人工智能的数学基础

华东师范大学 数学科学学院 黎芳(教授) 2019年9月18日

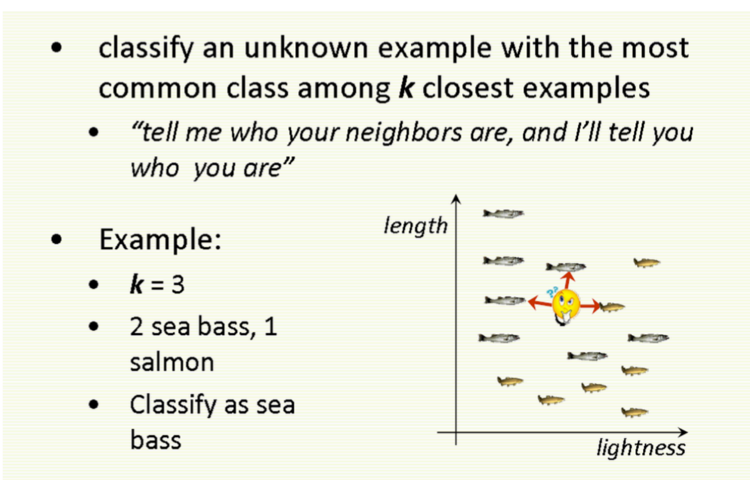
Chapter 3 k近邻法（k-nearest neighbor -- kNN）

Table of Contents

Chapter 3 k近邻法（k-nearest neighbor -- kNN） 1
3.1 k近邻算法 1
3.2 k近邻模型 2
3.2.1 模型 2
3.2.2 距离度量 3
3.2.3 k值的选取 3
3.2.3 分类决策规则 4
3.3 k近邻法的实现：kd树（k-dimensional tree） 7
3.3.1 构造kd树 7
3.3.2 搜索kd树 10
作业 11

3.1 k近邻算法

kNN classifier － the simplest classifier on earth



算法 3.1（k 近邻法）

输入：训练数据集

$$T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$$

其中，$x_i\in\mathcal{X}\subseteq\mathbf{R}^n$ 为实例的特征向量，$y_i\in\mathcal{Y}=\{c_1,c_2,\cdots,c_K\}$ 为实例的类别，$i=1,2,\cdots,N$；实例特征向量 x；

输出：实例 x 所属的类 y．

（1）根据给定的距离度量，在训练集 T 中找出与 x 最邻近的 k 个点，涵盖这 k 个点的 x 的邻域记作 $N_k(x)$；

（2）在 $N_k(x)$ 中根据分类决策规则（如多数表决）决定 x 的类别 y：

$$y=\arg\max_{c_j}\sum_{x_i\in N_k(x)}I(y_i=c_j),\quad i=1,2,\cdots,N;\quad j=1,2,\cdots,K \tag{3.1}$$

式 (3.1) 中，I 为指示函数，即当 $y_i=c_j$ 时 I 为 1，否则 I 为 0．

k=1, 最近邻法

3.2 k近邻模型

3.2.1 模型

k近邻法中，当训练集、距离度量（如欧氏距离）、K值及分类决策规则（如多数表决）确定后，对于任何一个新的输入实例，它所属的类唯一地确定．这相当于根据上述要素将特征空间划分为一些子空间，确定子空间里的每个点所属的类.

特征空间中，对每个训练实例点x_i，距离该点比其他点更近的所有点组成一 个区域，叫作单元(cell). 每个训练实例点拥有一个单元，所有训练实例点的单元构成对特征空间的一个划分．

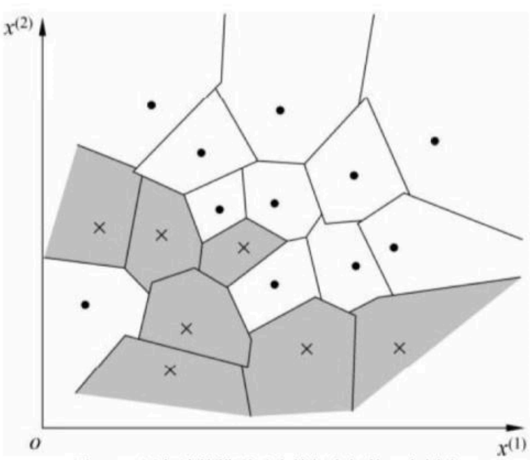
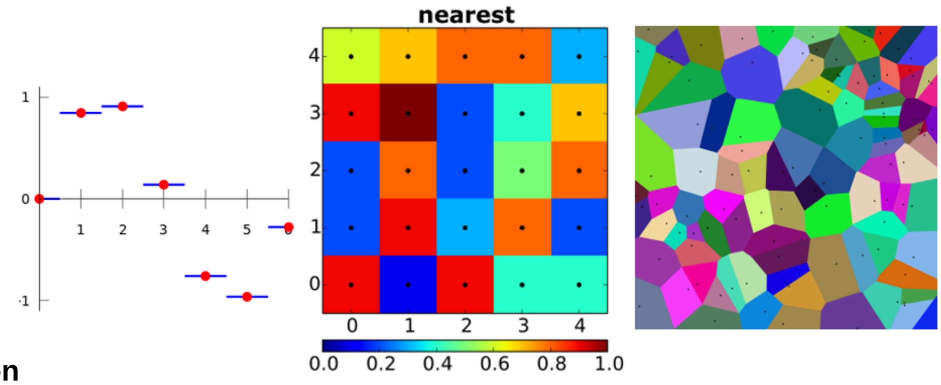


图3.1 k近邻法的模型对应特征空间的一个划分



Nearest-neighbor interpolation

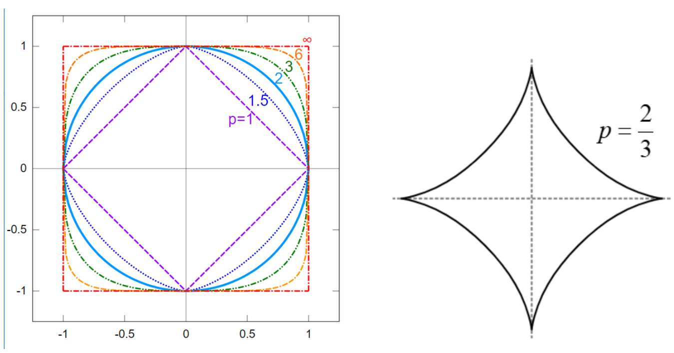
3.2.2 距离度量

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

$p = 2$，欧氏距离（Euclidean distance）：

$p = 1$，曼哈顿距离（Manhattan distance）：

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



Lp距离下的单位球

https://en.wikipedia.org/wiki/File:Lp_space_animation.gif

例**3.1** $x_1 = (1, 1)^T, x_2 = (5, 1)^T, x_3 = (4, 4)^T$，试求p取不同值时，$x_1$的最近邻点.

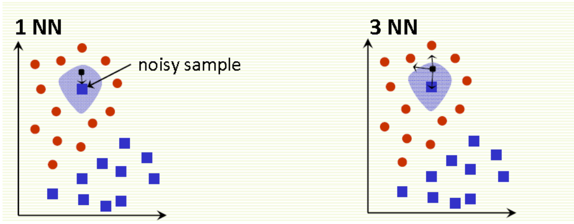
$L_p(x_1, x_2) = 4, \quad L_1(x_1, x_3) = 6, \quad L_2(x_1, x_3) = 4.24, \quad L_3(x_1, x_3) = 3.78, \quad L_4(x_1, x_3) = 3.57$

p=1,2时，最近邻点是x_2，p=3,4，最近邻点是x_3.

3.2.3 k值的选取

如果选择较小的K值

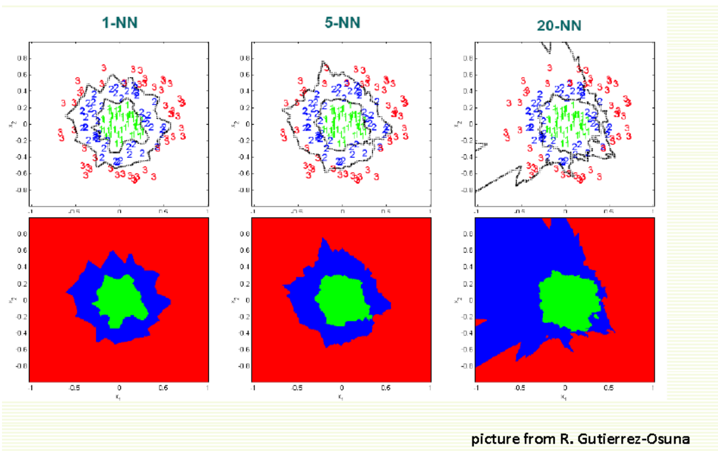
- “学习”的近似误差（approximation error）会减小，但“学习”的估计误差（estimation error）会增大

- 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.

<u>如果选择较大的K值</u>

- 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.



3.2.3 分类决策规则

多数表决规则（经验风险最小化）

分类函数：$f:\mathbf{R}^n \to \{c_1, c_2, \cdots, c_K\}$

误分类概率：$P(Y \neq f(X)) = 1 - P(Y = f(X))$

对给定的实例$x \in \mathcal{X}$,其最近邻的k个训练实例点构成集合$N_k(x)$. 如果涵盖$N_k(x)$的区域类别是c_j 那么误分类率是

$$\frac{1}{k}\sum_{x_i \in N_k(x)} I(y_i \neq c_j) = 1 - \frac{1}{k}\sum_{x_i \in N_k(x)} I(y_i = c_j)$$

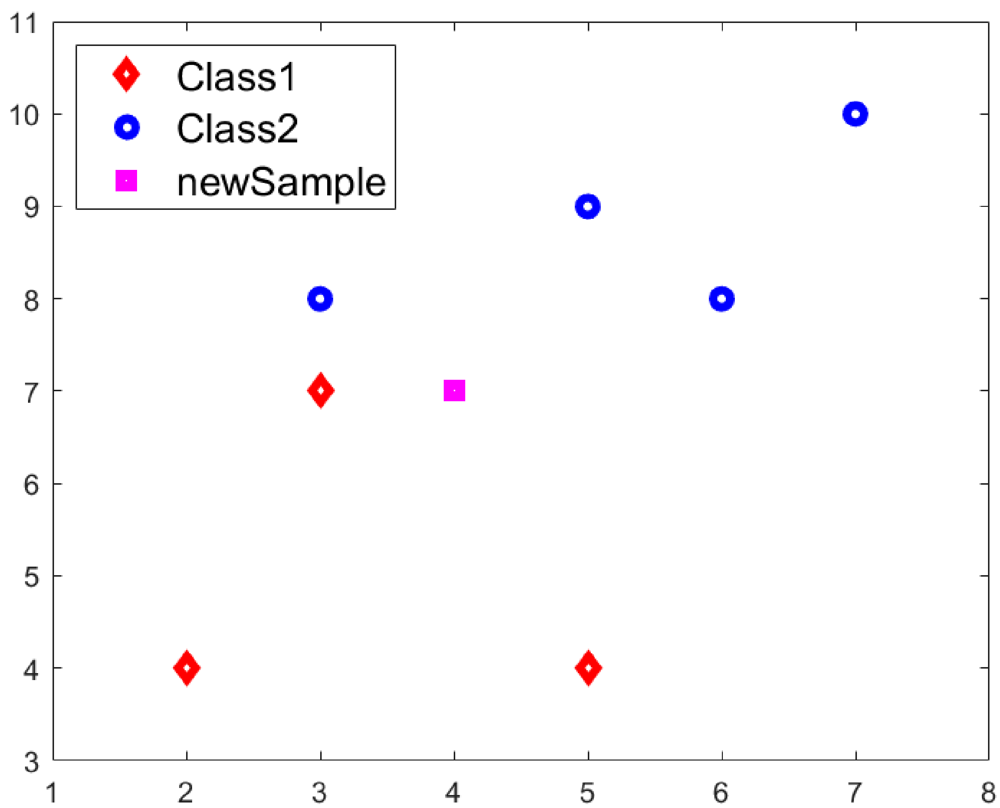
例子：

$$\text{class1} = \begin{bmatrix} 2 & 4 \\ 3 & 7 \\ 5 & 4 \end{bmatrix}, \text{class2} = \begin{bmatrix} 3 & 8 \\ 5 & 9 \\ 7 & 10 \\ 6 & 8 \end{bmatrix}$$

Want to classify $[4 \;\; 7]$

Show the data

```
Class1 = [2 4; 3 7; 5 4];
Class2 = [3 8; 5 9; 7 10; 6 8];
newSample = [4,7];
plot(Class1(:,1),Class1(:,2),'rd','LineWidth',3)
hold on;
plot(Class2(:,1),Class2(:,2),'bo','LineWidth',3)
plot(newSample(:,1),newSample(:,2),'ms','LineWidth',5)
axis([1 8 3 11])
legend('Class1', 'Class2','newSample','fontsize',14,'Location','northwest')
```




```
k = 1; % k nearest neighbour
numClass1 = size(Class1,1);
numClass2 = size(Class2,1);
totalSamples = numClass1+numClass2;
combinedSamples = [Class1;Class2]
```

```
combinedSamples = 7×2
     2     4
     3     7
     5     4
     3     8
     5     9
     7    10
```

```
     6     8
```

```
trueClass = [zeros(numClass1,1)+1;zeros(numClass2,1)+2;]
```

```
trueClass = 7×1
     1
     1
     1
     2
     2
     2
     2
```

```
absDiff = abs(combinedSamples-newSample).^2
```

```
absDiff = 7×2
     4     9
     1     0
     1     9
     1     1
     1     4
     9     9
     4     1
```

```
dist = sum(absDiff,2)
```

```
dist = 7×1
    13
     1
    10
     2
     5
    18
     5
```

```
[Y,I] = sort(dist)
```

```
Y = 7×1
     1
     2
     5
     5
    10
    13
    18
I = 7×1
     2
     4
     5
     7
     3
     1
     6
```

```
neighborsInd = I(1:k)
```

```
neighborsInd = 2
```

```
neighbors = trueClass(neighborsInd)
```

```
neighbors = 1
```

```
class1 = find(neighbors == 1)
```

```
class1 = 1
```

```
class2 = find(neighbors == 2)
```

```
class2 = 

     []
```

```
joint = [size(class1,1);size(class2,1)];
[value, class] = max(joint);
fprintf('The new sample is in class  %d',class)
```

```
The new sample is in class  1
```

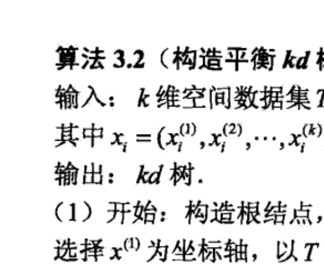
Use functions in MATLAB toolbox

see KNNClassifier_iris.mlx

3.3 k近邻法的实现：kd树（k-dimensional tree）

3.3.1 构造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.



1

算法 3.2（构造平衡 *kd* 树）

输入：*k* 维空间数据集 $T=\{x_1,x_2,\cdots,x_N\}$，

其中 $x_i=(x_i^{(1)},x_i^{(2)},\cdots,x_i^{(k)})^{\mathrm{T}}$，$i=1,2,\cdots,N$；

输出：*kd* 树．

（1）开始：构造根结点，根结点对应于包含 T 的 k 维空间的超矩形区域．

选择 $x^{(1)}$ 为坐标轴，以 T 中所有实例的 $x^{(1)}$ 坐标的中位数为切分点，将根结点对应的超矩形区域切分为两个子区域．切分由通过切分点并与坐标轴 $x^{(1)}$ 垂直的超平面实现．

由根结点生成深度为 1 的左、右子结点：左子结点对应坐标 $x^{(1)}$ 小于切分点的子区域，右子结点对应于坐标 $x^{(1)}$ 大于切分点的子区域.

将落在切分超平面上的实例点保存在根结点.

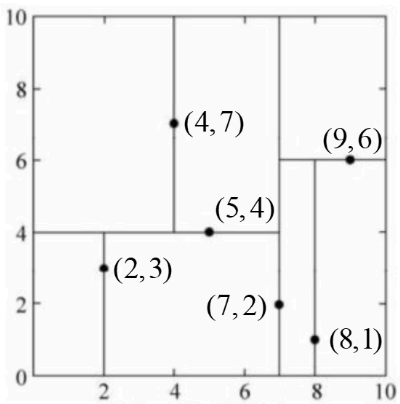
（2）重复：对深度为 j 的结点，选择 $x^{(l)}$ 为切分的坐标轴，$l = j(\bmod k)+1$，以该结点的区域中所有实例的 $x^{(l)}$ 坐标的中位数为切分点，将该结点对应的超矩形区域切分为两个子区域．切分由通过切分点并与坐标轴 $x^{(l)}$ 垂直的超平面实现.

由该结点生成深度为 $j+1$ 的左、右子结点：左子结点对应坐标 $x^{(l)}$ 小于切分点的子区域，右子结点对应坐标 $x^{(l)}$ 大于切分点的子区域.

将落在切分超平面上的实例点保存在该结点.

（3）直到两个子区域没有实例存在时停止．从而形成 *kd* 树的区域划分.

例3.2 给定一个二维空间的数据集：$T = \{(2,3)^{\mathrm{T}}, (5,4)^{\mathrm{T}}, (9,6)^{\mathrm{T}}, (4,7)^{\mathrm{T}}, (8,1)^{\mathrm{T}}, (7,2)^{\mathrm{T}}\}$，构造一个平衡kd树.



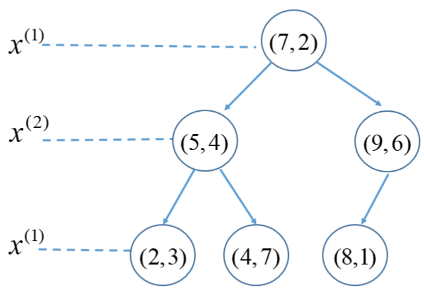
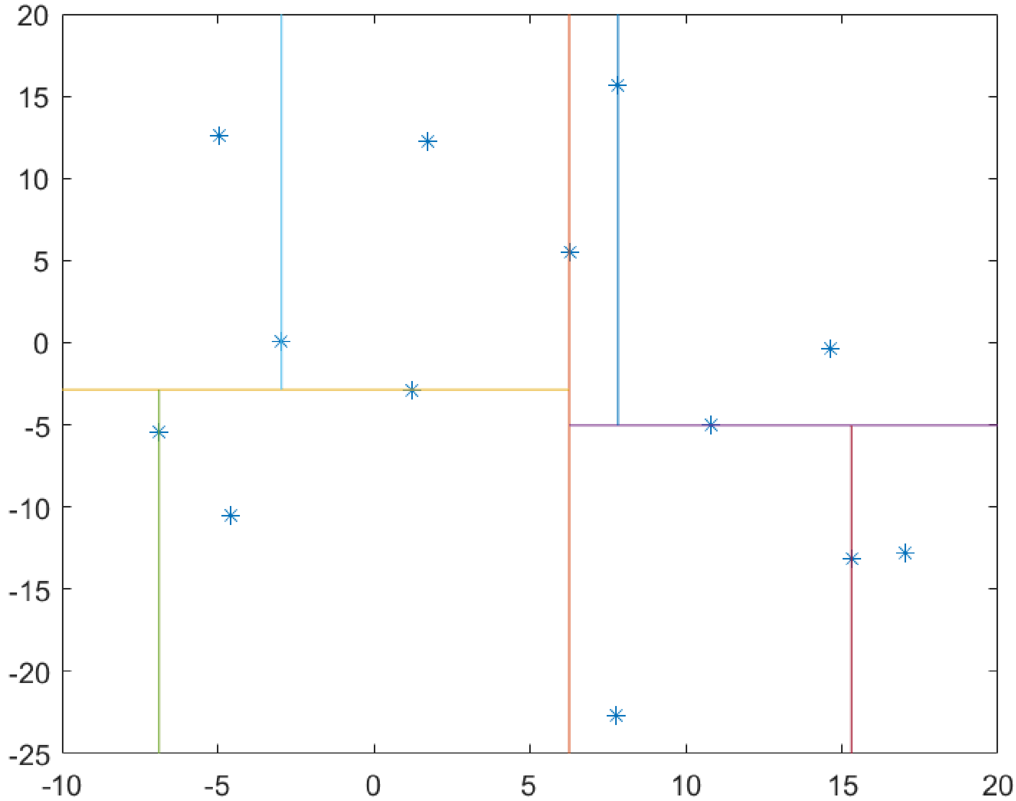


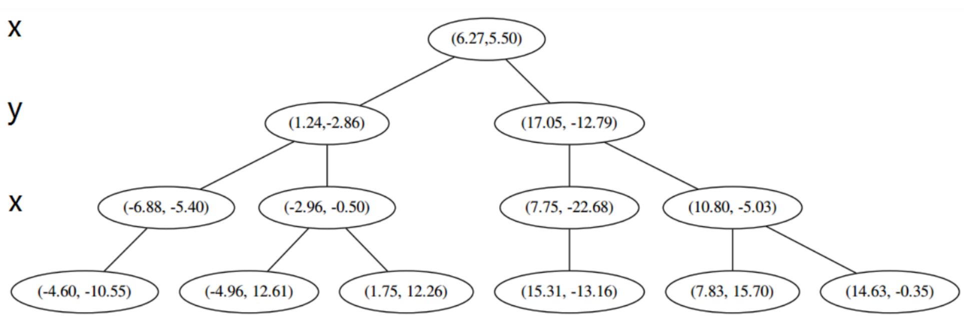
图3.3 特征空间的划分　　　　图3.4 kd树示意图

```
% 例
x = [6.27,5.50;-4.60,-10.55;-4.96,12.61;1.75,12.26;15.31,-13.16;...
    7.83,15.70;14.63,-0.35;-6.88,-5.40;-2.96,0.050; 7.75,-22.68;10.80,-5.03;...
    1.24,-2.86; 17.05,-12.79];
figure,plot(x(:,1),x(:,2),'*')

% cutting line along x
x0 = find_median(x(:,1));
hold on; plot([x0,x0],[-25,20])

% cutting line along y
set11 = x(x(:,1)<x0,:);
set12 = x(x(:,1)>=x0,:);
y01 = find_median(set11(:,2));
y02 = find_median(set12(:,2));
hold on;plot([-10,x0],[y01,y01])
hold on;plot([x0,20],[y02,y02])

%cutting line along x
set21 = set11(set11(:,2)<y01,:);
set22 = set11(set11(:,2)>=y01,:);
set23 = set12(set12(:,2)<y02,:);
set24 = set12(set12(:,2)>=y02,:);
```

```
x01 = find_median(set21(:,1));
x02 = find_median(set22(:,1));
x03 = find_median(set23(:,1));
x04 = find_median(set24(:,1));
hold on;plot([x01,x01],[-25,y01])
hold on;plot([x02,x02],[y01,20])
hold on;plot([x03,x03],[-25,y02])
hold on;plot([x04,x04],[y02,20])
```

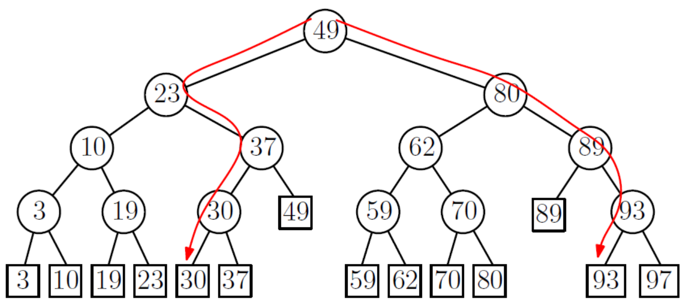


kd 树



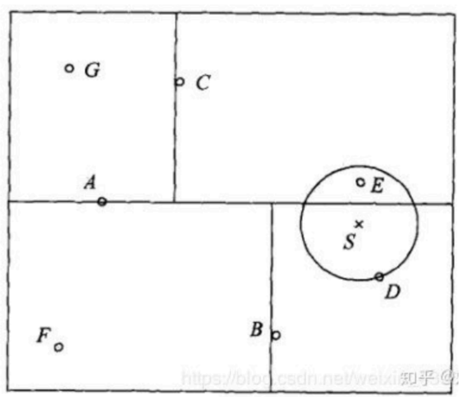
3.3.2 搜索kd树

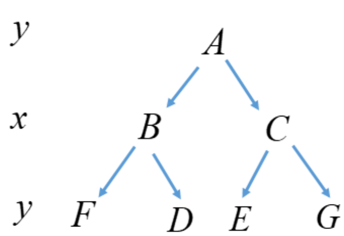
The search paths for 25 and for 90



1维情形

例3.3 给定一个如图3.5所示的kd树，根结点为A, 其子结点为B,C等；树上共存储7个实例点，另有一个输入实例S，求S的最近邻点.





搜索顺序ABDBFACE

算法 3.3（用 *kd* 树的最近邻搜索）

输入：已构造的 *kd* 树；目标点 x；

输出：x 的最近邻.

（1）在 *kd* 树中找出包含目标点 x 的叶结点：从根结点出发，递归地向下访问 *kd* 树．若目标点 x 当前维的坐标小于切分点的坐标，则移动到左子结点，否则移动到右子结点．直到子结点为叶结点为止.

（2）以此叶结点为“当前最近点”.

（3）递归地向上回退，在每个结点进行以下操作：

（a）如果该结点保存的实例点比当前最近点距离目标点更近，则以该实例点为“当前最近点”.

（b）当前最近点一定存在于该结点一个子结点对应的区域．检查该子结点的父结点的另一子结点对应的区域是否有更近的点．具体地，检查另一子结点对应

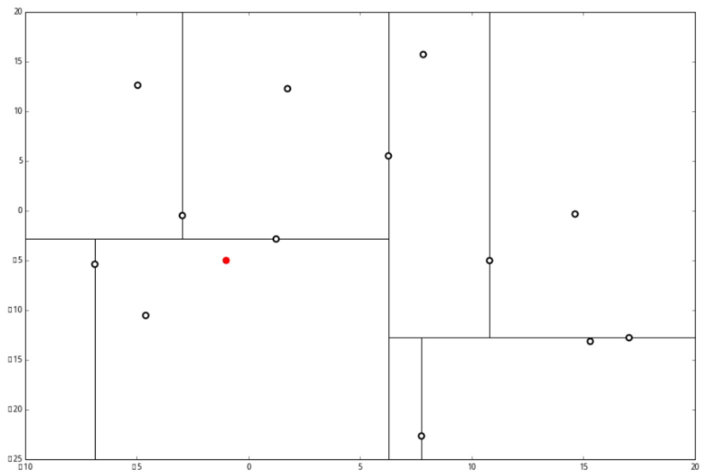
的区域是否与以目标点为球心、以目标点与“当前最近点”间的距离为半径的超球体相交.

如果相交，可能在另一个子结点对应的区域内存在距目标点更近的点，移动到另一个子结点．接着，递归地进行最近邻搜索；

如果不相交，向上回退.

（4）当回退到根结点时，搜索结束．最后的“当前最近点”即为 x 的最近邻点. ■

例：寻找**(-1,-5)**的**k**近邻，k=3



参考详细解析https://www.joinquant.com/view/community/detail/c2c41c79657cebf8cd871b44ce4f5d97

作业

3.1 参照图3.1, 在二维空间中给出实例点， 画出k为1和3时的k近邻法构成的空间划分，并对其进行比较，体会k值选择与模型复杂度及预测准确率的关系.

3.2 利用例题3.2构造的kd树求点 $x=(3,4.5)^T$ 的最近邻点.

```
function x0 = find_median(x)
x = sort(x);
if length(x)>1
    if mod(length(x),2)~=0
        x0 = median(x);
    else
        x0 = median(x(1:end-1));
    end
end
end
```