分类技术（二）

电子信息与通信学院 冯 镔
fengbin@hust.edu.cn

基于规则的分类器

- 用一组“If…then…”的规则来分类记录
- 规则：(条件) →y
  - 条件是一组属性测试的集合（规则前件、前提）
  - $(A_1 \text{ op } v_1)^ (A_2 \text{ op } v_2)^ \ldots ^ (A_k \text{ op } v_k)$
  - (A_j, v_j): 属性—值
  - op：逻辑运算符，{=, ≠, <, >, ≤, ≥}
  - y是类标号（规则后件）

基于规则的分类器

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
human	warm	yes	no	no	mammals
python	cold	no	no	no	reptiles
salmon	cold	no	no	yes	fishes
whale	warm	yes	no	yes	mammals
frog	cold	no	no	sometimes	amphibians
komodo	cold	no	no	no	reptiles
bat	warm	yes	yes	no	mammals
pigeon	warm	no	yes	no	birds
cat	warm	yes	no	no	mammals
leopard shark	cold	yes	no	yes	fishes
turtle	cold	no	no	sometimes	reptiles
penguin	warm	no	no	sometimes	birds
porcupine	warm	yes	no	no	mammals
eel	cold	no	no	yes	fishes
salamander	cold	no	no	sometimes	amphibians
gila monster	cold	no	no	no	reptiles
platypus	warm	no	no	no	mammals
owl	warm	no	yes	no	birds
dolphin	warm	yes	no	yes	mammals
eagle	warm	no	yes	no	birds

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds

R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes

R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals

R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles

R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

基于规则的分类器

- 如果规则r的前件和记录x的属性匹配，则称r覆盖x

R1: (胎生 = 否) ∧ (飞行动物 = 是) → 鸟类

R2: (胎生=否) ∧ (水生动物 = 是) → 鱼类

R3: (胎生= 是) ∧ (体温 = 恒温) → 哺乳类

R4: (胎生=否) ∧ (飞行动物 = 否) → 爬行类

R5: (水生动物 = 半) → 两栖类

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
hawk	warm	no	yes	no	?
grizzly bear	warm	yes	no	no	?

R1覆盖hawk →鸟类

R3覆盖grizzly bear →哺乳类

覆盖率和准确度

- 覆盖率
  - 满足规则前件的比例
  - **|A|/|D|**
- 准确度
  - 满足规则前件和后件的比例
  - **|A∩ y|/|A|**

Tid	Refund	Marital Status	Taxable Income	Class
1	Yes	**Single**	125K	**No**
2	No	Married	100K	**No**
3	No	**Single**	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	**Single**	85K	**Yes**
9	No	Married	75K	**No**
10	No	**Single**	90K	**Yes**

(Status=Single) → No

Coverage = 40%, Accuracy = 50%

工作原理

R1: (胎生 = 否) ∧ (飞行动物 = 是) → 鸟类
R2: (胎生=否) ∧ (水生动物 = 是) → 鱼类
R3: (胎生= 是) ∧ (体温 = 恒温) → 哺乳类
R4: (胎生=否) ∧ (飞行动物 = 否) → 爬行类
R5: (水生动物 = 半) → 两栖类

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
lemur	warm	yes	no	no	?
turtle	cold	no	no	sometimes	?
dogfish shark	cold	yes	no	yes	?

狐猴触发规则 **R3,** 因此归为哺乳类
海龟触发规则 **R4** 和 **R5**，如何解决？
角鲛鲨没有触发任何规则，如何解决？

基于规则的分类器的性质

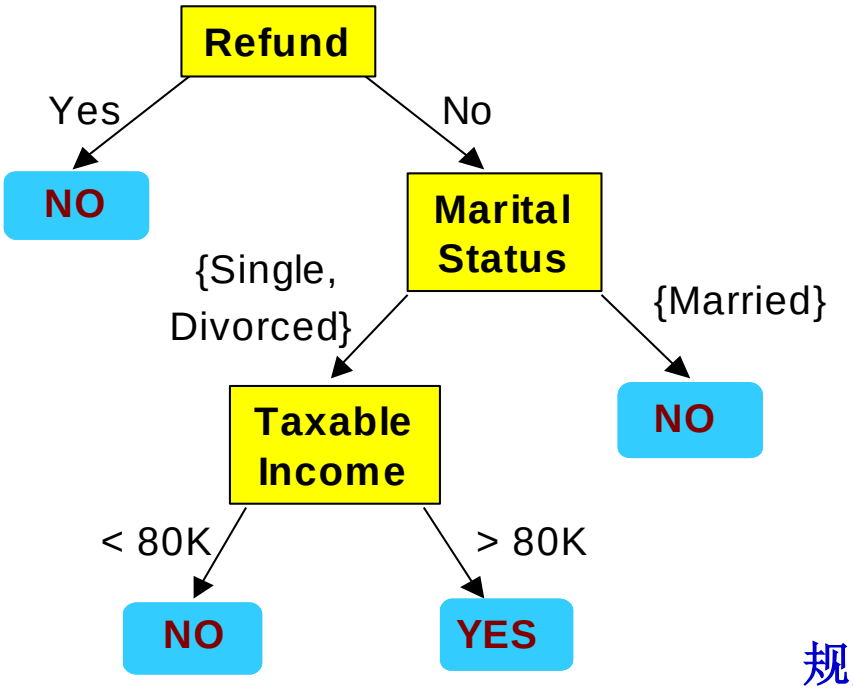
- 互斥规则
  - 如果规则集R中不存在两条规则被同一条记录触发，则称规则集R中的规则是互斥的
  - 每条记录至多被R中的一条规则覆盖
- 穷举规则
  - 如果对属性值的任一组合，R中都存在一条规则加以覆盖，则称规则集R具有穷举覆盖
  - 每条记录至少被一条规则覆盖

基于规则的分类器

R1: (体温 = 冷血) → 非哺乳类
R2: (体温=恒温) ∧ (胎生 = 是) → 哺乳类
R3: (体温=恒温) ∧ (胎生 = 否) → 非哺乳类

一个互斥和穷举规则集的例子

从决策树到规则



Classification Rules

(Refund=Yes) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income<80K) ==> No

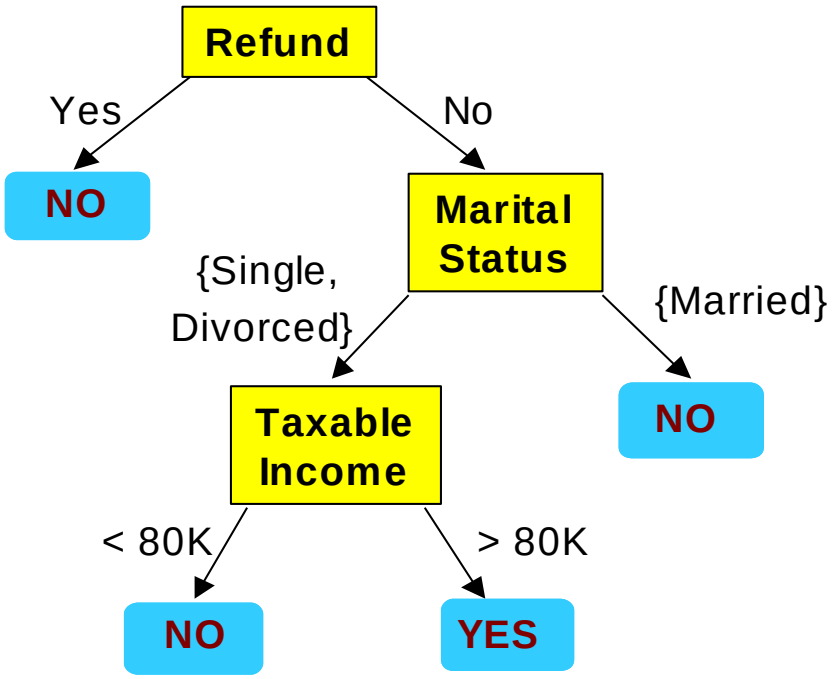
(Refund=No, Marital Status={Single,Divorced}, Taxable Income>80K) ==> Yes

(Refund=No, Marital Status={Married}) ==> No

规则集是互斥且穷举的

规则集包含了决策树中所有的信息

规则可进行简化



Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Initial Rule: $(\text{Refund=No}) \wedge (\text{Status=Married}) \rightarrow \text{No}$

Simplified Rule: $(\text{Status=Married}) \rightarrow \text{No}$

规则简化的影响

- 规则不再穷举
  - 一条记录可能无法触发任何规则
  - 如何解决？
    - 使用默认规则 r_d:()→y_d
- 规则不再互斥
  - 一条记录有可能被多条规则触发
  - 如何解决？
    - 有序规则集
    - 无序规则集—投票机制

有序规则集

- 规则集中的规则按照优先级降序排列
  - 决策表
- 当对一条记录进行分类时
  - 由覆盖记录优先级最高的规则对其进行分类
  - 没有任何规则触发时，指定为缺省类

R1: (Give Birth = no) ∧ (Can Fly = yes) → Birds
R2: (Give Birth = no) ∧ (Live in Water = yes) → Fishes
R3: (Give Birth = yes) ∧ (Blood Type = warm) → Mammals
R4: (Give Birth = no) ∧ (Can Fly = no) → Reptiles
R5: (Live in Water = sometimes) → Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
turtle	cold	no	no	sometimes	?

无序规则集

- 记录触发的每条规则看作是对相应类的一次投票
  - 可以用规则的准确率加权
- 不容易受到不合适的规则顺序影响
- 建立模型开销较小
  - 不需要维护规则的顺序
- 分类记录计算量大
  - 记录的属性要与规则集中的每一条规则比较

规则排序的方案

- 基于规则的排序
  - 根据规则质量的“度量”对规则排序
- 基于类的排序
  - 属于同一个类的规则在规则集中一起出现

Rule-Based Ordering

(Skin Cover=feathers, Aerial Creature=yes) ==> Birds

(Body temperature=warm-blooded, Gives Birth=yes) ==> Mammals

(Body temperature=warm-blooded, Gives Birth=no) ==> Birds

(Aquatic Creature=semi)) ==> Amphibians

(Skin Cover=scales, Aquatic Creature=no) ==> Reptiles

(Skin Cover=scales, Aquatic Creature=yes) ==> Fishes

(Skin Cover=none) ==> Amphibians

Class-Based Ordering

(Skin Cover=feathers, Aerial Creature=yes) ==> Birds

(Body temperature=warm-blooded, Gives Birth=no) ==> Birds

(Body temperature=warm-blooded, Gives Birth=yes) ==> Mammals

(Aquatic Creature=semi)) ==> Amphibians

(Skin Cover=none) ==> Amphibians

(Skin Cover=scales, Aquatic Creature=no) ==> Reptiles

(Skin Cover=scales, Aquatic Creature=yes) ==> Fishes

基于规则的分类器

- 习题1
- 考虑一个二值分类问题，属性集和属性值为
  - 空调={可用，不可用}
  - 引擎={好，差}
  - 行车里程={高，中，低}
  - 生锈={是，否}

行车里程=高 → 价值=低
行车里程=低 → 价值=高
空调=可用，引擎=好 → 价值=高
空调=可用，引擎=差 → 价值=低
空调=不可用 → 价值=低

- 假设一个基于规则的分类器产生的规则集为
- 互斥？完全？需要排序？需要默认类？

如何建立分类规则

- 直接方法
  - 直接从数据中提取分类规则
  - 如：**RIPPER, CN2, Holte's 1R**
- 间接方法
  - 从其他分类模型中（决策树，神经网络等）提取分类规则
  - 如：**C4.5规则**

直接方法：顺序覆盖

顺序覆盖算法

1: 令 E 是训练记录，A 是属性-值对的集合{(A_j, v_j)}

2: 令 Y_0 是类的有序集{y_1, y_2,…, y_k}

3: 令 R={}是初始规则列表

4: for 每个类 $y \in Y_0 - \{ y_k \}$ do

5: while 终止条件不满足 do

6: r<--Learn-One-Rule(E, A, y)

7: 从 E 中删除被 r 覆盖的训练记录

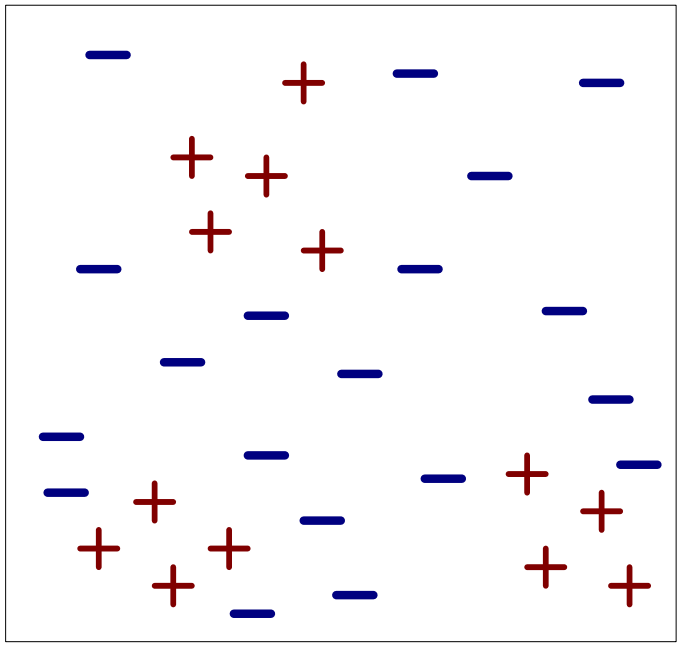
8: 追加 r 到规则列表尾部：R<--R∨r

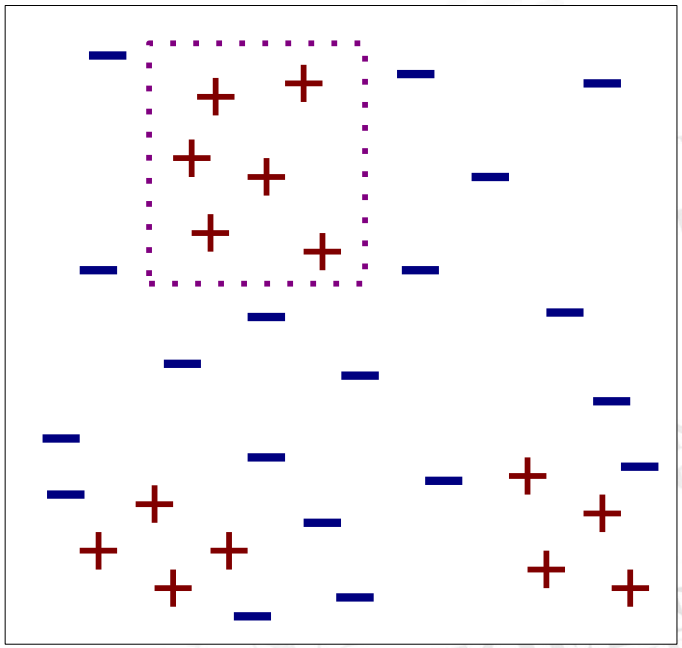
9: end while

10: end for

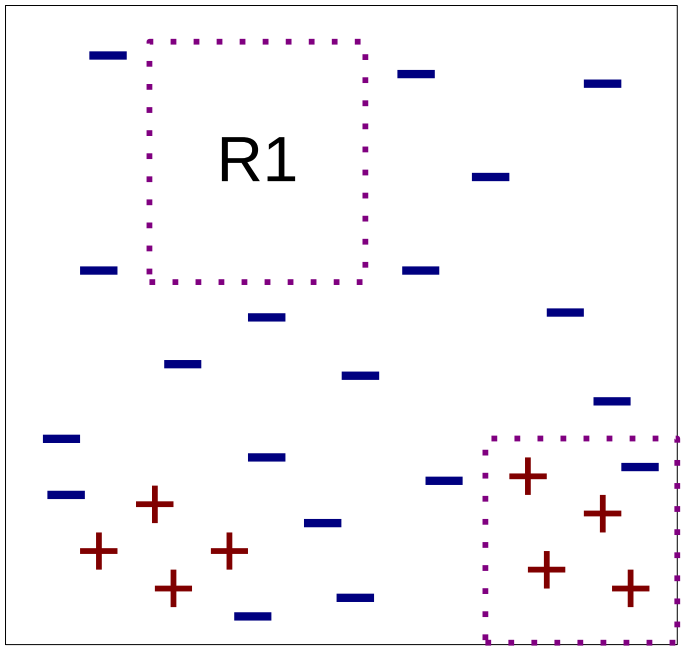
11: 把默认规则{}→ y_k 插入到规则列表 R 尾部

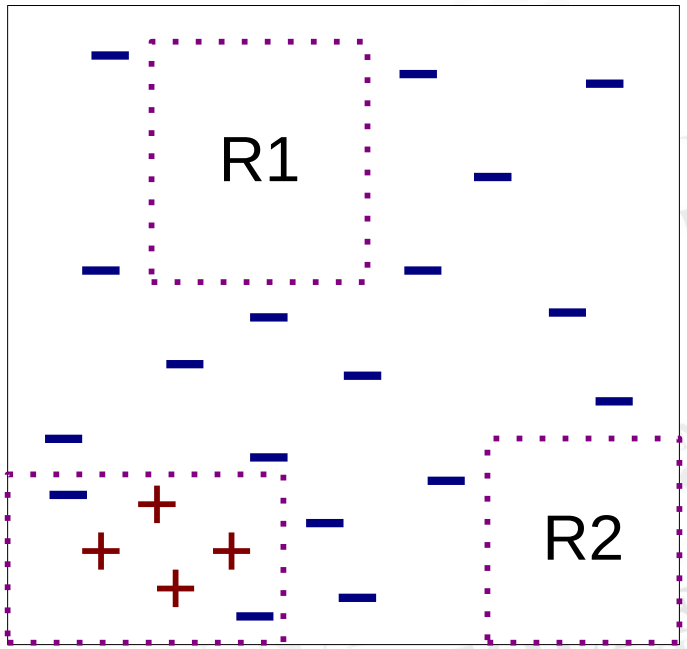
顺序覆盖的例子





顺序覆盖的例子





顺序覆盖

- 规则评估
- 规则增长
- 记录删除
- 停止条件
- 规则剪枝

规则评估

- 准确率
  - $\text{Accuracy} = \frac{n_c}{n}$
  - 局限性是没有考虑规则的覆盖率
  - 一个训练集包含60个正例和100个反例
  - 规则r1：覆盖50个正例和5个反例
  - 规则r2：覆盖2个正例和0个反例
  - Accuracy(r1) = 90.9%
  - Accuracy(r2) = 100%
  - 但是……？

规则评估

- 似然比统计量

$$R = 2\sum_{i=1}^{k} f_i \log(f_i / e_i)$$

  - k是类的个数
  - f_i是被规则覆盖的类i的样本的观测频率
  - e_i是规则做随机猜测的期望频率
  - R(r1)=2×[50×$\log_2$(50/20.625)+5×$\log_2$(5/34.375)]=99.9
  - R(r2)=2×[2×$\log_2$(2/0.75)+0×$\log_2$(0/1.25)]=5.66

规则评估

- **Laplace度量**
  - $$L = \frac{n_c + 1}{n + k}$$
- **M估计**
  - $$M = \frac{n_c + kp}{n + k}$$

n：规则覆盖的样例数

n_c：规则覆盖的正例数

k：类的总数

p：先验概率

p=1/k时，m估计等价于laplace度量

规则不覆盖任何训练样例时，
laplace度量减小到1/k，
m估计减小到先验概率p

覆盖率很高时，两个度量渐近的趋向于规则的准确率n_c/n

Laplace度量

r1：51/57 = 89.47%

r2：3/4 = 75%

规则评估

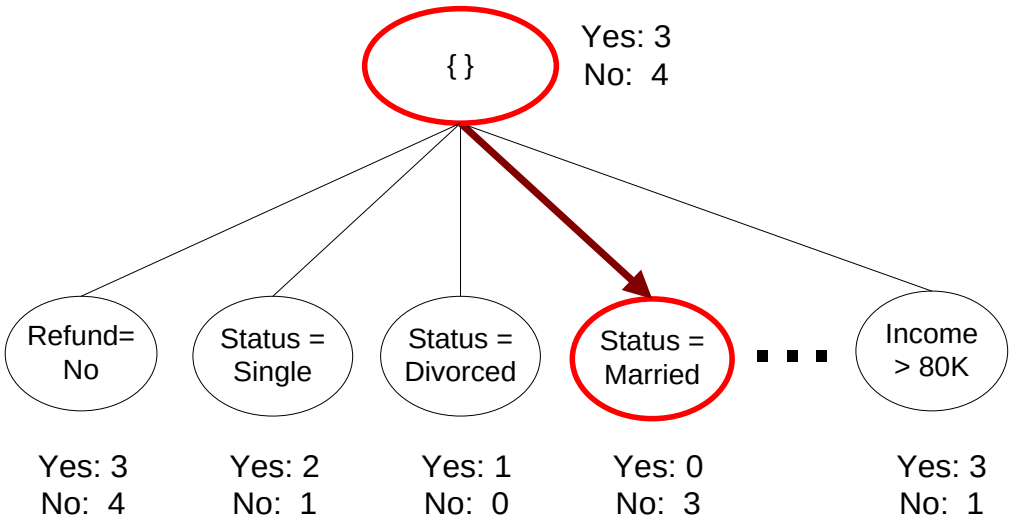
- FOIL信息增益
  - R0: {} => class (initial rule)
  - R1: {A} => class (rule after adding conjunct)
  - Gain(R0, R1) = t [log (p1/(p1+n1)) – log (p0/(p0 + n0))]
  - where
  - t: number of positive instances covered by both R0 and R1
  - p0: number of positive instances covered by R0
  - n0: number of negative instances covered by R0
  - p1: number of positive instances covered by R1
  - n1: number of negative instances covered by R1

规则评估

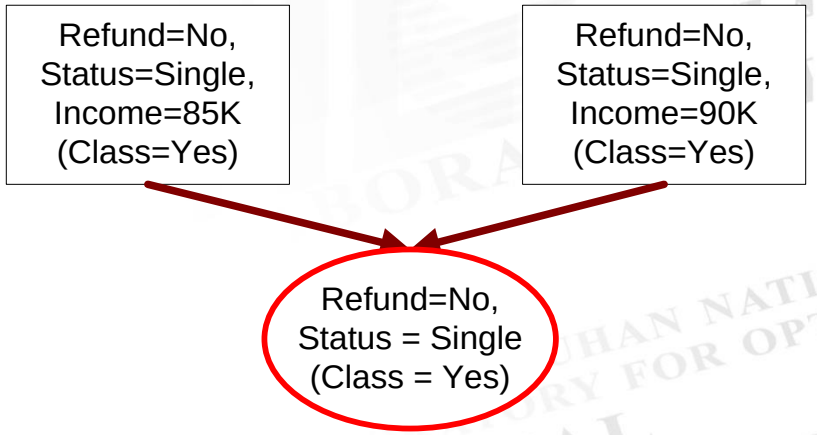
- 习题4
- 考虑一个训练集，包含100个正例和400个反例，对于下面的候选规则：
  - R1: A→+（覆盖4个正例和1个反例）
  - R2: B→+（覆盖30个正例和10个反例）
  - R3: C→+（覆盖100个正例和90个反例）
- 根据下面的度量，确定最好规则和最差规则
- (a)规则准确率；(b)FOIL信息增益；(c)似然比统计量；(d)拉普拉斯度量；(e)m度量

规则增长

- 两种常用策略



(a) General-to-specific



(b) Specific-to-general

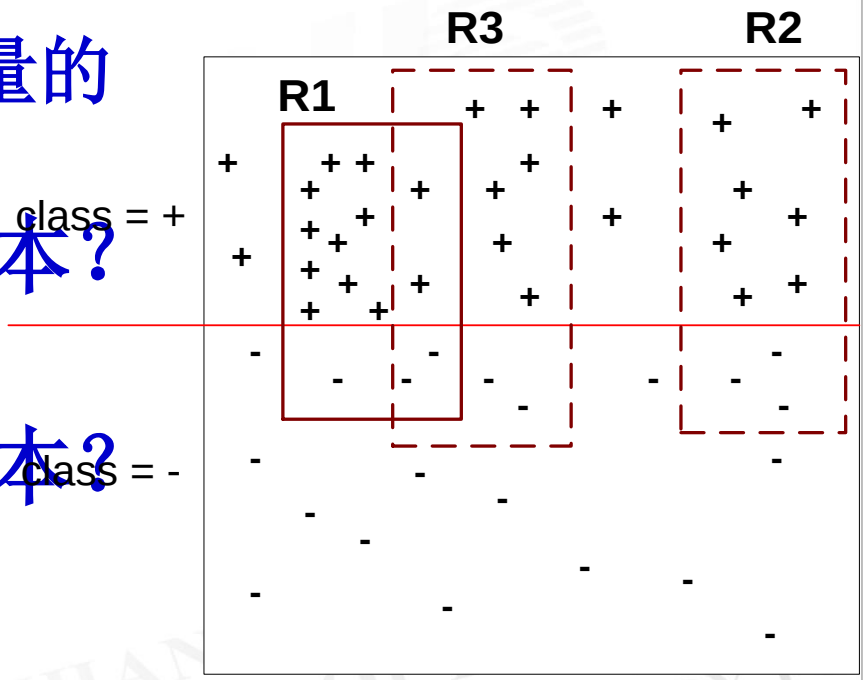
- 束状搜索
  - 维护**k**个最佳候选规则，独立增长，进入下一轮迭代

规则增长例子

- **CN2算法:**
  - 初始化一个空的规则: **{} => class**
  - 通过最小化熵度量来增加合取项: **{A}, {A,B}, …**
  - 根据规则覆盖的记录主要的类分布来决定规则后件
- **RIPPER算法:**
  - 初始化一个空的规则: **{} => class**
  - 根据最大化**FOIL**信息增益度量来增加合取项

记录删除

- 为什么需要删除记录？
  - 产生尽可能下一个高质量的规则
- 为什么需要删除正例样本？
  - 避免高估下一个规则
- 为什么需要删除反例样本？
  - 避免低估下一个规则



停止条件和规则剪枝

- 停止条件
  - 计算增益度量
  - 当增益不明显时，放弃新的规则
- 规则剪枝
  - 类似于决策树的后剪枝
  - 删除规则的一个合取项
  - 比较剪枝前后在校验集上的错误率
  - 若错误率改善，则进行剪枝

直接方法总结

- 生长一个单独的规则
- 删除规则覆盖的记录
- 规则剪枝（如果有必要）
- 将规则增加到规则集中
- 重复上述步骤

直接方法：RIPPER

- 对于两类问题，选择其中一个作为正类，另一个作为负类
  - 学习正类的规则
  - 负类被作为缺省类（多数类）
- 对于多类问题
  - 按照类的频率，进行升序排列（最不频繁-->最频繁）
  - 针对频率最小的类学习规则，将其他类的样例作为反例
  - 重复迭代过程，直到剩下最频繁类为止，将其作为缺省类

直接方法：RIPPER

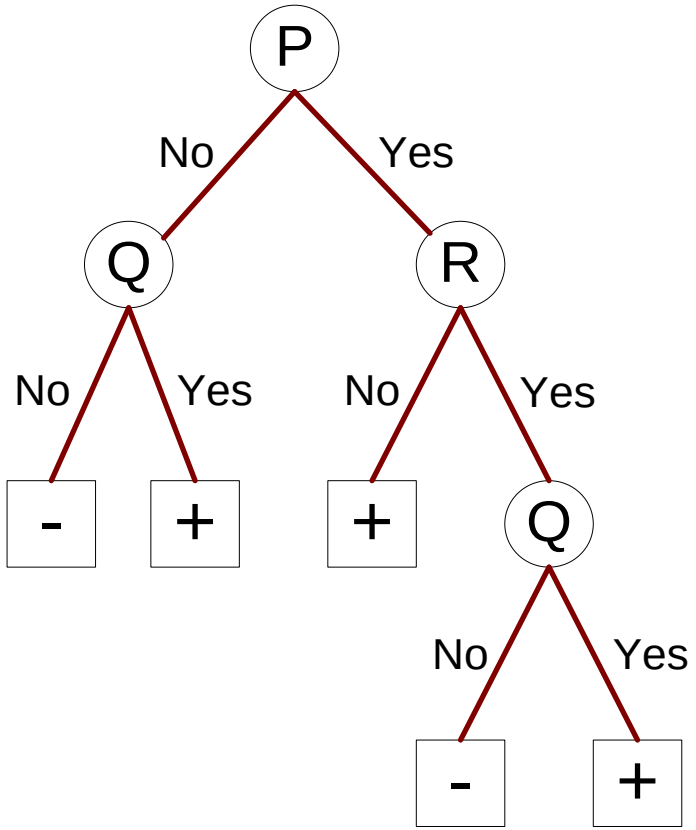
- 规则生长：
  - 初始化一个空规则
  - 根据FOIL信息增益度量增加合取项
  - 当规则不再覆盖负样本时停止
  - 根据在确认集上的性能进行剪枝
  - 修剪的度量： v = (p-n)/(p+n)
    - p: 正例样本数量
    - n: 负例样本数量
  - 修剪方法: 删除可以最大化v的合取项
  - 从最后添加的合取项开始修剪

直接方法：RIPPER

- 建立规则集
  - 找到覆盖当前正例样本集合的最佳规则
  - 删除该规则所覆盖的所有正例和反例
  - 只要该规则不违反基于最小描述长度原则的终止条件，就将它添加到规则集中
    - 若新规则将规则集的总描述长度增加了d比特以上，则停止加入规则集
    - 若新规则在确认集上的错误率超过50%，则停止加入新规则集

间接方法

- 决策树从根节点到叶节点的每一条路径都可以表示为一个分类规则



Rule Set

r1: (P=No,Q=No) ==> -
r2: (P=No,Q=Yes) ==> +
r3: (P=Yes,R=No) ==> +
r4: (P=Yes,R=Yes,Q=No) ==> -
r5: (P=Yes,R=Yes,Q=Yes) ==> +

间接方法：C4.5规则算法

- 从未修剪的决策树中提取规则
- 对于每条规则r: A → y
  - 考虑一个简化的替换规则r': A' → y，其中A'是从A中去掉一个合取项得到的
  - 若简化后规则的误差率低于原规则的误差率，就保留其中悲观误差率最低的规则
  - 重复剪枝步骤，直到泛化误差不能再改进为止

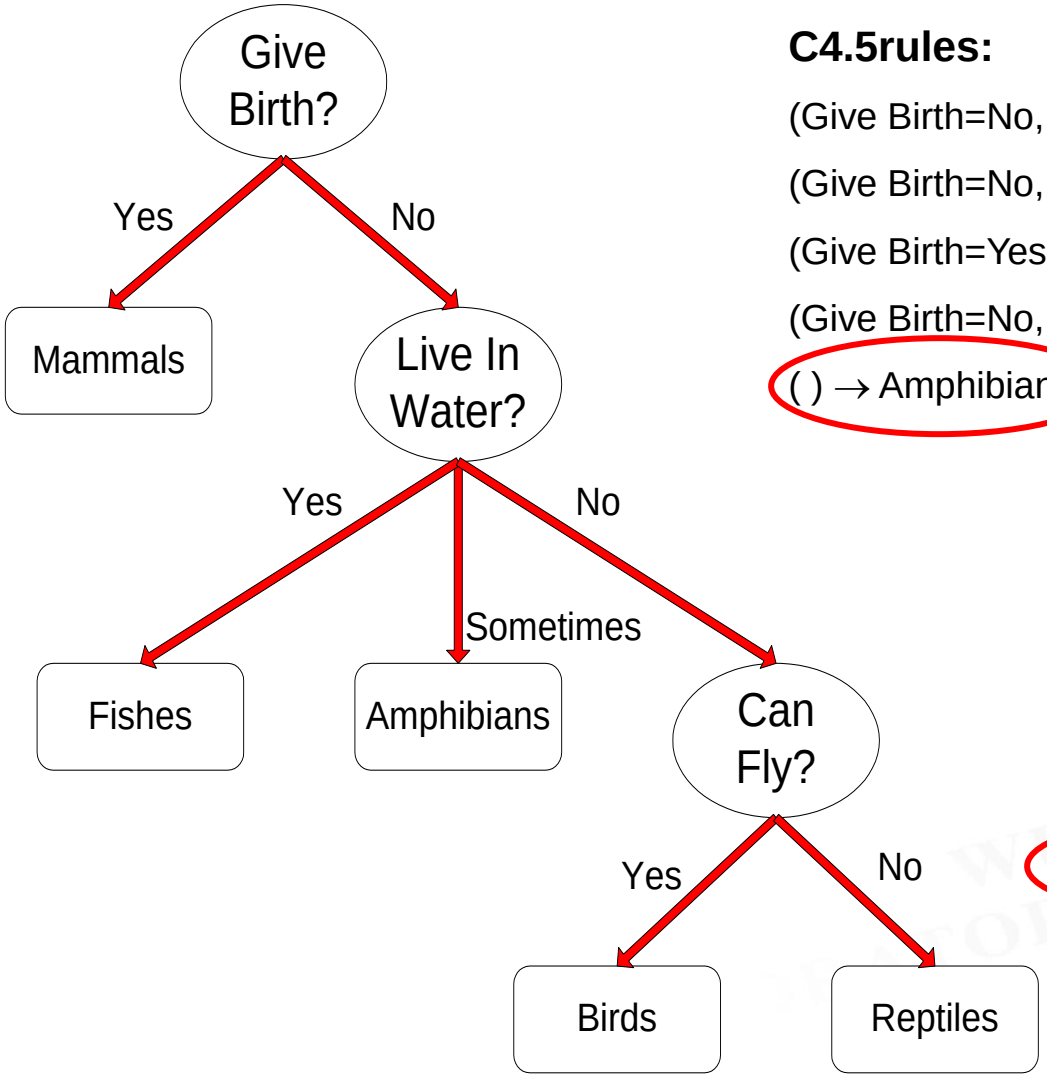
间接方法：C4.5规则算法

- 产生规则集后，进行排序（基于类的排序）
  - 每个规则子集是具有相同规则后件（类）的规则集合
  - 计算每个子集的描述长度，按照由小到大的顺序排序
  - 描述长度 = L(error) + g L(model)
  - g是根据模型中冗余属性的数量调节的参数
  - 冗余属性越多，g越小

例子

Name	Give Birth	Lay Eggs	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	no	yes	mammals
python	no	yes	no	no	no	reptiles
salmon	no	yes	no	yes	no	fishes
whale	yes	no	no	yes	no	mammals
frog	no	yes	no	sometimes	yes	amphibians
komodo	no	yes	no	no	yes	reptiles
bat	yes	no	yes	no	yes	mammals
pigeon	no	yes	yes	no	yes	birds
cat	yes	no	no	no	yes	mammals
leopard shark	yes	no	no	yes	no	fishes
turtle	no	yes	no	sometimes	yes	reptiles
penguin	no	yes	no	sometimes	yes	birds
porcupine	yes	no	no	no	yes	mammals
eel	no	yes	no	yes	no	fishes
salamander	no	yes	no	sometimes	yes	amphibians
gila monster	no	yes	no	no	yes	reptiles
platypus	no	yes	no	no	yes	mammals
owl	no	yes	yes	no	yes	birds
dolphin	yes	no	no	yes	no	mammals
eagle	no	yes	yes	no	yes	birds

C4.5 versus C4.5规则 versus RIPPER



C4.5rules:

(Give Birth=No, Can Fly=Yes) → Birds

(Give Birth=No, Live in Water=Yes) → Fishes

(Give Birth=Yes) → Mammals

(Give Birth=No, Can Fly=No, Live in Water=No) → Reptiles

( ) → Amphibians

RIPPER:

(Live in Water=Yes) → Fishes

(Have Legs=No) → Reptiles

(Give Birth=No, Can Fly=No, Live In Water=No) → Reptiles

(Can Fly=Yes,Give Birth=No) → Birds

() → Mammals

C4.5 versus C4.5规则 versus RIPPER

C4.5 and C4.5rules:

			PREDICTED CLASS			
		Amphibians	Fishes	Reptiles	Birds	Mammals
ACTUAL	Amphibians	2	0	0	0	0
CLASS	Fishes	0	2	0	0	1
	Reptiles	1	0	3	0	0
	Birds	1	0	0	3	0
	Mammals	0	0	1	0	6

RIPPER:

			PREDICTED CLASS			
		Amphibians	Fishes	Reptiles	Birds	Mammals
ACTUAL	Amphibians	0	0	0	0	2
CLASS	Fishes	0	3	0	0	0
	Reptiles	0	0	3	0	1
	Birds	0	0	1	2	1
	Mammals	0	2	1	0	4

基于规则的分类器

- 习题3
- C4.5规则是从决策树生成规则的间接方法的一个实现，而RIPPER是从数据中生成规则的直接方法的一个实现
  - (a)两种方法的优缺点；
  - (b)考虑一个数据集，其中类的大小差别很大，在为较小的类寻找高准确率规则方面，哪一种方法更好？

基于规则的分类器

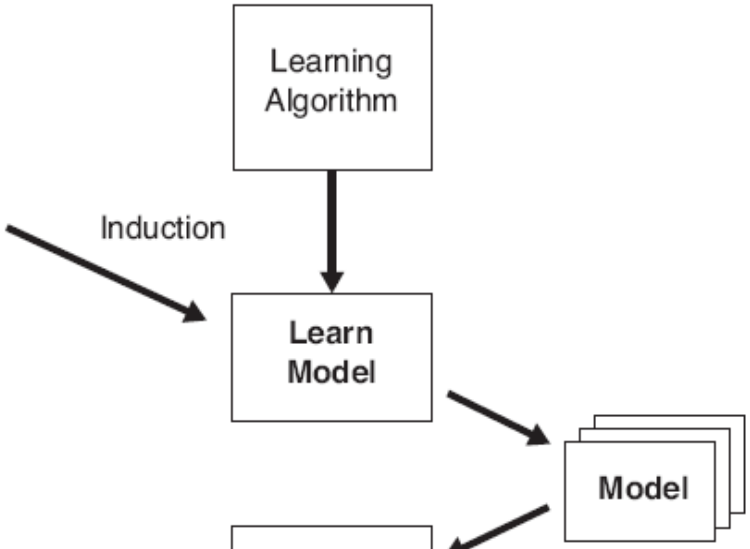
- 基于规则分类器的优点
- 规则集的表达能力几乎等价于决策树
  - 决策树可以用互斥和穷举的规则集表示
  - 性能和决策树相当
- 基于规则的分类器通常被用来产生更易于解释的描述性模型

基于实例的分类器

- 积极学习方法：决策树，基于规则分类器

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes



Test Set

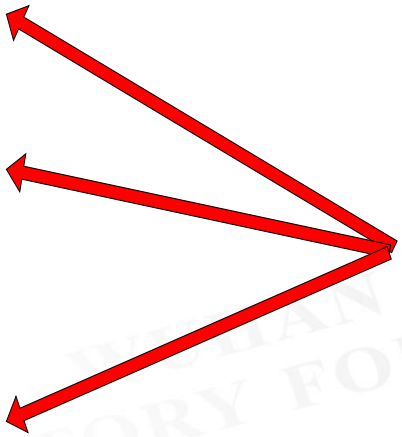
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

基于实例的分类器

➢消极学习方法

Set of Stored Cases

Atr1	……	AtrN	Class
			A
			B
			B
			C
			A
			C
			B



- 存储训练数据
- 使用训练样本来预测测试数据的类别

Unseen Case

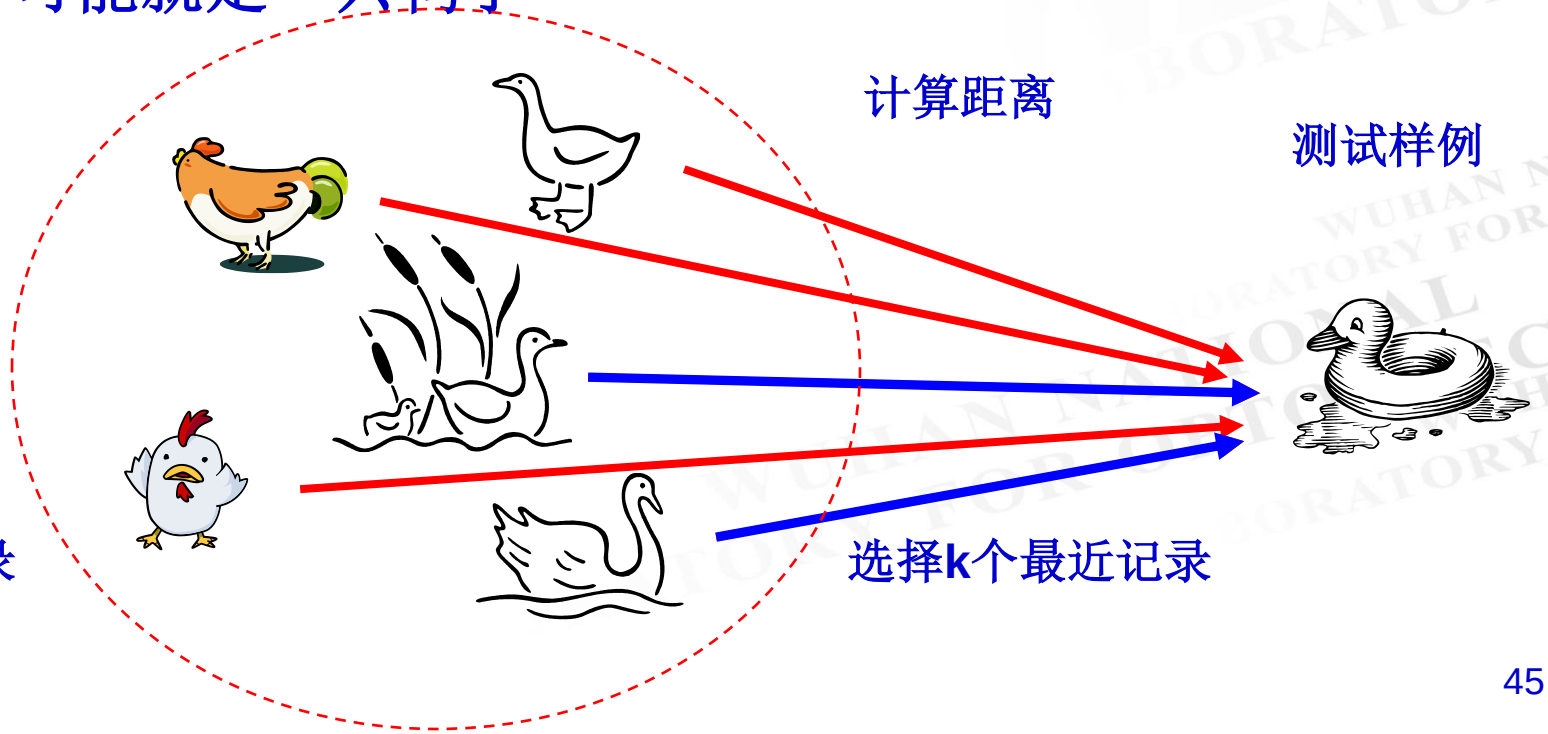
Atr1	……	AtrN

基于实例的分类器

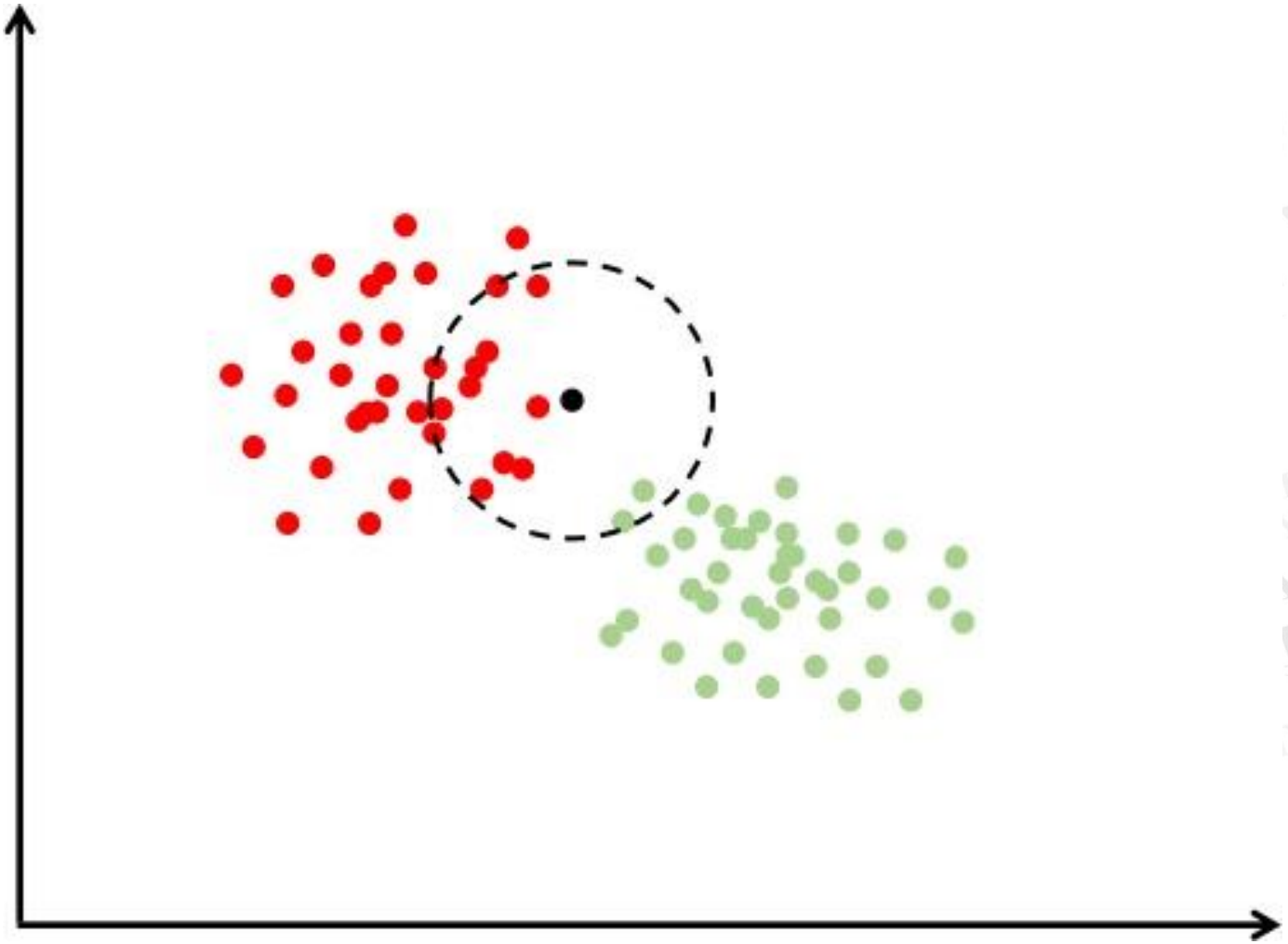
- **Rote分类器**
  - 记住整个训练数据，只有当测试记录的属性和一个训练样例完全匹配时才进行分类
- **最近邻分类器**
  - 采用**k**个和测试样例属性相对接近的训练样例（最近邻）来进行分类

最近邻分类器

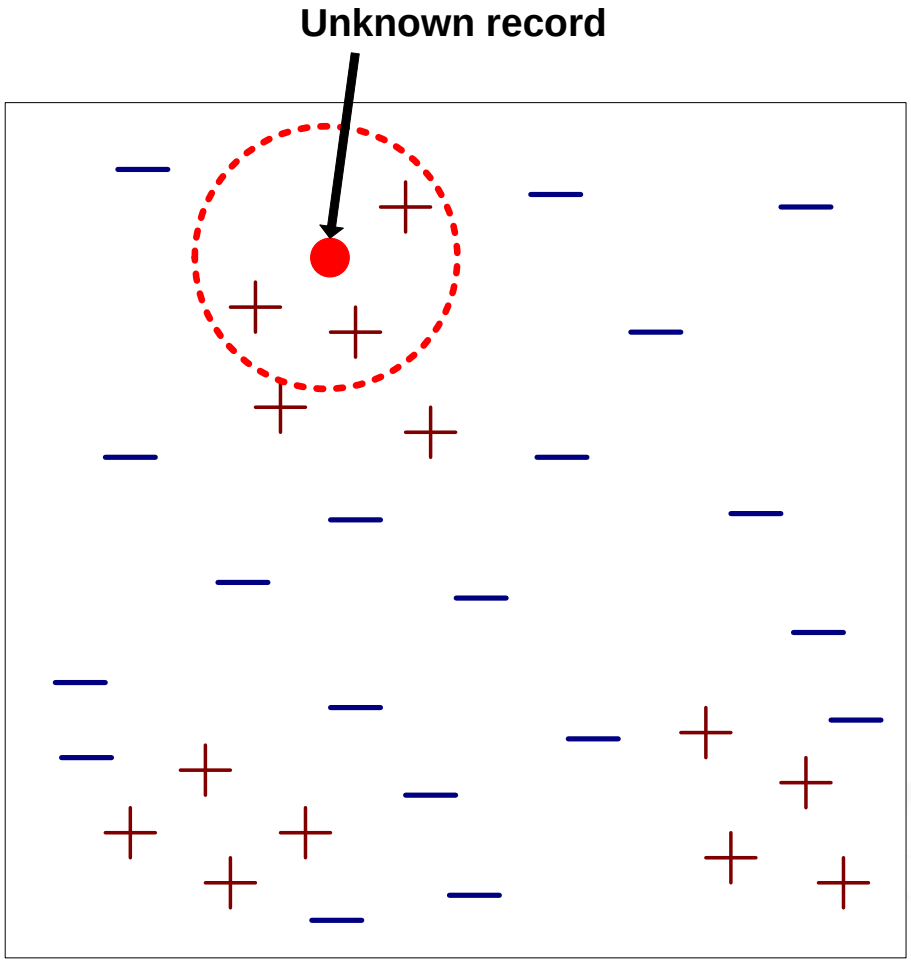
- 核心是模板匹配，将样本分到离它最相似的样本所属的类
- 基本原理：
  - 如果走像鸭子，叫像鸭子，看起来还像鸭子，那么它很可能就是一只鸭子



最近邻分类器

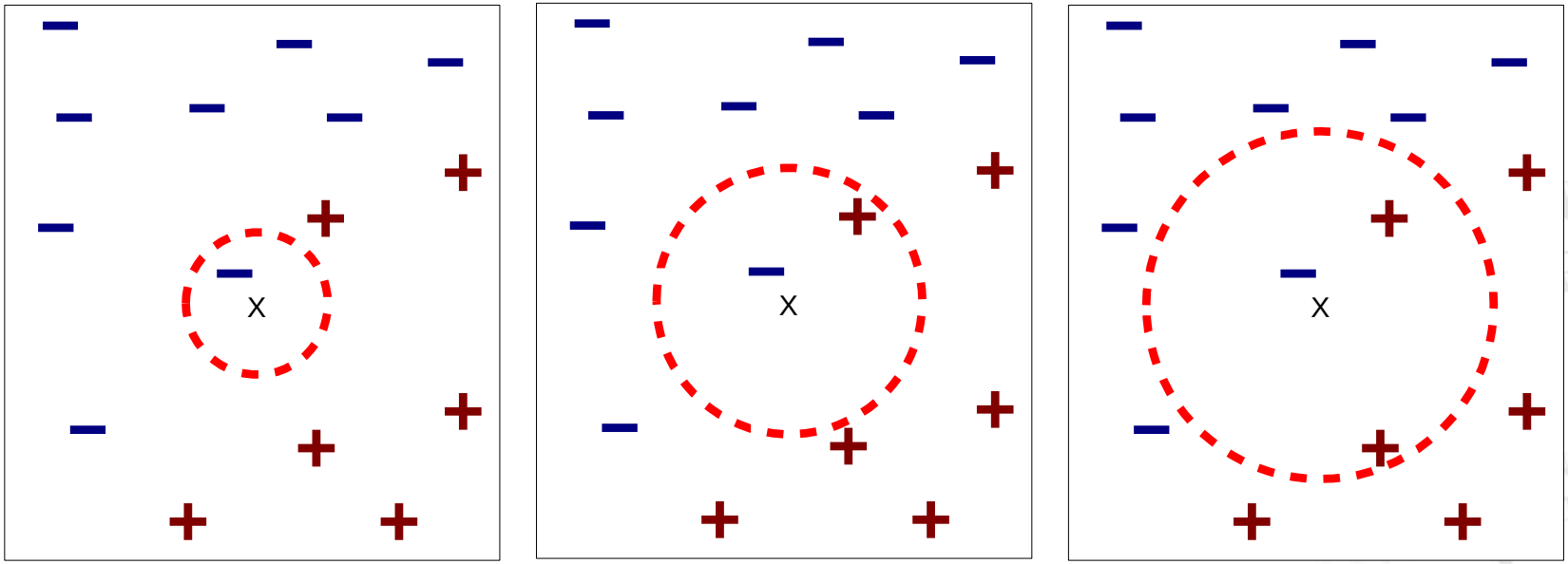


最近邻分类器



- 需要：
  - 存储的记录集
  - 计算记录间距离的度量
  - 需要考虑的最近邻数量k的值
- 为了分类未知记录
  - 计算其到其他训练记录的距离
  - 确认k个最近邻记录
  - 使用最近邻的类标号来确定未知记录的类标号 (如投票机制)

最近邻的定义

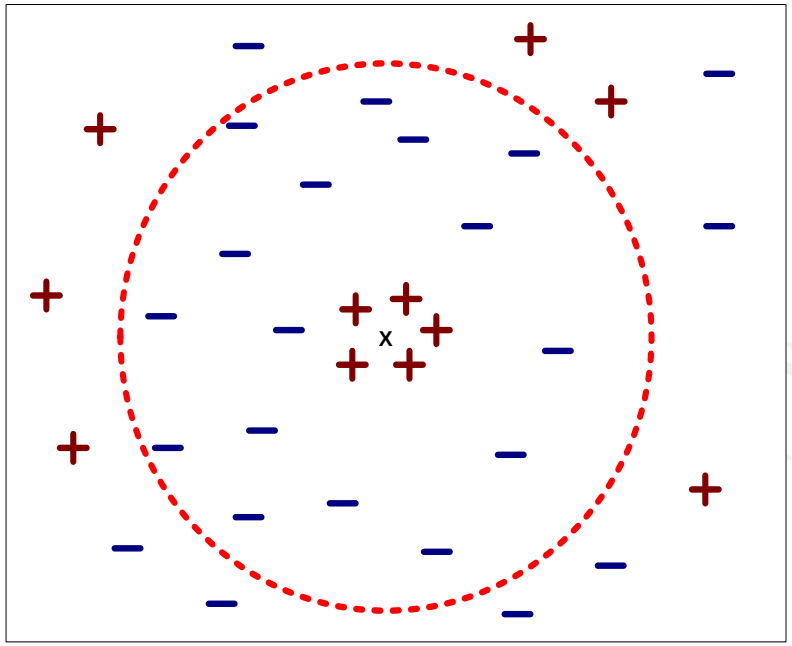


(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

记录x的k-最近邻就是和x的距离最小的k个数据点

最近邻分类器

- K的定义
  - 若k太小，容易受到噪声的影响
  - 若k太大，会错误的受到距离较远的数据点影响



最近邻分类器

K-最近邻分类算法

1: 令 k 是最近邻数目，D 是训练样例的集合

2: for 每个测试样例 z=(x', y') do

3: 计算 z 和每个样例(x, y) ∈D 之间的距离 d(x', x)

4: 选择离 z 最近的 k 个训练样例的集合 $D_z \subseteq D$

5: $y' = \arg\max \sum_{(x_i, y_i) \in D_z} I(v = y_i)$

6: end for

- 欧式距离 $d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$
- 距离加权， **w = 1/d²**

$$\arg\max \sum_{(x_i, y_i) \in D_z} w_i \times I(v = y_i)$$

最近邻分类器

- 节约了建立模型的时间，增加了分类的时间
- 基于局部信息进行决策，对噪声非常敏感
- 利用高效的部分排序算法或k-d tree实现快速的近邻样本查找
- 带权重的k近邻算法 $y=\left(\sum_{i=1}^{k} w_i y_i\right)/k$
- 可以用于回归问题 $y=\left(\sum_{i=1}^{k} y_i\right)/k$
- 天然支持多分类问题

最近邻分类器

- 需要采用适当的距离度量和数据预处理
  - 通过对属性进行缩放来避免距离度量被其中一个属性所左右
  - 身高：1.5～2.00
  - 体重：40kg～150kg
  - 收入：10,000～1,000,000

最近邻分类器

- 欧式距离度量的问题
  - 会给出违反常理的结果

1 1 1 1 1 1 1 1 1 1 1 0		1 0 0 0 0 0 0 0 0 0 0 0	
0 1 1 1 1 1 1 1 1 1 1 1	vs	0 0 0 0 0 0 0 0 0 0 0 1	
d = 1.4142		d = 1.4142	

  - 解决：归一化到单位矢量

Bayes分类器

- 一个解决分类问题的概率框架
  - 一种把类的先验知识和从数据中收集的新证据相结合的统计原理
  - 将样本判定为后验概率最大的类
- 条件概率 $P(C \mid A) = \frac{P(A,C)}{P(A)}$ $P(A \mid C) = \frac{P(A,C)}{P(C)}$
- **Bayes定理**

$$P(C \mid A) = \frac{P(A \mid C)P(C)}{P(A)} \quad posterior = \frac{likelihood \times prior}{evidence}$$

例子

- 已知：
  - 脑膜炎导致颈部僵硬的概率是**50%**
  - 一个病人患脑膜炎的先验概率是**1/50,000**
  - 一个病人患颈部僵硬的概率是**1/20**
- 如果有个病人患有颈部僵硬，那么他患有脑膜炎的概率是多少？

$$P(M|S)=\frac{P(S|M)P(M)}{P(S)}=\frac{0.5\times 1/50000}{1/20}=0.0002$$

习题6

- (a)假设本科生中锻炼的比例是15%，研究生中锻炼的比例是23%，如果大学生中研究生占1/5，其余是本科生，则锻炼的学生是研究生的概率是多少？
- (b)随机选择一个学生，该生是研究生或本科生的可能性哪个大？
- (c)随机选择一个锻炼的学生，。。。。。。
- (d)假设30%的研究生住学生宿舍，只有10%的本科生住学生宿舍，如果一个学生锻炼又住宿舍，他是研究生或本科生的可能性哪个大？（住宿舍和锻炼的学生相互独立）

Bayes分类器

- 将每个属性以及类标号看作随机变量
- 已知一条记录，其属性集为$(A_1, A_2,\ldots,A_n)$
  - 目标是预测类别C
  - 特别的，希望找到类别C使得$P(C| A_1, A_2,\ldots,A_n)$的值最大
- 我们能够从数据中直接预测出$P(C| A_1, A_2,\ldots,A_n)$吗？

Bayes分类器

- 方法
  - 根据Bayes定理，计算所有类别C的后验概率 $P(C| A_1, A_2,\ldots,A_n)$

$$P(C \mid A_1 A_2 \ldots A_n) = \frac{P(A_1 A_2 \ldots A_n \mid C)P(C)}{P(A_1 A_2 \ldots A_n)}$$

  - 选择最大化$P(C| A_1, A_2,\ldots,A_n)$的类别C
  - 等效于选择最大化$P(A_1, A_2, \ldots, A_n|C)\ P(C)$的类别C
- 如何估计$P(A_1, A_2, \ldots, A_n|C)$？

朴素Bayes分类器

- 假设在给定类别C的条件下，各属性间独立
- $P(A_1, A_2, \ldots, A_n | C) = P(A_1 | C_j)\, P(A_2 | C_j) \ldots P(A_n | C_j)$
- 对于所有的A_i 和C_j ， $P(A_i | C_j)$是可以估计的
- 新的记录将归类到使$P(C_j) \prod P(A_i | C_j)$ 最大化的类C_j

如何从数据中估计概率？

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- 类概率：$P(C) = N_c/N$
  - 如，$P(No) = 7/10$, $P(Yes) = 3/10$
- 对于离散属性：
  $P(A_i \mid C_k) = |A_{ik}|/ N_c$
  - 其中$|A_{ik}|$表示具有属性A_i同时又属于类C_k的记录个数
  - 如，
  - $P(Status=Married|No) = 4/7$
    $P(Refund=Yes|Yes)=0$

如何从数据中估计概率？

- 对于连续属性：
  - 将连续属性离散化，用相应的离散区间替换连续属性值
    - 离散区间数目的选择影响估计误差
  - 概率密度估计
    - 假设属性服从高斯分布
    - 利用训练数据估计分布参数，如均值和方差
    - 用得到的分布模型估计概率

如何从数据中估计概率？

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- 正态分布:

$$P(A_i \mid c_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(A_i-\mu_{ij})^2}{2\sigma_{ij}^2}}$$

  - 对于每一个(A_i, c_i)组合各训练一个

- 对于(收入, 类别=No):
  - If Class=No
    - sample mean = 110
    - sample variance = 2975

$$P(Income = 120 \mid No) = \frac{1}{\sqrt{2\pi}(54.54)} e^{-\frac{(120-110)^2}{2(2975)}} = 0.0072$$

朴素Bayes分类器

- 给定测试记录

$$X = (\text{Refund} = \text{No}, \text{Married}, \text{Income} = 120\text{K})$$

naive Bayes Classifier:

P(Refund=Yes|No) = 3/7
P(Refund=No|No) = 4/7
P(Refund=Yes|Yes) = 0
P(Refund=No|Yes) = 1
P(Marital Status=Single|No) = 2/7
P(Marital Status=Divorced|No)=1/7
P(Marital Status=Married|No) = 4/7
P(Marital Status=Single|Yes) = 2/7
P(Marital Status=Divorced|Yes)=1/7
P(Marital Status=Married|Yes) = 0

For taxable income:
If class=No: sample mean=110
sample variance=2975
If class=Yes: sample mean=90
sample variance=25

- **P(X|Class=No) = P(Refund=No|Class=No)**
  × P(Married| Class=No)
  × P(Income=120K| Class=No)
  = 4/7 × 4/7 × 0.0072 = 0.0024

- **P(X|Class=Yes) = P(Refund=No| Class=Yes)**
  × P(Married| Class=Yes)
  × P(Income=120K| Class=Yes)
  = 1 × 0 × 1.2 × 10^{-9} = 0

- **Since P(X|No)P(No) > P(X|Yes)P(Yes)**
- **Therefore P(No|X) > P(Yes|X)**
  => Class = No

朴素Bayes分类器

- 只要有一个条件概率为0，则整个表达式结果为0
- 概率估计

$$\text{Original}: P(A_i \mid C) = \frac{N_{ic}}{N_c}$$

$$\text{Laplace}: P(A_i \mid C) = \frac{N_{ic} + 1}{N_c + c}$$

$$\text{m-estimate}: P(A_i \mid C) = \frac{N_{ic} + mp}{N_c + m}$$

c: 类数量

p: 先验概率

m: 样本大小参数

习题7

- (a)估计条件概率P(A|+), P(B|+), P(C|+), P(A|-), P(B|-), P(C|-)
- (b)预测测试样本(A=0, B=1, C=0)的类标号
- (c)使用m估计方法估计条件概率(p=1/2且m=4)
- (d)使用(c)中的条件概率预测类标号

Record	A	B	C	Class
1	0	0	0	+
2	0	0	1	−
3	0	1	1	−
4	0	1	1	−
5	0	0	1	+
6	1	0	1	+
7	1	0	1	−
8	1	0	1	−
9	1	1	1	+
10	1	0	1	+

朴素Bayes分类器（总结）

- 对于孤立噪声点是鲁棒的
- 通过忽略训练数据来处理属性值遗漏的情况
- 对于无关属性是鲁棒的
- 属性间独立的假设太强，相关的属性会降低朴素Bayes分类器的性能
  - 采用其他技术，如Bayes置信网络

朴素Bayes分类器（总结）

- 考虑二元属性A和二元类变量Y关系如下

$P(A=0|Y=0)=0.4,\ P(A=1|Y=0)=0.6$
$P(A=0|Y=1)=0.6,\ P(A=1|Y=1)=0.4$

- 假设存在一个二元属性B，当Y=0时，B与A完全相关；当Y=1时，B与A相互独立且概率相同，给定一个记录属性为A=0, B=0，则

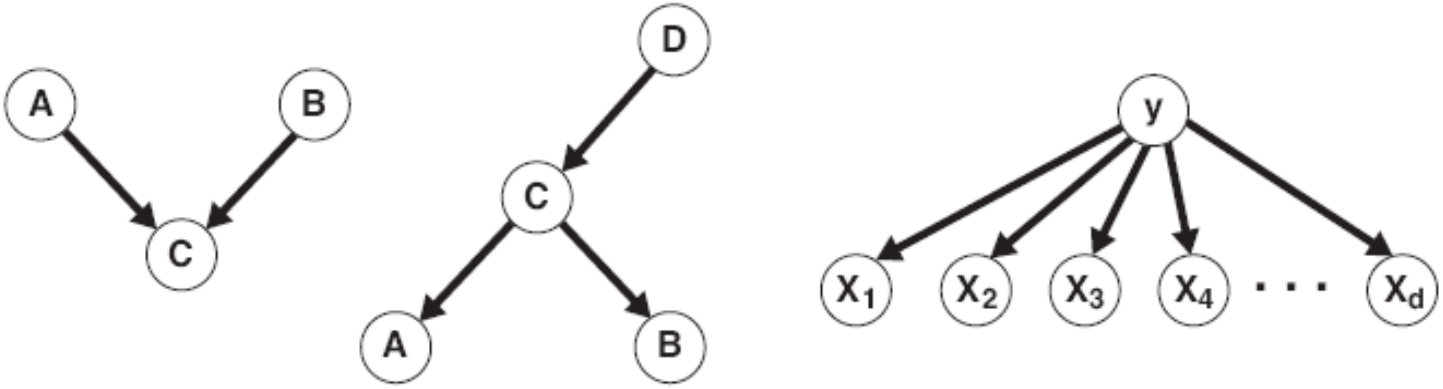
$P(Y=0|A=0, B=0)= P(A=0|Y=0)P(B=0|Y=0)P(Y=0)/P(A=0, B=0) \propto 0.16P(Y=0)$
$P(Y=1|A=0, B=0)= P(A=0|Y=1)P(B=0|Y=1)P(Y=1)/P(A=0, B=0) \propto 0.36P(Y=1)$

实际上

$P(Y=0|A=0, B=0)= P(A=0|Y=0)P(Y=0)/P(A=0, B=0)$
$\propto 0.4P(Y=0)$

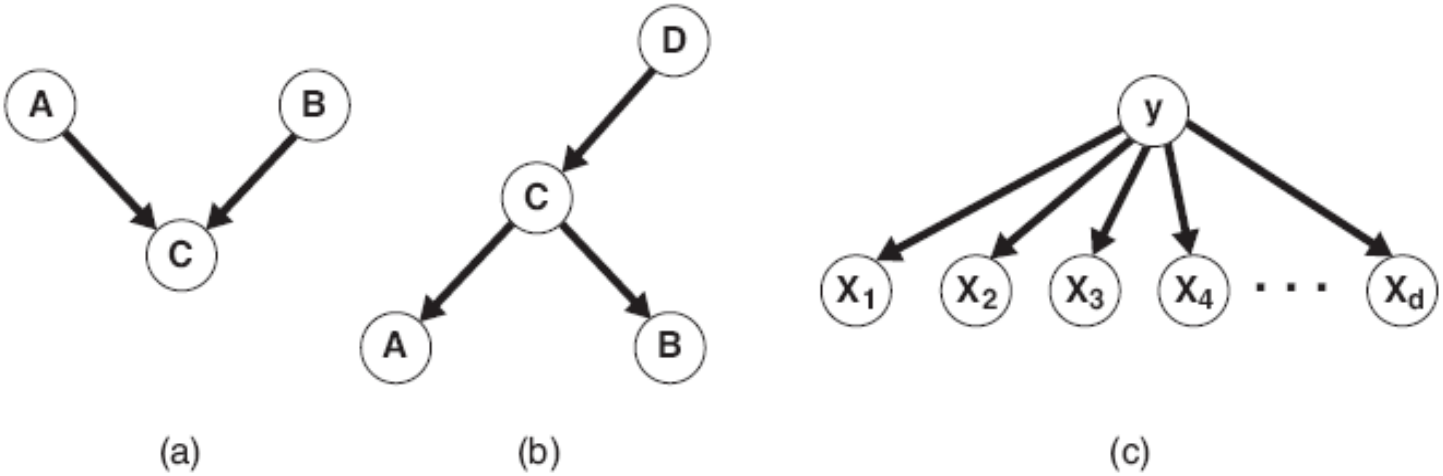
Bayes置信网络

- **Bayesian Belief Networks, BBN**
  - 一个有向无环图（dag），表示变量之间的依赖关系
  - 一个概率表，把各结点与其直接父结点关联起来



Bayes置信网络

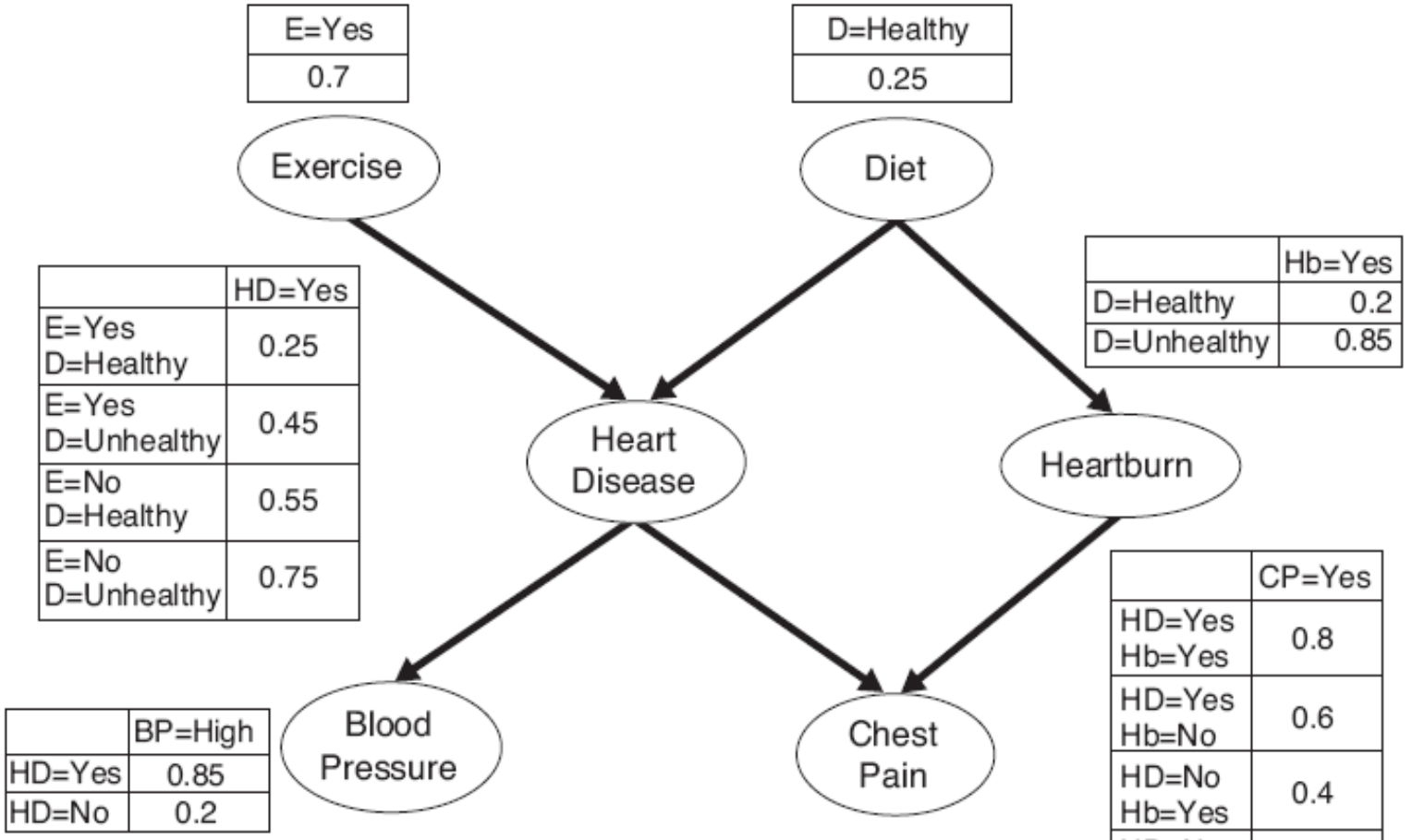
- 条件独立
  - Bayes置信网络中的一个结点，如果它的父母结点已知，则它条件独立于它的所有非后代结点



Bayes置信网络

- 概率表
  - 如果结点X没有父母结点，则表中只包含先验概率P(X)
  - 如果结点X只有一个父母结点Y，则表中包含条件概率P(X|Y)
  - 如果结点X有多个父母结点{$Y_1,Y_2,\ldots Y_k$}，则表中包含条件概率P(X| $Y_1,Y_2,\ldots Y_k$)

Bayes置信网络


E=Yes
0.7
D=Healthy
0.25
Exercise
Diet
HD=Yes
E=Yes D=Healthy 0.25
E=Yes D=Unhealthy 0.45
E=No D=Healthy 0.55
E=No D=Unhealthy 0.75
Hb=Yes
D=Healthy 0.2
D=Unhealthy 0.85
Heart Disease
Heartburn
BP=High
HD=Yes 0.85
HD=No 0.2
Blood Pressure
Chest Pain
CP=Yes
HD=Yes Hb=Yes 0.8
HD=Yes Hb=No 0.6
HD=No Hb=Yes 0.4
HD=No Hb=No 0.1

Bayes置信网络

- （1）创建网络结构
  - 利用专家知识编码获得
- （2）估计每一个结点的概率表中的概率值

贝叶斯网络拓扑结构的生成算法

1: 设 $T=(X_1, X_2, \ldots, X_D)$表示变量的一个总体次序

2: for j=1 to d do

3: 令 $X_{T(j)}$表示 T 中第 j 个次序最高的变量

4: 令 $\pi(X_{T(j)})=\{ X_{T(1)}, X_{T(2)}, \ldots, X_{T(j-1)}\}$表示排在 $X_{T(j)}$前面的变量的集合

5: 从 $\pi(X_{T(j)})$中去掉对 X_j 没有影响的变量（使用先验知识）

6: 在 $X_{T(j)}$和 $\pi(X_{T(j)})$中剩余的变量之间画弧

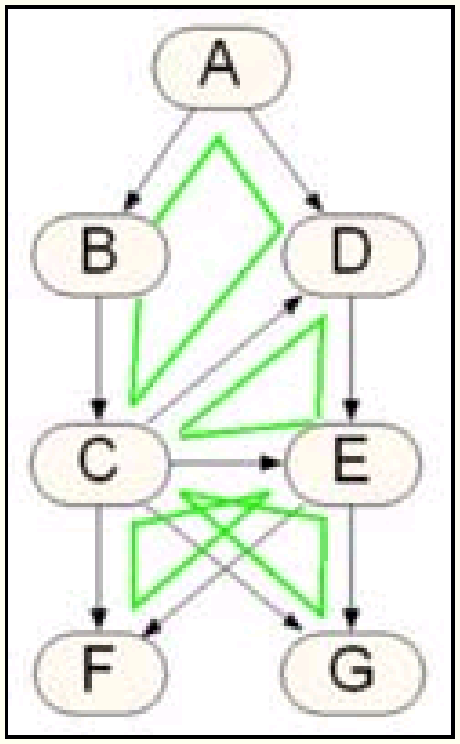
7: end for

Bayes置信网络

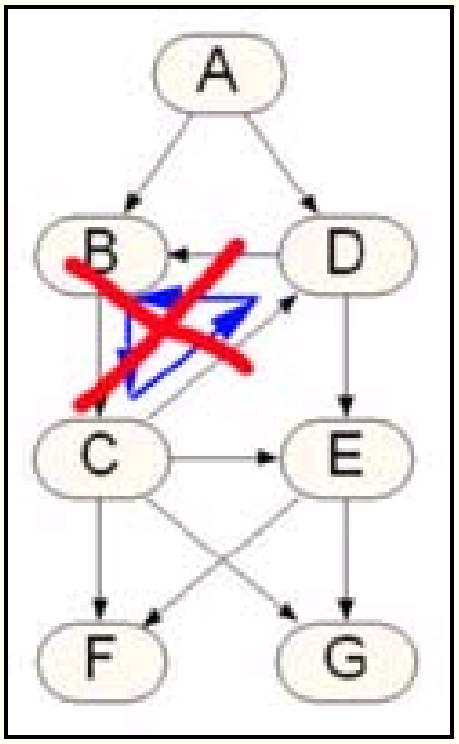
- 步骤1：（E, D, HD, Hb, CP, BP）
- 步骤2到7：
  - P(D|E)化简为P(D)
  - P(HD|E, D)不能化简
  - P(Hb|HD, E, D)化简为P(Hb|D)
  - P(CP|Hb, HD, E, D)化简为P(CP|Hb, HD)
  - P(BP|CP, Hb, HD, E, D)化简为P(BP|HD)

Bayes置信网络

- 拓扑结构中不包含环



A valid Bayes net



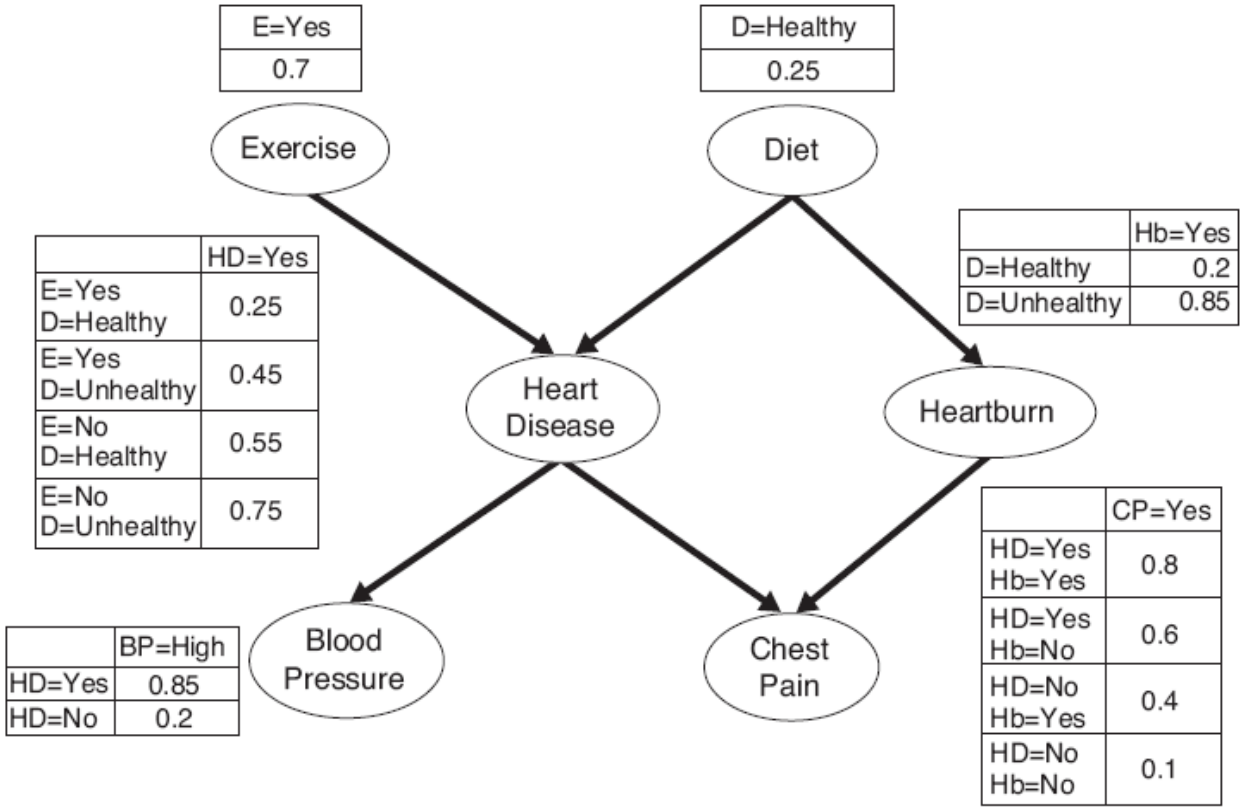
Not a Bayes net

Bayes置信网络

- 如何利用BBN来诊断一个人是否患有心脏病

$$P(HD = Yes) = \sum_{a}\sum_{\beta} P(HL$$
$$= \sum_{a}\sum_{\beta} P(HL$$
$$= 0.25 \times 0.7 \times$$
$$= 0.49$$

$$P(HD = No) = 1 - P(H$$



Bayes置信网络

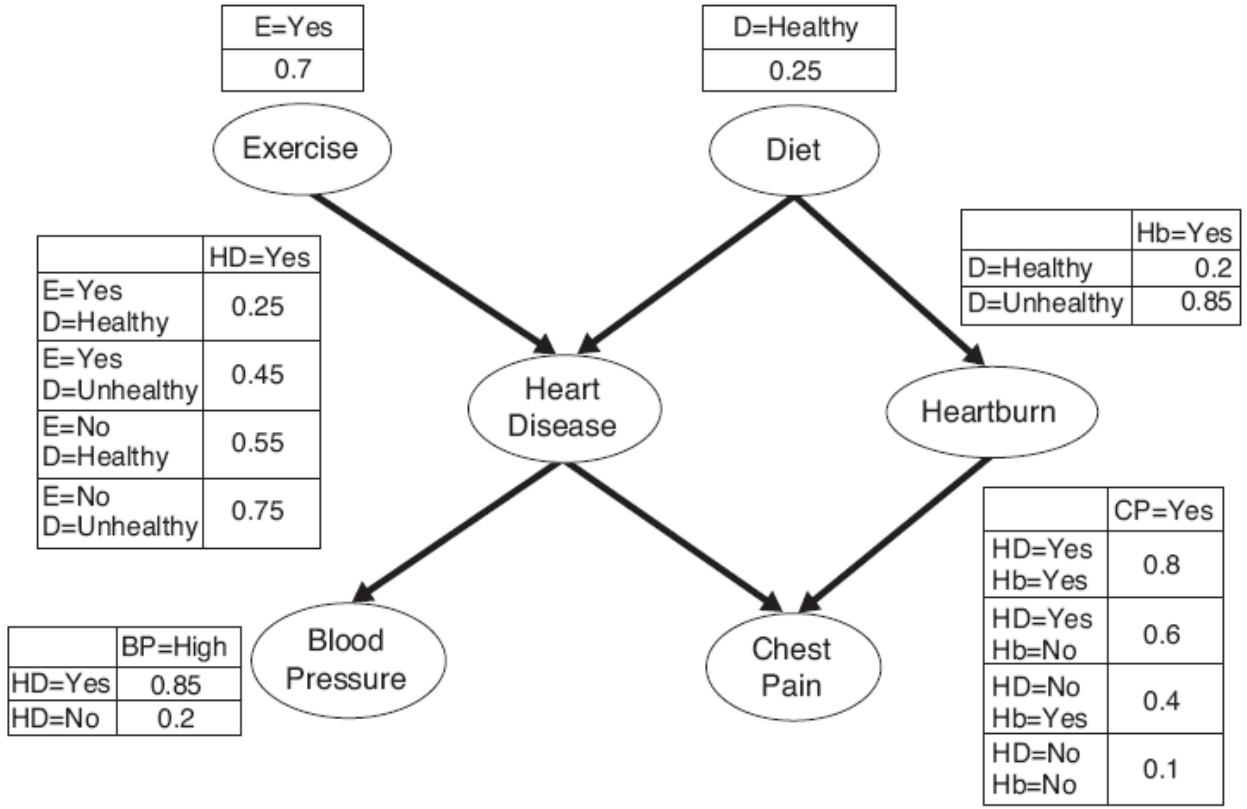
- 当有先验信息时
  - 患有高血压情况下，患心脏病的后验概率是？

$$P(HD = Yes \mid BP$$

$$P(HD = No \mid BP$$

$$P(BP = 高) = \sum_{\gamma}$$

$$= 0.8$$



Bayes置信网络

➢情况三：高血压，饮食健康，经常锻炼身体

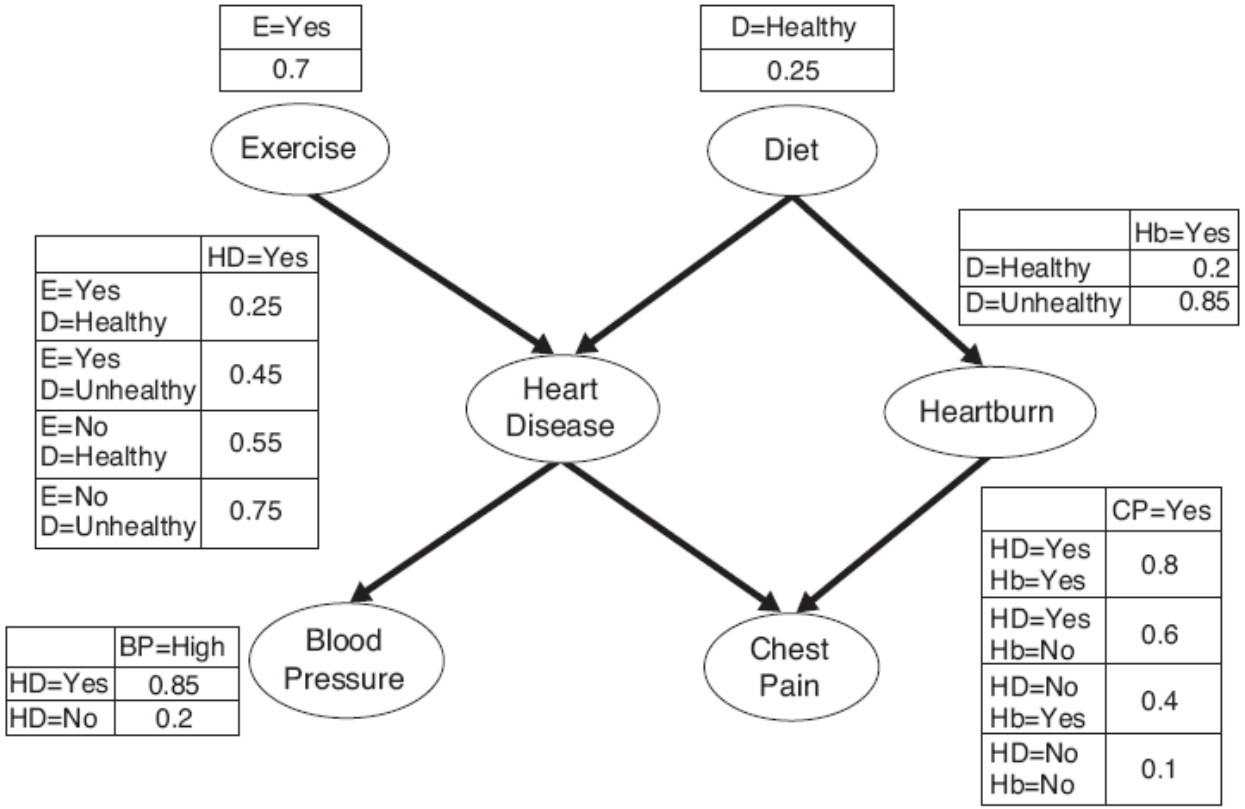
$$P(HD=Yes \mid BP=高, D=健康, E=Yes)$$

$$=\left[\frac{P(BP=高 \mid HD=Yes,}{P(BP=高 \mid D=}\right.$$

$$=\frac{P(BP=高 \mid HD=Yes)}{\sum_{\gamma} P(BP=高 \mid HD=\gamma}$$

$$=\frac{0.85\times 0.25}{0.85\times 0.25+0.2\times 0.75}$$

$$P(HD=No \mid BP=高, D$$

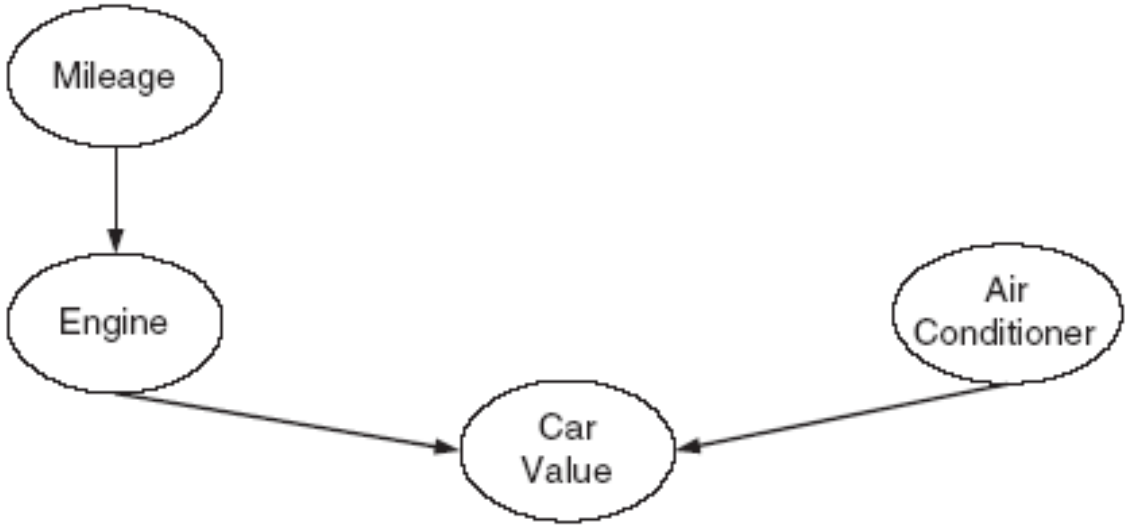


Bayes置信网络

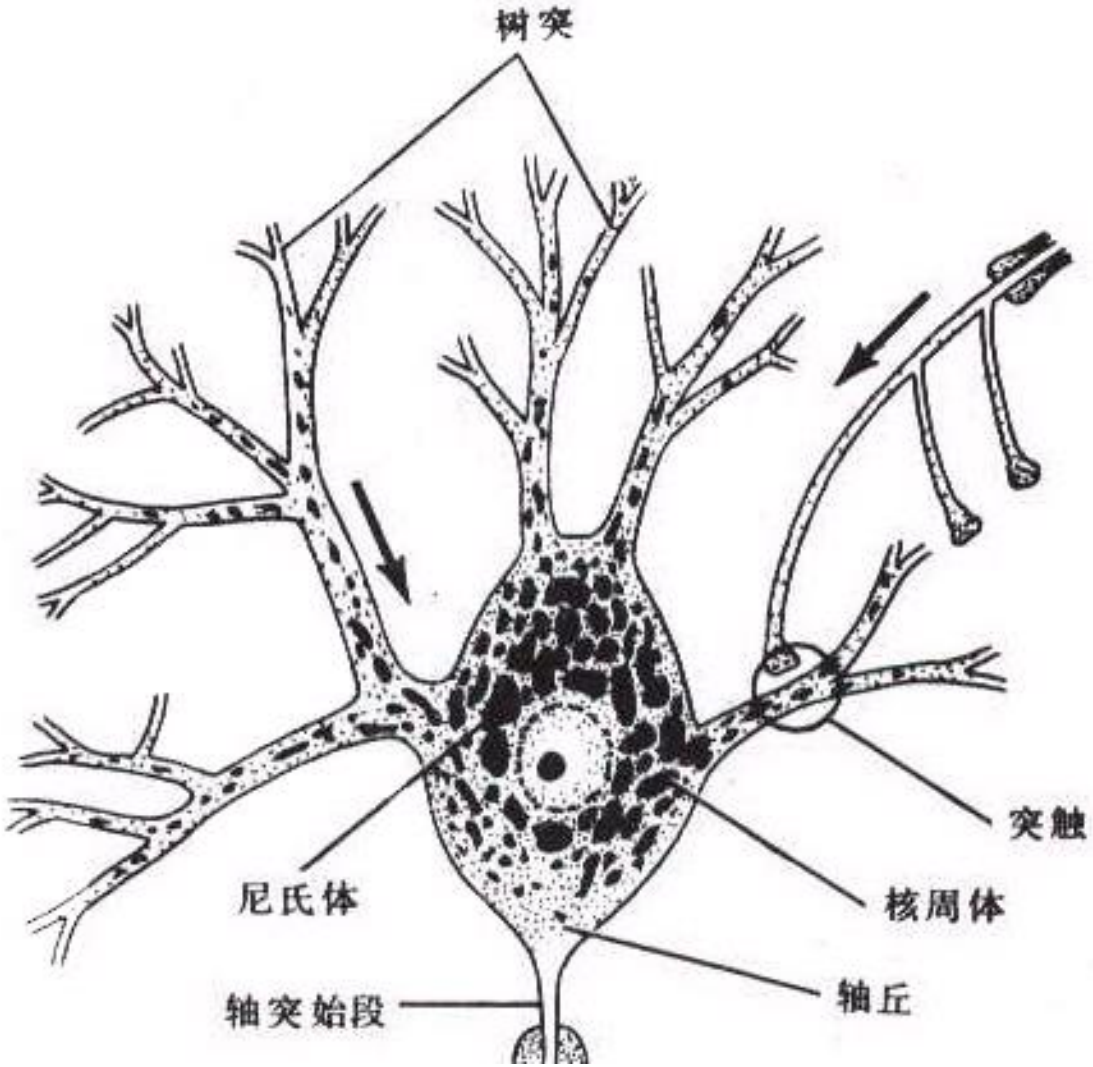
- 习题11
- (a)画出网络中每个结点对应的概率表
- (b)使用贝叶斯网络计算

  P(引擎=差, 空调=不可用)

Mileage	Engine	Air Conditioner	Number of Records with Car Value=Hi	Number of Records with Car Value=Lo
Hi	Good	Working	3	4
Hi	Good	Broken	1	2
Hi	Bad	Working	1	5
Hi	Bad	Broken	0	4
Lo	Good	Working	9	0
Lo	Good	Broken	5	1
Lo	Bad	Working	1	2
Lo	Bad	Broken	0	2

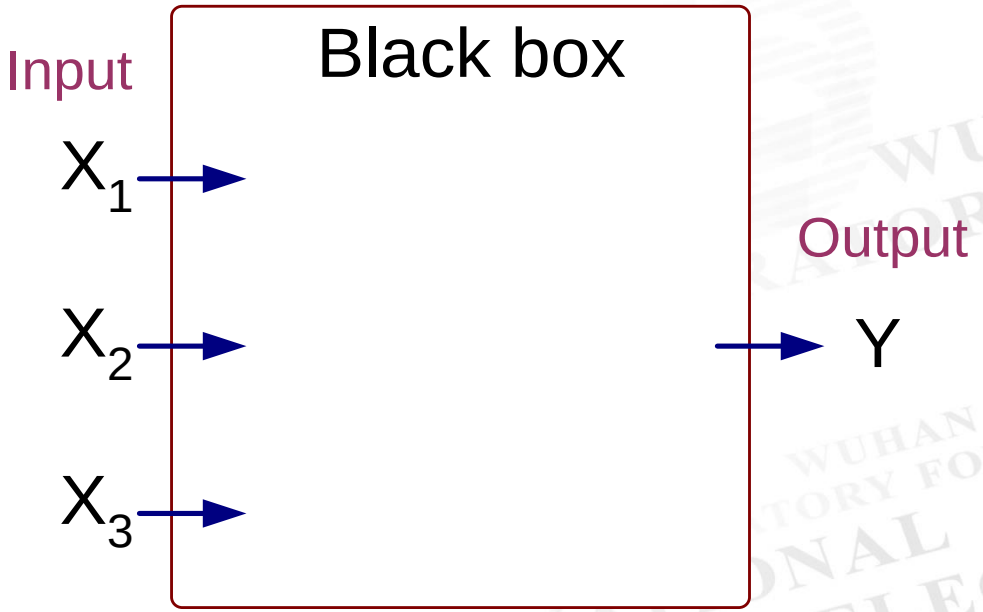


人工神经网络（ANN）



人工神经网络（ANN）

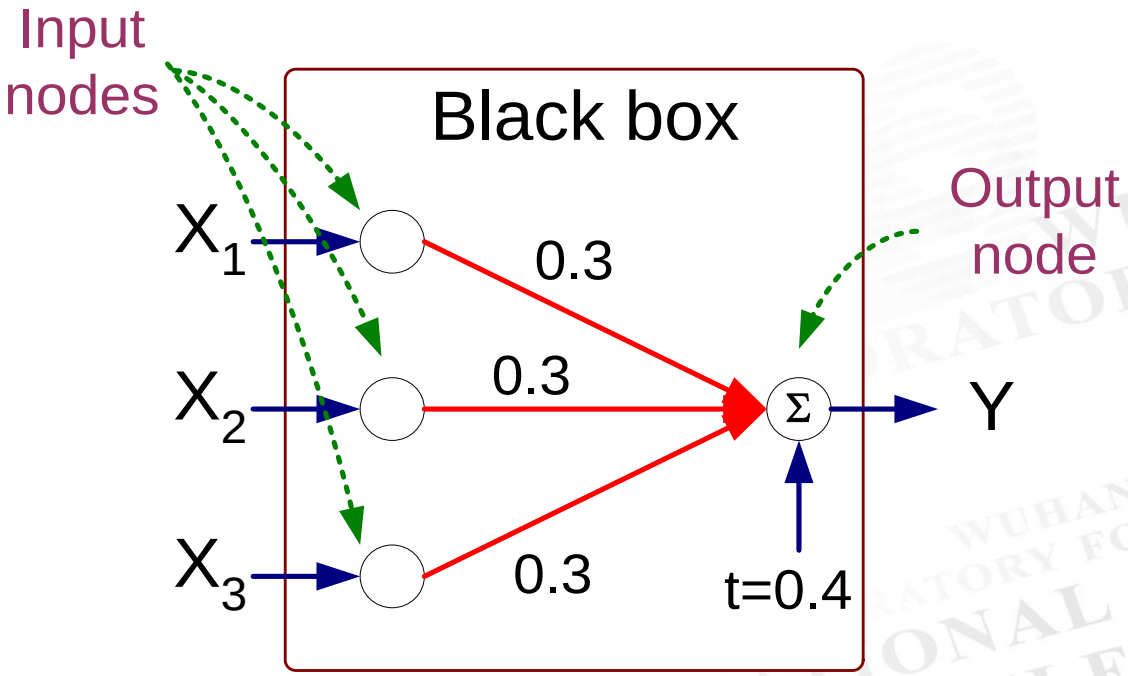
X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0



当有两个或两个以上的输入为1时，输出为1；否则为0

人工神经网络（ANN）

X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0



$$Y = I(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4 > 0)$$

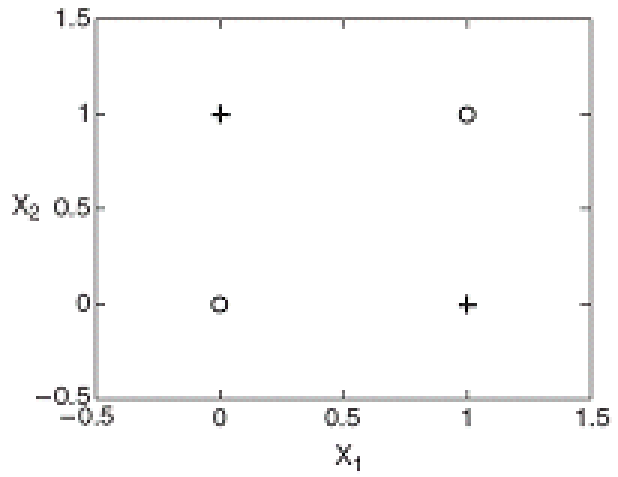
$$\text{where } I(z) = \begin{cases} 1 & \text{if } z \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

$$\hat{y} = \text{sign}[w_d x_d + w_{d-1} x_{d-1} + \ldots + w_1 x_1 + w_0 x_0] = \text{sign}(\mathbf{w} \bullet \mathbf{x})$$

人工神经网络（ANN）

- 学习率
  - 0和1之间
  - 自适应调整

X_1	X_2	y
0	0	−1
1	0	1
0	1	1
1	1	−1



感知器学习算法

1: 令 $D=\{(x_i, y_i)|i=1,2,\ldots,N\}$是训练样例集

2: 用随机值初始化权值向量 $\mathbf{w}^{(0)}$

3: repeat

4: for 每个训练样例$(x_i, y_i)\in D$ do

5: 计算预测输出 $\hat{y}_i^{(k)}$

6: for 每个权值 w_j do

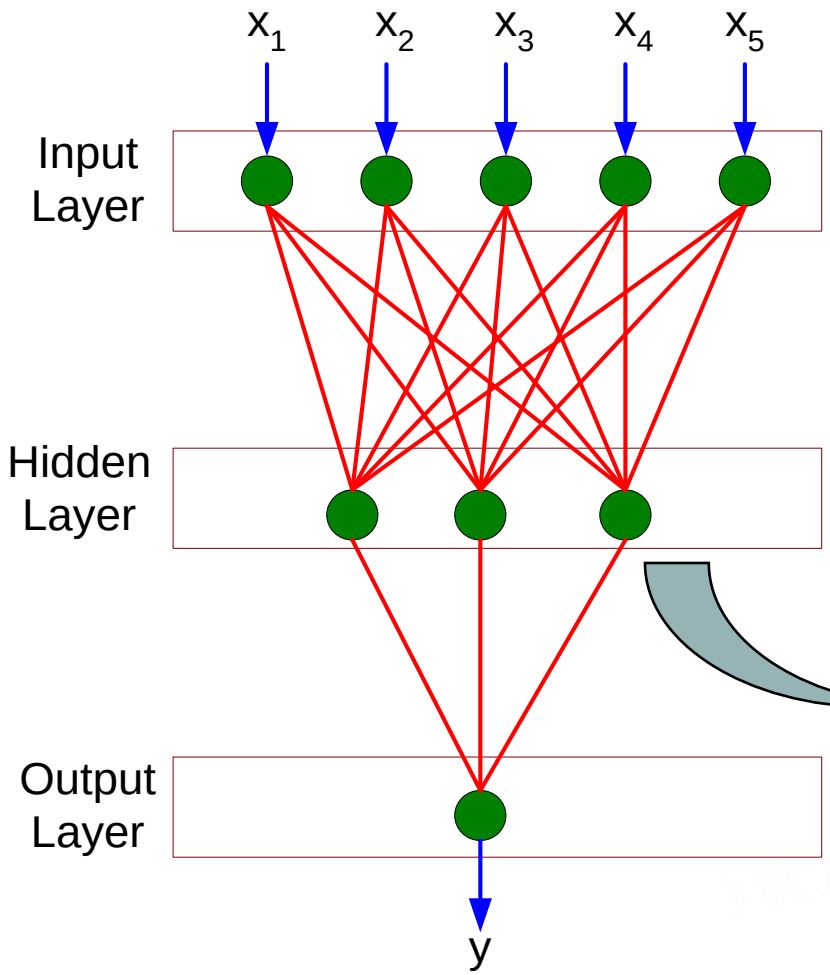
7: 更新权值 $w_j^{(k+1)}=w_j^{(k)}+\lambda(y_i-\hat{y}_i^{(k)})x_{ij}$

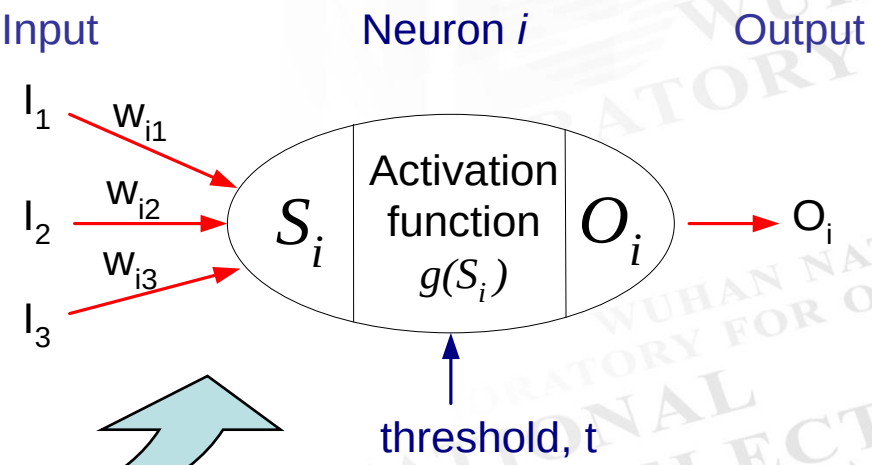
8: end for

9: end for

10: until 满足终止条件

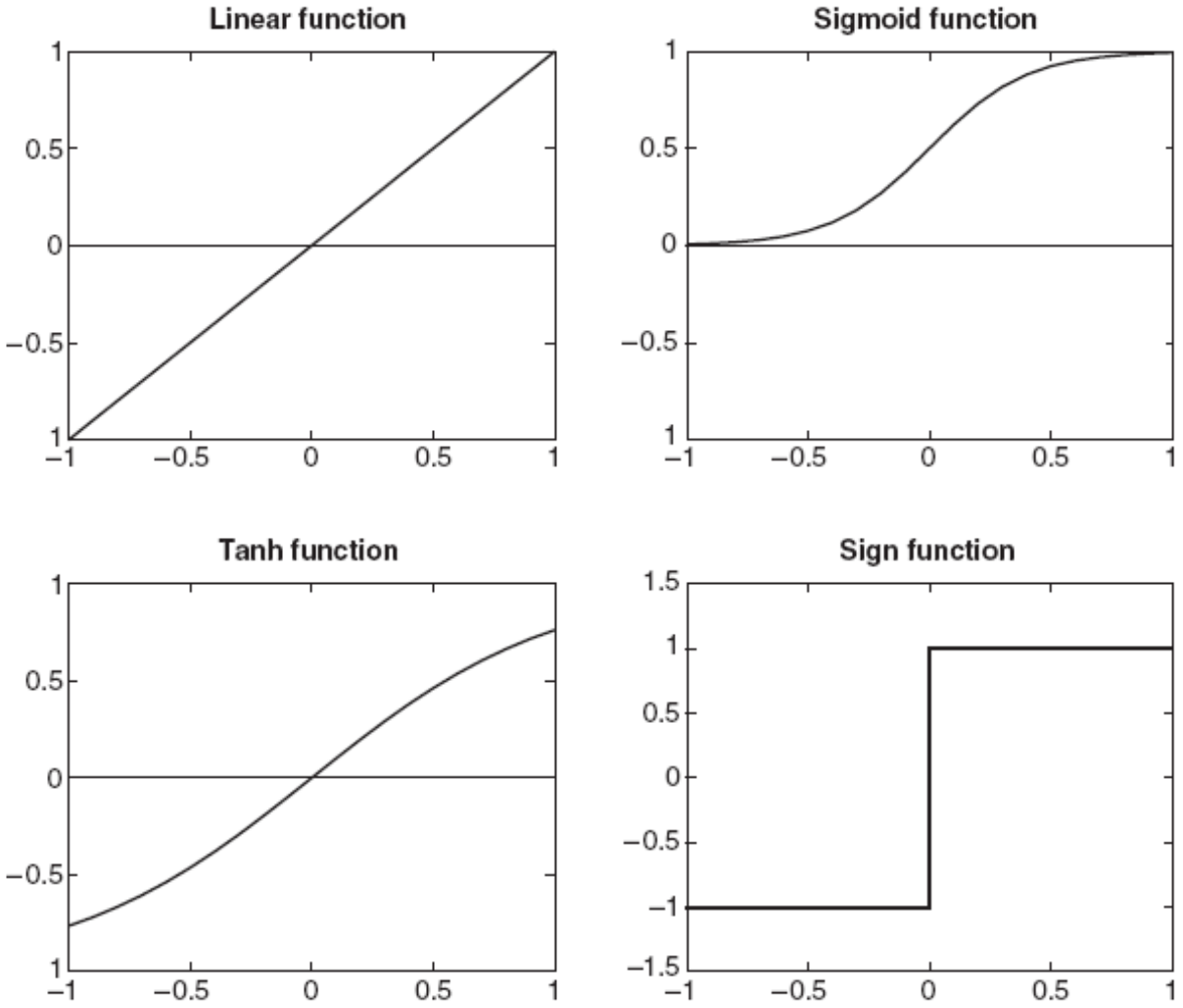
人工神经网络（ANN）



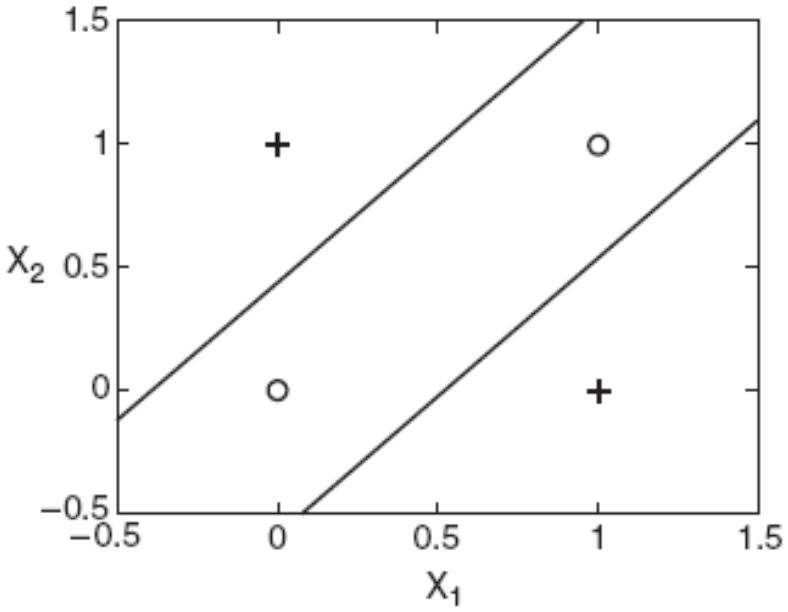


ANN的训练即意味着神经元权重的学习

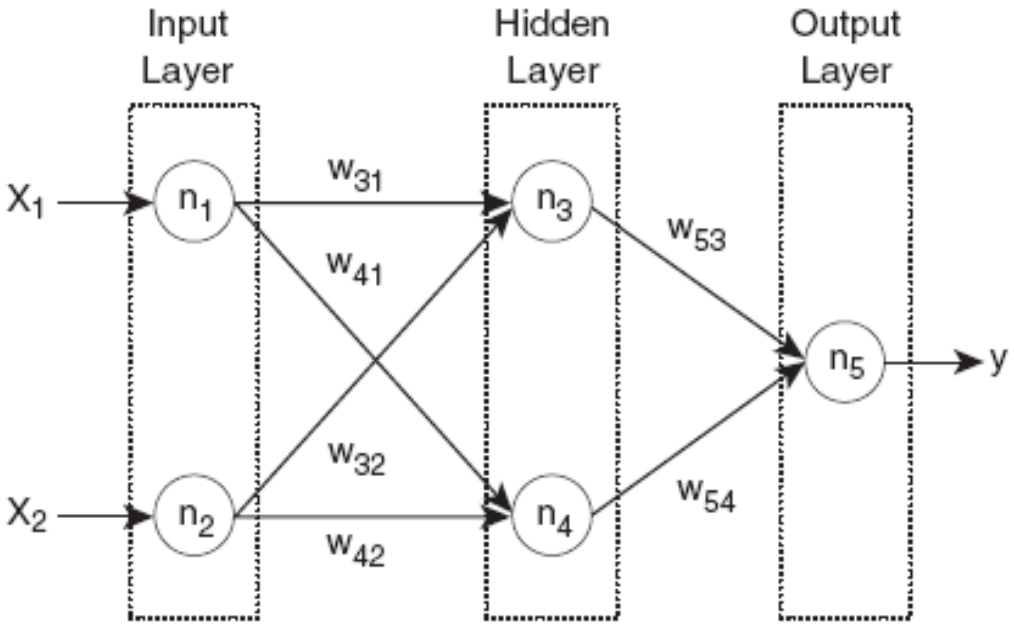
人工神经网络（ANN）


Linear function
Sigmoid function
Tanh function
Sign function

人工神经网络（ANN）



(a) Decision boundary.

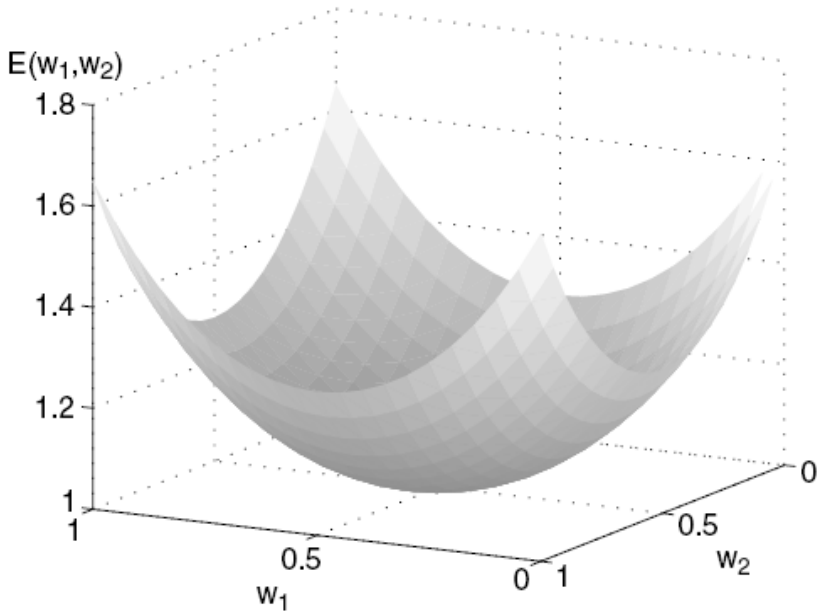


(b) Neural network topology.

人工神经网络（ANN）

- ANN学习算法的目的是确定一组权值，最小化误差的平方和

$$E(w)=\frac{1}{2}\sum_{i=1}^{N}(y_i-\hat{y}_i)^2$$



- 大多数情况下，ANN的输出是参数的非线性函数

$$w_j \leftarrow w_j-\lambda\frac{\partial E(w)}{\partial w_j}$$

反向传播技术：
前向阶段：计算输出
后向阶段：更新权值

人工神经网络（ANN）

- 人工神经网络在本质上是一个多层复合函数

$$y = f_1\left(f_2\left(\ldots f_l(x)\right)\right)$$

- 由于使用了非线性的激活函数，人工神经网络是一个非线性函数
- 是一种有监督的学习算法，既可以用于分类问题，也可以用于回归问题，天然的支持多分类

深度学习的崛起

- 2006年，Hinton等人发现多层前馈神经网络可以先通过逐层训练，再用反向传播算法进行有效学习
- 在语音识别和图像分类等任务上的巨大成功
- 大规模并行计算能力的提升
- 可供机器学习的数据规模也越来越大

深度学习的三个概念

- 反向传播
- 梯度下降
- 损失函数

- 深度学习的基本思想是：先“猜”一个结果，观察它和正确结果之间的差距，然后调整策略，向正确的方向靠近。反复多次，直到预测结果和真实结果足够接近，则训练结束。

深度学习的崛起

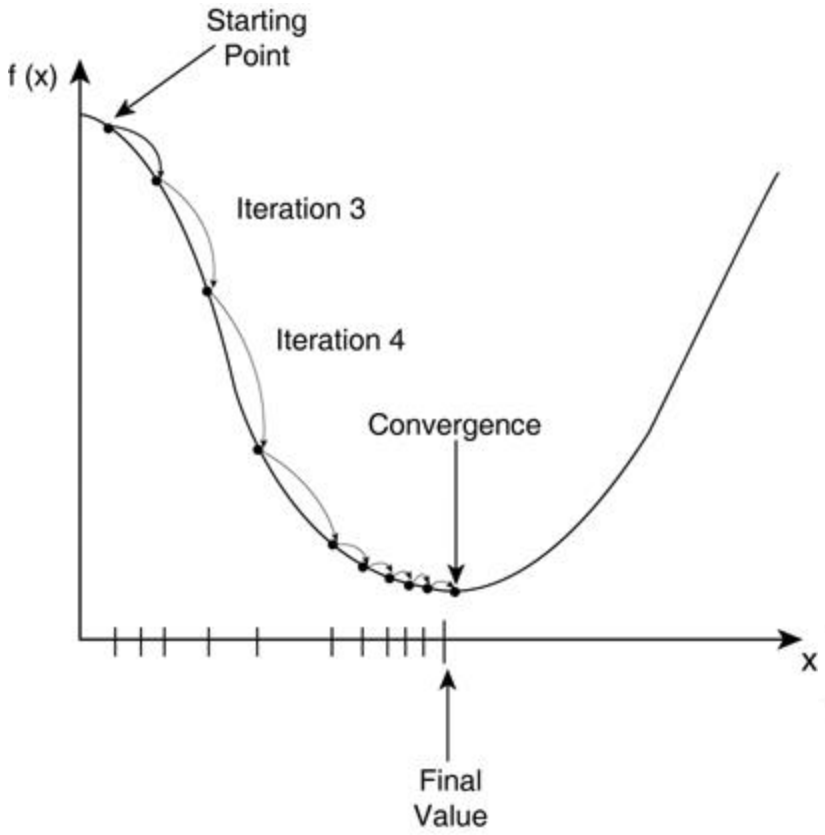
- 例：甲乙两人玩猜数的游戏，乙在心里想好一个数，让甲来猜。
- 目的：猜到乙想好的数字
- 初始化：甲猜5
- 前向计算：甲每次猜新的数字
- 损失函数：乙根据甲猜的数字，得出“大了”或“小了”的结论
- 反向传播：乙告诉甲“大了”或“小了”
- 梯度下降：甲根据乙的反馈调整下一轮的猜测值

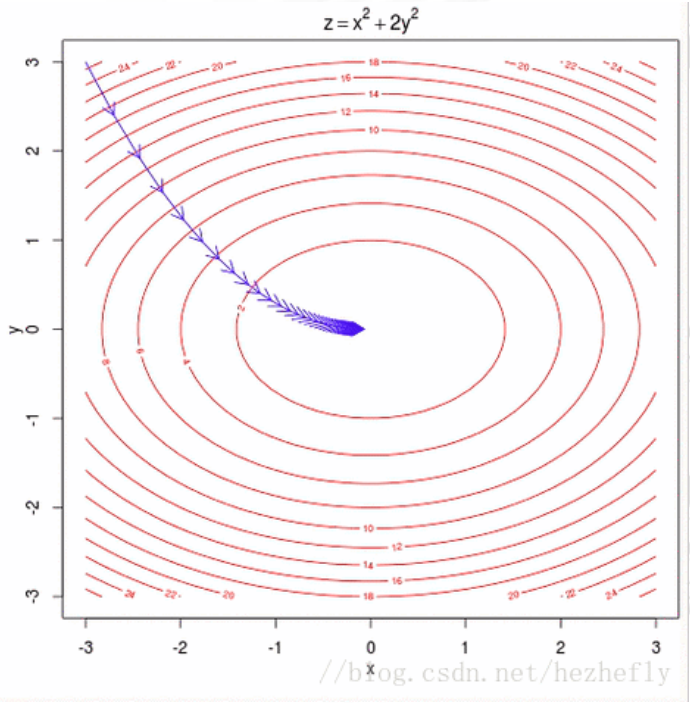
深度学习的崛起

- 初始化
- 前向计算
- 损失函数：提供了计算损失的方法
- 梯度下降：在损失函数基础上向着损失最小的点靠近，从而指导网络权重调整的方向
- 反向传播：将损失值反向传递到神经网络的各层，让各层调整权重
- 重复该过程，直到精度满足要求

深度学习的崛起

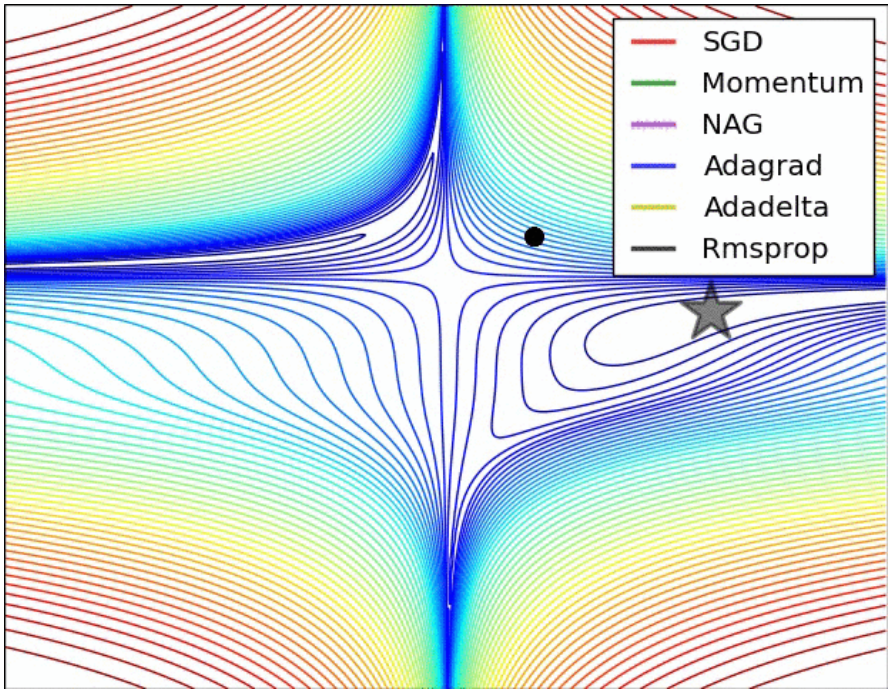
- 梯度下降法

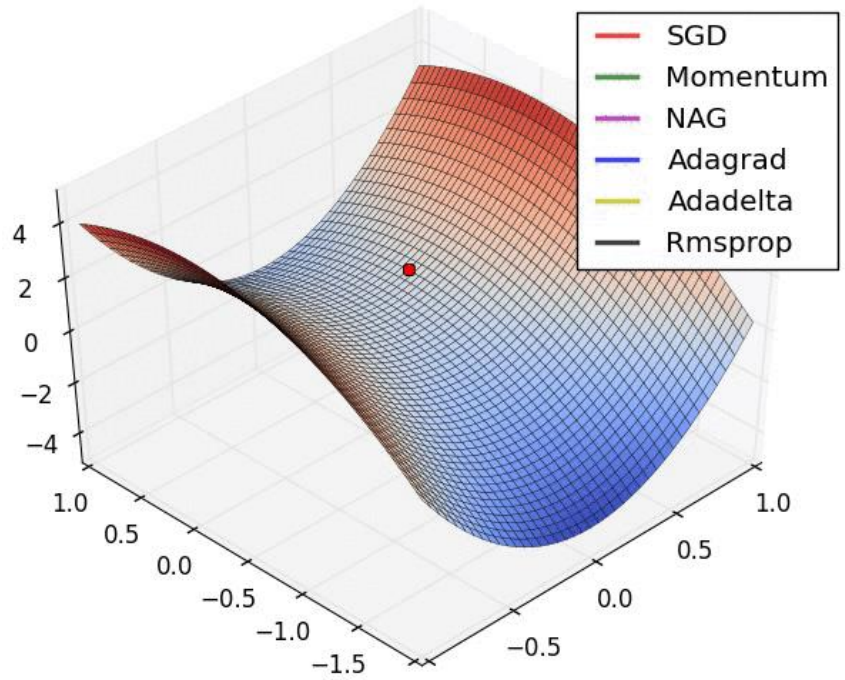




深度学习的崛起

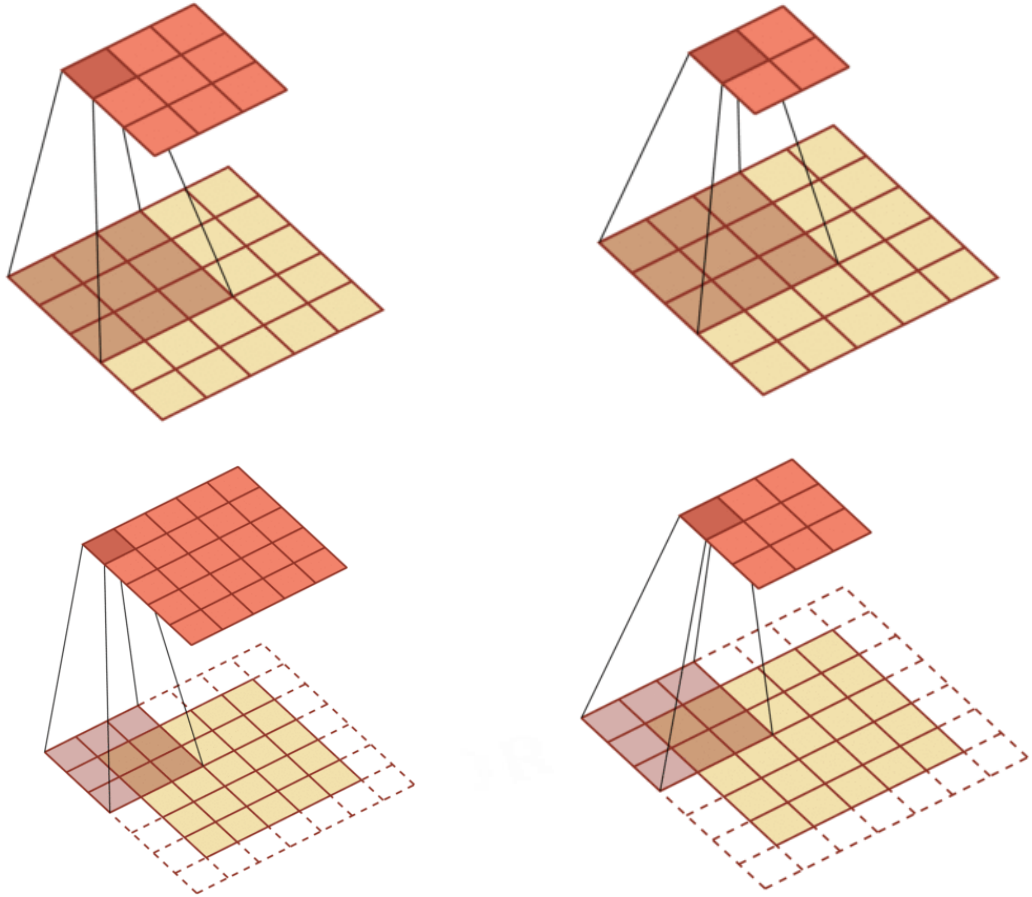
- 不同优化算法的比较





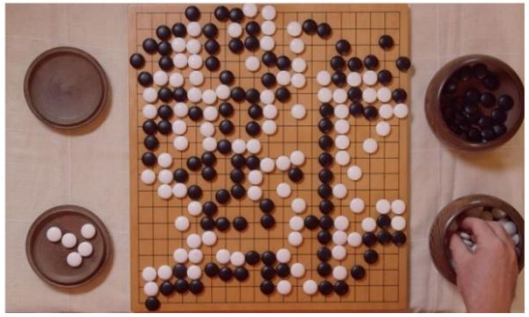
深度学习的崛起

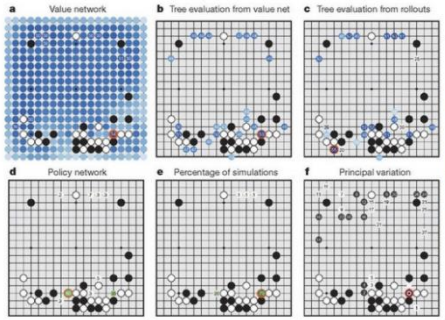
- 卷积神经网络



卷积神经网络的应用

- **AlphaGo**





policy network:
[19x19x48] Input
CONV1: 192 5x5 filters , stride 1, pad 2 => [19x19x192]
CONV2..12: 192 3x3 filters, stride 1, pad 1 => [19x19x192]
CONV: 1 1x1 filter, stride 1, pad 0 => [19x19] *(probability map of promising moves)*

- 分布式系统：**1202** 个**CPU** 和**176** 块**GPU**
- 单机版：**48** 个**CPU** 和 **8** 块**GPU**
- 走子速度：**2** 微秒 **- 3** 毫秒

卷积神经网络的应用

➢ Mask RCNN

Figure 4. More results of **Mask R-CNN** on COCO test images, using ResNet-101-FPN and running at 5 fps, with 35.7 mask AP (Table 1).

卷积神经网络的应用

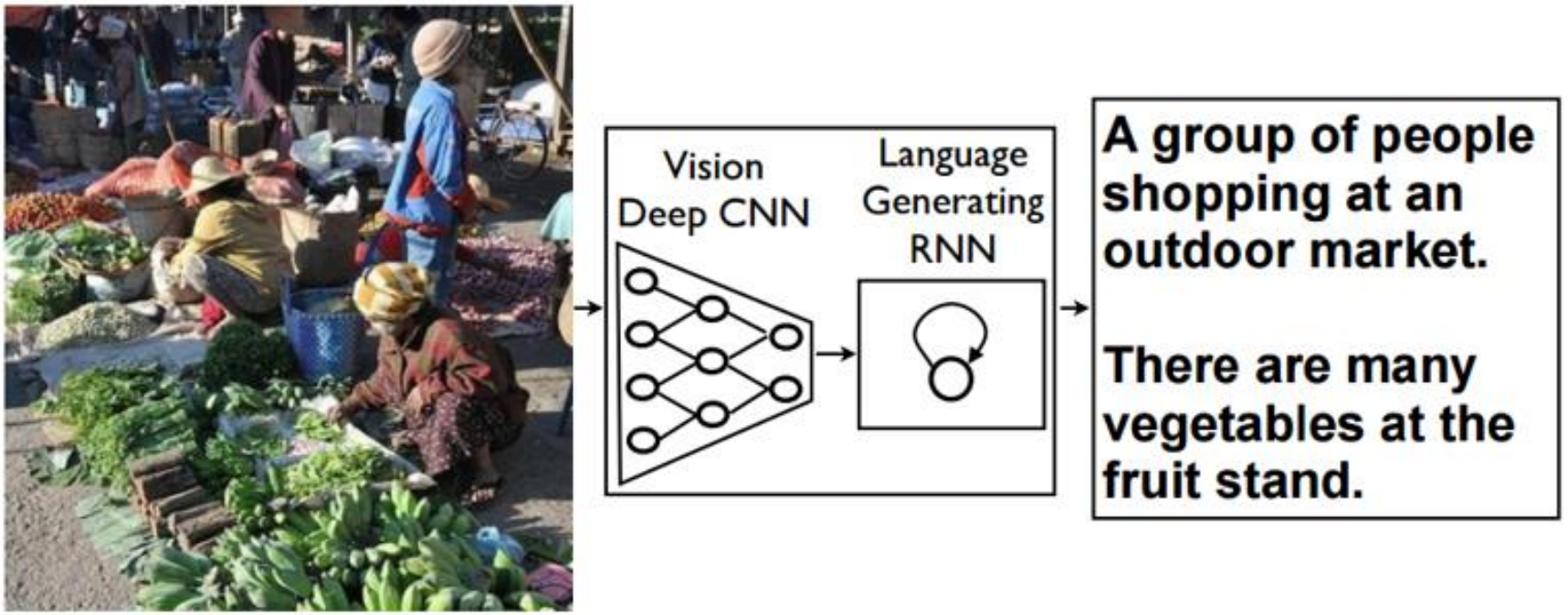
- 画风迁移

卷积神经网络的应用

- **DeepDream**

循环神经网络的应用

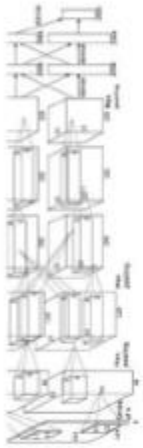
- 看图说话



超级深的网络结

8 layers

16.4%



AlexNet (2012)

19 layers

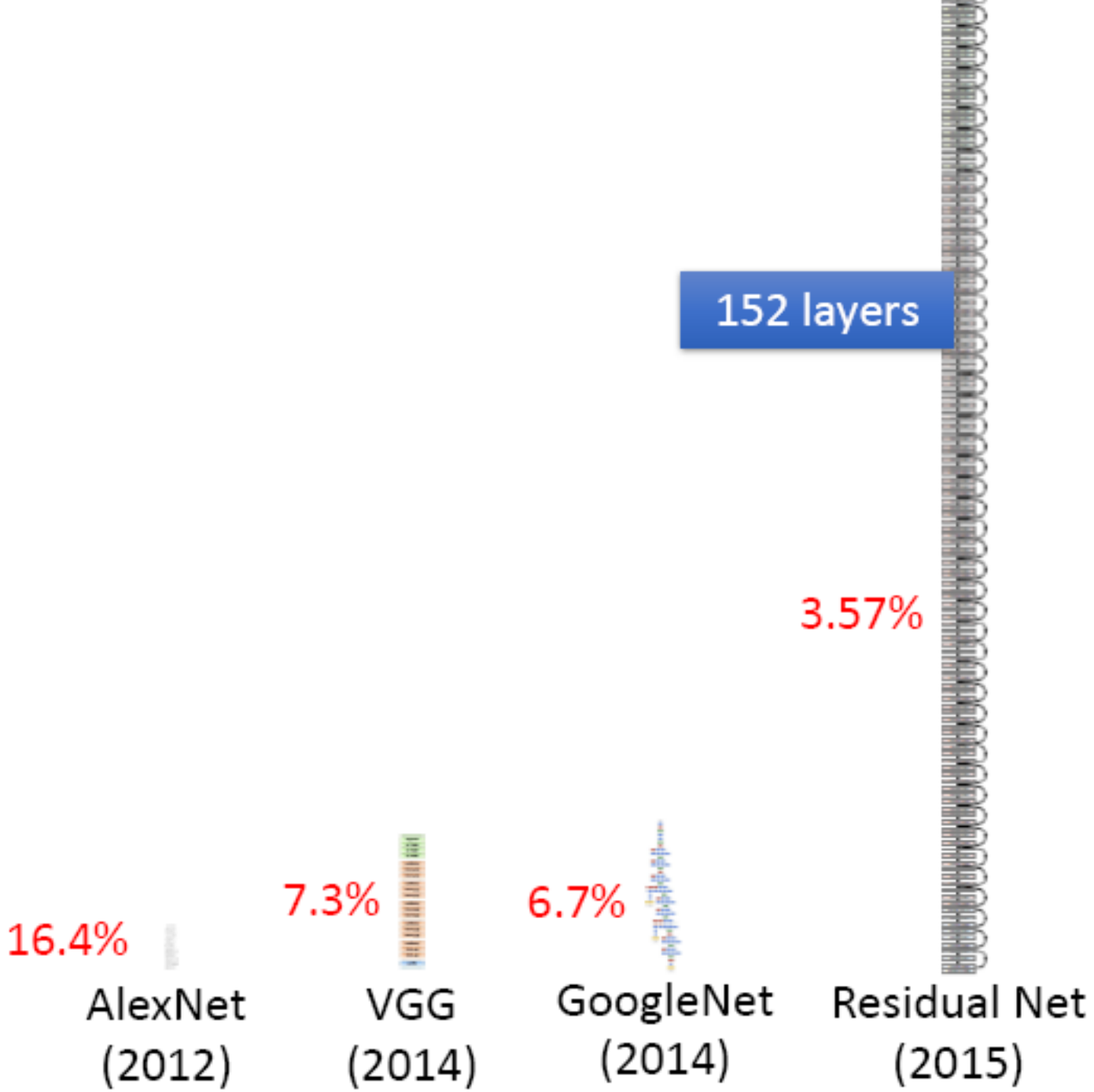
7.3%

VGG (2014)

22 layers

6.7%

GoogleNet (2014)


152 layers
3.57%
16.4%
7.3%
6.7%
AlexNet
(2012)
VGG
(2014)
GoogleNet
(2014)
Residual Net
(2015)

人工神经网络（ANN）

- ANN学习中的设计问题
  - 确定输入层的结点数目
  - 确定输出层的结点数目
  - 选择网络拓扑结构
  - 初始化权值和偏置
  - 去掉有遗漏值的训练样例

人工神经网络（ANN）

- **ANN的特点**
  - 至少含有一个隐藏层的多层神经网络是一种普适近似，即可以用来近似任何目标函数
  - 可以处理冗余特征
  - 训练过程非常耗时

人工神经网络（ANN）

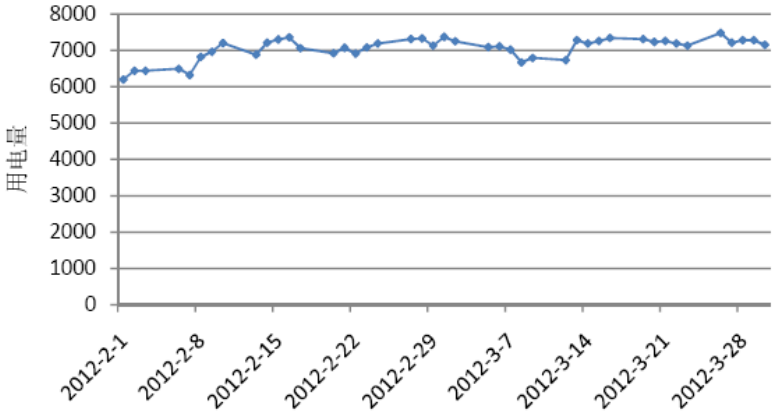
- 习题16
- (a)说明感知器模型怎样表示两个布尔变量之间的AND和OR函数
- (b)使用线性函数作为多层神经网络的激活函数有何缺点

防窃漏电自动诊断

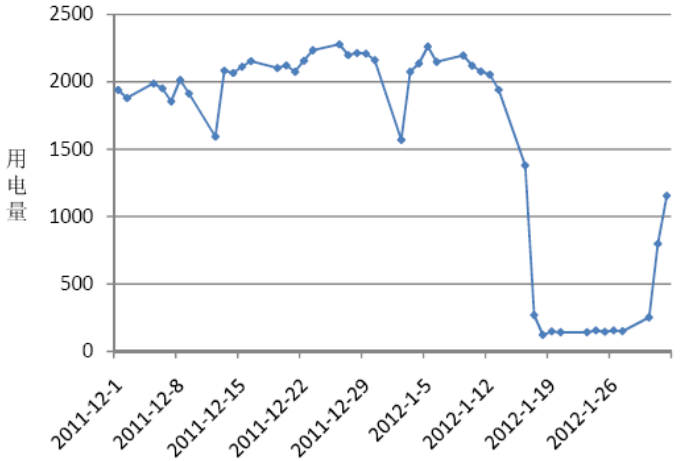
防窃漏电自动诊断

- 传统方法通过定期巡检来发现窃电或故障
- 目前供电局已建成计量自动化应用平台
  - 发电侧、供电侧、配电侧、售电侧统一数据的自动采集监控
- 数据接口实现了数据共享，存储大量用户信息

用电特征分析

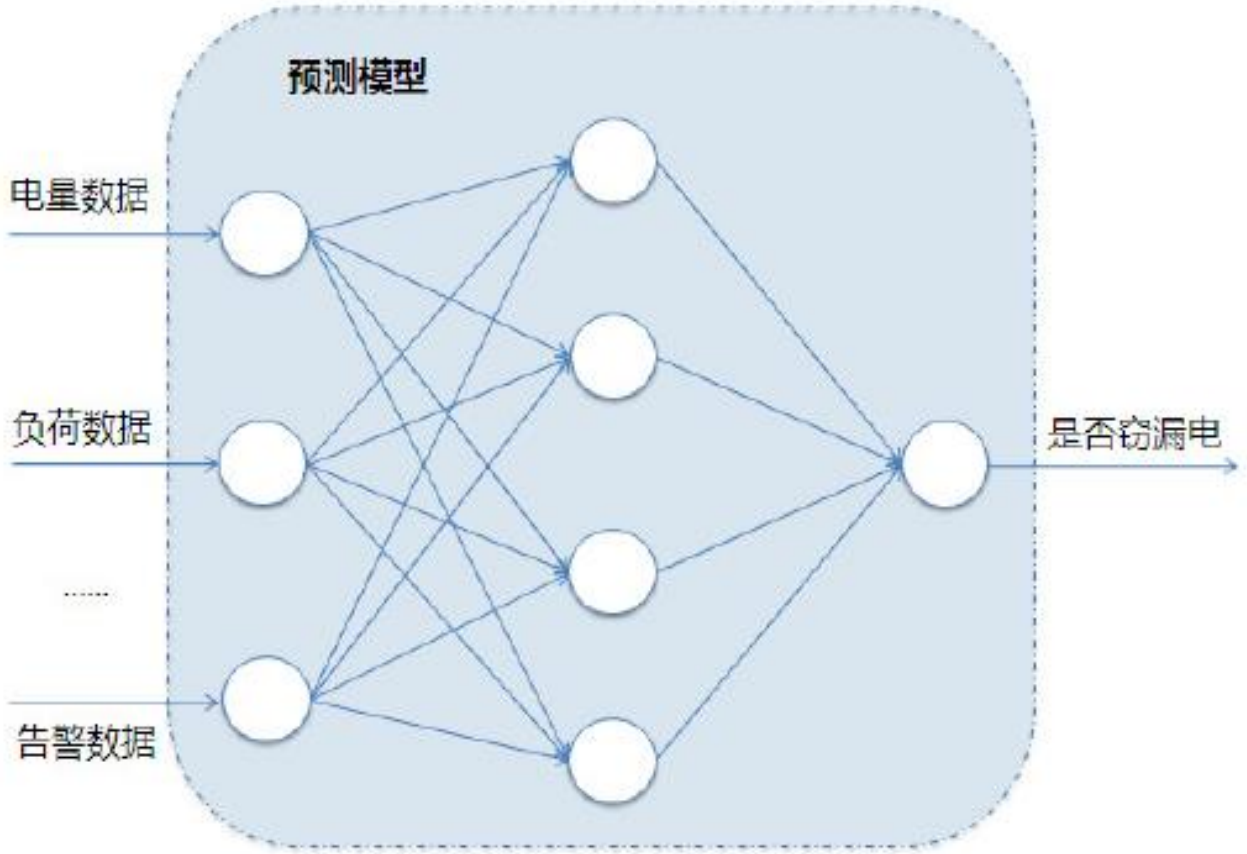


正常用电量走势



窃电电量走势

窃漏电评价模型



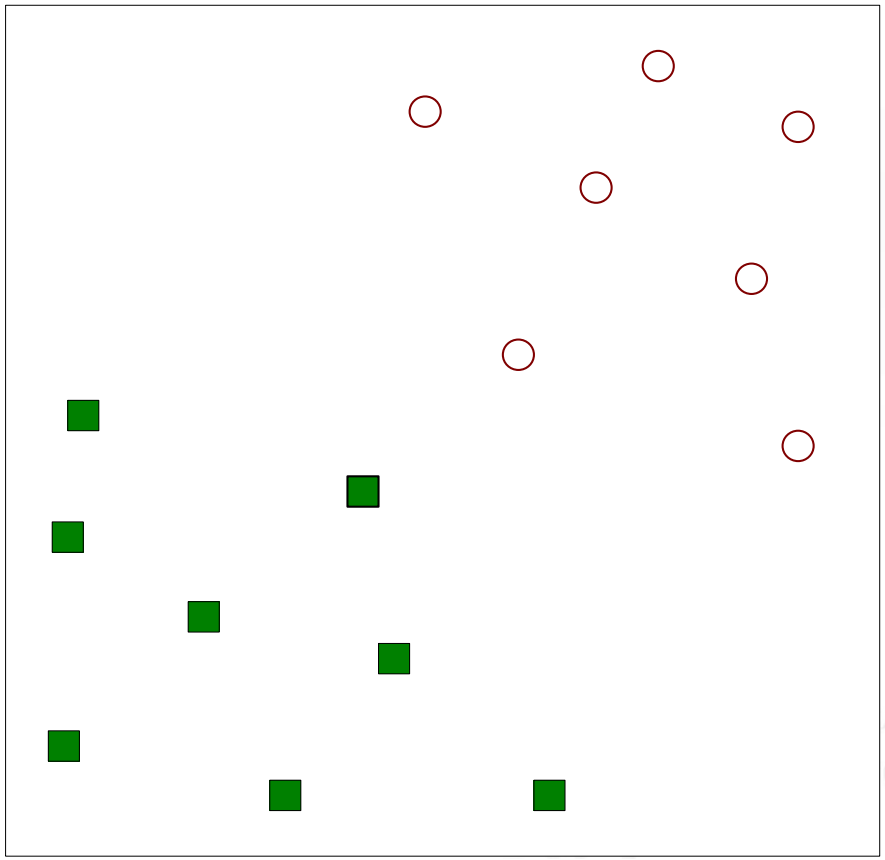
模型构建

- 一、样本数据抽取
  - 电量类、负荷类、线损类、报警类指标
  - 包含近年来所有的窃漏电用户及部分正常用户
  - 窃漏电用户包含关键时间点前后两个月数据
- 二、数据预处理
  - 缺失值处理
  - 节假日数据修正

支持向量机（SVM）

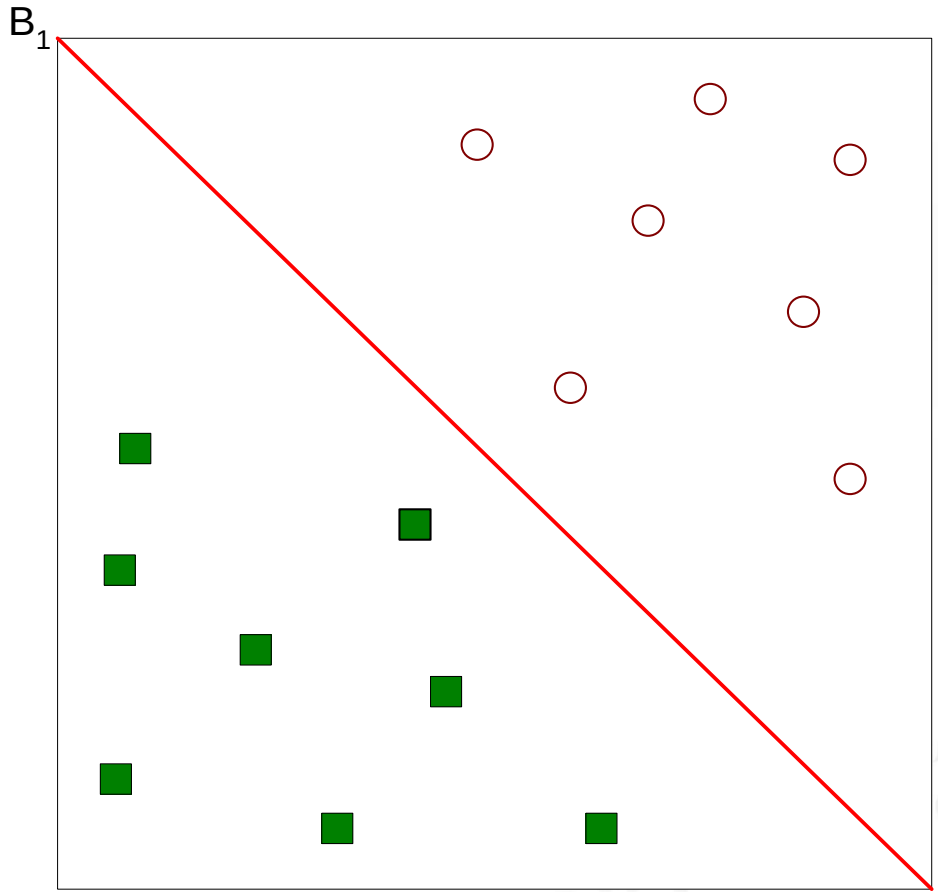
- SVM是由统计学习理论推导出的一种分类器
  - Vapnik和Chervonenkis，1995年
- 使用训练实例的一个子集来表示决策边界
  - 支持向量

支持向量机（SVM）



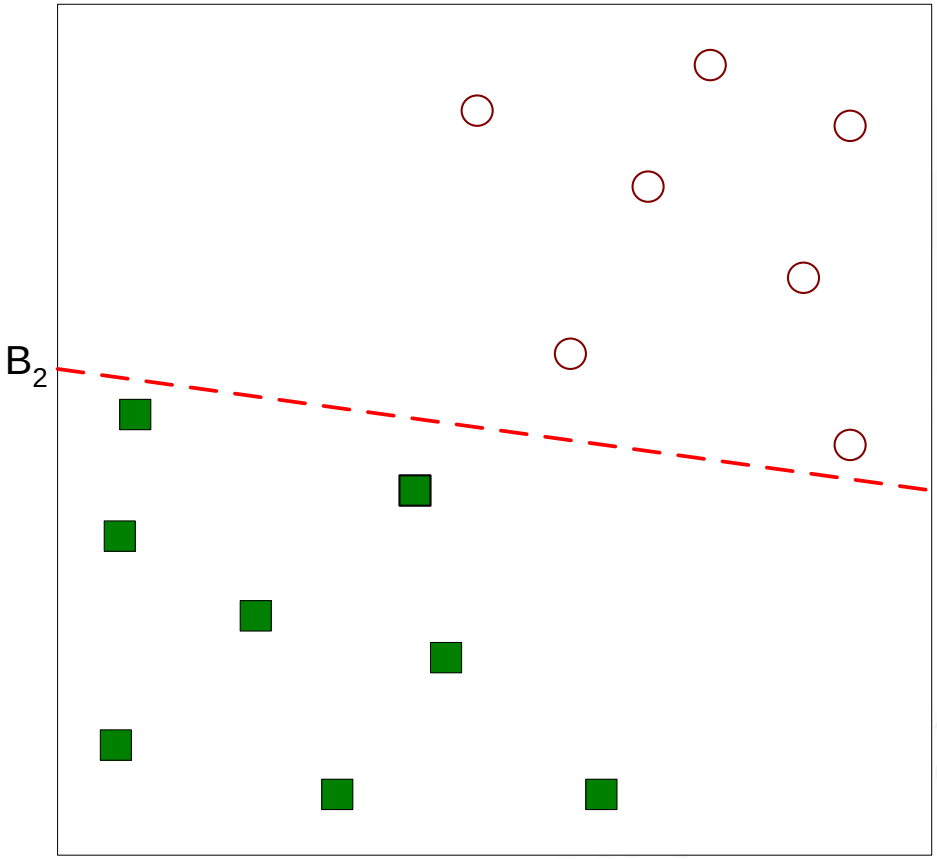
- 找到一个线性超平面（判决边界）来分割训练数据集合

支持向量机（SVM）



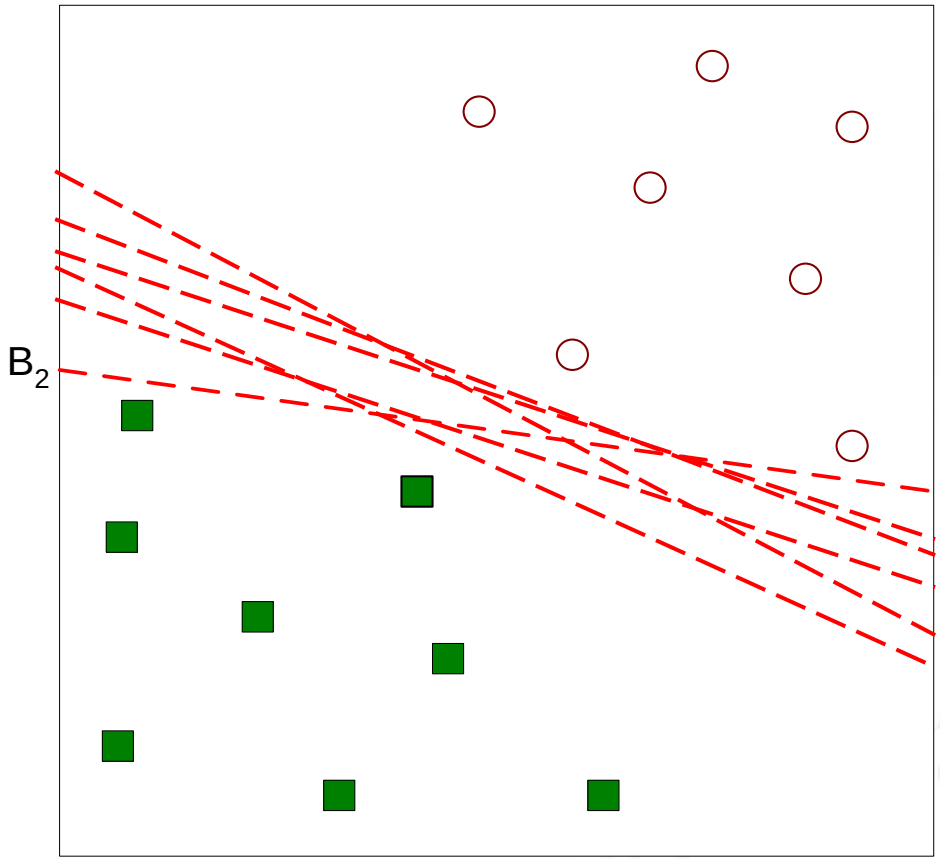
- 一个可能的划分

支持向量机（SVM）



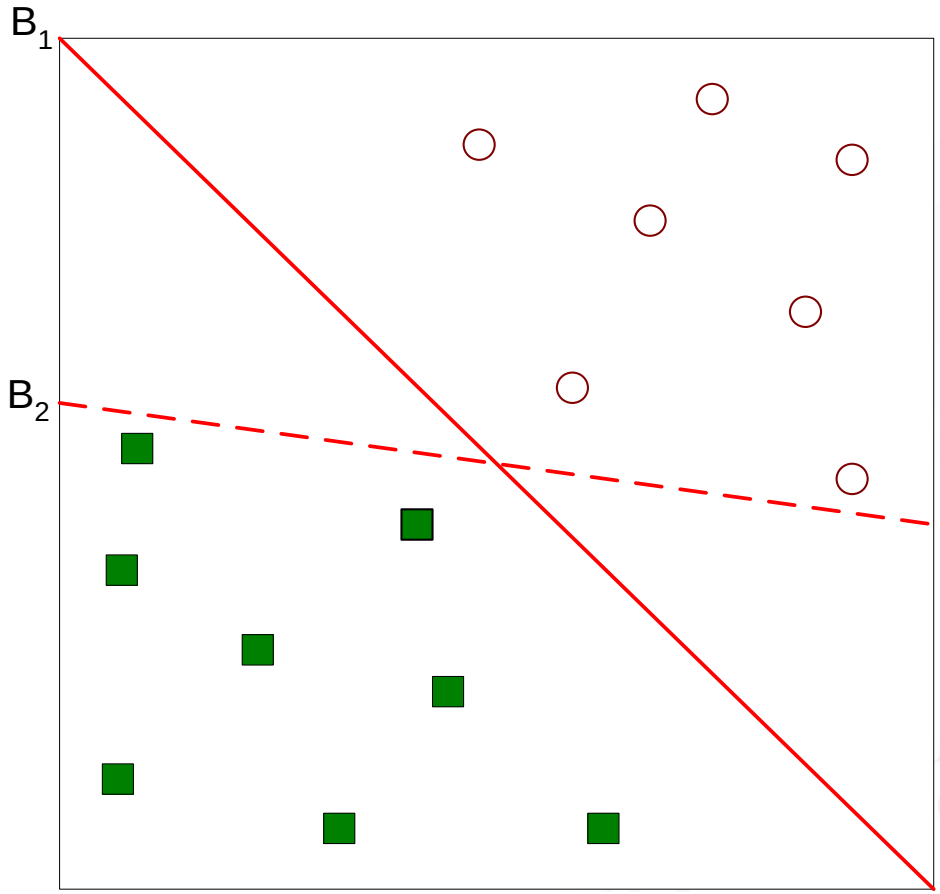
- 另一个可能的划分

支持向量机（SVM）



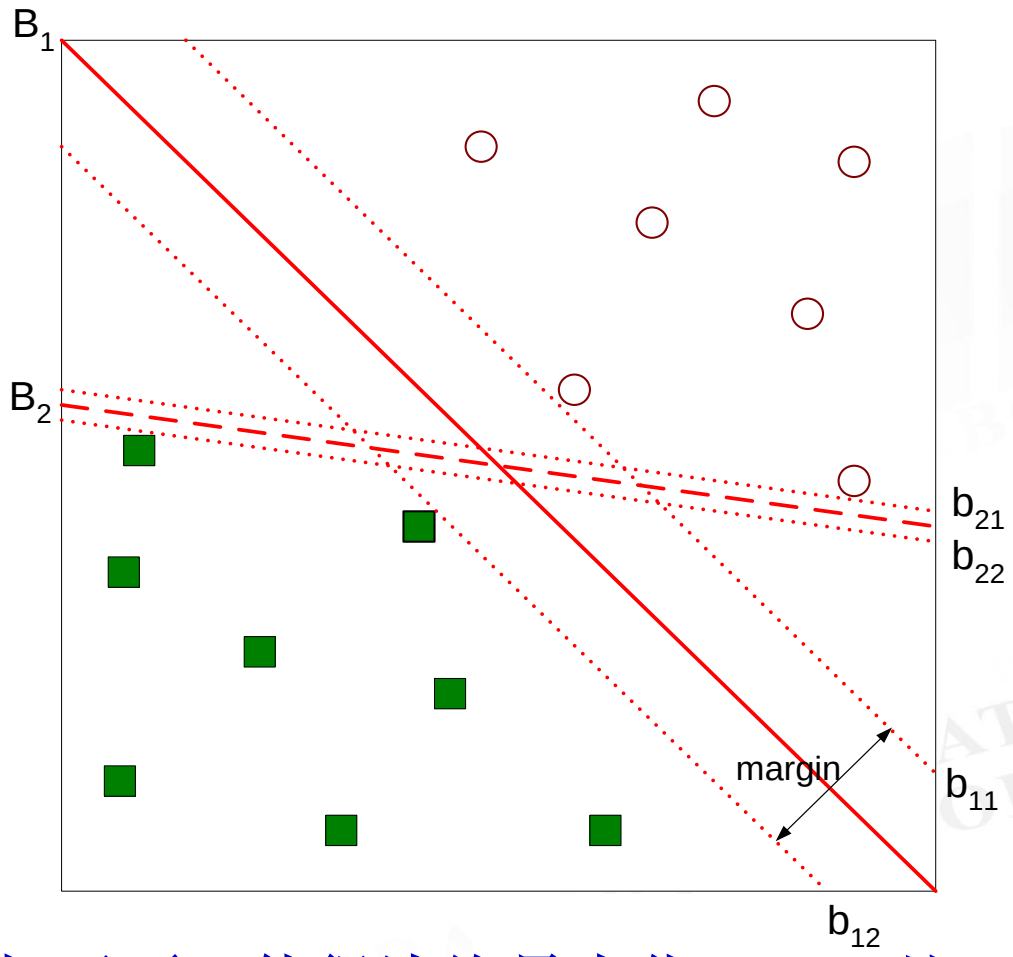
- 其他可能的划分

支持向量机（SVM）



- 哪一个是更好的划分，B_1还是B_2？
- 如何定义“更好的划分”？

支持向量机（SVM）



- 找到一个超平面，使得边缘最大化==> B_1比B_2更好

支持向量机（SVM）

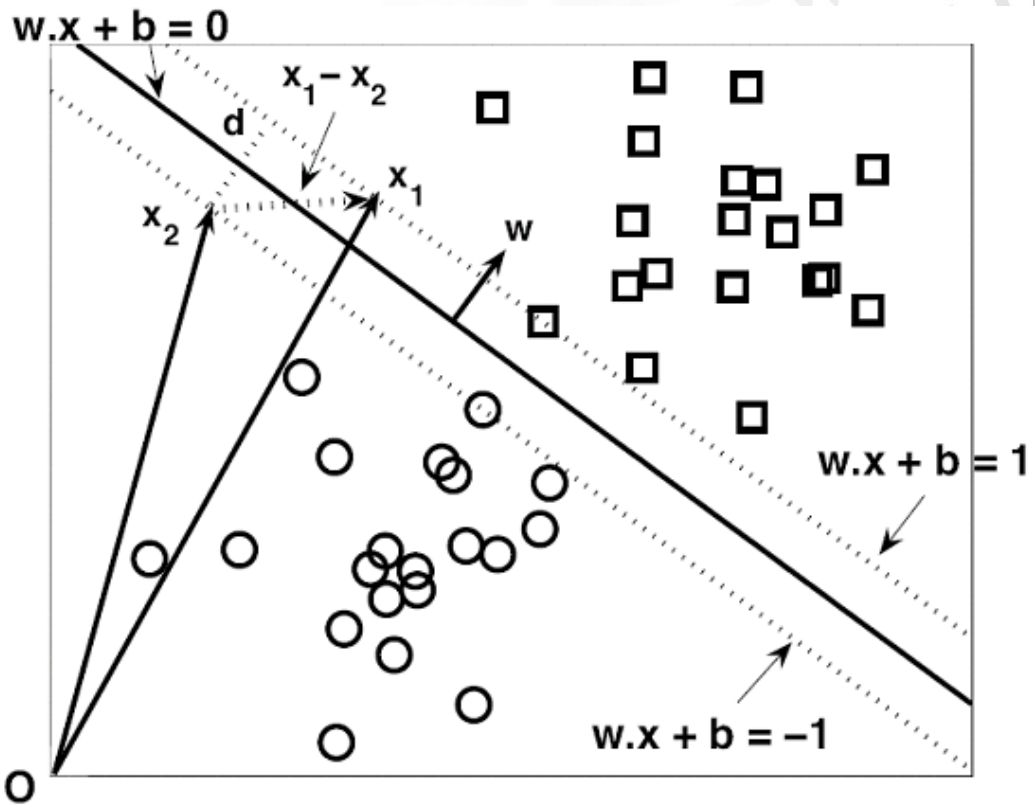
- 具有较大边缘的决策边界具有更好的泛化能力
  - 对于较小的边缘，决策边界任何轻微的扰动都可能对分类产生显著的影响
- 结构风险最小化

$$R \le R_e + \varphi\left(\frac{h}{N}, \frac{\log(\eta)}{N}\right)$$

  - 训练误差和泛化误差之间的折中

支持向量机（SVM）

- 线性SVM：寻找具有最大边缘的超平面，经常被称为最大边缘分类器
  - 样本(x_i, y_i)
  - 属性集(x_{i1},…x_{id})^T
  - 决策边界w*x+b=0
  - 若w*x+b>0，y=1
  - 若w*x+b<0，y=-1



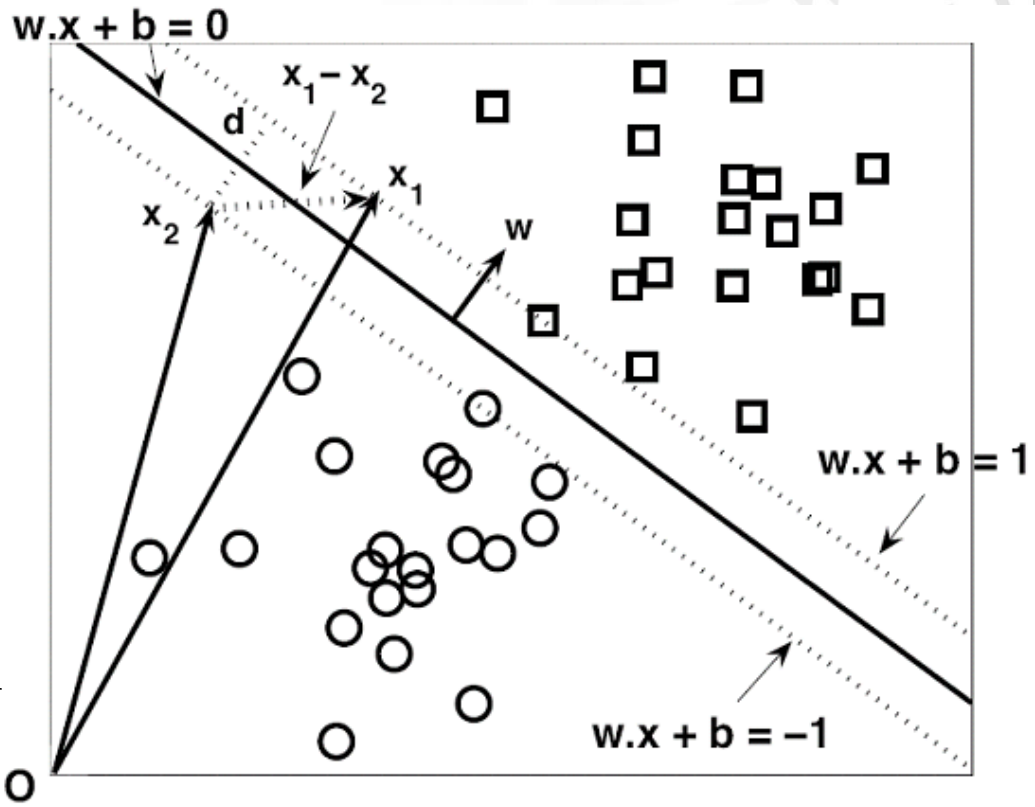
支持向量机（SVM）

- 通过调整参数w和b，两个平行超平面可以表示为

b_{i1}: w*x + b = 1
b_{i2}: w*x + b = -1

- 两超平面间的距离 **(Margin)** $d = \frac{2}{\| w \|}$

$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$$



支持向量机（SVM）

- SVM在训练阶段估计决策边界的参数w和b，满足条件

$$y_i(w \cdot x_i + b) \geq 1, i = 1, 2, ..., N$$

- SVM同时要求决策边界的边缘最大，等价于最小化函数

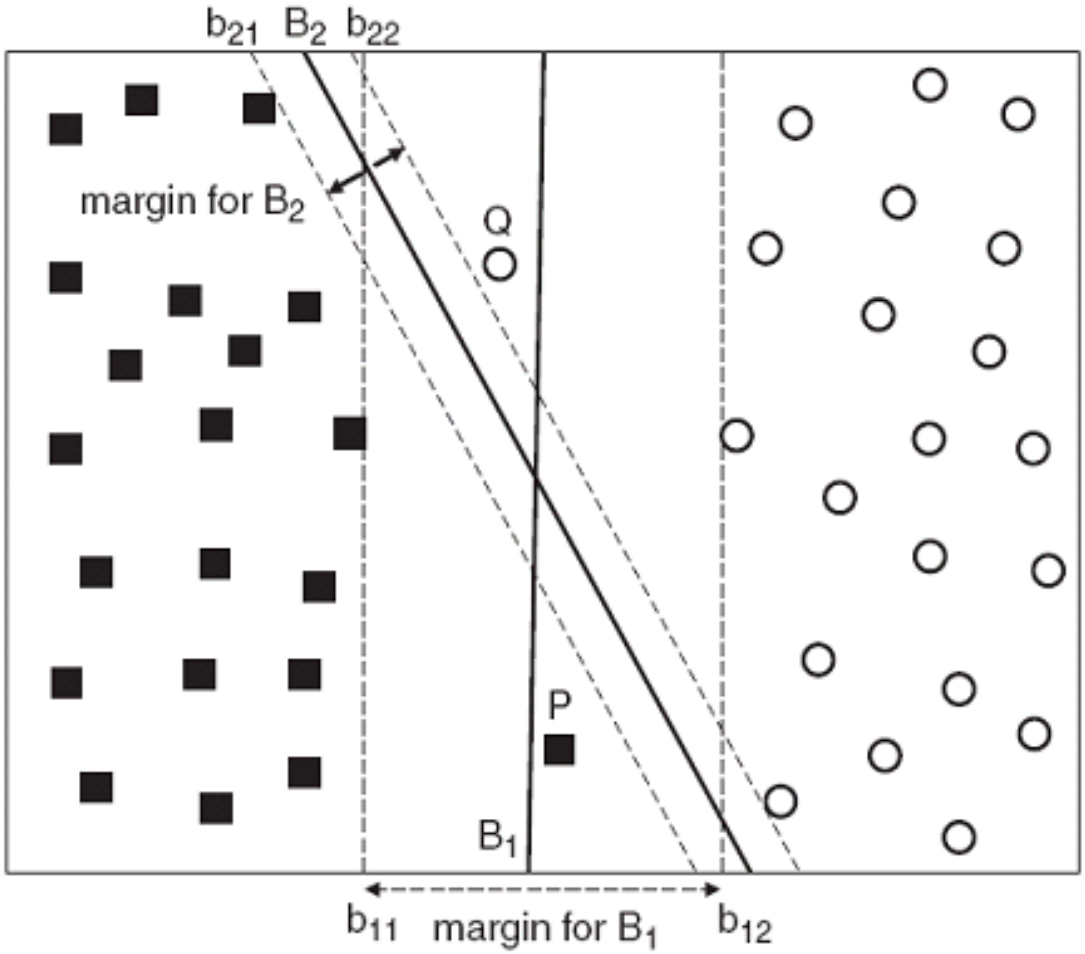
$$f(w) = \frac{\| w \|^2}{2}$$

- minimizing $f(w) = \frac{\| w \|^2}{2}$

Subject to $y_i(w \cdot x_i + b) \geq 1, i = 1, 2, ..., N$

支持向量机（SVM）

- 不可分情况



考虑边缘的宽度与线性决策边界允许的训练错误数目之间的折中

支持向量机（SVM）

- 边缘宽度与允许的训练错误数目之间的折衷

$$f(w)=\frac{\|w\|^2}{2}$$

$$y_i(w\cdot x_i+b)\geq 1,\ i=1,2,...,N$$

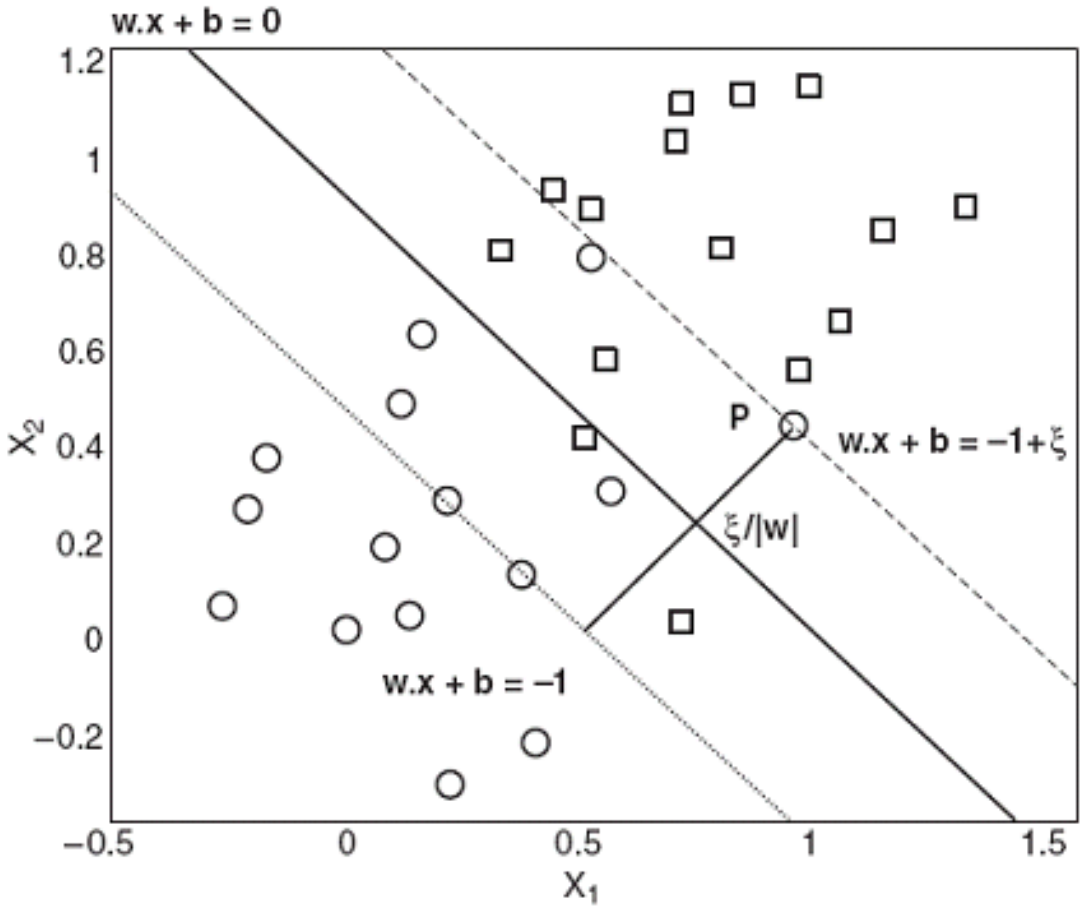
$$\Downarrow$$

$$w\cdot x_i+b\geq 1-\xi_i \qquad if\ \ y_i=1$$

$$w\cdot x_i+b\leq -1+\xi_i \qquad if\ \ y_i=-1$$

$$\xi_i>0$$ 松弛变量（**slack variable**）

支持向量机（SVM）



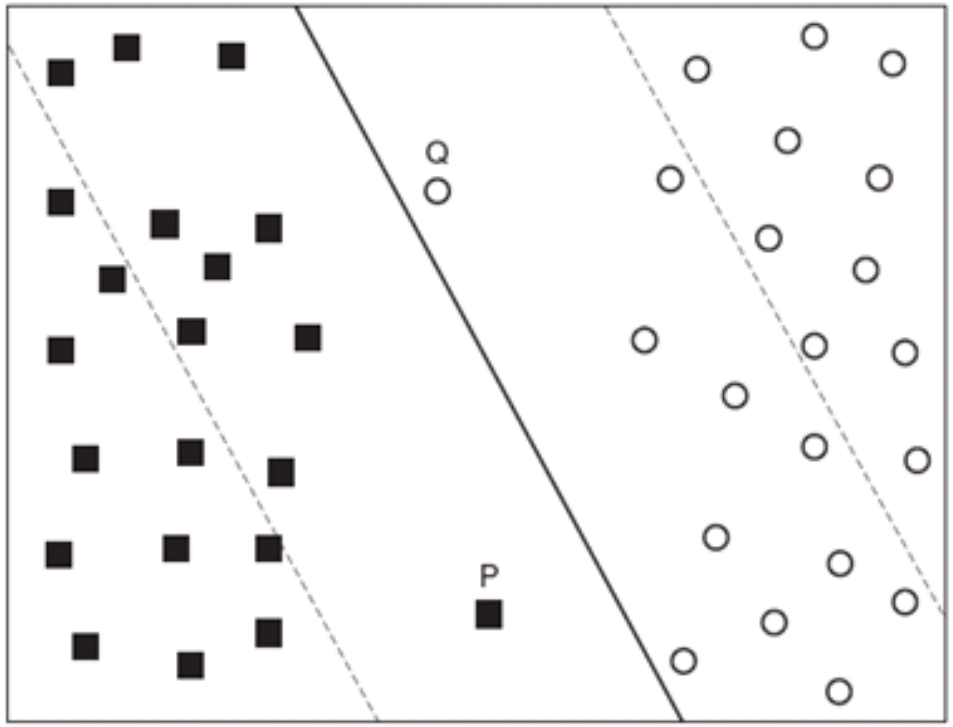
支持向量机（SVM）

- 新的目标函数和约束条件

$$f(w)=\frac{\|w\|^2}{2}$$

$$w\cdot x_i+b\geq 1-\xi_i \quad if\ \ y_i=1$$

$$w\cdot x_i+b\leq -1+\xi_i \quad if\ \ y_i=-1$$



支持向量机（SVM）

- 修改后的目标函数

$$f(w)=\frac{\|w\|^2}{2}+C(\sum_{i=1}^{N}\xi_i)^k$$

- 约束条件

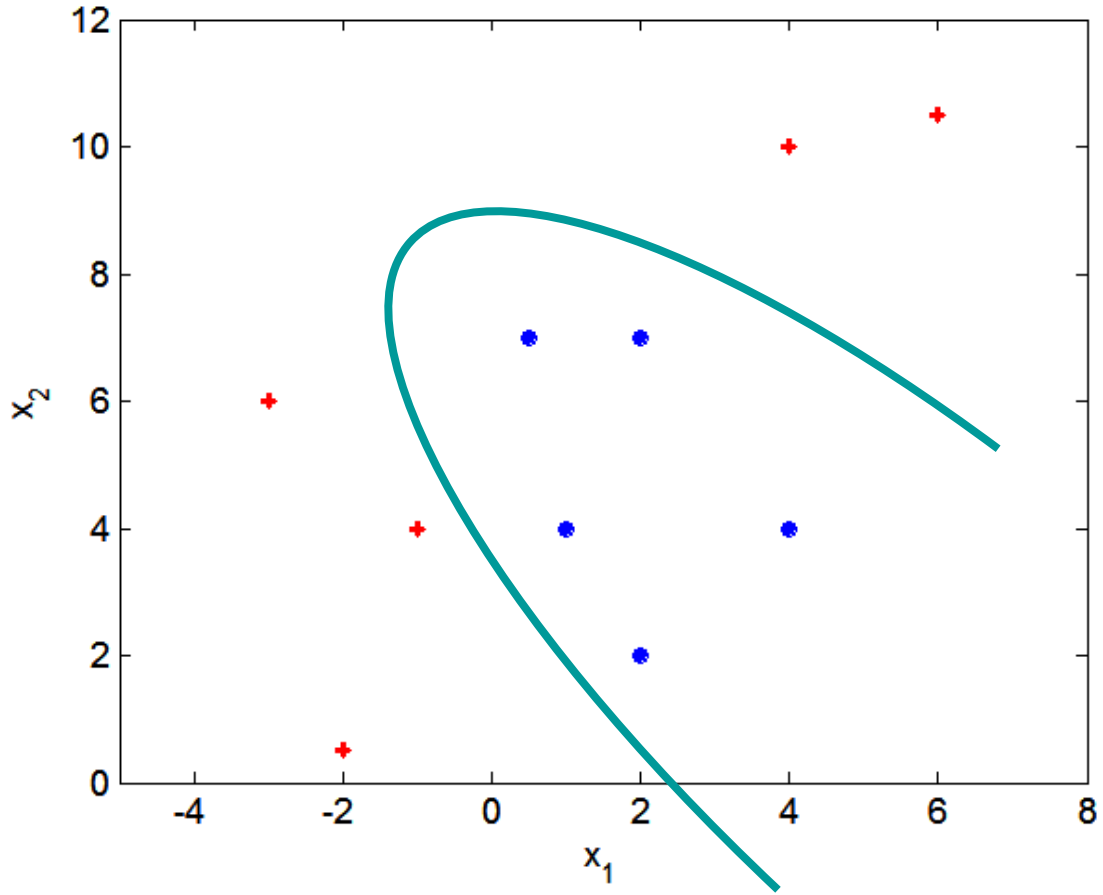
$$w\cdot x_i+b\geq 1-\xi_i \quad if\ \ y_i=1$$

$$w\cdot x_i+b\leq -1+\xi_i \quad if\ \ y_i=-1$$

$$\xi_i>0$$

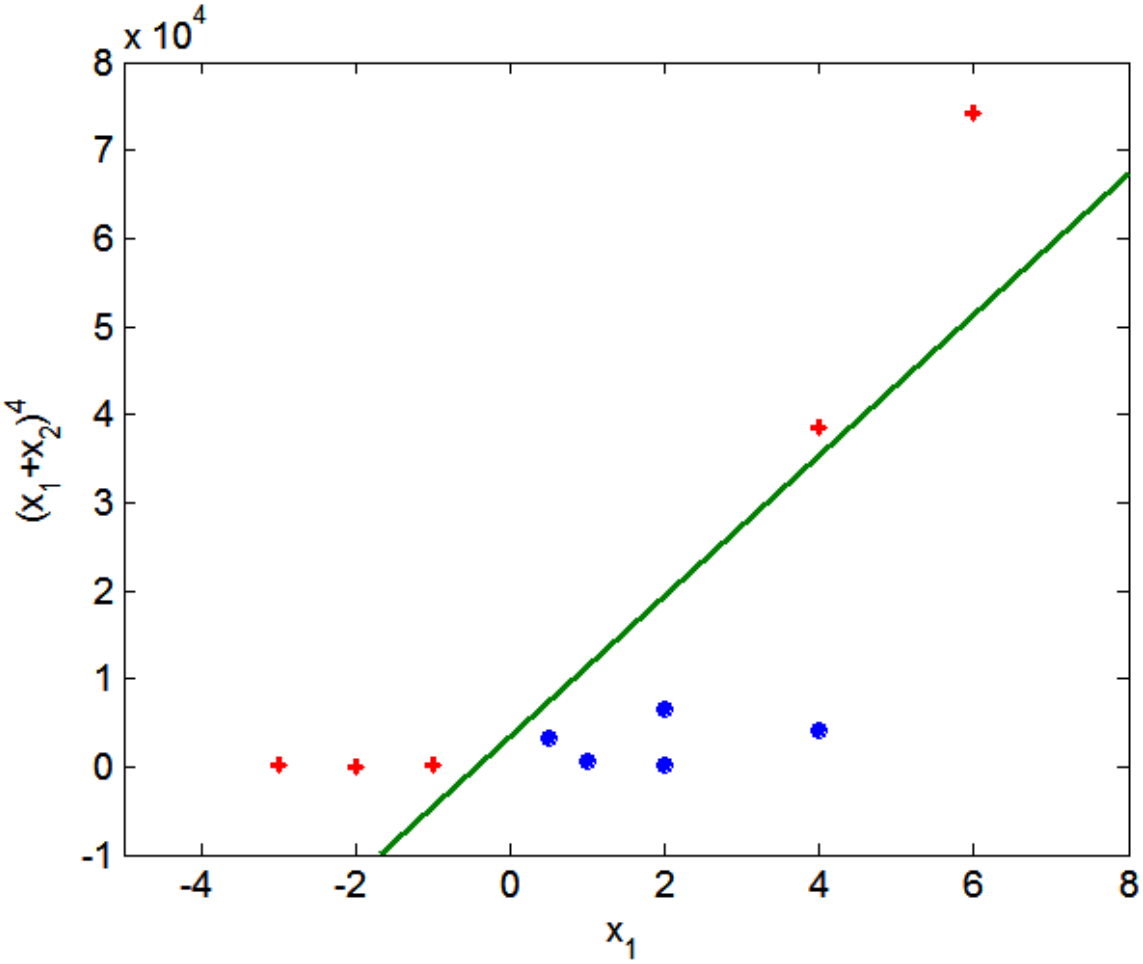
支持向量机（SVM）

- 非线性支持向量机

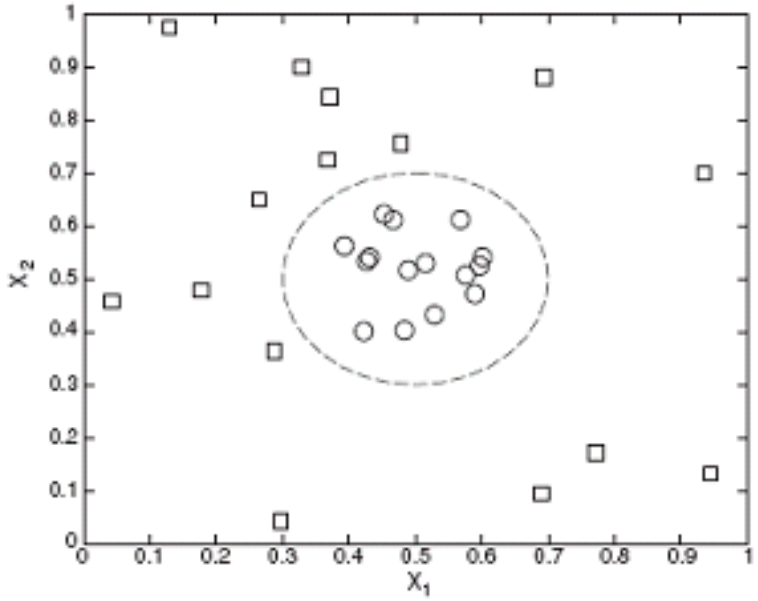


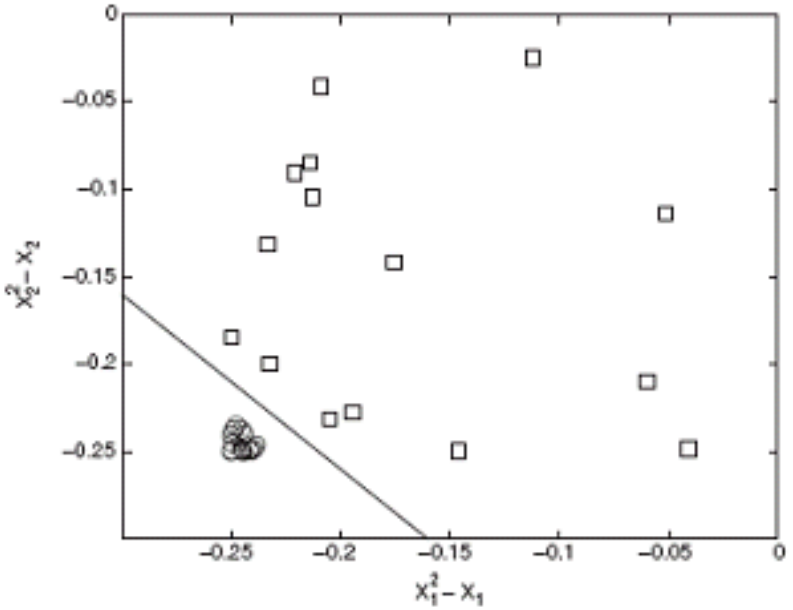
支持向量机（SVM）

- 将数据变换到高维空间中



支持向量机（SVM）





$$y(x_1, x_2) = \begin{cases} 1 & if\ \sqrt{(x_1 - 0.5)^2 + (x_2 - 0.5)^2} > 0.2 \\ -1 & else \end{cases}$$

$$x_1^2 - x_1 + x_2^2 - x_2 = -0.46$$

支持向量机（SVM）

- 学习非线性SVM模型：
- 1. 不清楚应当使用什么类型的映射函数，确保可以在变换空间构建线性决策边界
- 2. 即使知道合适的映射函数，在高维特征空间中解决被约束的优化问题仍然是计算代价很高的任务

支持向量机（SVM）

- SVM的特征
- 1. SVM具有“弱”学习能力，强“分类”能力
  - 弱学习能力表示只有部分的支持向量
  - 强分类能力表示具有最大边缘的最优超平面
- 2. SVM问题可以利用已知的有效算法来发现目标函数的全局最小值
- 3. 可以推导至多类问题

组合方法

- 从训练数据中构造一系列分类器
  - 基分类器
- 对每个基分类器的预测进行投票来分类

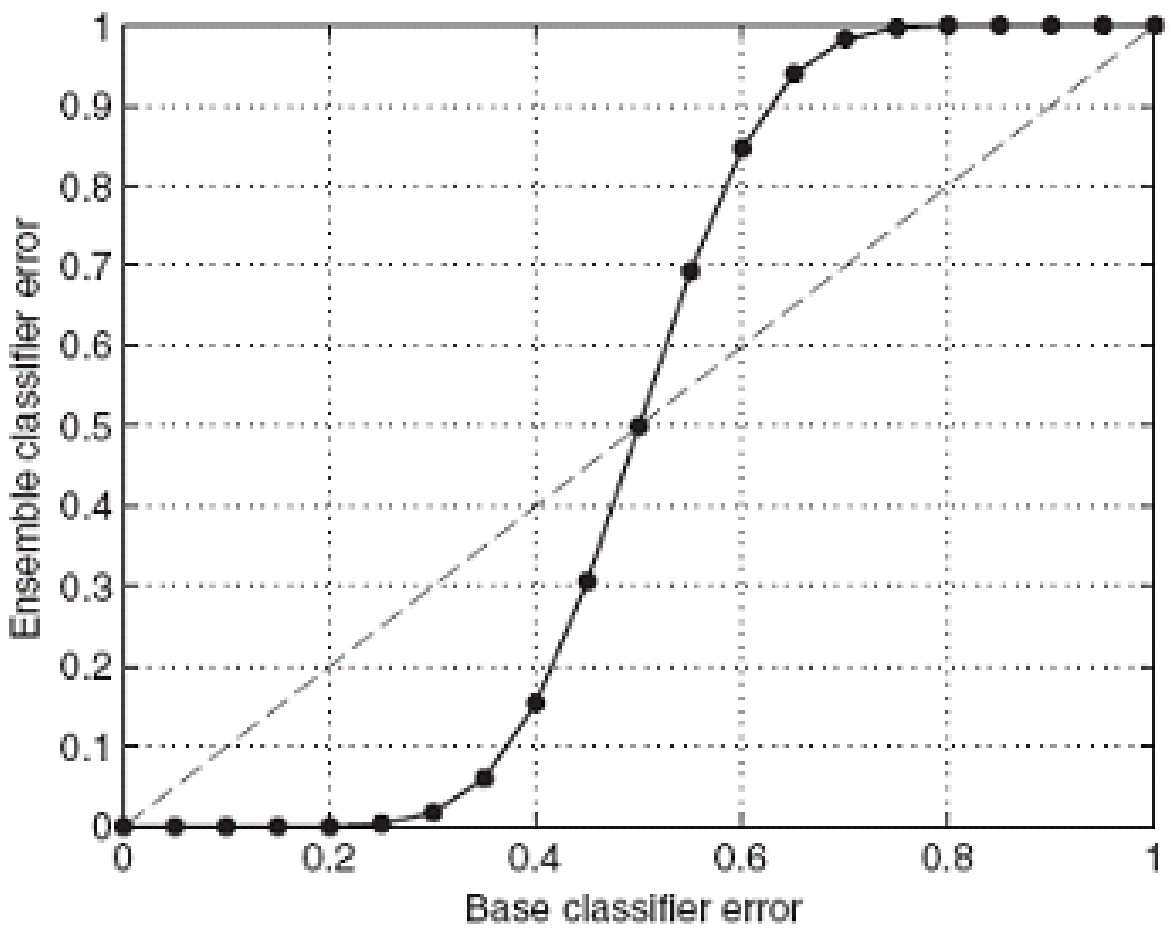
组合方法

- 假设有25个基分类器
  - 每个基分类器的误差为0.35
  - 假设分类器之间是独立的
  - 组合分类器的输出采用投票机制，则其误差率为

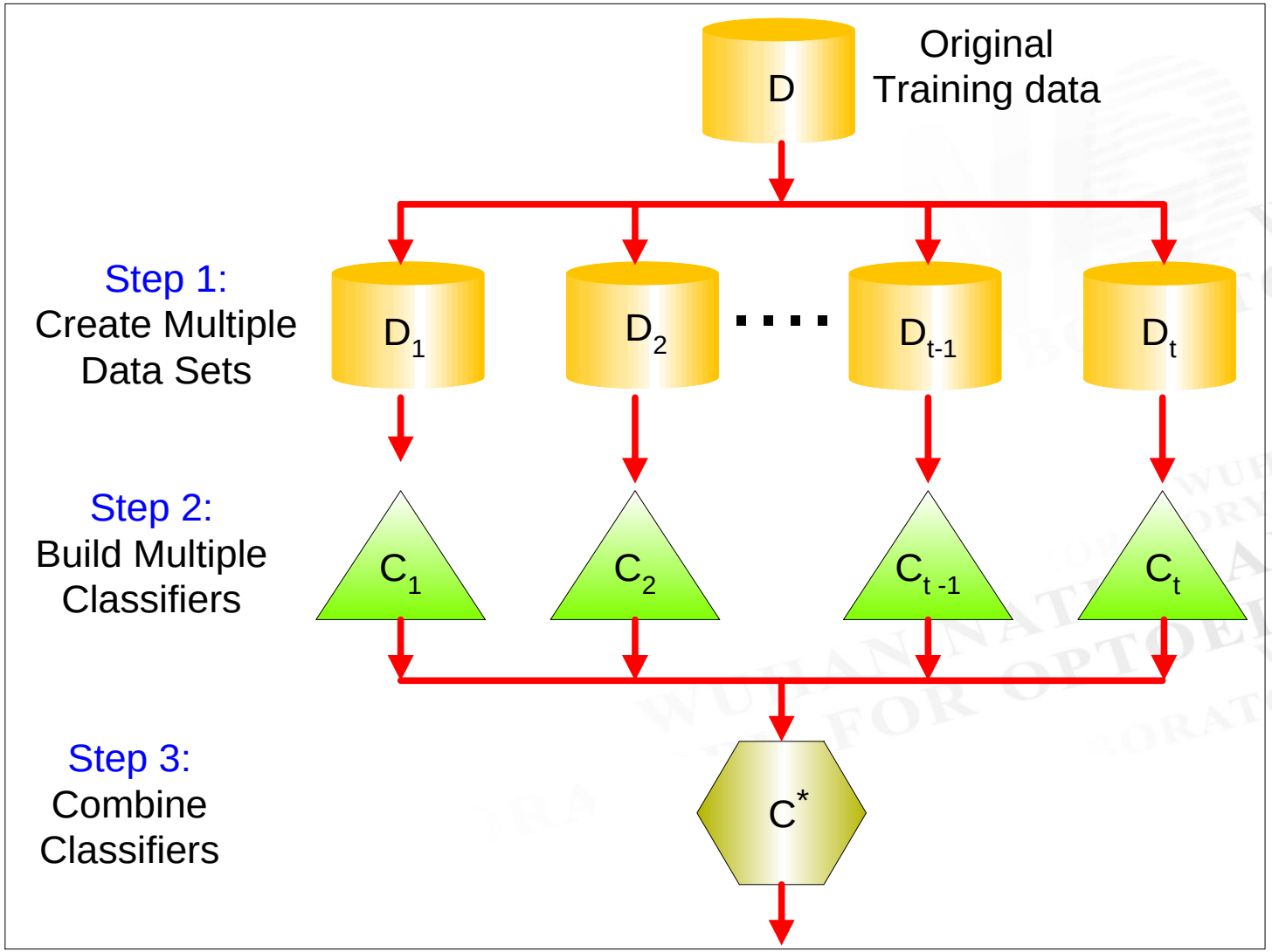
$$\sum_{i=13}^{25}\binom{25}{i}\varepsilon^{i}(1-\varepsilon)^{25-i}=0.06$$

组合方法

- 虚线对应所有基分类器等同的情况
- 实线对应基分类器独立的情况
- 基分类器应好于随机猜测分类器



组合方法


D
Original
Training data
Step 1:
Create Multiple
Data Sets
D_1
D_2
D_{t-1}
D_t
Step 2:
Build Multiple
Classifiers
C_1
C_2
C_{t-1}
C_t
Step 3:
Combine
Classifiers
C^*

组合方法

组合方法的一般过程

1: 令 D 表示原始训练数据集，k 表示基分类器的个数，T 表示检验数据集

2: for i=1 to k do

3: 由 D 创建训练集 D_i

4: 由 D_i 创建基分类器 C_i

5: end for

6: for 每一个检验记录 $x \in T$ do

7: $C^*(x) = Vote(C_1(x), C_2(x), \cdots C_k(x))$

8: end for

组合方法

- 1. 通过处理训练数据集
  - 根据某种抽样分布，得到多个训练集
  - Bagging，boosting
- 2. 通过处理输入特征
  - 通过选择输入特征的子集来形成每个训练集
  - Random forest

组合方法

- **3. 通过处理类标号**
  - 通过将类标号随机划分成两个不相交的子集，将训练数据变换成二类问题
  - 错误-纠正输出编码
- **4. 通过处理学习算法**
  - 通过在算法执行过程中引入随机性，对于同一个训练数据集，可以得到不同的模型

组合方法

- **Bagging**
- 是一种根据均匀概率分布从数据集中重复抽样（有放回）的技术
- 一个样本被抽到的概率为$1-(1-1/n)^n$
  - 近似为63%

装袋算法

1: 设 k 为自助样本集的数目

2: for i=1 to k do

3: 生成一个大小为 N 的自助样本集 D_i

4: 在自助样本集 D_i 上训练一个基分类器 C_i

5: end for

6: $C^*(x) = \arg\max_y \sum_i \delta(C_i(x) = y)$

{如果参数为真则 δ(.)=1，否则 δ(.)=0}

x	y
0.1	1
0.2	1
0.3	1
0.4	-1
0.5	-1
0.6	-1
0.7	-1
0.8	1
0.9	1
1.0	1

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Round 2:

x	0.1	0.2	0.3	0.4	0.5	0.8	0.9	1	1	1
y	1	1	1	-1	-1	1	1	1	1	1

x <= 0.65 ==> y = 1
x > 0.65 ==> y = 1

Bagging Round 3:

x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Round 4:

x	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

x <= 0.3 ==> y = 1
x > 0.3 ==> y = -1

Bagging Round 5:

x	0.1	0.1	0.2	0.5	0.6	0.6	0.6	1	1	1
y	1	1	1	-1	-1	-1	-1	1	1	1

x <= 0.35 ==> y = 1
x > 0.35 ==> y = -1

Bagging Round 6:

x	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	1
y	1	-1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 7:

x	0.1	0.4	0.4	0.6	0.7	0.8	0.9	0.9	0.9	1
y	1	-1	-1	-1	-1	1	1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 8:

x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 9:

x	0.1	0.3	0.4	0.4	0.6	0.7	0.7	0.8	1	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

x <= 0.75 ==> y = -1
x > 0.75 ==> y = 1

Bagging Round 10:

x	0.1	0.1	0.1	0.1	0.3	0.3	0.8	0.8	0.9	0.9
y	1	1	1	1	1	1	1	1	1	1

x <= 0.05 ==> y = -1
x > 0.05 ==> y = 1

组合方法

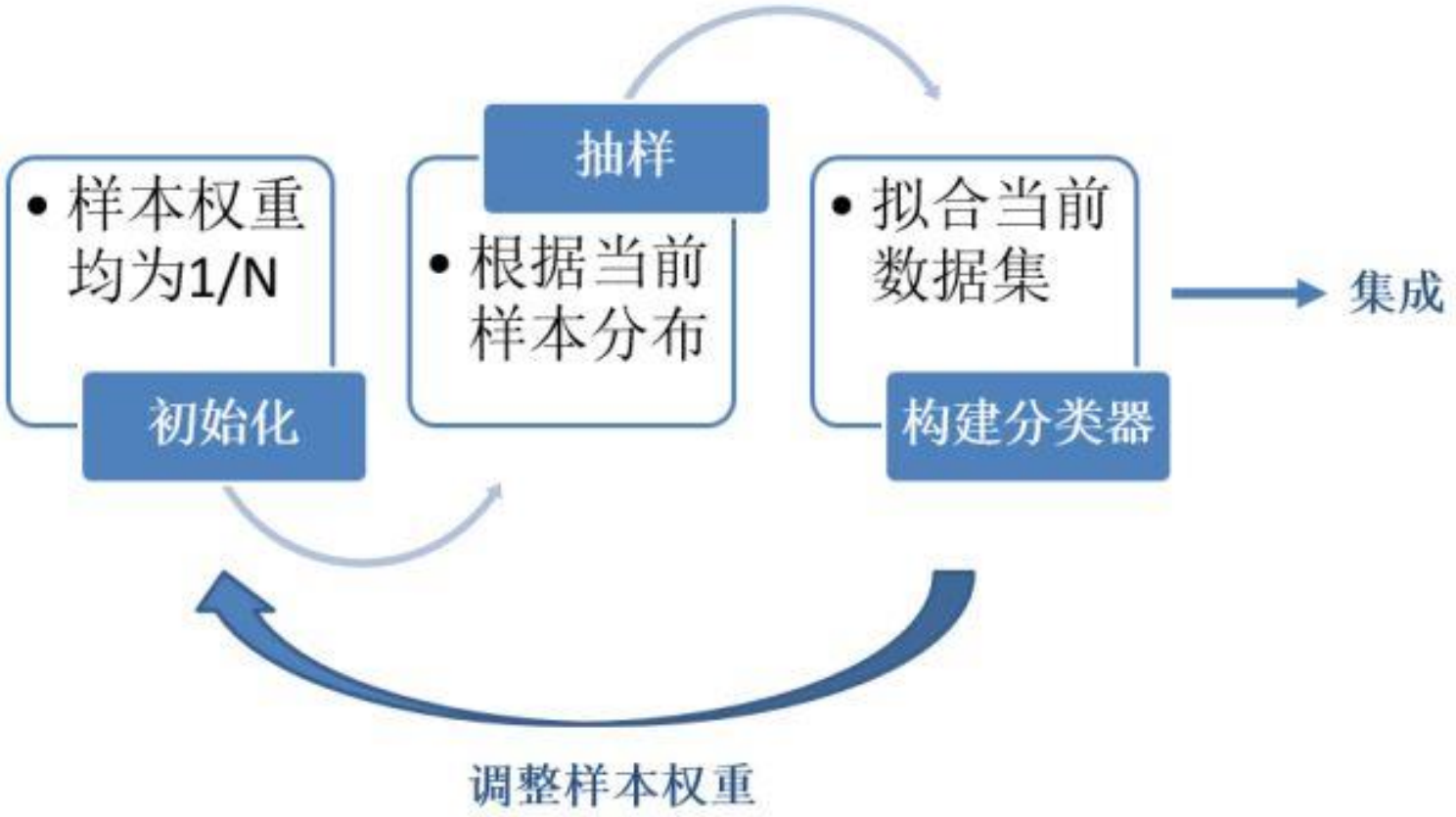
- 装袋构建的组合分类器

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	1	1	1	-1	-1	-1	-1	-1	-1	-1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
4	1	1	1	-1	-1	-1	-1	-1	-1	-1
5	1	1	1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	-1	-1	1	1	1
7	-1	-1	-1	-1	-1	-1	-1	1	1	1
8	-1	-1	-1	-1	-1	-1	-1	1	1	1
9	-1	-1	-1	-1	-1	-1	-1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
Sum	2	2	2	-6	-6	-6	-6	2	2	2
Sign	1	1	1	-1	-1	-1	-1	1	1	1
True Class	1	1	1	-1	-1	-1	-1	1	1	1

组合方法

- **Boosting**
- 通过迭代过程来自适应的改变训练样本的分布，使得基分类器聚焦在很难分类的样本上
  - 初始时，所有**N**个记录赋予同样的权重
  - 权重可以在每一轮提升后动态调整（与**bagging**不同）

组合方法



组合方法

- 权值可用于:
- 1. 抽样分布，从原始数据中提取出自助样本集
- 2. 基分类器可以使用权值学习有利于高权值样本的模型
  - 高权值样本即难以正确分类的样本

组合方法

- 错误分类的记录权重增加
- 正确分类的记录权重减小

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

➢ AdaBoost 算法

AdaBoost 算法

1: 令 W={w_j=1/N | j=1,2,…,N}。{初始化 N 个样本的权值}

2: 令 k 表示提升的轮数

3: for i=1 to k do

4: 根据 w，通过对 D 进行抽样（有放回）产生训练集 D_i

5: 在 D_i 上训练基分类器 C_i

6: 用对原训练集 D 中的所有样本分类

7: $\varepsilon_i = \frac{1}{N}\left[\sum_j w_j \delta(C_i(x_j) \neq y_j\right]$，{计算加权误差}

8: if ε_i >0.5 then

9: w={w_j=1/N | j=1,2,…, N}。{重新设置 N 个样本的权值}

10: 返回步骤 4。

11: end if

12: $a_i = \frac{1}{2}\ln\frac{1-\varepsilon_i}{\varepsilon_i}$

13: $w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \times \begin{cases} e^{-a_j} & if\ C_j(x_i) = y_i \\ e^{a_j} & if\ C_j(x_i) \neq y_i \end{cases}$。{更新每个样本的权值}

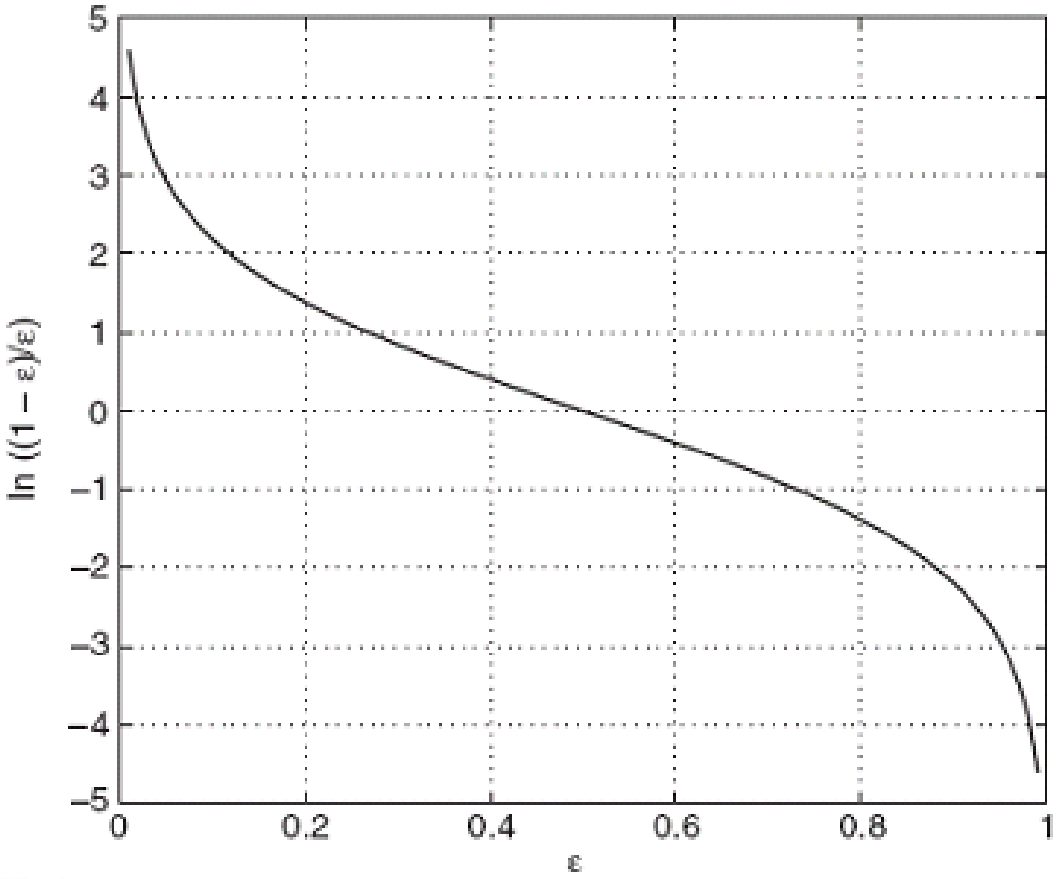
14: end for

15: $C^*(x) = \arg\max_y \sum_{j=1}^{T} a_j \delta(C_j(x) = y)$

组合方法

$$a_i = \frac{1}{2}\ln\frac{1-\varepsilon_i}{\varepsilon_i}$$

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \times \begin{cases} e^{-a_j} & if\ C_j(x_i) = y_i \\ e^{a_j} & if\ C_j(x_i) \neq y_i \end{cases}$$



组合方法

x	y
0.1	1
0.2	1
0.3	1
0.4	-1
0.5	-1
0.6	-1
0.7	-1
0.8	1
0.9	1
1.0	1

Boosting Round 1:

x	0.1	0.4	0.5	0.6	0.6	0.7	0.7	0.7	0.8	1
y	1	-1	-1	-1	-1	-1	-1	-1	1	1

Boosting Round 2:

x	0.1	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3
y	1	1	1	1	1	1	1	1	1	1

Boosting Round 3:

x	0.2	0.2	0.4	0.4	0.4	0.4	0.5	0.6	0.6	0.7
y	1	1	-1	-1	-1	-1	-1	-1	-1	-1

(a) Training records chosen during boosting

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
2	0.311	0.311	0.311	0.01	0.01	0.01	0.01	0.01	0.01	0.01
3	0.029	0.029	0.029	0.228	0.228	0.228	0.228	0.009	0.009	0.009

(b) Weights of training records

组合方法

Round	Split Point	Left Class	Right Class	α
1	0.75	-1	1	1.738
2	0.05	1	1	2.7784
3	0.3	1	-1	4.1195

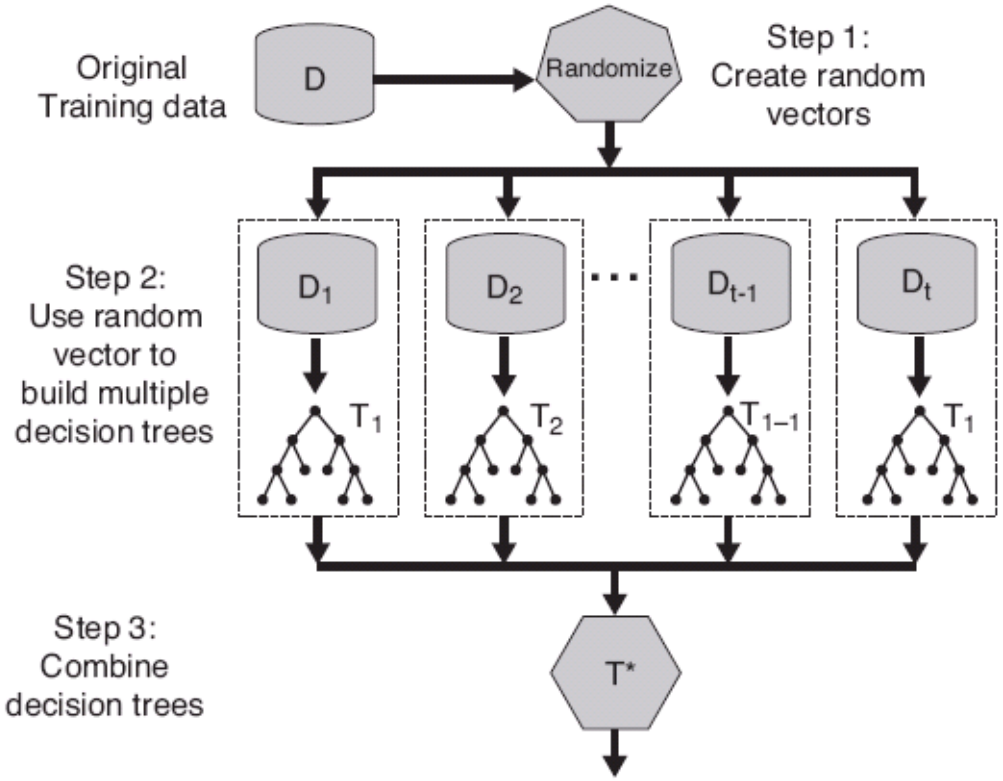
(a)

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	-1	-1	-1	-1	-1	-1	-1	1	1	1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
Sum	5.16	5.16	5.16	-3.08	-3.08	-3.08	-3.08	0.397	0.397	0.397
Sign	1	1	1	-1	-1	-1	-1	1	1	1

(b)

组合方法

- 随机森林：是一类专门为决策树分类器设计的组合方法
- 组合多棵决策树作出的预测，其中每棵树都是基于随机向量的一个独立集合的值产生的



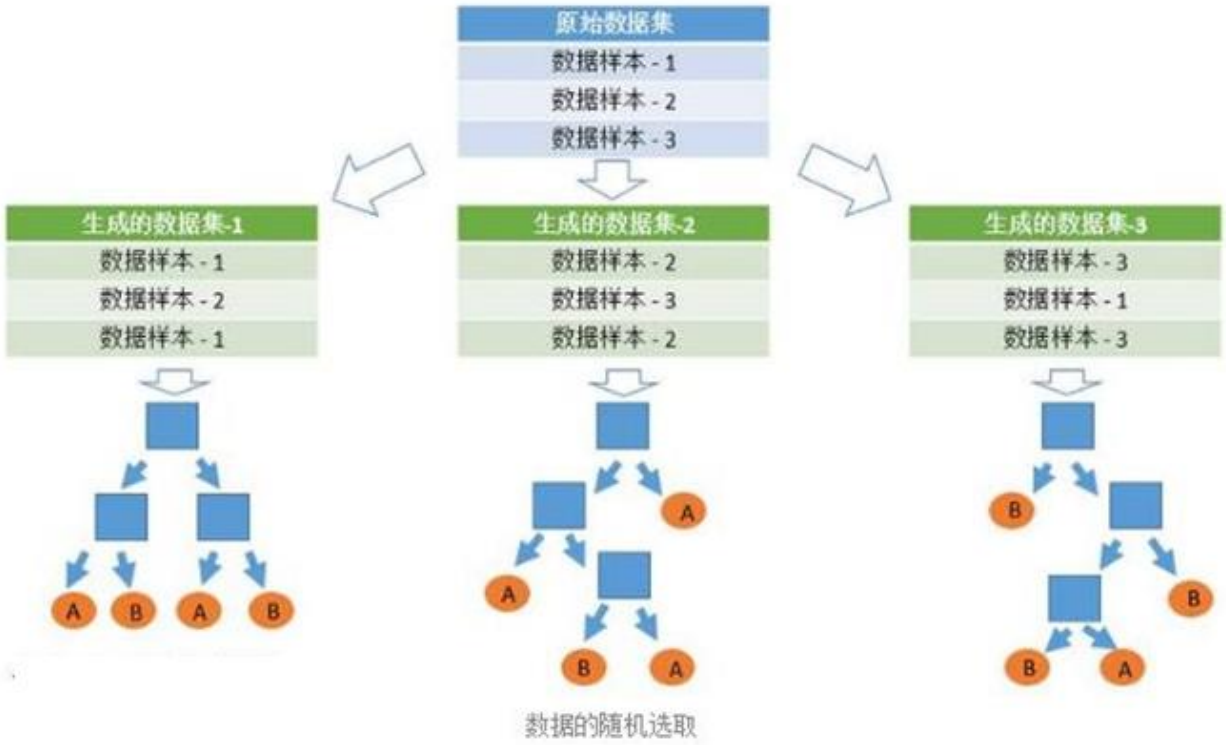
组合方法

- 使用多棵决策树联合进行预测可以有效降低模型的方差

$$\mathrm{D}\left(\frac{1}{n}\sum_{i}^{n}x_i\right)=\sigma^2/n$$

组合方法

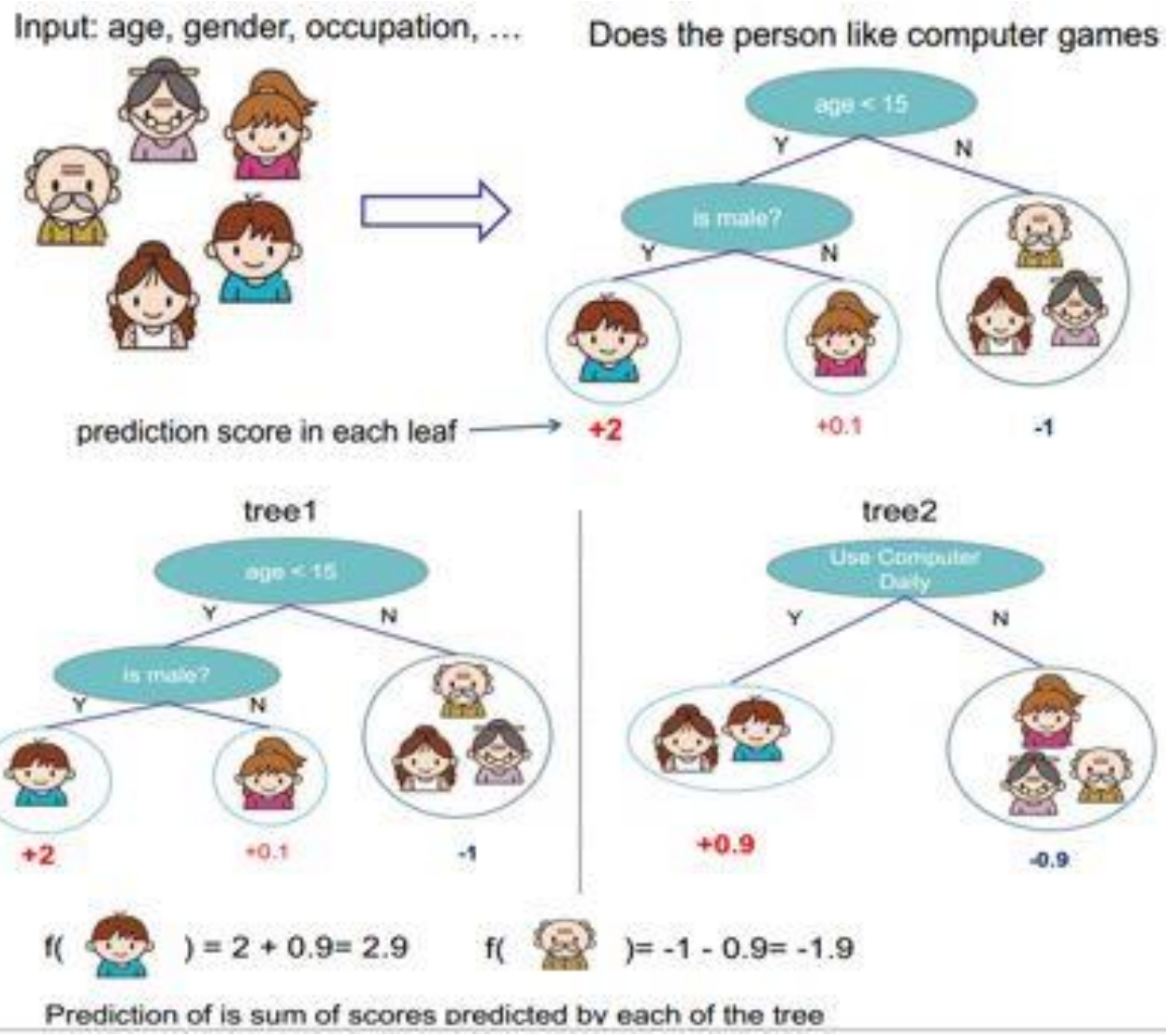
- 1.每棵树的训练样本从原始训练集中随机抽样得到



组合方法

- 2.创建输入特征的线性组合来加大特征空间
- 3.随机选择L个输入特征来对决策树的结点进行分裂
- 4. 在决策树的每一个结点，从F个最佳划分中随机选择一个

组合方法


Input: age, gender, occupation, …
Does the person like computer games
age < 15
Y
N
is male?
Y
N
prediction score in each leaf
+2
+0.1
-1
tree1
age < 15
Y
N
is male?
Y
N
+2
+0.1
-1
tree2
Use Computer Daily
Y
N
+0.9
-0.9
f( ) = 2 + 0.9= 2.9
f( )= -1 - 0.9= -1.9
Prediction of is sum of scores predicted by each of the tree

组合方法

- 组合分类器的优势：
- 1. 适合处理过大或过小的数据集
  - 数据集较大时，将数据集划分成多个子集，对子集构建分类器
  - 数据集较小时，可通过多种抽样方式从原始数据集抽样产生多组不同的数据集，构建分类器
- 2. 整合各个子模型的分类结果，得到更合理的决策边界，提升模型精度

组合方法

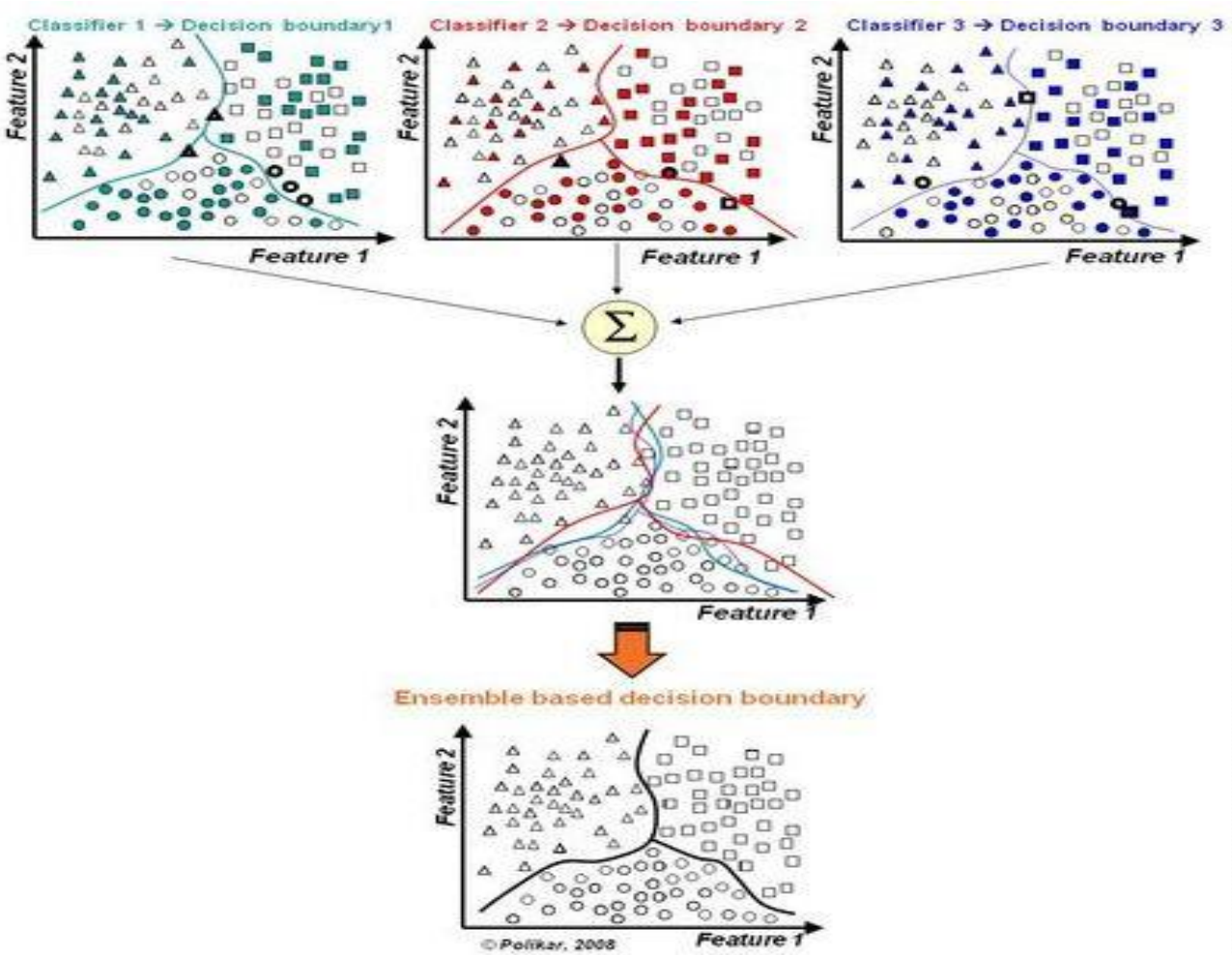
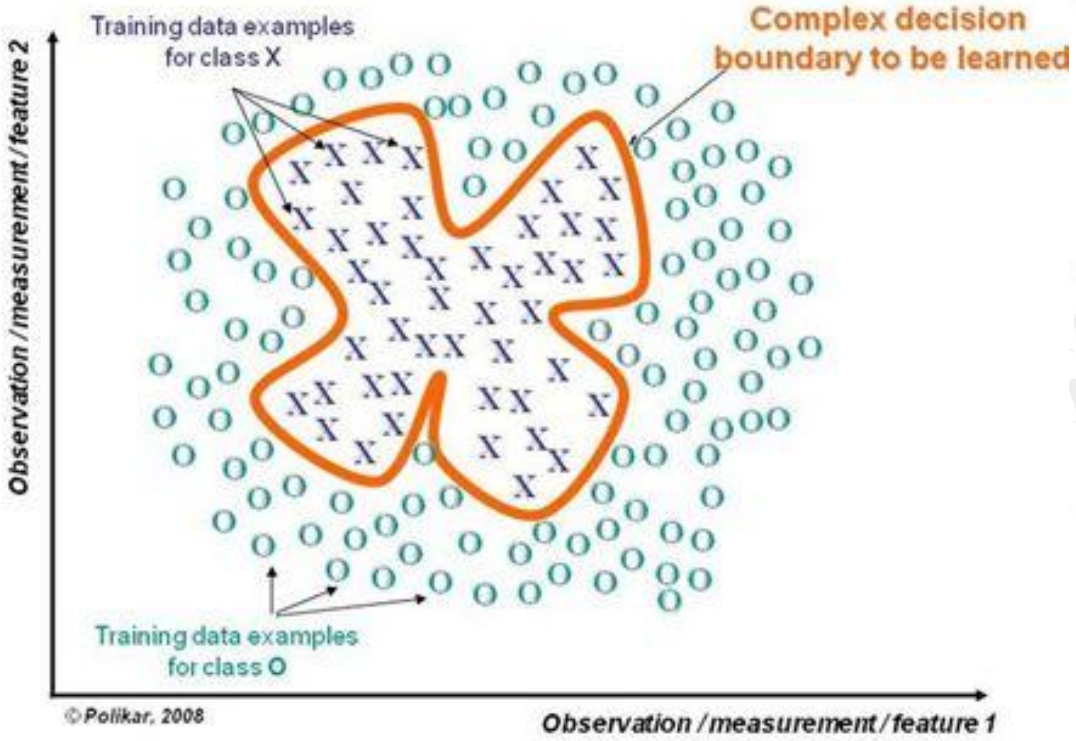


Figure 1: Combining an ensemble of classifiers for reducing classification error and/or model selection.

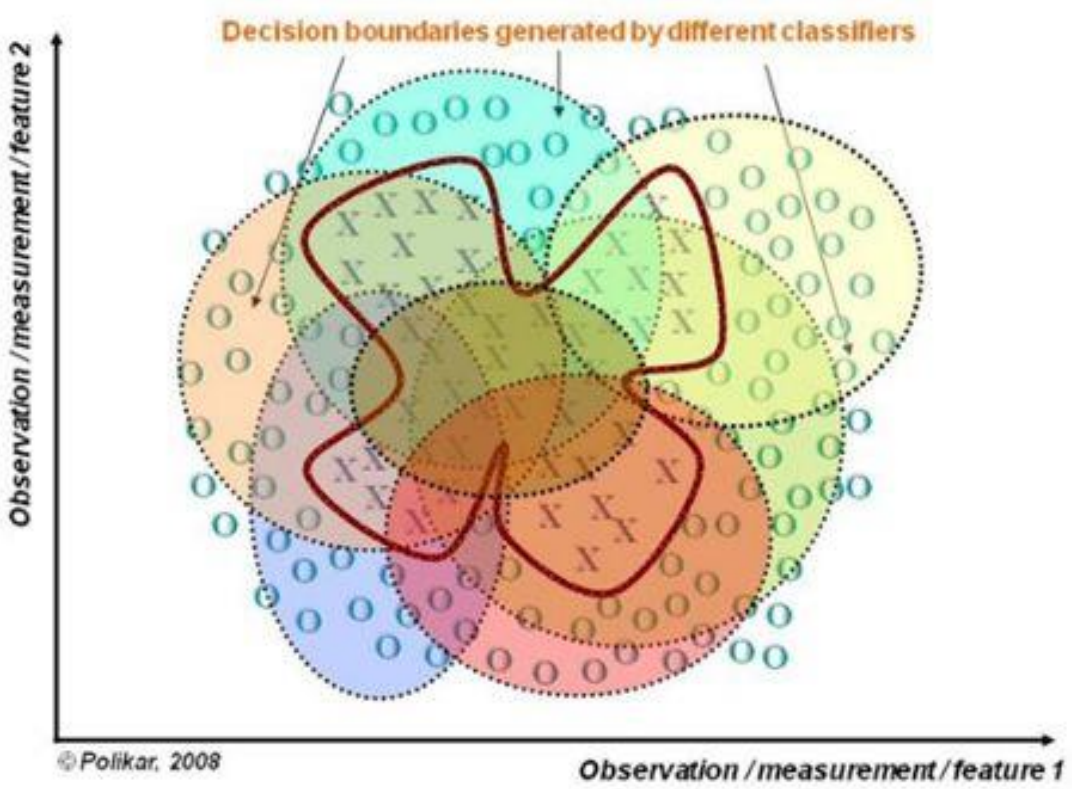
组合方法

- 3.若决策边界过于复杂，则线性模型不能很好地适应，因此先对于特定区域的数据集，训练多个线性分类器，再将它们集成



组合方法

- **4. 适合处理多源异构的数据**
  - 离散型、连续型、时序型、网络结构数据等



GBDT

- 梯度提升决策树
  - Gradient Boosting Decison Tree
- 每一次的计算是都为了减少上一次的残差，进而在残差减少的方向上建立一个新的模型

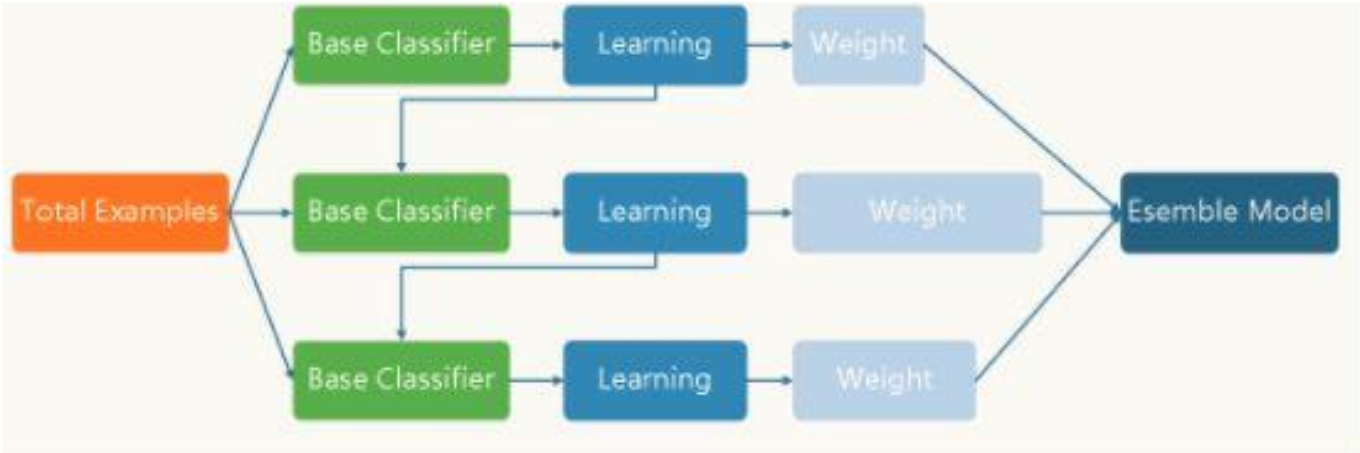
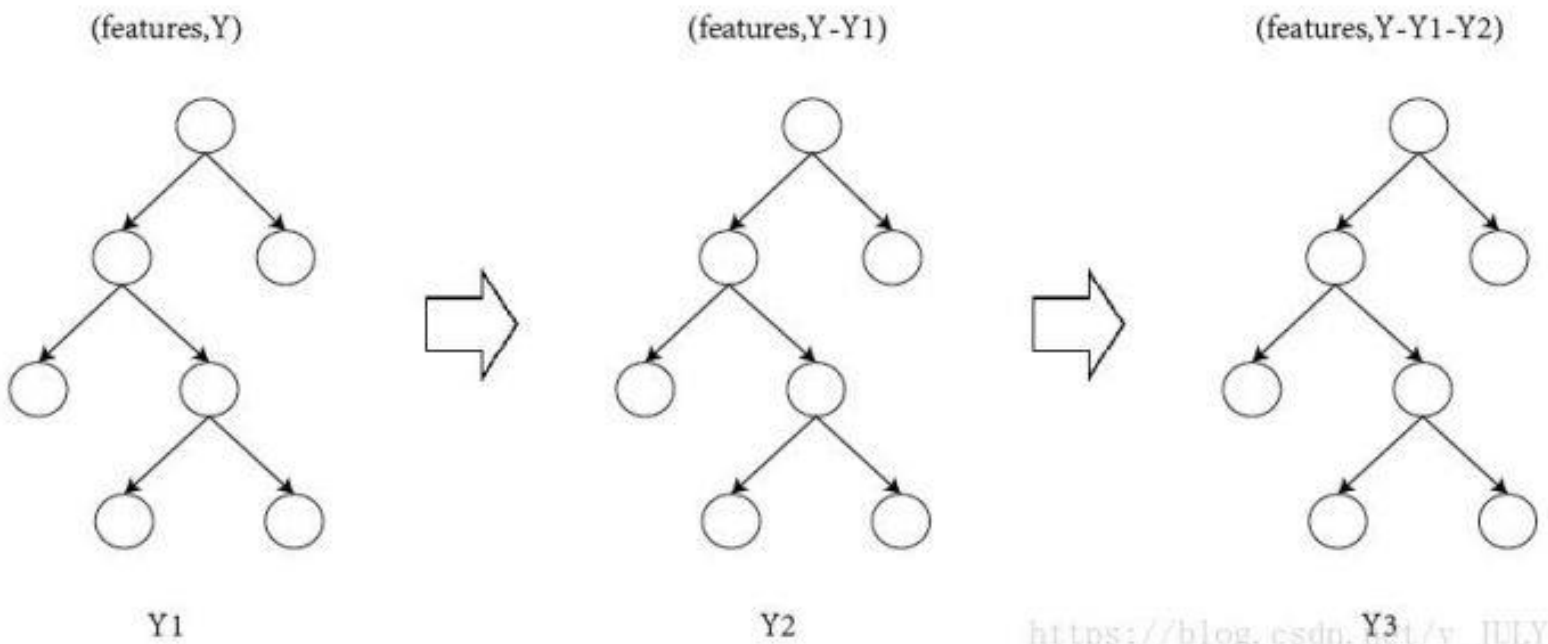


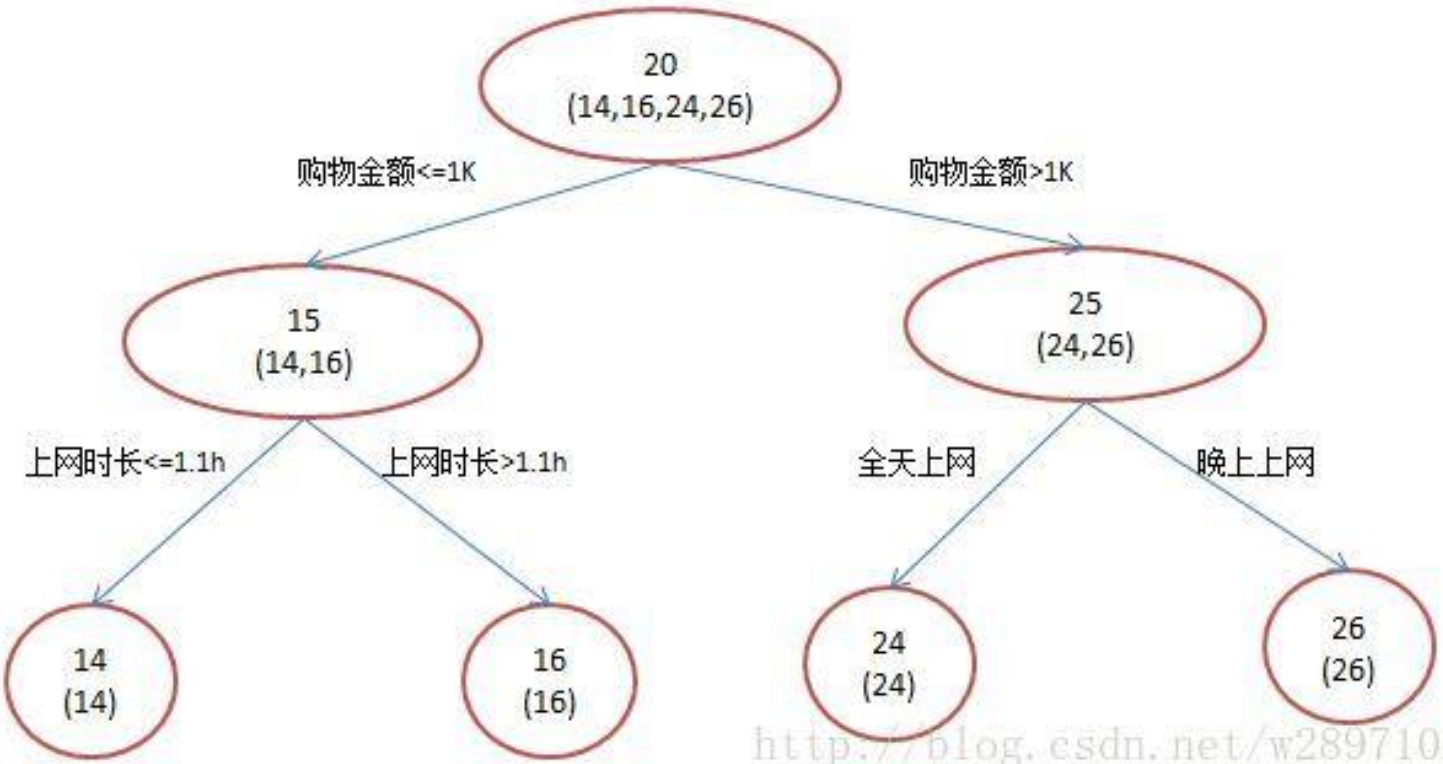
图 1：GBDT 的训练过程

GBDT



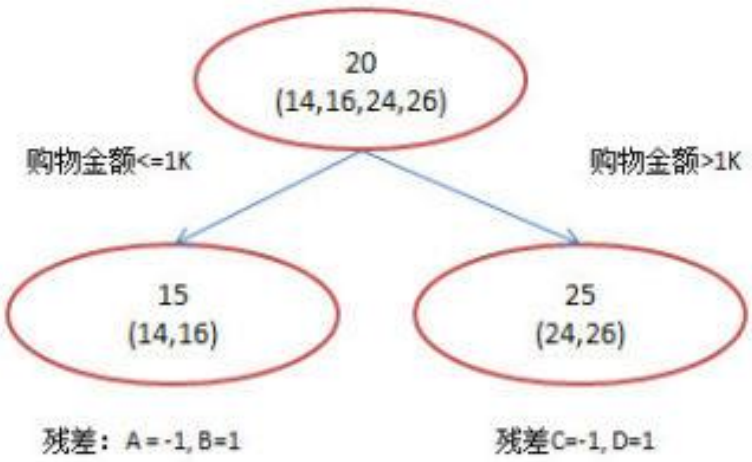
GBDT

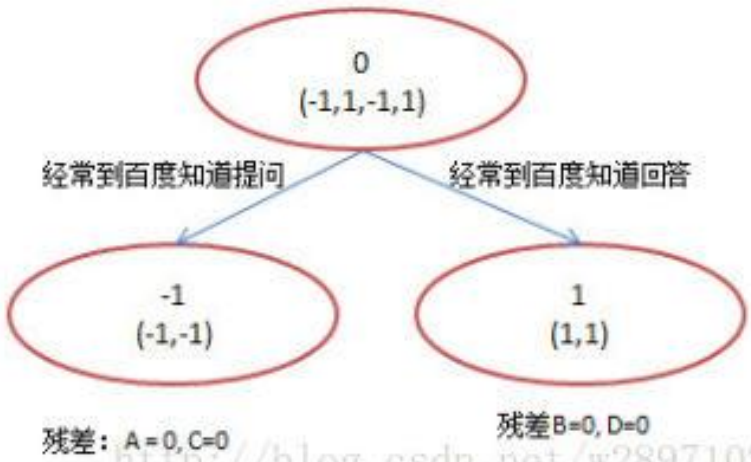
- 年龄预测的例子



GBDT

- 使用GBDT解决该问题，限定叶子节点最多有两个，即每棵树都只有一个分枝，并且限定只学两棵树





GBDT

- 模型描述如下

$$F_m(x) = \sum_{m=1}^{M} T(x;\theta_m)$$

- 弱分类器的损失函数为

$$\hat{\theta}_m = \arg\min_{\theta_m} \sum_{i=1}^{N} L\left(y_i, F_{m-1}(x_i) + T(x_i;\theta_m)\right)$$

GBDT

- GBDT的不足
- 1.需要生成一定数量的树才能达到满意的准确率
  - 计算开销大
- 2. 弱分类器之间存在依赖关系，难以并行训练

GBDT

- RF与GBDT的区别：
- （1）相同点
- 都是由多棵树组成
- 最终的结果都是由多棵树一起决定
- （2）不同点
- RF可以包含分类树或回归树，GBDT只能由回归树组成
- RF可以并行生成，GBDT串行生成
- RF的结果是多数表决得到，GBDT是多棵树累加之和
- RF对异常值不敏感，GBDT对异常值比较敏感
- RF通过减少模型方差来提高性能，GBDT减少模型的偏差来提高性能
- RF不需要进行数据预处理，GBDT需要进行特征归一化

XGBoost

- eXtreme Gradient Boosting
  - 华盛顿大学陈天奇
- 目标函数依然是所有树的预测值相加等于目标值
- 损失函数变的复杂
  - 引入了一阶导数和二阶导数

XGBoost

- Model: assuming we have K trees

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F}$$

- Objective

$$Obj = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

Training loss　　Complexity of the Trees

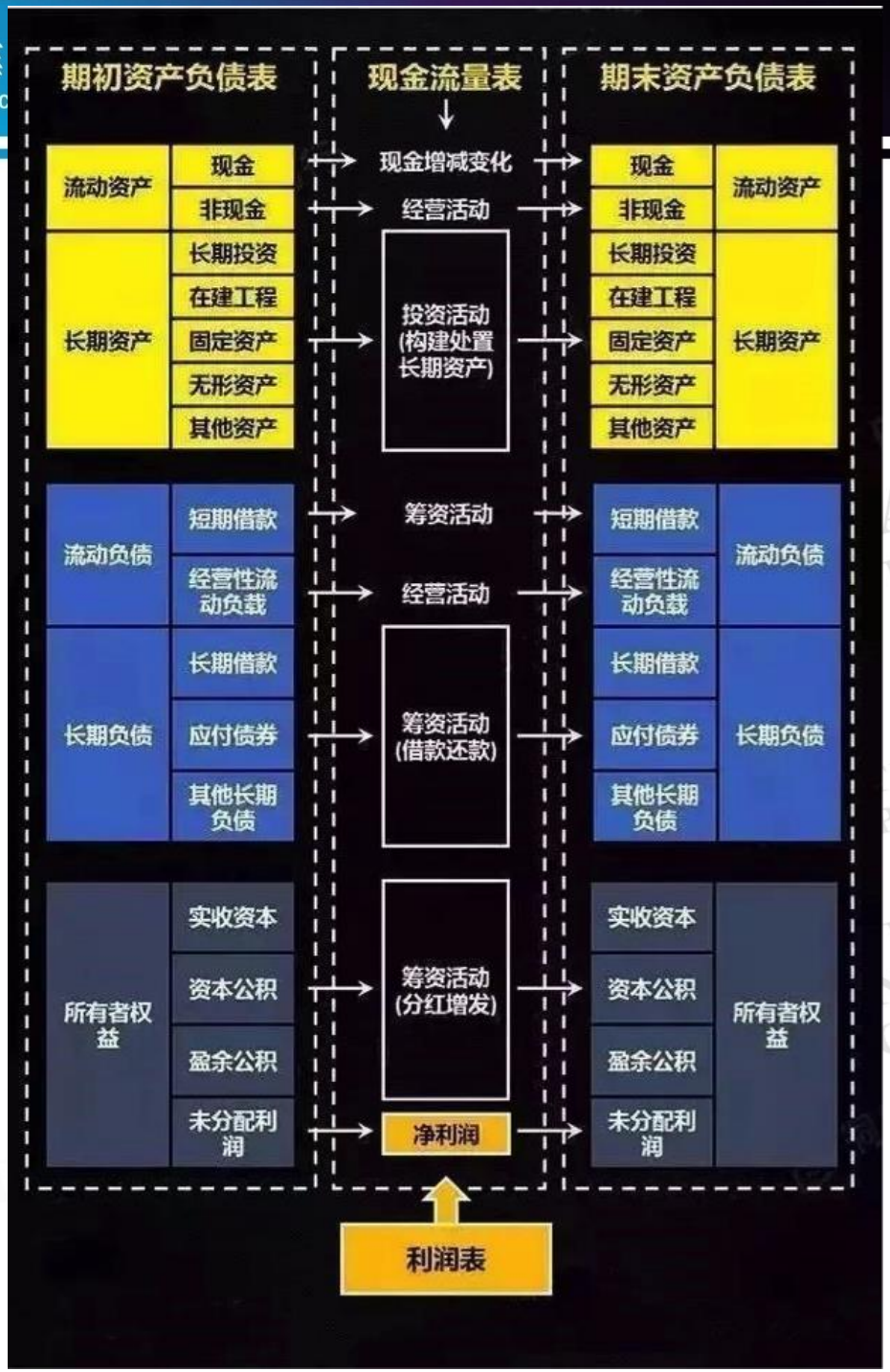
  - Square loss: $l(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$
  - Logistic loss: $l(y_i, \hat{y}_i) = y_i \ln(1 + e^{-\hat{y}_i}) + (1 - y_i) \ln(1 + e^{\hat{y}_i})$
  - L2 norm: $\Omega(w) = \lambda \|w\|^2$
  - L1 norm (lasso): $\Omega(w) = \lambda \|w\|_1$

XGBoost

- XGBoost相比于GBDT的创新之处：
- 1.传统GBDT在优化时只用到一阶导数信息， XGBoost则对代价函数进行了二阶泰勒展开，同时用到了一阶和二阶导数
- 2. XGBoost在代价函数里加入了正则项，用于控制模型的复杂度
- 3. 引入了学习速率的思想，每次迭代增加新的模型时，乘以小于1的系数，降低优化的速度
- 4.列抽样（column subsampling），借鉴了随机森林的做法，每次只采用部分特征

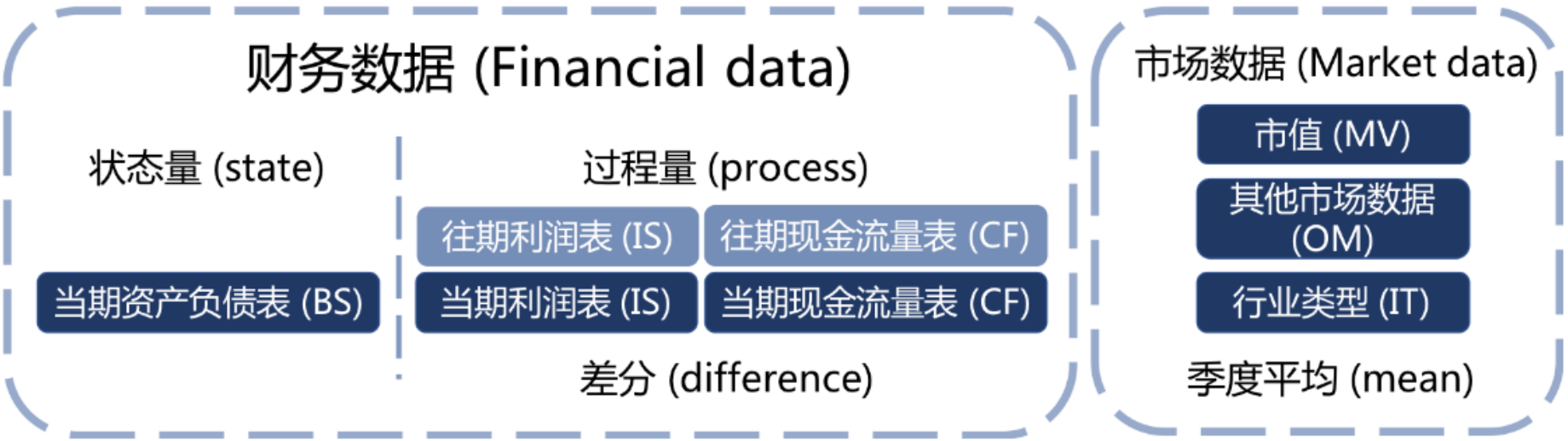
实际案例

- A股上市公司季度营收预测
  - （1）财务数据
  - （2）宏观数据
  - （3）行情数据
  - （4）行业数据



实际案例

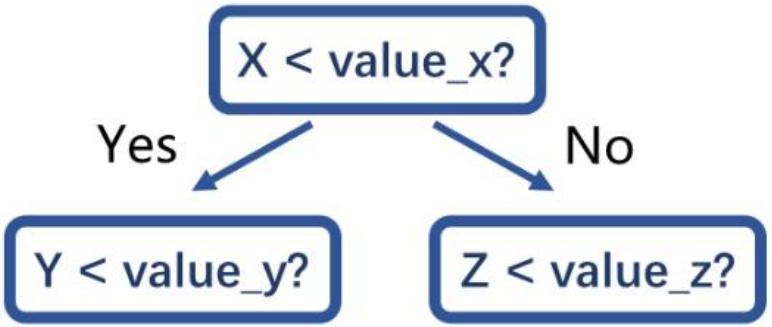
- 数据预处理


财务数据 (Financial data)
状态量 (state)
当期资产负债表 (BS)
过程量 (process)
往期利润表 (IS)
往期现金流量表 (CF)
当期利润表 (IS)
当期现金流量表 (CF)
差分 (difference)
市场数据 (Market data)
市值 (MV)
其他市场数据 (OM)
行业类型 (IT)
季度平均 (mean)

实际案例

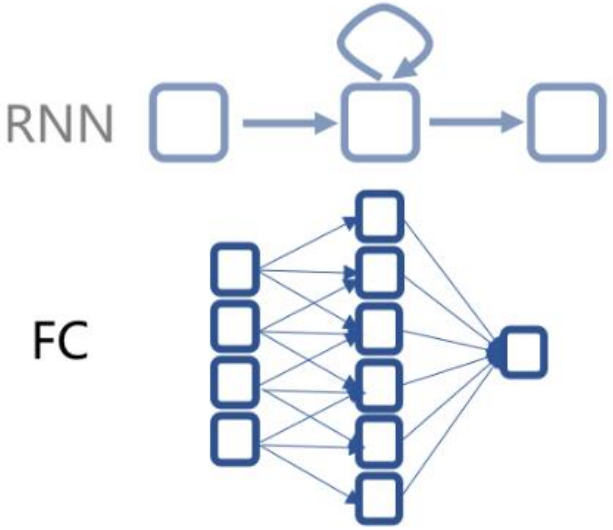
- 多模型集成

多因子模型
(Multi-factors Model)
决策树模型
(Decision Tree, XGBoost)



除去营业收入的财报数据
(Financial Data **ex. Revenue**) ,
市场数据 (Market Data)

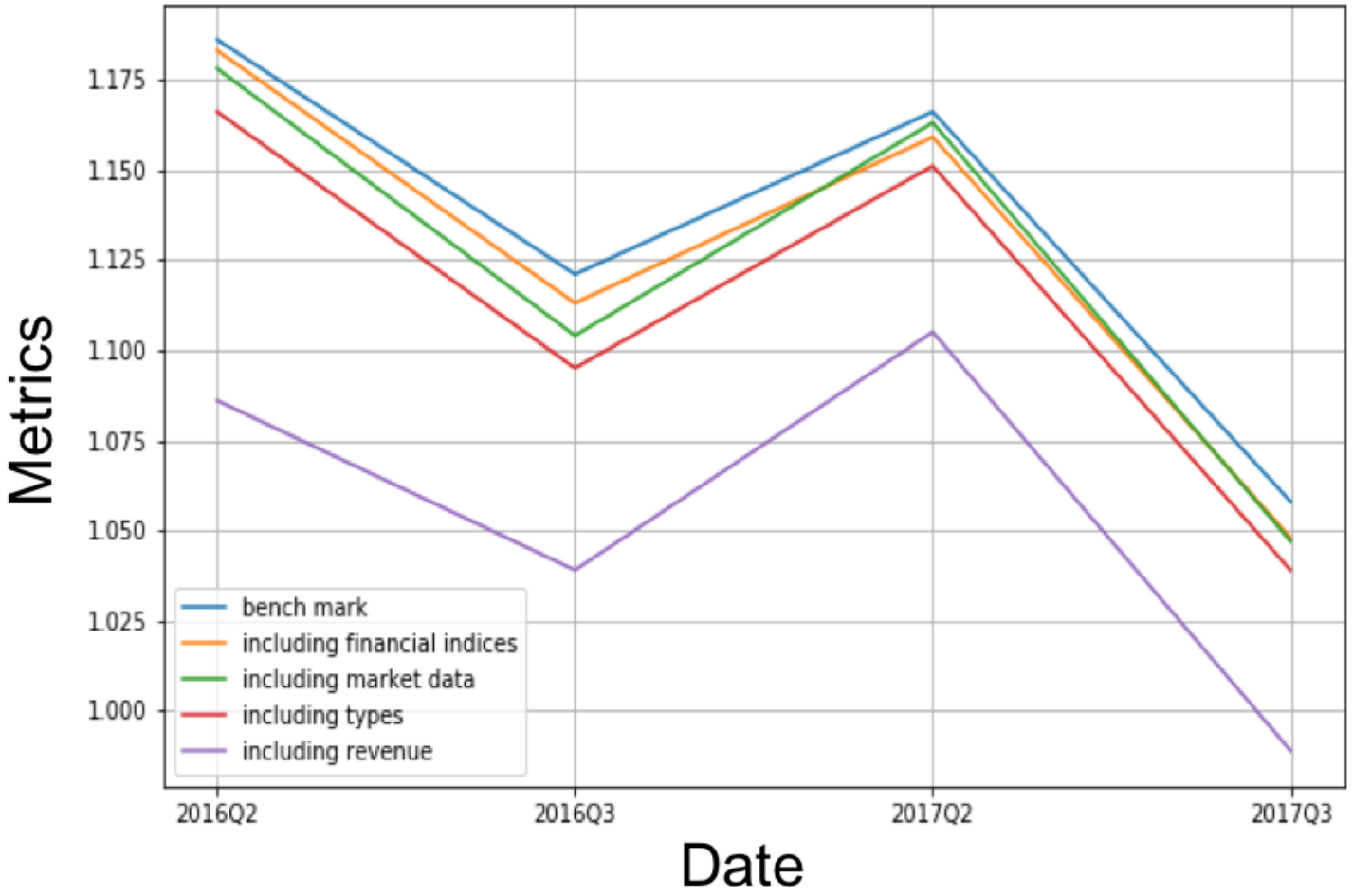
深度学习模型
(Deep Learning Model)
神经网络模型
(Neural Network)



营业收入(Revenue) ,
市场数据 (Market Data)

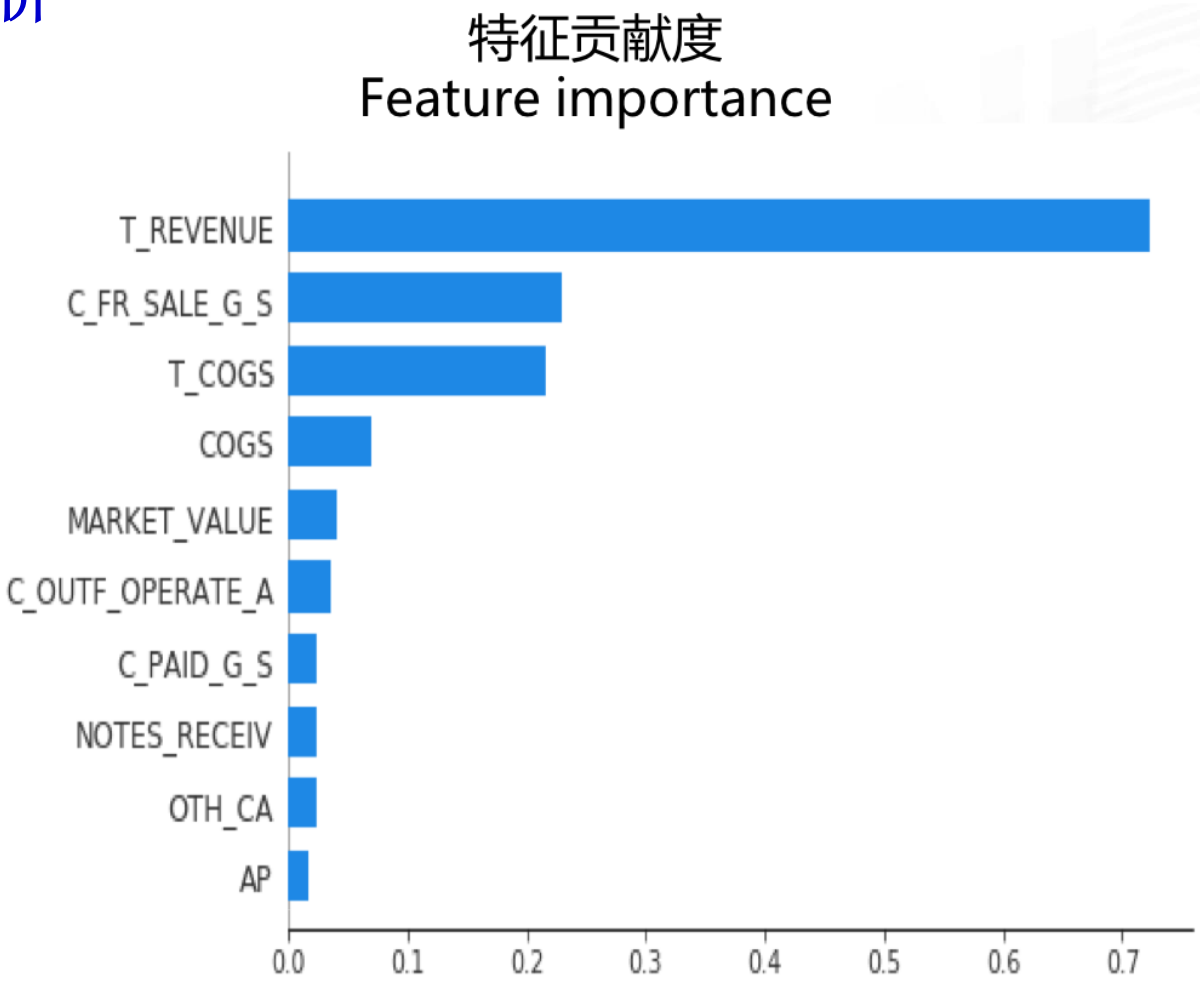
实际案例

- 多因子选择


Metrics
1.175
1.150
1.125
1.100
1.075
1.050
1.025
1.000
bench mark
including financial indices
including market data
including types
including revenue
2016Q2
2016Q3
2017Q2
2017Q3
Date

实际案例

- 特征评价


特征贡献度
Feature importance
T_REVENUE
C_FR_SALE_G_S
T_COGS
COGS
MARKET_VALUE
C_OUTF_OPERATE_A
C_PAID_G_S
NOTES_RECEIV
OTH_CA
AP
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7

不平衡类问题

- 属于不同类的实例数量不成比例
  - 不合格产品数量远低于合格产品的数量
  - **class 1:class2 > 4:1**
- 稀有类的正确分类比多数类的正确分类更有价值
- 准确率不适合评价从不平衡数据集得到的模型
- 许多已有的算法不能很好的检测稀有类的实例

不平衡类问题

- 稀有类通常记为正类，多数类被认为是负类

		预测的类	
		+	-
实际的类	+	f_{++}(TP)	f_{+-}(FN)
	-	f_{-+}(FP)	f_{--}(TN)

- 精度 $p = \frac{TP}{TP + FP}$

- 召回率 $r = \frac{TP}{TP + FN}$

召回率和精度的调和均值

$$F_1 = \frac{2rp}{r + p}$$

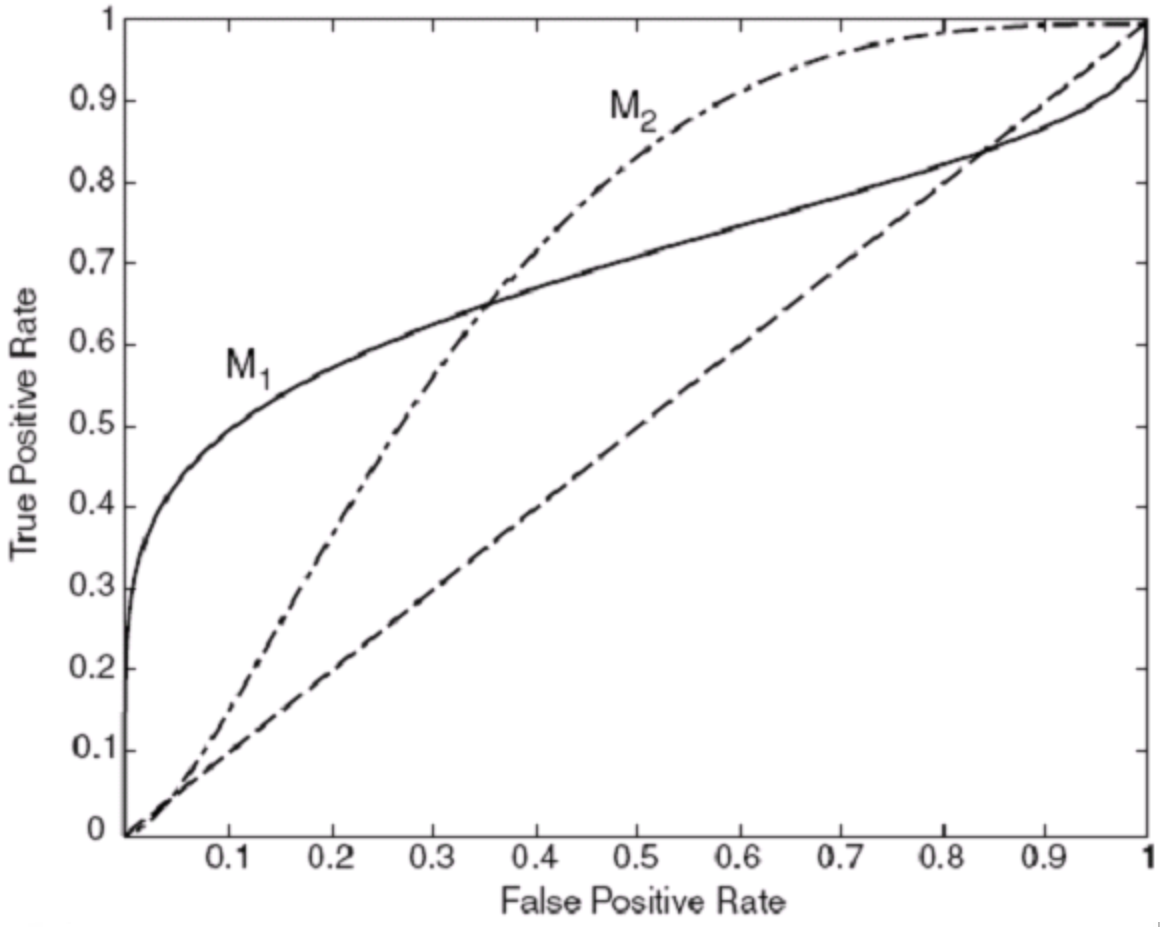
不平衡类问题

- ROC曲线：接收者操作特性曲线，显示分类器真正率和假正率之间折中的图形化方法
  - TPR=TP/(TP+FN)
    - 纵轴，越大越好
  - FPR=FP/(TN+FP)
    - 横轴，越小越好

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

不平衡类问题

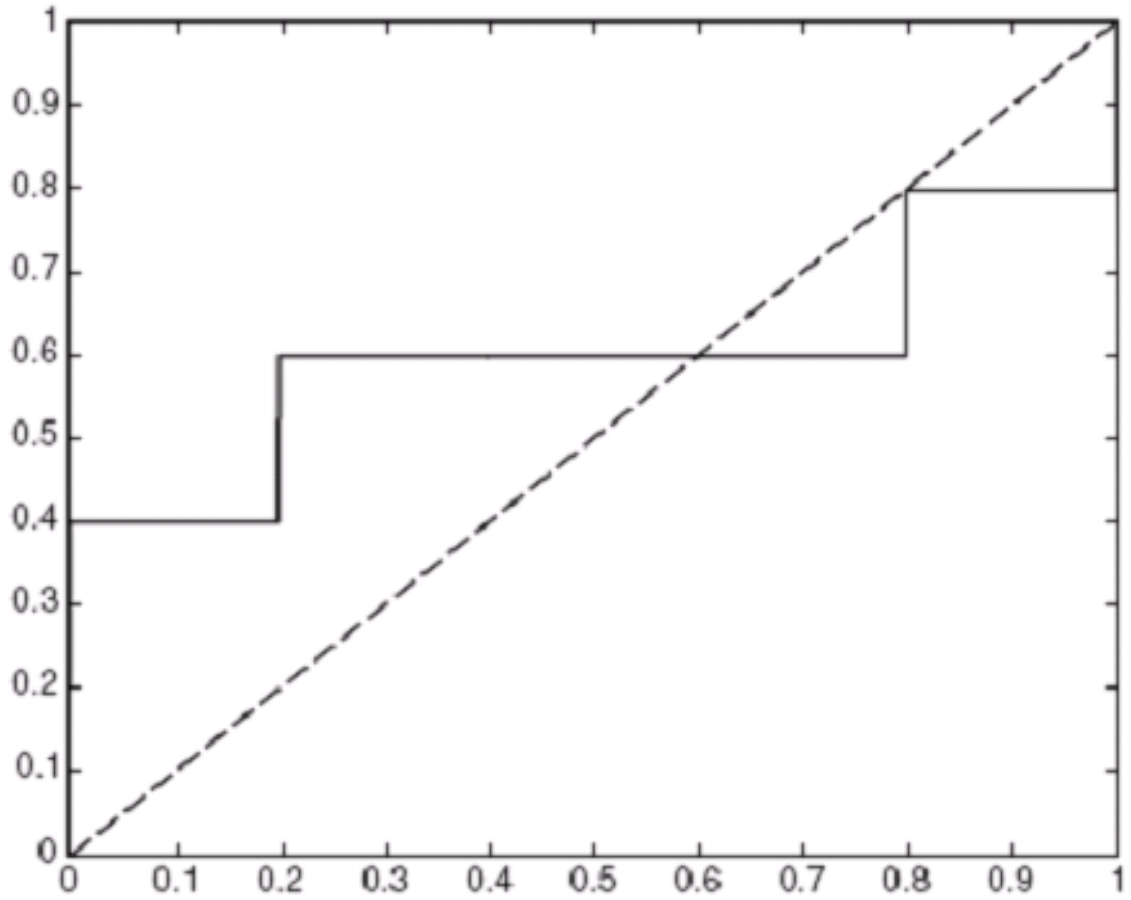
- (T=0,F=0)
  - 每个实例都预测为负类
- (T=1,F=1)
  - 每个实例都预测为正类
- (T=1,F=0)
  - 理想模型



不平衡类问题

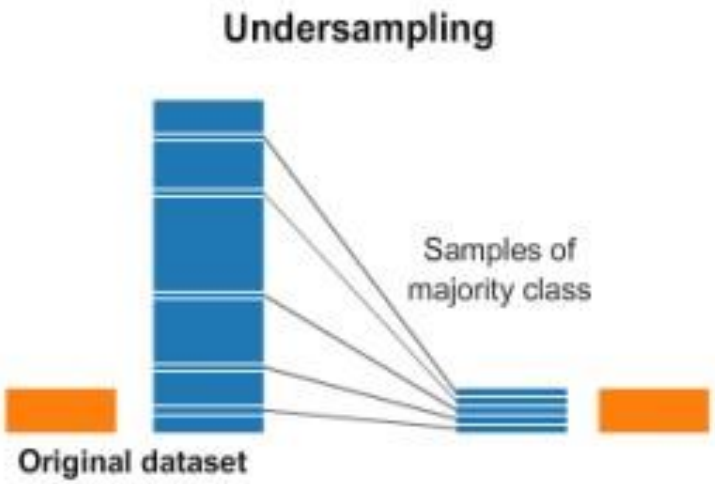
Class	+	−	+	−	−	−	+	−	+	+	
	0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95	1.00
TP	5	4	4	3	3	3	3	2	2	1	0
FP	5	5	4	4	3	2	1	1	0	0	0
TN	0	0	1	1	2	3	4	4	5	5	5
FN	0	1	1	2	2	2	2	3	3	4	5
TPR	1	0.8	0.8	0.6	0.6	0.6	0.6	0.4	0.4	0.2	0
FPR	1	1	0.8	0.8	0.6	0.4	0.2	0.2	0	0	0

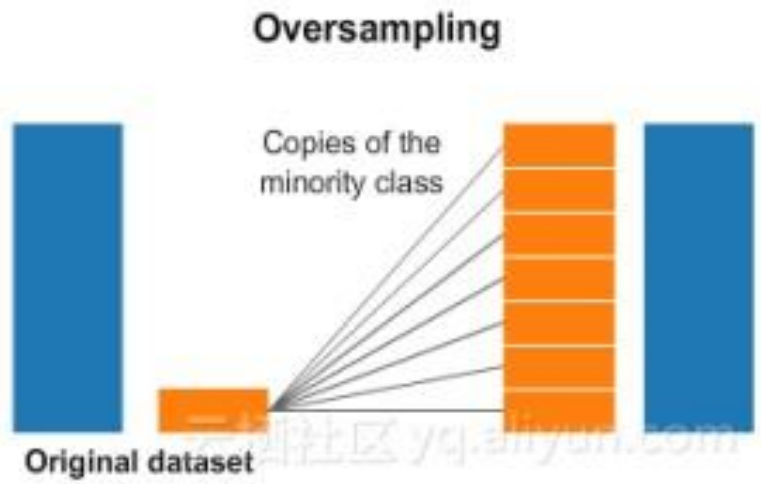
不平衡类问题



不平衡类问题

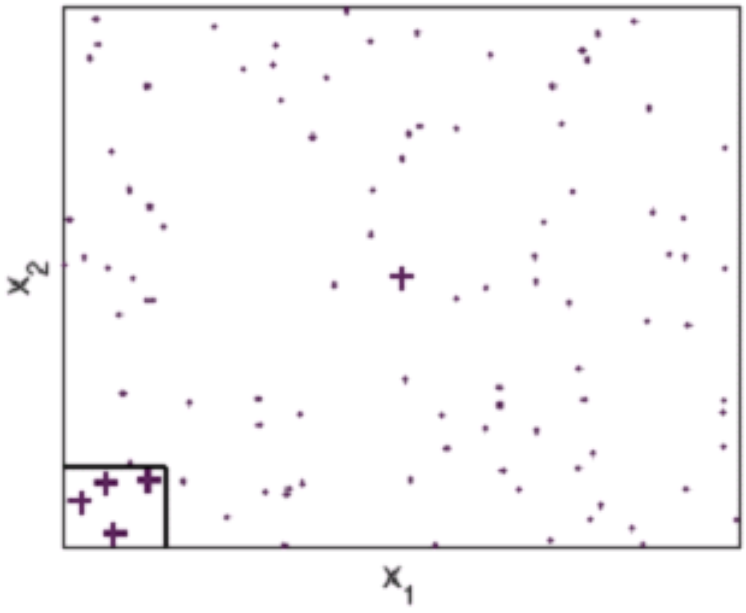
- 对数据集进行重采样
  - 对小样本过采样，对大样本欠采样



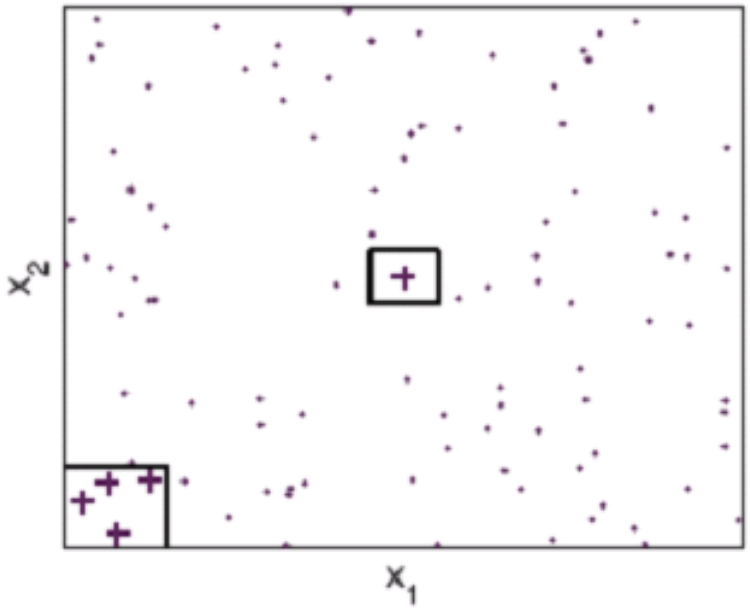


不平衡类问题

- 抽样技术通过改变实例的分布，从而使得稀有类在训练数据集中得到很好的表示



(a) Without oversampling



(b) With oversampling

不平衡类问题

- 产生人工数据样本
  - **SMOTE**：过采样算法，它构造新的小类样本而不是产生小类中已有的样本的副本
  - 产生的策略是对每个少数类样本**a**，从它的最近邻中随机选一个样本**b**，然后在**a**、**b**之间的连线上随机选一点作为新合成的少数类样本

不平衡类问题

- 尝试不同的分类算法
- 尝试对模型进行惩罚
  - 对小类样本数据增加权值，降低大类样本的权值
- 尝试将小样本视作异常点，转化为异常点检测问题来解决

多类问题

- 1. 分解成K个二类问题，其中所有属于y_i的样本被看作正类，其他样本作为负类
  - 1 vs K-1
- 2. 分解成K(K-1)/2个二类分类器，每个用来区分一对类(y_i, y_j)
  - 1 vs 1