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[ 1.00648771  1.49594294  1.00087372  0.99206881 -0.00342384  0.99667574]
训练集上的RMSE = 0.00019352913291170577
```

第二问

```
参数为[ 0.5179899   0.25808409 -0.20057979  0.01669909  0.00244388]
训练集上的RMSE = 0.005030538218058056
```

第三问

```
参数为[ 5.26179041e+00  8.40349247e+00 -5.95401649e+00  4.95768451e-02
  1.76014097e+00  2.78530770e+00 -9.24994223e-01  2.59799173e-01
 -5.58182582e-01  4.22387307e-01  1.35376618e+02  1.59727436e+02
  1.03139610e+02  4.46082635e+01  1.28722885e+01 -1.07503971e+00
 -4.92631614e+00 -2.79260785e+00 -1.30623661e+00  3.75621228e-01
 -1.77895958e+01 -9.73372597e+01 -7.65951663e+01 -2.52770726e+01
 -2.50439297e+00  2.80742394e+00  1.85659106e-01  1.34525197e+00
  1.85952569e+00 -2.50104422e-01 -4.12052509e+01 -1.77460808e+02
 -3.13841048e+02 -7.57379198e+02  2.29300973e+01 -1.30357118e+03
 -3.18848742e+02  2.38955315e+02 -1.47259431e+02 -1.20974059e+02
 -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  3.10728662e+01 -3.96947838e+01
 -1.10079321e+02 -1.02018376e+02 -4.26431152e+01 -4.76494283e+00
  2.47370853e+00  2.41048520e+00 -1.26789344e+00 -2.82238330e+00
  3.93243983e+01  1.27468020e+02  9.95699241e+01  4.36604153e+01
  9.52429625e+00 -6.51681124e-01 -1.84203320e+00 -3.57202953e-01
 -1.06529822e+00  8.62232301e-01  7.73654946e+01  2.63986253e+01
 -5.07954913e+01 -4.24052288e+01 -1.96989554e+01 -5.13078768e+00
 -6.38593132e-01  9.66981107e-01  5.09035088e-01 -8.06024947e-01
 -1.66070413e+01 -1.16467025e+02  2.57480056e-01  7.04624950e+01
  5.51595163e+01  2.55948236e+01  5.25604926e+00  1.61260828e-01
  1.47400526e+00 -2.48562473e+00  1.18961083e-01 -2.76086236e+00
  3.48594533e+01 -2.77452549e+02  1.42086497e+03 -4.41270888e+03
  7.04327600e+03 -2.55070912e+03 -3.89629847e+03 -2.14470199e+03
  9.42632396e-01  1.37823849e+00  2.25871657e-01 -4.13067099e-01
 -1.24773729e+00  7.65871107e-01 -1.53814332e+00  1.23776132e+00
  9.39910934e-02  1.06248336e-01  2.93736131e-01  9.23109865e-01
  1.89705988e+00  7.12553064e-01  4.42714146e-02 -8.81757070e-01
 -2.13622978e+00  1.27037372e+00 -1.47101698e+00  6.87601047e-02]
训练集上的RMSE = 0.00034758825604431213
```

第四问

使用随机参数

```
参数为[ 0.37385926 -0.60339092  0.2532812 ]
训练集上的RMSE = 0.033219606073691996
```

当b很大（10000）时

```
参数为[-1.53423200e-02  1.00004714e+04  3.54741376e-01]
训练集上的RMSE = 0.21403164012547363
```

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)))
```

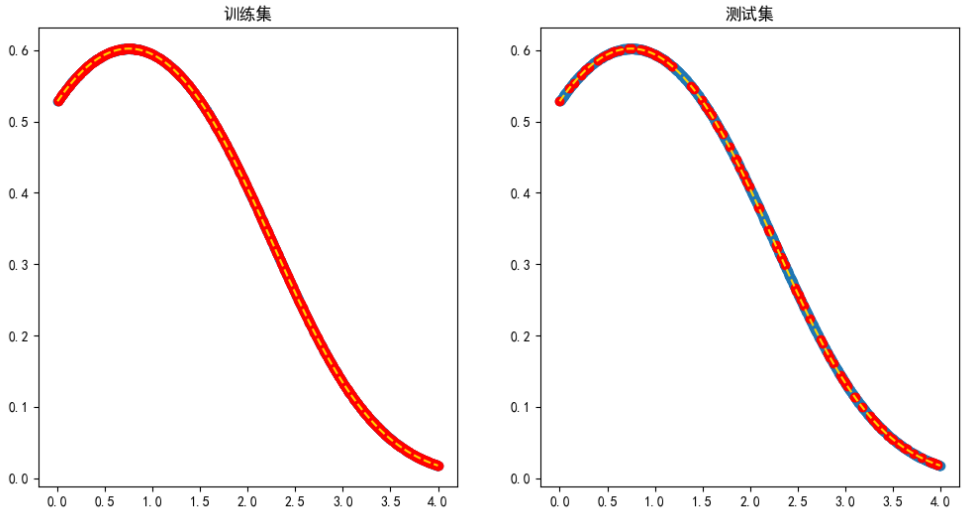
画训练集上的图

```
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")
```

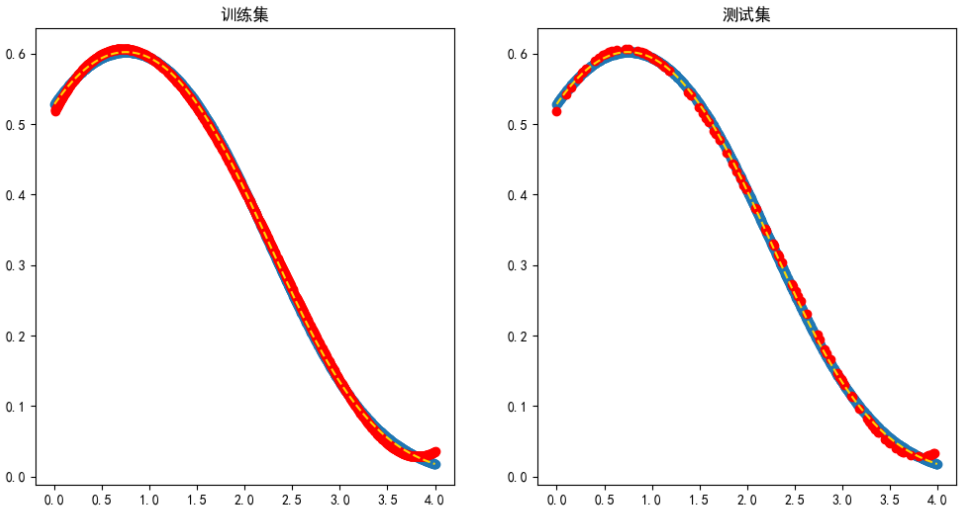
画测试集上的图

```
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")
```

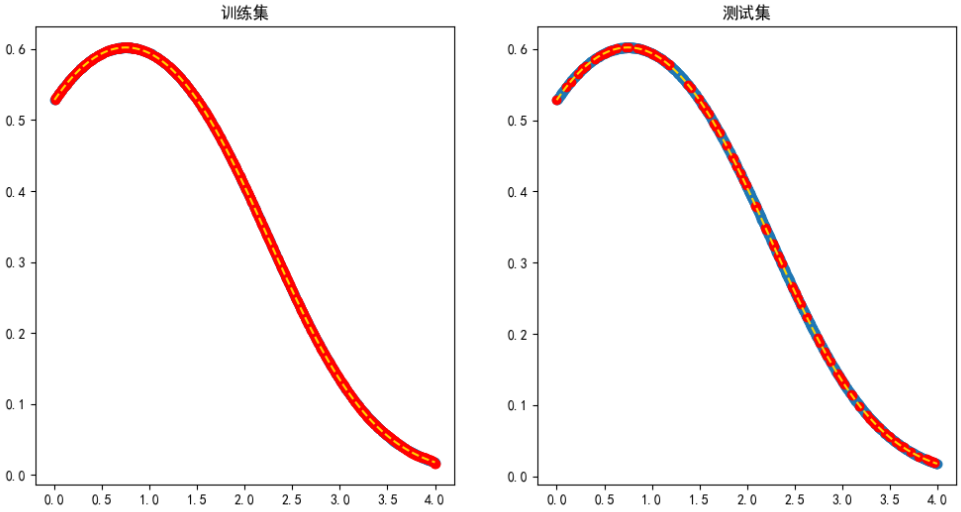
第一问



第二问

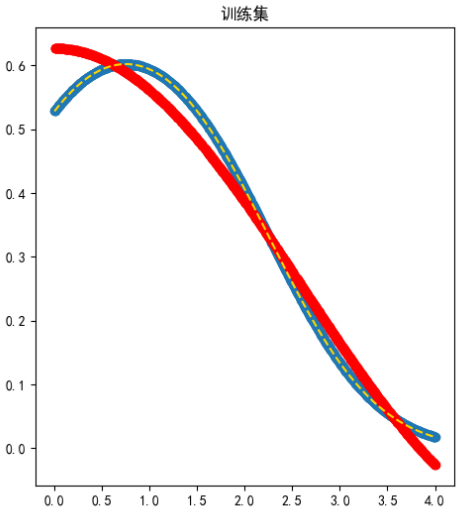


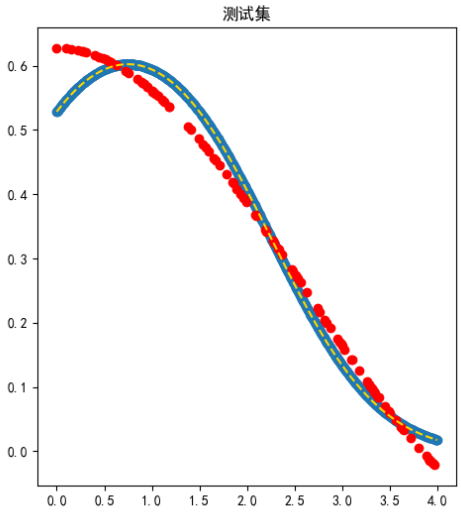
第三问



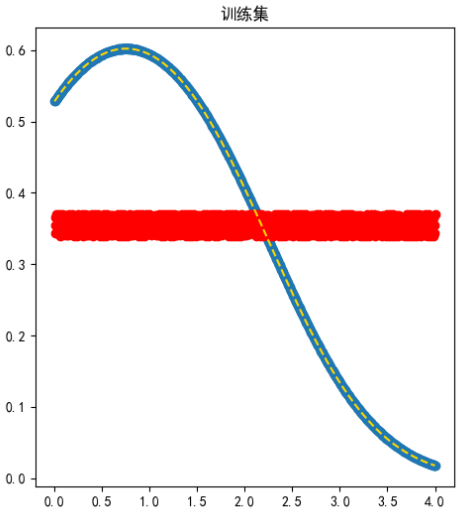
第四问

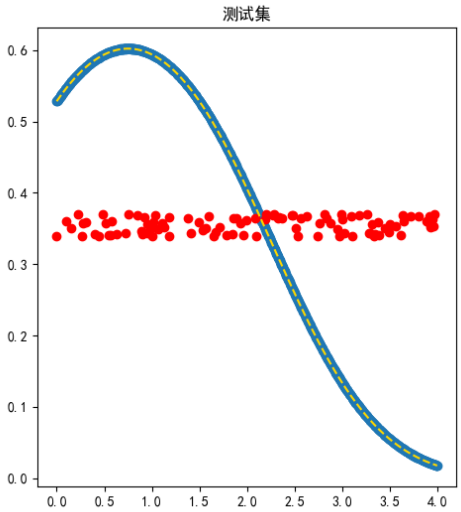
使用随机参数





b很大时





测试（初始值/N/h）

最优模型

初步看模型一和模型三拟合效果最好