Project Report of Machine Learning (Assignment 3)

Author: 赵文浩 23020201153860 (计算机科学系)

I. Plot Curve for Regularization Path

1.1 Task Description

以基于 L_2 正则化的多项式曲线拟合为例，试画出阶数 $M=3$ 时的正则化路径(Regularization Path)图，即以 $\|w_\lambda\|/\|w_\infty\|$ 为横坐标，w_λ^i 为纵坐标的模型参数变化曲线。其中：

$$w_\infty = \left(X^T X\right)^{-1} X^T y$$

$$w_\lambda = \left(X^T X + \lambda I\right)^{-1} X^T y$$

$$y(x,w) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

1.2 Dataset

本文仍基于正弦函数 $\sin(2\pi x)$ 生成原始数据，并在此基础上添加高斯噪声 $N(\mu=0.25,\sigma=1.0)$，得到的训练数据（保留三位小数）如表 1-1 和图 1-1 所示：

表 1-1 待拟合训练数据

x	0.000	0.111	0.222	0.333	0.444	0.555	0.666	0.777	0.888	1.000
$\sin(2\pi x)$	0.000	0.643	0.985	0.866	3.420	-3.420	-0.866	-0.985	-0.643	-0.245
y	-0.041	0.504	1.658	0.703	0.209	-0.591	-1.439	-0.947	-0.435	0.364

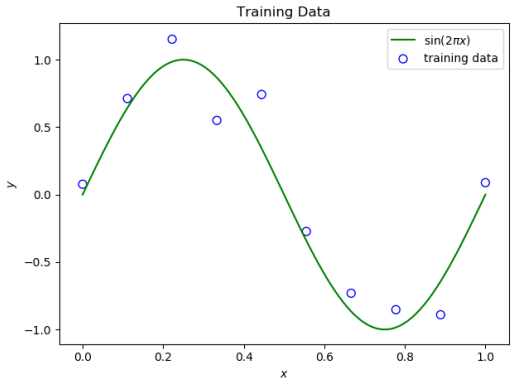


图 1-1 待拟合训练数据

1.3 Regularization Path and Analysis

1.3.1 Regularization Path with M=3

基于 Assignment 2 对正则化线性拟合的实现，可以很容易地绘制正则曲线，详细代码见 Section III。多项式阶数 $M=3$ 对应的正则化路径如图 1-2 所示。

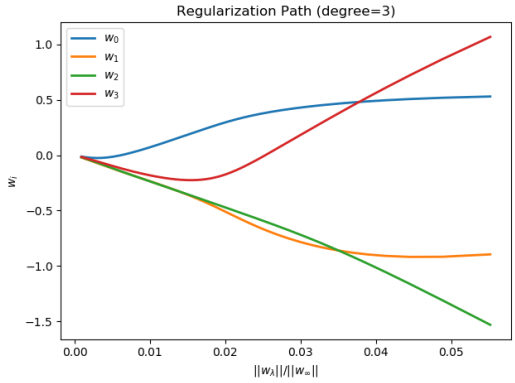


图 1-2 Regularization Path with M=3

1.3.2 Analysis of Regularization Path

图 1-2 展示的正则化路径是在模型阶数 $M=3$ 时、且正则化系数 $\lambda\in[0.04,100]$ 的范围内绘制的。引入正则化前后的模型参数求解结果如下：

$$w_\infty=\left(X^TX\right)^{-1}X^Ty$$
$$w_\lambda=\left(X^TX+\lambda I\right)^{-1}X^Ty$$

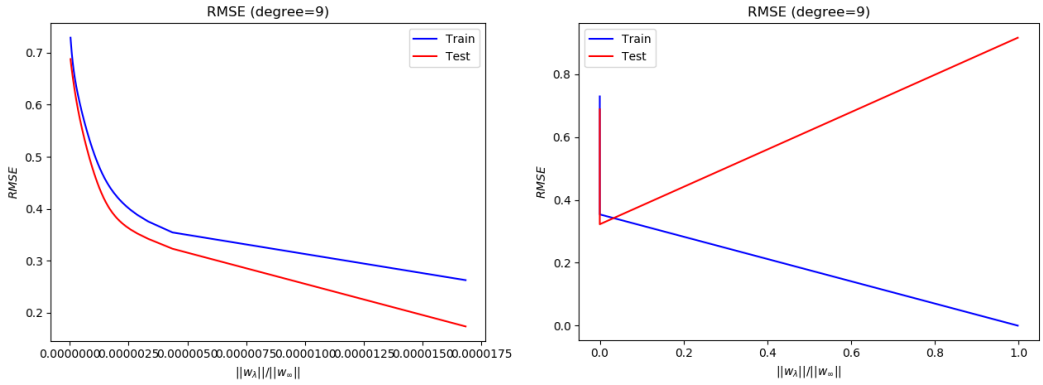
根据上式易知，随着 λ 的增大，w_λ 减小，进而使得正则化路径图的横坐标对应的坐标值减小；随着 λ 的减小，w_λ 增大，进而使得正则化路径对应坐标值增大。对应于如下极限情况：

$$\frac{\|w_\lambda\|}{\|w_\infty\|}=\begin{cases}1 & \lambda=0\\ 0 & \lambda\to\infty\end{cases}$$

为了更好的展示模型参数的细节变化，这里对于正则项系数选定的考察范围为 $\lambda\in[0.04,100]$，并在该区间内等间距地选取 10000 个坐标点进行绘制。从上图 1-2 中可以看出，随着 λ 的增大，即 $\|w_\lambda\|/\|w_\infty\|$ 降低，引起即多项式模型参数 w_λ^i 的量级 $\left|w_\lambda^i\right|$ 不断减小，这一现象符合正则化的目标——减小模型参数量级、缓和过拟合现象；随着 λ 的减小，即 $\|w_\lambda\|/\|w_\infty\|$ 增大，正则化的程度不断降低，减轻了对模型参数量级的限制，引起 $\left|w_\lambda^i\right|$ 整体回升（注意这里考量的是量级，因此是对参数绝对值进行走势分析），同样符合正则化目标。

1.3.3 Analysis of RMSE Regularization Path

下面从误差的角度分析正则化路径，这里以模型阶数 $M=9$ 为例，考察正则化操作分别对于模型在训练集和测试集上误差的影响。首先仍然将正则项系数 λ 设定在区间 $[0.04,100]$ 内，此时得到的 RMSE 正则化路径如图 1-3 (a)所示。



(a) $\lambda \in [0.04,100]$ (b) $\lambda \in [0,100]$

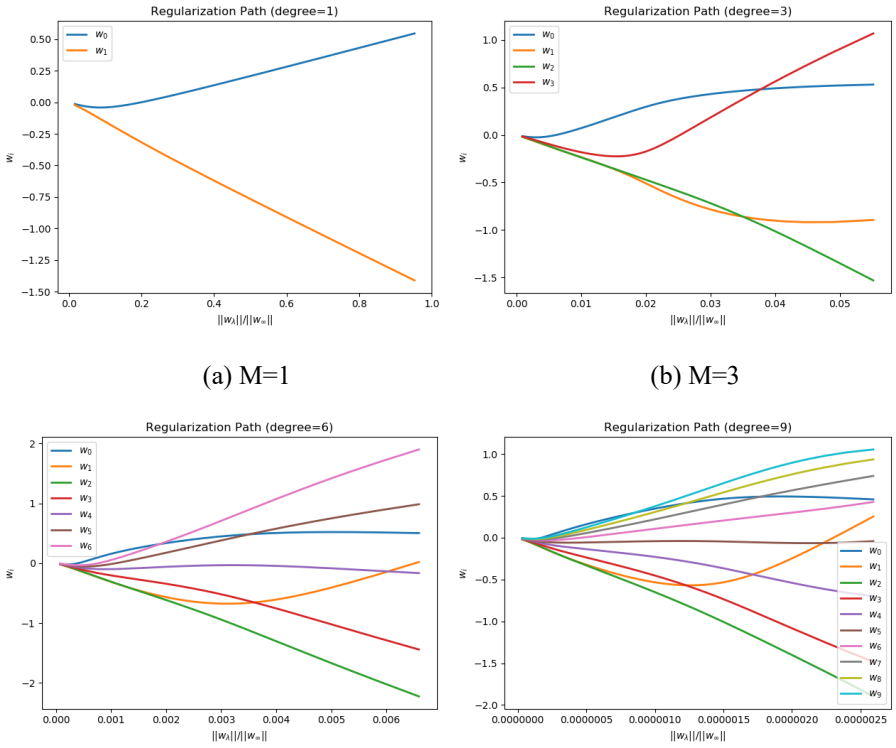
图 1-3 RMSE Regularization Path with M=9

从图 1-3 (a)中可以看出，随着λ的增大（横坐标向左移动），正则化的程度上升，模型在训练集上的误差有升高的趋势，这是因为正则化操作在一定程度上以牺牲训练集准确度为代价，提高模型的泛化能力；但从图中还可以发现，随着正则化程度的上升，模型在测试集上的准确度也在下降，不太符合预期。实际上，这一现象出现的主要原因是图(a)考察区间设定的过小，接下来我们将正则项系数的考察区间扩展到$[0,100]$，此时得到的 RMSE 正则化路径如图 1-3 (b)所示。

观察图 1-3 中图(a)和图(b)的横坐标区间，可以看出图(a)展示的 RMSE 正则化路径只是图(b)非常小的一部分。从图(b)的整体上看，当正则项系数$\lambda = 0$时，即$\|w_\lambda\|/\|w_\infty\| = 1$，此时模型对于训练集的准确度极高，但对于测试集的准确度极低（对应于图中两极分化的现象）。随着λ不断增大（横坐标向左移动），模型的泛化能力得到了提高。结合图 1-3 (a)和图 1-3 (b)可以得出如下结论——即使很小的正则项系数也会对模型的泛化能力有着很大程度的提高，但同时也需要在实际应用场景下选择适当的正则项系数，使之更好的应对新的数据。

1.4 Regularization Path with Different M

图 1-4 (a), (b), (c), (d)分别展示了阶数分别为 1、3、6、9 时的正则化路径图。与上述描述类似，随着正则化程度不断升高，模型参数量级在整体上呈减小趋势。



(a) M=1 (b) M=3

(c) M=6 (d) M=9

图 1-4 Regularization Path with Different M

II. Fit Points in Image with Two Dimension

2.1 Task Description

给定图片 2-1，试拟合该图片中的形状Ω。图像采用 Excel 标记，各个点的位置坐标见{第三次作业_第二题.xlsx}表格文件。

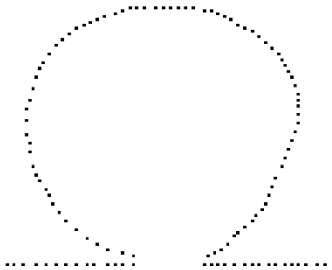


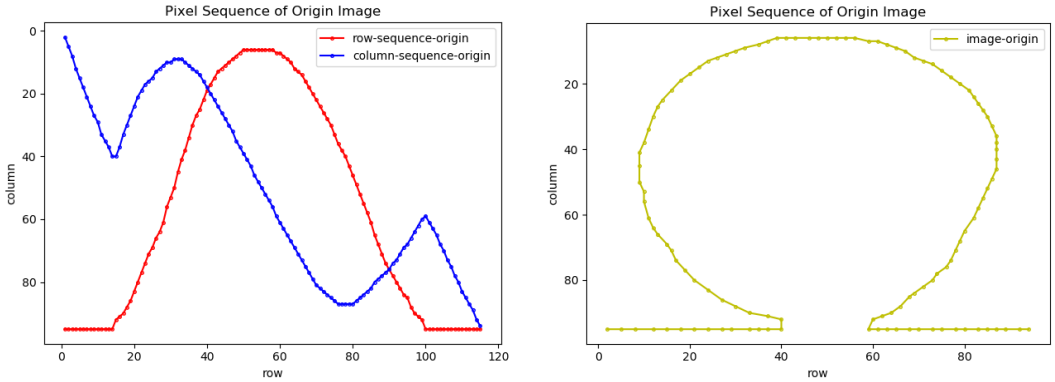
图 2-1 待拟合形状Ω

2.2 Task Analysis and Decomposition

函数定义：设非空数集$D \subset R$，若存在对应法则f，使得对于每个$x \in D$都有唯一确定的$y \in R$与之对应，则称该对应法则f为D上的函数，其表达式记为$y = f(x)$，$x \in D$。其中x称作自变量，y称作因变量[1]。

从图 2-1 中可以看出，待拟合形状对应的模型并不是一个函数，因为存在 x 使得对应的因变量 y 不唯一，因此无法直接采用最小二乘法对原始数据进行线性拟合，需要换个角度考虑。根据提供的数据，我们不妨将该形状看作为一张二维图片上 115 个像素点的组合，图像在这些像素点的像素值为 1，其余为 0。

基于此，我们可以通过分别对水平和竖直方向上像素点的位置序列（已提供）进行拟合，进而实现对整个形状的拟合，由此对问题进行了分解处理。在图像处理领域，像素点的位置均从左上角开始编号，图 2-1 所示形状对应的水平和竖直像素点位置序列如图 2-2 (a)所示；像素点组合形成的原始形状如图 2-2 (b)所示。



(a) Horizonal and Vertical Pixel Sequence　　(b) Image Pixel Sequence

图 2-2 Origin Image

2.3 Linear Fitting for H/V Pixel Sequence

从图 2-2 中可以看出，原始形状拟合的问题被分解为两个线性拟合问题，此时可采用最小二乘法进行求解并借助正则化操作优化多项式模型。这里采用正则项系数 $\lambda = 0.01$ 的多项式模型，对应的损失函数如下：

$$\tilde{E}(w) = \frac{1}{2}\sum_{k=1}^{N}\{y(x_k, w) - y_k\}^2 + \frac{\lambda}{2}\|w\|^2$$

接下来需要考虑模型的复杂度即多项式阶数的选择。题目给定了大小为 115 的训练数据，目的是尽可能地对原始形状进行拟合，因此这里使用全部训练数据且需要尽可能的降低预测值与真实值之间的差异，即在完成本次拟合任务时，不再考虑过拟合现象。通过 Assignment2 提交的《基于多项式的曲线拟合方法与分析》可知，在忽略过拟合现象的条件下，可以通过提高复杂度的方式提高模型对于训练集的准确度，但模型阶数过大也会引起计算量的上升，因此仍需选择合适的多项式模型。图 2-3 展示了引入正则化后，模型阶数对于训练集准确度的影响。

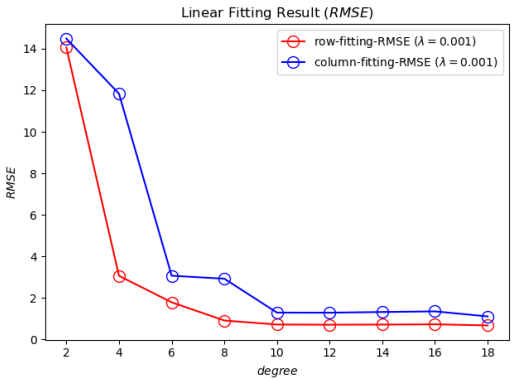


图 2-3 Fitting Result with Different Orders

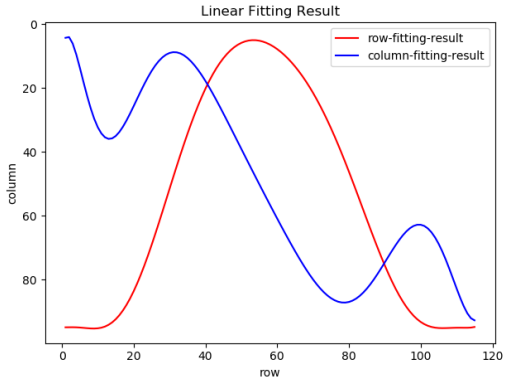
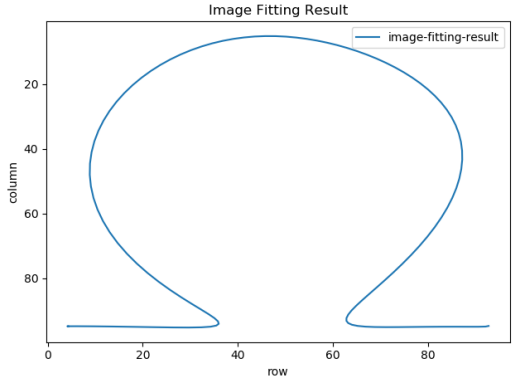


图 2-4 Linear Fitting Result

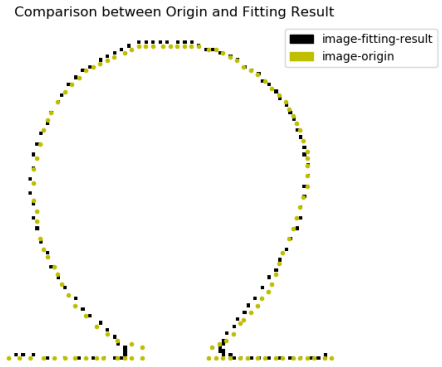
从图 2-3 中可以看出，水平和垂直方向上的像素序列的拟合效果都在 $M=10$ 附近达到最优，因此本文选用 $M=10$ 作为多项式模型的阶数，对应的拟合结果如图 2-4 所示，与图 2-2 (a)对应的原始像素序列吻合度较高。

2.4 Fitting Result Combination

Section 2.3 完成了水平和竖直方向上像素点位置序列的拟合，二者结合得到的形状即为最终的拟合结果，如图 2-5 (a)所示。为了更好地展示拟合效果与原始形状的匹配程度，这里创建大小为 96×93（基于水平和竖直方向上坐标的最大值得到）的画布，并将拟合结果对应的像素点突出显示，得到的二值图像与原始形状对比图如图 2-5 (b)所示，从图中可以看出，此方法达到了良好的匹配效果。



(a) Image Fitting Result on Coordinate



(b) Binary Image

图 2-5 Image Fitting Result

III. Code Analysis

3.1 Regularization Path

表 3-2 和表 3-3 分别展示了参数和 RMSE 误差正则化路径的绘制方法，二者实际上具有类似的代码结构；表 3-1 为主函数整体流程。

P.S. 完整代码见与本报告一同提交的工程文件 P3-Assignment-RegularizationPath.py

表 3-1 主函数整体流程

- 3-1 MainFunction

```
if __name__ == '__main__':
    # 生成训练和测试数据
    x_train, y_train = create_toy_data(func, 10, 0.25)
    x_test = np.linspace(0, 1, 100)
    y_test = func(x_test)
    # 绘制正则化路径曲线
    plot_regularization_path(1)
    plot_rmse(1)
    plot_regularization_path(3)
    plot_rmse(3)
    plot_regularization_path(6)
    plot_rmse(6)
    plot_regularization_path(9)
    plot_rmse(9)
```

表 3-2 参数正则化路径绘制

- 3-2 PlotRegularizationPath

```
def plot_regularization_path(degree):
    # 填充训练数据
    feature = PolynomialFeature(degree)
    X = feature.transform(x_train)
    # 未经正则化操作的 w_infinite
    w_infinite = np.linalg.pinv(X) @ y_train
    # 正则项系数（范围可调节，0.04-100 是较好的范围）
    alpha = np.linspace(0.04, 100, 10000)
    w_lambda = None
    # 依次计算不同正则项系数下的 w_lambda
    for i in alpha:
        w_lambda_i = np.linalg.solve(i * np.eye(np.size(X, 1)) + X.T @ X, X.T @ y_train)
        w_lambda = w_lambda_i if w_lambda is None else np.vstack((w_lambda, w_lambda_i))
    # 横坐标 ||w_lambda||/||w_infinite||
    path_x = [np.linalg.norm(i) / np.linalg.norm(w_infinite) for i in w_lambda]
    plt.title('Regularization Path (degree={0})'.format(degree))
    plt.xlabel('$||w_\lambda||/||w_∞||$')
    plt.ylabel('$w_i$')
    for i in range(0, degree + 1):
        # 依次绘制 wi 的正则化路径
        plt.plot(path_x, w_lambda[:, i], label='$w_{0}$'.format(i), linewidth='2')
    plt.show()
```

表 3-3 RMSE 误差正则化路径绘制

- 3-3 PlotRegularizationPathRMSE

```
def plot_rmse(degree):
    # 填充训练数据
    feature = PolynomialFeature(degree)
    X = feature.transform(x_train)
    # 未经正则化操作的 w_infinite
    w_infinite = np.linalg.pinv(X) @ y_train
    # 正则项系数
    alpha = np.linspace(0.04, 100, 10000)
    # 不同正则项系数下的 w_lambda
    w_lambda = None
    # 训练集和测试集 RMSE 误差
    training_errors = []
    test_errors = []
    # 依次计算不同正则项系数下的 w_lambda
    for i in alpha:
        w_lambda_i = np.linalg.solve(i * np.eye(np.size(X, 1)) + X.T @ X, X.T @ y_train)
        w_lambda = w_lambda_i if w_lambda is None else np.vstack((w_lambda, w_lambda_i))
        model = RidgeRegression(i)
        model.fit(X, y_train)
        training_errors.append(rmse(model.predict(X), y_train))
        test_errors.append(rmse(model.predict(feature.transform(x_test)), y_test))
    # 横坐标 ||w_lambda||/||w_infinite||
    path_x = [np.linalg.norm(i) / np.linalg.norm(w_infinite) for i in w_lambda]
    # 绘制 RMSE 曲线
    plt.title('RMSE (degree={0})'.format(degree))
    plt.xlabel('$||w_\lambda||/||w_∞||$')
    plt.ylabel("$RMSE$")
    plt.plot(path_x, training_errors, '-', mfc="none", mec="b", ms=10, c="b", label="Train")
    plt.plot(path_x, test_errors, '-', mfc="none", mec="r", ms=10, c="r", label="Test")
    plt.legend()
    plt.show()
```

3.2 Two Dimensions Points Fitting

以下展示了二维形状拟合的主要代码，在编程和绘图时需要注意原始数据从左上角开始编号，因此需要将纵坐标逆置。

P.S. 完整代码见与本报告一同提交的工程文件 P3-Assignment-ImageFitting.py

表 3-4 主函数整体流程

- 3-4 MainFunction

```
if __name__ == '__main__':
    # 读取数据
    index, row, column = load_data('./ImageData.txt')
    # 拟合模型阶数和对应的正则项系数
    degree_row, degree_column, alpha_row, alpha_column = 10, 10, 1e-3, 1e-3
    # 绘制原始图像
    plot_origin_image()
    # 获取拟合效果最佳的 degree
    find_best_degree_based_on_rmse()
    # 绘制拟合结果
    plot_fitting_result()
    # 绘制拟合得到的目标图像
    fitting_result_to_image()
```

表 3-5 原始图像绘制

- 3-5 PlotOriginImage

```
def plot_origin_image():
    plt.title('Pixel Sequence of Origin Image')
    plt.xlabel('row')
    plt.ylabel('column')
    # 绘制原始数据
    plt.plot(index, row, '.-', mfc="none", mec="r", ms=5, c='r', label='row-sequence-origin')
    plt.plot(index, column, '.-', mfc="none", mec="b", ms=5, c='b', label='column-sequence-
origin')
    plt.plot(column, row, '.-', mfc="none", mec="y", ms=5, c='y', label='image-origin')
    # 注意原始数据从左上角开始编号，因此需要将纵坐标逆置
    plt.gca().invert_yaxis()
    plt.legend()
    plt.show()
```

表 3-6 基于 RMSE 指标寻找最佳阶数 degree

- 3-6 FindBestDegreeBasedOnRMSE

```
def find_best_degree_based_on_rmse():
    row_training_errors = []
    column_training_errors = []
    degrees = np.arange(2, 20, 2)
    for degree in degrees:
        X = PolynomialFeature(degree).transform(index)
        model_row = RidgeRegression(alpha_row)
        model_column = RidgeRegression(alpha_column)
```

- 3-6 FindBestDegreeBasedOnRMSE

```
            model_row.fit(X, row)
            model_column.fit(X, column)
            row_training_errors.append(rmse(model_row.predict(X), row))
            column_training_errors.append(rmse(model_column.predict(X), column))
        plt.title('Linear Fitting Result ($RMSE$)')
        plt.plot(degrees, row_training_errors, 'o-', mfc="none", mec="r", ms=10, c="r",
                 label="row-fitting-RMSE ($\lambda=%.3f$)" % alpha_row)
        plt.plot(degrees, column_training_errors, 'o-', mfc="none", mec="b", ms=10, c="b",
                 label="column-fitting-RMSE ($\lambda=%.3f$)" % alpha_column)
        plt.xlabel("$degree$")
        plt.ylabel("$RMSE$")
        plt.legend()
        plt.show()
```

表 3-7 绘制拟合结果

- 3-7 PlotFittingResult

```
def plot_fitting_result():
    # 对图像横坐标执行拟合
    R, w_row = linear_fit(index, row, degree_row, alpha_row)
    # 对图像纵坐标执行拟合
    C, w_column = linear_fit(index, column, degree_column, alpha_column)
    plt.title('Linear Fitting Result')
    plt.xlabel('row')
    plt.ylabel('column')
    # 绘制拟合曲线
    plt.plot(index, R @ w_row, c="r", label="row-fitting-result")
    plt.plot(index, C @ w_column, c="b", label="column-fitting-result")
    # 注意原始数据从左上角开始编号，因此需要将纵坐标逆置
    plt.gca().invert_yaxis()
    plt.legend()
    plt.show()
```

表 3-8 由线性拟合结果生成图像

- 3-8 FittingResultToImage

```
def fitting_result_to_image():
    # 对图像横坐标执行拟合
    x_row, w_row = linear_fit(index, row, degree_row, alpha_row)
    # 对图像纵坐标执行拟合
    x_column, w_column = linear_fit(index, column, degree_column, alpha_column)
    # 计算得到的纵坐标
    y_row = x_row @ w_row
    y_column = x_column @ w_column
```

- 3-8 FittingResultToImage

```
plt.title('Image Fitting Result')
plt.xlabel('row')
plt.ylabel('column')
# 以 y_column、y_row 分为图像横坐标和纵坐标绘图
plt.plot(y_column, y_row, label='image-fitting-result')
plt.xlim((0, 100))
# 注意原始数据从左上角开始编号，因此需要将纵坐标逆置
plt.gca().invert_yaxis()
plt.legend()
plt.show()
# 图像生成
image = np.ones((int(max(y_row)) + 1, int(max(y_column)) + 1), dtype=np.int8)
for i in range(0, len(index)):
    image[int(y_row[i]), int(y_column[i])] = 0
plt.title('Comparison between Origin and Fitting Result')
plt.axis('off')
plt.imshow(image, cmap='gray')
plt.scatter(column, row, marker='.', c='y')
patches = [mpatches.Patch(color='k', label='image-fitting-result'),
           mpatches.Patch(color='y', label='image-origin')]
plt.legend(handles=patches, bbox_to_anchor=(0.8, 1))
plt.show()
```

参考文献

[1] 高等数学.映射与函数[Online]. https://zhuanlan.zhihu.com/p/350727454, 2021-03-20/2021-04-02