构建一个KNN 回归器

- import numpy as np
- import matplotlib.pyplot as plt
- from sklearn import neighbors
- '''　功能作用： 构建一个KNN 回归器　工作原理： 回归器的目标是预测连续值的输出。这个例子中并没有固定数量的输出类别，仅有一组实际输出值，我们希望回归器可以预测未知数据点的输出值。这个例子中用到一个sinc函数来演示KNN回归器，该函数也被称为基本正弦函数。
- sinc(x) = sin(x)/x　(x != 0)
- 　　= 1　(x = 0)
- 当x为0时，sin(x)/x变成0/0不定式，因此需要计算该函数在x无限趋近于0时函数的极限。我们用到了一组值做训练，并且定义了一个更密集的网格点来进行测试。正如在之前的图中看到的，输出曲线接近于训练输出。'''# 生成样本数据amplitude = 10num_points = 100X = amplitude * np.random.rand(num_points, 1) - 0.5 * amplitude
- # 计算目标 添加噪声y = np.sinc(X).ravel()
- # 目标y += 0.2 * (0.5 - np.random.rand(y.size)) # 噪声
- # 画出输入数据的图形plt.figure()
- plt.scatter(X, y, s=40, c='k', facecolors='none')
- plt.title('输入数据')
- # 用输入数据10倍的密度创建一个一维网络'''
- # np.newaxis 添加一个新的维度例如：
- 原型：　x= array([4, 6, 6, 6, 5])
- 处理过程：x2 = x[:, np.newaxis]
- 结果:　array([[4],[6],[6],[6],[5]])'''
- x_values = np.linspace(-0.5*amplitude, 0.5*amplitude, 10*num_points)[:, np.newaxis]
- # 定义最近邻的个数n_neighbors = 8
- # 定义 并 训练回归器
- knn_regressor = neighbors.KNeighborsRegressor(n_neighbors, weights='distance')
- y_values = knn_regressor.fit(X, y).predict(x_values)
- plt.figure()
- plt.scatter(X, y, s=40, c='k', facecolors='none', label='input data')
- plt.plot(x_values, y_values, c='k', linestyle='--', label='predicted values')
- plt.xlim(X.min() - 1, X.max() + 1)
- plt.ylim(y.min() - 0.2, y.max() + 0.2)
- plt.axis('tight')
- plt.legend()
- plt.title('K Nearest Neighbors Regressor')
- plt.show()

幕布 - 思维概要整理工具