数学建模

数据生成

高斯分布函数定义：

```
def Gauss(x, a, b):
    return 1 / (b * np.sqrt(2 * math.pi)) * \
           np.exp(-((x - a) ** 2) / (2 * b ** 2))
```

生成x，y和噪音数据

```
N = 1000         # 超参数，点个数
h = 0.0002       # 超参数，应该是噪音方差
np.random.seed(1)  # 便于复现
x = np.linspace(0, 4, N)
y_withoutNoise = Gauss(x, 0, 1) + Gauss(x, 1.5, 1)
y = y_withoutNoise + h * np.random.normal(size=y_withoutNoise.shape)
```

打乱数据，方便分开不同的数据集

```
index = [i for i in range(len(y))]
np.random.shuffle(index)
x_shuffle = x[index]
y_shuffle = y[index]
```

定义训练集、测试集、验证集

```
train_rate = 0.8
validate_rate = 0.1
test_rate = 0.1
train_size = int(len(x_shuffle) * train_rate)
validate_size = int(len(x_shuffle) * validate_rate)
test_size = int(len(x_shuffle) * test_rate)
x_train = x_shuffle[0: train_size]
y_train = y_shuffle[0: train_size]
x_validate = x_shuffle[train_size: train_size + validate_size]
y_validate = y_shuffle[train_size: train_size + validate_size]
x_test = x_shuffle[train_size + validate_size: train_size + validate_size +
test_size]
y_test = y_shuffle[train_size + validate_size: train_size + validate_size +
test_size]
```

模型定义

定义RMSE函数

```
def RMSE(y_pred, y_label):
    loss = 0.0
    for i in range(len(y_label)):
        loss += (y_label[i] - y_pred[i])**2
    return np.sqrt(loss / len(y_label))
```

第一问

$$y = w_1 G(x; a_1, b_1) + w_2 G(x; a_2, b_2)$$

其中 w_i, a_i, b_i 为参数

```
def my_func(x, w1, a1, b1, w2, a2, b2):
    return w1 * Gauss(x, a1, b1) + w2 * Gauss(x, a2, b2)
```

对数据集进行拟合

```
popt = curve_fit(my_func, x_train, y_train, maxfev=500000)[0]
```

这里使用初始参数均为1（默认）

第二问

$$y = \sum_{i=0}^{K} a_i x^i$$

其中 $\{a_i\}_{i=0}^{K}$为参数

```
K = 5 # 超参数，表示参数a的个数
def my_func(x, *arguments):
    k = len(arguments)
    y_pred = np.zeros(x.shape)
    for i in range(k):
        y_pred += arguments[i] * np.power(x, i)
    return y_pred
```

对数据集进行拟合

```
popt = curve_fit(my_func, x_train, y_train, maxfev=500000,
p0=np.random.normal(size=K))[0]
```

这里使用随机数参数

第三问

$$y = \sum_{i=1}^{K} \sum_{j=0}^{M} w_{ij} x^j G(x; a_i, b_i)$$

其中 $\{w_{ij}, a_i, b_i\}$ 为参数

```
K = 10
M = 10
def my_func(x, *arguments):
    global K
    global M

    y_pred = np.zeros(x.shape)
    for i in range(K):
        for j in range(M):
            y_pred += arguments[i * M + j] * np.power(x, j) * \
                      Gauss(x, arguments[K * M + i], arguments[K * M + M + i])
    return y_pred
```

对数据集进行拟合

```
popt = curve_fit(my_func, x_train, y_train, maxfev=500000,
                 p0=np.random.normal(size=K * M + 2 * K))[0]
```

第四问

$$y = a\cos(bx) + c$$

其中 $\{a, b, c\}$ 为参数

```
def my_func(x, *arguments):
    a = arguments[0]
    b = arguments[1]
    c = arguments[2]
    y_pred = a * np.cos(b * x) + c
    return y_pred
```

对数据集进行拟合

```
popt = curve_fit(my_func, x_train, y_train, maxfev=500000, p0=[4, 1000000.0, 1])[0] # b 很大时的拟合
popt = curve_fit(my_func, x_train, y_train, maxfev=500000, p0=np.random.normal(size=3))[0]
```

结果

第一问

```
参数为[ 1.24797823  1.33574198  1.04082177  0.67541904 -0.08339295  0.85500209]
训练集上的RMSE = 0.013684885961983001
验证集上的RMSE = 0.014518364354273634
测试集上的RMSE = 0.014672473057087694
```

第二问

```
参数为[ 5.21187951e-01  2.77633834e-01 -4.45090178e-01  6.43147464e-01
 -7.48625940e-01  5.18173890e-01 -2.18656726e-01  5.54910163e-02
 -7.72527833e-03  4.50606290e-04]
训练集上的RMSE = 0.013662379374811604
验证集上的RMSE = 0.014551122594372872
测试集上的RMSE = 0.014673037209585947
```

第三问

```
参数为[ 4.29349052e+01  4.76624590e+01  5.55340387e+01  5.85743390e+01
  4.51179924e+01  2.70910797e+01  5.95881286e+00 -2.31271516e+00
 -3.32081450e+00  2.30643127e+00 -5.76504071e+01 -3.31801179e-02
  3.42628006e+01  2.34951253e+01  4.02333323e+00 -5.88245992e+00
 -6.21657073e+00 -2.51482707e+00 -7.67937020e-01  7.09569259e-01
 -1.57717036e+02 -2.29961640e+02 -1.04934438e+02 -2.86765341e+01
 -3.35451515e+00  2.59050417e+00 -4.21143143e-02  1.32260322e+00
  1.83799983e+00 -2.72699569e-01  7.26286987e+01 -5.65267967e+00
 -5.58988123e+02  6.64269030e+02 -5.87891868e+02  6.54107201e+02
  1.11191383e+03  1.34304595e+03  1.42917651e+03  1.17750148e+03
 -8.90610064e-01  6.79909376e-01 -3.02979686e-01 -8.27707124e-01
  4.49126585e-01 -2.51826321e-01  4.97377399e-01 -2.17327382e-03
 -1.55771910e+00 -9.70182124e-01 -1.80342495e+01  8.83534221e+01
  1.50946811e+02  1.10731534e+02  3.99921764e+01 -5.44064360e+00
 -1.65051913e+01 -1.03693088e+01 -4.54289191e+00 -2.26775034e+00
  1.40197201e+02  3.74259816e+02  2.03160737e+02  6.00293015e+01
  1.09459665e+01 -7.76816826e-01 -1.74096481e+00 -3.35458325e-01
 -1.06108636e+00  8.70339653e-01 -9.39545354e+01 -1.24617259e+02
 -8.00695994e+01 -3.95034115e+01 -1.62527678e+01 -4.24702865e+00
 -7.54430982e-01  7.55219497e-01  3.93163043e-01 -8.48297539e-01
  3.05362017e+01  1.58661062e+02  1.43341916e+02  9.19707021e+01
  4.97093334e+01  2.22420021e+01  4.95099864e+00  5.99398077e-01
  1.78827638e+00 -2.35672696e+00  2.46663240e+01  2.06198209e+02
  3.45467637e+02 -2.09973903e+02  1.52436275e+03  9.18768463e+02
 -3.54592914e+03  7.30951080e+03 -5.35819450e+03  2.21894809e+03
  1.22437404e+00  1.46688236e+00  3.23466392e-01 -2.64944273e-01
 -1.24773729e+00  7.69201198e-01 -1.48242229e+00  1.26499893e+00
  4.29286136e-02  1.57310573e-01  5.08172890e-01  9.24803046e-01
  2.06402102e+00  5.04069263e-01  4.42714146e-02 -8.28960976e-01
 -2.27183175e+00  1.21633269e+00 -1.47266182e+00  3.36039231e-01]
训练集上的RMSE = 0.013774398241012075
验证集上的RMSE = 0.015083139266189906
测试集上的RMSE = 0.015209934967291176
```

第四问

使用随机参数

```
参数为[ 0.37334422 -0.60487487  0.25474133]
训练集上的RMSE = 0.03645865116731783
验证集上的RMSE = 0.03510180262072297
测试集上的RMSE = 0.033907799361535707
```

当b很大（10000）时

```
参数为[5.63498618e-03 9.99999681e+05 3.55315070e-01]
训练集上的RMSE = 0.21508399948937978
验证集上的RMSE = 0.21120644724800114
测试集上的RMSE = 0.22239055521813747
```

画图

设置中文并定义画布

```
# 设置显示中文
plt.rcParams['font.sans-serif'] = ['SimHei']  # 指定默认字体
plt.rcParams['axes.unicode_minus'] = False  # 正常显示负号
plt.figure(figsize=(12, 6), dpi=100)
```

根据参数求解训练集和测试集上的结果

```
print("参数为" + str(popt))
y_train_pred = my_func(x_train, *popt)
print("训练集上的RMSE = " + str(RMSE(y_train_pred, y_train)))
y_test_pred = my_func(x_test, *popt)
print("测试集上的RMSE = " + str(RMSE(y_test_pred, y_test)))
```

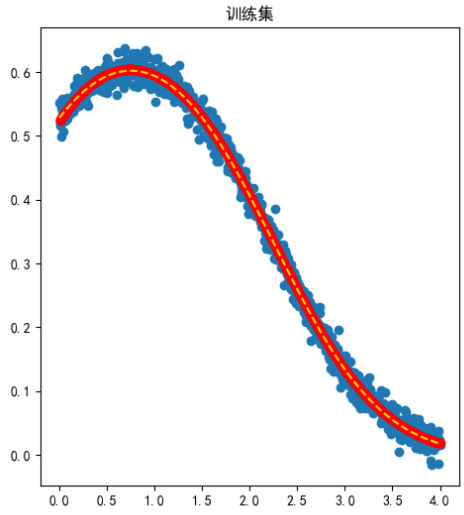
画训练集上的图

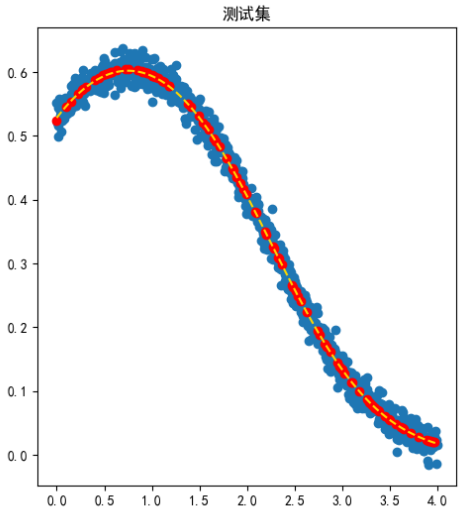
```
plt.subplot(1, 2, 1)
plt.plot(x, y_withoutNoise, linestyle='--', c='gold')
plt.scatter(x, y)
plt.scatter(x_train, y_train_pred, c='r')
plt.title("训练集")
```

画测试集上的图

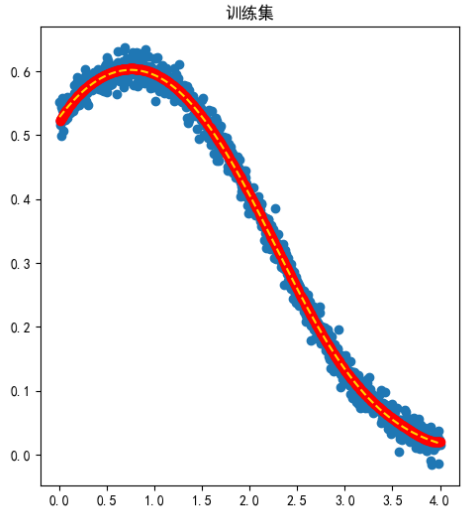
```
plt.subplot(1, 2, 2)
plt.plot(x, y_withoutNoise, linestyle='--', c='gold')
plt.scatter(x, y)
plt.scatter(x_test, y_test_pred, c='r')
plt.title("测试集")
```

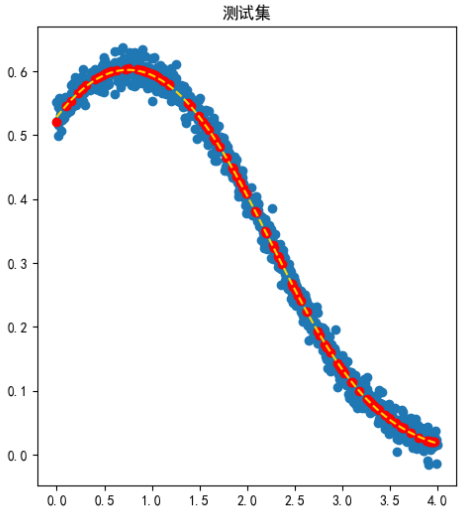
第一问


训练集
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0

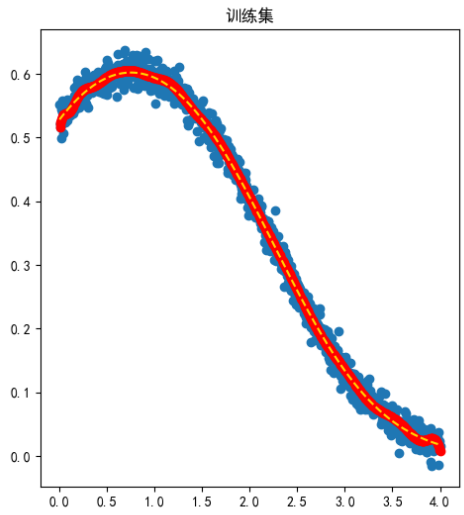

测试集
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0

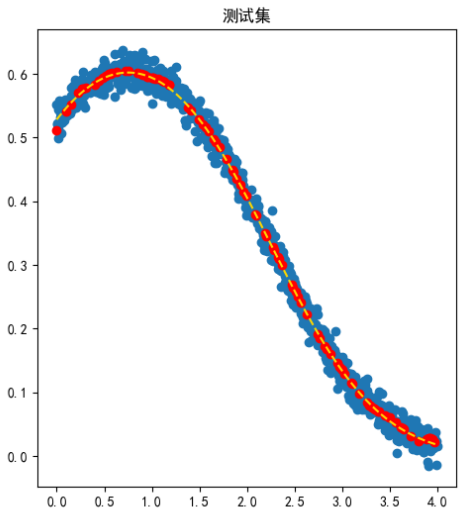
第二问


训练集
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0


测试集
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0

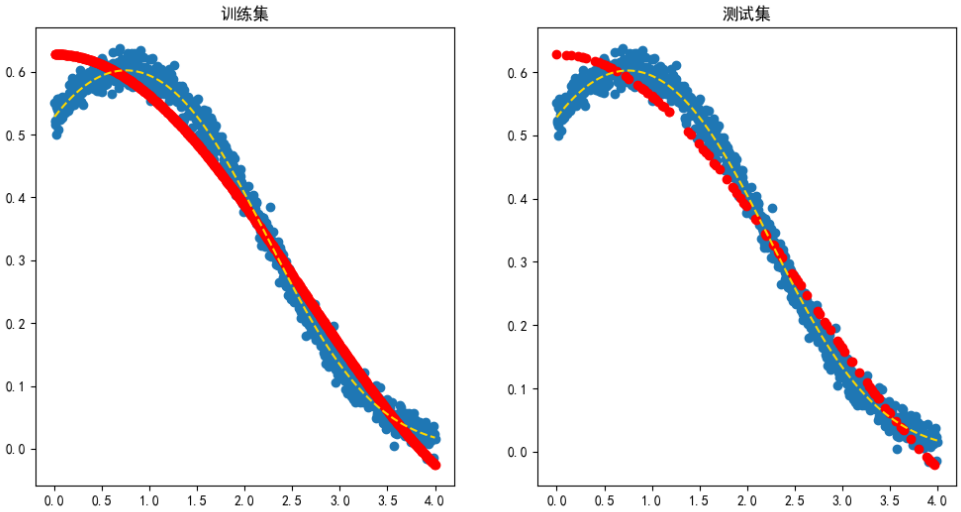
第三问


训练集
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0

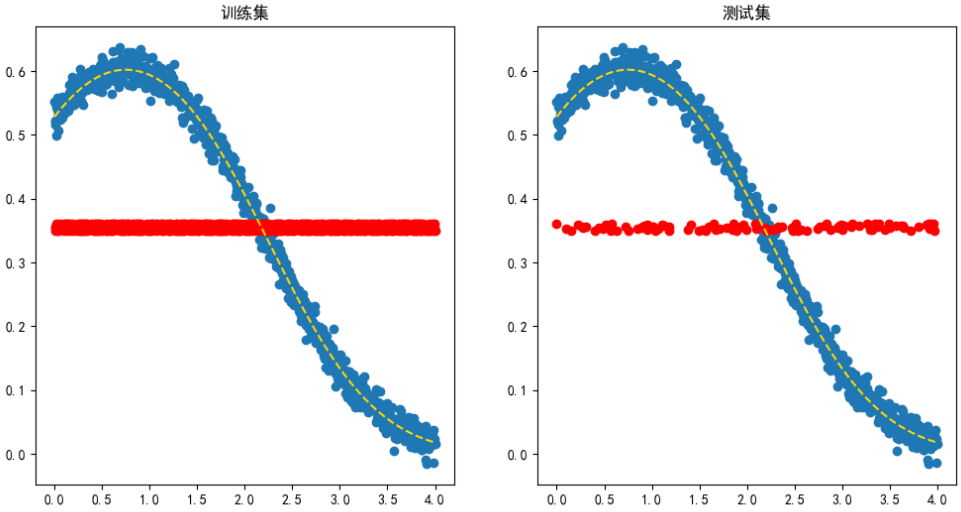

测试集
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0

第四问

使用随机参数



b很大时

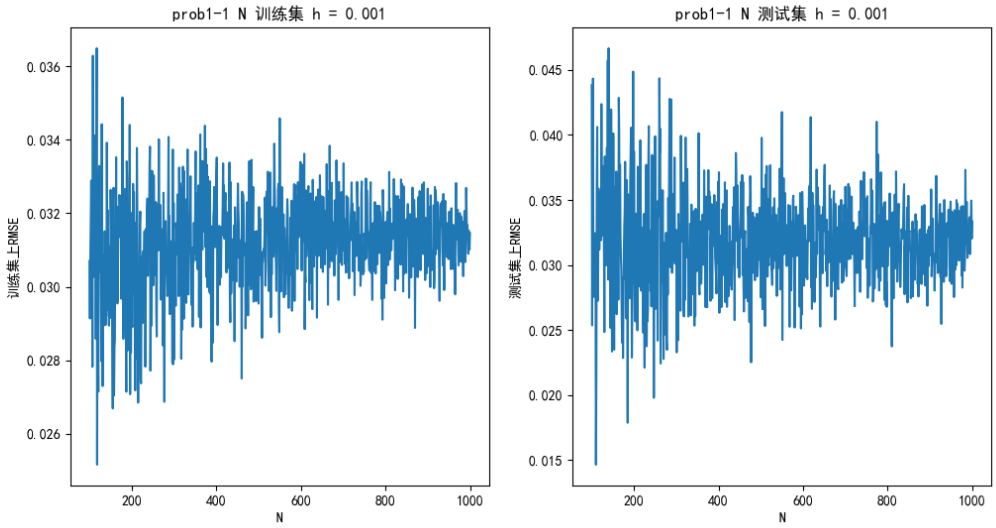


测试（初始值/N/h）

第一问

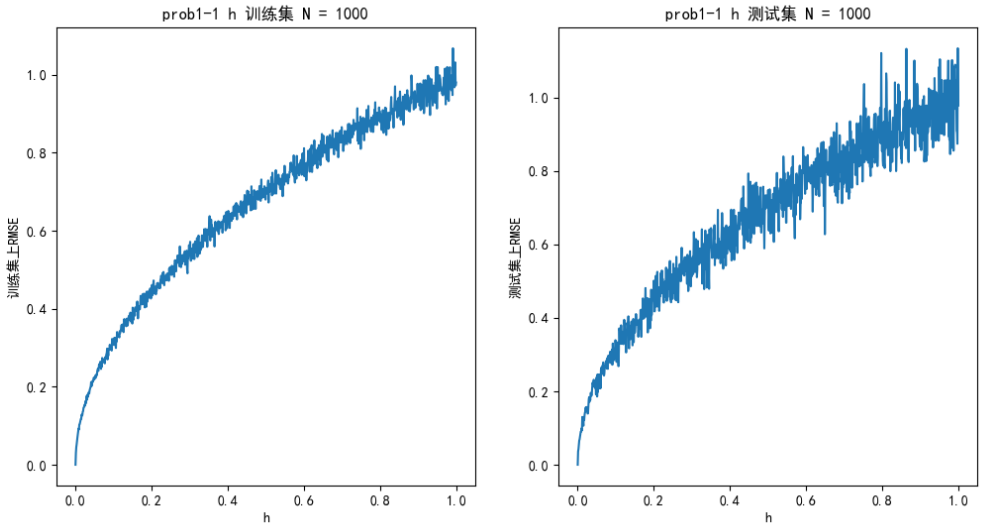
如图：固定h为0.001，探究N对于训练误差的影响

可以看到随着N增加，RMSE的突变值逐渐减少，整体趋向于平稳，说明数据量虽然不能影响整体的RMSE，但是可以降低方差

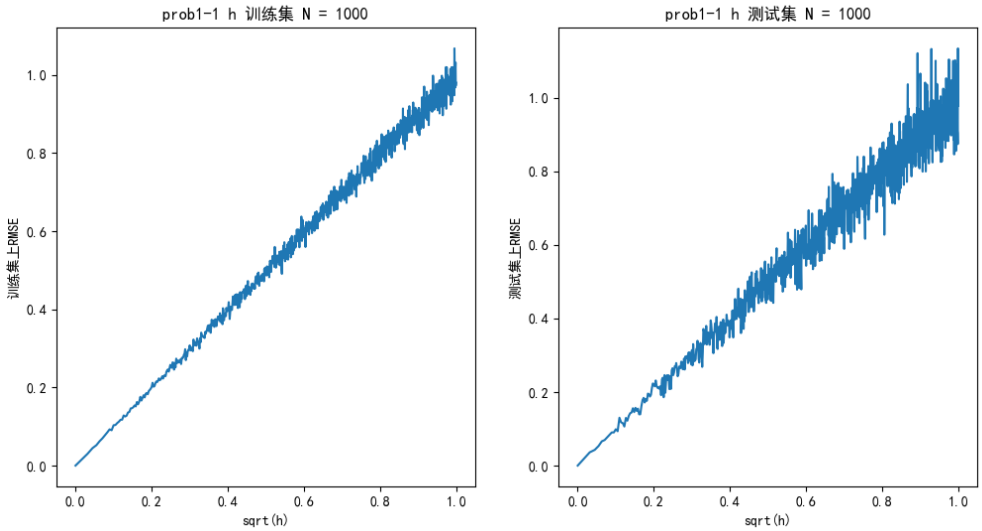


如图：固定N为1000，探究h对于RMSE的影响

可以看到，随着h的增加训练集和测试集上误差均增加，其中可以看到整体的弧度呈现出特殊趋势，因此猜想RMSE的增加和 $\sqrt{h}$ 有关



如图：同样的条件，画出RMSE - 根号h的图



可以看到几乎成正比，因此猜想正确

第二问

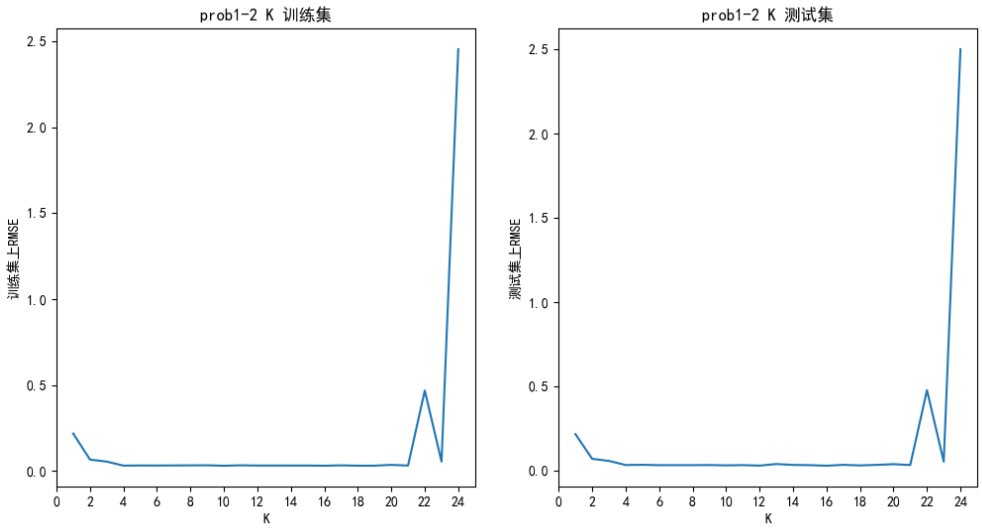
```
K = 10
参数为[ 5.28520749e-01  1.90662237e-01 -8.98229287e-02 -1.12805698e-01
  1.59241842e-01 -1.25485486e-01  5.54355584e-02 -1.32369940e-02
  1.61080935e-03 -7.87376717e-05]
训练集上的RMSE = 0.00019536470477344537
```

```
K = 12
参数为[ 5.28336742e-01  1.96252407e-01 -1.30812188e-01  1.72582598e-02
 -6.30264922e-02  1.02464918e-01 -9.30061874e-02  4.96415632e-02
 -1.56598309e-02  2.88775415e-03 -2.89539725e-04  1.22568993e-05]
训练集上的RMSE = 0.00019324850718960567
```

```
K = 15
参数为[ 5.28431666e-01  1.92678055e-01 -9.79093156e-02 -1.12718435e-01
  2.06617619e-01 -2.14293056e-01  1.09467172e-01  9.27156187e-03
 -5.64910329e-02  4.30502886e-02 -1.79958706e-02  4.64587009e-03
 -7.38369289e-04  6.64626031e-05 -2.60033635e-06]
训练集上的RMSE = 0.00019349942406152312
```

根据如上几次零散测试发现当K为10左右整体预测比较好

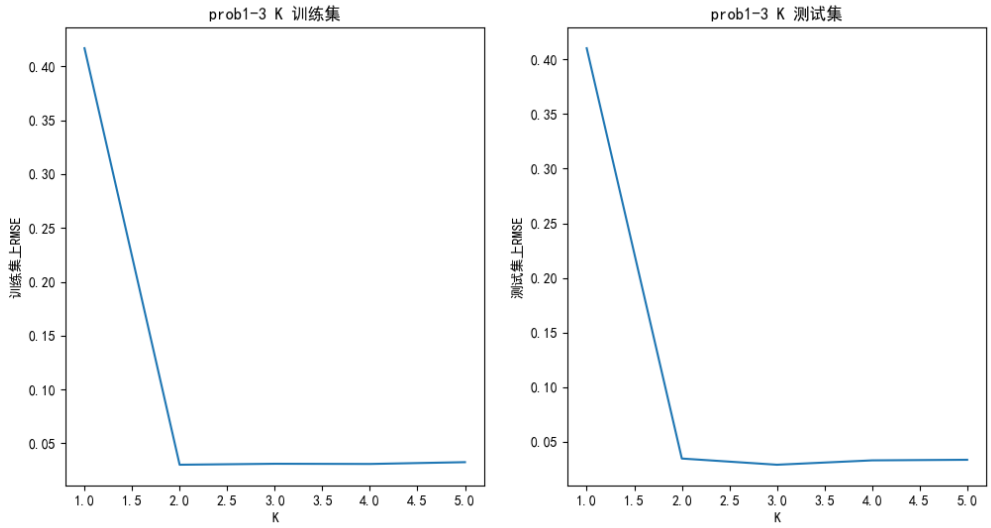
如图：画出RMSE随K变化的图



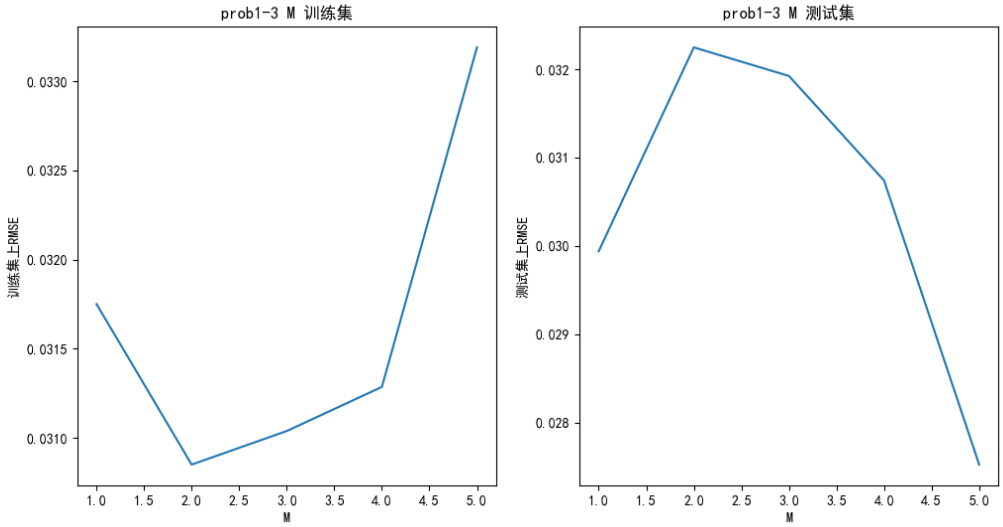
可以看出在很长的一段范围内RMSE很好

第三问

如图：固定M=3，画出RMSE随着K变化的曲线



如图：固定K=3，画出RMSE随着M变化的曲线



综上可以看到：随着K和M的增加，在测试集上RMSE的误差均减少，在训练集上的误差普遍减小（虽有升高，但是普遍很低，可以认为是误差）

最优模型

综上可以看到模型一二拟合最好，模型三可能由于噪音较大，因此拟合效果反不如模型二

最终较优的RMSE为0.014左右

所有源码均在压缩包中（展开则太大

prob1_x.py为拟合曲线

prob1_x_xxx.py为探究超参数与RMSE关系