机器学习 **Lecture 7**

KNN

一、 距离度量

Defination 1 (欧式距离).

$$d(\mathbf{x},\mathbf{y}) = ||\mathbf{x}-\mathbf{y}||^2$$

Defination 2 (曼哈顿距离).

$$d(\mathbf{x},\mathbf{y}) = ||\mathbf{x}-\mathbf{y}||_1$$

Defination 3 (切比雪夫距离).

$$d(\mathbf{x},\mathbf{y}) = ||\mathbf{x}-\mathbf{y}||_\infty$$

Defination 4 (闵可夫斯基距离).

$$d(\mathbf{x},\mathbf{y}) = ||\mathbf{x}-\mathbf{y}||_p$$

二、 算法

KNN 没有训练过程，只是将数据集储存起来，每次预测的时候遍历数据集进行预测。

Algorithm 1 KNN

Input: 数据集 $D = \{x_i, y_i\}_{i=1}^m$; 新的样本点 y;k

1: **for** $i = 1, 2, \cdots, m$ **do**
2: **Compute** $d_i = d(x, y)$
3: **end for**
4: 对 d_i 进行排序
5: 选择前 k 个样本点，以 k 个样本点中频次最高的类作为 y 的类别

1. 代码实现

```python
from numpy.linalg import norm
import numpy as np
from tqdm import tqdm
import matplotlib.pyplot as plt


class KNN():
```

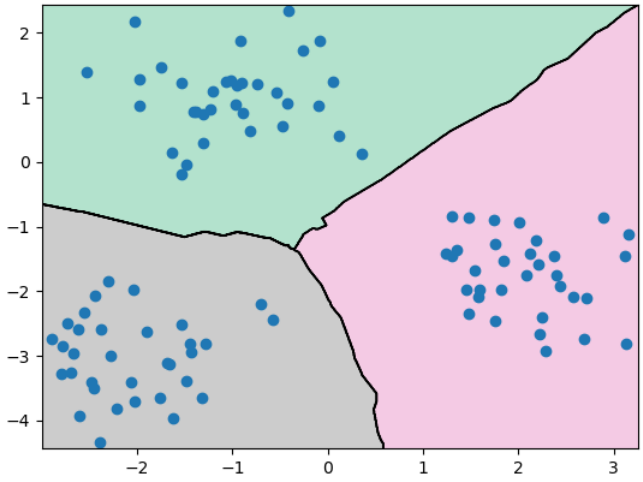

图 1: KNN 决策边界

```
    def __init__(self, X, y, k):
        self.X = X
        self.y = y
        self.k = k
        self.m = self.X.shape[0]
        self.n = self.X.shape[1]

    def predict(self, x):
        dists = []
        for i in range(self.m):
            dists.append([self.dist(x, self.X[i, :]), self.y[i]])
        dists = np.array(dists)
        idx = np.argsort(dists[:, 0])
        points = dists[idx[:self.k]]
        counts = np.bincount(np.array(points[:, 1], dtype=np.int32))
        counts = np.argmax(counts)
        return counts

    def dist(self, x, y):
        return norm(x - y, ord=2)

    def plot_decision_boundaries(self, resolustion=1000):
        '''
        绘制决策边界
        :param resolustion: 网格密度
        :param iteration: 迭代次数
        :return:
        '''
        mins = self.X.min(axis=0) - 0.1
        maxs = self.X.max(axis=0) + 0.1
```

```
        xx, yy = np.meshgrid(np.linspace(mins[0], maxs[0], resolustion),
                             np.linspace(mins[1], maxs[1], resolustion))
        grid = np.c_[xx.ravel(), yy.ravel()]
        predict = []
        for i in tqdm(grid):
            predict.append(self.predict(i))
        predict = np.array(predict)
        predict = predict.reshape(xx.shape)
        # print(predict)

        plt.contourf(predict, extent=(mins[0], maxs[0], mins[1], maxs[1]),
                     cmap='Pastel2')

        plt.contour(predict, extent=(mins[0], maxs[0], mins[1], maxs[1]),
                    linewidths=1, colors='k')

        plt.scatter(self.X[:, 0], self.X[:, 1])
        plt.savefig('KNN.png')
        plt.close()
```