

Pattern Recognition and Machine Learning

Assignment - 02

July 21, 2025

Name : Rohan Baghel
Student ID: 202116011

About

In this, we will study regularized Polynomial regression and implement it using Matlab on random X values and $f(x) = \sin(2\pi X)$ and also for $f(x) = \sin(2\pi \|X\|)$

Polynomial regression can be used for nonlinear models. When solving polynomial regression problems, polynomial regression can be transformed into linear regression to solve. In order to avoid over-fitting in polynomial regression, a regularization method can be used to suppress the coefficients of higher-order polynomial, and the article evaluates the influence of regularization coefficients on polynomial regression.

0.0.1 Regularization Polynomial regression Model

Regularized regression is a type of regression where the coefficient estimates are constrained to zero. The magnitude (size) of coefficients, as well as the magnitude of the error term, are penalized. Complex models are discouraged, primarily to avoid overfitting.

File1

In this file we are generating X vector with random points and adding noise for the function $f(x) = \sin(2\pi x)$ with $E \sim N(0, 0.25)$.

Taking

- Training points = 20
- Testing points = 50

And Estimate regularized Least square polynomial regression mode for $M = 1, 2, 3, 9$ and finding Coefficient for each M

computing the RMSE for polynomial regression models for order $M = 1, 2, 3$ and 9 .

```
1
2 clear
3 clc
4
5 %-----%
6 %Q1
7 %Random Number for Trainig set
8 x_train= rand(20,1);
9 x_train = sort(x_train);
10 %Q2
11 %training set Y
12 y_train= sin(2*pi*x_train) + randn(20,1)*0.25;
13
14 %-----%
15 %Q3
16 %for Testing setnnext_trainingttile
17 x_test = rand(50,1);
18 x_test = sort(x_test);
19
20 y_test= sin(2*pi*x_test) + randn(50,1)*0.25;
21
22
23 %-----%
24 %4
25 fprintf("Q4, Q5. Estimate regularized least square polynomial regression mode ...
26         for M =1,2,3,9 \n and Coffecient \n\n")
27 L=0.01; %Value of lambda
28
29 %M=1;
30 %For Training
31 A = [x_train, ones(20,1)];
32 Coff_1=inv(A'*A+L*eye(size(A'*A)))*(A'*y_train);
33 Y_cap_11= A*Coff_1;
34
```

```

35 %For Testing
36 A_test= [x_test, ones(50,1)];
37 Y_cap_12= A_test*Coff_1;
38
39
40 % M=2
41 %For training
42 B = [x_train,x_train.^2,ones(20,1)];
43 Coff_2=inv(B'*B+L*eye(3,3))*(B'*y_train);
44 Y_cap_21= B*Coff_2;
45
46 %For Testing
47 B_test1= [x_test,x_test.^2, ones(50,1)];
48 Y_cap_22= B_test1*Coff_2;
49
50
51 % M=3
52 %For Training
53 C = [x_train,x_train.^2,x_train.^3,ones(20,1)];
54 Coff_3=inv(C'*C+0.01*eye(4,4))*(C'*y_train); %coeff
55 Y_cap_31= C*Coff_3;
56
57 %For Testing
58 B_test2= [x_test,x_test.^2,x_test.^3, ones(50,1)];
59 Y_cap_32= B_test2*Coff_3;
60
61
62 % M=9
63 %For Training
64 D = [x_train,x_train.^2,x_train.^3,x_train.^4,x_train.^5,x_train.^6,x_train.^7,...
        x_train.^8,x_train.^9,ones(20,1)];
65 Coff_9=inv(D'*D+0.01*eye(size(D'*D)))*(D'*y_train); %coeff
66 Y_cap_91= D*Coff_9;
67
68 %For Testing
69 B_test3= [x_test,x_test.^2,x_test.^3,x_test.^4,x_test.^5,x_test.^6,x_test.^7,...
        x_test.^8,x_test.^9, ones(50,1)];
70 Y_cap_92= B_test3*Coff_9;
71
72 %-----%
73 %Q6 RMSE Values of Testing values
74
75 RMSE_test_1 = sqrt(mean(Y_cap_12-y_test).^2 );
76 RMSE_test_2 = sqrt(mean(Y_cap_22-y_test).^2 );
77 RMSE_test_3 = sqrt(mean(Y_cap_32-y_test).^2 );
78 RMSE_test_9 = sqrt(mean(Y_cap_92-y_test).^2 );
79 disp("RMSE M= 1")
80 disp(RMSE_test_1)
81 disp("RMSE M= 2")
82 disp(RMSE_test_2)
83 disp("RMSE M= 3")
84 disp(RMSE_test_3)
85 disp("RMSE M= 9")
86 disp(RMSE_test_9)
87
88 % Q7
89 y = sin(2*pi*x_train);
90
91

```

```

92 ax1 = nexttile;
93 plot(x_train,y,x_train,y_train,'o',x_train,Y_cap_11,'r'), legend('Sin(2      x)', '...
      sin(2      x) +      -N(0,0.25)', 'y');
94 title(ax1, 'M= 1')
95
96 ax2 = nexttile;
97 plot(x_train,y,x_train,y_train,'o',x_train,Y_cap_21,'r'), legend('Sin(2      x)', '...
      sin(2      x) +      -N(0,0.25)', 'y');
98 title(ax2, 'M= 2')
99
100 ax3 = nexttile;
101 plot(x_train,y,x_train,y_train,'o',x_train,Y_cap_31,'r'), legend('Sin(2      x)', '...
      sin(2      x) +      -N(0,0.25)', 'y');
102 title(ax3, 'M= 3')
103
104 ax4 = nexttile;
105 plot(x_train,y,x_train,y_train,'o',x_train,Y_cap_91,'r'), legend('Sin(2      x)', '...
      sin(2      x) +      -N(0,0.25)', 'y');
106 title(ax4, 'M= 9')
107
108 % Q8
109
110 y = sin(2*pi*x_test);
111
112 ax1 = nexttile;
113 plot(x_test,y,x_test,y_test,'o',x_test,Y_cap_12,'r'), legend('Sin(2      x)', 'sin...
      (2      x) +      -N(0,0.25)', 'y');
114 title(ax1, 'M= 1')
115
116 ax2 = nexttile;
117 plot(x_test,y,x_test,y_test,'o',x_test,Y_cap_22,'r'), legend('Sin(2      x)', 'sin...
      (2      x) +      -N(0,0.25)', 'y');
118 title(ax2, 'M= 2')
119
120 ax3 = nexttile;
121 plot(x_test,y,x_test,y_test,'o',x_test,Y_cap_32,'r'), legend('Sin(2      x)', 'sin...
      (2      x) +      -N(0,0.25)', 'y');
122 title(ax3, 'M= 3')
123
124 ax4 = nexttile;
125 plot(x_test,y,x_test,y_test,'o',x_test,Y_cap_92,'r'), legend('Sin(2      x)', 'sin...
      (2      x) +      -N(0,0.25)', 'y');
126 title(ax4, 'M= 9')

```

0.0.2 Output of training set:

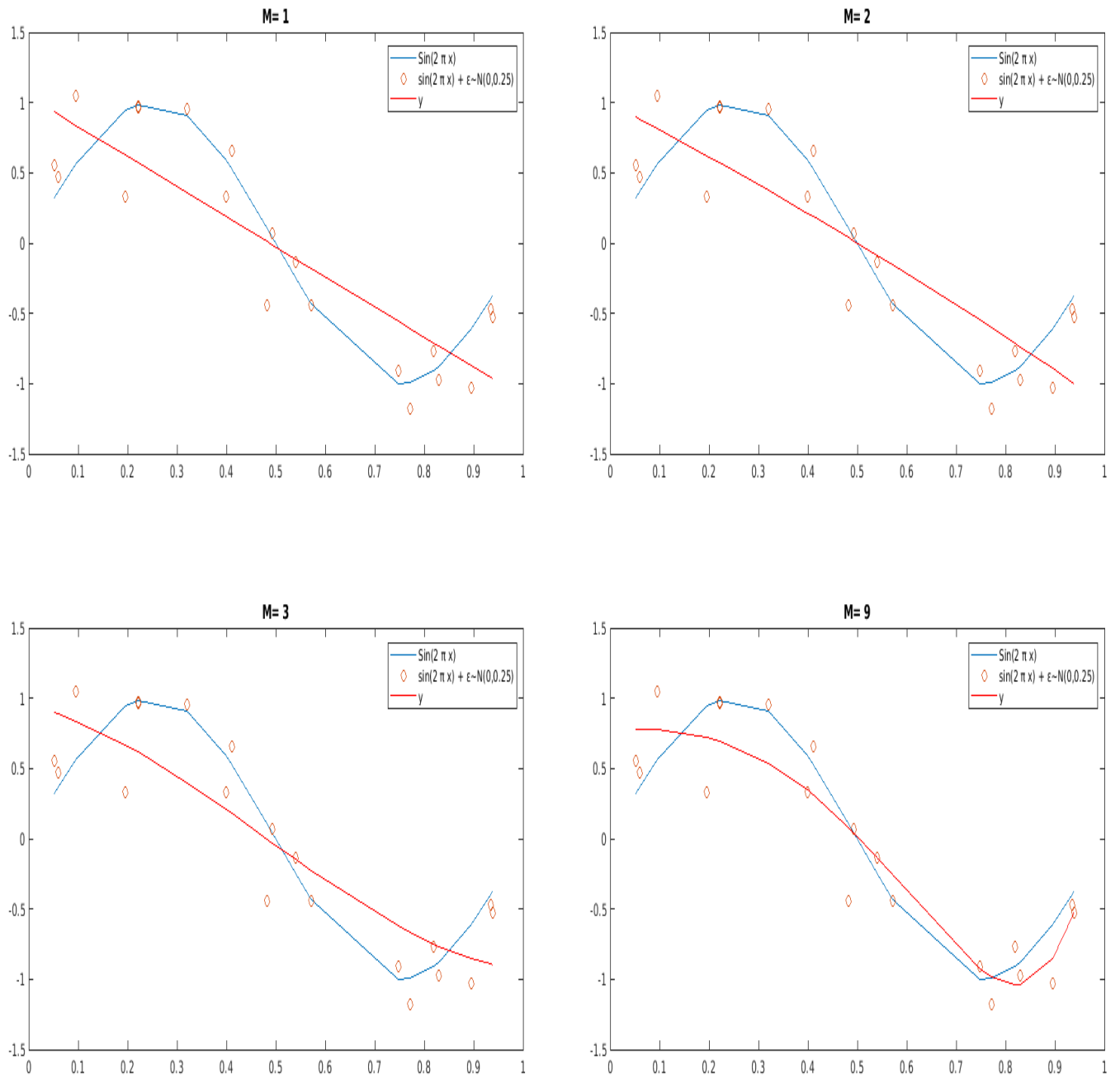


Figure 1: for Training set

0.0.3 Output of testing set:

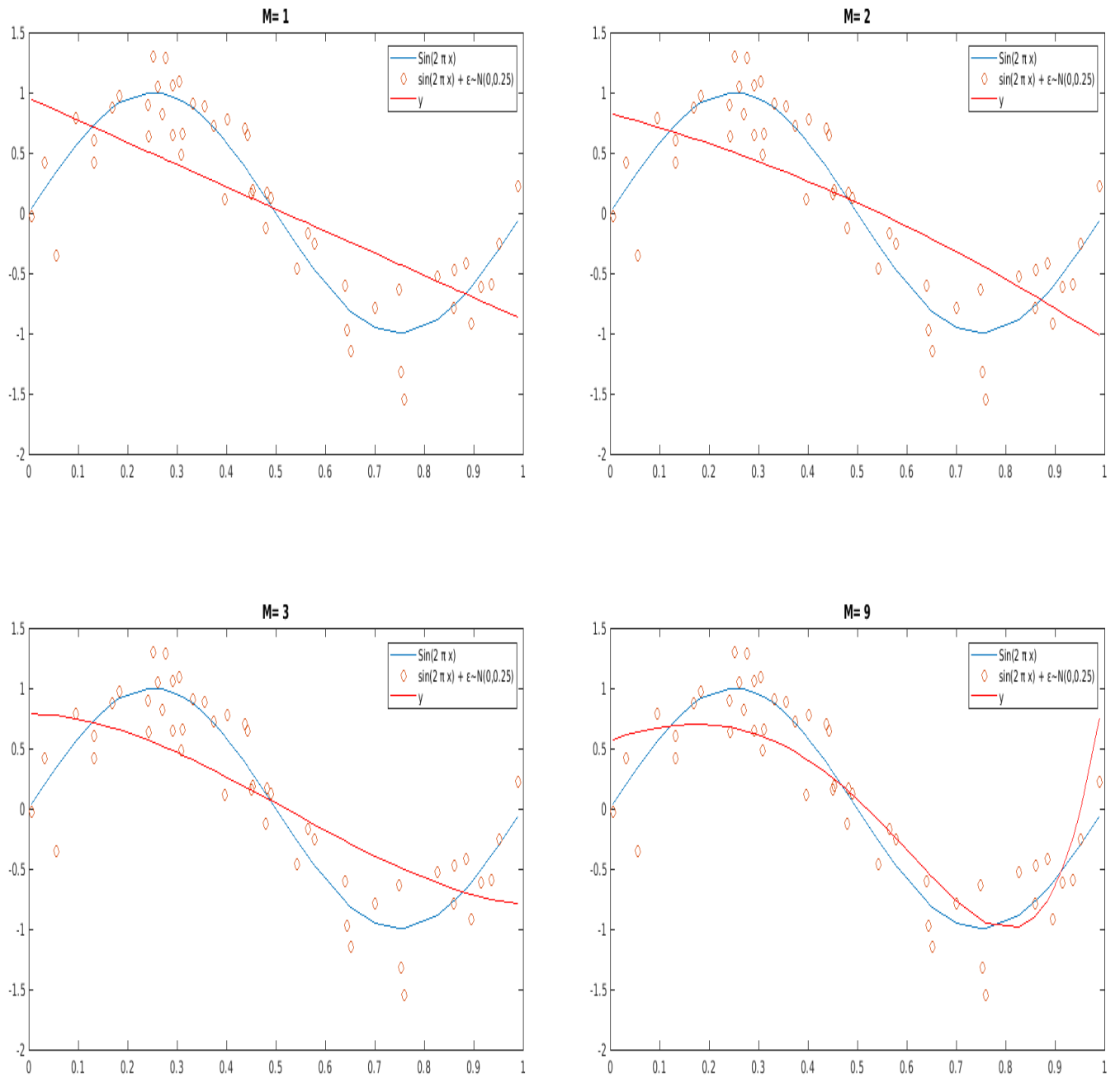


Figure 2: for Training set

File 2

It contains 9th part with X norm and for $M = 1, 2, 5$

```
1
2 clear
3 clc
4
5 x_train= rand(20,1);
6 x_train = sort(x_train);
7
8
9 norm=sqrt(x_train.^2+(x_train.^2).^2);
10 y_train= sin(2*pi*norm) + randn(20,1)*0.25;
11
12
13 x_test = rand(50,1);
14 x_test = sort(x_test);
15 norm_test=sqrt(x_test.^2+(x_test.^2).^2);
16 y_test= sin(2*pi*norm_test) + randn(50,1)*0.25;
17
18
19
20 L=0.01; %Value of lambda
21
22 %M=1;
23 %For Training
24 A = [x_train, ones(20,1)];
25 Coff_1=inv(A'*A+L*eye(size(A'*A)))*(A'*y_train);
26 Y_cap_11= A*Coff_1;
27
28 %For Testing
29 A_test= [x_test, ones(50,1)];
30 Y_cap_12= A_test*Coff_1;
31
32
33 % M=2
34 %For training
35 B = [x_train,x_train.^2,ones(20,1)];
36 Coff_2=inv(B'*B+L*eye(3,3))*(B'*y_train);
37 Y_cap_21= B*Coff_2;
38
39 %For Testing
40 B_test1= [x_test,x_test.^2, ones(50,1)];
41 Y_cap_22= B_test1*Coff_2;
42
43
44
45 % M=5
46 %For Training
47 D = [x_train,x_train.^2,x_train.^3,x_train.^4,x_train.^5,ones(20,1)];
48 Coff_5=inv(D'*D+0.01*eye(size(D'*D)))*(D'*y_train); %coeff
49 Y_cap_51= D*Coff_5;
50
51 %For Testing
52 B_test3= [x_test,x_test.^2,x_test.^3,x_test.^4,x_test.^5, ones(50,1)];
53 Y_cap_52= B_test3*Coff_5;
54
```

```

55
56 RMSE_test_1 = sqrt(mean(Y_cap_12-y_test).^2 );
57 RMSE_test_2 = sqrt(mean(Y_cap_22-y_test).^2 );
58 RMSE_test_5 = sqrt(mean(Y_cap_52-y_test).^2 );
59 disp("RMSE M= 1")
60 disp(RMSE_test_1)
61 disp("RMSE M= 2")
62 disp(RMSE_test_2)
63 disp("RMSE M= 5")
64 disp(RMSE_test_5)
65
66
67 y = sin(2*pi*x_train);
68
69
70 ax1 = nexttile;
71 plot(x_train,y,x_train,y_train,'o',x_train,Y_cap_11,'r'), legend('Sin(2      x)','...
      sin(2      x) +      -N(0,0.25)','y');
72 title(ax1,'M= 1')
73
74 ax2 = nexttile;
75 plot(x_train,y,x_train,y_train,'o',x_train,Y_cap_21,'r'), legend('Sin(2      x)','...
      sin(2      x) +      -N(0,0.25)','y');
76 title(ax2,'M= 2')
77
78
79 ax4 = nexttile;
80 plot(x_train,y,x_train,y_train,'o',x_train,Y_cap_51,'r'), legend('Sin(2      x)','...
      sin(2      x) +      -N(0,0.25)','y');
81 title(ax4,'M= 5')
82
83
84 y = sin(2*pi*x_test);
85
86 ax1 = nexttile;
87 plot(x_test,y,x_test,y_test,'o',x_test,Y_cap_12,'r'), legend('Sin(2      x)','sin...
      (2      x) +      -N(0,0.25)','y');
88 title(ax1,'M= 1')
89
90 ax2 = nexttile;
91 plot(x_test,y,x_test,y_test,'o',x_test,Y_cap_22,'r'), legend('Sin(2      x)','sin...
      (2      x) +      -N(0,0.25)','y');
92 title(ax2,'M= 2')
93
94
95 ax4 = nexttile;
96 plot(x_test,y,x_test,y_test,'o',x_test,Y_cap_52,'r'), legend('Sin(2      x)','sin...
      (2      x) +      -N(0,0.25)','y');
97 title(ax4,'M= 5')

```


0.0.4 Output of training set:

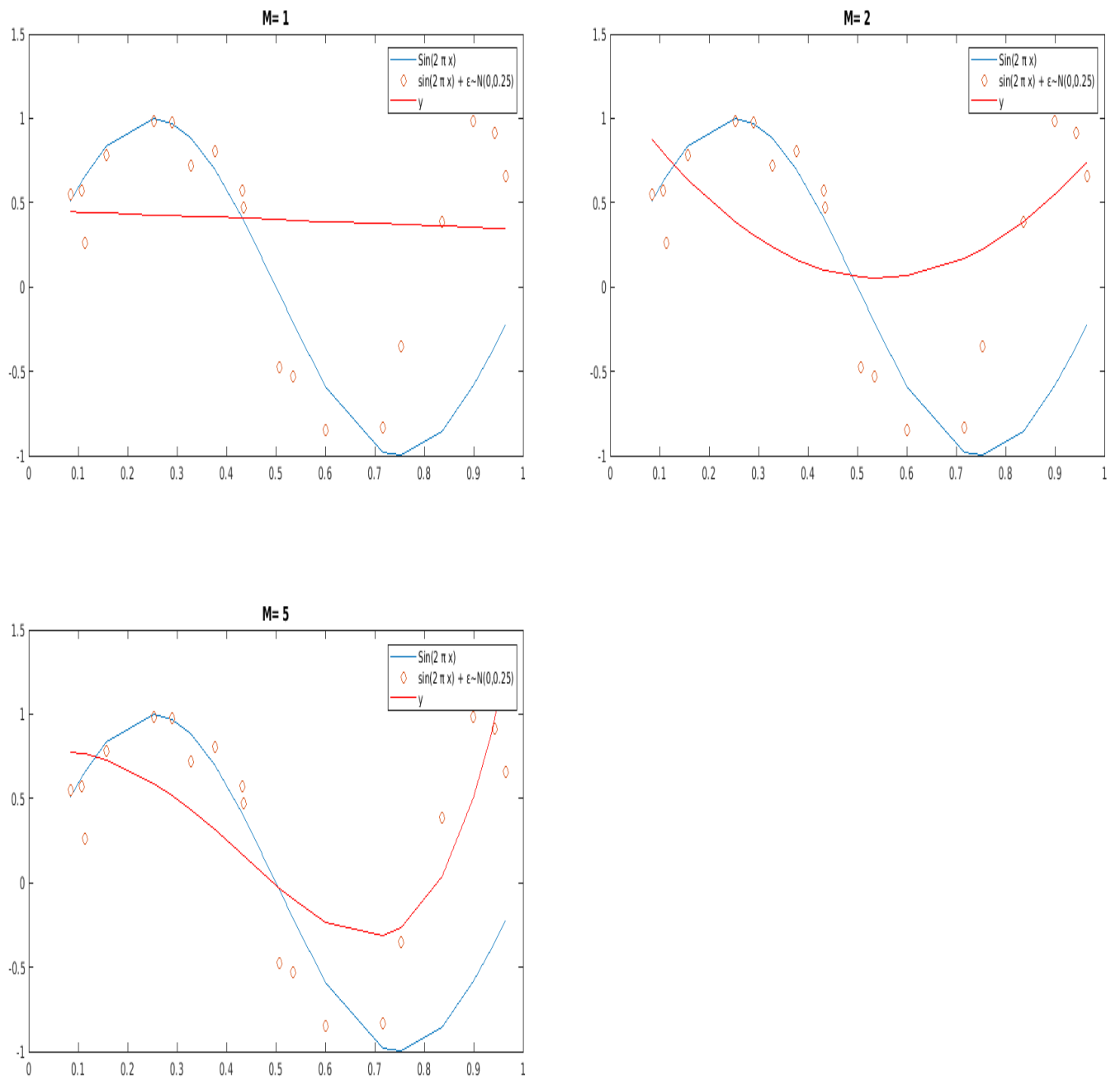


Figure 3: for Training set

0.0.5 Output of testing set

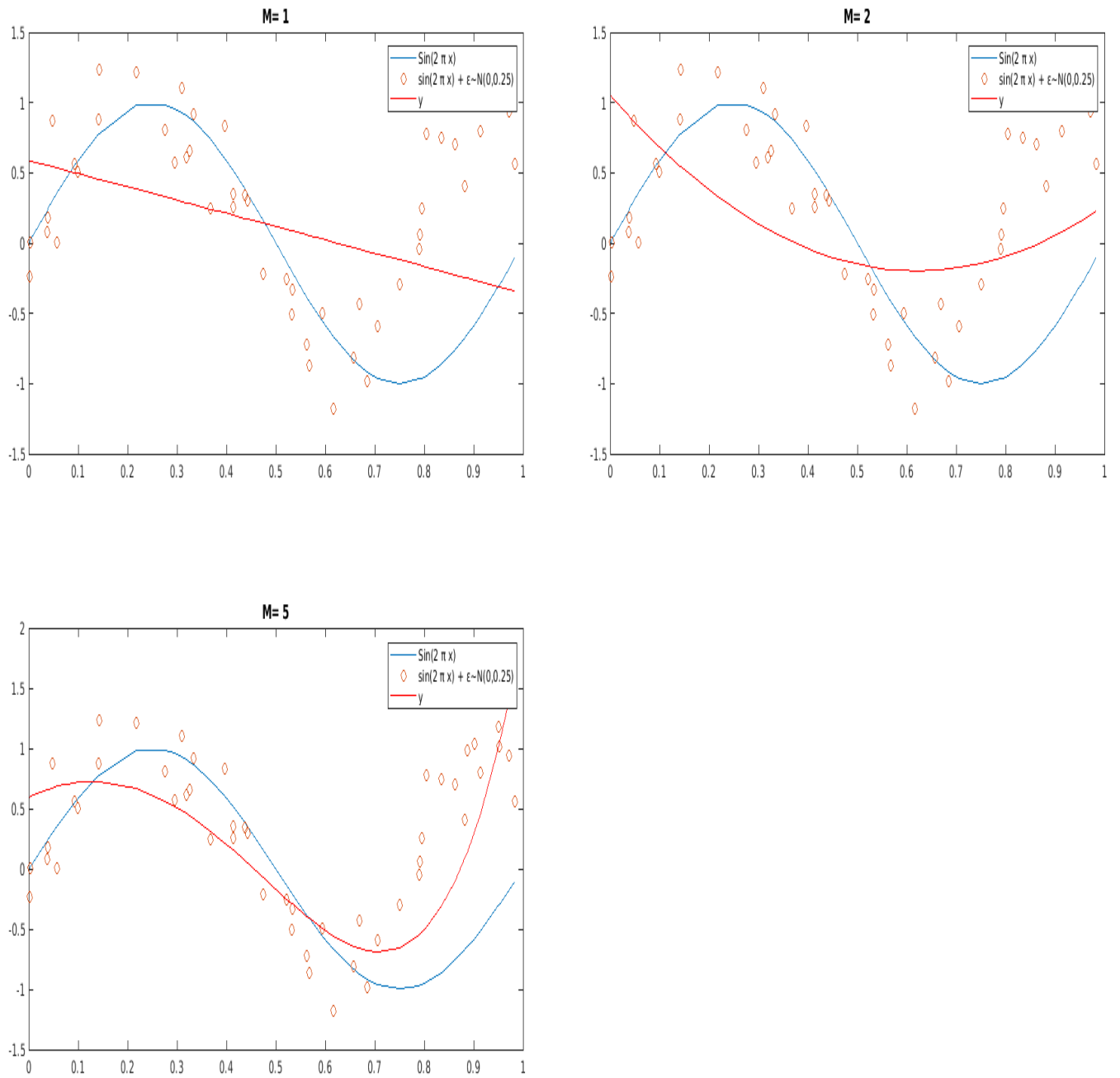


Figure 4: for Training set

Observation

After using the regularized polynomial regression model for higher degree polynomial RMSE value is also low.

As it reduces the overfitting for higher degree polynomial.

For Lambda Value

When choosing a lambda value, the goal is to strike the right balance between simplicity and training-data fit:

- If your lambda value is too high, your model will be simple, but you run the risk of underfitting your data. Your model won't learn enough about the training data to make useful predictions.
- If your lambda value is too low, your model will be more complex, and you run the risk of overfitting your data. Your model will learn too much about the particularities of the training data, and won't be able to generalize to new data.
- As the value of lambda increases, the value of coefficients will get closer to 0 until the value is ultimately 0.

Overall, it's an important technique that can substantially improve the performance of our model.