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 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 .

```

clear
clc

%----------------------------------------------------------------------------%
%Q1
%Random Number for Trainig set
x_train= rand(20,1);
x_train = sort(x_train);
%Q2
%training set Y
y_train= sin(2*pi*x_train) + randn(20,1)*0.25;

%----------------------------------------------------------------------------%
%Q3
%for Testing setnex_trainingttile
x_test = rand(50,1);
x_test = sort(x_test);

y_test= sin(2*pi*x_test) + randn(50,1)*0.25;


%----------------------------------------------------------------------------%
%4
fprintf("Q4, Q5. Estimate regularized least square polynomial regression mode ...
    for M =1,2,3,9  \n and Coffecient \n\n")

L=0.01; %Value of lambda

%M=1;
%For Training
A = [x_train, ones(20,1)];
Coff_1=inv(A'*A+L*eye(size(A'*A)))*(A'*y_train);
Y_cap_11= A*Coff_1;

```

```
%For Testing
A_test= [x_test , ones(50,1)];
Y_cap_12= A_test*Coff_1;


% M=2
%For training
B = [x_train,x_train.^2,ones(20,1)];
Coff_2=inv(B'*B+L*eye(3,3))*(B'*y_train);
Y_cap_21= B*Coff_2;

%For Testing
B_test1= [x_test,x_test.^2, ones(50,1)];
Y_cap_22= B_test1*Coff_2;


% M=3
%For Training
C = [x_train,x_train.^2,x_train.^3,ones(20,1)];
Coff_3=inv(C'*C+0.01*eye(4,4))*(C'*y_train); %coff
Y_cap_31= C*Coff_3;

%For Testing
B_test2= [x_test,x_test.^2,x_test.^3, ones(50,1)];
Y_cap_32= B_test2*Coff_3;


% M=9
%For Training
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)];
Coff_9=inv(D'*D+0.01*eye(size(D'*D)))*(D'*y_train); %coeff
Y_cap_91= D*Coff_9;

%For Testing
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)];
Y_cap_92= B_test3*Coff_9;

%---------------------------------------------------------------------------%
%Q6   RMSE Values of Testing values

RMSE_test_1 = sqrt(mean(Y_cap_12-y_test).^2 );
RMSE_test_2 = sqrt(mean(Y_cap_22-y_test).^2 );
RMSE_test_3 = sqrt(mean(Y_cap_32-y_test).^2 );
RMSE_test_9 = sqrt(mean(Y_cap_92-y_test).^2 );
disp("RMSE M= 1")
disp(RMSE_test_1)
disp("RMSE M= 2")
disp(RMSE_test_2)
disp("RMSE M= 3")
disp(RMSE_test_3)
disp("RMSE M= 9")
disp(RMSE_test_9)

% Q7
y = sin(2*pi*x_train);
```

```
ax1 = nexttile;
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');
title(ax1,'M= 1')

ax2 = nexttile;
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');
title(ax2,'M= 2')

ax3 = nexttile;
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');
title(ax3,'M= 3')

ax4 = nexttile;
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');
title(ax4,'M= 9')

% Q8

y = sin(2*pi*x_test);

ax1 = nexttile;
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');
title(ax1,'M= 1')

ax2 = nexttile;
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');
title(ax2,'M= 2')

ax3 = nexttile;
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');
title(ax3,'M= 3')

ax4 = nexttile;
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');
title(ax4,'M= 9')
```

0.0.2 Output of training set:

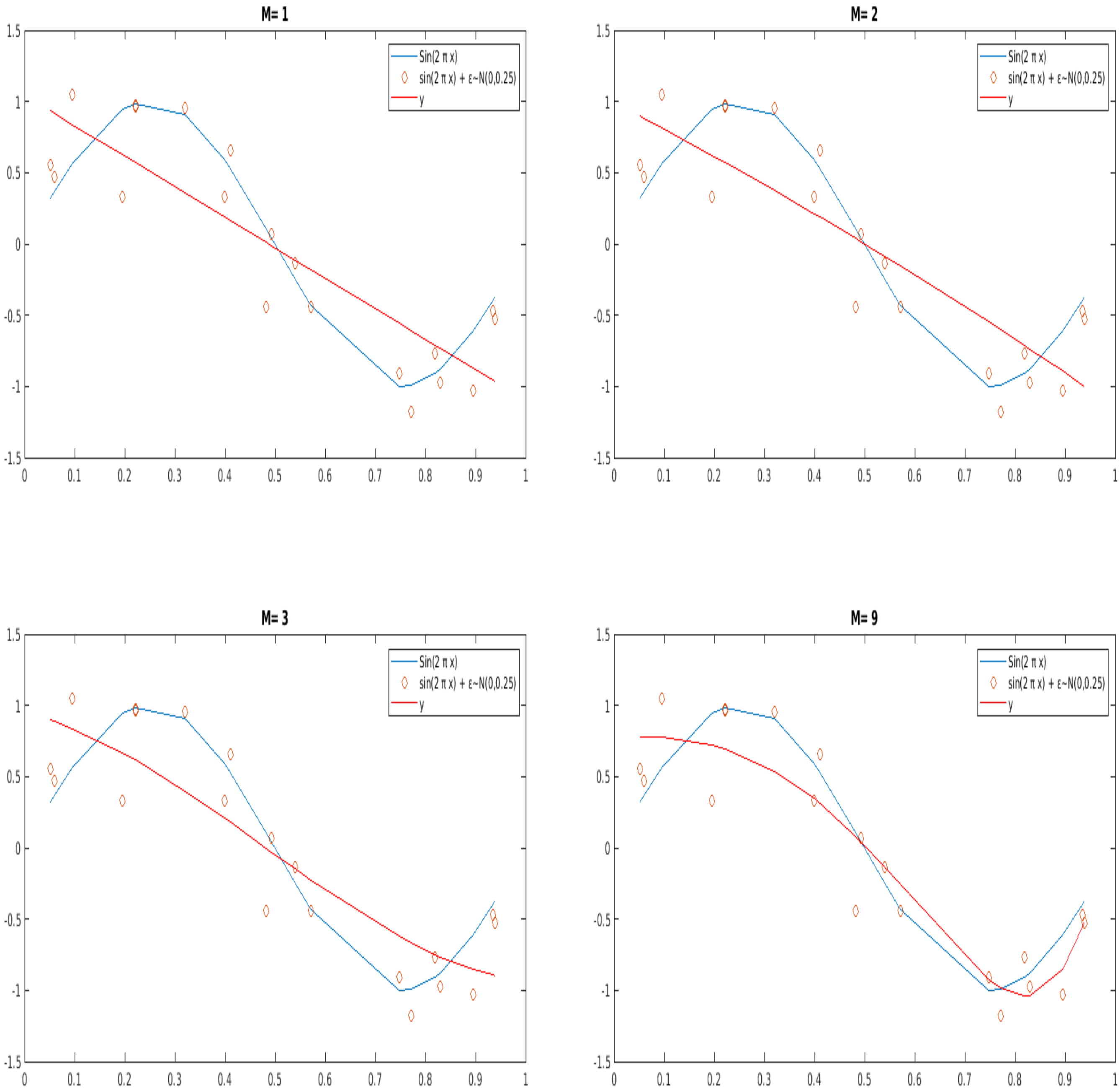

Figure 1: for Training set

0.0.3 Output of testing set:

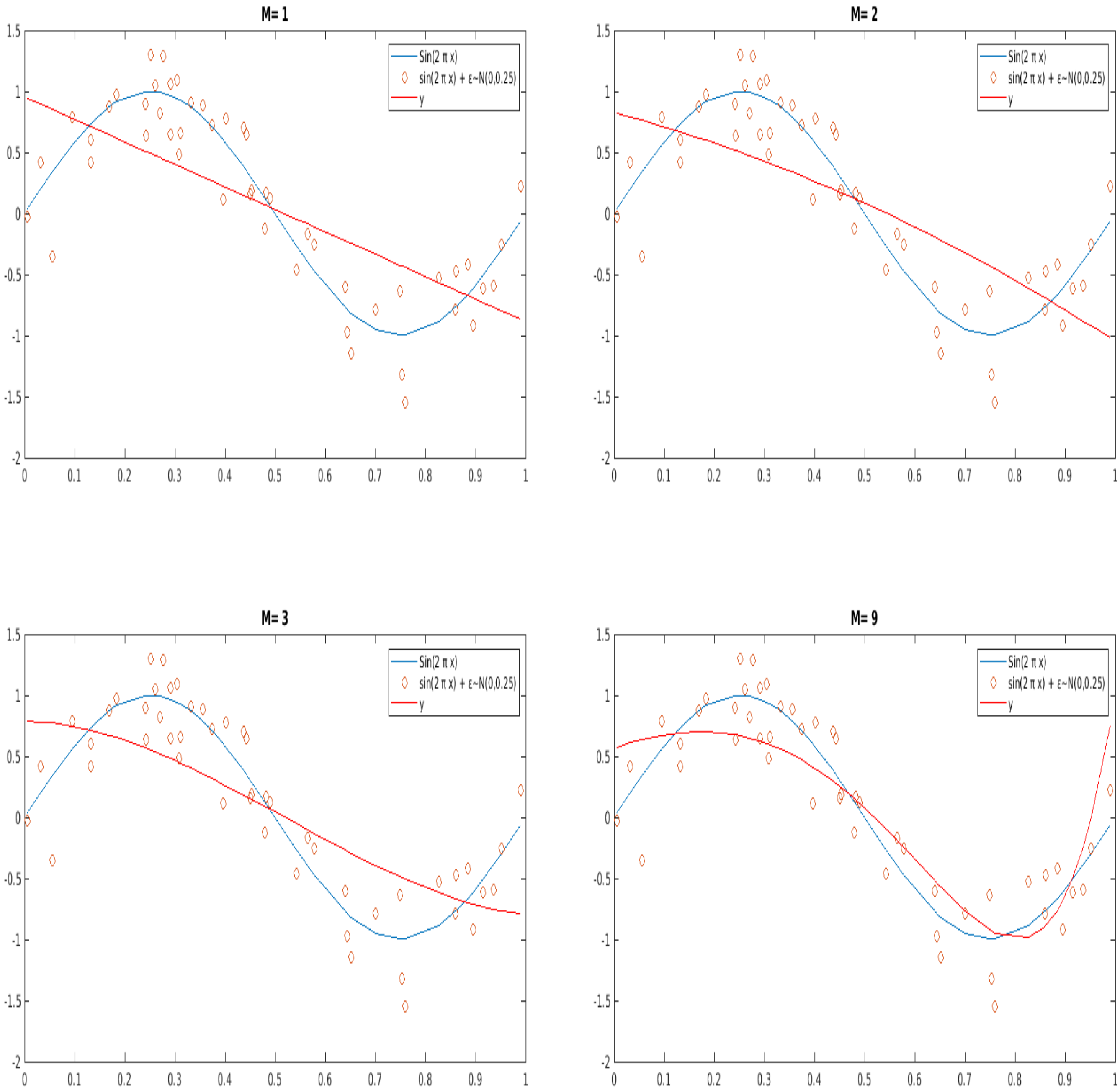

Figure 2: for Training set

File 2

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

```

clear
clc

x_train= rand(20,1);
x_train = sort(x_train);


norm=sqrt(x_train.^2+(x_train.^2).^2);
y_train= sin(2*pi*norm) + randn(20,1)*0.25;


x_test = rand(50,1);
x_test = sort(x_test);
norm_test=sqrt(x_test.^2+(x_test.^2).^2);
y_test= sin(2*pi*norm_test) + randn(50,1)*0.25;



L=0.01; %Value of lambda

%M=1;
%For Training
A = [x_train, ones(20,1)];
Coff_1=inv(A'*A+L*eye(size(A'*A)))*(A'*y_train);
Y_cap_11= A*Coff_1;

%For Testing
A_test= [x_test, ones(50,1)];
Y_cap_12= A_test*Coff_1;


% M=2
%For training
B = [x_train,x_train.^2,ones(20,1)];
Coff_2=inv(B'*B+L*eye(3,3))*(B'*y_train);
Y_cap_21= B*Coff_2;

%For Testing
B_test1= [x_test,x_test.^2, ones(50,1)];
Y_cap_22= B_test1*Coff_2;



% M=5
%For Training
D = [x_train,x_train.^2,x_train.^3,x_train.^4,x_train.^5,ones(20,1)];
Coff_5=inv(D'*D+0.01*eye(size(D'*D)))*(D'*y_train); %coeff
Y_cap_51= D*Coff_5;

%For Testing
B_test3= [x_test,x_test.^2,x_test.^3,x_test.^4,x_test.^5, ones(50,1)];
Y_cap_52= B_test3*Coff_5;

```

```
RMSE_test_1 = sqrt(mean(Y_cap_12-y_test).^2 );
RMSE_test_2 = sqrt(mean(Y_cap_22-y_test).^2 );
RMSE_test_5 = sqrt(mean(Y_cap_52-y_test).^2 );
disp("RMSE M= 1")
disp(RMSE_test_1)
disp("RMSE M= 2")
disp(RMSE_test_2)
disp("RMSE M= 5")
disp(RMSE_test_5)


y = sin(2*pi*x_train);


ax1 = nexttile;
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');
title(ax1,'M= 1')

ax2 = nexttile;
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');
title(ax2,'M= 2')


ax4 = nexttile;
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');
title(ax4,'M= 5')


y = sin(2*pi*x_test);

ax1 = nexttile;
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');
title(ax1,'M= 1')

ax2 = nexttile;
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');
title(ax2,'M= 2')


ax4 = nexttile;
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');
title(ax4,'M= 5')
```

0.0.4 Output of training set:

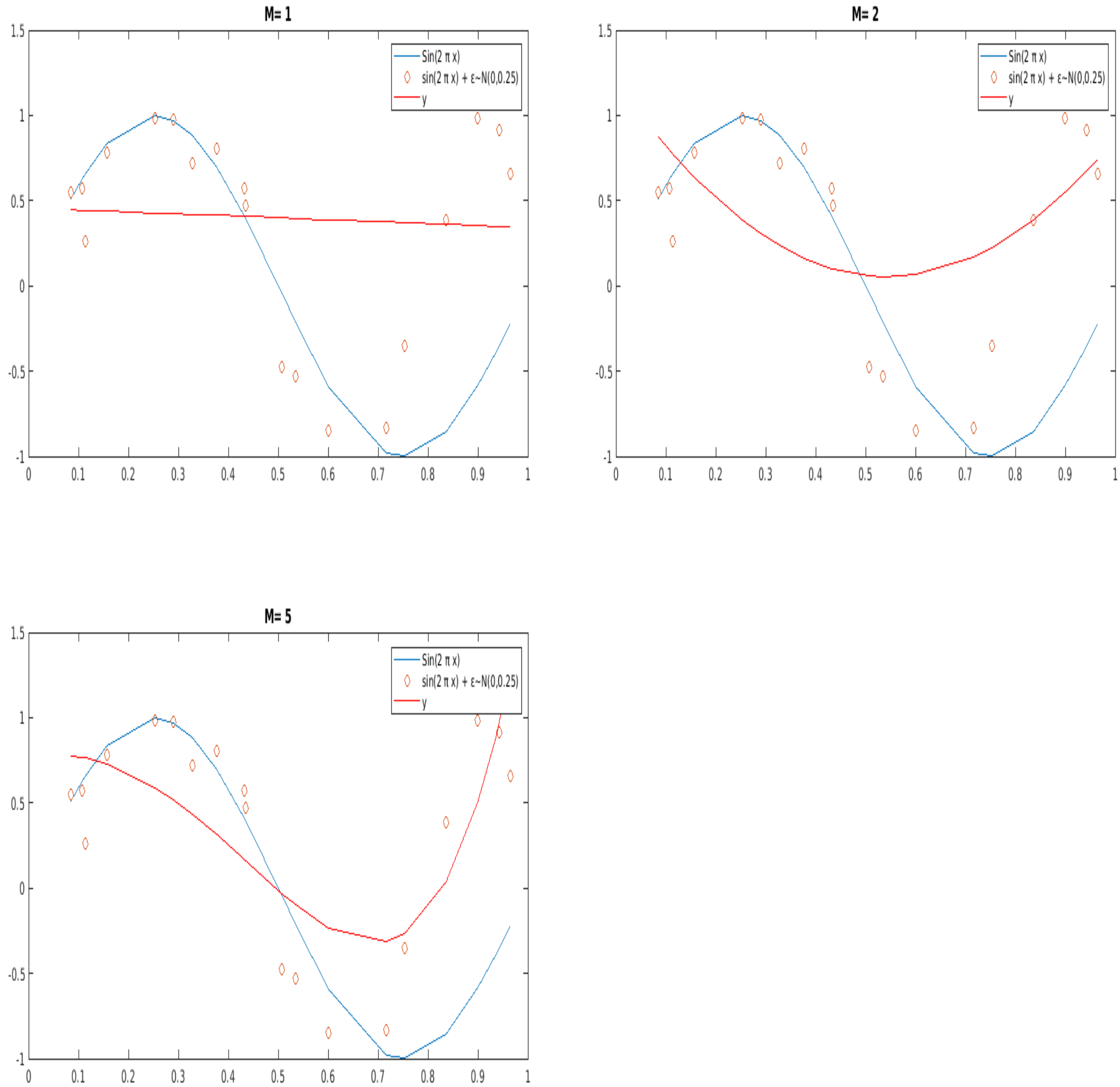

Figure 3: for Training set

0.0.5 Output of testing set

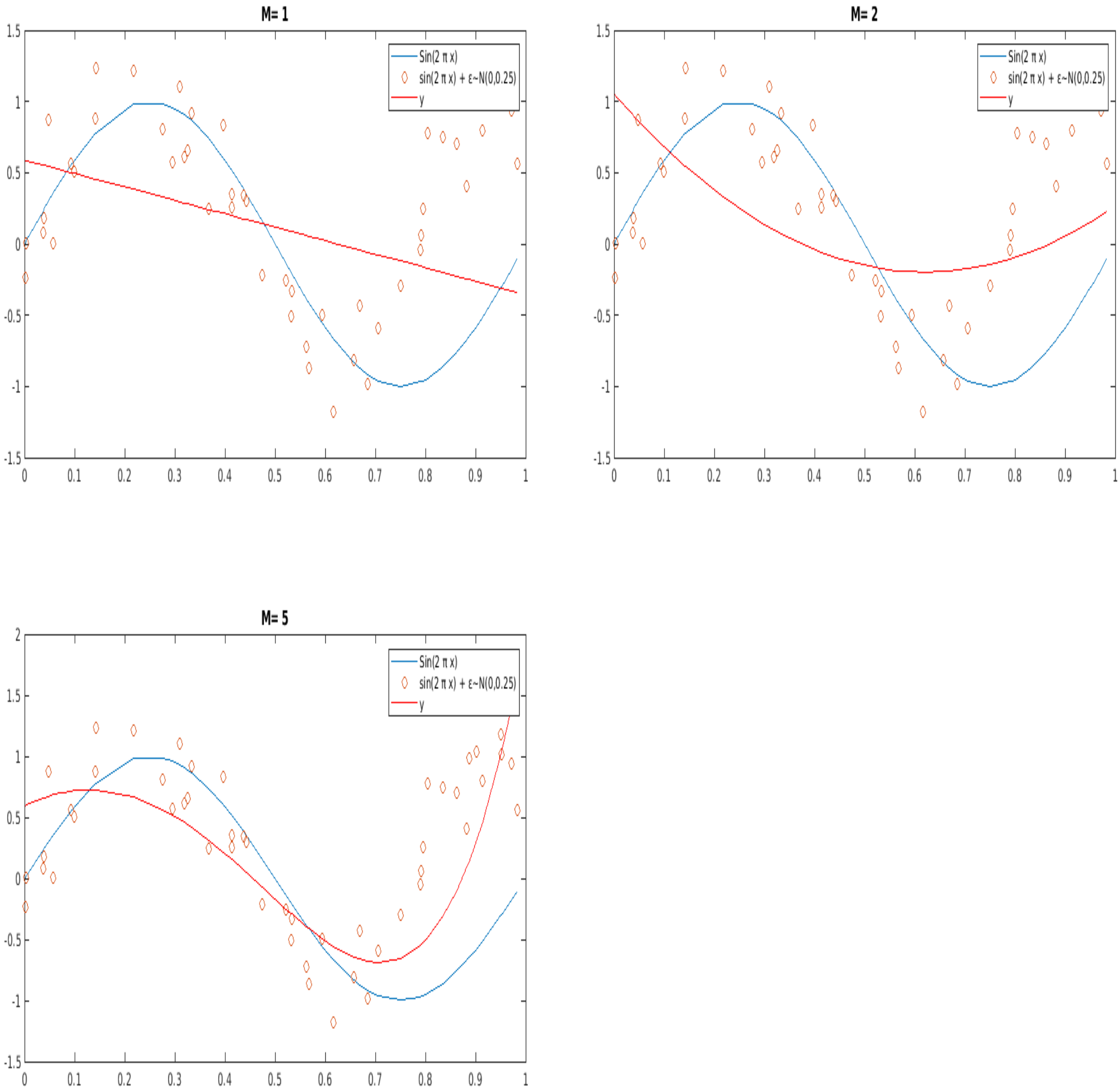

Figure 4: for Training set

Observation

After using the regularized polynomial regression model for higer 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.