$ is a constant, we can ignore it when finding $w$ that maximizes $\log L(w)$.
Thus, our goal now is to find $w$ that maximizes $-J(w)$; in other word, we want to find $w$ that minimizes $J(w)$.
Hence,

$$\begin{aligned}
w^* &= \arg\max_w \prod_{n=1}^{N} P(y_i|x_i, w)(2\pi)^{m/2} e^{-\frac{1}{2}\sum_{i=1}^{m} w_i^2} \\
&= \arg\max_w \log L(w) \\
&= \arg\max_w \; -J(w) \\
&= \arg\min_w \; J(w).
\end{aligned}$$

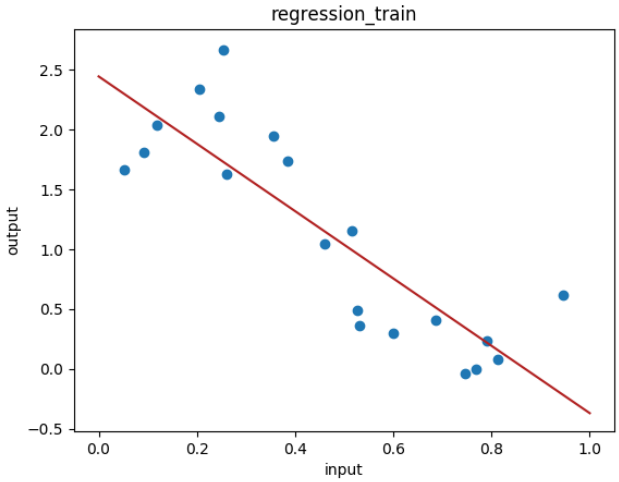
Therefore, the MAP estimation is equivalent to minimizing the logistic regression objective with L2 regularization as shown in problem 4.

6.
(a)




As we can see in the plots above, the data points in both train the test datasets have relatively similar distribution; the larger the input value is, the smaller the output value. Also, the data points are, roughly speaking, linearly distributed. Hence, given the train dataset, linear regression algorithm is likely to give a relatively good prediction on the test dataset.

(b)



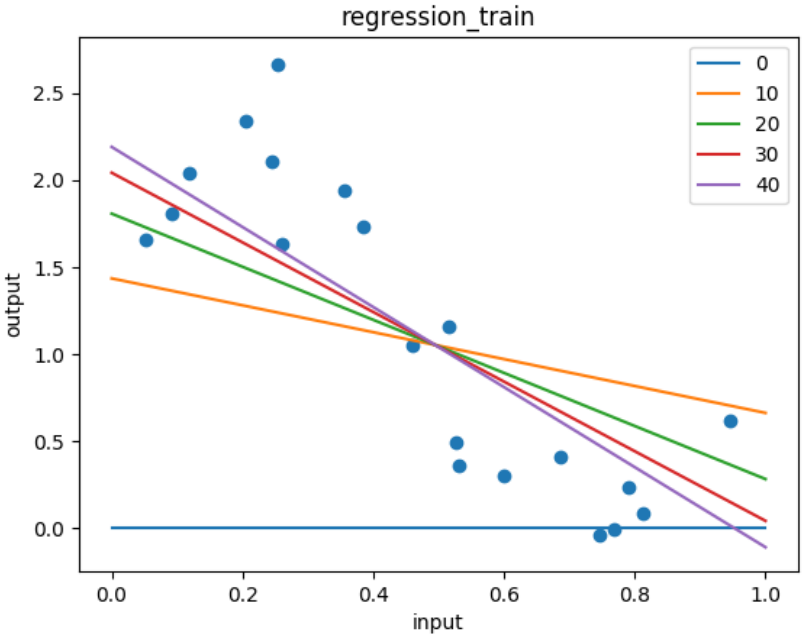

```
w =  [ 2.44640709 -2.81635359]
J(w) =  3.912576405791464
```

(c)

| Learning Rate | $w$ | $J(w)$ | Number of iterations |
|---|---|---|---|
| 0.0407 | [ 2.43297089 -2.789192] | 3.9135899229431 | 104 |
| 0.01 | [ 2.41824214 -2.75941751] | 3.9170298493292600 | 363 |
| 0.001 | [ 2.35668234 -2.63497301] | 3.9577726508563877 | 2608 |
| 0.0001 | [ 1.91573585 -1.74358989] | 5.493565588736025 | 10000 |

As shown in the table in the previous page, learning rate affects the number of iteration needed for gradient descent algorithm to converge. In particular, if the learning rate is too small (e.g. =0.0001), gradient descent algorithm does not converge within the maximum number of iterations (=10000). In theory, the gradient descent should converge at one of minima if the objective function is concave. However, in practice, we often set a maximum number of iterations as a hyper parameter. So, it is important to tune learning rate appropriately to fit the model.

(d)




| Learning Rate | w | J(w) | Number of iterations |
|---|---|---|---|
| 0.0407 | [0, 0] | 40.23384740967100 | 0 |
| 0.0407 | [ 1.4358061 -0.77340127] | 9.64630623255847 | 10 |
| 0.0407 | [ 1.80818327 -1.52616998] | 6.199353365496530 | 20 |
| 0.0407 | [ 2.04335025 -2.00156523] | 4.824609065409560 | 30 |
| 0.0407 | [ 2.19186501 -2.30179112] | 4.276321356377880 | 40 |

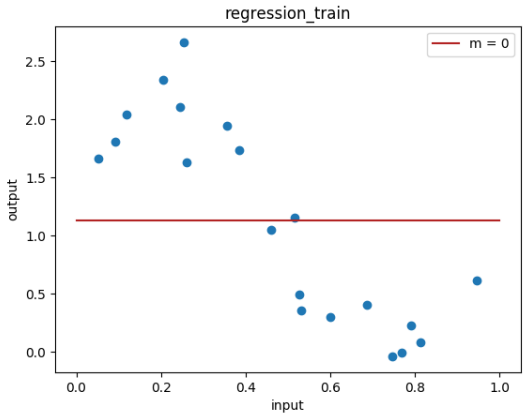
We can clearly see that as the number of iterations increases, the values of $w$ and $J(w)$ get closer to the value found in part (b). This makes sense because gradient descent method update $w$ on each iteration so that the updated $w$ is better than before. So, in theory, the more iterations, the better $w$ we get.

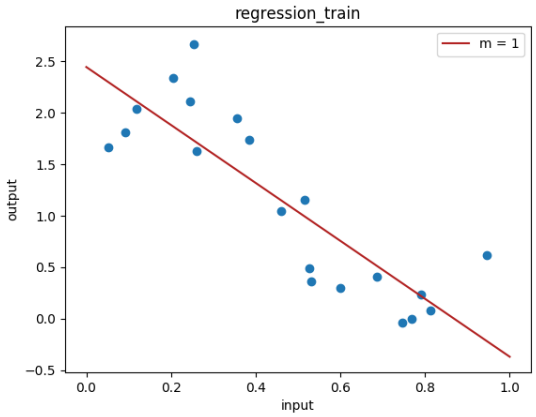
(e)

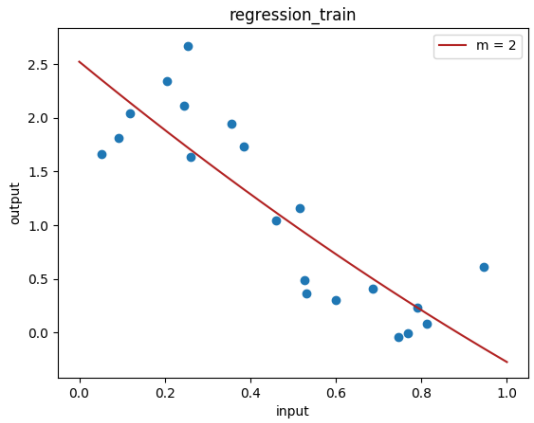
The table below shows the $J(w)$ on the train dataset for each degree.

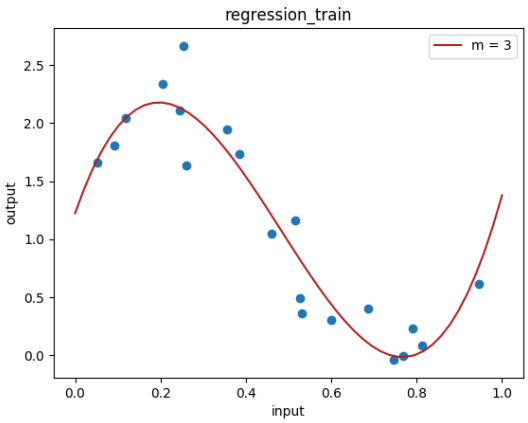
| Degree | J(w) |
|---|---|
| 0 | 14.69105824516290 |
| 1 | 3.91257640579146 |
| 2 | 3.89517797871432 |
| 3 | 1.1933490268629 |
| 4 | 1.05508341638251 |
| 5 | 1.02886759302538 |
| 6 | 1.00758251983219 |
| 7 | 0.988185212741428 |
| 8 | 0.987995263640667 |
| 9 | 0.897005399558621 |
| 10 | 0.896240358619512 |

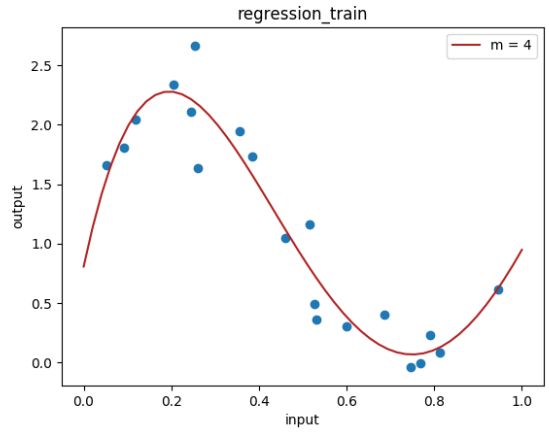
In terms of the square error, we can say that $m = 10$ is the best fit regression model on the train dataset. Also, by visualizing the polynomials for each $m = 0,1,\ldots,10$ (shown below), we can see that $m = 10$ fits the best on the training dataset.

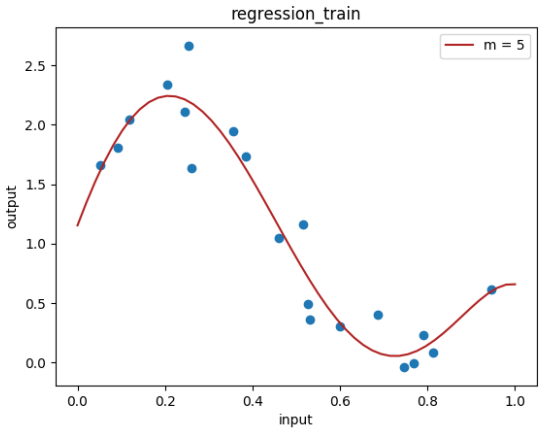


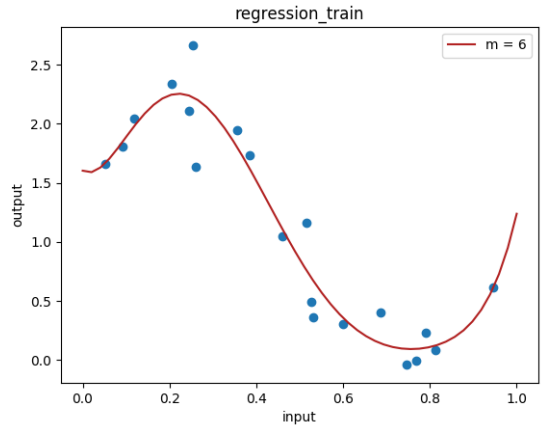



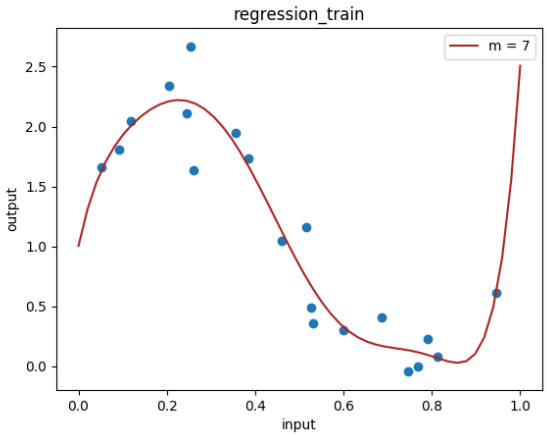



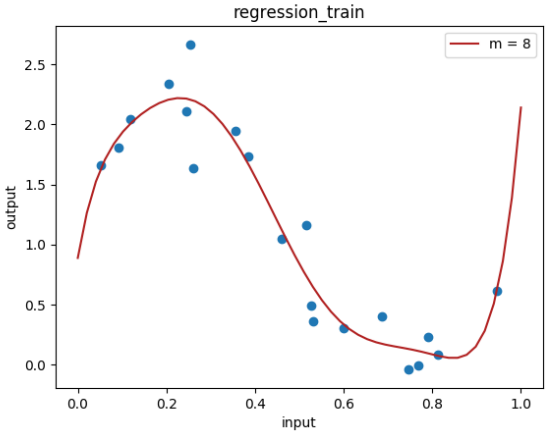



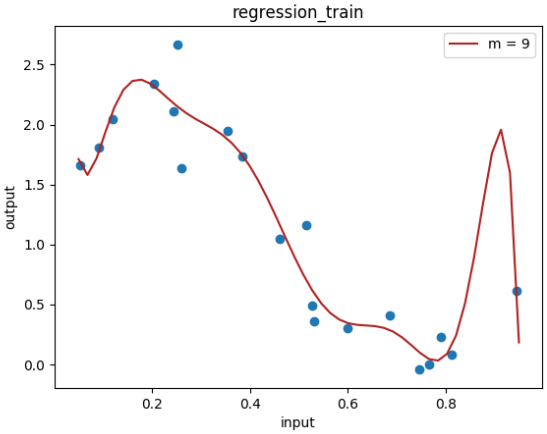

regression_train
m = 4
output
input

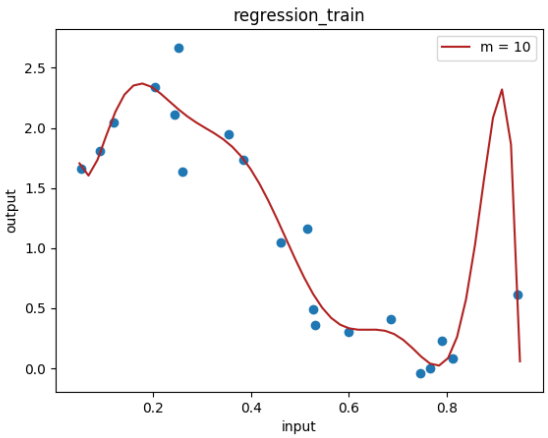

regression_train
m = 5
output
input


regression_train
m = 6
output
input


regression_train
m = 7
output
input


regression_train
m = 8
output
input

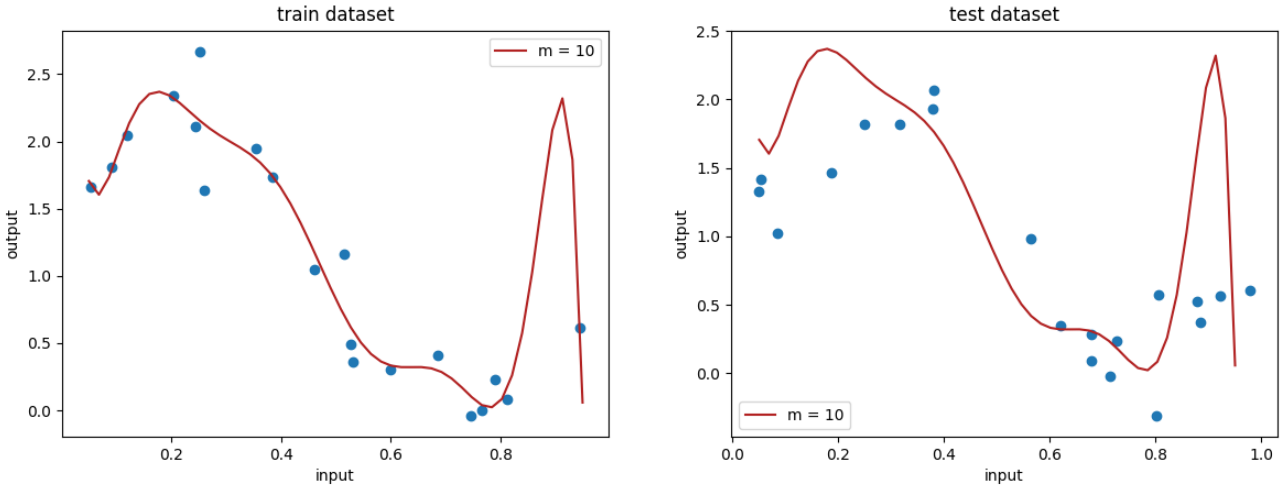

regression_train
m = 9
output
input


regression_train
m = 10
output
input

For $m = 10$, we have:

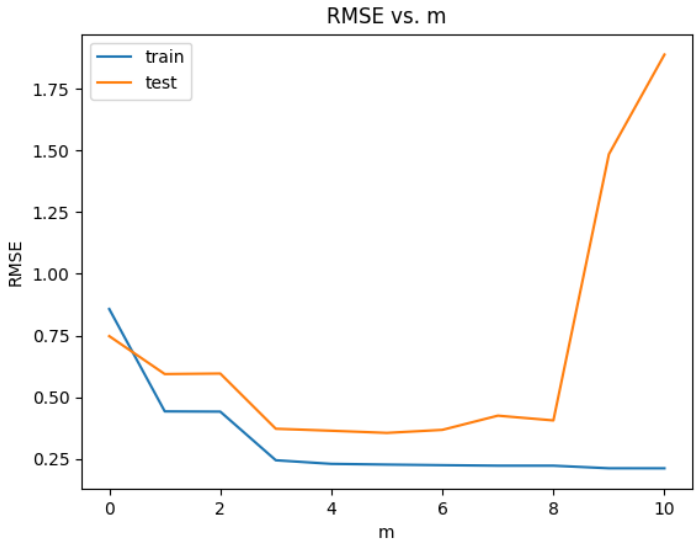
```
RMSE on train data:  0.21168849267491038
RMSE on test data:  1.8880095702180946
```

Although $m = 10$ resulted in the best fit on the train dataset, it might not be the best fit for the test dataset. This is an example of overfitting, which is illustrated in the plots below:




Notice that data points on the train dataset are more likely to be closer to the curve compared to the points on the test dataset. This is an evidence of overfitting.

The plots below are depicting how RMSE (both training and testing) varies with model complexity (polynomial degree m):




Even though RMSE on the train dataset decreases as m increases, it is not the case for the test dataset. In fact, from the plot above, RMSE on the test dataset is minimized approximately at $m = 5$. This plot shows the existence of overwriting very well. In general, if we fit a model using overly complicate method (e.g. $m = 10$), it results in overfitting the model, which is not desirable. Rather, we want to train a model so that it can be applied to other datasets other than the train dataset.