提纲

南京大学 NANJING UNIVERSITY

- **样例学习**
  - 监督学习
  - 决策树学习
  - 模型选择与模型优化
  - 线性回归与分类
  - 非参数模型
  - 集成学习

学习的形式

南京大学 NANJING UNIVERSITY

- 伴随输入有3种类型的反馈（feedback），它们决定了3种类型的学习:
  - 监督学习
    - 智能体观测到输入-输出对
    - 学习从输入到输出的一个函数映射
  - 无监督学习
    - 智能体从没有任何显式反馈的输入中学习模式
    - 聚类（clustering）：通过输入样例来检测潜在的有价值的聚类簇
  - 强化学习
    - 智能体从一系列强化（奖励和惩罚）中学习

监督学习

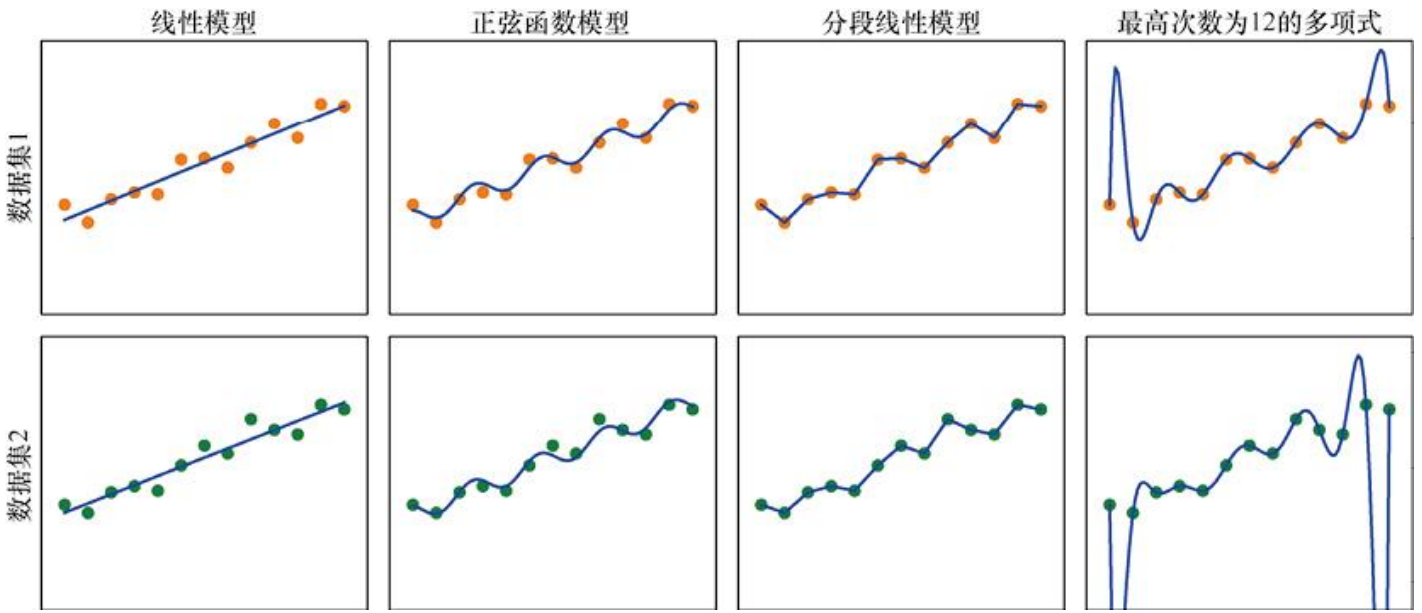
- 给定一个训练集含有N个“输入–输出”对样例：
  - $(x_1, y_1), (x_2, y_2), \ldots (x_N, y_N)$,
  - $y = f(x)$

- 函数h 被称为关于世界的假设来近似真实的函数 f
  - 取自一个假设空间H，其中包含所有可能的函数
  - h是关于数据的模型,取自模型类 H

- **一致性假设**: 假设 h 使得训练集中任意一个 x_i 都有 $h(x_i) = y_i$.
  - 寻找最佳拟合函数使得 $h(x_i)$ 接近于 y_i

- 衡量一个假设的标准不是看它在训练集上的表现，而是取决于它如何处理尚未观测到的输入。
  - 使用一个测试集 (x_i, y_i) 来评估假设，如果h 准确地预测了测试集的输出，我们称h具有很好的泛化能力。

监督学习

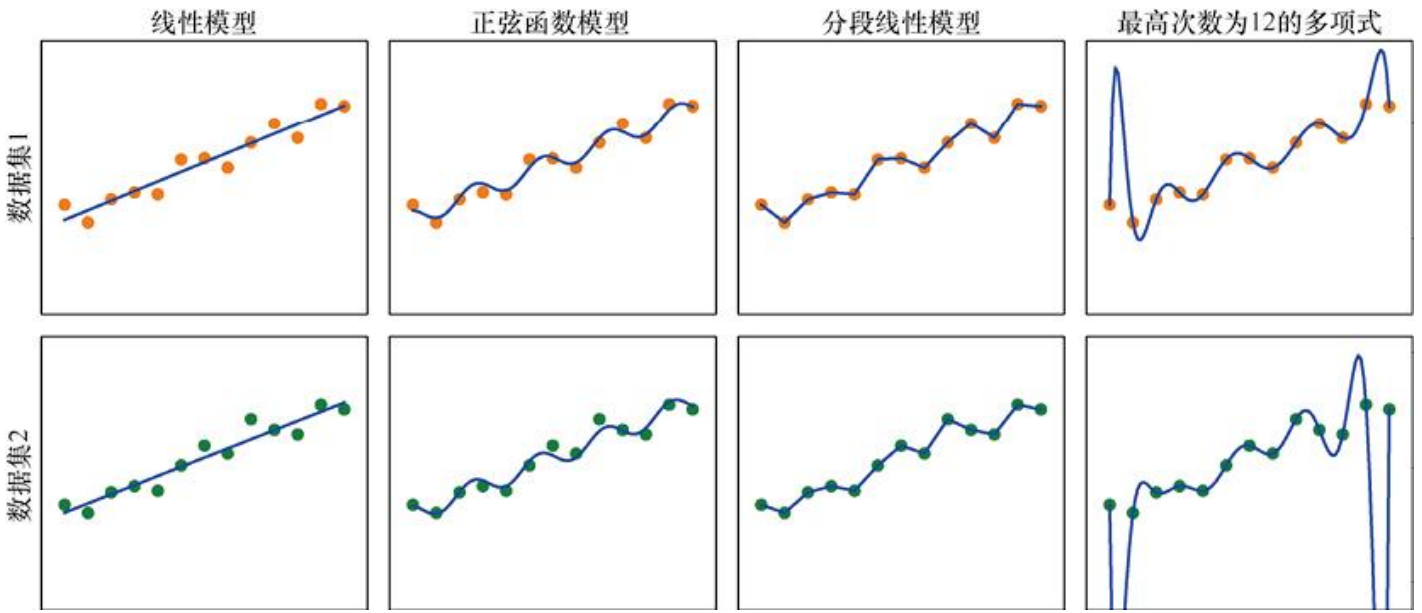


寻找拟合数据的假设。第一行：在数据集1上训练的来自4个不同假设空间的最佳拟合函数的4个图像。第二行：同样的4个函数，但是在稍有不同的数据集上进行训练得到的结果（数据集采样自相同的函数f (x)）

监督学习

- 使用**偏差（*bias*）**分析假设空间
  - 在不同的训练集上，假设所预测的值偏离期望值的平均趋势。
- **欠拟合（*underfitting*）：**无法找到数据中的模式。
- **方差（*variance*）：**指由训练数据波动而导致假设的变化量。
- **过拟合（*overfitting*）：**当一个函数过于关注它用来训练的特定训练数据集，进而导致它在没有见过的数据上表现较差。
- **偏差-方差权衡（*bias–variance tradeoff*）**:在更复杂、低偏差的能较好拟合训练集的假设与更简单、低方差的可能泛化得更好的假设中做出选择。
  - 选择与数据相符的最简单的假设。
  - **奥卡姆剃刀原则（Ockham's razor）：**如无必要，勿增实体。

监督学习



寻找拟合数据的假设。第一行：在数据集1上训练的来自4个不同假设空间的最佳拟合函数的4个图像。第二行：同样的4个函数，但是在稍有不同的数据集上进行训练得到的结果（数据集采样自相同的函数f (x)）

监督学习

- 相比于仅仅判断一个假设是可能还是不可能的，给出一个假设可能发生的概率是更合理的策略。
- 给定数据集，概率最大的假设 h^*：

$$h^* = \underset{h \in H}{\operatorname{argmax}} \; P(h|data)$$

- 根据贝叶斯法则，上式等价于

$$h^* = \operatorname{argmax} \; P(data|h)\, P(h)$$

问题示例：餐厅等待问题

- 决定是否在一家餐厅等待位置的问题。
- 输出y是一个布尔变量，我们将其称为是否等待（*WillWait*）
- 输入x是有10个属性值的向量，每个属性都是离散的值:
  1. 候补（Alternate）：附近是否存在其他合适的可以代替的餐厅
  2. 吧台（Bar）：该餐厅是否有舒适的吧台用于等待
  3. 周五/六（Fri/Sat）：今天是否为周五或周六
  4. 饥饿（Hungry）：现在是不是饿了
  5. 顾客（Patrons）：目前餐厅有多少顾客（值为None、Some、Full）
  6. 价格（Price）：餐厅的价格范围（$、$$,、$$$）
  7. 下雨（Raining）：外面是否正在下雨.
  8. 预约（Reservation）：我们是否有预订.
  9. 种类（Type）：餐厅种类（French、Italian、Thai或Burger）.
  10. 预计等待（WaitEstimate）：对等待时间的估计（0～10分钟 10～30分

监督学习

样例	输入属性										输出
	Alternate	*Bar*	*Fri/Sat*	*Hungry*	*Patrons*	*Price*	*Raining*	*Reservation*	*Type*	*WaitEstimate*	*WillWait*
x_1	Yes	No	No	Yes	Some	$$$	No	Yes	French	0 ~ 10	y_1 = Yes
x_2	Yes	No	No	Yes	Full	$	No	No	Thai	30 ~ 60	y_2 = No
x_3	No	Yes	No	No	Some	$	No	No	Burger	0 ~ 10	y_3 = Yes
x_4	Yes	No	Yes	Yes	Full	$	Yes	No	Thai	10 ~ 30	y_4 = Yes
x_5	Yes	No	Yes	No	Full	$$$	No	Yes	French	> 60	y_5 = No
x_6	No	Yes	No	Yes	Some	$$	Yes	Yes	Italian	0 ~ 10	y_6 = Yes
x_7	No	Yes	No	No	None	$	Yes	No	Burger	0 ~ 10	y_7 = No
x_8	No	No	No	Yes	Some	$$	Yes	Yes	Thai	0 ~ 10	y_8 = Yes
x_9	No	Yes	Yes	No	Full	$	Yes	No	Burger	> 60	y_9 = No
x_{10}	Yes	Yes	Yes	Yes	Full	$$$	No	Yes	Italian	10 ~ 30	y_{10} = No
x_{11}	No	No	No	No	None	$	No	No	Thai	0 ~ 10	y_{11} = No
x_{12}	Yes	Yes	Yes	Yes	Full	$	No	No	Burger	30 ~ 60	y_{12} = Yes

餐厅等待问题领域的样例（样本）

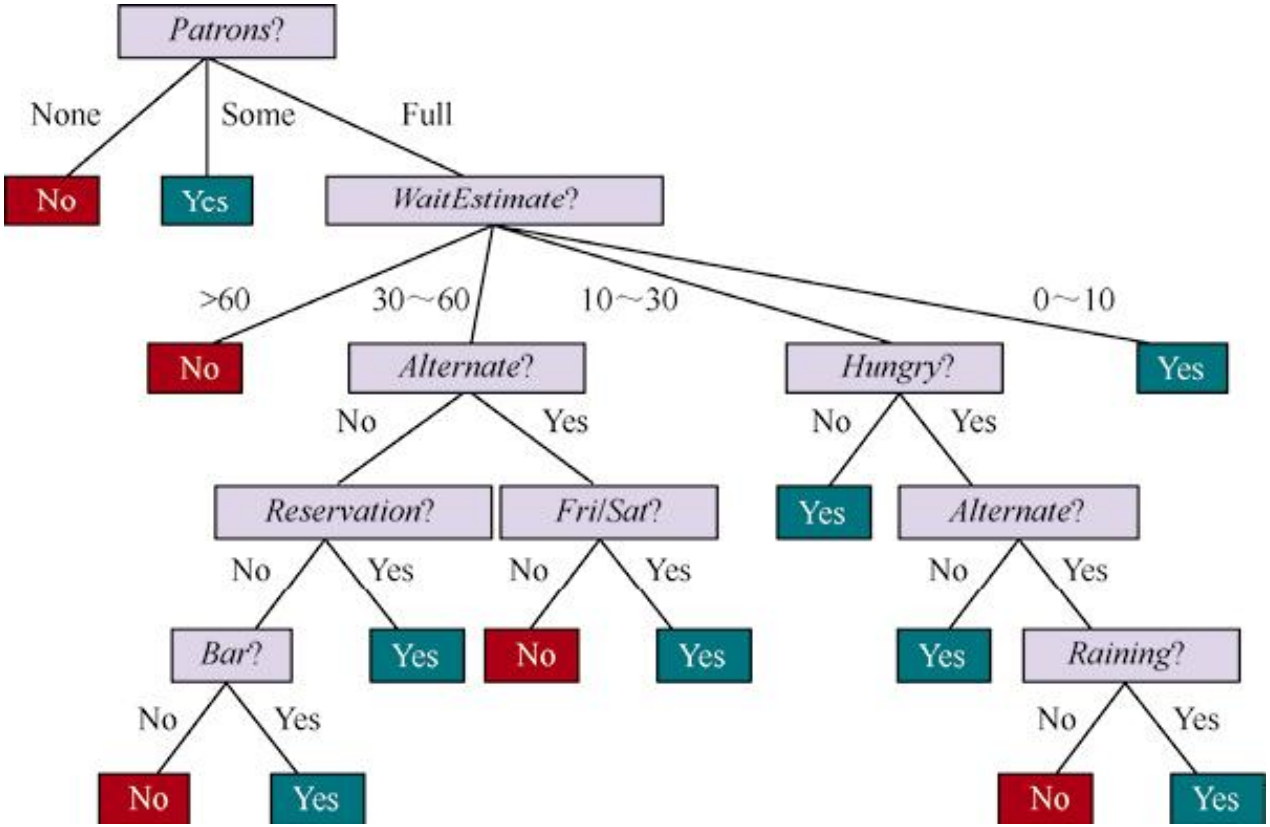
决策树学习

- 决策树（*decision tree*）将属性值向量映射到单个输出值（即“决策”）。
  - 执行一系列测试来实现其决策，它从根节点出发，沿着适当的分支，直到到达叶节点为止。
  - 树中的每个内部节点对应于一个输入属性的测试
  - 该节点的分支用该属性的所有可能值进行标记
  - 叶节点指定了函数要返回的值

- 布尔型的决策树等价于如下形式的逻辑语句：

$$Output \Leftrightarrow (Path_1 \vee Path_2 \vee \cdots)$$

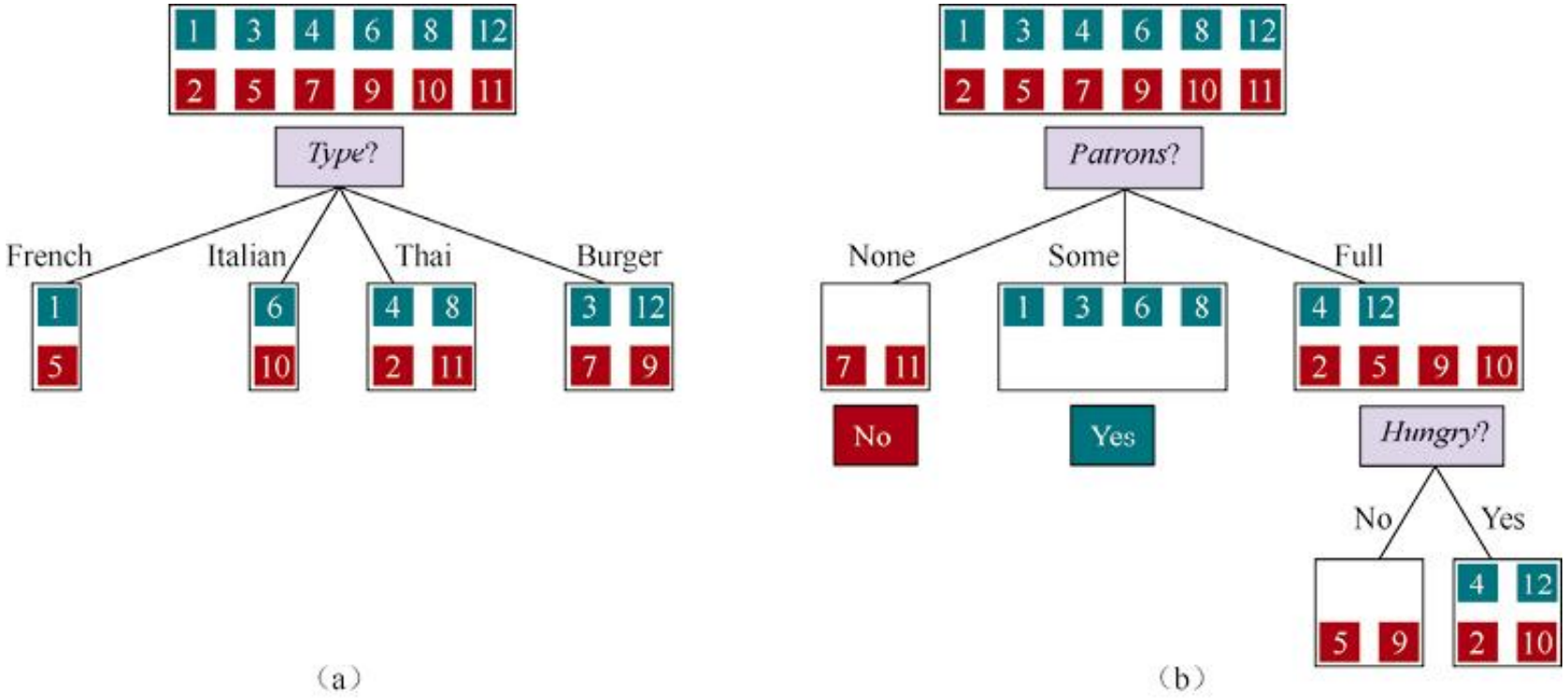
决策树学习



决定是否在餐厅等待的决策树

决策树学习

从样例中学习决策树



通过测试属性来对样例进行分割。在每一个节点中我们给出剩余样例的正（绿色方框）负（红色方框）情况。（a）根据*Type*分割样例，没有为我们分辨正负带来帮助。（b）根据*Patrons*分割样例，很好地区分了正负样例。在根据*Patrons*进行分类之后，*Hungry*是相对较好的第二个测试属性

决策树学习

南京大学 NANJING UNIVERSITY

从样例中学习决策树

function LEARN-DECISION-TREE(*examples, attributes, parent examples*) **returns** 一棵树

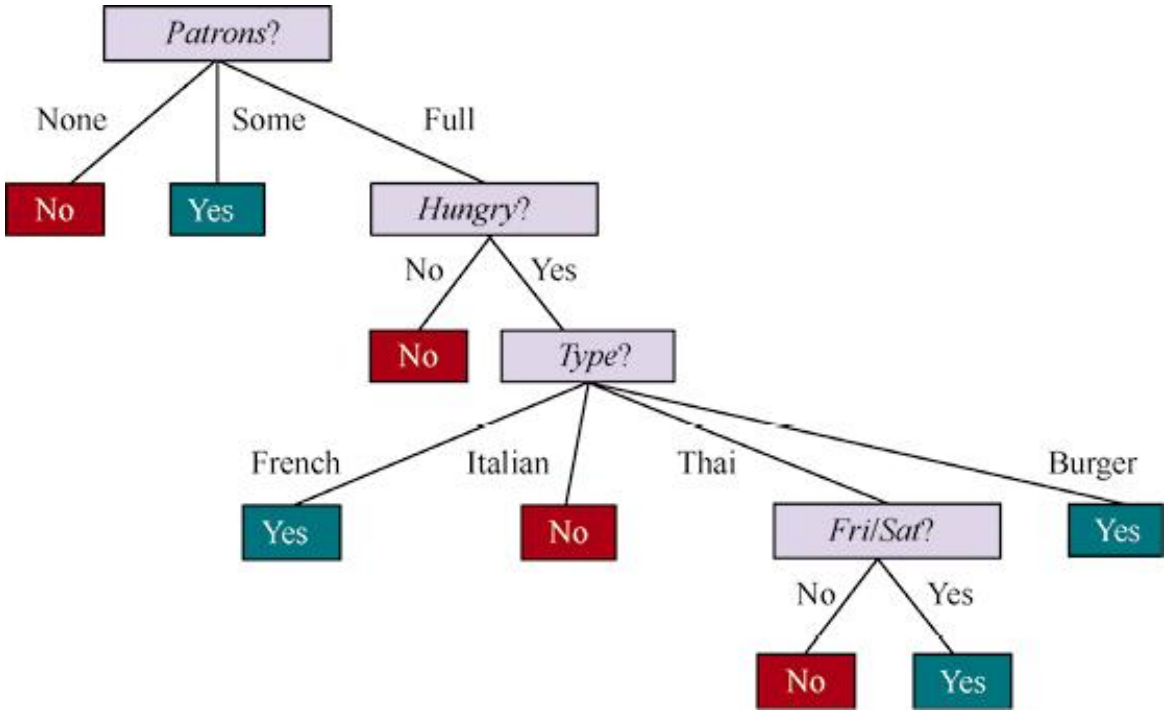
if *examples*不为空 **then return** PLURALITY-VALUE(*parent examples*)
else if 所有*examples*有相同的分类 **then return** 分类
else if *attributes*为空 **then return** PLURALITY-VALUE(*examples*)
else
  $A \leftarrow \text{argmax}_{a \in attributes}$ IMPORTANCE(a, *examples*)
  tree ← 一个以测试A为根的新的决策树
  for each A中的值v **do**
    $exs \leftarrow \{e : e \in examples \text{ and } e.A = v\}$
    subtree←LEARN-DECISION-TREE (*exs*, *attributes*−A, *examples*)
    将一个带有标签（$A = v$）和子树 *subtree*的分支加入*tree*
  return *tree*

决策树学习算法。函数Plurality-Value将选择样例集中占比最高的输出，并随机地断开连接

决策树学习

南京大学 NANJING UNIVERSITY

从样例中学习决策树

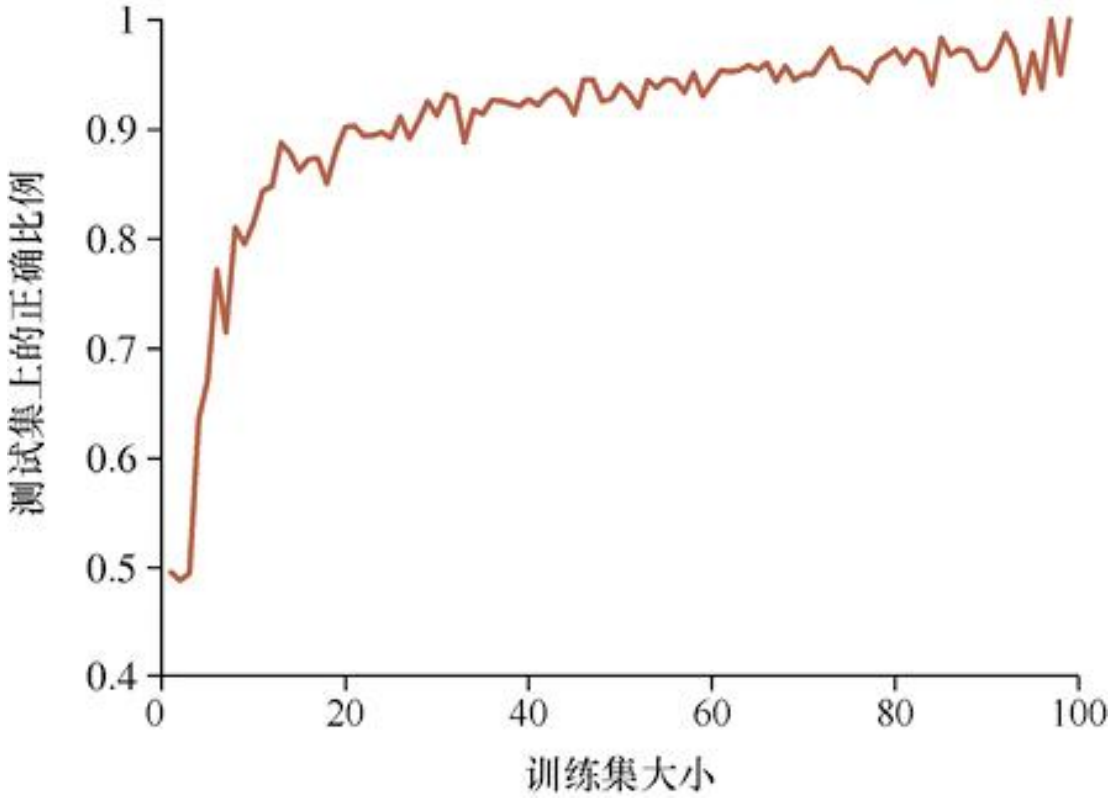


根据12样例训练集推断出的决策树

决策树学习

南京大学 NANJING UNIVERSITY

从样例中学习决策树



一个决策树的学习曲线，数据集为从餐厅等待问题领域中随机产生的100个样例。图中每个点都是20次试验的均值

选择测试属性

- **熵（entropy）**：是随机变量不确定性的度量;
  - 信息量越多，熵越小
  - 信息论中最基本的量

- 一般情况下，若一个随机变量V取值为 v_k 的概率为$P(v_k)$，那么它的熵定义为：

$$H(V)=\sum_k P(v_k)\log_2\frac{1}{P(v_k)}=-\sum_k P(v_k)\log_2 P(v_k).$$

- 通过测试属性A获得的信息增益定义为熵减少的期望值：

$$Gain(A)=B(\frac{p}{p+n})-Remainder(A).$$

$$Remainder(A)=\sum_{k=1}^{d}\frac{p_k+n_k}{p+n}B\left(\frac{p_k}{p_k+n_k}\right)$$

决策树学习

南京大学 NANJING UNIVERSITY

泛化与过拟合

- 决策树剪枝的技术可以用于减轻过拟合
  - 删去不明显相关的节点
- 我们需要多大的增益才能在特定属性上进行分割？
- 显著性检验
  - 从零假设开始
  - 对实际数据进行分析，并计算它们偏离零假设的程度
  - 如果偏离程度在统计上不太可能发生（通常我们取5%或更低的概率作为阈值），那么这在一定程度上证明了数据中仍存在显著的模式。
- 样本集包含 p 个正样本 和 n 个负样本。期望数量, $\hat{p}_k$ 和 $\hat{n}_k$,
- 衡量偏差以及总偏差

$$\hat{p}_k = p \times \frac{p_k + n_k}{p+n} \qquad \hat{n}_k = n \times \frac{p_k + n_k}{p+n}. \qquad \Delta = \sum_{k=1}^{d} \frac{(p_k - \hat{p}_k)^2}{\hat{p}_k} + \frac{(n_k - \hat{n}_k)^2}{\hat{n}_k}.$$

拓展决策树的适用范围

- 通过处理以下复杂情况，决策树可以得到更广泛的应用：
  - 缺失数据
  - 连续属性与多值输入属性
  - 连续值输出属性

- 决策树也是不稳定的，因为仅添加一个新的样例，就可能更改根上的测试结果，从而更改整个树。

模型选择与模型优化

南京大学 NANJING UNIVERSITY

- 找到一个好的假设这一目标可以分作两个子任务:
  - **模型选择：**选择一个好的假设空间；
  - **模型优化 (训练)** ：在这个空间中找到最佳假设。

- 一组为用于训练从而得到假设的训练集，另一组为用于评估假设的测试集。

- **错误率:** 对于样本 (x, y)，$h(x) \neq y$ 的比例

- 需要三个数据集:

1. 训练集用于训练备选模型
2. 验证集也被称为开发集，用 于评估备选模型并选择最佳的备选模型
3. 测试集用于无偏地估计最佳模型

- 当没有足够的数据来构成这3个数据集: k-折交叉验证
- 将数据分割成k个相等大小的子集；
- 然后进行k轮学习；
- 在每一轮中，$1/k$的数据被作为验证集，其余的样例被用于训练；
- 常用k值为5或10
- **留一交叉验证：** $k=n$

模型选择与模型优化

南京大学 NANJING UNIVERSITY

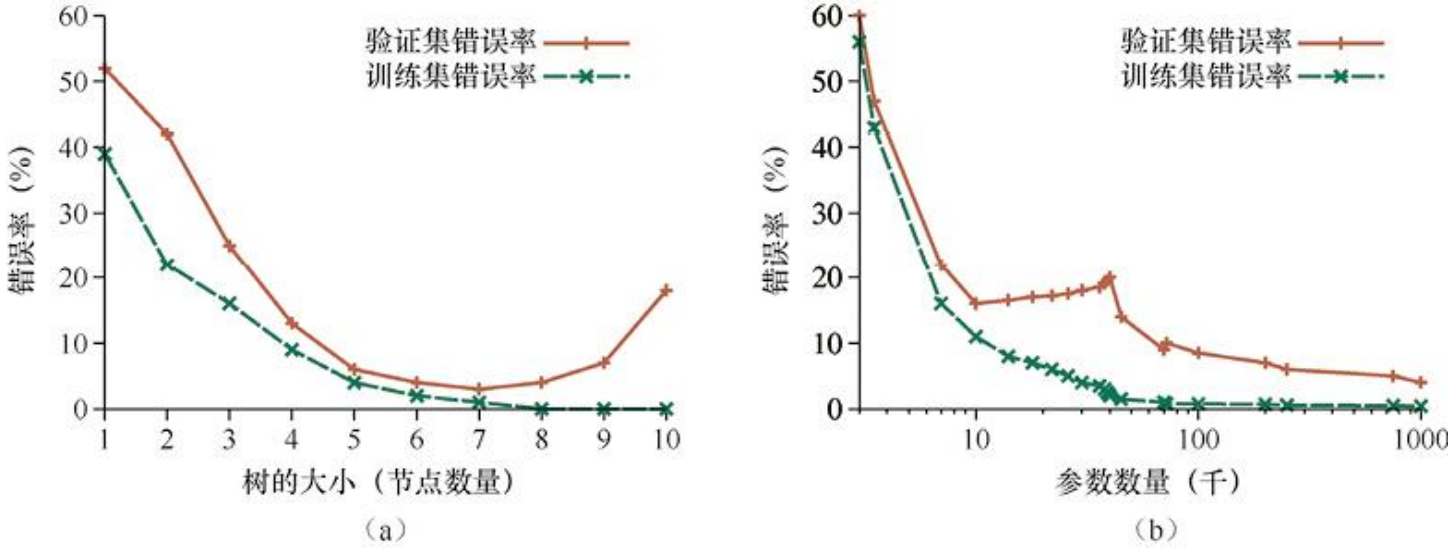
模型选择

```
function MODEL-SELECTION(Learner, examples, k) returns 一个(假设, 错误率)对
  err ← 一个数组，以size为索引，存储验证集错误率
  training_set, test_set ← examples划分成两个集合
  for size = 1 to ∞ do
      err[size] ← CROSS-VALIDATION(Learner, size, training_set, k)
      if err开始显著增长 then
          best_size ← 使err[size]最小的size值
          h ← Learner(best_size, training_set)
          return h, ERROR-RATE(h, test_set)

function CROSS-VALIDATION(Learner, size, examples, k) returns 错误率
  N ← examples的个数
  errs ← 0
  for i = 1 to k do
      validation_set ← examples[(i − 1) × N/k:i × N/k]
      training_set ← examples − validation_set
      h ← Learner(size, training_set)
      errs ← errs + ERROR-RATE(h, validation_set)
  return errs / k      // 验证集的平均错误率，k折交叉验证
```

选择验证误差最小的模型的算法

模型选择



两个不同问题上不同复杂性模型的训练误差（下方绿线）和验证误差（上方橙色线）

模型选择与模型优化

南京大学 NANJING UNIVERSITY

从错误率到损失函数

- 最小化损失函数而不是最大化效用函数. 损失函数 $L(x, y, \hat{y})$ 定义为当正确的答案为 $f(x) = y$ 时，模型预测出 $h(x) = \hat{y}$ 的效用损失量:

$$L(x, y, \hat{y}) = Utility\,(\text{给定输入}x\text{，使用}y\text{的结果}) - Utility\,(\text{给定输入}x\text{，使用}\hat{y}\text{的结果})$$

- 独立于 x的简化版本: $L(y, \hat{y})$
  - 绝对值损失: $L_1(y, \hat{y}) = |y - \hat{y}|$
  - 平方误差: $L_2(y, \hat{y}) = (y - \hat{y})^2$

- 学习智能体通过选择最小化目前观测到的所有输入-输出对的预期损失的假设，来使其期望效用最大化。

- 样本的先验概率分布 $\mathbf{P}(X, Y)$

模型选择与模型优化

南京大学 NANJING UNIVERSITY

从错误率到损失函数

- 假设 h（关于损失函数L）的期望泛化损失:

$$GenLoss_L(h) = \sum_{(x,y)\in\mathcal{E}} L(y,h(x))\,P(x,y)\,,$$

- **经验损失**

$$EmpLoss_{L,E}(h) = \sum_{(x,y)\in E} L(y,h(x))\,\frac{1}{N}\,.$$

- 所估计的最佳假设 $\hat{h}^*$ 即为使得经验损失最小的假设

$$\hat{h}^* = \underset{h\in\mathcal{H}}{\operatorname{argmin}}\, EmpLoss_{L,E}(h)\,.$$

模型选择与模型优化

南京大学 NANJING UNIVERSITY

正则化

- 显式地惩罚复杂假设的复杂性。
- 最小化经验损失与假设复杂性度量的加权和：

$$Cost(h) = EmpLoss(h) + \lambda Complexity(h)$$
$$\hat{h}^* = \underset{h \in \mathcal{H}}{\operatorname{argmin}} Cost(h).$$

- 正则化函数的选择依赖于假设空间。

- **特征选择:**
  - 减少模型的维数
  - 丢弃不相关的属性

模型选择与模型优化

南京大学 NANJING UNIVERSITY

超参数调整

- **手动调参**：根据个人以往的经验来猜测参数，在该参数下训练模型，并在验证集上测试其表现并分析结果，根据直觉得到参数调整的结果
- **网格搜索**：尝试所有超参数值的组合，观察哪个组合在验证集上表现得最好。（适合只有几个超参数，且每个超参数都有比较少的可能值）
- **随机搜索**
- **贝叶斯优化**：将选择好的超参数值的任务本身视为一个机器学习问题。
- **基于群体的训练（*population-based training, PBT*）**

学习理论

南京大学 NANJING UNIVERSITY

- **概率近似正确的（*PAC*）：** 如果一个假设与足够多的训练样例相一致，那么它不太可能是严重不匹配的。

- ***PAC* 学习算法:** 返回 *PAC* 假设

- 当 error(h) 不大于 ε，假设 h 被称为近似正确, 其中 ε 是一个较小的常数

$$P(h_b \text{ 与 } N \text{ 个样例一致}) \leqslant (1-\varepsilon)^N$$

$$P(\mathcal{H}_{bad} \text{ 包含一个一致性假设}) \leqslant |\mathcal{H}_{bad}|(1-\varepsilon)^N \leqslant |\mathcal{H}|(1-\varepsilon)^N$$

- 我们希望能使这个事件发生的概率小于某个较小的正数

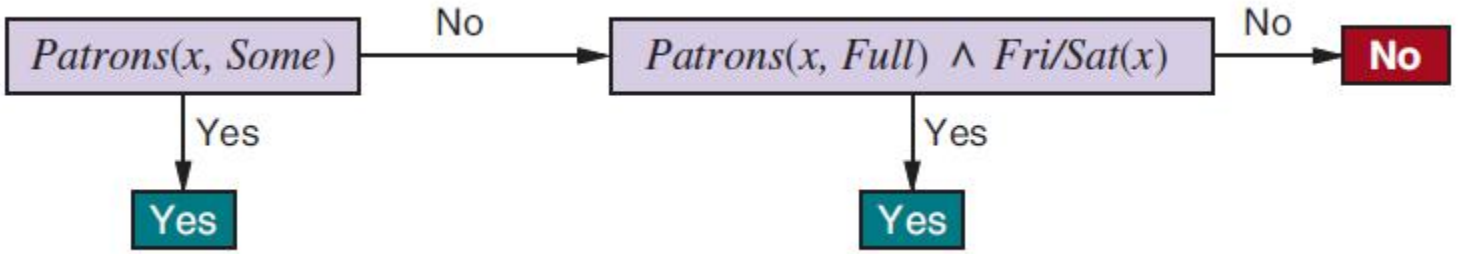
$$P(\mathcal{H}_{bad} \text{ 包含一个一致性假设}) \leqslant |\mathcal{H}|(1-\varepsilon)^N \leqslant \delta$$

- 由于$1-\varepsilon \leqslant e^{-\varepsilon}$

$$N \geq \frac{1}{\epsilon}\left(\ln\frac{1}{\delta} + \ln|\mathcal{H}|\right)$$

PAC学习示例：学习决策列表

$$WillWait \Leftrightarrow (Patrons = Some) \vee (Patrons = Full \wedge Fri/Sat) .$$



餐厅等待问题的决策列表

$$N \geq \frac{1}{\epsilon}\left(\ln\frac{1}{\delta} + O(n^k \log_2(n^k))\right)$$

function DECISION-LIST-LEARNING(*examples*) **returns** 一个决策列表或*failure*

if *examples*为空 **then return** 平凡的决策列表No

$t \leftarrow$ 与一个*examples*的非空子集$examples_t$对应的测试，

使得$examples_t$的元素均为正样例或者负样例

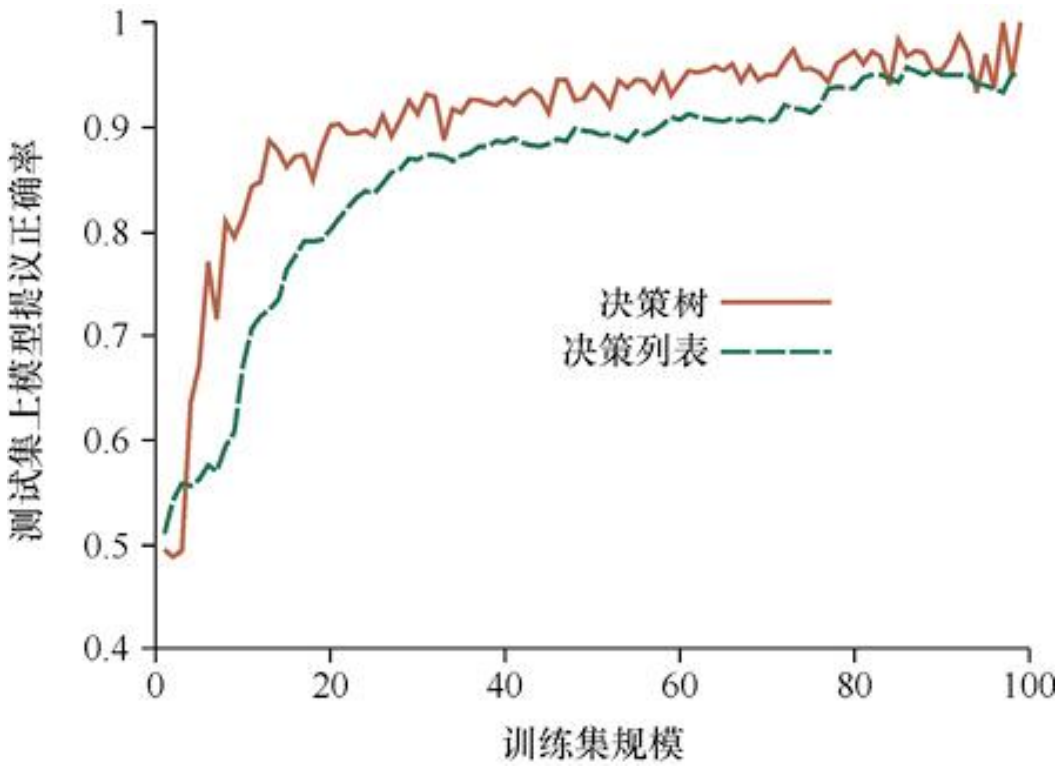
if 不存在这样的t **then return** *failure*

if 集合$examples_t$中的样例均为正 **then** $o \leftarrow$ Yes **else** $o \leftarrow$ No

return 一个初始测试为t，输出为o，

且由DECISION-LIST-LEARNING($examples - examples_t$)给出剩余测试的决策列表

学习决策列表的算法



Decision-List-Learning算法用于餐厅等待问题的学习曲线。Learn-Decision-Tree的曲线也在图中给出，用于比较；决策树在这个特定问题上表现得稍好一些

线性回归与分类

南京大学 NANJING UNIVERSITY

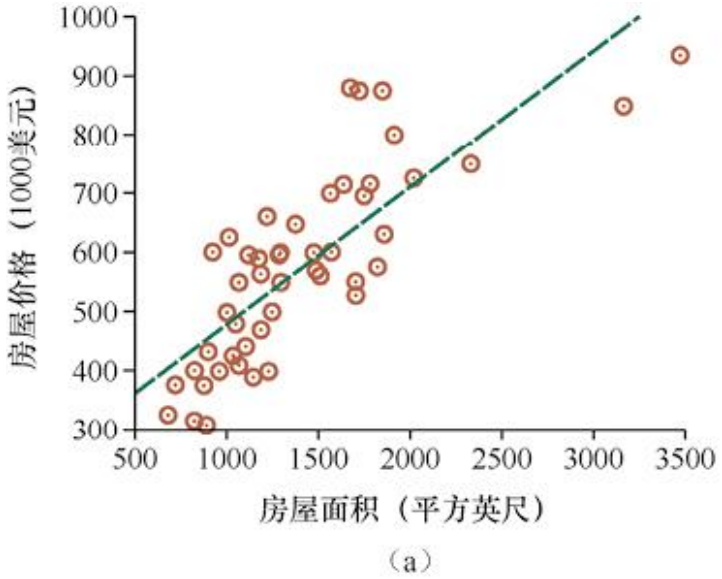
单变量线性回归

输入 x 和输出 y

$$y = w_1 x + w_0$$

线性函数

$$h_{\mathbf{w}}(x) = w_1 x + w_0$$



线性回归：找到最匹配这些数据的线性函数 $h_{\mathbf{w}}$

找到对应的权重值 (w_0, w_1) 使得其经验损失最小。

平方误差损失函数, L_2, 对所有训练样本求和：

$$Loss(h_{\mathbf{w}}) = \sum_{j=1}^{N} L_2(y_j, h_{\mathbf{w}}(x_j)) = \sum_{j=1}^{N} (y_j - h_{\mathbf{w}}(x_j))^2 = \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2 .$$

$$\frac{\partial}{\partial w_0} \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2 = 0 \text{ and } \frac{\partial}{\partial w_1} \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2 = 0 .$$

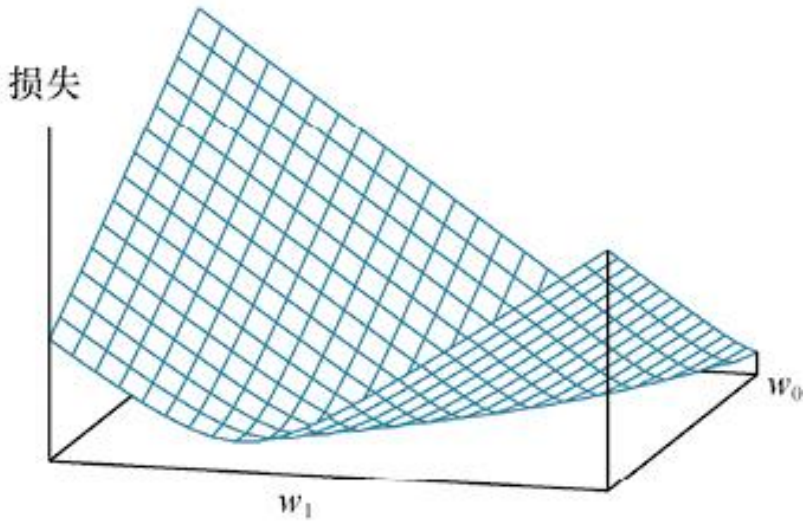
$$w_1 = \frac{N(\sum x_j y_j) - (\sum x_j)(\sum y_j)}{N(\sum x_j^2) - (\sum x_j)^2}; \quad w_0 = (\sum y_j - w_1(\sum x_j))/N .$$

线性回归与分类

南京大学 NANJING UNIVERSITY

单变量线性回归

$$\boldsymbol{w}^* = \text{argmin}_{\boldsymbol{w}} \, Loss(h_{\boldsymbol{w}})$$



（b）

$$Loss(h_{\mathbf{w}}) = \sum_{j=1}^{N} L_2(y_j, h_{\mathbf{w}}(x_j)) = \sum_{j=1}^{N} (y_j - h_{\mathbf{w}}(x_j))^2 = \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2 .$$

梯度下降

- 通过逐步修改参数来搜索连续的权重空间（最小化损失函数）

$\mathbf{w} \leftarrow$ any point in the parameter space
while not converged **do**
  for each w_i **in** $\mathbf{w}$ **do**

$$w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} Loss(\mathbf{w})$$

- α: 步长/学习率，可以是固定的常数，也可以随着学习过程的进行逐渐衰减。

$$\frac{\partial}{\partial w_i} Loss(\mathbf{w}) = \frac{\partial}{\partial w_i}(y - h_{\mathbf{w}}(x))^2 = 2(y - h_{\mathbf{w}}(x)) \times \frac{\partial}{\partial w_i}(y - h_{\mathbf{w}}(x))$$

$$= 2(y - h_{\mathbf{w}}(x)) \times \frac{\partial}{\partial w_i}(y - (w_1 x + w_0)) .$$

$$\frac{\partial}{\partial w_0} Loss(\mathbf{w}) = -2(y - h_{\mathbf{w}}(x)) \qquad \frac{\partial}{\partial w_1} Loss(\mathbf{w}) = -2(y - h_{\mathbf{w}}(x)) \times x$$

梯度下降

批梯度下降（Batch Gradient Descent）

- 最小化每个样例各自损失的总和

$$w_0 \leftarrow w_0 + \alpha \sum_j (y_j - h_{\mathbf{w}}(x_j)); \quad w_1 \leftarrow w_1 + \alpha \sum_j (y_j - h_{\mathbf{w}}(x_j)) \times x_j .$$

- 损失曲面是凸的
- 训练不会陷入局部极小值，到全局最小值的收敛性可以保证
- **轮（epoch）：**遍历了所有训练样例的一步更新

随机梯度下降（Stochastic gradient descent or SGD）

- 在每一步中随机选择少量训练样例
- 是从N个样例中选择一个大小为m的小批量（minibatch）样例
- 选择合适的m来利用并行向量运算

线性回归与分类

南京大学 NANJING UNIVERSITY

多变量线性回归

- $\mathbf{x}_j$ 是一个n元向量
- h :权重与输入向量的点积

$$h_{\mathbf{w}}(\mathbf{x}_j) = \mathbf{w} \cdot \mathbf{x}_j = \mathbf{w}^\top \mathbf{x}_j = \sum_i w_i x_{j,i}$$

- 最佳的权重向量 $\mathbf{w}^*$为最小化样例上的平方误差损失

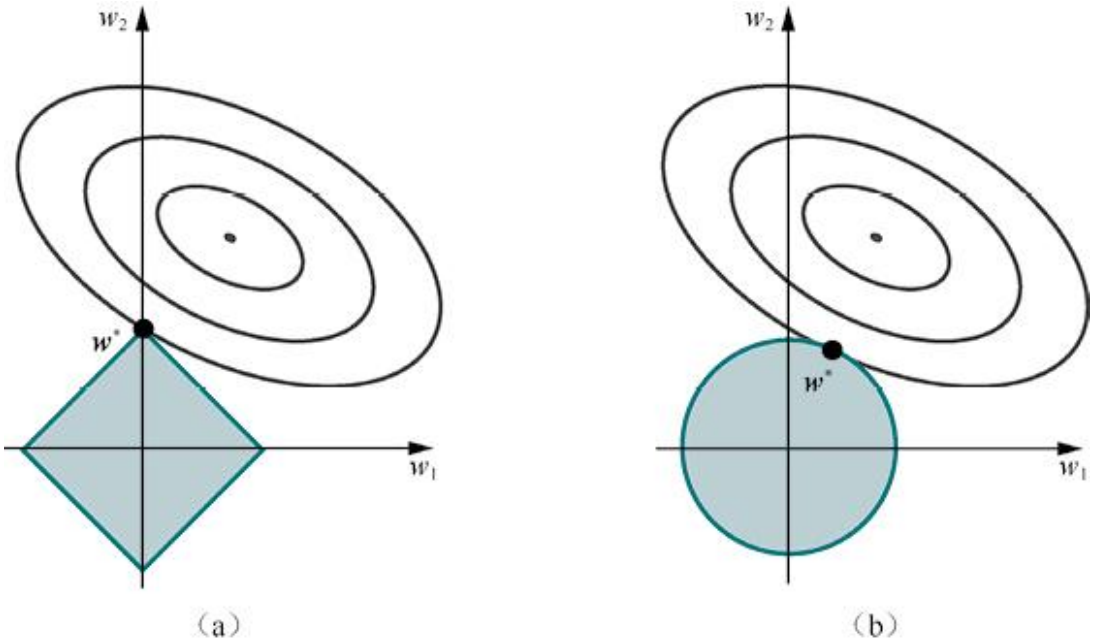
$$\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w}} \sum_j L_2(y_j, \mathbf{w} \cdot \mathbf{x}_j) .$$

- 梯度下降将收敛到损失函数的（唯一）最小值;每个权重 w_i的更新式：

$$w_i \leftarrow w_i + \alpha \sum_j (y_j - h_{\mathbf{w}}(\mathbf{x}_j)) \times x_{j,i} .$$

- 也可以写出解析解：

$$\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} .$$

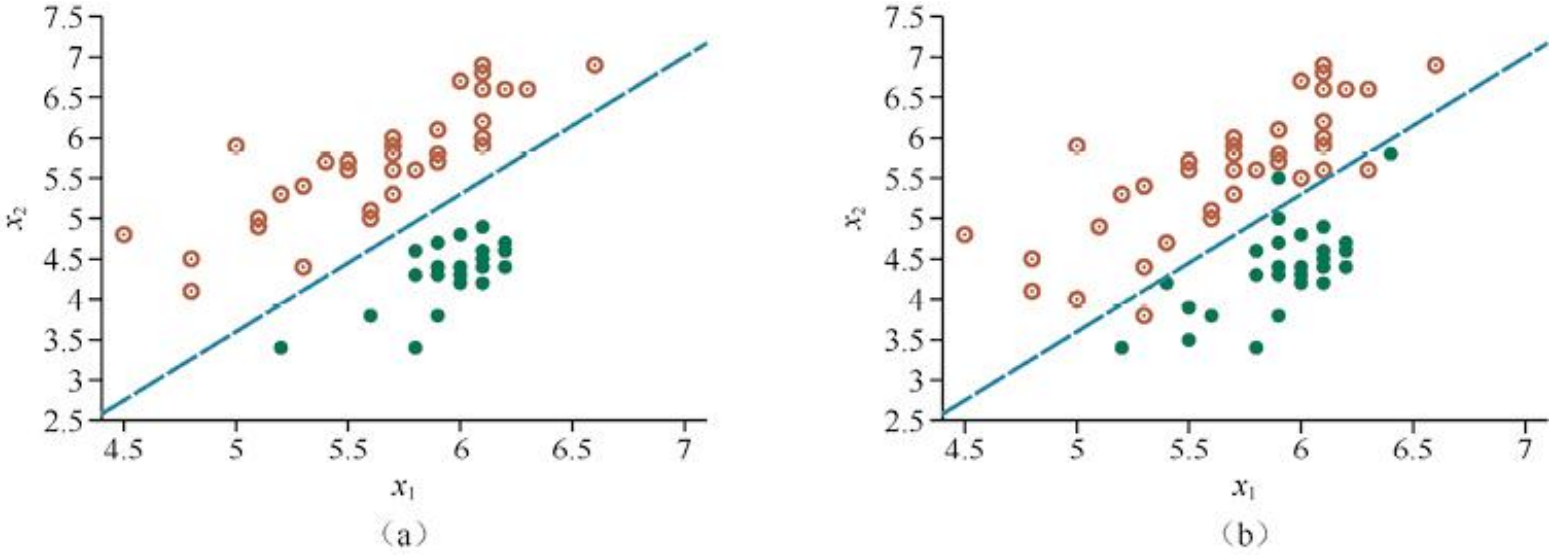


L_1正则化倾向于产生一个稀疏模型的解释。（a）在L_1正则化（菱形框）的情况下，正则化约束内的最小损失（同心等高线）一般会出现在某一条轴上，这意味着一个权重为0；（b）在L_2正则化（圆形）的情况下，最小损失可能出现在圆周上的任何位置，因而不会偏向0权重

线性回归与分类

南京大学 NANJING UNIVERSITY

带有硬阈值的线性分类器



（a）两种类型地震数据，包含体波震级x_1和面波震级x_2，数据来源于1982～1990年在亚洲和中东发生的地震（橙色空心圆）和地底爆炸（绿色实心圆）（Kebeasy *et al.*, 1998）。图中还绘制了类之间的决策边界。（b）同一个领域，有比之前更多的数据点。此时地震和爆炸不再是线性可分的

线性回归与分类

南京大学 NANJING UNIVERSITY

带有硬阈值的线性分类器

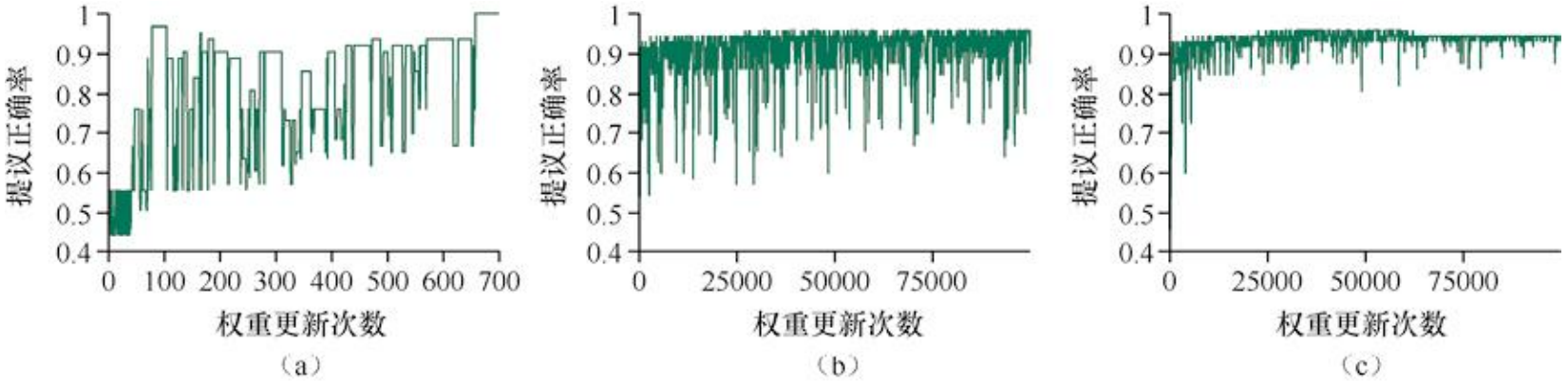
- 线性函数可用于回归，也可以用于分类.
- **决策边界（*decision boundary*）**是一条直线、一个平面或者更高维的面，将两类数据分离.
- **线性分离器:**对于线性可分数据线性决策边界

- **阈值函数**:
  $h_{\mathbf{w}}(\mathbf{x}) = Threshold(\mathbf{w} \cdot \mathbf{x})$ where $Threshold(z) = 1$ if $z \geq 0$ and 0 otherwise.

- 感知机学习规则: $w_i \leftarrow w_i + \alpha(y - h_w(\boldsymbol{x})) \times x_i$
  - 若输出是正确的（即$y = h_w(\boldsymbol{x})$），那么权重将不会改变。
  - 如果y为1而$hw(\boldsymbol{x})$为0，那么当相应的输入x_i是正数时w_i将增大，当x_i是负数时w_i将减小。这是有直观意义的，因为我们希望相应的$\boldsymbol{w} \cdot \boldsymbol{x}$会更大，进而使得$h_w(\boldsymbol{x})$的输出为1。
  - 如果y为0而$hw(\boldsymbol{x})$为1，那么当相应的输入x_i是正数时w_i将减小，当x_i是负数时w_i将增大。这是有直观意义的，因为我们希望相应的$\boldsymbol{w} \cdot \boldsymbol{x}$会更小，进而使得$h_w(\boldsymbol{x})$的输出为0。

线性回归与分类

带有硬阈值的线性分类器



（a）给定图19-15a中的地震/爆炸数据，模型在总训练集上的准确度与感知机学习规则训练的迭代次数的关系图。（b）对图19-15b中带噪声的、不可分离的数据绘制相同的图

基于逻辑斯谛回归的线性分类器

- 软化阈值函数—通过使用一个连续的、可微的函数来近似硬阈值函数
- 逻辑斯蒂函数:

$$Logistic(z) = \frac{1}{1+e^{-z}}$$

- 用来代替阈值函数:

$$h_{\mathbf{w}}(\mathbf{x}) = Logistic(\mathbf{w}\cdot\mathbf{x}) = \frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{x}}}.$$

- **逻辑斯谛回归：**拟合该模型的权重以最小化数据集上的损失的过程

基于逻辑斯谛回归的线性分类器

$$\begin{aligned}\frac{\partial}{\partial w_i}Loss(\boldsymbol{w}) &= \frac{\partial}{\partial w_i}(y-h_{\boldsymbol{w}}(\boldsymbol{x}))^2 \\ &= 2(y-h_{\boldsymbol{w}}(\boldsymbol{x}))\times\frac{\partial}{\partial w_i}(y-h_{\boldsymbol{w}}(\boldsymbol{x})) \\ &= -2(y-h_{\boldsymbol{w}}(\boldsymbol{x}))\times g'(\boldsymbol{w}\cdot\boldsymbol{x})\times\frac{\partial}{\partial w_i}\boldsymbol{w}\cdot\boldsymbol{x} \\ &= -2(y-h_{\boldsymbol{w}}(\boldsymbol{x}))\times g'(\boldsymbol{w}\cdot\boldsymbol{x})\times x_i\end{aligned}$$

$$g'(\boldsymbol{w}\cdot\boldsymbol{x}) = g(\boldsymbol{w}\cdot\boldsymbol{x})(1-g(\boldsymbol{w}\cdot\boldsymbol{x})) = h_{\boldsymbol{w}}(\boldsymbol{x})(1-h_{\boldsymbol{w}}(\boldsymbol{x}))$$

$$w_i \leftarrow w_i + \alpha(y-h_{\boldsymbol{w}}(\boldsymbol{x}))\times h_{\boldsymbol{w}}(\boldsymbol{x})(1-h_{\boldsymbol{w}}(\boldsymbol{x}))\times x_i$$

线性回归与分类

南京大学 NANJING UNIVERSITY

基于逻辑斯谛回归的线性分类器

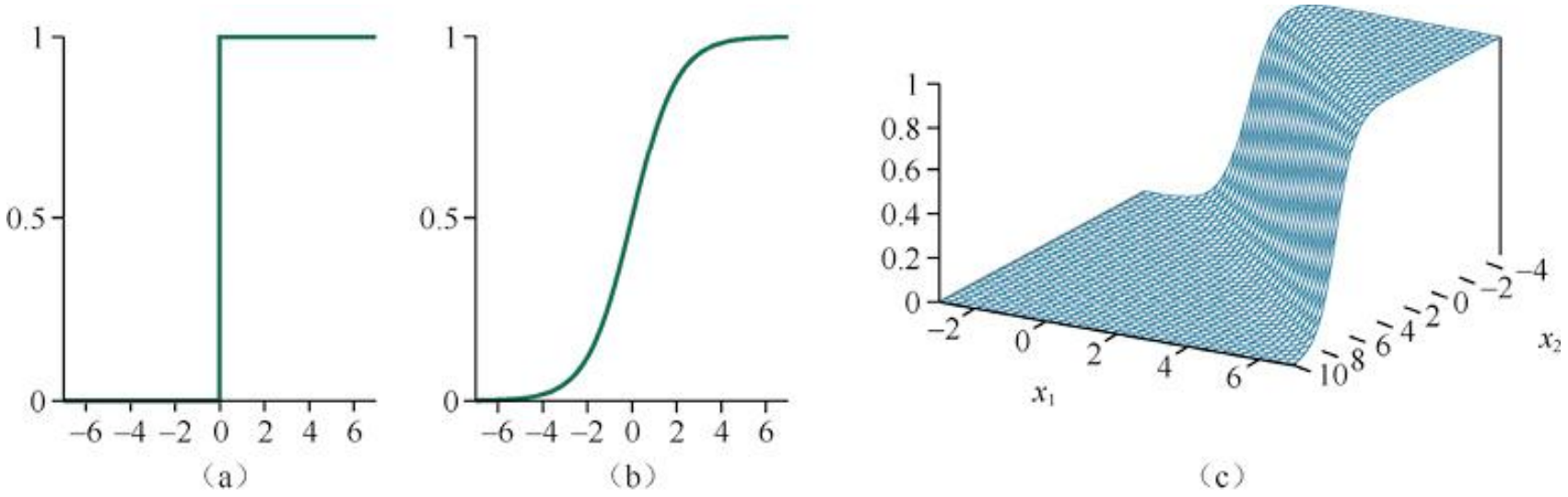
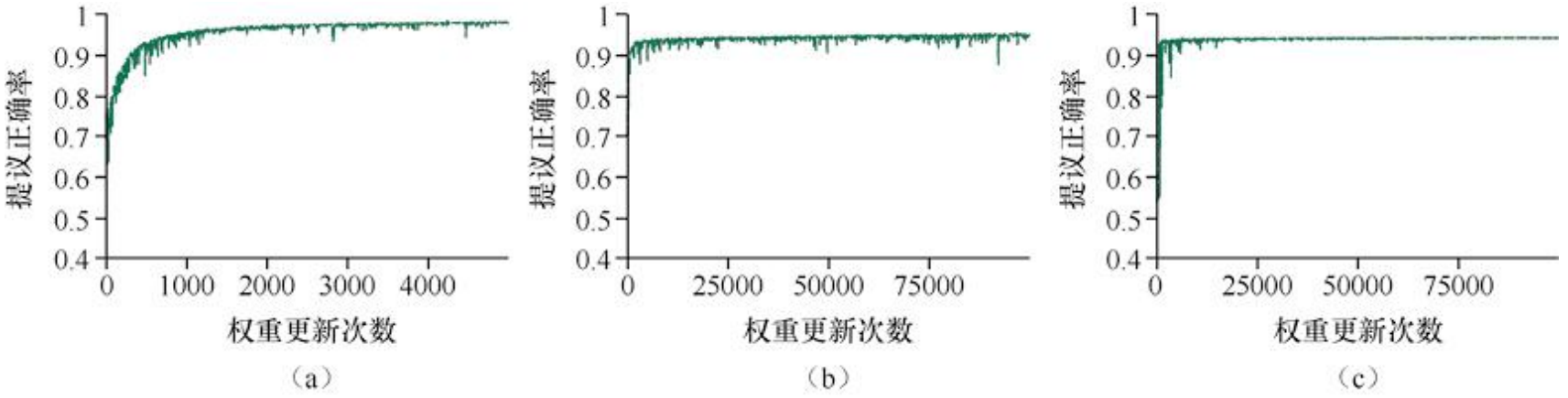


图19-17 (a) 硬阈值函数*Threshold*(*z*)，输出为0/1。注意，该函数在$z = 0$处是不可微的。(b) 逻辑斯谛函数$Logistic(z) = \frac{1}{1+e^{-z}}$，也称为sigmoid函数。(c) 对于图19-15b中数据的逻辑斯谛回归假设

$$h_{\mathbf{w}}(\mathbf{x}) = Logistic(\mathbf{w} \cdot \mathbf{x})$$

线性回归与分类

基于逻辑斯谛回归的线性分类器



重复使用逻辑斯谛回归进行图19-16中的实验。（a）中的图包含了5000次迭代，而不是700次，而（b）和（c）中的图使用与之前相同的坐标刻度

非参数模型

参数模型：通过参数个数固定的参数集（与训练样例的数量无关）来汇总数据信息的学习模型。

非参数模型：无法用参数个数固定的参数集来表示的模型

例如，分段线性函数将所有数据点保留为模型的一部分，也称为基于实例的学习（instance-based learning）或基于记忆的学习（memory-based learning）。

最简单的基于实例的学习方法是**查表**

- 取所有的训练样例，将它们放在查找表中，然后在需要访问$h(\mathbf{x})$时,查看$\mathbf{x}$是否在表中；如果是，则返回相应的 y.

非参数模型

南京大学 NANJING UNIVERSITY

最近邻模型

k-近邻

- 给定待查询的 $\mathbf{x}_q$, 寻找最接近$\mathbf{x}_q$的k个样例。
- 为了实现分类,我们寻找$\mathbf{x}_q$的一组邻居，并以占比最大的输出值为分类结果。例如 $k = 3$ 并且输出值为*Yes, No, Yes* , 则分类结果为 *Yes*.
- 为了实现回归，我们可以取k个邻居的平均值或中位数，也可以在最近邻邻居上求解一个线性回归问题。
- 闵可夫斯基距离（Minkowski distance）或 L^p范数

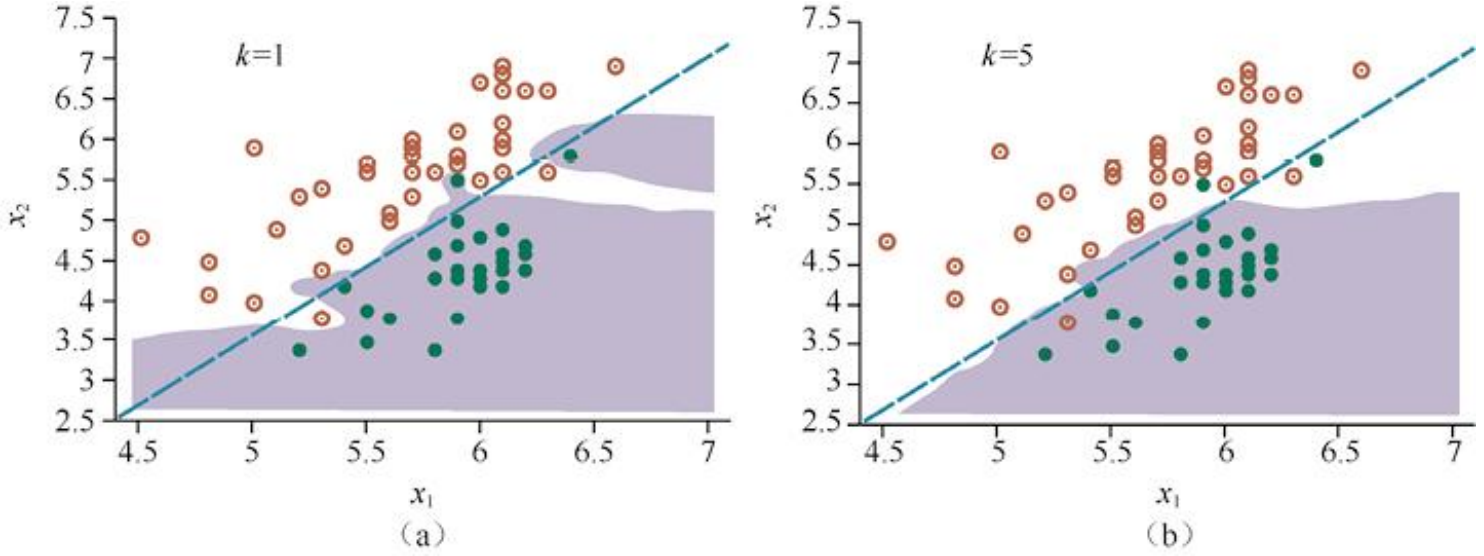
$$L^p(\mathbf{x}_j, \mathbf{x}_q) = (\sum_i |x_{j,i} - x_{q,i}|^p)^{1/p}.$$

- p =2, 欧几里得距离
- p =1, 曼哈顿距离
- 对于布尔属性值，汉明距离
- **马氏距离:** 考虑维度之间的协方差

$$D_M(x, y) = \sqrt{(x - y)^T \Sigma^{-1} (x - y)}$$

非参数模型

最近邻模型



（a）一个k近邻模型，该图显示了图19-15中数据的爆炸一类的范围，其中$k = 1$。可以明显看到有过拟合现象。（b）当$k = 5$时，该数据集上的过拟合现象消失了

非参数模型

南京大学 NANJING UNIVERSITY

局部敏感哈希

- 哈希表可能是一个比二叉树更快捷的查找方式。
- 我们希望将距离近的数据点分在同一组并分布于同一个箱子中，因此我们需要使用**局部敏感哈希（*locality-sensitive hash, LSH*）**。
- 创建多个随机投影并将其进行组合
  - 一个投影只是位字符串的随机子集。我们选择 *l* 个不同的随机投影并创建 *l* 个哈希表格, $g_1(\mathbf{x}),\ldots, g_\ell(\mathbf{x})$;
  - 将所有样例输入到每个哈希表中;
  - 给定查询点$\mathbf{x}_q$，在每个哈希表箱子$g_i(\mathbf{x}_q)$ 中获取点集;
  - 将这些集合取并集作为一组候选点C;
  - 然后计算 $\mathbf{x}_q$到C中每一个点的实际距离，并返回其中k个距离最近的点。

非参数模型

南京大学 NANJING UNIVERSITY

支持向量机

相比深度神经网络，**支持向量机（SVM）**仍拥有3个具有吸引力的特性:

- *SVMs*构造了一个极大边距分离器
  - 即到样例点距离最大的决策边界
- *SVMs*构造了一个线性分离超平面
- *SVMs*是非参数的
  - 其分离超平面是由一组样例点而不是参数值定义的

*SVM*并不是最小化训练数据上的期望经验损失，而是最小化期望**泛化**损失。

非参数模型

南京大学 NANJING UNIVERSITY

支持向量机

- 将分离器定义为点集$\{\mathbf{x} : \mathbf{w} \cdot \mathbf{x} + b = 0\}$.
- 通过求解对偶问题，可找到原问题的最优解

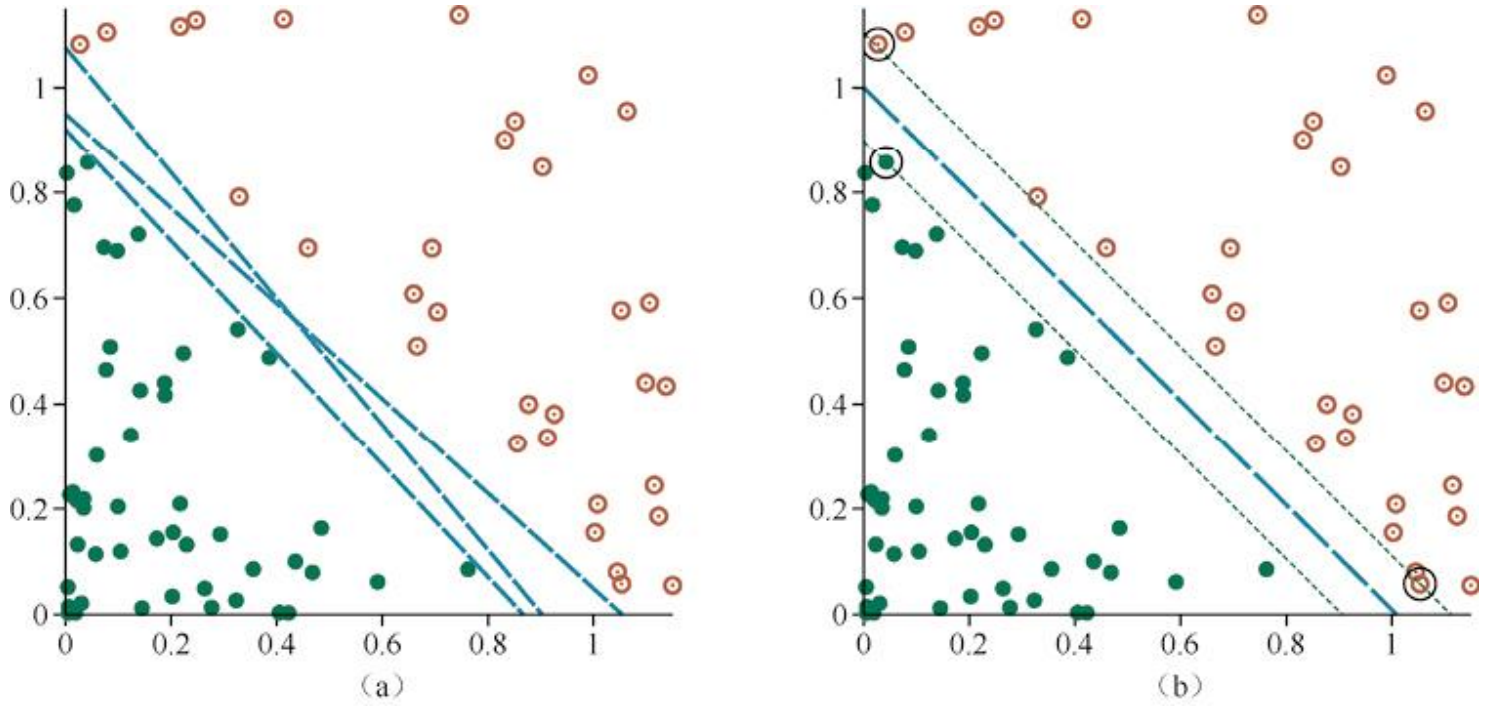
$$\underset{\alpha}{\operatorname{argmax}} \sum_j \alpha_j - \frac{1}{2} \sum_{j,k} \alpha_j \alpha_k y_j y_k (\mathbf{x}_j \cdot \mathbf{x}_k)$$

- 表达式是凸的；因此它有一个可以被有效方法找到的全局最大值。
- 数据仅以成对数据的点积形式输入表达式
- 最后一个重要的性质是，除支持向量（support vector，即最靠近分离器的点）外，每个数据点对应的权重α_j为0。

非参数模型

南京大学 NANJING UNIVERSITY

支持向量机



支持向量机分类器：（a）两类点（橙色空心圆和绿色实心圆）和3个候选线性分离器。（b）极大边距分离器（粗线）位于**边距**（虚线之间的区域）的中间。**支持向量**（带有大黑圈的点）是最靠近分离器的，本图中有3个

核技巧

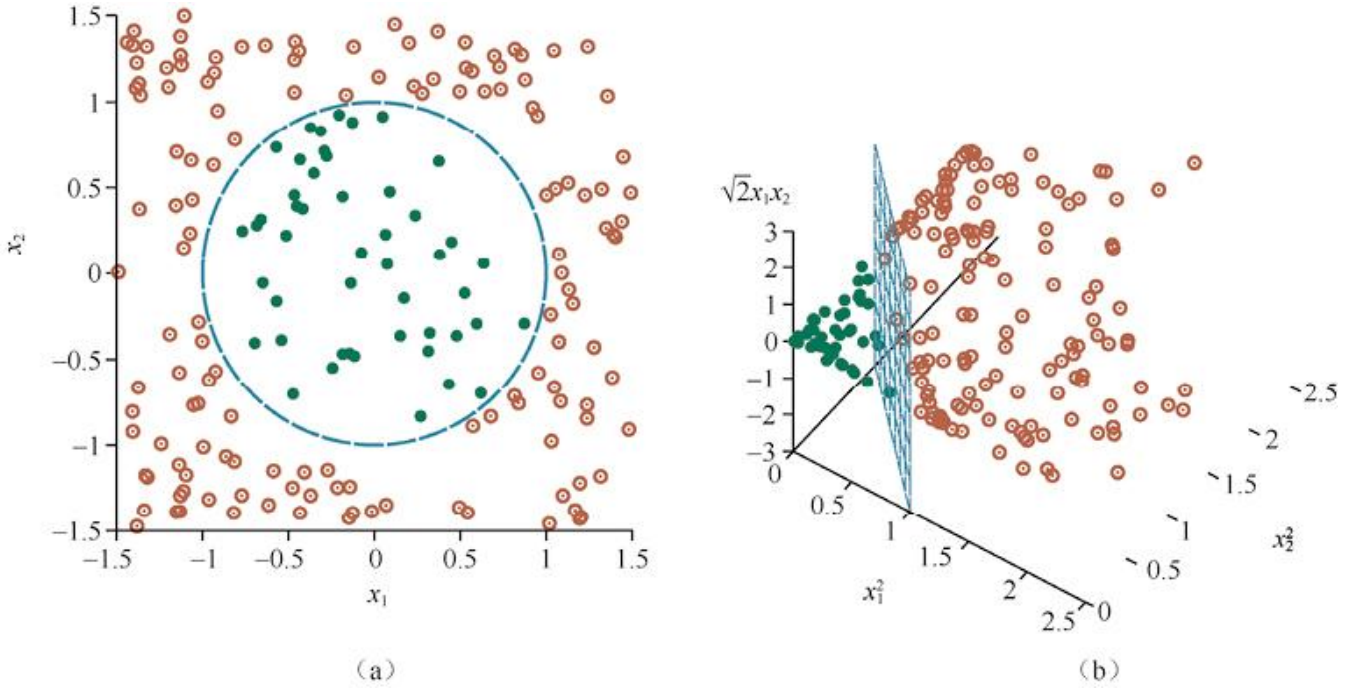


图19-22 (a) 一个二维训练集，其中正样例为绿色实心圆，负样例为橙色空心圆。图中还给出了真实的决策边界$x_1^2 + x_2^2 \leqslant 1$。(b) 映射到三维输入空间$(x_1^2, x_2^2, \sqrt{2}x_1x_2)$后的相同数据。(a) 中的圆形决策边界变成三维空间中的线性决策边界。图19-21b给出了 (b) 中分离器的特写

非参数模型

南京大学 NANJING UNIVERSITY

核技巧

- 将核函数带入下式，可以找到最优线性分类器

$$\underset{\alpha}{\operatorname{argmax}} \sum_j \alpha_j - \frac{1}{2} \sum_{j,k} \alpha_j \alpha_k y_j y_k (\mathbf{x}_j \cdot \mathbf{x}_k)$$

- 将生成的线性分离器映射回原始输入空间后，可以对应于任意一个在正样例和负样例之间的波动的非线性决策边界。

- 核方法不仅可以用于寻找最优线性分离器的学习算法，而且可以应用于任何其他可描述为仅用到数据对点积的算法。在将算法描述为点积后，将点积替换为核函数，我们就有了该算法的核化（kernelization）算法。

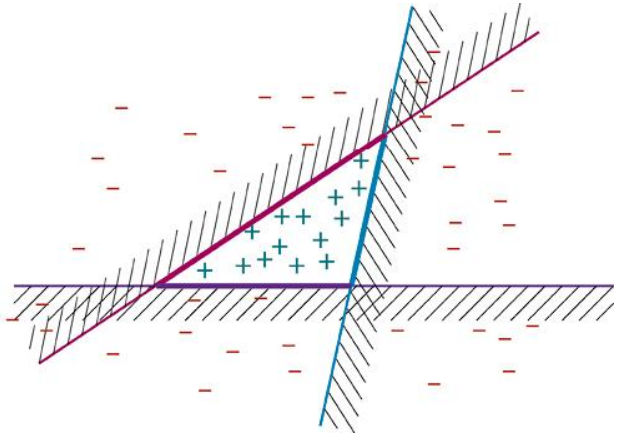
集成学习

- **集成学习（ensemble learning）**选择一个由一系列假设 $h_1, h_2, \ldots,$ h_n构成的集合，通过平均、投票或其他形式的机器学习方法将它们的预测进行组合。

- 单独的假设: **基模型**

- 组合后的模型: **集成模型**

- 集成学习的原因：
  - **减少偏差**，集成模型有更强大的表达能力，因此偏差会较小；
  - **减少方差**，多分类器的误分类率要更小。



通过集成学习的方法提升表达能力。我们使用3个线性阈值假设，每个假设在不带阴影的一面都是正分类，将所有被这3个面同时分类为正的样例分类为正。

自助聚合法

- 通过采样并替换原始训练集来生成K个不同的训练集；
- 从训练集中随机选取N个样例，这些样本中的每一个都可能是我们之前选择过的一个样例；
- 在这N个样例上运行机器学习算法，并获得一个假设；
- 重复这个过程K次即得到K个不同的假设；
- 汇总来自所有K个假设的预测；
- 对于分类问题，这意味着进行相对多数投票（对于二分类，为绝对多数投票）；
- 对于回归问题，最终输出为平均值：

$$h(\mathbf{x}) = \frac{1}{K}\sum_{i=1}^{K} h_i(\mathbf{x})$$

随机森林法

- 决策树自助聚合法的一种形式。
- 随机化属性选择，而不是训练样例。
- 在构造树的过程中，对于每个分割点，我们从属性中随机抽样，并计算哪个属性有最高的信息增益。
- **极端随机树(*ExtraTrees*):**
  - 对于每个选定的属性，从属性值域内随机均匀地采样几个候选值；
  - 选择其中具有最高信息增益的值。

堆叠法

- 对在相同数据上使用不同模型类训练的多个基模型进行组合。

- 方法步骤:
  - 使用同样的训练数据来训练每一个基模型；
  - 接下来考虑数据的验证集，对于每一行数据使用从基模型得出的预测进行增广，使用此验证集来训练新的集成模型；
  - 如果需要，也可以使用交叉验证。

$\boldsymbol{x}_1$ = Yes, No, No, Yes, Some, $$$, No, Yes, French, 0 ~ 10; y_1 = Yes

$\boldsymbol{x}_2$ = Yes, No, No, Yes, Full, $, No, No, Thai, 30 ~ 60, **Yes**, **No**, **No**; y_2 = No

- 可以将其视为由基模型构成第一层，并在基模型层的输出上进行操作进而建立的集成模型。

自适应提升法

- **加权训练集：**给每个样例赋予一个权重$w_j \geq 0$，该权重描述了样例在训练过程中应计数的次数。

- 从所有样例具有相等的权重开始，根据该训练集，训练第一个假设h_1.

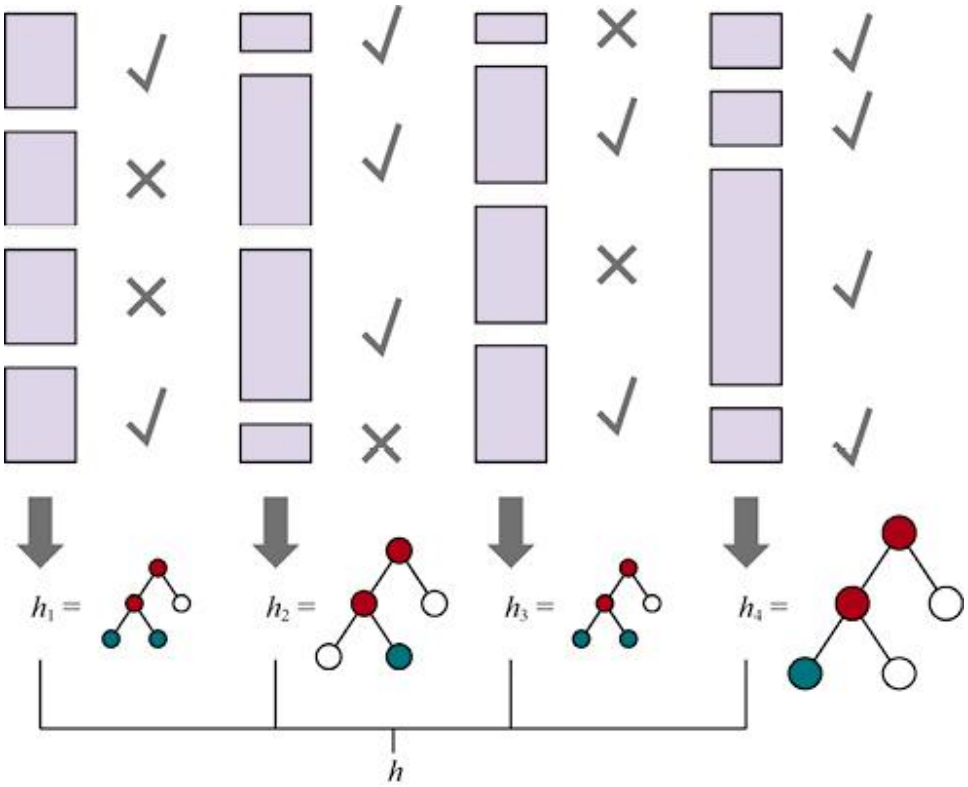
- 我们希望下一个假设能在被分类错误的样例中表现得更好，因此我们将增加它们的权重，同时减小正确分类的样例的权重。

- 基于这个重新进行加权得到的训练集，我们训练得到假设h_2。这一过程将以这种方式不断进行，直到生成K个假设。

- 类似于贪心算法，即不会回退，一旦算法选择了某个假设h_i，它就永远不会抛弃该选择，而是会添加新的假设：

$$h(\mathbf{x}) = \sum_{i=1}^{K} z_i h_i(\mathbf{x})$$

自适应提升法



自适应提升算法的运作方式。每个阴影矩形对应一个样例。矩形的高度对应于样例的权重。勾和叉号表示该样例是否在当前假设下正确分类。决策树的大小代表该假设在最终集成模型中的权重大小。

自适应提升法

function AdaBoost(*examples*, *L*, *K*) **returns** 一个假设

inputs: *examples*, 由N个带标签的样例$(x_1, y_1), \cdots, (x_N, y_N)$组成的集合

L, 学习算法

K, 集成中的假设个数

local variables: $\boldsymbol{w}$, 代表样例权重的N维向量，初始为全$1/N$

$\boldsymbol{h}$, 代表K个假设的向量

$\boldsymbol{z}$, 代表K个假设权重的向量

$\varepsilon \leftarrow$ 一个小的正数，用于规避除以零的情况

for $k = 1$ **to** K **do**

  $\boldsymbol{h}[k] \leftarrow L(examples, \boldsymbol{w})$

  $error \leftarrow 0$

  for $j = 1$ **to** N **do** // 计算 $\boldsymbol{h}[k]$的总错误率

    if $\boldsymbol{h}[k](x_j) \neq y_j$ **then** $error \leftarrow error + \boldsymbol{w}[j]$

  if $error > 1/2$ **then break** from loop

  $error \leftarrow \min(error, 1 - \varepsilon)$

  for $j = 1$ **to** N **do** // 赋予$\boldsymbol{h}[k]$出错的样例更大的权重

    if $\boldsymbol{h}[k](x_j) = y_j$ **then** $\boldsymbol{w}[j] \leftarrow \boldsymbol{w}[j] \cdot error/(1 - error)$

  $\boldsymbol{w} \leftarrow$ Normalize($\boldsymbol{w}$)

  $\boldsymbol{z}[k] \leftarrow \frac{1}{2}\log((1 - error)/error)$ // 赋予正确的$\boldsymbol{h}[k]$更大的权重

return $Function(x)$: $\sum \boldsymbol{z}_i \boldsymbol{h}_i(x)$

自适应提升法

弱学习算法: L 总是在训练集上返回准确度略高于随机猜想的假设

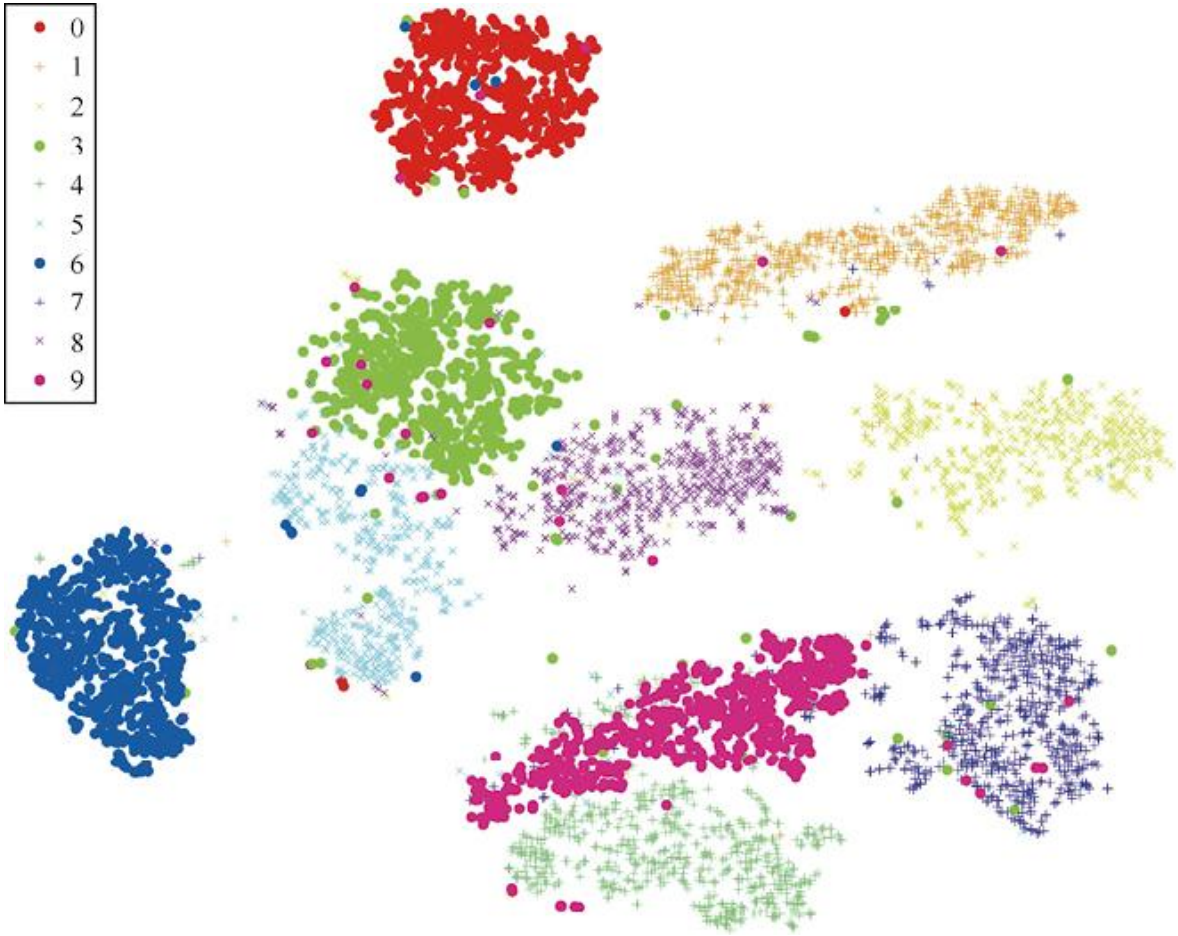
如果输入的学习算法 L是弱学习算法，ADABOOST

- 返回一个假设，对于足够大的K，该假设能对训练数据进行完美的分类；
- 提升了原始学习算法对训练数据预测的准确性；
- 只要基模型比随机猜测更好，我们就可以通过自适应提升模型中的任何大小的偏差。

开发机器学习系统

- **问题形式化**
  - 确定要解决的问题，制定一个损失函数；
  - 确定是在处理监督学习、无监督学习还是强化学习的问题。

- **数据收集、评估和管理**
  - 当数据有限时，数据增强的方法会有所助益；
  - 对于不均衡类，可以对多数类进行欠采样或对少数类过采样。也可以使用一个加权损失函数，对误判一个少数类样例进行更大的惩罚；
  - 考虑数据中的离群值；
  - 特征工程，数据分析和可视化



MNIST数据集上的二维*t*-SNE平面图，该数据集收集了60 000幅手写数字图像，每幅28像素×28像素，因此是784维。

开发机器学习系统

- **模型选择与训练**
  - 选择一个模型类（如随机森林模型、深度神经网络模型或集成模型），使用训练数据训练模型；
  - 利用验证数据来调优模型类的所有超参数；
  - 用测试数据对模型进行评估。

- **信任和可解释性**
  - 源码控制、测试、审查、监控、问责；
  - 如果能观察实际模型并理解为什么它会在得到给定输入后给出的特定输出，以及当输入发生变化时输出将如何变化，我们就称此机器学习模型是可解释的。

- **操作，监控和维护**
  - 监控模型在实时数据上的表现；
  - 非平稳性问题——外部世界会随时间变化。

小结

- 学习有多种形式，这取决于智能体的形式、可以进行改进的组件和可获得的反馈。
- 如果可用的反馈提供了每个样例输入的正确答案，那么我们称该学习问题为**监督学习**。当学习一个输出为连续或有序值的函数时，我们称之为**回归**；当学习一个具有少量可能输出类别的函数时，我们称之为**分类**。
- 我们希望学习到一个函数，它不仅与现有的数据保持一致，而且与未来出现的数据也很可能一致。我们需要在与数据的一致性和假设的简单性之间做出权衡。
- **决策树**可以表示任何一个布尔函数。基于信息增益的启发式算法为寻找一个简单且一致的决策树提供了一个有效途径。
- 一个学习算法的表现可以通过**学习曲线**进行可视化，它所给出的是模型在测试集上的预测准确率关于训练集大小的函数关系。
- 当有多个模型可供选择时，**模型选择**可以通过在验证集上进行**交叉验证**来选出较好的超参数。
- **损失函数**可以告诉我们一个错误的严重程度，那么我们的目标则是最小化验证集上的损失函数。

小结

- **线性回归**是一个被广泛运用的模型，它的最优参数值可以被精确地计算出来或者通过梯度下降搜索找到，其中梯度下降搜索是一种可以用于解决不存在闭式解的模型的技术。
- 一个带有硬阈值的线性分类器，也被称为**感知机**，可以通过简单的参数更新规则训练，并且能拟合线性可分的数据。对于线性不可分的数据，这个更新规则将不能收敛。
- **逻辑斯谛回归**将感知机所用的硬阈值替换为由逻辑斯谛函数定义的软阈值。即使是带噪声的线性不可分数据集，梯度下降也能在该模型中表现得很好。
- **非参数模型**使用所有的数据来单独做每一个预测，而不是试图用若干个参数总结出数据中的信息。例如**最近邻**与**局部加权回归**。
- **支持向量机**通过寻找带有**最大边距**的线性分离器来改进分类的泛化表现。即使原数据不是线性可分的，**核方法**也可以隐式地将输入数据投影到可能存在线性分离器的高维空间中。
- **自助聚合法**与**自适应提升法**这样的集成方法通常比单独的方法表现得更好。
- 构建一个良好的机器学习模型要求我们在从数据管理到模型选择和优化，再到持续性维护的整个开发过程中都有一定经验。