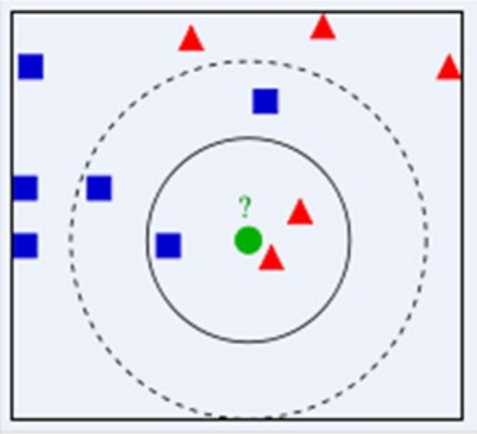
机器学习实验 2. 最近邻分类 KNN

K-近邻算法（KNN）原理概述

K 近邻算法是一种基本分类和回归方法。**本实验只讨论分类问题**的 K 近邻法。该方法通过测量不同特征值之间的距离进行分类。它的思路是：如果一个样本在特征空间中的 K 个最相似(即特征空间中最邻近)的样本中的大多数属于某一个类别，则该样本也属于这个类别，其中 K 通常是不大于 20 的整数。KNN 算法中，所选择的邻居都是已经正确分类的对象。该方法在定类决策上只依据最邻近的一个或者几个样本的类别来决定待分样本所属的类别。

下面通过一个简单的例子说明一下：如下图，绿色圆要被决定赋予哪个类，是红色三角形还是蓝色四方形？如果 K=3，由于红色三角形所占比例为 2/3，绿色圆将被赋予红色三角形那个类，如果 K=5，由于蓝色四方形比例为 3/5，因此绿色圆被赋予蓝色四方形类。



由此也说明了 KNN 算法的结果很大程度取决于 K 的选择。

在 KNN 中，通过计算对象间距离来作为各个对象之间的非相似性指标，避免了对象之间的匹配问题，在这里距离一般使用欧氏距离或曼哈顿距离：

欧式距离： $d(x,y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$ ；

曼哈顿距离： $d(x,y)=\sum_{i=1}^{n}|x_i-y_i|$ 。

同时，KNN 通过依据 k 个对象中占优的类别进行决策，而不是单一的对象类别决策。这两点就是 KNN 算法的优势。

接下来对 KNN 算法的思想总结一下：就是在训练集中数据和标签已知的情况下，输入测试数据，将测试数据的特征与训练集中对应的特征进行相互比较，找到训练集中与之最为相似的前 K 个数据，则该测试数据对应的类别就是 K 个数据中出现次数最多的那个分类。

KNN 分类算法描述

1）计算测试数据与各个训练数据之间的距离；

2）按照距离的递增关系进行排序；

3）选取距离最小的 K 个点；

4）确定前 K 个点所在类别的出现频率；

5）返回前 K 个点中出现频率最高的类别作为测试数据的预测分类。

KNN 分类算法在模拟数据中的一般实现

1）生成已标记的数据集：

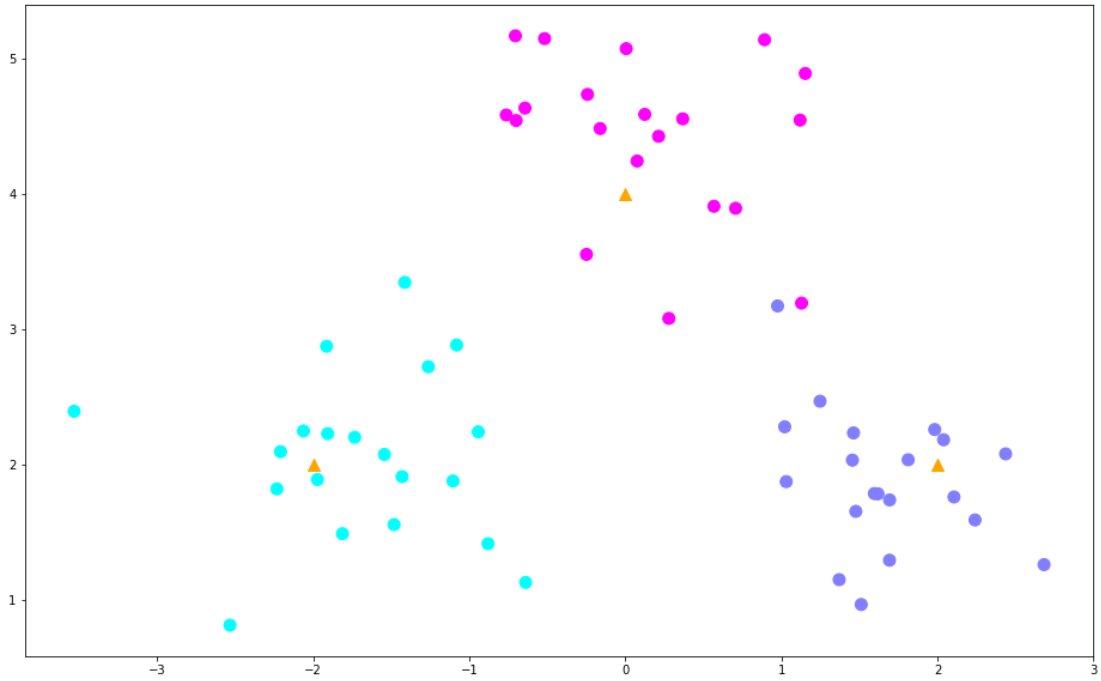
[1]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
```

[2]:
```
from sklearn.datasets.samples_generator import make_blobs
# 生成数据
centers = [[-2, 2], [2, 2], [0, 4]]
X, y = make_blobs(n_samples=60, centers=centers, random_state=0, cluster_std=0.60)
```

我们使用 sklearn.datasets.samples_generator 包下的 make_blobs()函数来生成模拟数据集。这里我们生成 60 个训练样本，并用 centers 参数指定 60 个训练样本分布的 3 个类中心。cluster_std 是标准差，用来指明生成的点分布的松散程度。生成的训练数据集放在变量 X 里面，数据集的类别标签放在 y 里面。

```
[3]: # 画出数据
plt.figure(figsize=(16, 10))
c = np.array(centers)
plt.scatter(X[:, 0], X[:, 1], c=y, s=100, cmap='cool');          # 画出样本
plt.scatter(c[:, 0], c[:, 1], s=100, marker='^', c='orange');    # 画出中心点
```

利用 matplotlib 库实现生成数据的直观可视化，如下图所示。



这些点的分布情况在坐标中一目了然，其中三角形的点即各个类别的中心节点。

2）使用 KNeighborsClassifier 来对算法进行训练，我们选择的参数是 K=5；

```
[4]: from sklearn.neighbors import KNeighborsClassifier
# 模型训练
k = 5
clf = KNeighborsClassifier(n_neighbors=k)
clf.fit(X, y);
```

3）对一个新的样本进行预测

```
[5]: # 进行预测
X_sample = [0, 2]
X_sample = np.array(X_sample).reshape(1, -1)
y_sample = clf.predict(X_sample);
neighbors = clf.kneighbors(X_sample, return_distance=False);
```

我们要预测的样本是[0, 2]，使用 kneighbors()方法，把这个样本周围距离最近的 5 个点取出来，取出来的点数训练样本里的索引，从 0 开始计算。

4）把待预测的样本以及和其最近的 5 个点标记出来：

[6]:
```
# 画出示意图
plt.figure(figsize=(16, 10))
plt.scatter(X[:, 0], X[:, 1], c=y, s=100, cmap='cool')    # 样本
plt.scatter(c[:, 0], c[:, 1], s=100, marker='^', c='k')   # 中心点
plt.scatter(X_sample[0][0], X_sample[0][1], marker="x",
            s=100, cmap='cool')    # 待预测的点

for i in neighbors[0]:
    # 预测点与距离最近的 5 个样本的连线
    plt.plot([X[i][0], X_sample[0][0]], [X[i][1], X_sample[0][1]],
             'k--', linewidth=0.6);
```

从图中可以清楚地看出 K 近邻算法的原理。