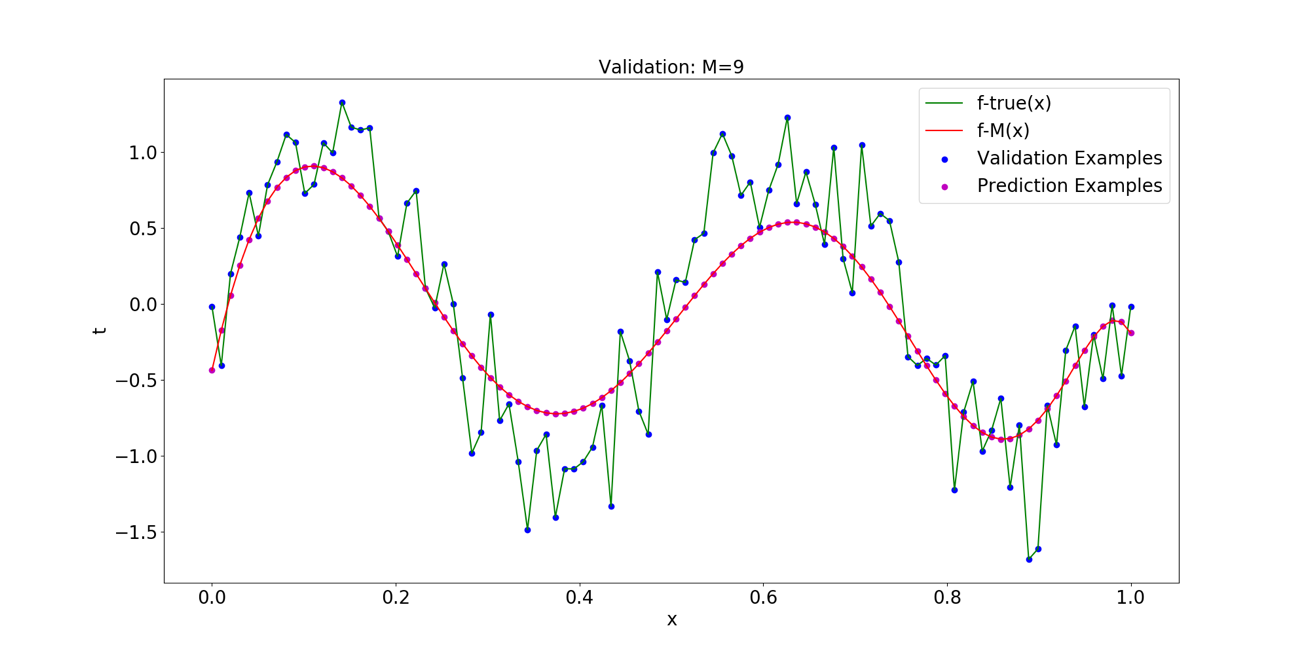
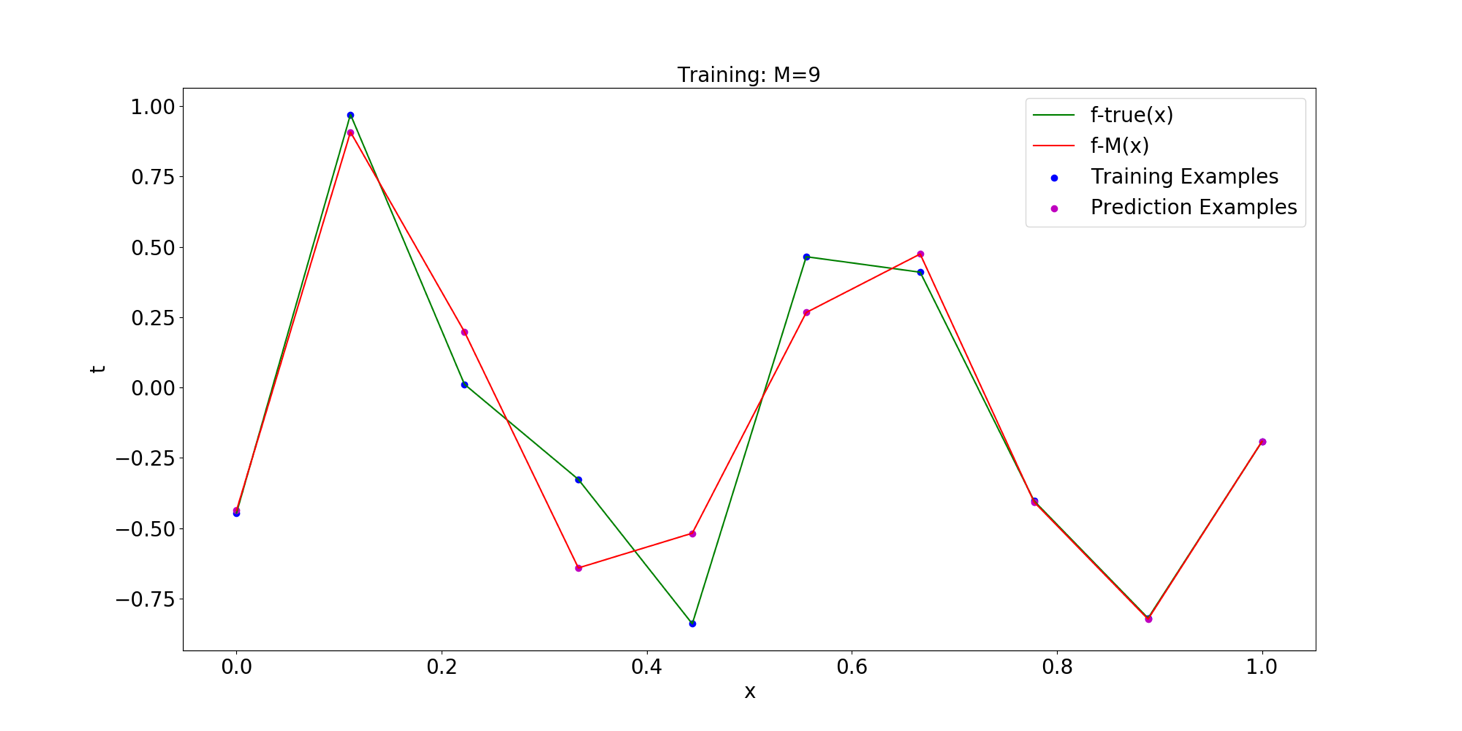
Assignment 1

**Trade-off between Overfitting and Underfitting**

*September 29, 2021*



Catalogue

[Part 1 The Introduction 3](#_Toc83858030)

[1.1The purposes 3](#_Toc83858031)

[1.2The Requirements 3](#_Toc83858032)

[Part 2 The Principle 3](#_Toc83858033)

[2.1 Polynomial Regression 3](#_Toc83858034)

[2.2 Ridge Regression 4](#_Toc83858035)

[2.3 Overfitting and Underfitting 5](#_Toc83858036)

[2.3.1 Overfitting 5](#_Toc83858037)

[2.3.2 Underfitting 5](#_Toc83858038)

[2.4 Model Evaluation 5](#_Toc83858039)

[2.4.1 Mean Absolute Error 5](#_Toc83858040)

[2.4.2 Coefficient of Determination 5](#_Toc83858041)

[Part 3 The Procedure 6](#_Toc83858042)

[3.1 Experimental Environment 6](#_Toc83858043)

[3.2 Program Flow Chart 6](#_Toc83858044)

[3.3 Program Description 7](#_Toc83858045)

[3.3.1 Function Description 7](#_Toc83858046)

[3.3.2 Program Operation Instructions 7](#_Toc83858047)

[Part 4 The Results 8](#_Toc83858048)

[4.1 Fitting Situation 8](#_Toc83858049)

[4.1.1 M = 0 9](#_Toc83858050)

[4.1.2 M = 1 10](#_Toc83858051)

[4.1.3 M = 2 10](#_Toc83858052)

[4.1.4 M = 3 11](#_Toc83858053)

[4.1.5 M = 4 12](#_Toc83858054)

[4.1.6 M = 5 13](#_Toc83858055)

[4.1.7 M = 6 14](#_Toc83858056)

[4.1.8 M = 7 15](#_Toc83858057)

[4.1.9 M = 8 16](#_Toc83858058)

[4.1.9 M = 9 17](#_Toc83858059)

[4.2 Influence of M on Error 21](#_Toc83858060)

[Part 5 The Conclusion & Discussion 22](#_Toc83858061)

[5.1 Conclusion 22](#_Toc83858062)

[5.2 Discussion 22](#_Toc83858063)

[Reference 23](#_Toc83858064)

[Appendix 24](#_Toc83858065)

# Part 1 The Introduction

## The purposes

1. Master the basic principles of polynomial regression;
2. Master the basic principles of ridge regression;
3. Master the calculation method of model training and verification error;
4. Compare the fitting of data under different orders of polynomials
5. Compare the effects of different penalty factors on the model in ridge regression
6. Proficient in using Python for algorithm implementation

## The Requirements

1. Generate training and verification data according to the requirements of Formula (1)
2. Plot the fitting situation of polynomial model under different orders. Use different colours for the training examples, the validation examples, the prediction function and the true function.
3. When M = 9, train the model with regularization and compare the fitting situation under different penalty factor λ.
4. Plot the average error between the targets and the true function for the examples in the validation set.

# Part 2 The Principle

## 2.1 Polynomial Regression

Data of this experiment is generated from the function with random noise included in the target values. That means the data is based on only one feature, denoted by . The target is and it satisfies the following relation

|  |  |
| --- | --- |
|  | (1) |

where is random noise with a Gaussian distribution with 0 mean and variance 0.09. Now suppose that the training set consists of *N* observations of , denoted , then the corresponding observations can be obtained from the relation (1), denoted . The value range of is [0,1]. There are 10 samples in the training set and 100 samples in the validation set.

The goal of curve fitting is to use this training set to predict the value of the target variable for some new value of the input variable. In the polynomial regression, we use the following form of polynomial function to fit the data

|  |  |
| --- | --- |
|  | (2) |

where is the order of the polynomial, and denotes raised to the power of . The polynomial coefficients are collectively denoted by the vector .

The value of the coefficient can be determined by adjusting the polynomial function to fit the training data. This can be achieved by minimizing the error function which measures the difference between the function and the training data for any given value of . One commonly used error function is obtained by calculating the square sum of the errors between the predictions for each data point and the corresponding target values . The expression is as follows

|  |  |
| --- | --- |
|  | (3) |

The order of the polynomial is also very important to the fitting effect of the model. Too small value of may give rather poor fits to the data and consequently rather poor representations of the function while too large value of may result the fitted curve oscillates wildly.

## 2.2 Ridge Regression

In the polynomial regression, as M increases, the magnitude of the coefficients typically gets larger. And the certainty of the coefficient will become worse and show high variance. Although the model can accurately match the training set data, the prediction effect will become worse. This phenomenon is called overfitting.

One technique often used to control the overfitting phenomenon is regularization, which involves adding a penalty term to the error function so that the coefficient will not reach a very high value. Regularization can be implemented in two ways, called *L1* *norm* and *L2* *norm*. *L1* *norm* is expressed as and *L2* *norm* is expressed as . Their respective expressions are as follows

|  |  |
| --- | --- |
|  | (4) |
|  | (5) |

Linear regression with *L1* *norm* is called *lasso regression* and linear regression with *L2* *norm* is called *ridge regression.* The error function of ridge regression is as follows

|  |  |
| --- | --- |
|  | (6) |

where the coefficient is called penalty factor. It governs the relative importance of the regularization term compared with the error term, controls the effective complexity of the model and hence determines the degree of overfitting.

## 2.3 Overfitting and Underfitting

### 2.3.1 Overfitting

when the model adapts "too well" to the training sample, it is likely to have regarded some characteristics of the training sample as the general properties of all potential samples, which will lead to the decline of generalization performance. This phenomenon is called overfitting. The specific performance is that the final model has a good effect on the training set while has a poor effect on the test set and there has a large gap between the training error and the test error.

### 2.3.2 Underfitting

When the model cannot fit the training samples “sufficiently well”, leading to a high training error. This phenomenon is called underfitting. The specific performance is that the final model in training set and test set is not good.

## 2.4 Model Evaluation

### 2.4.1 Mean Absolute Error

The *mean absolute error* *(MAE)* can avoid the problem of error cancellation, so it can accurately reflect the actual prediction error. Therefore, the *MAE* is used to evaluate the ability of the model to fit the data in this experiment. *MAE* is calculated as follows

|  |  |
| --- | --- |
|  | (7) |

### 2.4.2 Coefficient of Determination

In this experiment, the *coefficient of determination (R-square)* is used to evaluate the fitting effect of the model. From the above expression, we can know that the normal value range of *R-square* is [0,1]. The closer it is to 1, the better the model fits the data. *R-square* is calculated as follows

|  |  |
| --- | --- |
|  | (8) |

# Part 3 The Procedure

## 3.1 Experimental Environment

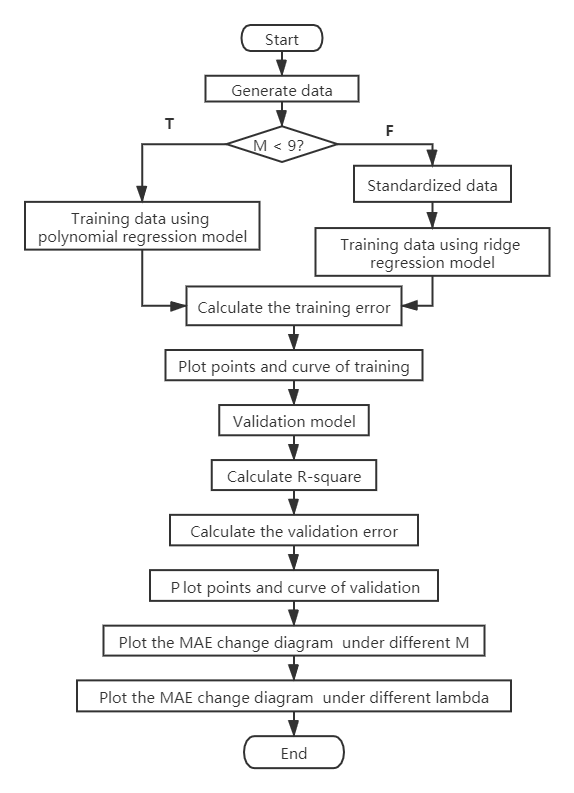
The experimental environment is shown in Table 1. See Appendix for codes.

#### Table 1. Experimental environment

|  |  |
| --- | --- |
| *IDE* | *Pycharm* |
| *Language* | *Python 3.7* |
| *Package* | *os* |
| *numpy* |
| *matplotlib* |
| *sklearn* |

## 3.2 Program Flow Chart

The program flow chart is shown in Fig.1.



#### Fig.1: The program flow chart

## 3.3 Program Description

### 3.3.1 Function Description

*1) DataGenerate.* The training set and the validation set are generated by this function.

*2) train.* Use training set data to train model. When *M* is equal to 0 to 8, polynomial regression is used to train the model. The polynomial regression is realized by calling function *polyfit()* in *numpy* library*.* The ridge regression is realized by function *Ridge()* in *sklearn.linear\_model.*

3) valid. Use validation set data to verify the model.

4) *cal\_MAE.* Calculate MAE using the method in equation (7)

5) *plot\_fig.* Plot as required (2). Plot the fitting situation of polynomial model under different orders. Use different colours for the training examples, the validation examples, the prediction function and the true function.

6) *plot\_error\_fig.* Plot as required (4). Plot the changes of training error and verification error under different m values.

7) *plot\_curve\_lamb.* Plot the changes of training error and verification error under different λ values.

The function is described in Table 2.

### 3.3.2 Program Operation Instructions

All the requirements of this experiment can be completed directly by running the code without entering any parameters manually. The pictures required for the experiment will be saved in the folder “Pics” under the root directory of the project. The program will print the training error, the validation error, the R-square and the under different M. When M = 9, The program will additionally output the λ, the training error, the validation error and the R-square with the best generalization performance of the model.

**Table 2.** Function Description

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Function name* | *In* | *Notes* | *Out* | *Notes* |
| *DataGenerate* | *train\_num* | *Number of training set samples* | *x\_train* | *x (training)* |
| *t\_train* | *t (training)* |
| *valid\_num* | *Number of training set samples* | *x\_valid* | *x (validation)* |
| *t\_valid* | *t (validation)* |
| *train* | *x\_train* | *x (training)* | *model* | *model* |
| *t\_train* | *t (training)* | *Mean\_error* | *MAE (training)* |
| *M* | *M* | *prediction* | *t (training)* |
| *lamb* | *λ* | *f* |  |
| *valid* | *x\_valid* | *x (validation)* | *Mean\_error* | *MAE (validation)* |
| *t\_valid* | *t (validation)* | *R\_square* | *R\_square (Coefficient of Determination)* |
| *model* | *model* | *prediction* | *t (validation)* |
| *M* | *M (Polynomial order)* |  |  |
| *cal\_MAE* | *y* | *t* | *error* | *MAE* |
| *y\_pre* |  |  |  |
| *plot\_fig* | *x* | *x* |  |  |
| *y* | *t* |  |  |
| *prediction* |  |  |  |
| *Training\_or\_Validation* | *Used to determine the figure title* |  |  |
| *M* | *M (Polynomial order)* |  |  |
| *lamb* | *λ* |  |  |
| *plot\_error\_fig* | *train\_error\_list* | *List for saving MAE of validation* |  |  |
| *valid\_error\_list* | *List for saving MAE of validation* |  |  |
| *plot\_curve\_lamb* | *M* | *M (Polynomial order)* |  |  |

# Part 4 The Results

## 4.1 Fitting Situation

This section will show the fitting situation of the model on the training set and verification set when the polynomial order *M* takes different values. In the figure, the abscissa represents the value of *x*, and the ordinate represents the corresponding value of *t*, the blue dots represent the training set samples, the purple dots represent the training set samples, the green curve represents the real function value and the red curve represents the prediction function value.

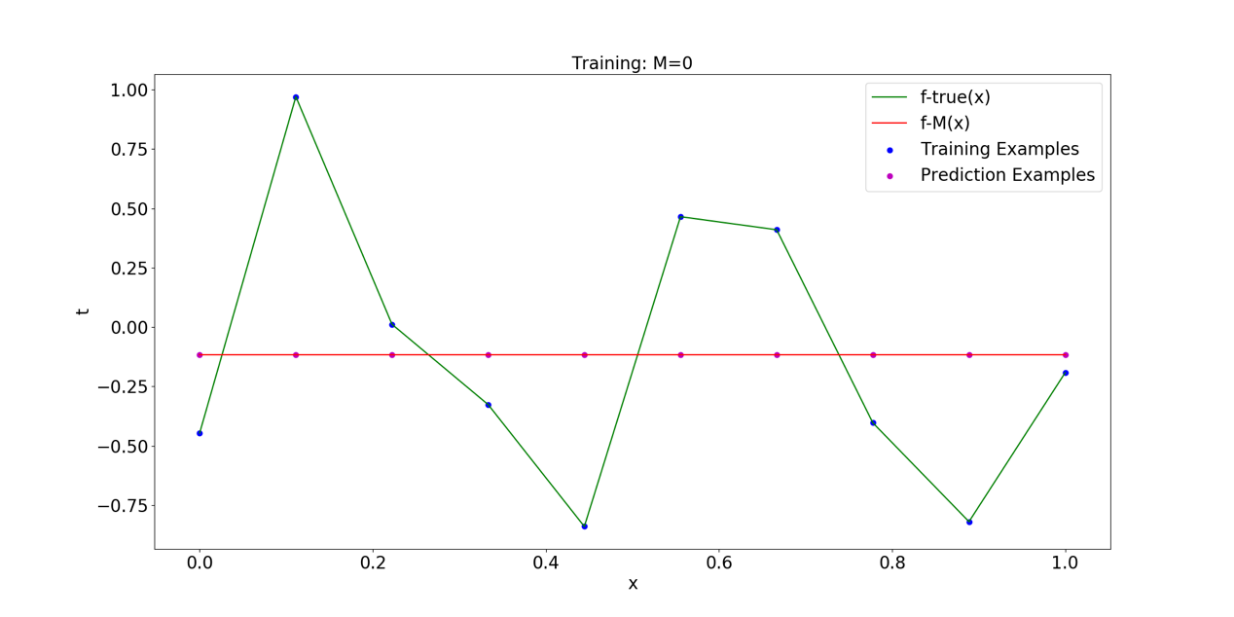
**Table 3.** Plot description

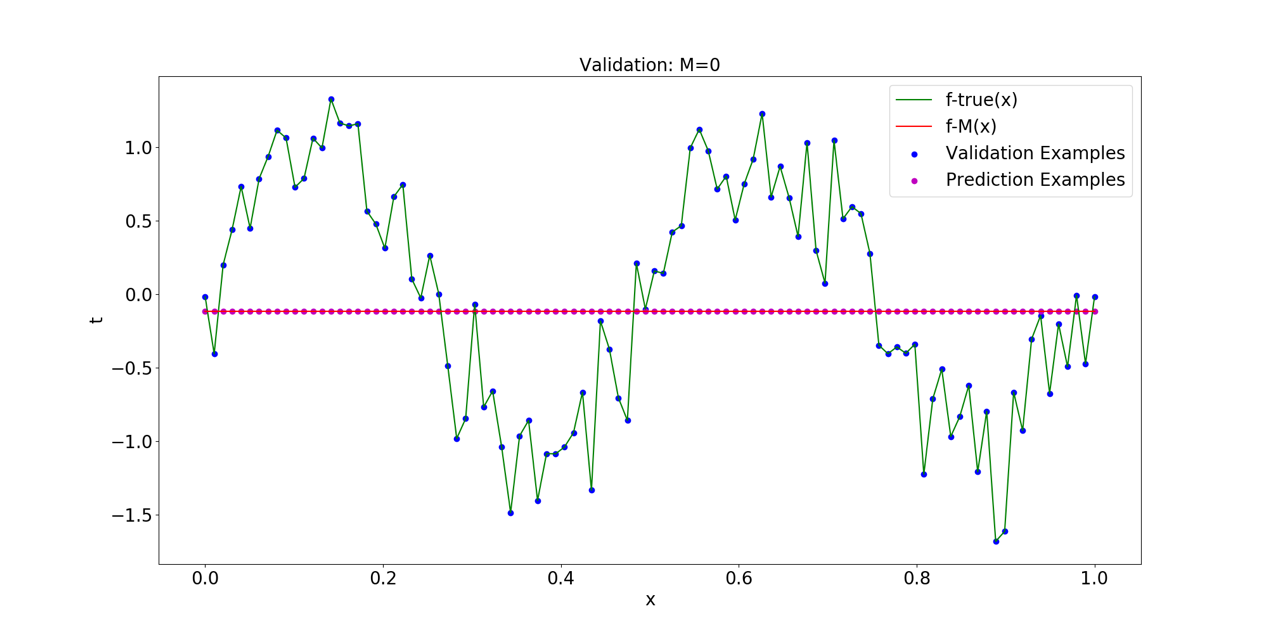
|  |  |  |
| --- | --- | --- |
| *Name* | *Shape* | *Colour* |
| *training examples* | *dot* | *blue* |
| *validation examples* | *dot* | *purple* |
| *prediction function* | *curve* | *red* |
| *true function* | *curve* | *green* |

### 4.1.1 M = 0

when *M* = 0, . It can also be seen from the figure that the fitted curve is a straight line. The relevant records of the model are as follows

The errors of training set and verification set are very high, and the value of determination coefficient is very small, so the model is underfitting.



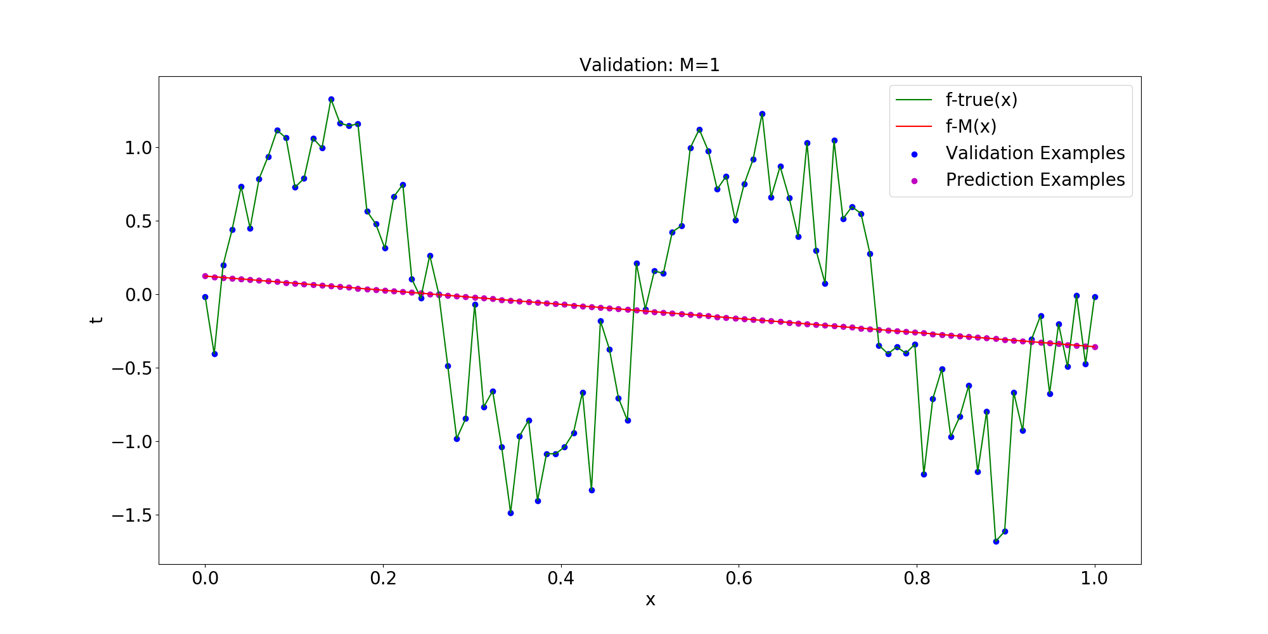
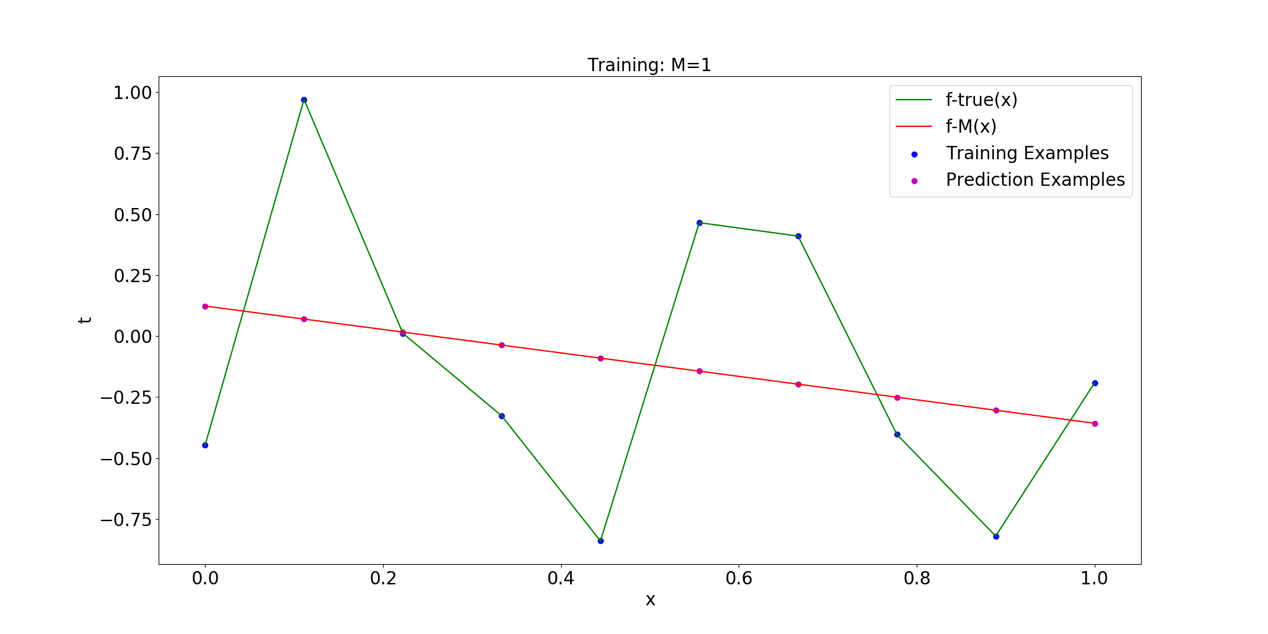


#### Fig.2: Fitting situation (M=0)

### 4.1.2 M = 1

when *M* = 1, the relevant records of the model are as follows

The errors of training set and verification set are very high, and the value of determination coefficient is very small, so the model is underfitting.

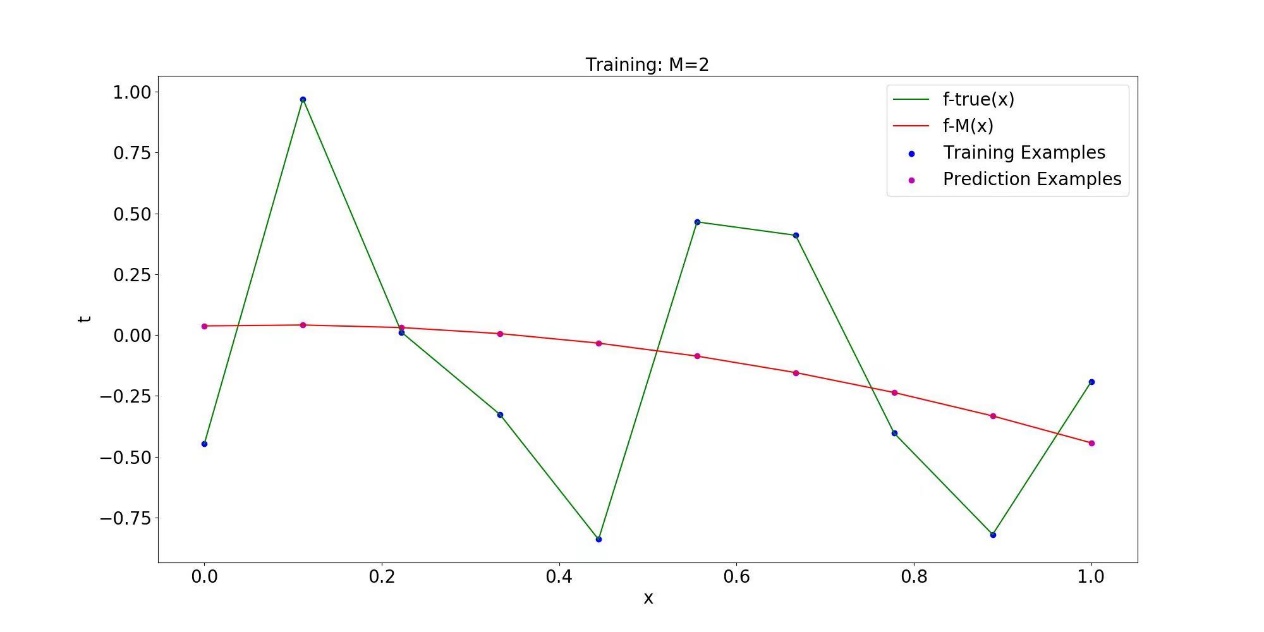


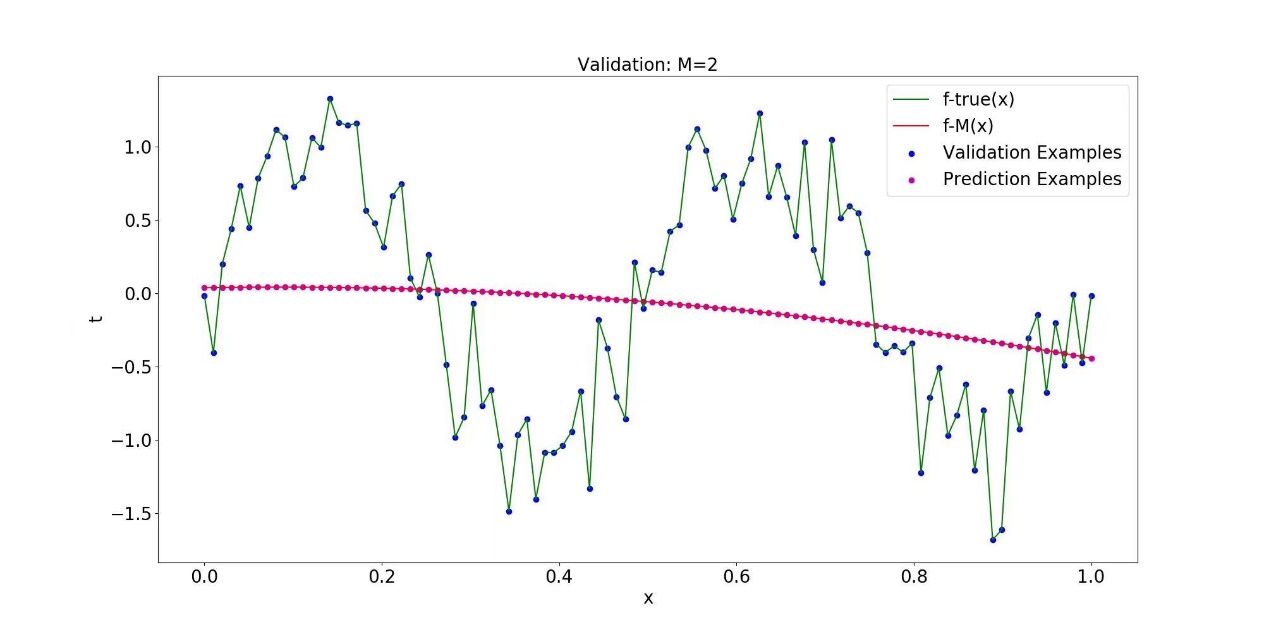
#### Fig.3: Fitting situation (M=1)

### 4.1.3 M = 2

when *M* = 2, the relevant records of the model are as follows

The errors of training set and verification set are very high, and the value of determination coefficient is very small, so the model is underfitting.

****

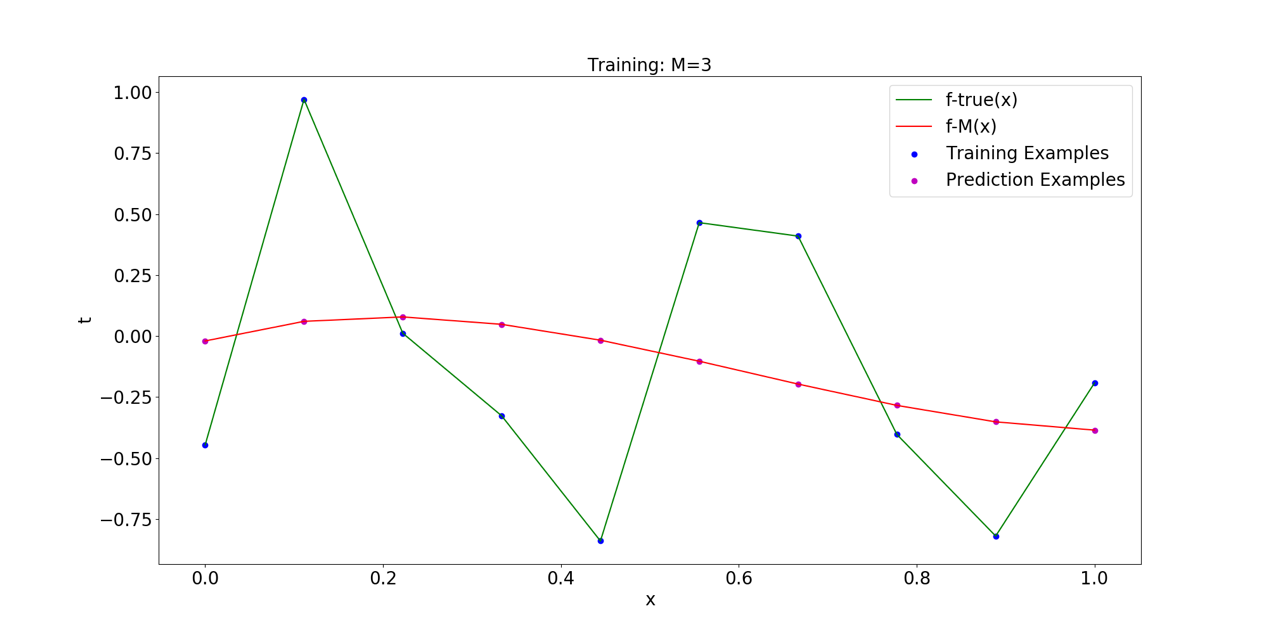


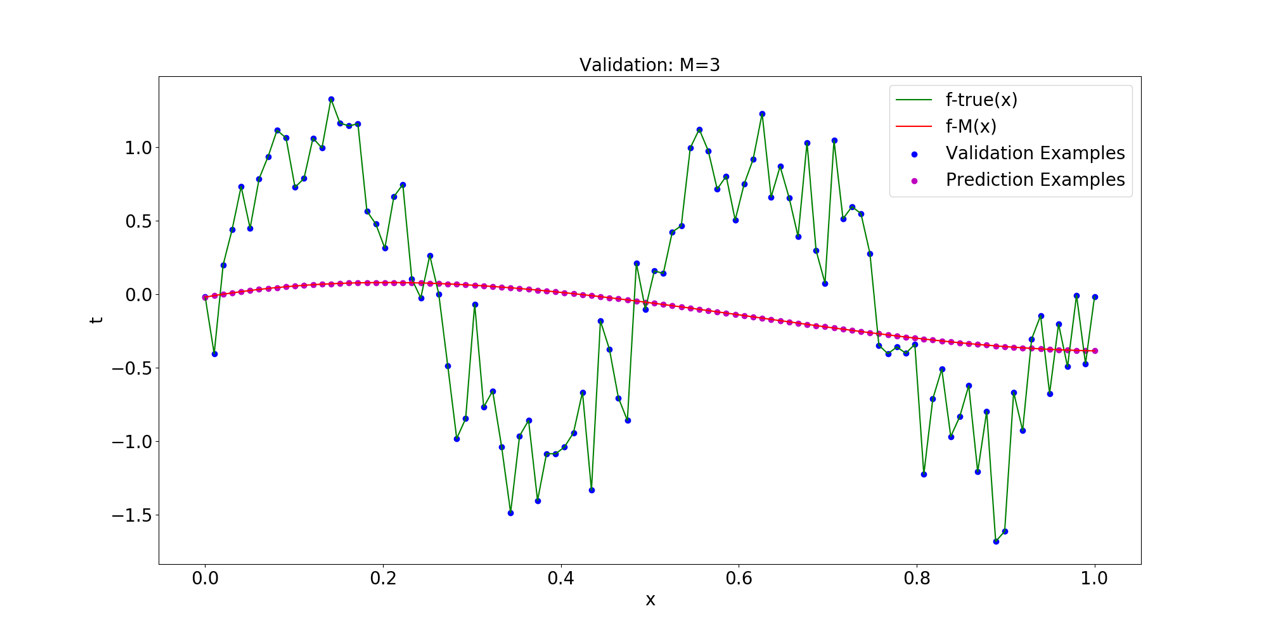
#### Fig.3: Fitting situation (M=2)

### 4.1.4 M = 3

when *M* = 3, the relevant records of the model are as follows

The errors of training set and verification set are very high, and the value of determination coefficient is very small, so the model is underfitting.



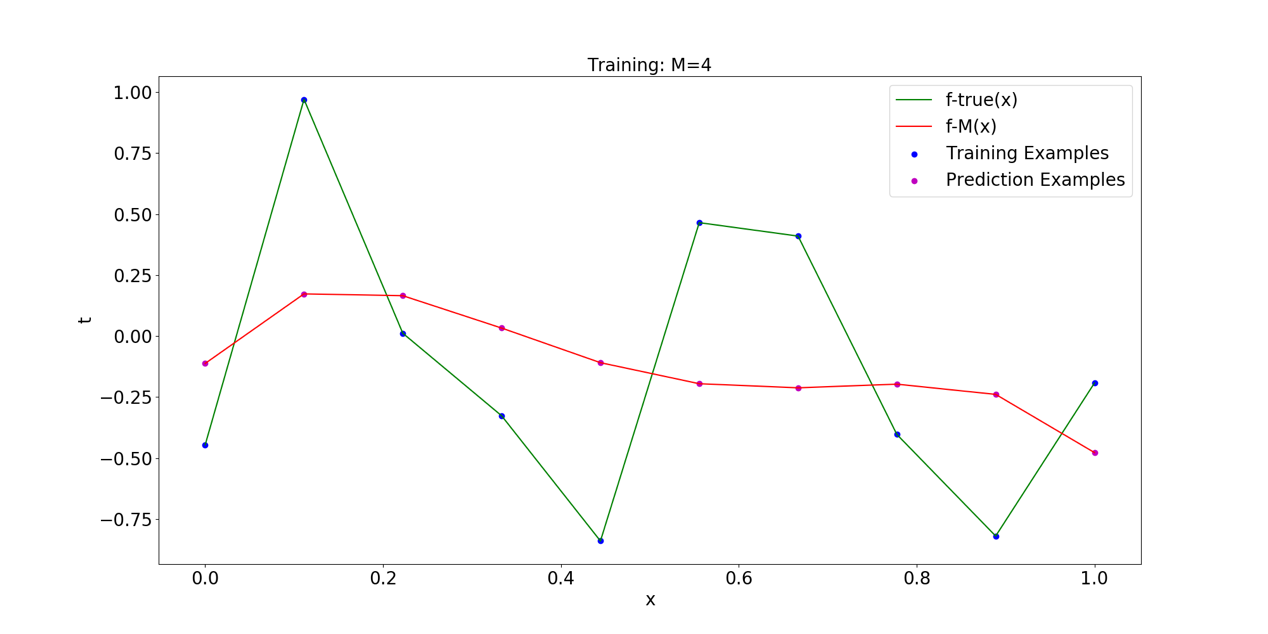


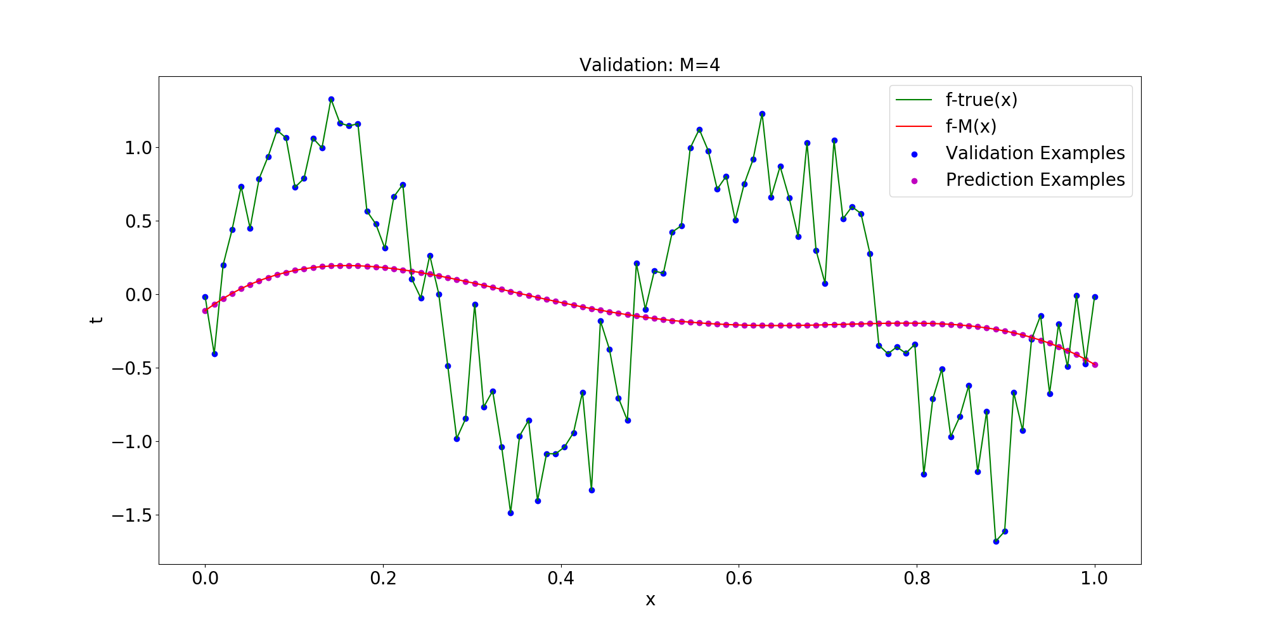
#### Fig.4: Fitting situation (M=3)

### 4.1.5 M = 4

when *M* = 4, the relevant records of the model are as follows

The errors of training set and verification set are very high, and the value of determination coefficient is very small, so the model is underfitting.



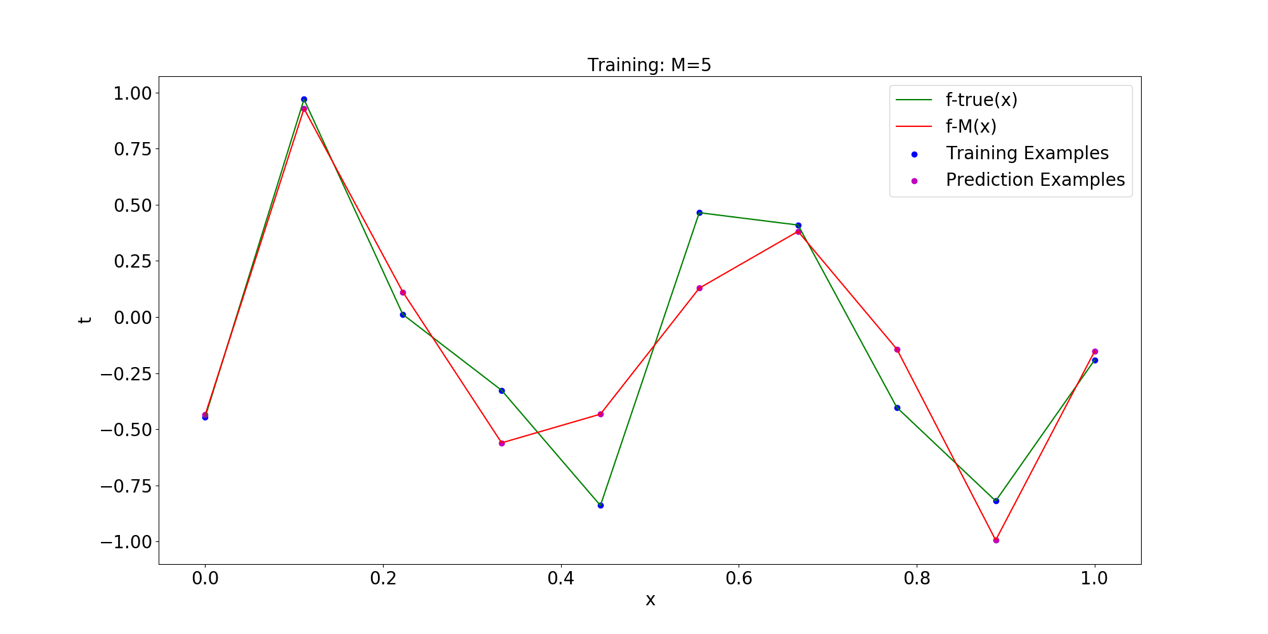


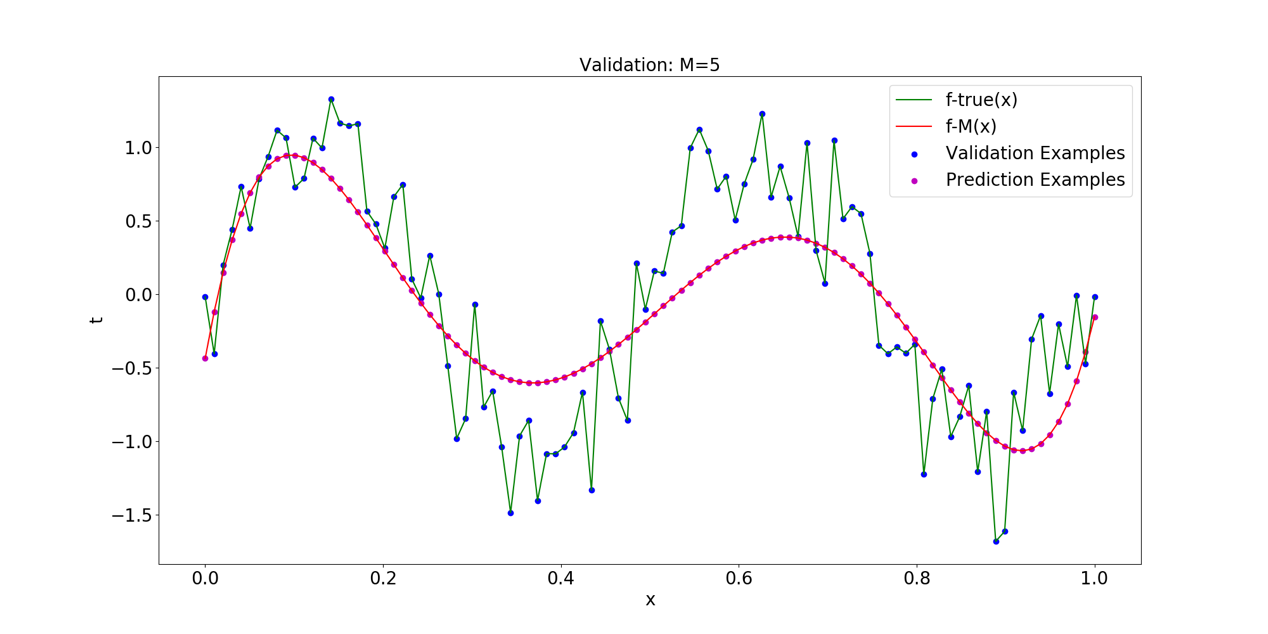
#### Fig.5: Fitting situation (M=4)

### 4.1.6 M = 5

when *M* = 5, the relevant records of the model are as follows

The errors of training set and verification set decrease, and the value of determination coefficient increases, so there is no underfitting in the model.



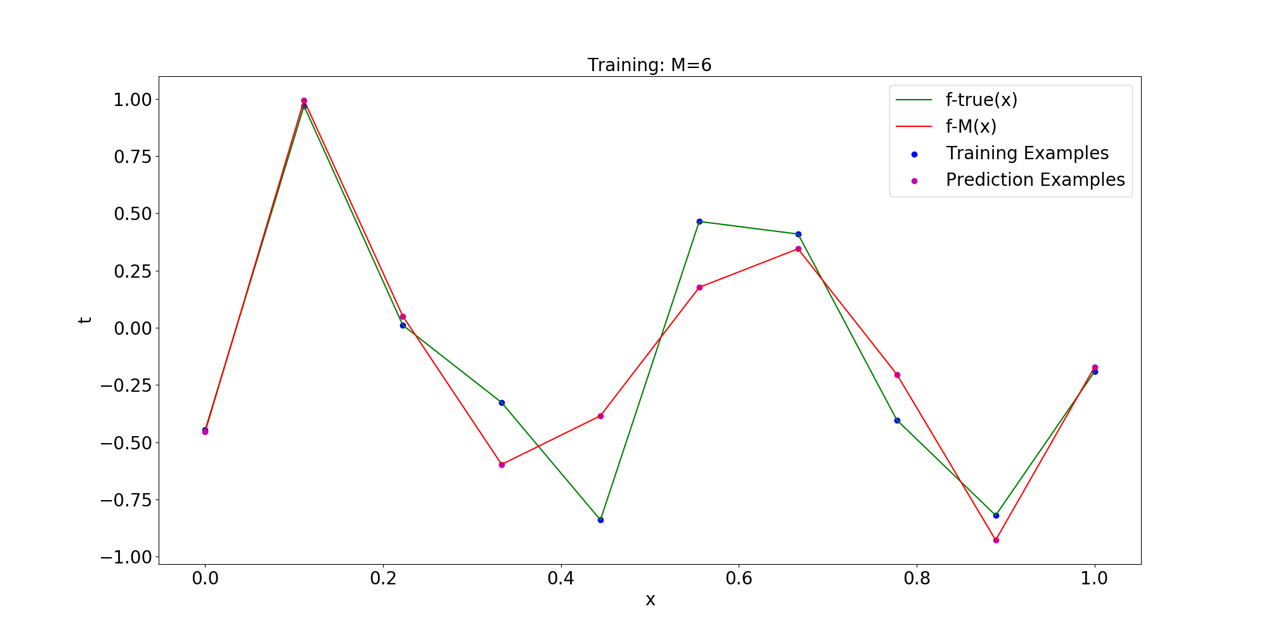


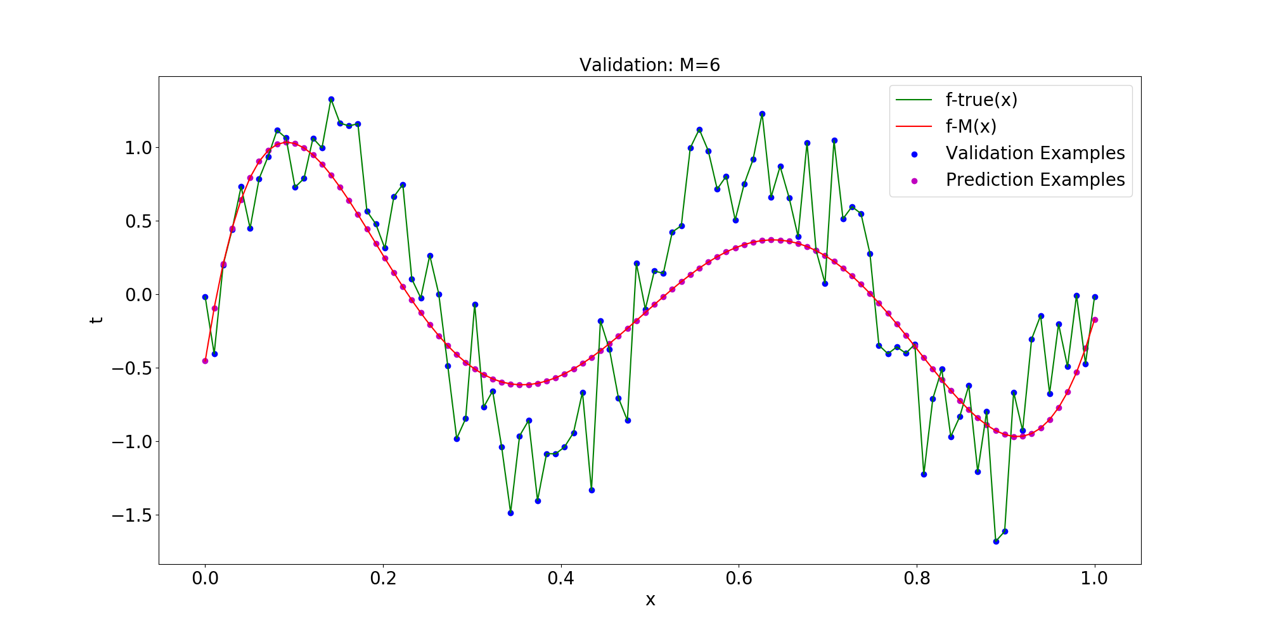
#### Fig.6: Fitting situation (M=5)

### 4.1.7 M = 6

when *M* = 6, the relevant records of the model are as follows

The errors of training set and verification set decrease, and the value of determination coefficient increases, so there is no underfitting in the model.



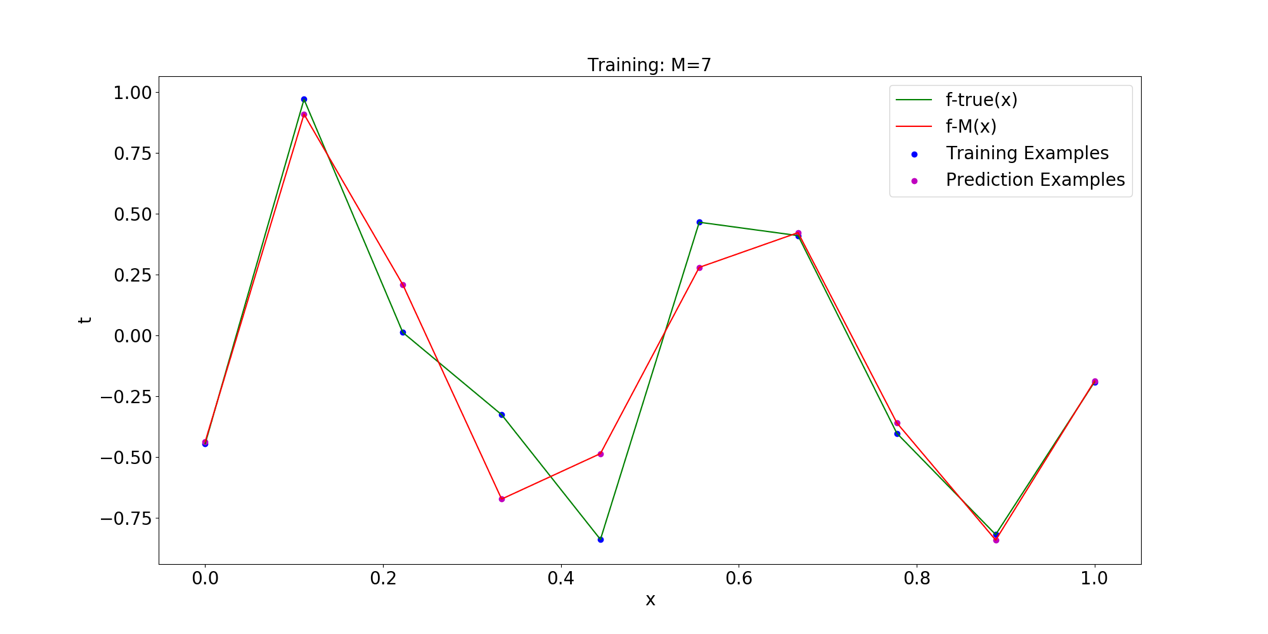


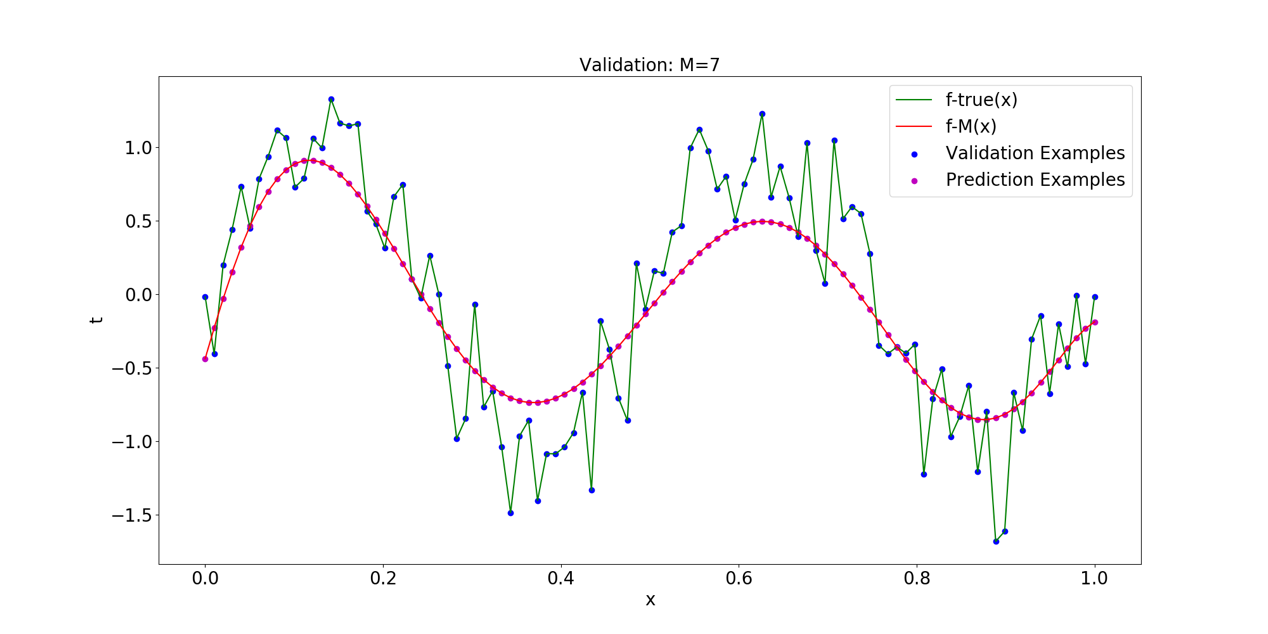
#### Fig.7: Fitting situation (M=6)

### 4.1.8 M = 7

when *M* = 7, the relevant records of the model are as follows

The errors of training set and verification set decrease, and the value of determination coefficient increases, so there is no underfitting in the model.



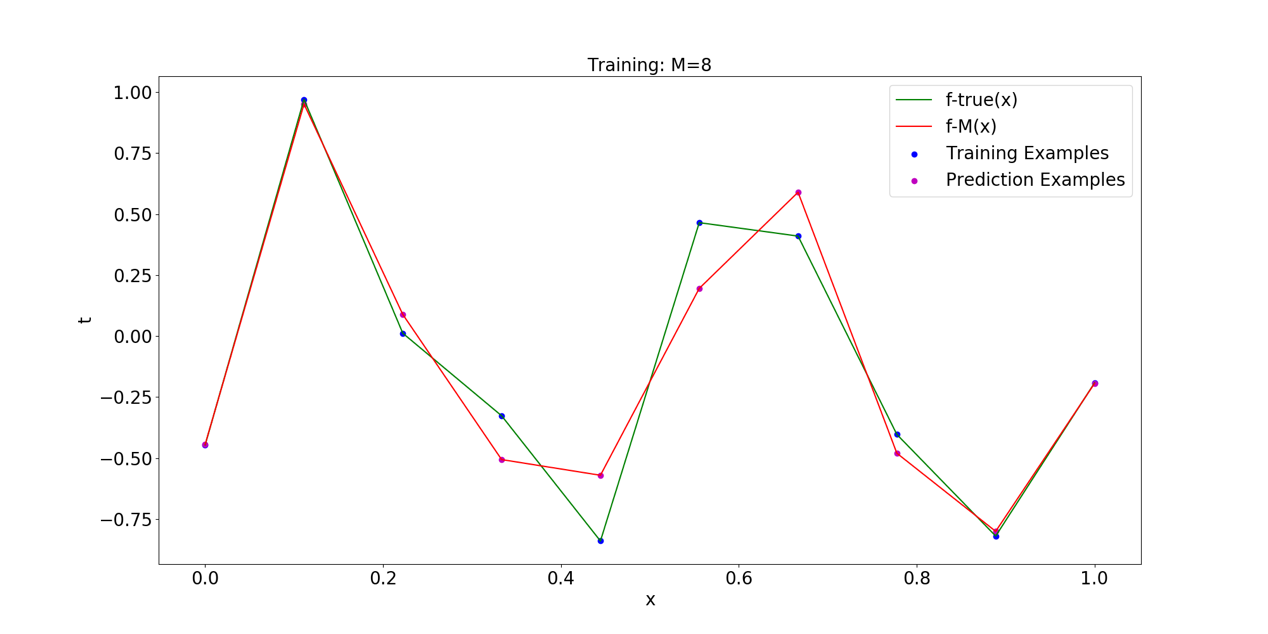


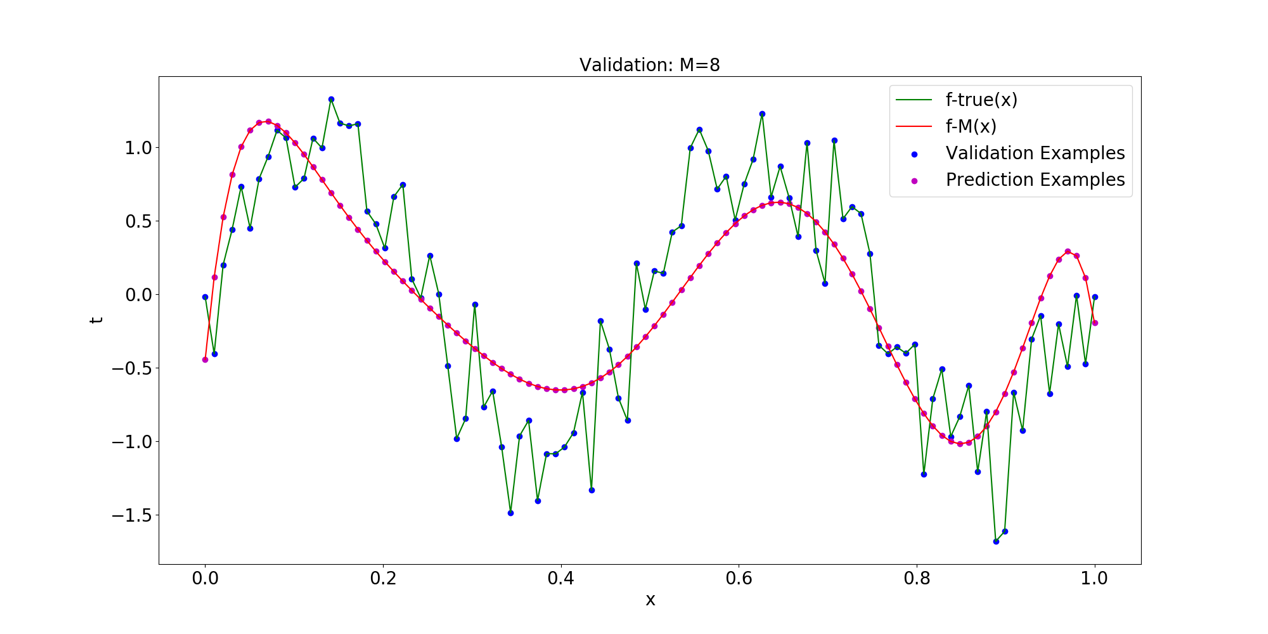
#### Fig.8: Fitting situation (M=7)

### 4.1.9 M = 8

when *M* = 8, the relevant records of the model are as follows

The error of training set decreases, but the error of verification set increases, the value of determination coefficient decreases and the polynomial coefficients become large, so the model began to deteriorate and there was a certain overfitting.





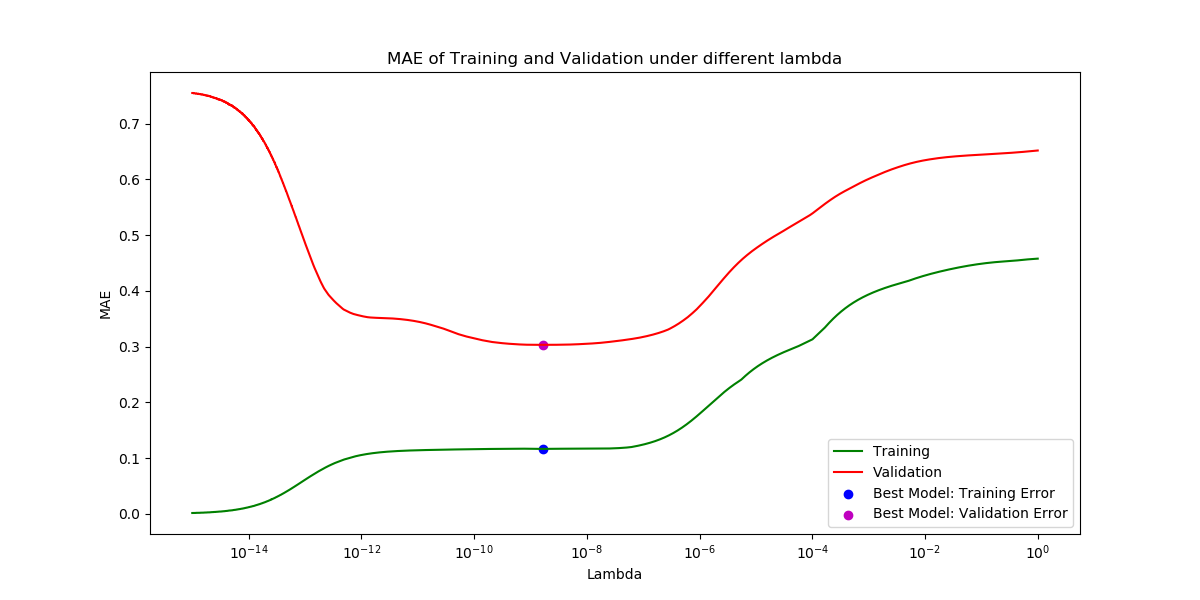
#### Fig.9: Fitting situation (M=8)

### 4.1.9 M = 9

When *M* = 8, the model is overfitting to a certain extent, and the parameter scale becomes large. Therefore, when *M* = 9, regularization needs to be added to control the overfitting.

In this experiment, L2-norm is used, that is, ridge regression is used to complete data fitting. In ridge regression, different penalty factors λ will have a certain impact on the fitting effect. The impact of λ value on training error and verification error is shown in the fig.10. The abscissa represents the penalty factor , the ordinate represents MAE, the green curve represents the real function value and the red curve represents the prediction function value.

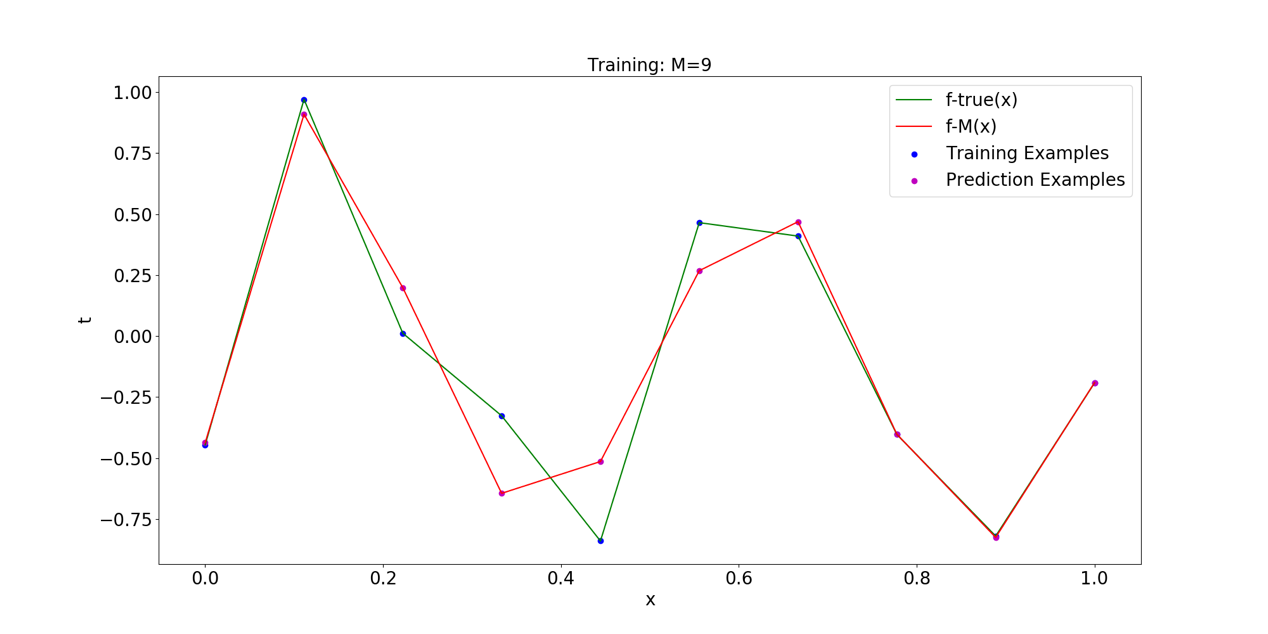
It can be seen from the figure that when , the error of training set and verification set is high, and the model is underfitting. When , the error of training set decreases and the error of verification set increases, so the model is overfitting. When , the effect of the model is the best. At this time, the error of the verification set is the smallest, as shown by the points marked in the fig.10.

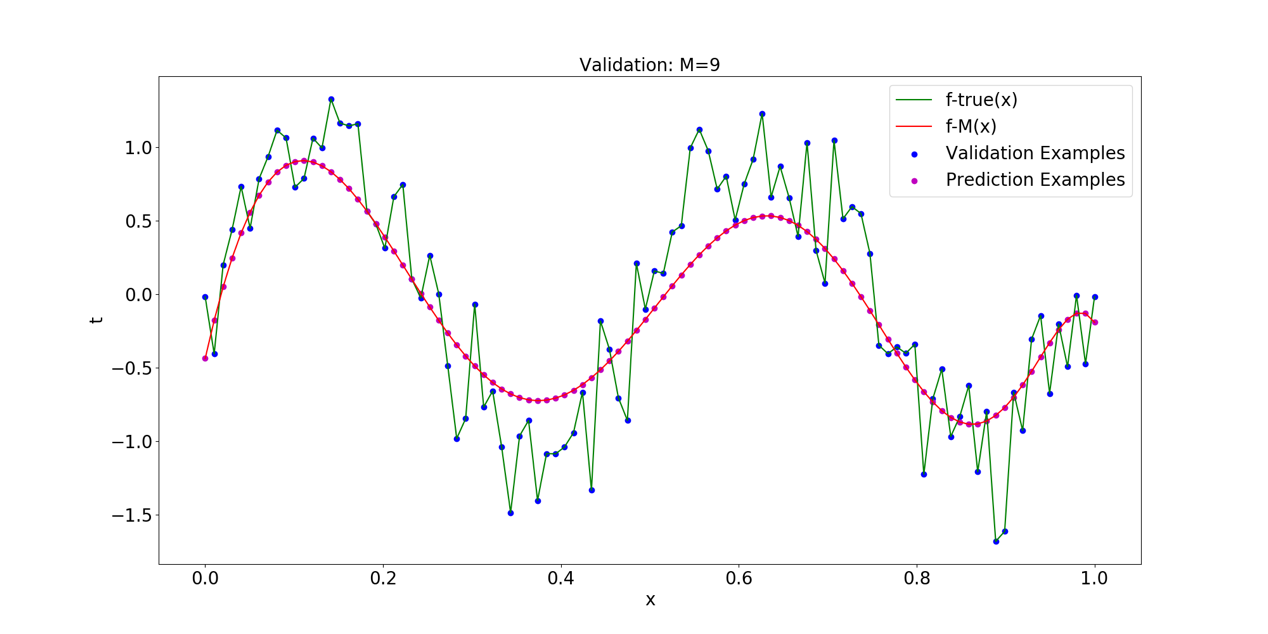


#### Fig.9: Influence of λ on Fitting (M=9)

Additionally, 3 different value of will be selected below to illustrate this phenomenon.

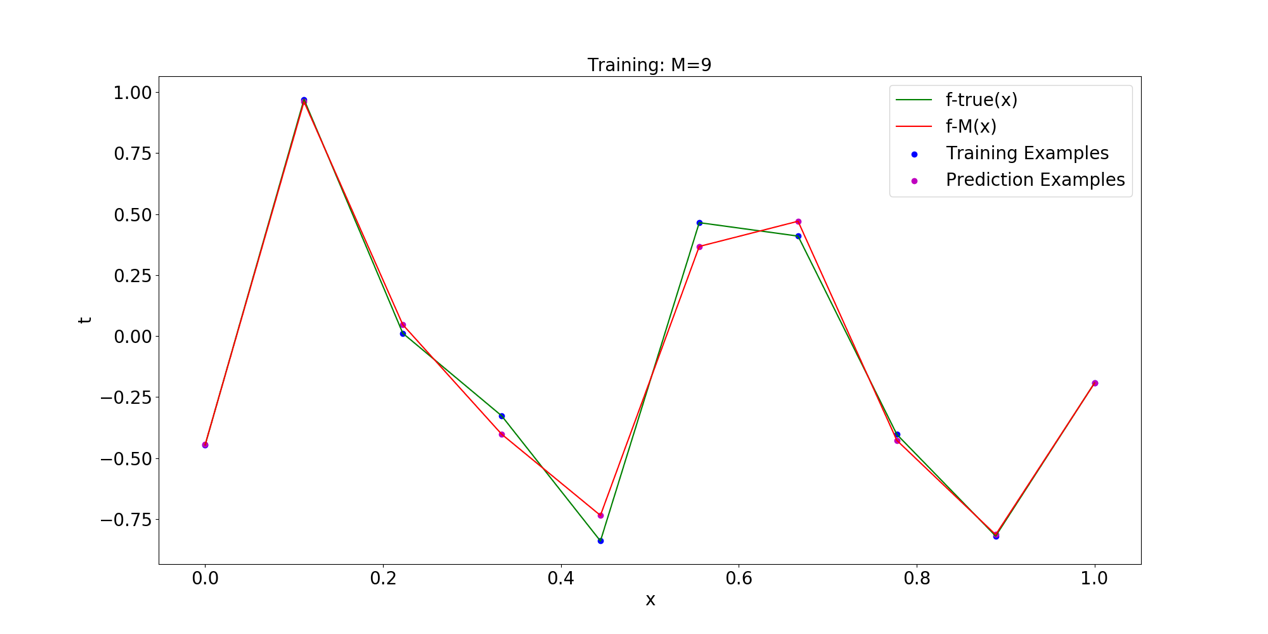
1. : The model fits well.

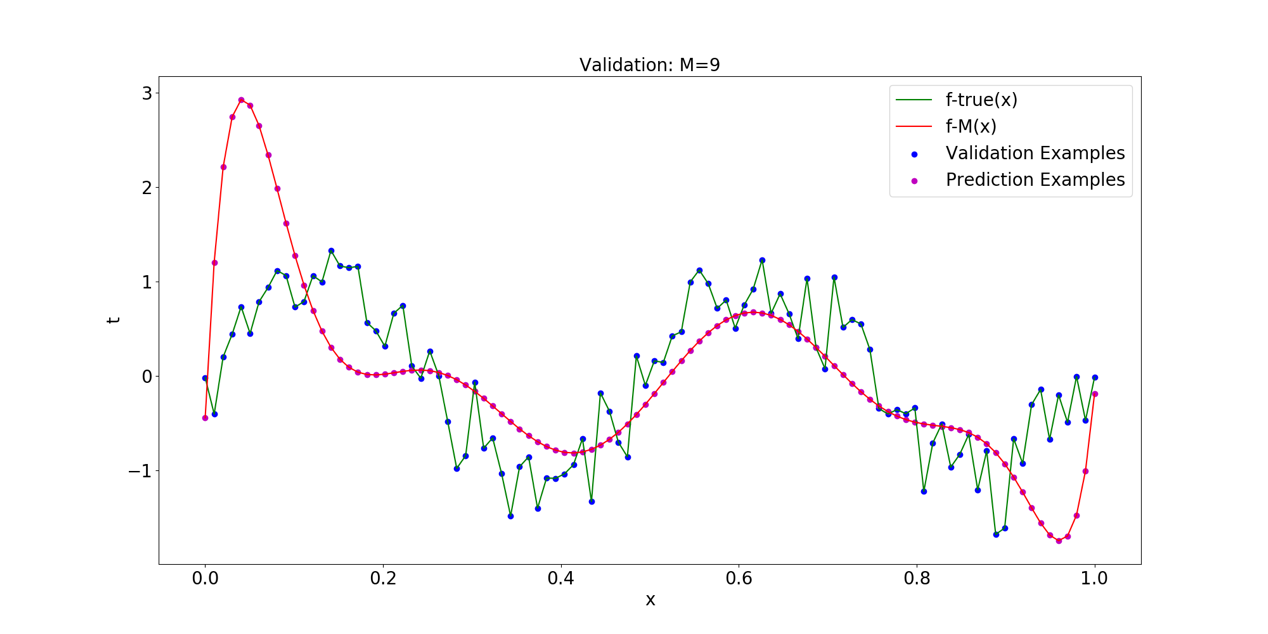




#### Fig.11: Fitting situation (M=9, )

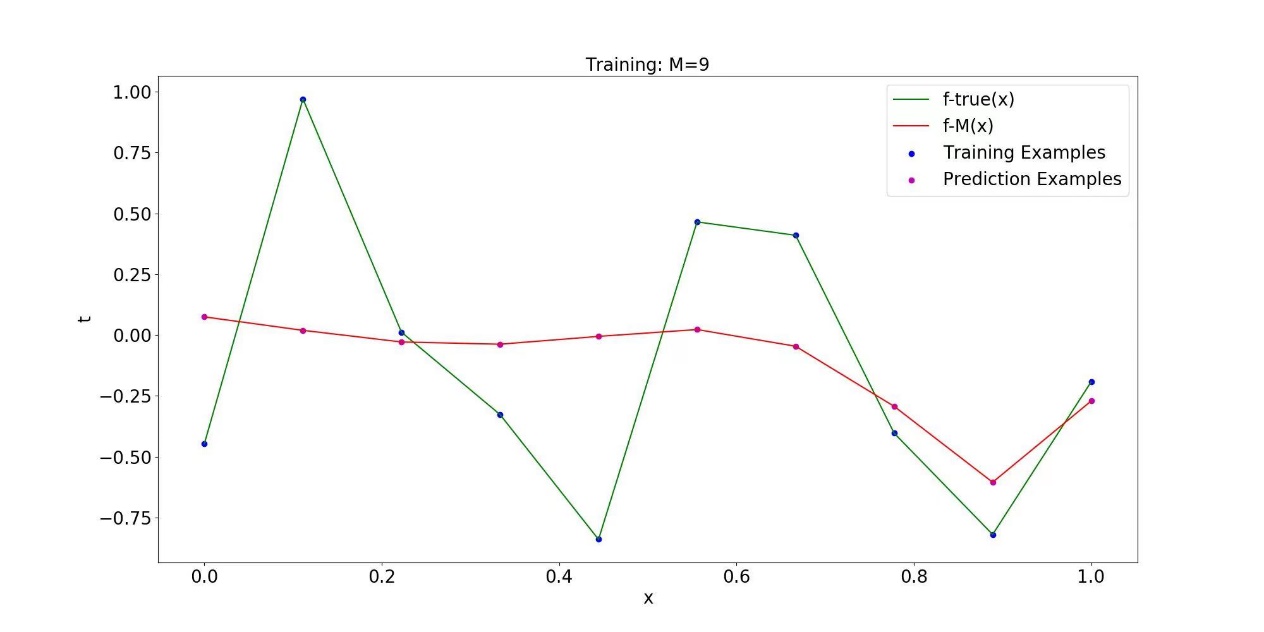
The error of training set is very low, but the error of verification set is high and the value of determination coefficient approaching 0. It can also be seen from the fig.12 that the prediction curve fits almost all the training data but the prediction curve fluctuates violently on the verification set, so the model is overfitting.

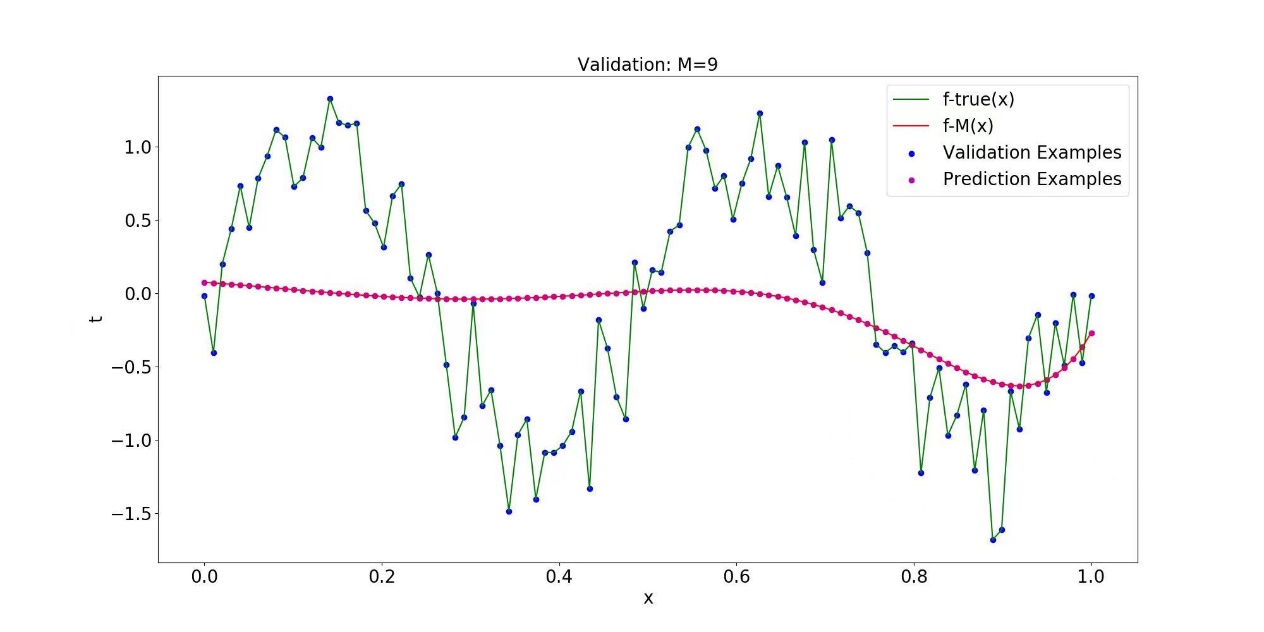




#### Fig.12: Fitting situation (M=9, )

The error of the training set and the verification set are both high and the value of determination coefficient approaching 0. It can also be seen from the fig.13 that the prediction curve fits not well both on the training data and the verification set, so the model is underfitting.



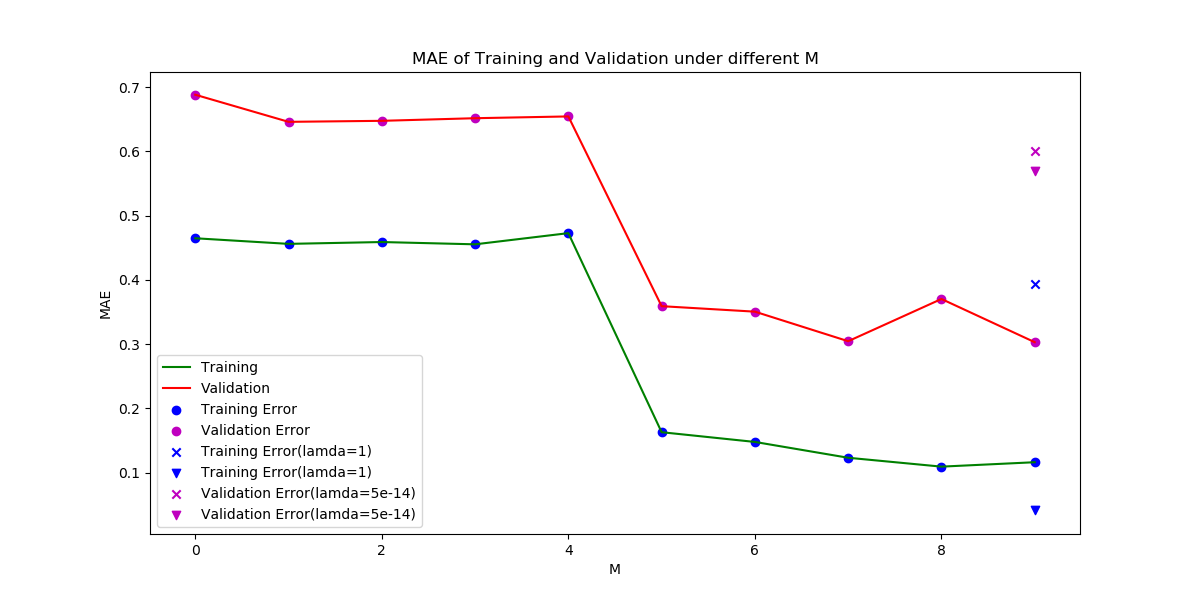


#### Fig.13: Fitting situation (M=9, )

## 4.2 Influence of M on Error

Figure 13 shows the influence of polynomial orders *M* with different values on training error and verification error.

When *M* is less than 4, the training error and verification error are both high, the model is underfitting. With the increase of M, the fitting effect of the model is better and better, but when the order M reaches 8, the model is overfitting. When m = 9, the overfitting phenomenon of the model can be effectively controlled by using ridge regression and applying an appropriate penalty factor.



#### Fig.13: Influence of M on Error

# Part 5 The Conclusion & Discussion

## 5.1 Conclusion

The summary of model evaluation indicators is shown in Table 4.

**Table 4.** Summary of model evaluation indicators

|  |  |  |  |
| --- | --- | --- | --- |
| *M* |  |  | *R-square* |
| 0 | 0.4649 | 0.6883 | -0.02 |
| 1 | 0.4561 | 0.6461 | 0.07 |
| 2 | 0.4590 | 0.6477 | 0.07 |
| 3 | 0.4555 | 0.6517 | 0.05 |
| 4 | 0.4729 | 0.6545 | 0.06 |
| 5 | 0.1630 | 0.3592 | 0.69 |
| 6 | 0.1479 | 0.3507 | 0.70 |
| 7 | 0.1233 | 0.3047 | 0.77 |
| 8 | 0.1095 | 0.3704 | 0.69 |
| 9 () | 0.1163 | 0.3031 | 0.77 |
| 9 () | 0.0415 | 0.5698 | 0.02 |
| 9 () | 0.3936 | 0.6009 | 0.17 |

1) In polynomial regression, the choice of polynomial order is very important. Small order is difficult to fit complex functions, which will lead to underfitting of the model, but too large order will lead to a sharp increase in model parameters. Make the model overfitting.

2) When the polynomial order is high, regularization technology can be used to control overfitting. At this time, the selection of penalty factors is very important. The smaller penalty factors have insufficient constraints on the model and cannot effectively control overfitting, while the larger penalty factors will produce great constraints, resulting in under fitting of the model.

## 5.2 Discussion

1) In the experiment, when the penalty factor is minimum or maximum, R-square appears negative value. It may be due to the poor data fitting effect.

2) When selecting the regularization technology, L2-norm is selected in this experiment. In the future work, L1-norm can be used and the effects of the two can be compared.

# Reference

[1] Bishop, Christopher M. Pattern Recognition and Machine Learning[M]. Springer, 2006.

[2] Meisner B N. Ridge Regression-Time Extrapolation Applied to Hawaiian Rainfall Normals[J]. Journal of Applied Meteorology, 2010, 18(7):904-912.

# Appendix

1. **import** numpy as np
2. **import** os
3. **import** matplotlib.pyplot as plt
4. **from** sklearn.metrics **import** r2\_score
5. **from** sklearn.linear\_model **import** Ridge
6. **from** sklearn.preprocessing **import** PolynomialFeatures
8. os.makedirs("Pics", exist\_ok=True)
9. os.makedirs("Pics/Pics\_M9/", exist\_ok=True)
11. # calculate MAE
12. **def** cal\_MAE(y, y\_pre):
13. error = abs((y-y\_pre)).sum()/y.shape[0]
14. **return** error
16. # generate data
17. **def** DataGenerate(train\_num, valid\_num, M):
18. x\_train = np.linspace(0., 1., train\_num)  # training set
19. x\_valid = np.linspace(0., 1., valid\_num)  # validation set
20. np.random.seed(640)
21. t\_train = np.sin(4 \* np.pi \* x\_train) + 0.3 \* np.random.randn(train\_num)
22. t\_valid = np.sin(4 \* np.pi \* x\_valid) + 0.3 \* np.random.randn(valid\_num)
23. **if** M == 9:
24. x\_train = x\_train.reshape(train\_num, 1)
25. x\_valid = x\_valid.reshape(valid\_num, 1)
26. t\_train = t\_train.reshape(train\_num, 1)
27. t\_valid = t\_valid.reshape(valid\_num, 1)
28. **return** x\_train, t\_train, x\_valid, t\_valid
30. # plot figure (fitting situation under different M)
31. **def** plot\_fig(x, y, prediction, Training\_or\_Validation, M, lamb):
32. plt.figure(figsize=(20, 10))
33. plt.scatter(x, y, color='b', label=Training\_or\_Validation + ' Examples')
34. plt.scatter(x, prediction, color='m', label='Prediction Examples')
35. plt.plot(x, y, color='g', label='f-true(x)')
36. plt.plot(x, prediction, 'r-', label='f-M(x)')
37. plt.title(Training\_or\_Validation + ': M=' + str(M), fontsize=20)
38. plt.xlabel('x', fontsize=20)
39. plt.xticks(fontsize=20)
40. plt.ylabel('t', fontsize=20)
41. plt.yticks(fontsize=20)
42. plt.legend(fontsize=20)
43. plt.savefig('./Pics/' + Training\_or\_Validation + str(M) + '\_' + str(lamb) + '.png')
44. plt.close()
46. **def** train(x\_train, t\_train, M, lamb):
47. **if** M == 9:
48. poly\_features\_d = PolynomialFeatures(degree=M, include\_bias=False)
49. x\_poly\_d = poly\_features\_d.fit\_transform(x\_train)
50. model = Ridge(alpha=lamb, solver="cholesky")
51. model.fit(x\_poly\_d, t\_train)
52. x\_plot\_poly = poly\_features\_d.fit\_transform(x\_train)
53. prediction = np.dot(x\_plot\_poly, model.coef\_.T) + model.intercept\_
54. f = 'Ridge'
55. **else**:
56. model = np.polyfit(x\_train, t\_train, M)
57. prediction = np.polyval(model, x\_train)
58. f = np.poly1d(model)
59. mean\_error = cal\_MAE(t\_train, prediction)
60. **return** model, mean\_error, prediction, f
62. **def** valid(x\_valid, t\_valid, model, M):
63. **if** M == 9:
64. poly\_features\_d = PolynomialFeatures(degree=M, include\_bias=False)
65. x\_plot\_poly = poly\_features\_d.fit\_transform(x\_valid)
66. prediction = np.dot(x\_plot\_poly, model.coef\_.T) + model.intercept\_
67. R\_square = r2\_score(t\_valid, prediction)
68. **else**:
69. prediction = np.polyval(model, x\_valid)
70. f = np.poly1d(model)
71. R\_square = r2\_score(t\_valid, f(x\_valid))
73. mean\_error = cal\_MAE(t\_valid, prediction)
74. **return** mean\_error, R\_square, prediction
76. # plot figure (MAE under different M)
77. **def** plot\_error\_fig(train\_error\_list, valid\_error\_list):
78. plt.figure(figsize=(12, 6))
79. x\_plot = list(range(0, 10))
80. plt.scatter(x\_plot, train\_error\_list[0:10], color='b', label='Training Error')
81. plt.scatter(x\_plot, valid\_error\_list[0:10], color='m', label='Validation Error')
82. plt.plot(x\_plot, train\_error\_list[0:10], color='g', label='Training')
83. plt.plot(x\_plot, valid\_error\_list[0:10], 'r-', label='Validation')
84. plt.scatter(9, train\_error\_list[10], color='b', marker='x', label='Training Error(lamda=1)')
85. plt.scatter(9, train\_error\_list[11], color='b', marker='v', label='Training Error(lamda=1)')
86. plt.scatter(9, valid\_error\_list[10], color='m', marker='x', label='Validation Error(lamda=5e-14)')
87. plt.scatter(9, valid\_error\_list[11], color='m', marker='v', label='Validation Error(lamda=5e-14)')
88. plt.title('MAE of Training and Validation under different M')
89. plt.xlabel('M')
90. plt.xticks()
91. plt.ylabel('MAE')
92. plt.yticks()
93. plt.legend()
94. plt.savefig('./Pics/MAE\_M.png')
95. plt.close()
97. # plot figure (learning curve under different lambda)
98. **def** plot\_curve\_lamb(M):
99. **if** M != 9:
100. **return**
101. **else**:
102. arr = (np.array(list(range(-15000, 1)))) / 1000
103. lamb\_plot = []
104. train\_error\_lamb = []
105. valid\_error\_lamb = []
106. R\_square\_lamb = []
107. **for** i **in** range(len(arr)):
108. lamb\_plot.append(10 \*\* arr[i])
109. **for** i **in** range(len(arr)):
110. lamb = lamb\_plot[i]
111. x\_train, t\_train, x\_valid, t\_valid = DataGenerate(train\_num, valid\_num, M)
112. model, train\_error, train\_prediction, f\_train = train(x\_train, t\_train, M, lamb)
113. valid\_error, R\_square, valid\_prediction = valid(x\_valid, t\_valid, model, M)
114. train\_error\_lamb.append(train\_error)
115. valid\_error\_lamb.append(valid\_error)
116. R\_square\_lamb.append(R\_square)
118. min\_index = valid\_error\_lamb.index(min(valid\_error\_lamb))
119. **print**('The best model | Lambda = ', lamb\_plot[min\_index], '| Training error = ', train\_error\_lamb[min\_index],
120. '| Validation error = ', valid\_error\_lamb[min\_index], '| R-square: ', R\_square\_lamb[min\_index])
122. plt.figure(figsize=(12, 6))
123. plt.scatter(lamb\_plot[min\_index], train\_error\_lamb[min\_index], color='b', label='Best Model: Training Error')
124. plt.scatter(lamb\_plot[min\_index], valid\_error\_lamb[min\_index], color='m', label='Best Model: Validation Error')
125. plt.plot(lamb\_plot, train\_error\_lamb, color='g', label='Training')
127. plt.plot(lamb\_plot, valid\_error\_lamb, 'r-', label='Validation')
128. plt.semilogx()
129. plt.title('MAE of Training and Validation under different lambda')
130. plt.xlabel('Lambda')
131. plt.xticks()
132. plt.ylabel('MAE')
133. plt.yticks()
134. plt.legend()
135. plt.savefig('./Pics/Pics\_M9/MAE\_lambda.png')
136. plt.close()
138. lamb\_list = [1.67e-9, 1e-3, 5e-14]  # well-fitting, underfitting, overfitting
139. train\_num = 10     # numbers of train samples
140. valid\_num = 100    # numbers of valid samples
141. train\_error\_list = []
142. valid\_error\_list = []
143. R\_square\_list = []
144. **for** i **in** range(12):
145. **if** i < 9:
146. M = i
147. lamb = None
148. **print**('M = ', M)
149. **else**:
150. M = 9
151. lamb = lamb\_list[i-9]
152. x\_train, t\_train, x\_valid, t\_valid = DataGenerate(train\_num, valid\_num, M)
153. model, train\_error, train\_prediction, f\_train = train(x\_train, t\_train, M, lamb)
154. plot\_fig(x\_train, t\_train, train\_prediction, 'Training', M, lamb)
155. valid\_error, R\_square, valid\_prediction = valid(x\_valid, t\_valid, model, M)
156. plot\_fig(x\_valid, t\_valid, valid\_prediction, 'Validation', M, lamb)
158. **print**('f' + str(M) + '(x) = ')
159. **print**(f\_train)
160. **print**('Training Error:', train\_error, '| Validation Error:', valid\_error)
161. **print**('R\_square: {:.2f}'.format(R\_square))
162. **print**('----------------------------------------------------------------------------------------------------')
163. train\_error\_list.append(train\_error)
164. valid\_error\_list.append(valid\_error)
165. R\_square\_list.append(R\_square)
167. plot\_error\_fig(train\_error\_list, valid\_error\_list)
168. plot\_curve\_lamb(9)