3.a) $P(R=1 \mid D=1) == P(R=0 \mid D=0) == \theta$

$$P(D=1 \mid R=1) = \frac{P(R=1 \mid D=1) \cdot P(D=1)}{P(R=1)}$$

$$P(R=1) = P(R=1 \cap D=1) + P(R=1 \cap D=0)$$

$$= P(R=1 \mid D=1) \cdot P(D=1) + P(R=1 \mid D=0) \cdot P(D=0)$$

$$= \theta \cdot a + P(R=1 \mid D=0) \cdot P(D=0)$$

$$= \theta \cdot a + P(R=1 \mid D=0) \cdot (1-a)$$

$$= \theta \cdot a + (1-\theta) \cdot (1-a)$$

$$P(D=1 \mid R=1) = \frac{\theta \cdot a}{\theta a + (1-\theta) \cdot (1-a)}$$

$\theta = 0.9 \quad a = 0.06$

$$\frac{0.9 \cdot 0.06}{(0.9 \cdot 0.06) + (1-0.9) \cdot (1-0.06)} = \frac{0.054}{0.054 + 0.1 \cdot (0.94)}$$

$$= \frac{0.054}{0.148} = 0.36486486$$

b) $P(D=1 \mid R_1=1, R_2=1)$

$$= \frac{\widetilde{P(R_1=1, R_2=1 \mid D=1) \cdot P(D=1)}}{P(R_1=1, R_2=1)}$$

$$= \frac{P(R_1=1 \mid D=1) \cdot P(R_2=1 \mid D=1) \cdot P(D=1)}{P(R_1=1) \cdot P(R_2=1)}$$

$$= \frac{P(R_1=1 \mid D=1) \cdot P(D=1)}{P(R_1=1)} \cdot \frac{P(R_2=1 \mid D=1)}{P(R_2=1)}$$

$$= \frac{\theta \cdot \alpha}{\theta\alpha + (1-\alpha)(1-\theta)} \cdot \frac{\theta}{\theta\alpha + (1-\alpha)(1-\theta)} = \frac{\theta^2 \cdot \alpha}{[\theta\alpha + (1-\alpha)(1-\theta)]^2}$$

$\alpha = 0.06 \quad \theta = 0.9$

$$P(D=1 \mid R_1=1, R_2=1) = \frac{0.0486}{0.148^2} = \frac{0.0486}{0.021904}$$

$= 2.21877283$

c) Test$_1$:

$$P(D=1 \mid R=1) = \frac{\theta \cdot \alpha}{\theta\alpha + (1-\theta)\cdot(1-\alpha)}$$

Test 1+2 :

$$P(D=1 \mid R_1=1, R_2=1) = \frac{\theta^2 \cdot \alpha}{[\theta\alpha + (1-\alpha)(1-\theta)]^2}$$

$$\frac{\theta^2 \cdot \alpha}{[\theta\alpha + (1-\alpha)(1-\theta)]^2} > \frac{\theta \cdot \alpha}{\theta\alpha + (1-\theta)\cdot(1-\alpha)}$$

$$\theta^2 \cdot \alpha > \theta \cdot \alpha \cdot (\theta\alpha + (1-\theta)\cdot(1-\alpha))$$

$$\theta > \theta\alpha + (1-\theta)\cdot(1-\alpha)$$

$$\theta > \theta\alpha + \theta\alpha - \theta - \alpha + 1$$

$$1 > \alpha + \alpha - 1 - \frac{\alpha}{\theta} + \frac{1}{\theta}$$

$$1 > 2\alpha + \frac{1-\alpha}{\theta} - 1$$

$$0 > 2\alpha - \frac{1-\alpha}{\theta} - 2$$

$$0 > 2\left(\alpha - \frac{1-\alpha}{2\theta} - 1\right)$$

$$\alpha - \frac{1-\alpha}{2\theta} < 0$$

$$\alpha < \frac{1-\alpha}{2\theta}$$

4a) Let the binary expression of b be $E(b)$ and let $x = b-1$

$E(b) = \lfloor x/2^i \rfloor$ i from 0 to $k-1$

$x = x \bmod 2^i$

eg: $\lfloor 2.1 \rfloor = 2$

For example $E(5) = 4_2 = 100$

the probability of 5 is $\alpha(1-\alpha)^2$ is one 1 and two 0s in $E(5)$

assume $E(b) = b_0, b_1 \cdots b_{k-1}$, $b_i = 0$ or 1, $i \in [0, k-1]$ and let $f(b)$ = the num of 1s in $E(b)$

$$P(x=b) = \alpha^{f(b)}(1-\alpha)^{k-f(b)}$$

b) let the n random be $x_1 \cdots x_n$

$$L(\alpha) = \prod_{i=1}^{n} P(x_i, \alpha) = \prod_{i=1}^{n} \alpha^{f(i)}(1-\alpha)^{k-f(x_i)}$$

$$\frac{dL(\alpha)}{d\alpha} = 0 \Rightarrow \Sigma(1-\alpha) + \alpha(nk - \Sigma)(-1) = 0$$

$$\Rightarrow \alpha = \frac{\Sigma}{nk} \qquad \Sigma = f(x_1) + f(x_2) + \cdots + f(x_n)$$

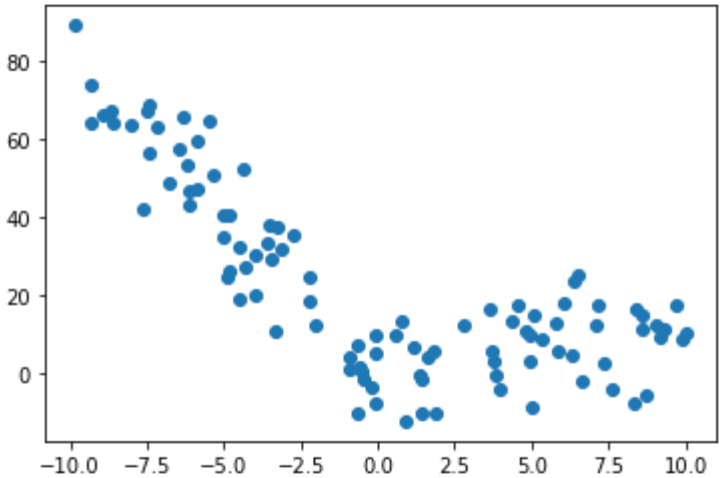
c) $\mathcal{L}(\alpha) = P(\alpha) \prod_{i=1}^{n} \alpha^{S(i)} (1-\alpha)^{k - S(i)}$

$= 6\alpha(1-\alpha)\alpha^{\Sigma} (1-\alpha)^{nk - \Sigma}$

d)

In [106…

```
import matplotlib.pyplot as plt
import numpy as np
import math
from numpy.linalg import inv
from random import randint
x = np.loadtxt('dataset1_inputs.txt').reshape(100, 1)
t = np.loadtxt('dataset1_outputs.txt').reshape(100, 1)
plt.plot(x, t, 'o')
plt.show()
```



ERM

In [107…

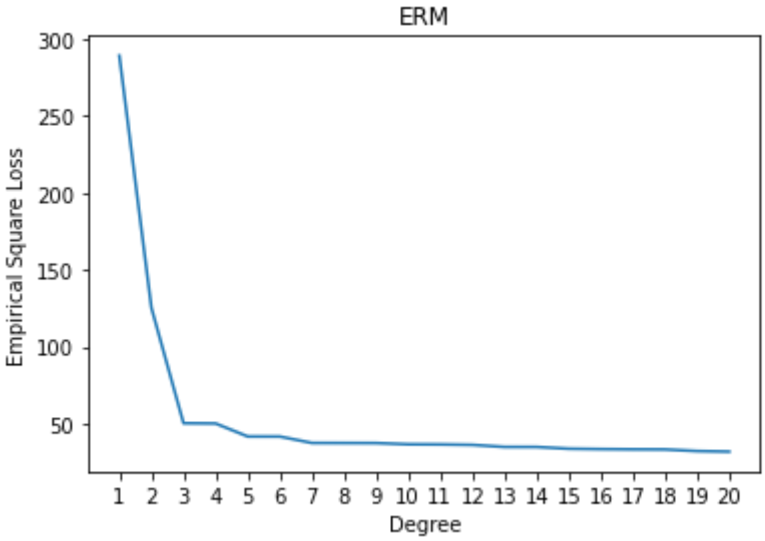
```
def gen_design_matrix(x, dim):
    my_matrix = np.zeros((len(x),dim))
    for i in range(len(x)):
        for j in range(dim):
            my_matrix[i][j] = x[i][0] ** j
    return my_matrix

def ERM(train_x,train_t,test_x,degree):
    answer=[]
    for i in range(1,degree+1):
        dm = gen_design_matrix(train_x,i)
        test_matrix = gen_design_matrix(test_x,i)
        v = np.dot(np.dot(inv(np.dot(np.transpose(dm),dm)),np.transpose(dm)),train_t)
        l = np.dot(test_matrix,v).tolist()
        answer.append(l)
    return np.array(answer)
def cal_error(train_x,trian_t,degree):
    answer=[]
    for i in range(degree):
        distance =0
        for x,y in zip(train_x[i],trian_t):
            distance +=((x-y)**2)/2
        answer.append(distance/len(trian_t))
    return np.array(answer)
```

In [108…

```
pred_erm= ERM(x,t,x,20)
loss_erm = cal_error(pred_erm,t,20)
plt.plot([i for i in range(1,21)],loss_erm)
plt.xticks(np.arange(1, 21, 1.0))
plt.xlabel('Degree')
```

```
plt.title('ERM')
plt.ylabel('Empirical Square Loss')
plt.show()
```



From the plot we can know that when W>=3 and W < 8 will be suitable

RLM

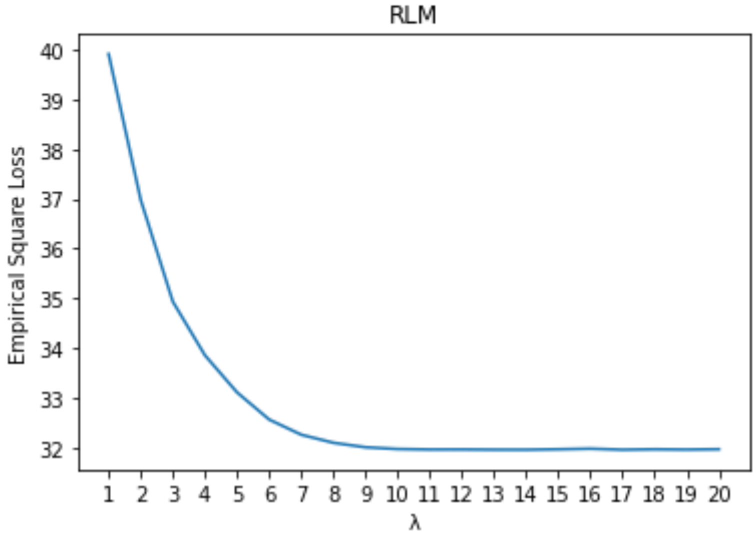
In [109…

```
def RLM(train_x,train_t,degree):
    answer=[]
    for i in range(1,degree+1):
        dm = gen_design_matrix(train_x,degree)
        reg = math.e**(10-i)
        v = np.dot(np.dot(inv(np.add(np.multiply(np.identity(degree),reg),np.dot(np.t
        l = np.dot(dm,v).tolist()
        answer.append(l)
    return answer
```

In [110…

```
pred_rlm= RLM(x,t,20)
loss_rlm = cal_error(pred_rlm,t,20)

plt.plot([i for i in range(1,21)],loss_rlm)
plt.xticks(np.arange(1, 21, 1.0))
plt.xlabel('λ')
plt.title('RLM')
plt.ylabel('Empirical Square Loss')
plt.show()
```



With the help of the regulazator λ we can see the curve from rlm is way smoother than erm means the overfitting problem is been improved

cross validation

In [111…

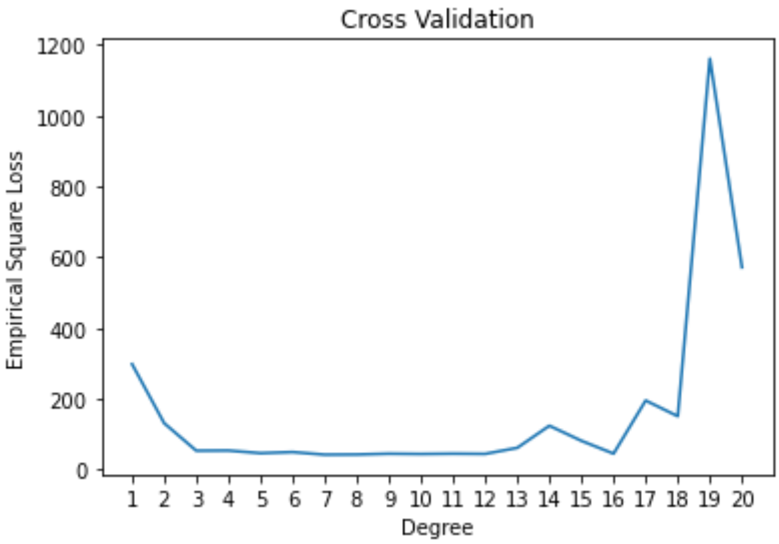
```
folds = 10
random_value = np.random.permutation(len(x))
split = np.split(random_value,folds)#get random split indexs


answer_cross=[]
for i in range(folds):#10 loops
    sub_test_index = []
    sub_train_index=[]
    sub_test_index.append(split[i])#Take one folds of data as our testing data
    for j in range(folds):#Take 9 folds of data as our traning data
        if(split[j][0]!=sub_test_index[0][0]):
            sub_train_index.append(split[j])
    sub_train_data_x =[]
    sub_train_data_y =[]
    sub_test_data_x =[]
    sub_test_data_y =[]

    for q in range(len(sub_train_index)):#convert index to data
        for w in range(len(sub_train_index[0])):
            sub_train_data_x.append(x[sub_train_index[q][w]][0])
            sub_train_data_y.append(t[sub_train_index[q][w]][0])
    for a in range(len(sub_test_index)):#convert index to data
        for b in range(len(sub_test_index[0])):
            sub_test_data_x.append(x[sub_test_index[a][b]][0])
            sub_test_data_y.append(t[sub_test_index[a][b]][0])
    sub_train_data_x = np.array(sub_train_data_x).reshape(90,1)#the 9 folds
    sub_train_data_y = np.array(sub_train_data_y).reshape(90,1)
    sub_test_data_x = np.array(sub_test_data_x).reshape(10,1)# the 1 fold
    sub_test_data_y = np.array(sub_test_data_y).reshape(10,1)
    erm = ERM(sub_train_data_x,sub_train_data_y,sub_test_data_x,20)#traning with 9 fol
    score = cal_error(erm,sub_test_data_y,20)#cal the loss
    #print(score)
    answer_cross.append(score)#record the loss
'''answer_avg=[]
for c in range(len(answer_cross)):
    value =0
```

```
    for d in range(len(answer_cross[0])):
        value+=answer_cross[c][d][0]
    answer_avg.append(value/10)'''
answer_avg = [np.average(i)for i in zip(*answer_cross)]# use np to calculate the avg


plt.plot([i for i in range(1,21)],answer_avg)
plt.xticks(np.arange(1, 21, 1.0))
plt.xlabel('Degree')
plt.title('Cross Validation')
plt.ylabel('Empirical Square Loss')
plt.show()
```



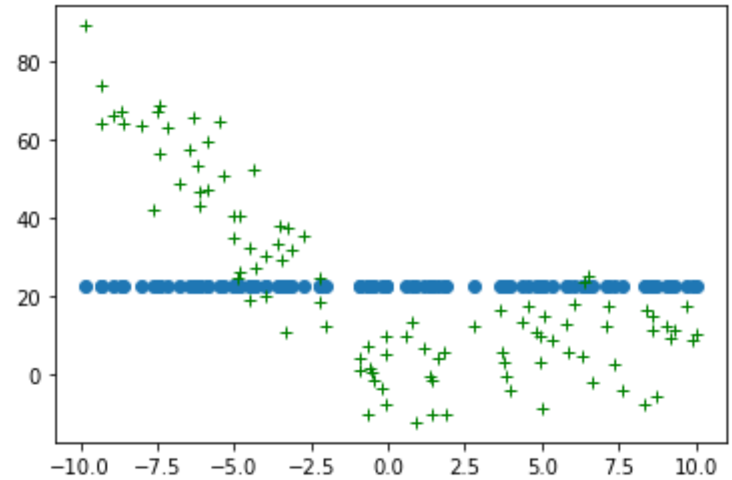
W>=3 and W<=13 would be suitable, loss has large decrease till W=3 and raise again at W=13

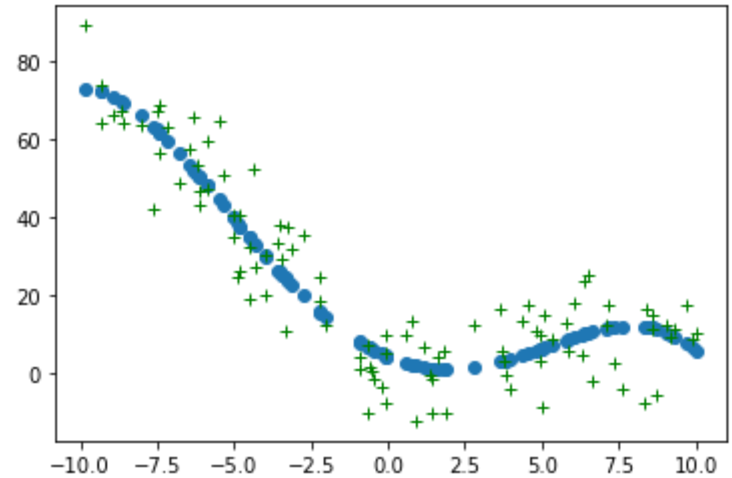
visualization

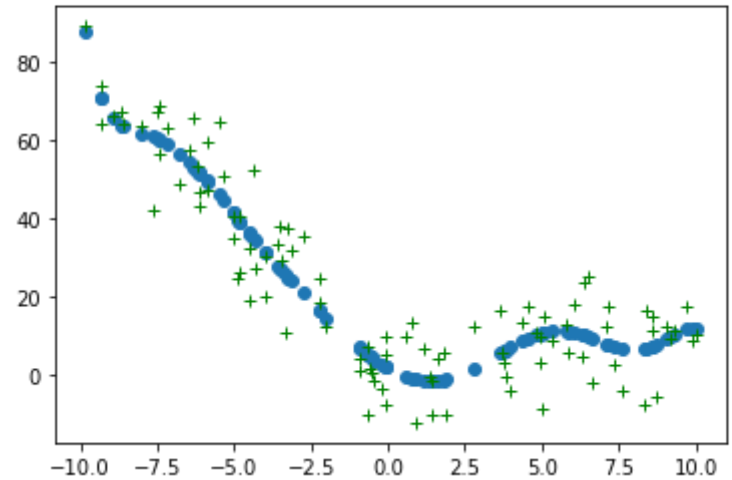
ERM

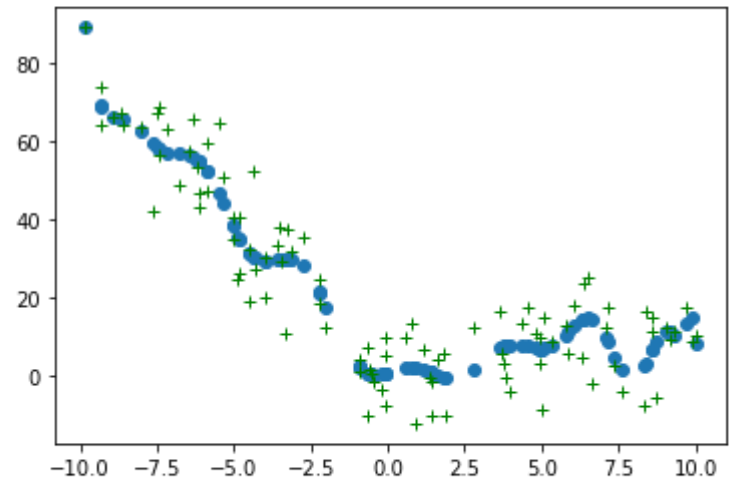
In [112…

```
in_put = x
out_put= t
plt.plot(in_put, out_put, '+g', label="Data")
erm_1 = ERM(x,t,x,1)
plt.scatter(x,erm_1)
plt.show()
erm_5 = ERM(x,t,x,5)
plt.plot(in_put, out_put, '+g', label="Data")
plt.scatter(x,erm_5[-1])
plt.show()
erm_10 = ERM(x,t,x,10)
plt.plot(in_put, out_put, '+g', label="Data")
plt.scatter(x,erm_10[-1])
plt.show()
erm_20 = ERM(x,t,x,20)
plt.plot(in_put, out_put, '+g', label="Data")
plt.scatter(x,erm_20[-1])
plt.show()
```


80
60
40
20
0
−10.0 −7.5 −5.0 −2.5 0.0 2.5 5.0 7.5 10.0


80
60
40
20
0
−10.0 −7.5 −5.0 −2.5 0.0 2.5 5.0 7.5 10.0

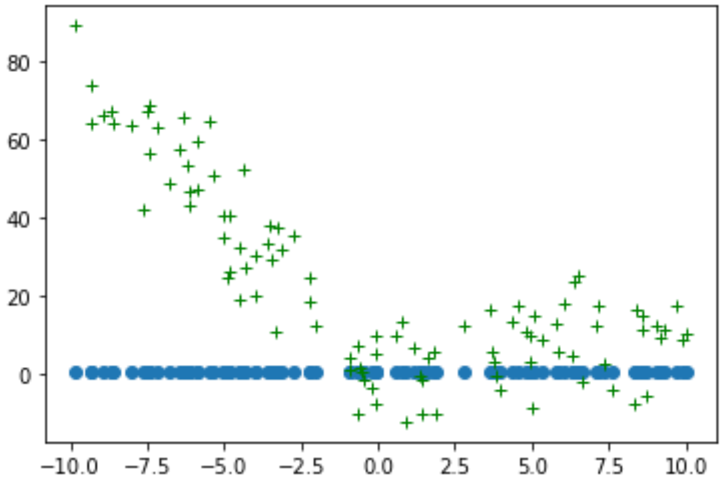

80
60
40
20
0
−10.0 −7.5 −5.0 −2.5 0.0 2.5 5.0 7.5 10.0

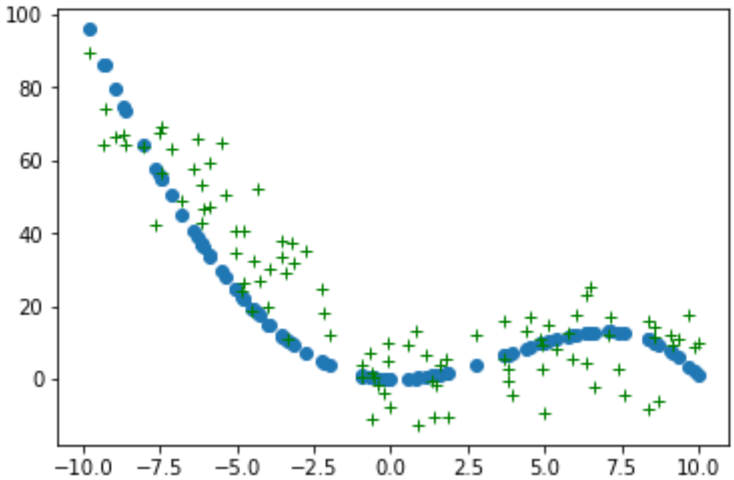

80
60
40
20
0
−10.0 −7.5 −5.0 −2.5 0.0 2.5 5.0 7.5 10.0

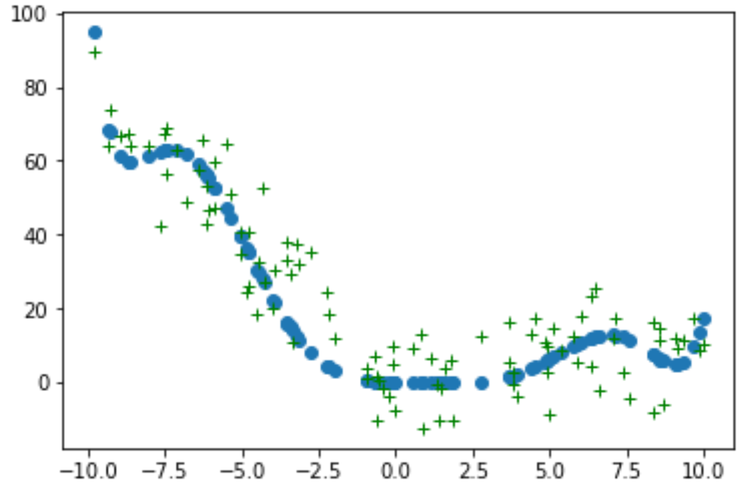
RLM

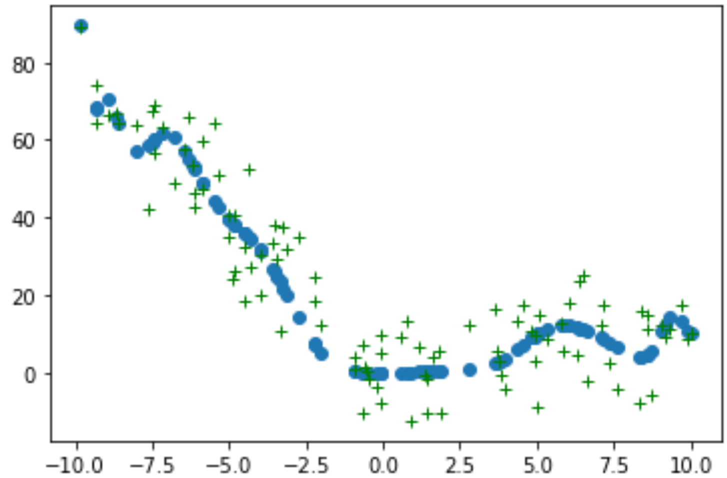
In [113…

```
in_put = x
out_put= t
plt.plot(in_put, out_put, '+g', label="Data")
rlm_1 = RLM(x,t,1)
plt.scatter(x,rlm_1)
plt.show()
rlm_5 = RLM(x,t,5)
plt.plot(in_put, out_put, '+g', label="Data")
plt.scatter(x,rlm_5[0])
plt.show()
rlm_10 = RLM(x,t,10)
plt.plot(in_put, out_put, '+g', label="Data")
plt.scatter(x,rlm_10[0])
plt.show()
rlm_20 = RLM(x,t,20)
plt.plot(in_put, out_put, '+g', label="Data")
plt.scatter(x,rlm_20[0])
plt.show()
```









From ther observation, we can find out that RLM made the graph more smothier which means less overfitting occurs and we can easily tell that when W=5 is the best suitable case.