一个简单的lwlr模型

一些事项

- lwlr(locally weighed linear regression) 局部加权线性回归模型预测不同的点时，需要训练不同的模型，因为预测点不同时损失函数就不同
- 经典线性回归的损失函数是：

$$J(\theta) = \sum (y^{(i)} - \theta^T x^{(i)})^2$$

  lwlr的基本想法是靠近数据集中靠近预测点的样本对预测点的影响更大，远离预测点的样本对预测点的影响小，因此在损失函数前乘了一项权重，来反映距离预测点的影响：

$$J(\theta) = \sum_i \omega^{(i)} (y^{(i)} - \theta^T x^{(i)})^2$$

  最常用的核是高斯核：

$$\omega^{(i)} = \exp(-\frac{(x_i - x_0)^2}{2\sigma^2})$$

  其中x_0是要预测的点，x_i是数据集中的点，σ的值决定权重的衰减快慢，每预测不同的点时，由于x_0的值不同，因此需要重新训练一个模型

- 高斯核只衡量点之间的距离，因此无论x的特征是多少维的，高斯核的输出都是一个标量，有多少个ω只取决于有多少个数据集，与特征维数无关
- 对$J(\theta)$求导为0可以求得：

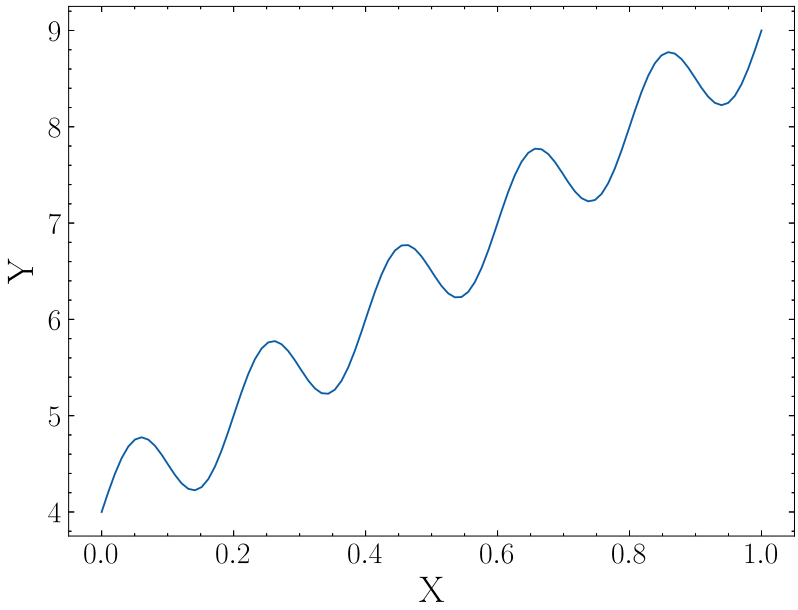
$$\theta = \left(X^T W X\right)^{-1} X^T W \vec{y}$$

- 参数向量、输入数据集、标签的样式的统一接口都是$N_{(\text{数据集维数})} \times M_{(\text{特征维数})}$
- numpy：`[1,2].T`没有任何变化，`[[1,2]].T`才会求转置，因为二者的shape不同.

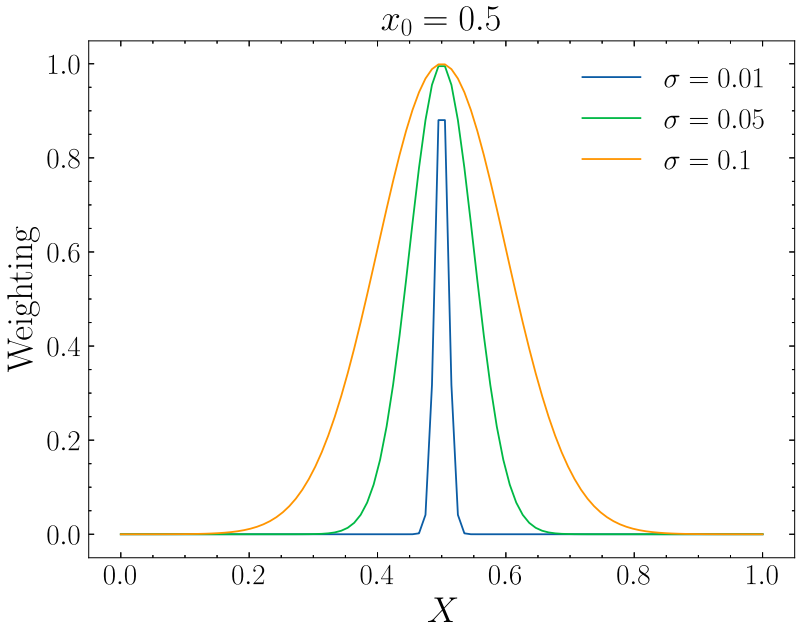
```
# 简单配置
import matplotlib.pyplot as plt
import numpy as np

plt.style.use("science")
%config InlineBackend.figure_format = 'svg'

# 生成训练集
X = np.dstack([np.linspace(0, 1, 100), np.ones(100)]).squeeze()
Y = X @ np.array([[5, 4]]).T + 0.5 * np.sin(10 * np.pi * X[:, 0].reshape(-1, 1))
```




```
# 高斯权函数
def kernel(x0, X, sigma):
    return np.exp(-np.sum(((X - x0) ** 2), axis=1) / 2 / sigma ** 2)
```



```
# 计算的参数值
def theta(x0, X, sigma):
    W = np.diag(kernel(x0, X, sigma))
    return np.linalg.inv(X.T @ W @ X) @ X.T @ W @ Y

# 生成数据
theta_list = [np.array([theta(x0, X, sigma) for x0 in X]).squeeze() for sigma in
[0.01, 0.05, 0.1]]
result_list = [[np.dot(x, theta) for x, theta in zip(X, thetas)] for thetas in
theta_list]
```