机器学习与数据挖掘

Machine Learning & Data Mining

权小军 教授

中山大学数据科学与计算机学院

quanxj3@mail.sysu.edu.cn

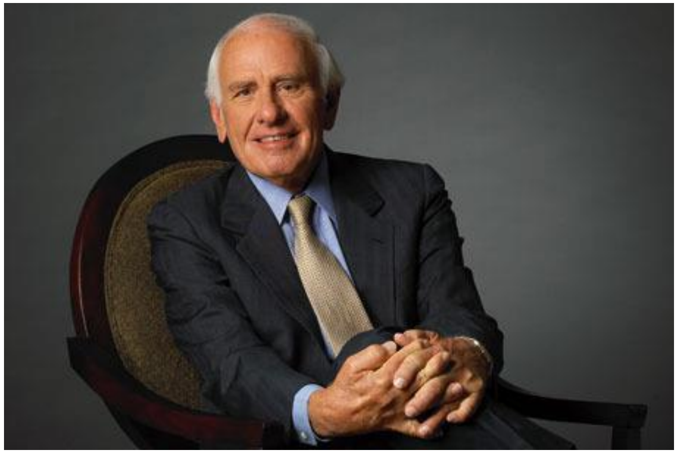
1. Preface

Preface

"You are the average of the five people you spend the most time with."

— Jim Rohn

"Jim" Rohn was an American entrepreneur, author and motivational speaker. Hs rags to riches story played a large part in his work, which influenced others in the personal development industry.



Preface

"You are the average of the five people you spend the most time with."

— Jim Rohn

1.物以类聚，人以群分

2.近朱者赤，近墨者黑

Preface

与凤凰同飞，必是俊鸟

与虎狼通行，必是猛兽

Preface

- Tell me about your friends(*who your neighbors are*) and *I will tell you who you are*.

Preface

Instance-based Learning

2. K-Nearest Neighbors Algorithm

k-nearest neighbors algorithm

In pattern recognition, the *k*-nearest neighbors algorithm (*k*-NN) is a non-parametric method used for **classification** and **regression**. In both cases, the input consists of the *k* closest training examples in the feature space.

-Wikipedia

k-nearest neighbors algorithm

Different names:

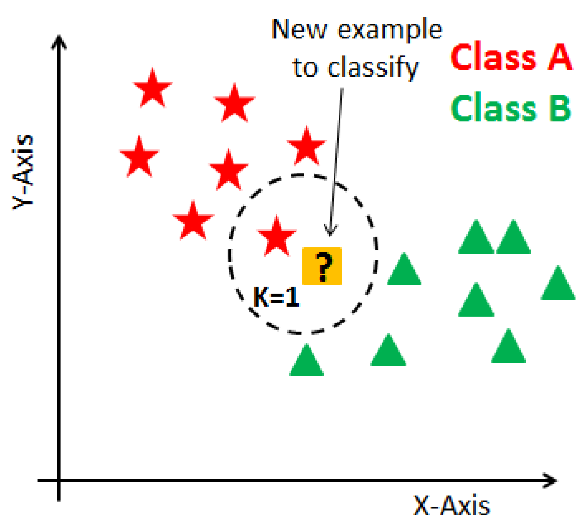
- K-Nearest Neighbors
- Example-Based Reasoning
- Instance-Based Learning
- Lazy Learning

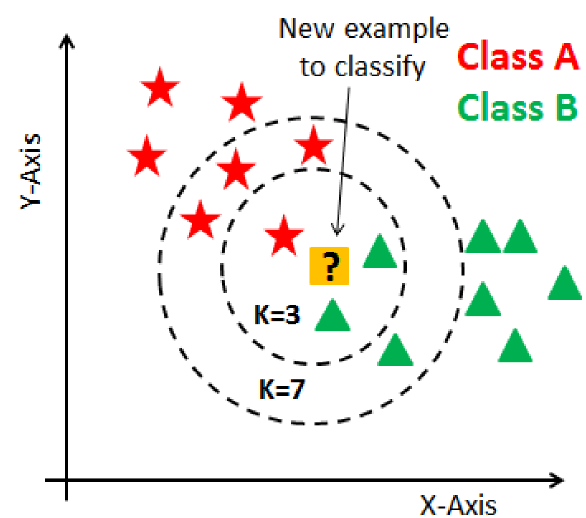
k-nearest neighbors algorithm

- A powerful classification algorithm used in pattern recognition.
- K nearest neighbors stores all available cases and classifies new cases based on a *similarity measure*(e.g **distance function**)
- One of the top data mining algorithms used today.
- A non-parametric lazy learning algorithm (An Instance-based Learning method).

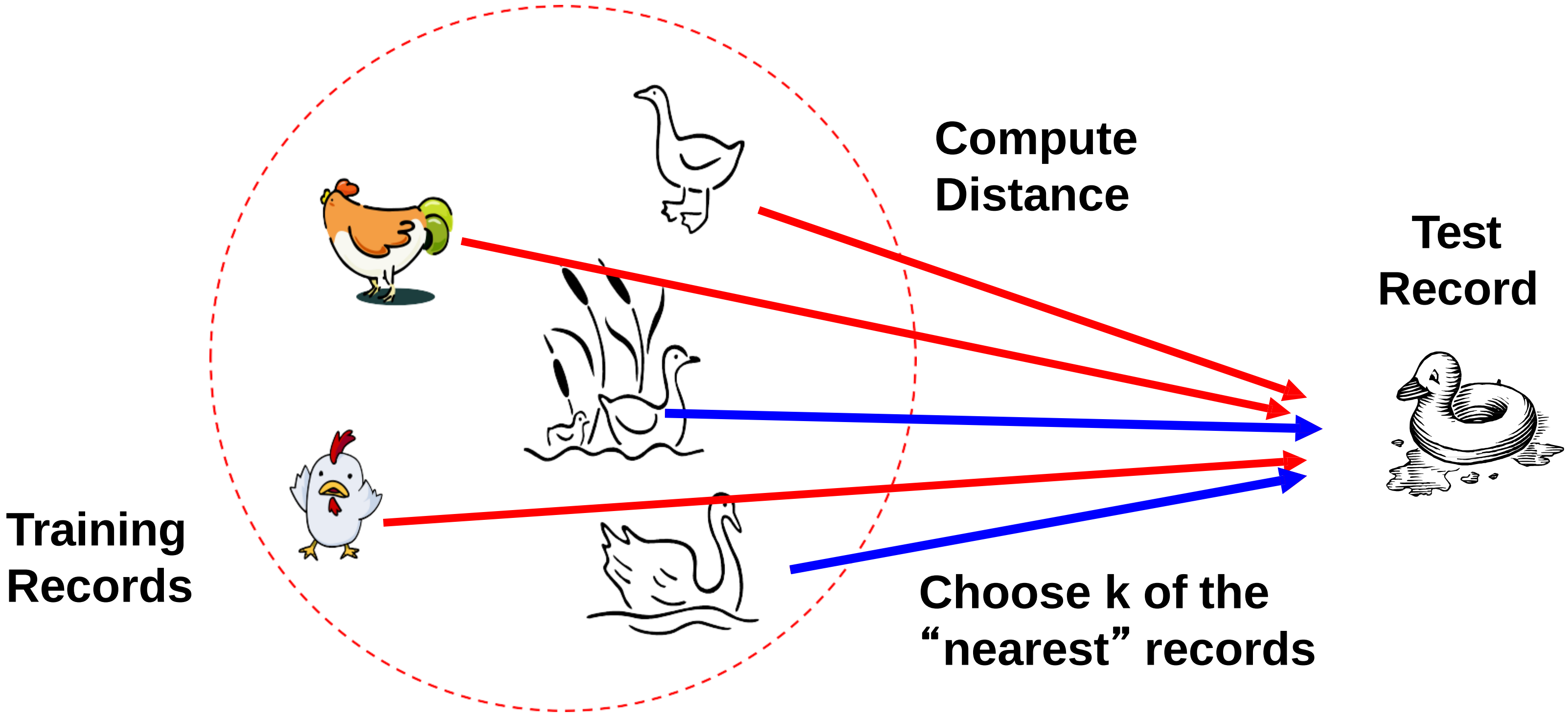
KNN: Classification Approach

- An object (a new instance) is classified by a majority votes for its neighbor classes.
- The object is assigned to the most common class amongst its K nearest neighbors. (*measured by a distant function*)



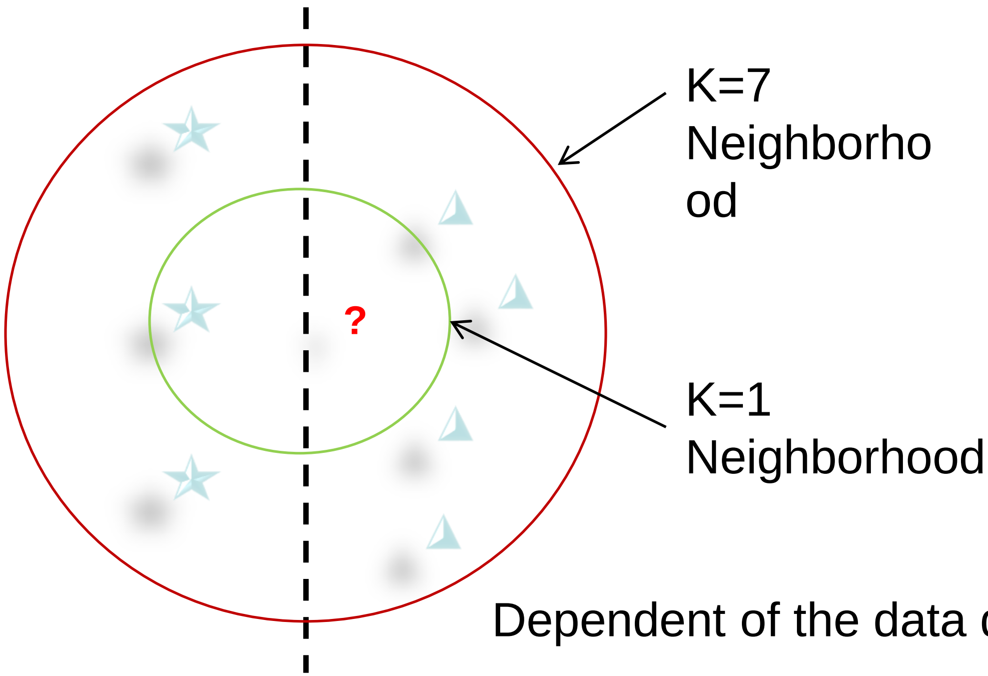


Distance Measure



3. KNN算法原理和流程

KNN算法原理和流程



Dependent of the data distributions.

Can make mistakes at boundaries.

K-Nearest Neighbors

- **工作原理**
  - 存在一个样本数据集合，也称作训练样本集，样本集中每个数据都存在标签，即我们知道样本集中每个数据和所属分类
  - 输入没有标签的新数据后，将新数据的每个特征与样本集中数据对应的特征进行比较，然后算法提取样本集中特征最相似数据（最近邻）的分类标签
  - 一般来说，只选择样本数据集中前*k*个最相似的数据。 最后，选择*k*个中出现次数最多的分类，作为新数据的分类

KNN算法的一般流程

- 收集数据：可以使用任何方法
- 准备数据：距离计算所需要的数值，最后是结构化的数据格式。
- 分析数据：可以使用任何方法
- 测试算法：计算错误率
- 使用算法：首先需要输入样本数据和结构化的输出结果，然后运行k-近邻算法判定输入数据分别属于哪个分类，最后应用对计算出的分类执行后续的处理。

距离度量

Distance functions

Euclidean

$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Manhattan

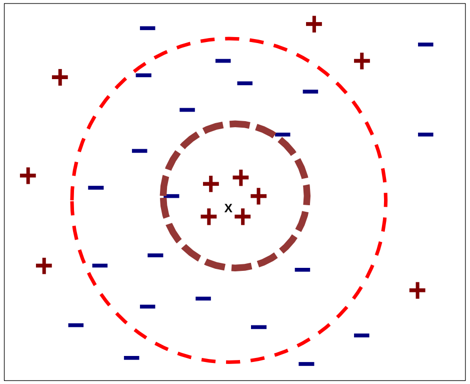
$$\sum_{i=1}^{k}|x_i - y_i|$$

Minkowski

$$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$$

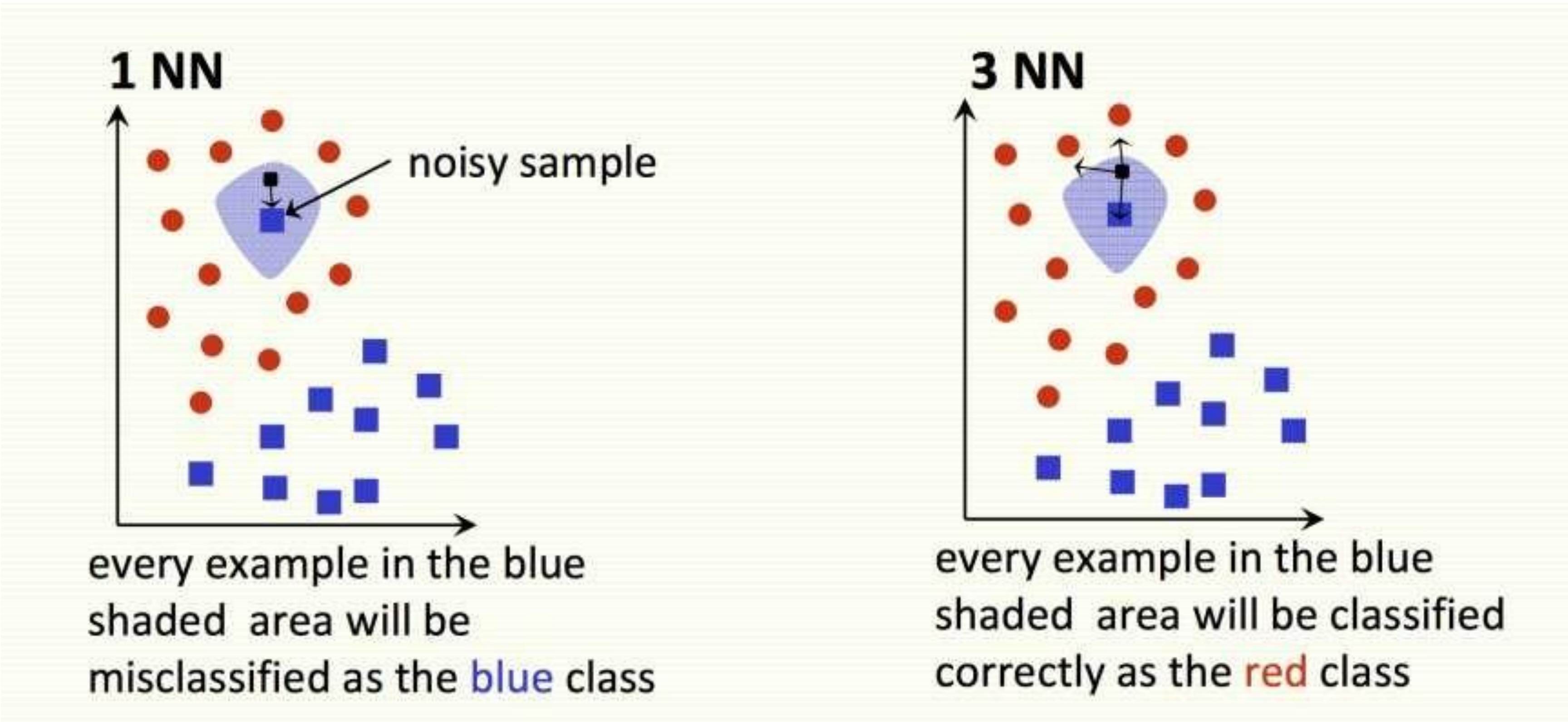
How to choose K?

- If K is too small it is sensitive to noise points.
- Larger K works well. But too large K may include majority points from other classes.

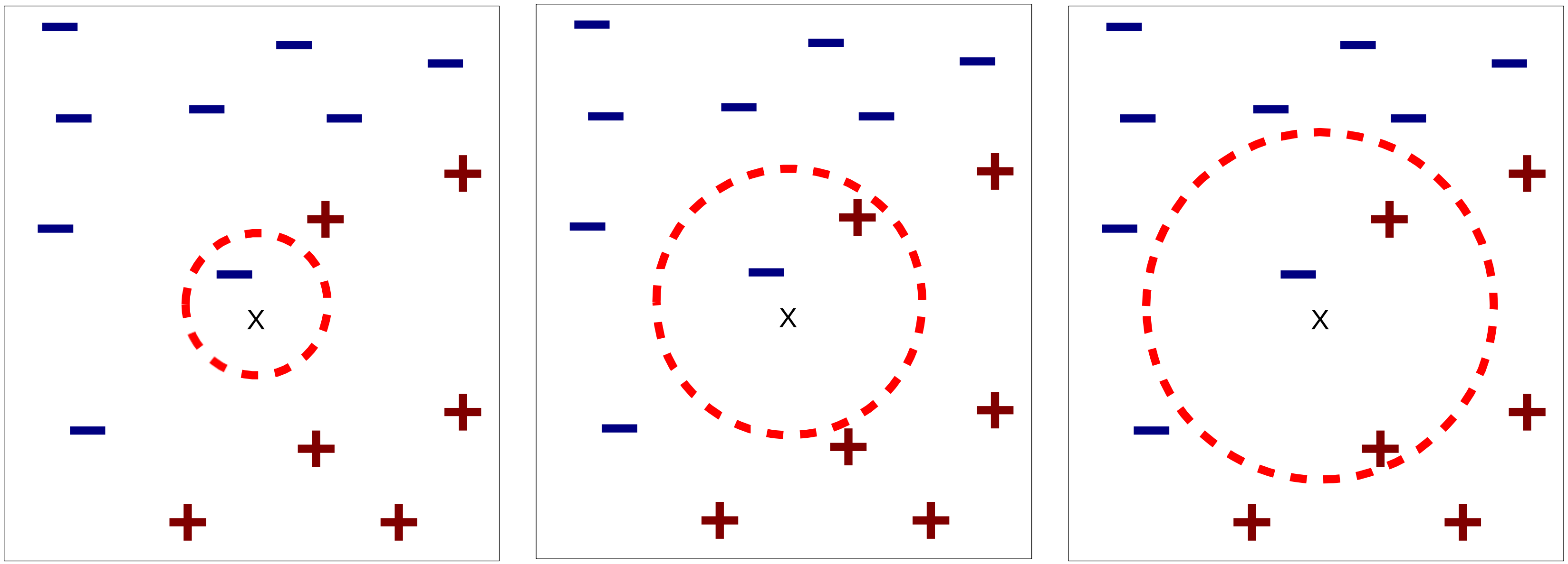


- Rule of thumb is K < sqrt(n), n is number of examples.

How to choose K?



How to choose K?



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x

KNN Feature Weighting

- Scale each feature by its importance for classification

$$D(a,b) = \sqrt{\sum_k \boxed{w_k} (a_k - b_k)^2}$$

- Can use our prior knowledge about which features are more important

Feature Normalization

- Distance between neighbors could be dominated by some attributes with relatively large numbers.
  - e.g., income of customers in our previous example.

$$a_i = \frac{v_i - \min v_i}{\max v_i - \min v_i}$$

- Arises when two features are in different scales.

- Important to normalize those features.
  - Mapping values to numbers between 0 – 1.

Nominal/Categorical Data

- Distance works naturally with numerical attributes.
- Binary value categorical data attributes can be regarded as 1 or 0.

Hamming Distance

$$D_H = \sum_{i=1}^{k} |x_i - y_i|$$

$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

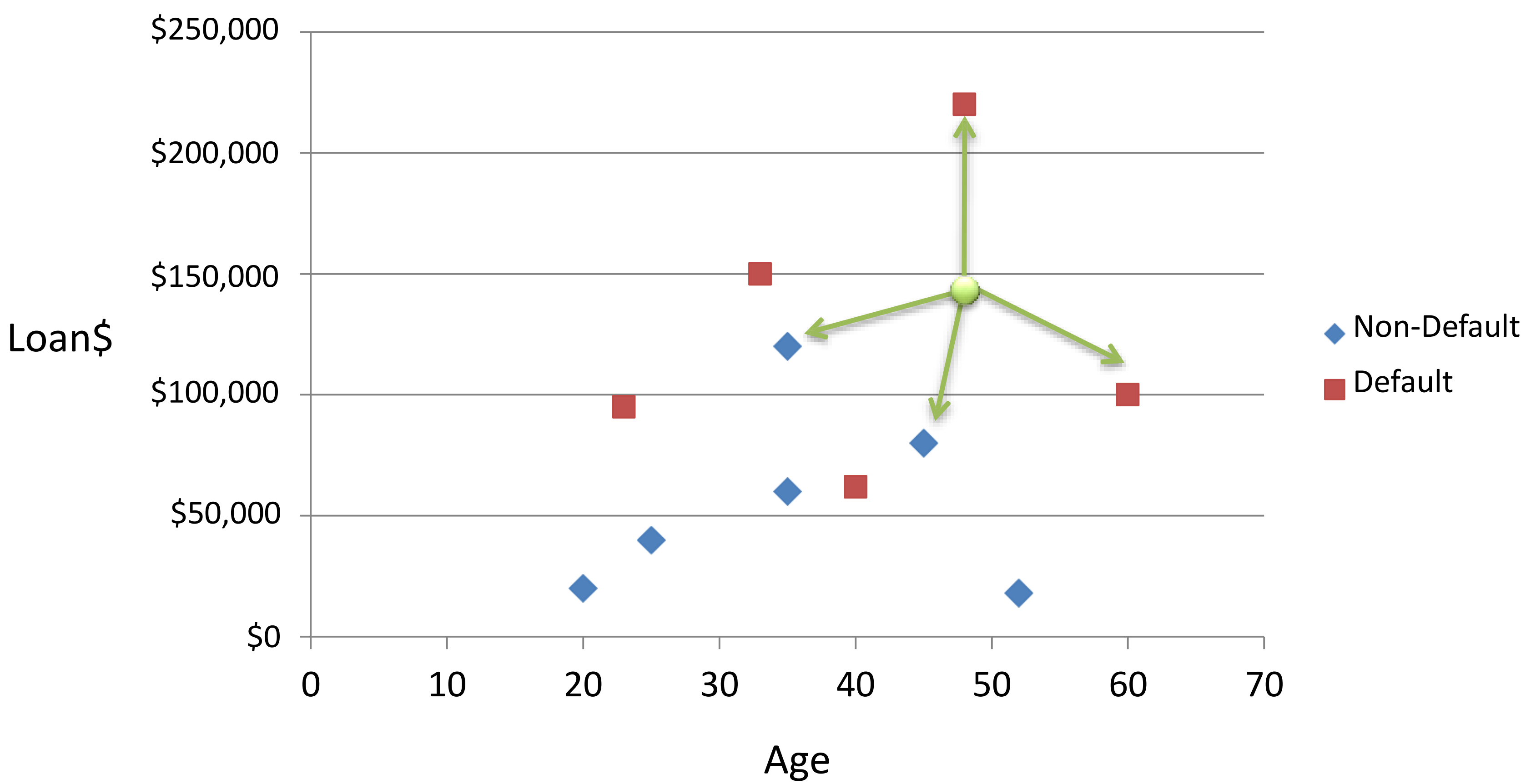
X	Y	Distance
Male	Male	0
Male	Female	1

Exercise

Exercise #1: How to measure the distance for categorical attributes with more than two values?

(Please consider from both sequential and non-sequential aspects.)

KNN Classification



K-Nearest Neighbors算法特点

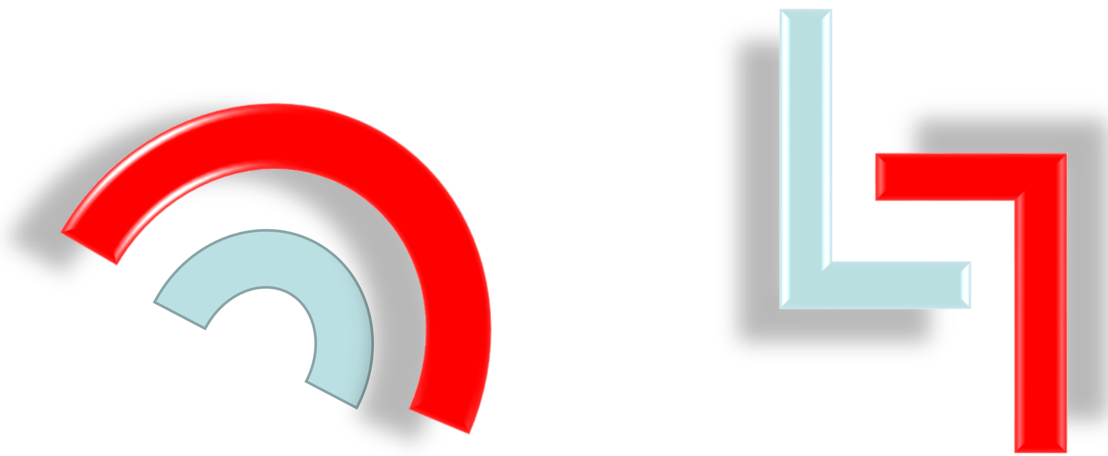
Strengths of KNN

- Very simple and intuitive.
- Can be applied to the data from any distribution.
- Good classification if the number of samples is large enough.

Weaknesses of KNN

- Takes more time to classify a new example.
  - need to calculate and compare distance from new example to all other examples.
- Choosing k may be tricky.
- Need large number of samples for accuracy.

KNN面临的挑战



Instance-Based Learning

No explicit description of the target function

Cannot handle complicated situations.

4. Python程序实现

Python分类算法流程

- 对未知类别的数据集中的每个点依次执行以下操作
  - 计算已知类别数据集众多点与当前点之间的距离
  - 按照距离递增次序排序
  - 选取与当前点距离最小的k个点
  - 群定前k个点所在类别的出现频率
  - 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

Python分类算法流程

kNN中的分类算法：

```
def classify0(inX, dataSet, labels, k):
    dataSetSize = dataSet.shape[0]
    diffMat = tile(inX, (dataSetSize,1)) - dataSet
    sqDiffMat = diffMat**2
    sqDistances = sqDiffMat.sum(axis=1)
    distances = sqDistances**0.5
    sortedDistIndicies = distances.argsort()
    classCount={}
    for item in range(k):
        votellabel = labels[sortedDistIndicies[item]]
        classCount[votellabel] = classCount.get(votellabel,0) + 1
    sortedClassCount = sorted(classCount.iteritems(), key=operator.itemgetter(1), reverse=True)
    return sortedClassCount[0][0]
```

Python分类算法流程

Shape函数

- group,labels=kNN.createDataSet()

- group.shape
- group.shape[0]

Python分类算法流程

Tile函数

- tile([1.0,1.2],(4,1))
- array([[1. , 1.2],
-         [1. , 1.2],
-         [1. , 1.2],
-         [1. , 1.2]])
- tile([1.0,1.2],(4,1))-group
- array([[0. , 0.1],
-         [0. , 0.2],
-         [1. , 1.2],
-         [1. , 1.1]])
- a=(tile([1.0,1.2],(4,1))-group)**2
- array([[0. , 0.01],
-         [0. , 0.04],
-         [1. , 1.44],
-         [1. , 1.21]])

Python分类算法流程

Argsort（）

- b=a.sum(axis=1)
- c=b**0.5
- d=c.argsort()
- >>> d
- array([0, 1, 3, 2])

Python分类算法流程

字典的使用

- classCount={}   #字典
- for i in range(k):   #列表的扩展
- votellabel = labels[sortedDistIndicies[i]]
- classCount[votellabel] = classCount.get(votellabel,0) + 1
- sortedClassCount = sorted(classCount.iteritems(), key=operator.itemgetter(1), reverse=True)
- return sortedClassCount[0][0]

- kNN.classify0([0,0.2],group,labels,3)

'B'

Python分类算法流程

从文本文件中解析数据-打开文件

- def file2matrix(filename):
-     fr = open(filename)
-     numberOfLines = len(fr.readlines())   #get the number of lines in the file
-     returnMat = zeros((numberOfLines,3))   #prepare matrix to return
-     classLabelVector = []   #prepare labels return
-     fr = open(filename)
-     index = 0
-

Python分类算法流程

从文本文件中解析数据-获得数据

- for line in fr.readlines():
- line = line.strip()
- listFromLine = line.split('\t') 截取掉所有回车符号，\t分割成列表
- returnMat[index,:] = listFromLine[0:3]
- classLabelVector.append(int(listFromLine[-1]))
- index += 1
- return returnMat,classLabelVector

Python分类算法流程

使用Matplotlib创建散点图

- import matplotlib
- >>> import matplotlib.pyplot as plt
- >>> fig=plt.figure()
- >>> ax=fig.add_subplot(111)
- >>> ax.scatter(datingDataMat[:,1],datingDataMat[:,2])
- <matplotlib.collections.PathCollection object at 0x01D8F590>
- >>> plt.show()

Python分类算法流程

使用Matplotlib创建散点图

- >>> fig=plt.figure()
- >>> ax=fig.add_subplot(111)
- >>>ax.scatter(datingDataMat[:,1],datingDataMat[:,2],15.0*array(datingLabels),15.0*array(datingLabels))
- >>> plt.show()

Python分类算法流程

数据归一化

```
def autoNorm(dataSet):
    minVals = dataSet.min(0)
    maxVals = dataSet.max(0)
    ranges = maxVals - minVals
    normDataSet = zeros(shape(dataSet))
    m = dataSet.shape[0]
    normDataSet = dataSet - tile(minVals, (m,1))
    normDataSet = normDataSet/tile(ranges, (m,1))   #element wise divide
    return normDataSet, ranges, minVals
```

数据归一化

- >>> n,r,m=kNN.autoNorm(datingDataMat)
- >>> n
- array([[0.44832535, 0.39805139, 0.56233353],
- [0.15873259, 0.34195467, 0.98724416],
- [0.28542943, 0.06892523, 0.47449629],
- ...,
- [0.29115949, 0.50910294, 0.51079493],
- [0.52711097, 0.43665451, 0.4290048],
- [0.47940793, 0.3768091 , 0.78571804]])
- >>> r
- array([9.12730000e+04, 2.09193490e+01, 1.69436100e+00])
- >>> m
- array([0. , 0. , 0.001156])

测试算法：验证分类器

- def datingClassTest():
-     hoRatio = 0.50   #hold out 10%
-     datingDataMat,datingLabels = file2matrix('datingTestSet2.txt')   #load data setfrom file
-     normMat, ranges, minVals = autoNorm(datingDataMat)
-     m = normMat.shape[0]
-     numTestVecs = int(m*hoRatio)
-     errorCount = 0.0
-     for i in range(numTestVecs):
-         classifierResult = classify0(normMat[i,:],normMat[numTestVecs:m,:],datingLabels[numTestVecs:m],3)
-         print "the classifier came back with: %d, the real answer is: %d" % (classifierResult, datingLabels[i])
-         if (classifierResult != datingLabels[i]): errorCount += 1.0
-     print "the total error rate is: %f" % (errorCount/float(numTestVecs))
-     print errorCount

Exercise

Exercise #2: How to quickly retrieve K nearest neighbors of a given query (sample)?

Thank you!

权小军 中山大学数据科学与计算机学院