企业级实训
EPM

# kNN算法

# 主要内容

1. kNN算法描述
2. Python实现

# kNN算法描述

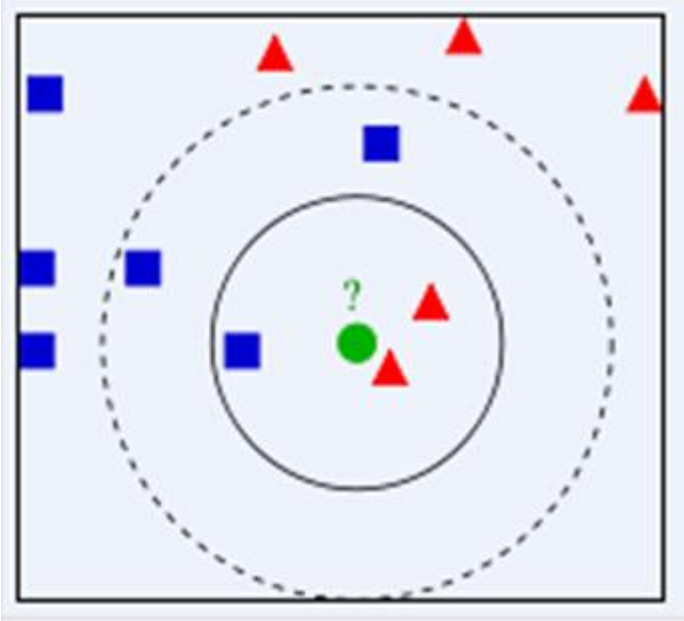
# 2.1 kNN算法三要素

K近邻的三要素：k值选择、距离度量和分类决策规则(取均值的决策规则)。

## K值选择

当k=1是的K近邻算法称为最近邻算法。此时将训练集中与$\vec{x}$最近的点的类别作为$\vec{x}$的分类。

K值的选择会对k近邻法的结果产生重大影响。



## K值选择

应用中，k值一般取一个较小的数值。通常采用交叉验证法来选取最优的k值，就是比较不同k值时的交叉验证平均误差率，选择误差率最小的那个k值。

# 2.1 kNN算法三要素

## 距离度量

二维平面上两点a(x1,y1)与b(x2,y2)间的欧氏距离：

$$d_{12} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

三维空间两点a(x1,y1,z1)与b(x2,y2,z2)间的欧氏距离：

$$d_{12} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

两个n维向量a(x11,x12,…,x1n)与 b(x21,x22,…,x2n)间的欧氏距离：

$$d_{12} = \sqrt{\sum_{k=1}^{n} (x_{1k} - x_{2k})^2}$$

# 2.1 kNN算法三要素

## 距离度量

样本有多个参数，每一个参数都有自己的定义域和取值范围，它们对距离计算的影响也就不一样，如取值较大的影响力会盖过取值较小的参数。为了公平，样本参数必须做一些归一化处理，最简单的方式就是所有特征的数值都采取归一化处置。

$$norm\_value = (value - min) \ / (max - min)$$

为待归一化处理数据集（matrix）

① 分别找出matrix中最大值和最小值

② 找出最大值与最小值差值

③ 根据公式对待归一化矩阵进行差与除操作

④ 返回归一化数据集和最小值

# 2.1 kNN算法三要素

## 分类决策规则

分类决策通常采用多数表决。也可以基于距离的远近进行加权投票，距离越近的样本权重越大。

# 2.2 kNN算法描述（伪代码）



**对未知类别属性的数据集中的每个点依次执行以下操作：**

1. 计算已知类别数据集中的点与当前点之间的距离；
2. 按照距离递增次序排序；
3. 选取与当前点距离最小的k个点；
4. 确定前k个点所在类别的出现频率；
5. 返回前k个点出现频率最高的类别作为当前点的预测分类；

企业级实训
EPM

# Python实现

Hewlett Packard
Enterprise

# 3.2 sklearn实现KNN

Scikit-learn中提供了一个KNeighborsClassifier类来实现k近邻法分类模型。

常用参数：

n_neighbors：一个整数，指定k值，默认为5。

常用方法：

fit(X,y)：训练模型

predict(X)：预测，返回待预测样本的标记。

score(X, y)：返回在(X,y)上预测的准确率。

# 3.2 sklearn实现KNN

```
from sklearn import neighbors, datasets
from sklearn.model_selection import train_test_split


# 加载手写识别数据集
def load_classification_data():
    digits = datasets.load_digits()
    X_train = digits.data
    y_train = digits.target
    return train_test_split(X_train, y_train, test_size=0.25,
random_state=0, stratify=y_train)
```

# 3.2 sklearn实现KNN

```
# 测试K近邻分类模型
def test_KNeighborsClassifier(*data):
    X_train, X_test, y_train, y_test = data
    clf = neighbors.KNeighborsClassifier()
    clf.fit(X_train, y_train)
    print("Training Score: %f" % clf.score(X_train, y_train))
    print("Testing Score: %f" % clf.score(X_test, y_test))


if __name__ == "__main__":
    X_train, X_test, y_train, y_test = load_classification_data()
    test_KNeighborsClassifier(X_train, X_test, y_train, y_test)
```

结果：

```
Training Score: 0.991091
Testing Score: 0.980000
```

# 3.2 sklearn实现特征值归一化

Sklearn.preprocessing中提供了MinMaxScaler类来实现最小最大值标准化。其使用方式如下：

```
from sklearn.preprocessing import MinMaxScaler
minMax = MinMaxScaler()
# 特征值归一化
X = minMax.fit_transform(X)
# 同比例归一化被识别样本
cnt = minMax.transform(cnt)
```

企业级实训
EPM

# The end

Hewlett Packard
Enterprise