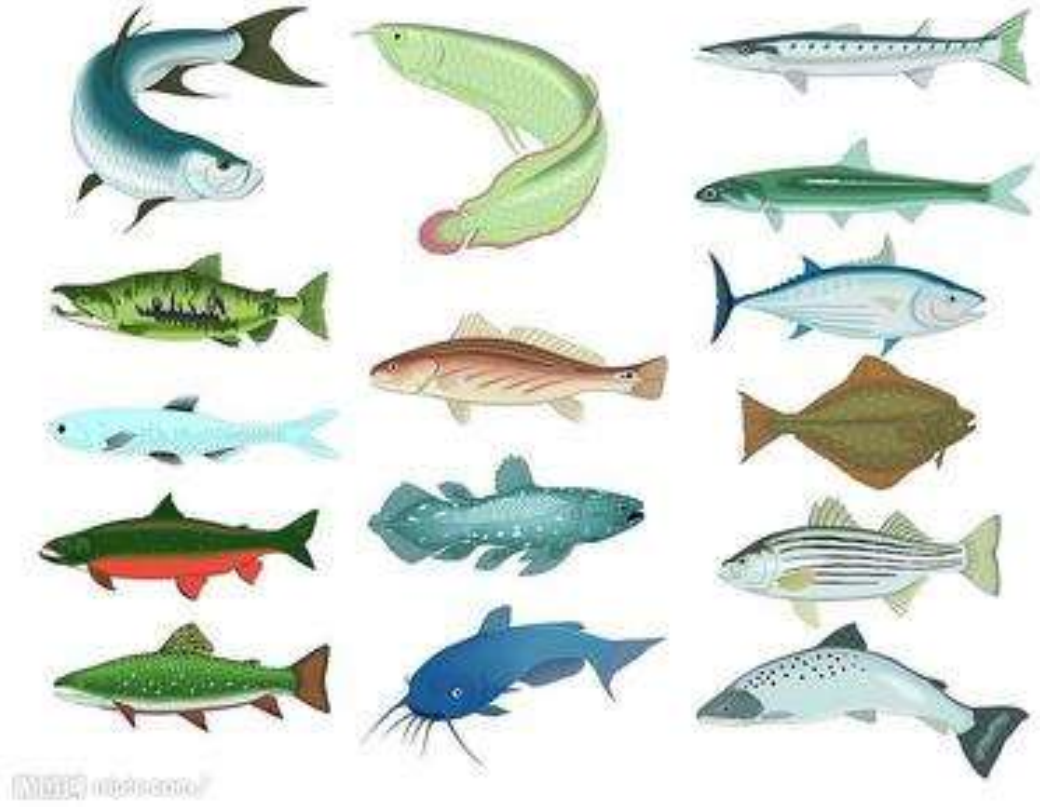
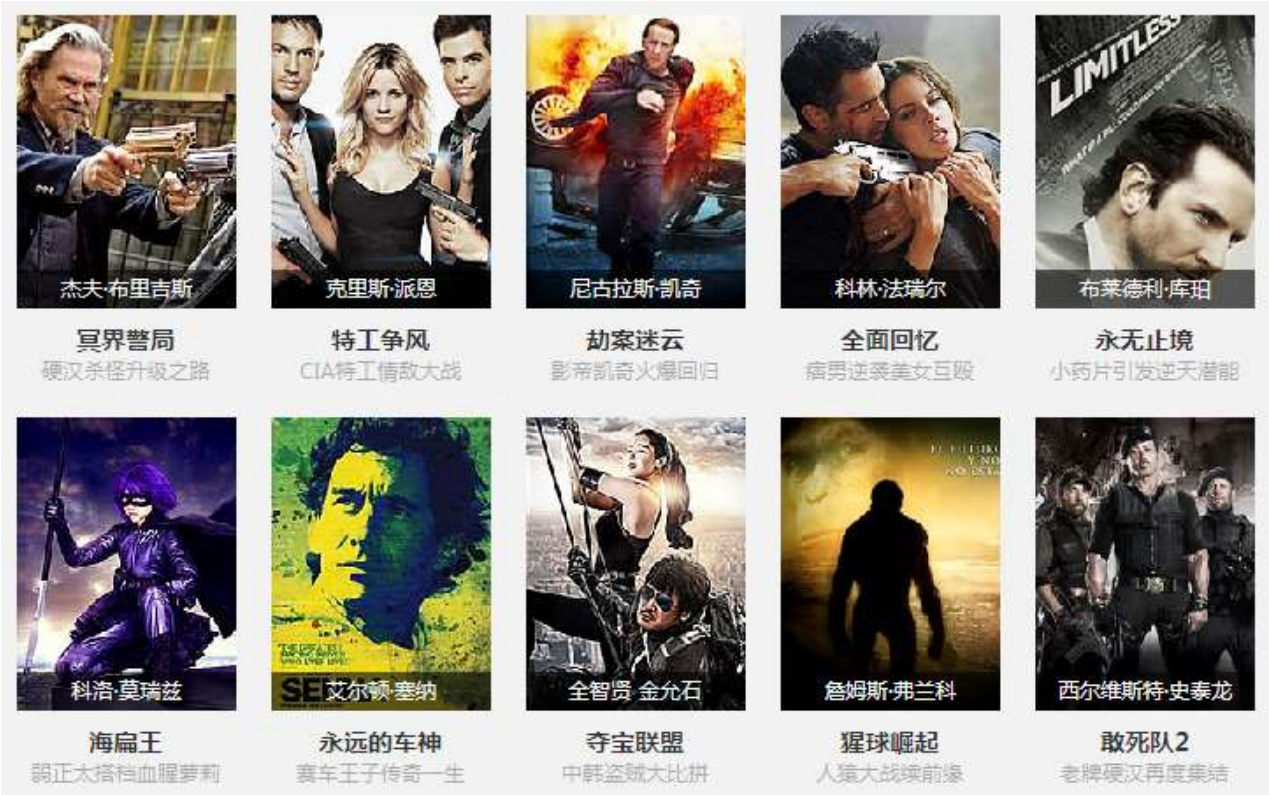
清華大學
Tsinghua University

第三章
k-近邻算法

分类问题



分类问题


杰夫·布里吉斯
克里斯·派恩
尼古拉斯·凯奇
科林·法瑞尔
LIMITLESS
布莱德利·库珀
冥界警局
硬汉杀怪升级之路
特工争风
CIA特工情敌大战
劫案迷云
影帝凯奇火爆回归
全面回忆
痞男逆袭美女互殴
永无止境
小药片引发逆天潜能
科洛·莫瑞兹
艾尔顿·塞纳
全智贤 金允石
詹姆斯·弗兰科
西尔维斯特·史泰龙
海扁王
弱正太搭档血腥萝莉
永远的车神
赛车王子传奇一生
夺宝联盟
中韩盗贼大比拼
猩球崛起
人猿大战续前缘
敢死队2
老牌硬汉再度集结

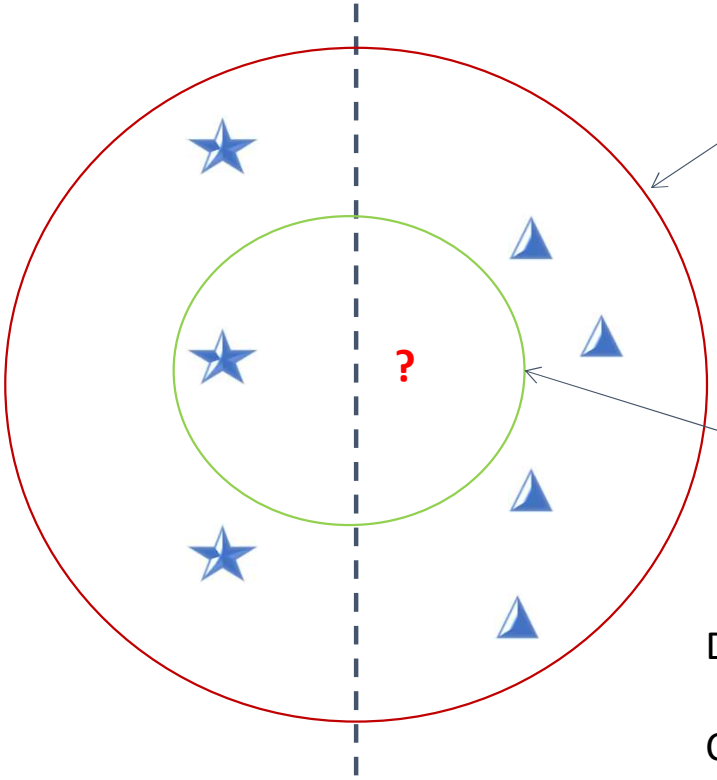
- 爱情片、剧情片、喜剧片、家庭片、伦理片、文艺片、音乐片、歌舞片、动漫片、西部片、武侠片、古装片、动作片、恐怖片、惊悚片、冒险片、犯罪片、悬疑片、记录片、战争片、历史片、传记片、体育片、科幻片、魔幻片、奇幻片

Supervised learning

提纲

- KNN算法原理和流程
- Python程序调试
  - Python文件类型
  - 模块
  - Idle调试环境
  - 数据载入
- 算法和关键函数分析
- 算法改进和实验作业

K-Nearest Neighbors算法原理



K=7 Neighborhood

K=1 Neighborhood

Dependent of the data distributions.

Can make mistakes at boundaries.

K-Nearest Neighbors算法特点

- 优点
  - 精度高
  - 对异常值不敏感
  - 无数据输入假定
- 缺点
  - 计算复杂度高
  - 空间复杂度高
- 适用数据范围
  - 数值型和标称型

K-Nearest Neighbors Algorithm

- 工作原理
  - 存在一个样本数据集合，也称作训练样本集，并且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与所属分类的对应关系。
  - 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N个最相似的数据。K一般不大于20，最后，选择k个中出现次数最多的分类，作为新数据的分类

K近邻算法的一般流程

- 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
- 训练算法：　（此步骤kNN）中不适用
- 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

$$x_i = (x_i^{(1)}, x_i^{(2)}, \cdots, x_i^{(n)})^{\mathrm{T}}$$

- Lp距离:

$$L_p(x_i, x_j) = \left(\sum_{l=1}^{n} | x_i^{(l)} - x_j^{(l)} |^p \right)^{\frac{1}{p}}$$

- 欧式距离:

$$L_2(x_i, x_j) = \left(\sum_{l=1}^{n} | x_i^{(l)} - x_j^{(l)} |^2 \right)^{\frac{1}{2}}$$

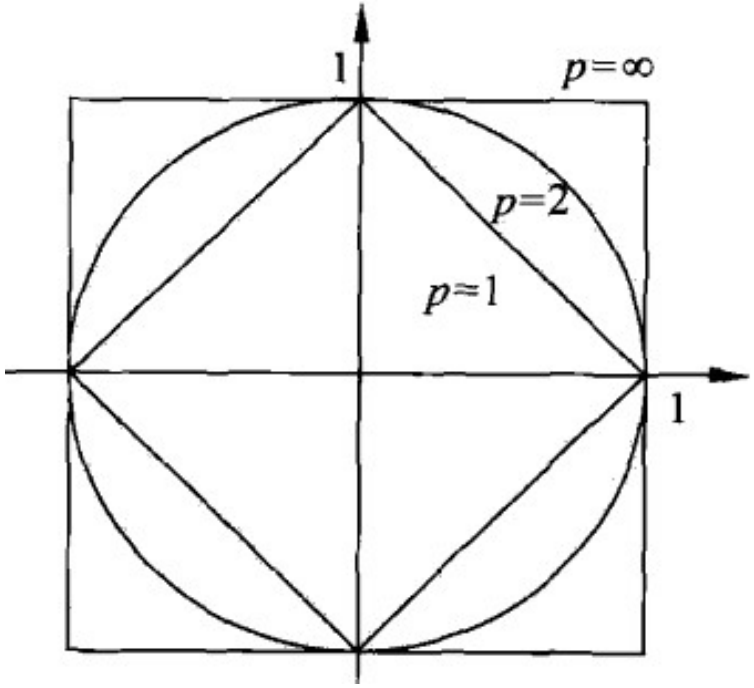
- 曼哈顿距离

$$L_1(x_i, x_j) = \sum_{l=1}^{n} | x_i^{(l)} - x_j^{(l)} |$$

- L∞距离

$$L_\infty(x_i, x_j) = \max_l | x_i^{(l)} - x_j^{(l)} |$$

距离度量



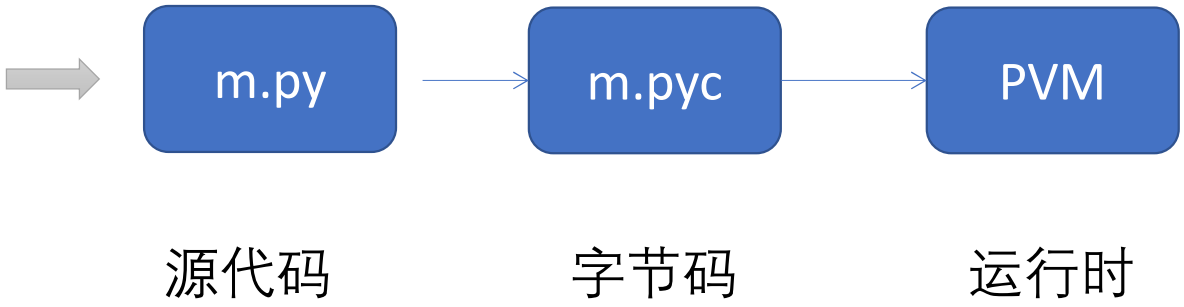
K值的选择

- 如果选择较小的K值
  - “学 习”的近似误差（approximation error)会减小，但 “学习”的估计误差（estimation error) 会增大，
  - 噪声敏感
  - K值的减小就意味着整体模型变得复杂，容易发生过 拟合.

- 如果选择较大的K值，
  - 减少学习的估计误差，但缺点是学习的近似误差会增大.
  - K值的增大 就意味着整体的模型变得简单.

Python程序调试

- Python传统运行模式：
  - Python解释器：运行Python程序的程序；
  - Python字节码：Python将程序编译后所得到的底层形式；Python自动将字节码保存为名为.pyc的文件中；



  - 录入的源码转换为字节码->字节码在PVM（Python虚拟机）中运行->代码自动被编译，之后再解释

Python程序调试

- Python无“build”和“make”的步骤，代码写好后立即运行
- Python字节码不是机器的二进制代码（so 不能像C++运行速度那么快，其速度介于传统编译语言和传统解释语言之间）

- raw_input（）的使用

Python模块

- 每一个.py文件都是一个模块，其他文件可以通过导入一个模块读取这个模块的内容，
- 相当于C中的include……
- 一个大型程序往往呈现出多模块的形式。
- 其中一个模块文件被设计为主文件（or顶层文件）。

- 模块是Python程序最大的程序结构
- 每个模块文件是一个独立完备的变量包装，即一个命名空间

模块的导入

- import，from 和 reload
  - import语句将模块作为一个整体引用，相当于引入一个类的object。
  - From 增加了对变量名的额外赋值，也就是拷贝模块的属性，因此能够以主模块导入，而不是原来的对象
- 例子：
  - A=“this”
  - B=“is”
  - C=“test”
  - Print A，B，C
  - Import test1
  - Test1.A
  - From test import *
  - A

Reload和重编译

- 修改文件如kNN注意
  - Reload（）
  - 或者：重新编译
    - import py_compile
    - py_compile.compile('D:\python\machinelearninginaction\Ch02\kNN.py')
  - 最好不要用中文，如果需要，用编码转换工具codec

Idle调试环境

- Idle 的使用：
  - Copy的结果是什么？
  - For语句
  - Reload的好处
    - 修改程序，显示修改时间
  - Import 和from A import *的关系
    - 空间，如numpy

Python导入数据

- >>> import os
- >>> os.getcwd()
- 'D:\\python'
- >>>os.chdir('D:\\python\\machinelearninginaction\\Ch02')
- >>> os.getcwd()
- 'D:\\python\\machinelearninginaction\\Ch02'
- >>> open('.\\testDigits\\0_0.txt')
- <open file '.\\testDigits\\0_0.txt', mode 'r' at 0x00D1FCD8>

Python导入数据

- from numpy import *
- >>> import operator
- >>> def createDataSet():
- 	group=array([[1.0,1.1],[1.0,1.0],[0,0],[0,0.1]])
- 	labels=['A','A','B','B']
- 	return group,lables

- >>>group,labels=kNN.createDataSet()

算法和关键函数分析

- 分类算法流程和关键函数
  - Shape
  - Tile
  - Argsort
  - 字典的使用
- 文本中解析数据
  - 文件读取相关函数
- 用matplotlib绘制散点图
- 数据归一化
- 使用k-近邻算法的手写识别系统

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

分类算法

- 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):
- voteIlabel = labels[sortedDistIndicies[item]]
- classCount[voteIlabel] = classCount.get(voteIlabel,0) + 1
- sortedClassCount = sorted(classCount.iteritems(), key=operator.itemgetter(1), reverse=True)
- return sortedClassCount[0][0]

Shape函数

- group,labels=kNN.createDataSet()

- group.shape
- group.shape[0]

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]])

Argsort（）

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

字典的使用

- classCount={}　　#字典
- 　for i in range(k):　#列表的扩展
- 　　voteIlabel = labels[sortedDistIndicies[i]]
- 　　classCount[voteIlabel] = classCount.get(voteIlabel,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'

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
```

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

文件读取相关函数

- Open()的使用
- Readlines的使用
- Zeros()的使用

文件读取相关函数

- Open()的使用
  - 工作路径和绝对路径
- Readlines的使用
- Zeros()的使用
- fr=open('test1.txt')
- >>> for line in fr.readlines():
- print line
- 执行a=fr.readlines()
- >>> a
- 结果是什么呢?
- >>> fr.seek(0)
- >>> a=fr.readlines()

文件读取相关函数

- >>> a=fr.readlines()
- >>> a
- ['1　3　4　12\n', '5　7　8　13\n', '9　10　11　14']
- >>> b=a[0]
- >>> b
- '1　3　4　12\n'
- >>> c=b.strip()
- >>> c
- '1　3　4　12'
- >>> d=c.split('\t')
- >>> d
- ['1　3　4　12']
- >>> d[0]
- '1　3　4　12'
- >>>

文件读取相关函数

- for line in a:
- 	line=line.strip()
- 	line=line.split()
- 	print line

-
- ['1', '3', '4', '12']
- ['5', '7', '8', '13']
- ['9', '10', '11', '14']

文件读取相关函数

- for line in a:
- 	line=line.strip()
- 	line=line.split()
- 	print line

-
- ['1', '3', '4', '12']
- ['5', '7', '8', '13']
- ['9', '10', '11', '14']

文件读取相关函数

- for line in a:
- 	line=line.strip()
- 	line=line.split()
- 	line=[int(x) for x in line]
- 	print line

-
- [1, 3, 4, 12]
- [5, 7, 8, 13]
- [9, 10, 11, 14]

数组和矩阵

- Python 数组和numpy矩阵的关系
- >>> a=[[1,2,3,4],[5,6,7,8],[9,10,11,12]]
- >>> c=zeros((3,4))
- >>> c
- array([[0., 0., 0., 0.],
- [0., 0., 0., 0.],
- [0., 0., 0., 0.]])
- >>> c[0,:]=a[0]
- >>> c
- array([[1., 2., 3., 4.],
- [0., 0., 0., 0.],
- [0., 0., 0., 0.]])

解析数据

- fr=open('datingTestSet.txt')
- >>> a=fr.readlines()
- b=len(a)
- line=a[0]
- line=line.strip()
- list=line.split('\t')
- >>> r[0,:]=list[0:3]
- r
- List[-1]
- ClassLat=[]
- classLab.append(list[-1])
- >>> classLab

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()

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()

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
```

使用k-近邻算法的手写识别系统

准备数据，将图像转换为测试向量 32x32

```
def img2vector(filename):
    returnVect = zeros((1,1024))
    fr = open(filename)
    for i in range(32):
        lineStr = fr.readline()
        for j in range(32):
            returnVect[0,32*i+j] = int(lineStr[j])
    return returnVect
```

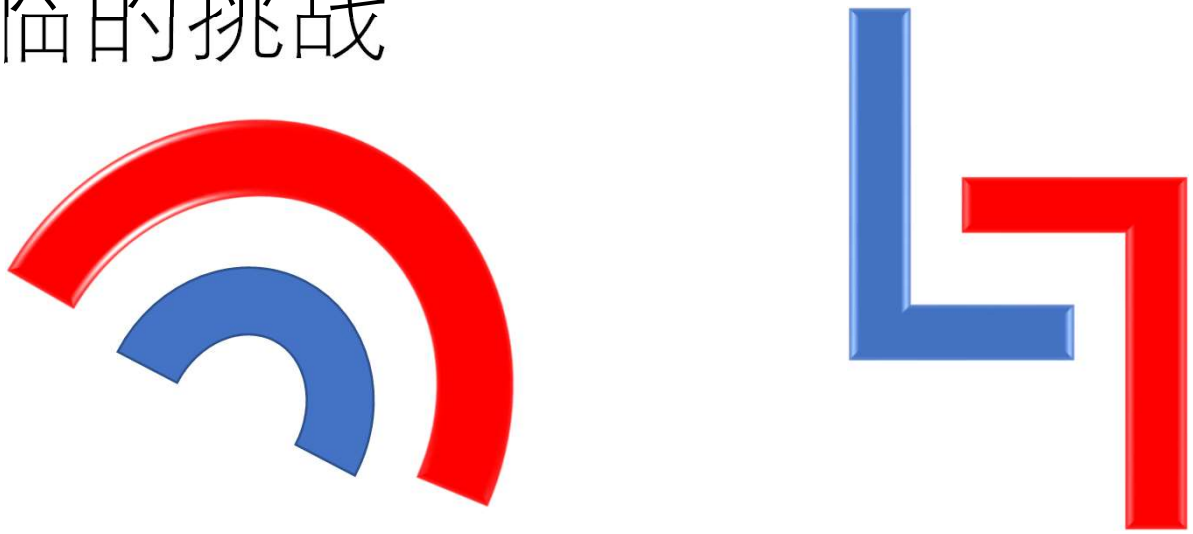
测试算法

- >>> testVector=kNN.img2vector('testDigits/0_13.txt')
- >>> tesVector[0,0:31]

KNN算法改进和实验作业

- KNN面临的挑战
- 算法改进
  - 距离度量
    - 马氏距离
  - KD树
- 实验要求

KNN面临的挑战



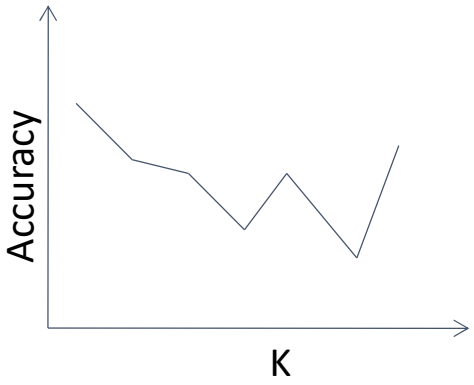
Instance-Based Learning

No explicit description of the target function

Can handle complicated situations.

KNN面临挑战

- K值确定
  - Non-monotonous impact on accuracy
  - Too Big vs. Too Small
  - Rule of thumbs
- 特征的选择
  - Different features may have different impact …
- 距离函数确定
  - There are many different ways to measure the distance.
  - Euclidean, Manhattan …
- 复杂度
  - Need to calculate the distance between X’ and all training data.
  - In proportion to the size of the training data.

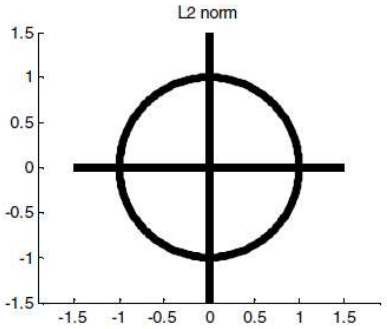


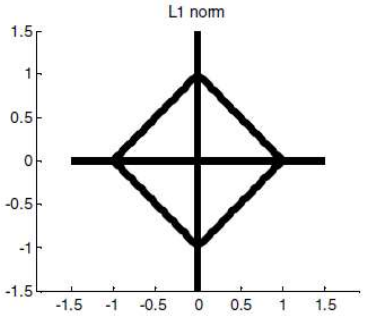
Distance Metrics

$$L_k(x,y)=\left(\sum_{i=1}^{d}|x_i-y_i|^k\right)^{1/k}$$

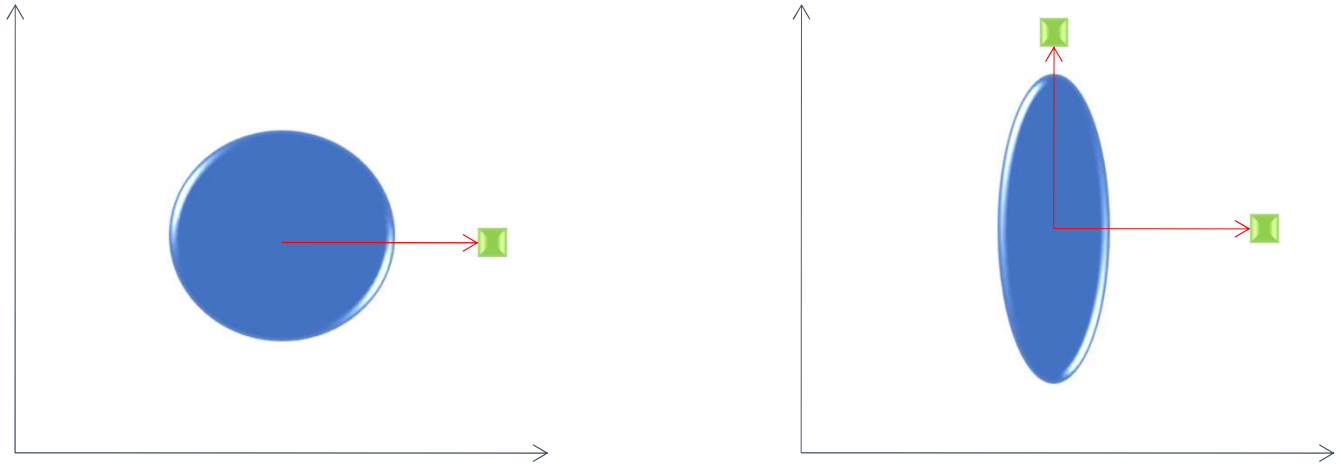
$$L_2(x,y)=\left(\sum_{i=1}^{d}|x_i-y_i|^2\right)^{1/2}$$

$$L_1(x,y)=\left(\sum_{i=1}^{d}|x_i-y_i|\right)$$





Mahalanobis Distance马氏距离



Distance from a point to a point set

马氏距离(Mahalanobis Distance)

- 由**P.C. Mahalanobis**提出
- 基于样本分布的一种距离测量
- 考虑到各种特性之间的联系（例如身高和体重），可以消除样本间的相关性
- 广泛用于分类和聚类分析

马氏距离(Mahalanobis Distance)

一维数据

均值：$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n}$

标准差：$S = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n-1}}$

方差：$S^2 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n-1}$

例：[0, 8, 12, 20]　8.3

[8, 9, 11, 12]　1.8

多维向量

$$var(X) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(X_i - \bar{X})}{n-1}$$

$$cov(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{n-1}$$

1、$cov(X,X) = var(X)$

2、$cov(X,Y) = cov(Y,X)$

协方差矩阵

$$C = \begin{pmatrix} \mathrm{cov}(x,x) & \mathrm{cov}(x,y) & \mathrm{cov}(x,z) \\ \mathrm{cov}(y,x) & \mathrm{cov}(y,y) & \mathrm{cov}(y,z) \\ \mathrm{cov}(z,x) & \mathrm{cov}(z,y) & \mathrm{cov}(z,z) \end{pmatrix}$$

$$C_{n\times n} = (c_{i,j}, \qquad c_{i,j} = \mathrm{cov}(\mathrm{Dim}_i, \mathrm{Dim}_j))$$

协方差矩阵

```
>> MySample=fix(rand(10,3)*50)

MySample =

    49     7    29
     8    19    16
    12     8    14
    19    37    22
     3    43    21
    34    17    17
    20    34    27
    49    14    37
    20    26    21
    31    41    21
```

```
>> dim1=MySample(:,1);
>> dim2=MySample(:,2);
>> dim3=MySample(:,3);
```

```
>> cov12=sum((dim1-mean(dim1)).*(dim2-mean(dim2)))/(size(MySample,1)-1);
>> cov13=sum((dim1-mean(dim1)).*(dim3-mean(dim3)))/(size(MySample,1)-1);
>> cov23=sum((dim2-mean(dim2)).*(dim3-mean(dim3)))/(size(MySample,1)-1);
```

```
>> var1=std(dim1)^2;
>> var2=std(dim2)^2;
>> var3=std(dim3)^2;
```

```
>> cov(MySample)

ans =

  254.9444  -96.5556   76.3889
  -96.5556  182.0444   -7.4444
   76.3889   -7.4444   47.1667
```

马氏距离 定义

一组向量$\{\vec{X}_1, \vec{X}_2, \vec{X}_3, \dots, \vec{X}_n\}$，其中，

$$\vec{X} = \{x_1, x_2, x_3, \dots, x_m\}$$

其均值为$\vec{\mu} = \{\mu_1, \mu_2, \mu_3, \dots, \mu_m\}$;

协方差矩阵为Σ，其中

$$\Sigma_{ij} = cov(x_i, x_j)$$

马氏距离 定义续

单向量的马氏距离定义为：

$$\mathbf{MD}(\vec{X}) = \sqrt{(\vec{X} - \vec{\mu})^T \boldsymbol{\Sigma}^{-1} (\vec{X} - \vec{\mu})}$$

向量间的马氏距离定义为：

$$\mathbf{MD}(\vec{X}, \vec{Y}) = \sqrt{(\vec{X} - \vec{Y})^T \boldsymbol{\Sigma}^{-1} (\vec{X} - \vec{Y})}$$

马氏距离 计算示例

一组向量：{3,4},{5,6},{2,2},{8,4}

均值： $\vec{\mu}$={4.5,4}

协方差矩阵：

$$\Sigma = \begin{bmatrix} 7 & 2 \\ 2 & 2.667 \end{bmatrix}, \Sigma^{-1} = \begin{bmatrix} 0.18 & -0.13 \\ -0.13 & 0.48 \end{bmatrix}$$

可以计算{3,4}和{5,6}之间的距离为：

$$\mathrm{MD} = \sqrt{(-2,-2)^T \Sigma^{-1} (-2,-2)} = 1.2$$

马氏距离 NUMPY示例

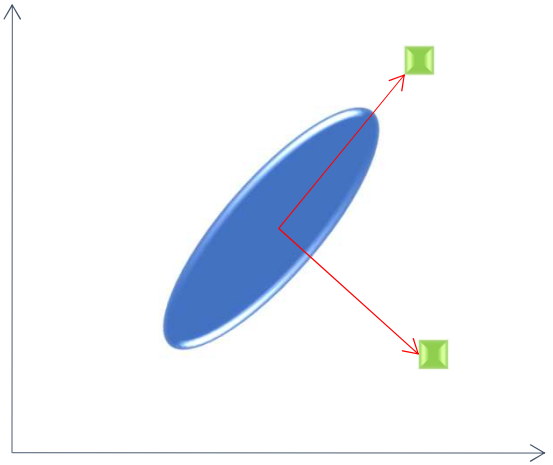
```
import numpy
x = numpy.array([[3,4],[5,6],[2,2],[8,4]])
xT = x.T
D = numpy.cov(xT)
invD = numpy.linalg.inv(D)
tp = x[0] – x[1]
print numpy.sqrt(dot(dot(tp, invD), tp.T))
```

Mahalanobis Distance

$$D_M(x) = \sqrt{(x-\mu)^T S^{-1}(x-\mu)}$$

For identity matrix S:

$$D_M(x) = \sqrt{(x-\mu)^T (x-\mu)}$$



For diagonal matrix S:

$$D_M(x) = \sqrt{\sum_{i=1}^{n} \frac{(x_i-\mu_i)^2}{\sigma_i^2}}$$

$$f_X(x) = \frac{1}{(2\pi)^{k/2}|\Sigma|^{1/2}} \exp\left(-\tfrac{1}{2}(x-\mu)'\Sigma^{-1}(x-\mu)\right)$$

KD树

- kd树是一种对K维空间中的实例点进行存储以便对其进行快速检索的树形数据结构.
- Kd树是二叉树，表示对K维空间的一个划分（partition).构造Kd树相 当于不断地用垂直于坐标轴的超平面将k维空间切分，构成一系列的k维超矩形区域.Kd树的每个结点对应于一个k维超矩形区域.

KD树

- 构造kd树:
- 对深度为j的节点，选择x^l为切分的坐标轴 $l = j(\mathrm{mod}\, k) + 1$
- 例: $T = \{(2,3)^T, (5,4)^T, (9,6)^T, (4,7)^T, (8,1)^T, (7,2)^T\}$

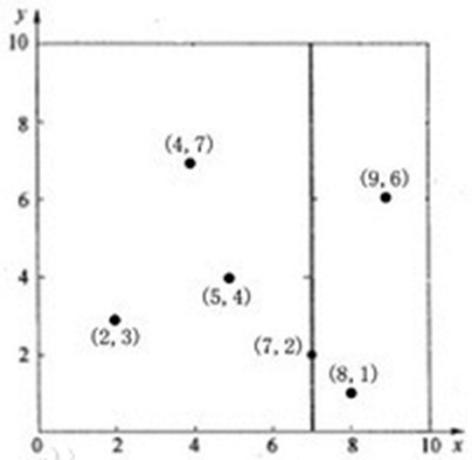


图2 x=7将整个空间分为两部分

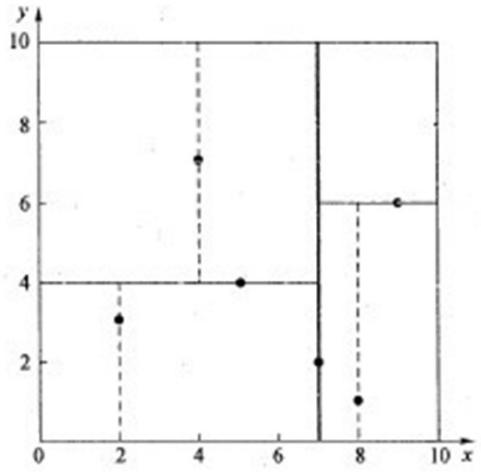
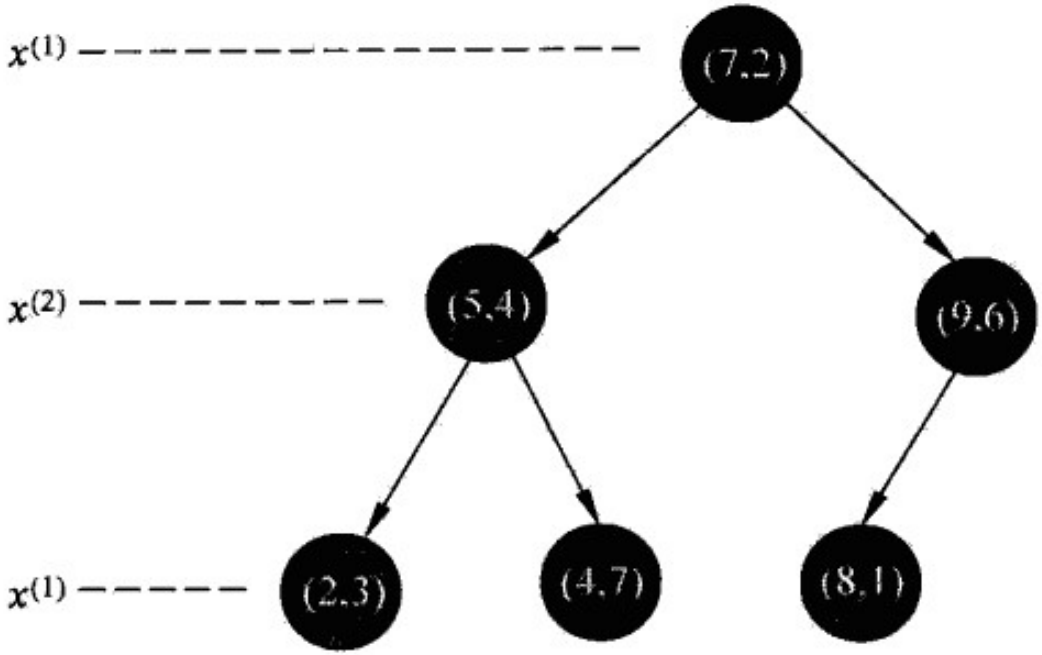


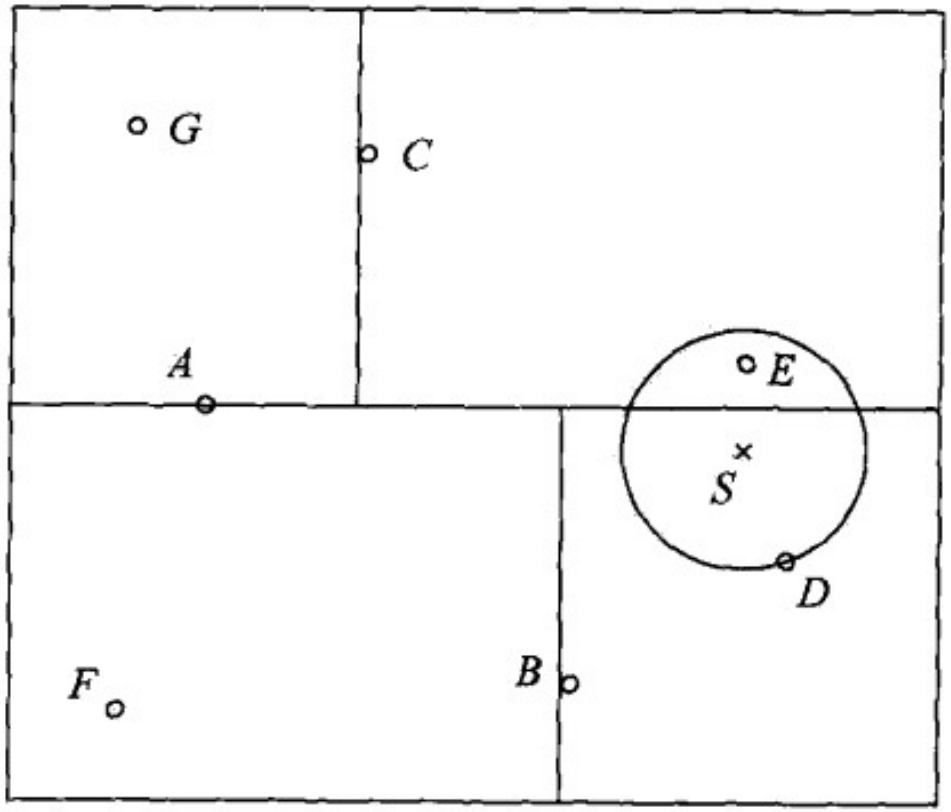
图1 二维数据k-d树空间划分示意图

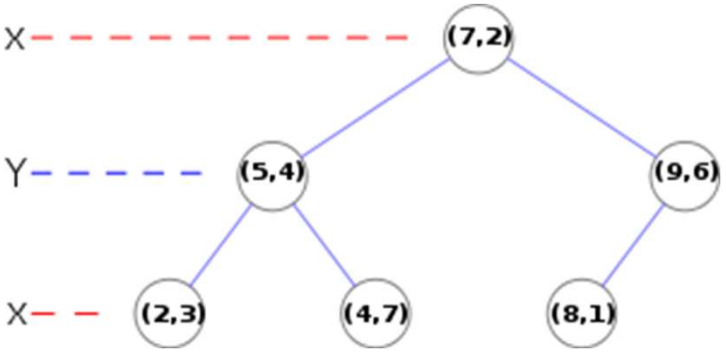
KD 树

- {(2,3),(5,4),(9,6),(4,7),(8,1),(7,2)},
- 建立索引



KD树搜索





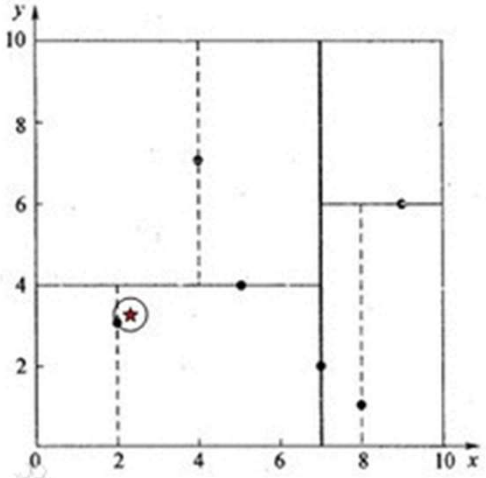


图4 查找（2.1，3.1）点的两次回溯判断

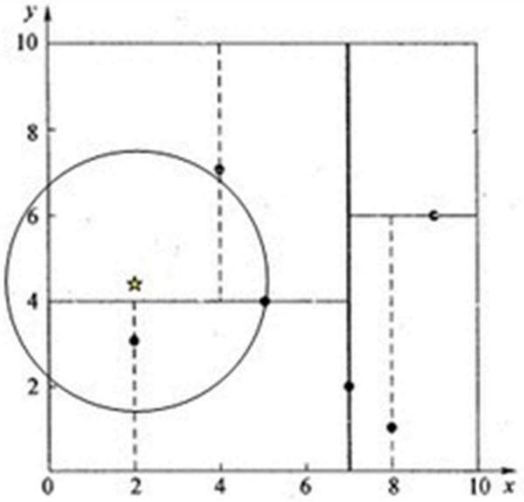


图5 查找（2，4.5）点的第一次回溯判断

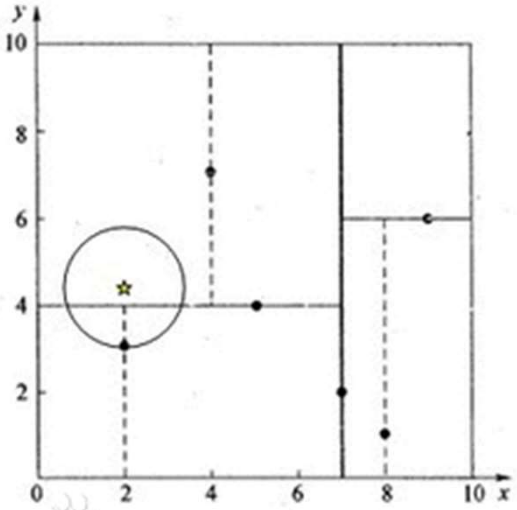


图6 查找（2，4.5）点的第二次回溯判断

作业要求

- 作业里包含如下文件:
  - 源代码（改进部分要有注释），
  - 实验报告：包含对改进算法的描述以及实验结果对比情况。
- 给分依据:
  - 代码质量（包含可读性），
  - 改进算法难度
  - 实验报告的详细性。

- 若有疑惑，可联系助教邱鑫。

- Q&A?

阅读材料

- 论文:
  - P. Viswanath and T. Hitendra Sarma,An Improvement to k-Nearest Neighbor Classifier 2011
- The key points of the paper
  - (i)weighted k-nearest neighbor classifier(wk-NNC),
    - where a weight for each training pattern is assigned and is used in the classification,
  - (ii) Bootstrapping method
    - generating an artificial training set by applying a bootstrapping method and to use the bootstrapped training set in the place of the original training set.

阅读材料

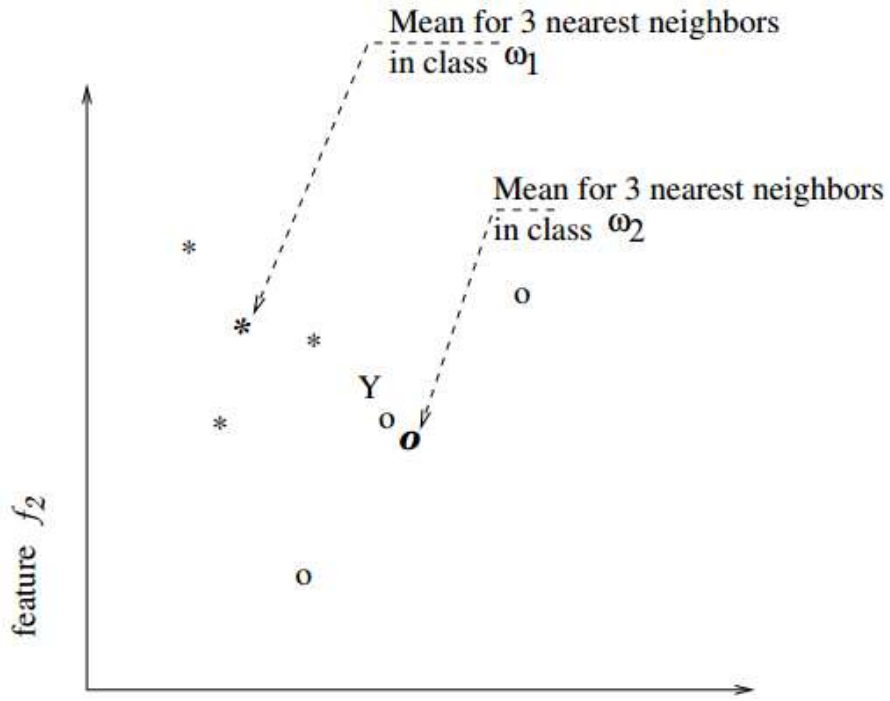
- NOTATION AND DEFINITIONS
  - 认识论文中的标注
- wk-NNC 方法
  - 例如：k=1，ω2,ω1,ω2,ω2,ω2,ω3,ω3.

    wi = (hk − hi)/(hk − h1)
- Bootstrapping 方法
  -

average of these neighbors. Let X be a training pattern and let $X^1, \ldots, X^r$ be its r nearest neighbors in its class. Then $X' = \sum_{i=1}^{r} w_i X^i$ (where $\sum_{i=1}^{r} w_i = 1$) is the bootstrapped pattern generated for X. Either all patterns in the original

阅读材料

- 算法结果


Mean for 3 nearest neighbors
in class ω_1
Mean for 3 nearest neighbors
in class ω_2
Y
feature f_2