机器学习

第6章 贝叶斯学习

概述

- 贝叶斯推理提供了一种概率手段，基于如下的假定：待考察的量遵循某概率分布，且可根据这些概率及已观察到的数据进行推理，以作出最优的决策。
- 贝叶斯推理为衡量多个假设的置信度提供了定量的方法
- 贝叶斯推理为直接操作概率的学习算法提供了基础，也为其他算法的分析提供了理论框架

简介

- 贝叶斯学习算法与机器学习相关的两个原因：
  - 贝叶斯学习算法能够计算显示的假设概率，比如朴素贝叶斯分类
  - 贝叶斯方法为理解多数学习算法提供了一种有效的手段，而这些算法不一定直接操纵概率数据，比如
    - Find-S
    - 候选消除算法
    - 神经网络学习：选择使误差平方和最小化的神经网络
    - 推导出另一种误差函数：交叉熵
    - 分析了决策树的归纳偏置
    - 考察了最小描述长度原则

贝叶斯学习方法的特性

- 观察到的每个训练样例可以增量地降低或升高某假设的估计概率。而其他算法会在某个假设与任一样例不一致时完全去掉该假设
- 先验知识可以与观察数据一起决定假设的最终概率，先验知识的形式是：1）每个候选假设的先验概率；2）每个可能假设在可观察数据上的概率分布
- 贝叶斯方法可允许假设做出不确定性的预测
- 新的实例分类可由多个假设一起做出预测，用它们的概率来加权
- 即使在贝叶斯方法计算复杂度较高时，它们仍可作为一个最优的决策标准衡量其他方法

贝叶斯方法的难度

- 难度之一：需要概率的初始知识，当概率预先未知时，可以基于背景知识、预先准备好的数据以及基准分布的假定来估计这些概率
- 难度之二：一般情况下，确定贝叶斯最优假设的计算代价比较大（在某些特定情形下，这种计算代价可以大大降低）。

内容安排

- 介绍贝叶斯理论
- 定义极大似然假设和极大后验概率假设
- 将此概率框架应用于分析前面章节的相关问题和学习算法
- 介绍几种直接操作概率的学习算法
  - 贝叶斯最优分类器
  - Gibbs算法
  - 朴素贝叶斯分类器
- 讨论贝叶斯信念网，这是存在未知变量时被广泛使用的学习算法

贝叶斯法则

- 机器学习的任务：在给定训练数据D时，确定假设空间H中的最佳假设。
- 最佳假设：一种方法是把它定义为在给定数据D以及H中不同假设的先验概率的有关知识下的最可能假设
- 贝叶斯理论提供了一种计算假设概率的方法，基于假设的先验概率、给定假设下观察到不同数据的概率以及观察到的数据本身

先验概率和后验概率

- 用P(h)表示在没有训练数据前假设h拥有的初始概率。P(h)被称为h的先验概率。
- 先验概率反映了关于h是一正确假设的机会的背景知识
- 如果没有这一先验知识，可以简单地将每一候选假设赋予相同的先验概率
- 类似地，P(D)表示训练数据D的先验概率，P(D|h)表示假设h成立时D的概率
- 机器学习中，我们关心的是P(h|D)，即给定D时h的成立的概率，称为h的后验概率

贝叶斯公式

- 贝叶斯公式提供了从先验概率P(h)、P(D)和P(D|h)计算后验概率P(h|D)的方法

$$P(h \mid D) = \frac{P(D \mid h)P(h)}{P(D)}$$

- P(h|D)随着P(h)和P(D|h)的增长而增长，随着P(D)的增长而减少，即如果D独立于h时被观察到的可能性越大，那么D对h的支持度越小

极大后验假设

- 学习器在候选假设集合H中寻找给定数据D时可能性最大的假设h，h被称为极大后验假设（MAP）
- 确定MAP的方法是用贝叶斯公式计算每个候选假设的后验概率，计算式如下

$$h_{MAP} = \arg\max_{h \in H} P(h \mid D) = \arg\max_{h \in H} \frac{P(D \mid h)P(h)}{P(D)} = \arg\max_{h \in H} P(D \mid h)P(h)$$

最后一步，去掉了P(D)，因为它是不依赖于h的常量

极大似然假设

- 在某些情况下，可假定H中每个假设有相同的先验概率，这样式子6.2可以进一步简化，只需考虑P(D|h)来寻找极大可能假设。
- P(D|h)常被称为给定h时数据D的似然度，而使P(D|h)最大的假设被称为极大似然假设

$$h_{ML} = \arg\max_{h \in H} P(D \mid h)$$

- 假设空间H可扩展为任意的互斥命题集合，只要这些命题的概率之和为1

举例：一个医疗诊断问题

- 有两个可选的假设：病人有癌症、病人无癌症
- 可用数据来自化验结果：正+和负-
- 有先验知识：在所有人口中，患病率是0.008
- 对确实有病的患者的化验准确率为98%，对确实无病的患者的化验准确率为97%
- 总结如下

  $P(cancer)=0.008, P(\neg cancer)=0.992$

  $P(+|cancer)=0.98, P(-|cancer)=0.02$

  $P(+|\neg cancer)=0.03, P(-|\neg cancer)=0.97$

举例：一个医疗诊断问题（2）

- 问题：假定有一个新病人，化验结果为正，是否应将病人断定为有癌症？求后验概率P(cancer|+)和P(¬cancer|+)
- 利用式子6.2找到极大后验假设
  - P(+|cancer)P(cancer)=0.0078
  - P(+|¬cancer)P(¬cancer)=0.0298
  - h_{MAP}=¬cancer
- 确切的后验概率可将上面的结果归一化以使它们的和为1
  - P(canner|+)=0.0078/(0.0078+0.0298)=0.21
  - P(¬cancer|-)=0.79
- 贝叶斯推理的结果很大程度上依赖于先验概率，另外不是完全接受或拒绝假设，只是在观察到较多的数据后增大或减小了假设的可能性

基本概率公式表

- 乘法规则：
  $P(A \wedge B)=P(A|B)P(B)=P(B|A)P(A)$
- 加法规则：$P(A \vee B)=P(A)+P(B)-P(A \wedge B)$
- 贝叶斯法则：$P(h|D)=P(D|h)P(h)/P(D)$
- 全概率法则：如果事件$A_1 \ldots A_n$互斥，且满足$\sum_{i=1}^{n} P(A_i)=1$，则 $P(B)=\sum_{i=1}^{n} P(B|A_i)P(A_i)$

贝叶斯法则和概念学习

- 贝叶斯法则为计算给定训练数据下任一假设的后验概率提供了原则性方法，因此可以直接将其作为一个基本的学习方法：计算每个假设的概率，再输出其中概率最大的。这个方法称为Brute-Force贝叶斯概念学习算法。
- 将上面方法与第2章介绍的概念学习算法比较，可以看到：在特定条件下，它们学习得到相同的假设，不同的是第2章的方法不明确计算概率，而且效率更高。

Brute-Force贝叶斯概念学习

- 概念学习问题：有限假设空间H定义在实例空间X上，任务是学习某个目标概念c。
- Brute-Force MAP学习算法
  - 对于H中每个假设h，计算后验概率 $P(h|D)=\frac{P(D|h)P(h)}{P(D)}$
  - 输出有最高后验概率的假设 $h_{MAP}=\arg\max_{h\in H}P(h|D)$
- 上面算法需要较大计算量，因为它要计算每个假设的后验概率，对于大的假设空间显得不切实际，但是它提供了一个标准以判断其他概念学习算法的性能

特定情况下的MAP假设

- 假定
  - 训练数据D是无噪声的，即$d_i=c(x_i)$
  - 目标概念c包含在假设空间H中
  - 每个假设的概率相同
- 求得
  - 由于所有假设的概率之和是1，因此 $P(h)=\frac{1}{|H|}$
  - 由于训练数据无噪声，那么给定假设h时，与h一致的D的概率为1，不一致的概率为0，因此

$$P(D|h)=\begin{cases}1 & \forall d_i, d_i=h(x_i)\\ 0 & otherwise\end{cases}$$

特定情况下的MAP假设（2）

- 考虑Brute-Force MAP算法的第一步
  - h与D不一致，$P(h \mid D) = \frac{0 \cdot P(h)}{P(D)} = 0$
  - h与D一致，$P(h \mid D) = \frac{1 \cdot \frac{1}{|H|}}{P(D)} = \frac{\frac{1}{|H|}}{\frac{|VS_{H,D}|}{|H|}} = \frac{1}{|VS_{H,D}|}$，

    $VS_{H,D}$是关于D的变型空间（见第2章，即与D一致的假设集）

特定情况下的MAP假设（3）

- P(D)的推导

$$
\begin{aligned}
P(D) &= \sum_{h_i \in H} P(D \mid h_i) P(h_i) \\
&= \sum_{h_i \in VS_{H,D}} 1 \times \frac{1}{|H|} + \sum_{h_i \notin VS_{H,D}} 0 \times \frac{1}{|H|} \\
&= \sum_{h_i \in VS_{H,D}} 1 \times \frac{1}{|H|} \\
&= \frac{|VS_{H,D}|}{|H|}
\end{aligned}
$$

- 假设的概率演化情况如图6-1所示，初始时所有假设具有相同的概率，当训练数据逐步出现后，不一致假设的概率变为0，而整个概率的和为1，它们均匀分布到剩余的一致假设中
- 每个一致的假设都是MAP假设

MAP假设和一致学习器

- 一致学习器：如果某个学习器输出的假设在训练样例上为0错误率，则称为一致学习器
- 如果H上有均匀的先验概率，且训练数据是确定性和无噪声的，任意一致学习器将输出一个MAP假设
- Find-S算法按照特殊到一般的顺序搜索架设空间H，并输出一个极大特殊的一致假设，因此可知在上面定义的P(h)和P(D|h)概率分布下，它输出MAP假设
- 更一般地，对于先验概率偏袒于更特殊假设的任何概率分布，Find-S输出的假设都是MAP假设

MAP假设和一致学习器（2）

- 贝叶斯框架提出了一种刻画学习算法行为的方法，即便该学习算法不进行概率操作，通过确定算法输出最优假设时使用的概率分布P(h)和P(D|h)，可以刻画出算法具有最优行为时的隐含假定
- 使用贝叶斯方法刻画学习算法，与揭示学习器中的归纳偏置在思想上是类似的
- 在第2章，将学习算法的归纳偏置定义为断言集合B，通过它可充分地演绎推断出学习器所执行的归纳推理结果，即学习器的输出是由其输入和隐含的归纳偏置所演绎得出的

MAP假设和一致学习器（3）

- 贝叶斯解释对于描述学习算法中的隐含假定提供了另一种方法，用基于贝叶斯理论的一个等效的概率推理系统来建模
- 贝叶斯解释隐含的假定形式为：H上的先验概率由P(h)分布给出，数据拒绝或接受假设的强度由P(D|h)给出
- 在已知这些假定的概率分布后，一个基于贝叶斯理论的概率推理系统将产生等效于Find-S、候选消除等算法的输入-输出行为

极大似然和最小误差平方假设

- 前面分析表明：某些学习算法即使没有显示地使用贝叶斯规则，或以某种形式计算概率，但它们输出的结果符合贝叶斯原理，是一个MAP假设
- 通过简单的贝叶斯分析，可以表明在特定前提下，任一学习算法如果使输出的假设预测和训练数据之间的误差平方和最小化，它将输出一极大似然假设
- 上面结论的意义是，对于许多神经网络和曲线拟合的方法，如果它们试图在训练数据上使误差平方和最小化，此结论提供了基于贝叶斯的理论依据

极大似然和最小误差平方假设（2）

- 问题框架：
  - 学习器L工作在实例空间X和假设空间H上，H中的假设为X上定义的某种实数值函数。
  - L面临的问题是学习一个从H中抽取出的未知目标函数f，给定m个训练样例的集合，每个样例的目标值被某随机噪声干扰，此随机噪声服从正态分布
  - 更精确地讲，每个训练样例是序偶$<x_i,d_i>$，$d_i=f(x_i)+e_i$，e_i是代表噪声的随机变量，假定e_i的值是独立抽取的，并且它们的分布服从0均值的正态分布
  - 学习器的任务是在所有假设有相等的先验概率前提下，输出极大似然假设（即MAP假设）

极大似然和最小误差平方假设（3）

- 用一个简单情况，即线性函数来说明问题。如图6-2所示，实线表示线性目标函数f，实点表示有噪声的训练样例集，虚线对应有最小平方训练误差的假设h_{ML}，即极大似然假设。
- 对于e这样的连续变量上的概率，使用概率密度表示概率分布，它在所有值上的积分为1，用小写的p表示。有限概率P有时又称为概率质量
- 概率密度函数：$p(x_0) = \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} P(x_0 \le x < x_0 + \varepsilon)$

极大似然和最小误差平方假设（4）

- 假定有一固定的训练实例集合，因此只考虑相应的目标值序列$D=<d_1...d_m>$，这里$d_i=f(x_i)+e_i$。
- 假定训练样例是相互独立的，给定h时，可将$P(D|h)$写成各$p(d_i|h)$的积

$$h_{ML} = \arg\max_{h \in H} \prod_{i=1}^{m} p(d_i \mid h)$$

- 如果误差e_i服从0均值和未知方差σ^2的正态分布，那么每个d_i服从均值为$f(x_i)$，方差不变的正态分布。因此，$p(d_i|h)$可写为方差σ^2、均值$f(x_i)$的正态分布
- 使用表5-4中的正态分布公式并将相应的参数代入，由于概率d_i的表达式是在h为目标函数f的正确描述条件下的，所以替换$\mu=f(x_i)=h(x_i)$

极大似然和最小误差平方假设（5）

- h_{ML}

$$
\begin{aligned}
&= \arg\max_{h\in H} \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(d_i-\mu)^2} \\
&= \arg\max_{h\in H} \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(d_i-h(x_i))^2} \\
&= \arg\max_{h\in H} \sum_{i=1}^{m} \ln\frac{1}{\sqrt{2\pi\sigma^2}} - -\frac{1}{2\sigma^2}(d_i-h(x_i))^2 \\
&= \arg\max_{h\in H} \sum_{i=1}^{m} -\frac{1}{2\sigma^2}(d_i-h(x_i))^2 \\
&= \arg\min_{h\in H} \sum_{i=1}^{m} (d_i-h(x_i))^2
\end{aligned}
$$

- 上式说明了极大似然假设等价于使训练值和假设预测值之间的误差的平方和最小的那个假设
- 这个结论的前提是：训练值等于真实目标值加上随机噪声，其中随机噪声从一个均值为0的正态分布中独立抽取

采用正态分布的合理性

- 数学计算的简洁性
- 对许多物理系统的噪声都有良好的近似
- 第5章中心极限定律显示，足够多的独立同分布随机变量的和服从正态分布
- 由许多独立同分布的因素的和所生成的噪声将成为正态分布（当然，现实中不同的分量对噪声的贡献也许不是同分布的）
- 使误差平方最小化的方法经常被用于神经网络、曲线拟合及其他许多实函数逼近的算法中
- 上面的分析只考虑了训练样例的目标值中的噪声，而没有考虑实例属性值的噪声

用于预测概率的极大似然假设

- 问题框架：
  - 学习一个不确定性函数f: X→{0,1}，它有两个离散的值输出
  - 这种不可预测性来源于未能观察到的因素，导致目标函数的输出是输入的概率函数
- 学习得到的神经网络（或其他实函数学习器）的输出是f(x)=1的概率，表示为f’: X→[0,1]，即f’=P(f(x)=1)

用于预测概率的极大似然假设（2）

- Brute-Force法
  - 首先收集对x的每个可能值观察到的1和0的频率，然后训练神经网络，对每个x输出目标频率
- 可以直接从f的训练样例中训练神经网络，然后推导出f’的极大似然假设
  - $D=\{<x_1,d_1>...<x_m,d_m>\}$
  - $P(D|h)=\prod_{i=1}^{m}P(x_i,d_i|h)=\prod_{i=1}^{m}P(d_i|h,x_i)P(x_i)$

用于预测概率的极大似然假设（3）

— $P(d_i \mid h, x_i) = \begin{cases} h(x_i) & d_i = 1 \\ 1 - h(x_i) & d_i = 0 \end{cases} = h(x_i)^{d_i}(1 - h(x_i))^{1-d_i}$

— $P(D \mid h) = \prod_{i=1}^{m} h(x_i)^{d_i}(1 - h(x_i))^{1-d_i} P(x_i)$

– $h_{ML} = \arg\max_{h \in H} \prod_{i=1}^{m} h(x_i)^{d_i}(1 - h(x_i))^{1-d_i} p(x_i)$

$$= \arg\max_{h \in H} \prod_{i=1}^{m} h(x_i)^{d_i}(1 - h(x_i))^{1-d_i}$$

$$= \arg\max_{h \in H} \sum_{i=1}^{m} d_i \ln h(x_i) + (1 - d_i)\ln(1 - h(x_i))$$

– 式子6.13与熵函数的一般式相似，因此它的负值常称为交叉熵

在神经网络中梯度搜索以达到似然最大化

- 前面讨论了利用式子6.13求极大似然假设，现用G(h,D)表示，为神经网络学习推导一个权值训练法则，使用梯度上升法使G(h,D)最大化

$$\frac{\partial G(h,D)}{\partial w_{jk}} = \sum_{i=1}^{m} \frac{\partial G(h,D)}{\partial h(x_i)} \frac{\partial h(x_i)}{\partial w_{jk}}$$

$$= \sum_{i=1}^{m} \frac{\partial (d_i \ln h(x_i) + (1-d_i)\ln(1-h(x_i)))}{\partial h(x_i)} \frac{\partial h(x_i)}{\partial w_{jk}}$$

$$= \sum_{i=1}^{m} \frac{d_i - h(x_i)}{h(x_i)(1-h(x_i))} \frac{\partial h(x_i)}{\partial w_{jk}}$$

- 考虑简单的情况，假定神经网络从一个单层的sigmoid单元建立，则

$$\frac{\partial h(x_i)}{\partial w_{jk}} = \sigma'(x_i) x_{ijk} = h(x_i)(1-h(x_i)) x_{ijk}$$

在神经网络中梯度搜索以达到似然最大化（2）

$$\frac{\partial G(h,D)}{\partial w_{jk}} = \sum_{i=1}^{m}(d_i - h(x_i))x_{ijk}$$

- 因为要使P(D|h)最大化而不是最小化，因此执行梯度上升搜索，而不是梯度下降搜索。

$$w_{jk} \leftarrow w_{jk} + \Delta w_{jk} \qquad \Delta w_{jk} = \eta \sum_{i=1}^{m}(d_i - h(x_i))x_{ijk}$$

- 与反向传播更新法则对比
  - 使误差平方最小化的法则寻找到极大似然假设的前提是：训练数据可以由目标函数值加上正态分布噪声来模拟
  - 使交叉熵最小化的法则寻找极大似然假设基于的前提是：观察到的布尔值为输入实例的概率函数

最小描述长度准则

- 奥坎姆剃刀可以概括为：为观察到的数据选择最短的解释
- 此处给出一个贝叶斯分析，提出最小描述长度准则，根据信息论中的基本概念来解释h_{MAP}的定义

$$
\begin{aligned}
h_{MAP} &= \arg\max_{h \in H} P(D \mid h) P(h) \\
&= \arg\max_{h \in H} \log_2 P(D \mid h) + \log_2 P(h) \\
&= \arg\min_{h \in H} -\log_2 P(D \mid h) - \log_2 P(h)
\end{aligned}
$$

- 上式可以解释为在特定的假设编码表示方案上“优先选择短的假设”

最小描述长度准则（2）

- 信息论中的编码理论
  - 设想要为随机传送的消息设计一个编码，其中遇到消息i的概率是p_i
  - 感兴趣的是，使得传输随机信息所需的最小期望传送位数的编码
  - 直观上，为使期望的编码长度最小，可能性大的消息应该赋予较短的编码
  - Shannon & Weaver证明了最优编码对消息i的编码长度为$-\log_2 p_i$
  - 使用代码C来编码消息i所需的位数被称为消息i关于C的描述长度，记为$L_C(i)$

最小描述长度准则（3）

- 使用编码理论的结论来解释等式6.16
  - $-\log_2 P(h)$是在假设空间H的最优编码下h的描述长度。换言之，这是假设h使用其最优表示时的大小
  - $L_{C_H}(h) = -\log_2 P(h)$，$C_H$为假设空间H的最优编码
  - $-\log_2 P(D|h)$是在给定假设h时，训练数据D的描述长度，$L_{C_{D|h}}(D|h) = -\log_2 P(D|h)$，$C_{D|h}$是假定发送者和接送者都知道假设h时描述数据D的最优编码
  - 因此式子6.16显示，h_{MAP}是使假设描述长度和给定假设下数据描述长度之和最小化的假设
- 最小描述长度准则：$h_{MDL} = \arg\min_{h \in H} L_{C_1}(h) + L_{C_2}(D|h)$

最小描述长度准则（4）

- 如果选择C_1为假设的最优编码C_H，C_2为最优编码$C_{D|h}$，那么$h_{MDL}=h_{MAP}$
- 可将MDL准则想象为选择最短的方法来重新编码训练数据，其中不仅计算假设的大小，并且计算给定假设时编码数据的附加开销
- 将MDL准则应用于决策树，如何选择假设和数据的表示C_1和C_2?
  - 对于C_1，很自然地选择某种明确的决策树编码方法，其中描述长度随着树中节点和边的增长而增加
  - 对于C_2，如果训练分类$f(x_i)$与假设的预计相同，那么就不需要传输有关这些样例的任何信息；如果不同，则要传输更正消息

最小描述长度准则（5）

- MDL准则提供了一种方法在假设的复杂性和假设产生错误的数量之间进行折中，它有可能选择一个较短的产生少量错误的假设，而不是完美地分类训练数据的较长的假设
- 上面讨论自然给出了一种处理数据过度拟合的方法
- Quinlan & Rivest描述了应用MDL准则选择决策树大小的几个实验，报告指出，基于MDL的方法产生的决策树的精度相当于第3章中讨论的标准树修剪方法
- 第125页，6.6节最后一段的含义？

贝叶斯最优分类器

- 前面我们讨论的问题是：给定训练数据，最可能的假设是什么？
- 另一个相关的更有意义的问题是：给定训练数据，对新实例的最可能的分类是什么？
- 显然，第二个问题的解决可以将第一个问题的结果（MAP）应用到新实例上得到，还存在更好的算法

贝叶斯最优分类器（2）

- 例子
  - 考虑一个包含三个假设h_1, h_2, h_3的假设空间。
  - 假定已知训练数据时三个假设的后验概率分别是0.4, 0.3, 0.3，因此h_1为MAP假设。
  - 若一新实例x被h_1分类为正，被h_2和h_3分类为反
  - 计算所有假设，x为正例的概率为0.4，为反例的概率为0.6
  - 因此，这时最可能的分类与MAP假设生成的分类不同

贝叶斯最优分类器（3）

- 一般而言，新实例的最可能分类可通过合并所有假设的预测得到，用后验概率来加权。
- 如果新实例的可能分类可取某集合V中的任一值v_j，那么概率$P(v_j|D)$表示新实例分类为v_j的概率

$$P(v_j \mid D) = \sum_{h_i \in H} P(v_j \mid h_i) P(h_i \mid D)$$

- 新实例的最优分类为使$P(v_j|D)$最大的v_j值，贝叶斯最优分类器为：

$$\arg\max_{v_j \in V} \sum_{h_i \in H} P(v_j \mid h_i) P(h_i \mid D)$$

贝叶斯最优分类器（4）

- 例子
  - 已知：
    - 新实例的可能分类集合为V={+,-}
    - $P(h_1|D)=0.4$, $P(-|h_1)=0$, $P(+|h_1)=1$
    - $P(h_2|D)=0.3$, $P(-|h_2)=1$, $P(+|h_2)=0$
    - $P(h_3|D)=0.3$, $P(-|h_3)=1$, $P(+|h_2)=0$
  - 因此：
    - $\sum_{h_i \in H} P(+ \mid h_i) P(h_i \mid D) = 0.4$
    - $\sum_{h_i \in H} P(- \mid h_i) P(h_i \mid D) = 0.6$
    - $\arg\max_{v_j \in \{+,-\}, h_i \in H} \sum_{h_i \in H} P(v_j \mid h_i) P(h_i \mid D) = -$

贝叶斯最优分类器（5）

- 贝叶斯最优分类器在给定可用数据、假设空间及这些假设的先验概率下使新实例被正确分类的可能性达到最大
- 贝叶斯最优分类器的一个属性：它所做的分类可以对应于H中不存在的假设
- 使用式子6.18来分类X中的每个实例，按此定义的实例标注不一定对应于H中的任一单个假设h对实例的标注
- 将贝叶斯分类器看成是不同于假设空间H的另一空间H’，在其上应用贝叶斯公式。H’有效地包含了一组假设，它能在H中多个假设的线性组合所作的预言中进行比较

Gibbs算法

- 贝叶斯最优分类器能从给定训练数据中获得最好的性能，但算法的开销很大
- 一个替代的、非最优的方法是Gibbs算法，定义如下：
  - 按照H上的后验概率分布，从H中随机选择假设h
  - 使用h来预言下一个实例x的分类
- 在一定条件下，Gibbs算法的误分类率的期望值最多为贝叶斯最优分类器的两倍。确切地讲，期望值是在随机抽取的目标概念上作出的，抽取过程按照学习器假定的先验概率
- 对概念学习问题的一个启示：如果学习器假定H上有均匀的先验概率，而且如果目标概念实际上也按该分布抽取，那么当前变型空间中随机抽取的假设对下一实例分类的期望误差最多为贝叶斯分类器的两倍（？）

朴素贝叶斯分类器

- 应用的学习任务：每个实例x可由属性值的合取描述，而目标函数f(x)从某有限集合V中取值
- 贝叶斯方法的新实例分类目标是在给定描述实例的属性值<a_1,...,a_n>下，得到最可能的目标值V_{MAP}

$$v_{MAP} = \arg\max_{v_j} P(v_j \mid a_1,...,a_n)$$

- 使用贝叶斯公式变化上式

$$\begin{aligned} v_{MAP} &= \arg\max_{v_j \in V} \frac{P(a_1,...,a_n \mid v_j)P(v_j)}{P(a_1,...,a_n)} \\ &= \arg\max_{v_j \in V} P(a_1,...,a_n \mid v_j)P(v_j) \end{aligned}$$

朴素贝叶斯分类器（2）

- 基于训练数据估计式子6.19中的两个数据项的值
  - 估计$P(v_j)$很容易：计算每个目标值v_j出现在训练数据中的频率
  - 估计$P(a_1,...a_n|v_j)$遇到数据稀疏问题，除非有一个非常大的训练数据集，否则无法获得可靠的估计
- 朴素贝叶斯分类器引入一个简单的假定避免数据稀疏问题：在给定目标值时，属性值之间相互条件独立，即 $P(a_1,...,a_n \mid v_j) = \prod_i P(a_i \mid v_j)$

朴素贝叶斯分类器（3）

- 朴素贝叶斯分类器的定义：$v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_i P(a_i \mid v_j)$
- 从训练数据中估计不同$P(a_i|v_j)$项的数量比要估计$P(a_1,...,a_n|v_j)$项所需的量小得多
- 只要条件独立性得到满足，朴素贝叶斯分类v_{NB}等于MAP分类，否则是近似
- 朴素贝叶斯分类器与其他已介绍的学习方法的一个区别：没有明确地搜索可能假设空间的过程（假设的形成不需要搜索，只是简单地计算训练样例中不同数据组合的出现频率）

朴素贝叶斯分类器（4）

- 举例
  - 表3-2提供了目标概念PlayTennis的14个训练样例，给新实例<sunny, cool, high, strong>分类
  - $v_{NB} = \arg\max_{v_j \in \{yes,no\}} P(v_j) \prod_i P(a_i \mid v_j) = \arg\max_{v_j \in \{yes,no\}} P(v_j) P(sunny \mid v_j) P(cool \mid v_j) P(high \mid v_j) P(strong \mid v_j)$
  - 根据表3-2，可以计算出上式需要的概率值
    - P(yes)=9/14=0.64
    - P(no)=5/14=0.36
    - P(strong|yes)=3/9=0.33
    - P(strong|no)=3/5=0.60
    - ...
  - 求v_{NB}
    - P(yes)P(sunny|yes)P(cool|yes)P(high|yes)P(strong|yes)=0.0053
    - P(no)P(sunny|no)P(cool|no)P(high|no)P(strong|no)=0.0206
    - v_{NB}=no

朴素贝叶斯分类器（5）

- 估计概率
  - 我们通过在全部事件基础上观察某事件出现的比例来估计概率
  - 当样本很小时，采用平滑技术，m-估计

$$\frac{n_c + mp}{n + m}$$

  - p是将要确定的概率的先验估计，而m是一称为等效样本大小的常量
  - 在缺少其他信息时，选择p的一种典型的方法是均匀概率，比如某属性有k个可能值，那么p=1/k
  - m被称为等效样本大小的原因是：式子6.22可被解释为将n个实际的观察扩大，加上m个按p分布的虚拟样本

举例：学习分类文本

- 利用贝叶斯方法学习目标概念，然后用于文本自动过滤，比如
  - 我感兴趣的电子新闻稿
  - 讨论机器学习的万维网页
- 本节描述一个基于朴素贝叶斯分类器的文本分类的通用算法，它是目前所知的文本分类的最有效方法之一
- 问题框架：实例空间X包含了所有的文本文档，给定某未知目标函数f(x)的一组训练样例，f(x)的值来自某有限集合V（作为示例，此处令V={like, dislike}）

举例：学习分类文本（2）

- 应用朴素贝叶斯分类器的两个主要设计问题：
  - 怎样将任意文档表示为属性值的形式
  - 如何估计朴素贝叶斯分类器所需的概率
- 表示文档的方法
  - 给定一个文本文档，对每个单词的位置定义一个属性，该属性的值为在此位置上找到的英文单词
- 假定我们共有1000个训练文档，其中700个分类为dislike，300个分类为like，现在要对下面的新文档进行分类：
  - This is an example document for the naive Bayes classifier. This document contains only one paragraph, or two sentences.

举例：学习分类文本（3）

- 计算式 $$v_{NB} = \arg\max_{v_j \in \{like, dislike\}} P(v_j) \prod_{i=1}^{19} P(a_i \mid v_j)$$
$$= \arg\max_{v_j \in \{like, dislike\}} P(v_j) P(a_1 = "this" \mid v_j) ... P(a_{19} = "sentences" \mid v_j)$$
- 注意此处贝叶斯分类器隐含的独立性假设并不成立。通常，某个位置上出现某个单词的概率与前后位置上出现的单词是相关的
- 虽然此处独立性假设不精确，但别无选择，否则要计算的概率项极为庞大。
- 另外实践中，朴素贝叶斯学习器在许多文本分类问题中性能非常好

举例：学习分类文本（4）

- 需要估计概率项$P(v_i)$和$P(a_i=w_k|v_i)$。前一项可基于每一类在训练数据中的比例很容易得到，后一项含三个参数，出现数据稀疏问题
- 再引入一个假定以减少需要估计的概率项的数量：假定单词w_k出现的概率独立于单词所在的位置，即$P(a_i=w_k|v_i)=P(w_k|v_j)$
- 作此假定的一个主要优点在于：使可用于估计每个所需概率的样例数增加了，因此增加了估计的可靠程度
- 采纳m-估计方法，即有统一的先验概率并且m等于词汇表的大小，因此

$$P(w_k \mid v_j) = \frac{n_k + 1}{n + |Vocabulary|}$$

表6-2 用于学习和分类文本的朴素贝叶斯算法

- Learn_Naive_Bayes_Text(Examples, V)

  Examples为一组文本文档以及它们的目标值。V为所有可能目标值的集合。此函数作用是学习概率项$P(w_k|v_j)$和$P(v_j)$。
  - 收集Examples中所有的单词、标点符号以及其他记号
    - Vocabulary←在Examples中任意文本文档中出现的所有单词及记号的集合
  - 计算所需要的概率项$P(v_j)$和$P(w_k|v_j)$
    - 对V中每个目标值v_j
      - $docs_j$←Examples中目标值为v_j的文档子集
      - $P(v_j)$←$|docs_j|$ / |Examples|
      - $Text_j$←将$docs_j$中所有成员连接起来建立的单个文档
      - n←在$Text_j$中不同单词位置的总数
      - 对Vocabulary中每个单词w_k
        - » n_k←单词w_k出现在$Text_j$中的次数
        - » $P(w_k|v_j)$←(n_k+1) / $(n+|Vocabulary|)$

表6-2 用于学习和分类文本的朴素贝叶斯算法（2）

- Classify_Naive_Bayes_Text(Doc)

  对文档Doc返回其估计的目标值，a_i代表在Doc中的第i个位置上出现的单词

  - positions←在Doc中的所有单词位置，它包含能在Vocabulary中找到的记号
  - 返回v_{NB}，$v_{NB} = \arg\max_{v_j \in V} P(v_j) \prod_{i \in positions} P(a_i \mid v_j)$

实验结果

- Joachims将此算法用于新闻组文章的分类
  - 每一篇文章的分类是该文章所属的新闻组名称
  - 20个新闻组，每个新闻组有1000篇文章，共2万个文档
  - 2/3作为训练样例，1/3进行性能测量
  - 词汇表不包含最常用词（比如the、of）和罕见词（数据集中出现次数少于3）
- Lang用此算法学习目标概念“我感兴趣的新闻组文章”
  - NewsWeeder系统，让用户阅读新闻组文章并为其评分，然后使用这些评分的文章作为训练样例，来预测后续文章哪些是用户感兴趣的
  - 每天向用户展示前10%的自动评分文章，它建立的文章序列中包含的用户感兴趣的文章比通常高3~4倍

贝叶斯信念网

- 朴素贝叶斯分类器假定各个属性取值在给定目标值v下是条件独立的，从而化简了最优贝叶斯分类的计算复杂度。但在多数情况下，这一条件独立假定过于严厉了。
- 贝叶斯信念网描述的是一组变量所遵从的概率分布，它通过一组条件概率来指定一组条件独立性假设
- 贝叶斯信念网中可表述变量的一个子集上的条件独立性假定，因此，贝叶斯信念网提供了一种中间的方法，它比朴素贝叶斯分类器的限制更少，又比在所有变量中计算条件依赖更可行

贝叶斯信念网（2）

- 贝叶斯信念网描述了一组变量上的概率分布
- 考虑一任意的随机变量集合$Y_1 ... Y_n$，其中每个Y_i可取的值集合为$V(Y_i)$
- 变量集合Y的联合空间为叉乘$V(Y_1) \times ... \times V(Y_n)$
- 在此联合空间上的概率分布称为联合概率分布，联合概率分布指定了元组的每个可能的变量约束的概率
- 贝叶斯信念网则对一组变量描述了联合概率分布

条件独立性

- 精确定义条件独立性
  - 令X, Y和Z为3个离散值随机变量，当给定Z值时X服从的概率分布独立于Y的值，称X在给定Z时条件独立于Y，即

$$(\forall x_i, y_j, z_k) P(X = x_i \mid Y = y_j, Z = z_k) = P(X = x_i \mid Z = z_k)$$

  - 上式通常简写成P(X|Y,Z)=P(X|Z)
- 扩展到变量集合
  - 下面等式成立时，称变量集合$X_1...X_l$在给定变量集合$Z_1...Z_n$时条件独立于变量集合$Y_1...Y_m$

$$P(X_1...X_l \mid Y_1...Y_m, Z_1...Z_n) = P(X_1...X_l \mid Z_1...Z_n)$$

- 条件独立性与朴素贝叶斯分类器的之间的关系

$$\begin{aligned} P(A_1, A_2 \mid V) &= P(A_1 \mid A_2, V) P(A_2 \mid V) \\ &= P(A_1 \mid V) P(A_2 \mid V) \end{aligned}$$

贝叶斯信念网的表示

- 贝叶斯信念网（简称贝叶斯网）表示一组变量的联合概率分布
- 一般地说，贝叶斯网表示联合概率分布的方法是指定一组条件独立性假定（有向无环图）以及一组局部条件概率集合
- 图6-3，联合空间中每个变量在贝叶斯网中表示为一个节点，每个变量需要两种类型的信息
  - 网络弧表示断言“此变量在给定其直接前驱时条件独立于其非后继”
  - 每个变量有一个条件概率表，描述了该变量在给定其立即前驱时的概率分布

贝叶斯信念网的表示（2）

- 对网络变量的元组<$Y_1 \ldots Y_n$>赋以所希望的值($y_1 \ldots y_n$)的联合概率计算公式如下：

$$P(y_1,\ldots,y_n)=\prod_{i=1}^{n}P(y_i \mid Parents(Y_i))$$

$$P(Campfire=True \mid Storm=True, BusTourGroup=True)=0.4$$

- 所有变量的局部条件概率表以及由网络所描述的一组条件独立假定，描述了该网络的整个联合概率分布

贝叶斯信念网的推理

- 可以用贝叶斯网在给定其他变量的观察值时推理出某些目标变量的值
- 由于所处理的是随机变量，所以一般不会赋予目标变量一个确切的值
- 真正需要推理的是目标变量的概率分布，它指定了在给予其他变量的观察值条件下，目标变量取每一个可能值的概率
- 在网络中所有其他变量都确切知道的情况下，这一推理步骤很简单
- 一般来说，贝叶斯网络可用于在知道某些变量的值或分布时计算网络中另一部分变量的概率分布

贝叶斯信念网的推理（2）

- 对任意贝叶斯网络的概率的确切推理已经知道是一个NP难题
- Monte Carlo方法提供了一种近似的结果，通过对未观察到的变量进行随机采样
- 理论上，即使是贝叶斯网络中的近似推理也可能是NP难题
- 实践中许多情况下近似的方法被证明是有效的

学习贝叶斯信念网

- 从训练数据中学到贝叶斯信念网，有多种讨论的框架：
  - 网络结构可以预先给出，或由训练数据中得到
  - 所有的网络变量可以直接从每个训练样例中观察到，或某些变量不能观察到
- 如果网络结构已知且变量可以从训练样例中完全获得，那么得到条件概率表就比较简单
- 如果网络结构已知，但只有一部分变量值能在数据中观察到，学习问题就困难多了。这类似于在人工神经网络中学习隐藏单元的权值
- Russtll（1995）提出了一个简单的梯度上升过程以学习条件概率表中的项，相当于对表项搜索极大似然假设

贝叶斯网的梯度上升训练

- 令w_{ijk}代表条件概率表的一个表项，即在给定父节点U_i取值u_{ik}时，网络变量Y_i值为y_{ij}的概率
- 例如图6-3，w_{ijk}为最右上方的表项，那么Y_i为变量Campfire，U_i是其父节点的元组<Storm, BusTourGroup>，y_{ij}=True，且u_{ik}=<False, False>

贝叶斯网的梯度上升训练（2）

- lnP(D|h)的梯度由对每个w_{ijk}求导数得到

$$\frac{\partial \ln P(D \mid h)}{\partial w_{ijk}} = \sum_{d \in D} \frac{P(Y_i = y_{ij}, U_i = u_{ik} \mid d)}{w_{ijk}}$$

- 例如，为计算图6-3中表左上方的表项的lnP(D|h)的导数，需要对D中每个训练样例d计算P(Campfire=True, Storm=False, BusTourGroup=False|d)
- 当训练样例中无法观察到这些变量时，这些概率可用标准的贝叶斯网从d中观察到的变量中推理得到
- 这些量能够很容易地从贝叶斯网推理过程中得到，几乎不需要附加的开销

贝叶斯网的梯度上升训练（3）

- 式子6.25的推导
  - 用$P_h(D)$来表示$P(D|h)$
  - 假定在数据集D中的各样例d都是独立抽取的

$$
\begin{aligned}
\frac{\partial \ln P_h(D)}{\partial w_{ijk}} &= \frac{\partial}{\partial w_{ijk}} \ln \prod_{d \in D} P_h(d) \\
&= \sum_{d \in D} \frac{\partial \ln P_h(d)}{\partial w_{ijk}} \\
&= \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial P_h(d)}{\partial w_{ijk}} \\
&= \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial}{\partial w_{ijk}} \sum_{j',k'} P_h(d \mid y_{ij'}, u_{ik'}) P_h(y_{ij'}, u_{ik'}) \\
&= \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial}{\partial w_{ijk}} \sum_{j',k'} P_h(d \mid y_{ij'}, u_{ik'}) P_h(y_{ij'} \mid u_{ik'}) P_h(u_{ik'}) \\
&= \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial}{\partial w_{ijk}} P_h(d \mid y_{ij}, u_{ik}) P_h(y_{ij} \mid u_{ik}) P_h(u_{ik}) \\
&= \sum_{d \in D} \frac{1}{P_h(d)} \frac{\partial}{\partial w_{ijk}} P_h(d \mid y_{ij}, u_{ik}) w_{ijk} P_h(u_{ik}) \\
&= \sum_{d \in D} \frac{1}{P_h(d)} P_h(d \mid y_{ij}, u_{ik}) P_h(u_{ik}) \\
&= \sum_{d \in D} \frac{1}{P_h(d)} \frac{P_h(y_{ij}, u_{ik} \mid d) P_h(d) P_h(u_{ik})}{P_h(y_{ij}, u_{ik})} \\
&= \sum_{d \in D} \frac{P_h(y_{ij}, u_{ik} \mid d) P_h(u_{ik})}{P_h(y_{ij}, u_{ik})} \\
&= \sum_{d \in D} \frac{P_h(y_{ij}, u_{ik} \mid d)}{P_h(y_{ij} \mid u_{ik})} \\
&= \sum_{d \in D} \frac{P_h(y_{ij}, u_{ik} \mid d)}{w_{ijk}}
\end{aligned}
$$

贝叶斯网的梯度上升训练（4）

- 更新权值

$$w_{ijk} \leftarrow w_{ijk} + \eta \sum_{d \in D} \frac{P_h(y_{ij}, u_{ik} \mid d)}{w_{ijk}}$$

- 归一化处理，保持在区间[0,1]之间，且$\sum_j w_{ijk}$对所有i,k保持为1

$$w_{ijk} \leftarrow \frac{w_{ijk}}{\sum_{i,k} w_{ijk}}$$

- 这个算法只保证找到局部最优解，替代梯度上升的一个算法是EM算法

学习贝叶斯网的结构

- 如果贝叶斯网的结构未知，那么需要学习贝叶斯网的结构
- Cooper & Herskovits提出了一个贝叶斯评分尺度，以便从不同网络中进行选择
- Cooper & Herskovits提出了算法K2，启发式算法，用于在数据完全可观察时学习网络结构
- 基于约束的学习贝叶斯网络结构：从数据中推导出独立和相关的关系，然后用这些关系来构造贝叶斯网

EM算法

- 在许多实际的学习问题框架中，相关实例特征中只有一部分可观察到
- 已有许多方法被提出来处理存在未观察到变量的问题
  - 比如，如果某些变量有时能观察到，有时不能，那么可以用观察到该变量的实例去预测未观察到的实例中的变量的值
- EM算法是存在隐含变量时广泛使用的一种学习方法，可用于变量的值从来没有被直接观察到的情形，只要这些变量所遵循的概率分布的一般形式已知
  - 用于贝叶斯网的训练
  - 用于马尔可夫模型的训练

估计k个高斯分布的均值

- 考虑D是一个实例集合，它由k个不同正态分布的混合所得分布生成
- 每个实例使用一个两步骤的过程形成：
  - 首先，随机选择k个正态分布中的一个
  - 其次，随机变量x_i按照此选择的分布生成
- 考虑一个简单情形：
  - 单个正态分布的选择基于均匀的概率进行，且k个正态分布有相同的方差
  - 学习任务：输出一个假设$h=<\mu_1...\mu_k>$，描述k个分布中每个分布的均值，找到极大似然假设，即使得$p(D|h)$最大化的假设

估计k个高斯分布的均值（2）

- 当给定从一个正态分布中抽取的数据实例$x_1,...,x_m$时，很容易计算该分布的均值的极大似然假设，它是6.4节中式子6.6的一个特例，表示如下

$$\mu_{ML} = \arg\min_{\mu} \sum_{i=1}^{m} (x_i - \mu)^2 = \frac{1}{m} \sum_{i=1}^{m} x_i$$

- 然而，现在的问题涉及k个不同正态分布，而且不知道哪个实例是哪个分布产生的。这是一个涉及隐藏变量的典型例子
- 对于图6-4的例子，每个实例的完整描述是三元组$<x_i,z_{i1},z_{i2}>$，其中x_i是第i个实例的观测值，z_{i1}和z_{i2}表示哪个正态分布被用来产生x_i，是隐藏变量

估计k个高斯分布的均值（3）

- 如果z_{i1}和z_{i2}的值可知，就可用式子6.27来解决，否则使用EM算法
- EM算法根据当前假设$<\mu_1 ... \mu_k>$，不断地再估计隐藏变量z_{ij}的期望值，然后用这些隐藏变量的期望值重新计算极大似然假设
- 以图6-4为例，先将假设初始化为$h=<\mu_1, \mu_2>$
  - 计算每个隐藏变量z_{ij}的期望值$E[z_{ij}]$，假定当前假设$h=<\mu_1, \mu_2>$成立
  - 计算一个新的极大似然假设$h'=<\mu'_1, \mu'_k>$，假定每个隐藏变量z_{ij}所取值是第一步得到的期望值$E[z_{ij}]$。将假设替换为$h'=<\mu'_1, \mu'_2>$，然后循环

两个步骤的计算式

- $E[z_{ij}]$正是实例x_i由第j个正态分布生成的概率

$$E[z_{ij}] = \frac{p(x = x_i \mid \mu = \mu_j)}{\sum_{n=1}^{2} p(x = x_i \mid \mu = \mu_n)}$$

$$= \frac{e^{-\frac{1}{2\sigma^2}(x_i - \mu_j)^2}}{\sum_{n=1}^{2} e^{-\frac{1}{2\sigma^2}(x_i - \mu_n)^2}}$$

- 第二步，使用第一步得到的$E[z_{ij}]$来导出一新的极大似然假设

$$\mu_j = \frac{\sum_{i=1}^{m} E[z_{ij}]x_i}{\sum_{i=1}^{m} E[z_{ij}]}$$

两个步骤的计算式（2）

- 第二步中的表达式类似于式6.28，只是变成了加权样本均值
- EM算法的要点：当前的假设用于估计未知变量，而这些变量的期望值再被用于改进假设
- 可以证明：算法的每一次循环中，EM算法能使似然P(D|h)增加，除非P(D|h)达到局部最大，因此算法收敛到一个局部最大似然假设

EM算法的一般表述

- EM算法可用于许多问题框架：其中需要估计一组描述基准概率分布的参数，只给定了由此分布产生的全部数据中能观察到的一部分。
  - 上面的二均值问题中，感兴趣的参数是$\theta=<\mu_1, \mu_2>$，全部数据是三元组$<x_i,z_{i1},z_{i2}>$，而只能观察到xi
- 一般地，令待估计参数是θ，全部数据$Y=X\cup Z$，其中X是可观察数据，Z是未观察数据。
- Z可看作一个随机变量，它的概率分布依赖于参数θ和已知数据X
- Y也是一个随机变量，因为它由随机变量Z定义

EM算法的一般表述（2）

- EM算法通过搜寻使E[lnP(Y|h')]最大的h'来寻找极大似然假设h'，其合理性是：
  - P(Y|h')是给定假设h'下全部数据Y的似然度，因此找到使得这个值最大的h'是合理的
  - 对数lnP(Y|h')最大化也使P(Y|h')最大化
  - 由于Y是一个随机变量，因此P(Y|h')无法计算，转而计算它的期望值E[lnP(Y|h')]
- Y的概率分布由待估计的参数决定，EM算法使用当前假设h代替实际参数，来估计Y的概率分布
- 定义函数Q(h'|h)=E[lnP(Y|h')|h,X]

EM算法的一般表述（3）

- EM算法的一般形式，重复下面的步骤，直至收敛
  - 估计（E）步骤：使用当前假设h和观察到的数据X来估计Y上的概率分布以计算Q(h'|h)

    $Q(h'|h) \leftarrow E[\ln P(Y|h')|h,X]$
  - 最大化（M）步骤：将假设h替换为使Q函数最大化的假设h'

    $h \leftarrow \text{argmax}_{h'} Q(h'|h)$
- 当函数Q连续时，EM算法收敛到似然函数P(Y|h')的一个不动点，它保证收敛到一个局部最大值

K均值算法的推导

- 问题框架
  - 要估计k个正态分布的均值$\theta=<\mu_1...\mu_k>$
  - 观察到的数据是$X=\{<x_i>\}$
  - 隐藏变量$Z=\{<z_{i1},...,z_{ik}>\}$表示k个正态分布中哪一个生成$x_i$
- 用于K均值问题的表达式Q(h'|h)的推导
  - 单个实例的概率

$$p(y_i \mid h') = p(x_i, z_{i1},..., z_{ik} \mid h') = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}\sum_{j=1}^{k} z_{ij}(x_i-\mu_j')^2}$$

K均值算法的推导（2）

– 所有实例的概率的对数

$$\begin{aligned}\ln P(Y \mid h') &= \ln \prod_{i=1}^{m} p(y_i \mid h') \\ &= \sum_{i=1}^{m} \ln p(y_i \mid h') \\ &= \sum_{i=1}^{m} \left(\ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \sum_{j=1}^{k} z_{ij} (x_i - \mu_j')^2 \right)\end{aligned}$$

– 计算期望值

$$\begin{aligned}E[\ln P(Y \mid h')] &= E\left[\sum_{i=1}^{m} \left(\ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \sum_{j=1}^{k} z_{ij} (x_i - \mu_j')^2 \right) \right] \\ &= \sum_{i=1}^{m} \left(\ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \sum_{j=1}^{k} E[z_{ij}] (x_i - \mu_j')^2 \right)\end{aligned}$$

K均值算法的推导（3）

- 求使Q函数最大的假设

$$\arg\max_{h'} Q(h' \mid h) = \arg\max_{h'} \sum_{i=1}^{m} \left(\ln \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \sum_{j=1}^{k} E[z_{ij}](x_i - \mu_j')^2 \right)$$

$$= \arg\max_{h'} \sum_{i=1}^{m} \sum_{j=1}^{k} E[z_{ij}](x_i - \mu_j')^2$$

- 解上式得到

$$\mu_j \leftarrow \frac{\sum_{i=1}^{m} E[z_{ij}] x_i}{\sum_{i=1}^{m} E[z_{ij}]}$$

- 另外

$$E[zij] \leftarrow \frac{e^{-\frac{1}{2\sigma^2}(x_i - \mu_j)2}}{\sum_{n=1}^{k} e^{-\frac{1}{2\sigma^2}(x_i - \mu_j)^2}}$$

小结

- 概率学习方法利用关于不同假设的先验概率，以及在给定假设时观察到不同数据的概率的知识
- 贝叶斯方法提供了概率学习方法的基础，基于这些先验和数据观察假定，赋予每个假设一个后验概率
- 贝叶斯方法确定的极大后验概率假设是最可能成为最优假设的假设
- 贝叶斯最优分类器将所有假设的预测结合起来，并用后验概率加权，以计算对新实例的最可能分类
- 朴素贝叶斯分类器增加了简化假定：属性值在给定实例的分类时条件独立
- 贝叶斯信念网能够表示属性的子集上的一组条件独立性假定

小结（2）

- 贝叶斯推理框架可对其他不直接应用贝叶斯公式的学习方法的分析提供理论基础
- 最小描述长度准则建议选取这样的假设，它使假设的描述长度和给定假设下数据的描述长度的和最小化。贝叶斯公式和信息论中的基本结论提供了此准则的根据
- EM算法提供了一个通用的算法，在存在隐藏变量时进行学习。算法开始于一个任意的初始假设，然后迭代地计算隐藏变量的期望值，再重新计算极大似然假设，这个过程收敛到一个局部极大似然假设和隐藏变量的估计值

补充读物

- Casella & Berger 1990在概率和统计方面的介绍性文章
- Maisel 1971, Speigel 1991的快速参考书籍
- Duda & Hart 1973对贝叶斯分类器和最小平方误差分类器的介绍
- Domigos & Pazzani 1996分析了朴素贝叶斯分类器输出最优分类的条件
- Cestnik 1990讨论了m-估计
- Michie et al. 1994将不同贝叶斯方法与决策树等其他算法进行比较
- Chauvin & Rumelhart1995提供了基于反向传播算法的神经网络的贝叶斯分析
- Rissanen1 1983, 1989讨论了最小描述长度准则
- Quinlan & Rivest 1989描述了利用最小描述长度准则避免决策树过度拟合的方法

机器学习

第11章 分析学习

概述

- 神经网络和决策树这样的学习方法需要一定数目的训练样例才能达到一定级别的泛化精度
- 分析学习使用先验知识和演绎推理来扩大训练样例提供的信息，因此它不受同样的界限制约
- 本章讨论一种称为基于解释的学习（EBL）的分析学习方法
- 基于解释的学习中，先验知识用于分析观察到的学习样例是怎样满足目标概念的
- 然后这个解释用于区分训练样例中哪些是相关的特征，哪些是不相关的
- 样例就可基于逻辑推理进行泛化，而不是基于统计推理

简介

- 前面章节讨论的各种归纳法，决策树、神经网络、归纳逻辑编程、遗传算法，在实践中的一个关键限制是：在可用数据不足时性能较差，正如第7章分析，给定数目的训练样例，学习的精度存在基本的上下界
- 我们希望开发出这样的学习方法：它们训练精度上的基本限制不受可用训练数据的数量所制约
- 基于解释的学习：
  - 使用先验知识来分析或解释每个训练样例，以推理出样例的哪些特征与目标函数相关，哪些不相关
  - 减小了待搜索假设空间的复杂度，减小了样本复杂度，提高了学习器的泛化精度

简介（2）

- 一个例子：下国际象棋的学习任务
  - 前面的概念学习算法需要大量的训练样例
  - 人类只要少数训练样例，原因是人类非常依赖合法移动棋子的先验知识来解释或分析训练样例
  - 但是，人类学习中包含了一个很长的发现先验知识的过程
- 本章内容安排
  - 给出一个特定的基于解释的学习算法，称为Prolog-EBG
  - 考查Prolog-EBG的一般特性以及与前面讨论的归纳算法之间的关系
  - 描述了应用基于解释的学习以提高大状态空间搜索的性能
- 本章假定生成解释所基于的先验知识是完全正确的，下一章讨论更一般的情况，即先验知识只是近似正确

归纳和分析学习问题

- 分析和归纳学习问题的重要区别是，它们设想的学习问题的形式不同
  - 在归纳学习中，学习器被赋予一个假设空间H和训练数据D，它从H中选择一个输出假设，并且希望这个假设与D一致
  - 在分析学习中，学习器的输入除了假设空间H和训练数据D，还有一个领域理论B，由可用于解释训练样例的背景知识组成，学习器中H中选择一个输出假设，并希望这个假设既与D一致，也与B一致
- 分析学习举例
  - 学习的目标概念：黑棋将在两步内失去王后的状态
  - 实例$<x_i, f(x_i)>$：x_i描述一特定棋盘状态，当黑棋两步内失去王后，$f(x_i)$值为真，否则为假
  - 假设空间：用Horn子句集表示，其中谓词表示棋子的位置
  - 领域理论：形式化的下棋规则

归纳和分析学习问题（2）

- 在分析学习中，引入一致性约束：当领域理论B不涵蕴h的否定时，则称h与B一致
- 一致性约束减少了当数据不能单独在H中决定h时，学习器面临的歧义性
- 领域理论也由一组Horn子句描述，它使系统原则上可以加入任何学习到的假设至后续的领域理论中

例子，表11-1分析学习问题：SafeToStack(x,y)

- 已知
  - 实例空间X：每个实例描述一对物理对象，它们由谓词Color, Volume, Owner, Material, Type, Density描述，它们之间的关系用谓词On描述
  - 假设空间H：每个假设是一组Horn子句规则。每个Horn子句的头部为一个包含目标谓词SafeToStack的文字，每个Horn子句为文字的合取，这些文字基于描述实例的谓词以及谓词LessThan, Equal, GreaterThan和函数plus, minus和time，如下例SafeToStack(x,y)←Volume(x,vx)∧Volume(y,vy)∧LessThan(vx,vy)
  - 目标概念：谓词SafeToStack(x,y)，表示两个物理对象，一个可被安全地叠放在另一个上
  - 训练样例：下面显示了一个典型的正例SafeToStack(Obj1,Obj2)：

    On(Obj1,Obj2)　　Owner(Obj1,Fred)

    Type(Obj1,Box)　　Owner(Obj2,Louise)

    ...
  - 领域理论B：

    SafeToStack(x,y)←¬Fragile(y)

    SafeToStack(x,y)←Lighter(x,y)

    ...
- 求解
  - H中一个与训练样例和领域理论一致的假设

用完美的领域理论学习：Prolog-EBG

- 本章考虑的基于解释的学习是在领域理论完美的情况下，即领域理论正确且完整
  - 当领域理论中每个断言都是客观的真实描述时，该领域理论被称为是正确的
  - 当领域理论覆盖了实例空间中所有正例时，该领域理论被称为是完整的
    - 每个满足目标概念的实例都可由领域理论证明其满足性
    - 根据Prolog惯例，不能证明的断言认定为假
    - 因此完整性定义包含全部正例和反例
- 对于学习器的完美领域理论的假定的合理性的解释
  - 在某些情况下，有可能提供完美领域理论。比如下棋问题，棋子的合法走子提供了完美的领域理论
  - 在许多情况下，不能够假定有完美的领域理论，但我们可以使用基于不完美领域理论的近似合理的解释，它以完美理论为基础

Prolog-EBG算法

- Prolog-EBG是一种基于解释的学习方法，是一种序列覆盖算法
  - 学习单个Horn子句规则，移去此规则覆盖的正例
  - 在剩余正例上重复这个过程，直到覆盖所有正例为止
- 对于任意的正例集合，Prolog-EBG输出的假设包含一组对应于领域理论的目标概念的逻辑充分条件

表11-2基于解释的学习算法 Prolog-EBG

Prolog-EBG(TargetConcept, TrainingExample, DomainTheroy)

- LearnedRules={}
- Pos=TrainingExamples中的正例
- 对Pos中没有被LearnedRules覆盖的每个正例，做
  - 解释：
    - Explanation=以DomainTheory表示的解释，说明正例满足TargetConcept
  - 分析：
    - SuffcientConditions=按照Explanation能够充分满足TargetConcept的正例的最一般特征集合
  - 改进：
    - LearnedRules=LearnedRules+NewHornClause，其中NewHornClause的形式是：TargetConcept←SufficientConditions
- 返回LearnedRules

Prolog-EBG的运行举例

- Prolog-EBG对每个还没有被某个Horn子句覆盖的正例，通过下列步骤生成一新Horn子句
  - 解释新的正例
  - 分析该解释以确定一合适的泛化
  - 通过加入一新的Horn子句以覆盖该正例以及其他相似实例来改进当前假设

解释训练样例

- 按照领域理论建立解释，说明该正例如何满足目标概念，当领域理论正确且完整时，此解释构成了训练样例满足目标概念的证明
- 例子图11-2
- 一般情况下，可能有多种解释，这些解释中任意一个或所有的都可被使用，每个解释可对训练样例形成不同的泛化，所有解释都将被给定的领域理论论证
- 在Prolog-EBG中，解释的生成使用了如Prolog中的反向链式搜索，找到第一个有效证明时终止

分析解释

- 由学习器构造的解释回答了哪些特征与目标概念相关，哪些无关
- 图11-2的例子
- 通过收集解释的叶节点中提及的特征，可形成一个由领域理论论证的一般规则
- 形成的规则构成了此训练样例的一个有意义的泛化，因为它去除了样例的许多与目标概念无关的属性
- 通过更仔细地分析解释，能够得到更一般的规则
- Prolog-EBG通过计算解释的最弱前像，能够得到由解释论证的最一般的规则

分析解释（2）

- 定义：结论C对应于证明P的最弱前像为最一般的初始断言集合A，使得A按照P涵蕴C
- Prolog-EBG计算目标概念的关于解释的最弱前像的过程，使用的是回归过程
- 回归过程的工作方式是在解释中反复后退
  - 首先对应于解释中最后证明步计算目标概念的最弱前像
  - 然后对应于其前一步计算结果表达式的最弱前像，依次类推
  - 这个过程在遍历过解释中所有步骤后终止，得到对应于解释的叶节点上的文字的目标概念的最弱前件
- 图11-3
- 回归过程的核心是，每一步通过领域理论的一条Horn子句回归当前边缘表达式的算法

分析解释（3）

- 回归算法的操作过程是，寻找一个置换使Horn子句的头与边缘中的相应文字合一，用规则体替换边缘中的表达式，再应用一个合一置换到整个边缘
- Prolog-EBG输出的最终Horn子句形式如下：子句体被定义为上述过程计算出的最弱前件，子句头为目标概念本身
- 应用置换到每一回归步中，以便子句头和子句体保持一致变量名

表11-3 通过一个Horn子句回归一组文字的算法

Regress(Frontier, Rule, Literal, θ_{hi})

Frontier：通过规则被回归的文字集合

Rule：Horn子句

Literal：在Frontier中的文字，它由解释中的Rule推得

θ_{hi}：是Rule的头与解释中的相应文字合一的置换

返回构成Frontier的关于Rule的最弱前像的文字集合

- head←Rule的头
- body←Rule的体
- θ_{hl}←head与Literal的最一般合一，使得存在置换θ_{li}满足：$\theta_{li}(\theta_{hl}(head))=\theta_{hi}(head)$
- 返回θ_{hl}(Frontier-head+body)

表11-3 通过一个Horn子句回归一组文字的算法（示例）

Regress(Frontier, Rule, Literal, θ_{hi})

Frontier={Volume(x,vx), Density(x,dx), Equal(wx,times(vx,dx), LessThan(wx,wy), Weight(y,wy)}

Rule=Weight(z,5)←Type(z,Endtable)

Literal=Weight(y,wy)

θ_{hi}={z/Obj2}

- head←Weight(z,5)
- body←Type(z,Endtable)
- θ_{hl}={z/y, wy/5}, θ_{li}={y/Obj2}
- 返回{Volume(x,vx), Density(x,dx), Equal(wx,times(vx,dx)), LessThan(wx,5), Type(y,Endtable)}

改进当前假设

- 每一阶段的当前假设由当时学习到的Horn子句集组成，每一阶段，算法选取还未被当前Horn子句覆盖的新正例，解释该正例并按照上面的过程形成新规则
- 新规则加入到当前假设中

对基于解释的学习的说明

- Prolog-EBG算法的要点
  - Prolog-EBG不像归纳的方法，它运用先验知识分析单个样例以产生合理的一般假设
  - 对样例如何满足目标概念的解释，确定了样例的哪些属性是相关的，即在解释中提及的属性
  - 对解释的进一步分析，即回归目标概念以确定其对应解释的最弱前像，可推导出相关特征值的一般约束
  - 每个学习到的Horn子句对应于满足目标概念的一个充分条件，学习到的Horn子句集覆盖了学习器遇到的正例，以及其他与此共享同样解释的实例
  - 学习到的Horn子句的泛化将依赖于领域理论的形式以及训练样例被考虑的序列
  - 算法隐含假定了领域理论是正确且完整的，如果领域理论不正确或不完整，学到的概念也将不正确

对基于解释的学习的说明（2）

- 基于解释的学习的特点
  - 作为理论引导的样例泛化
    - 使用给定的领域理论以从样例中合理泛化，区分出相关和不相关的样例属性，因此可以避免用于纯归纳推理中的样本复杂度界限
  - 作为样例引导的理论重建
    - Prolog-EBG算法被看作是一种重建领域理论到一种可操作形式的方式
    - 重建领域理论是通过创建这样的规则
      - 能从领域理论中演绎派生
      - 在一个推理步内分类观察到的训练样例
    - 学习到的规则可看作是领域理论的重组，它们能够在一个推理步内对目标概念的实例分类

对基于解释的学习的说明（3）

  - 仅仅重述学习器已经知道的
    - 学习器并没有学习新的知识，意义在于
      - 原则上已知的和实践上可有效计算的之间的区别很大，因此这种“知识重建”也是学习的重要形式
    - 学习到的规则直接从可观察到实例映射得到，方法是使其与基本领域理论一致
    - 使用原始的领域理论可能需要许多推理步和很乐观的搜索才能对任意实例分类
    - 学习到的规则可在一个推理步内分类观察到的实例
- 基于解释的学习致力于重建领域理论，产生单步推理出样例分类的一般规则
- 这种知识重建的过程有时被称为知识汇编，表示这种转换是为了增加知识使用的效率，而不改变知识的正确性和完备性

发现新特征

- Prolog-EBG一个有趣的能力是形成在训练样例的描述中没有显示出现的新特征
- 这些学习到的“新特征”类似于由神经网络的隐藏单元表示的特征类型
- 不像神经网络中使用统计过程从多个训练样例中推导出隐藏单元特征，Prolog-EBG应用了一个分析过程基于单个训练样例的分析推导新特征
- 领域理论中的最初项的特定合成和实例化导致了新特征的定义

演绎学习

- 纯粹的Prolog-EBG是一个演绎的而不是归纳的学习过程，它输出一个假设满足下面的约束
  - $(\forall \langle x_i, f(x_i)\rangle \in D)(h \wedge x_i) \Rightarrow f(x_i)$
  - $D \wedge B \Rightarrow h$
- 第一个约束只是简单地将机器学习的需求形式化，第二个约束描述了领域理论的作用：输出假设被进一步约束，使其符合领域理论和数据
- 第二个约束减少了学习器在选择假设时面临的歧义性，因此领域理论减少了假设空间的规模并降低了学习的样本复杂度
- 实质上，Prolog-EBG假定领域理论B涵蕴训练数据中实例的分类，即

  $(\forall \langle x_i, f(x_i)\rangle \in D)(B \wedge x_i) \Rightarrow f(x_i)$

演绎学习（2）

- Prolog-EBG和ILP的比较
  - ILP中也使用到了背景知识B’，B’一般不满足式子11.3的约束
  - ILP是一个归纳学习系统，而Prolog-EBG是演绎学习系统
  - ILP使用背景知识来扩大待考虑的假设集合，而Prolog-EBG使用领域理论来减小可接受假设的集合
  - ILP要求：$(\forall <xi,f(xi)> \in D)(B' \wedge h \wedge xi) \Rightarrow f(xi)$，而Prolog-EBG要求更严格：
    $(\forall <xi,f(xi)> \in D)(h \wedge xi) \Rightarrow f(xi)$

基于解释的学习的归纳偏置

- 根据第2章，一个学习算法的归纳偏置为一组断言，它们与训练样例一起演绎后续预测
- Prolog-EBG的归纳偏置
  - 似乎是领域理论B，但由于领域理论可涵蕴多个可选的Horn子句集，因此归纳偏置还需包含在这些子句集中做出选择的内容
  - 由于每个单独的Horn子句是当前训练样例的解释所许可的最一般子句，因此归纳偏置为对极大一般化Horn子句的小集合的偏好
  - 实际上，Prolog-EBG的贪婪算法只是为寻找极大一般化Horn子句的真正最短集合所需的彻底搜索算法的一个启发式的近似
- 近似的Prolog-EBG归纳偏置：领域理论B，加上对极大一般化Horn子句的小集合的偏好

基于解释的学习的归纳偏置（2）

- Prolog-EBG的归纳偏置在很大程度上由输入的领域理论决定，这与前面讨论的许多算法不同
- 前面讨论的算法的归纳偏置是学习算法的一个固定属性，一般由其假设表示的语法所确定
- 把归纳偏置作为一个输入参数而不是学习器的固定属性十分重要
- 一个通用的学习方法至少会允许归纳偏置能够随待解决的学习问题变化，通过修改输入参数比通过限制假设的语法形式来实现偏置性要方便得多
- 比如一个自治agent随着时间改进它的学习能力，那么最好有一个算法，它的泛化能力可在其获得更多的领域知识后增强

知识级的学习

- Prolog-EBG算法中假设h可以直接从B中派生，与D无关
  - 可以设想有一个条目枚举器，它基于领域理论B中的断言简单地枚举能得到目标概念的所有证明树
  - 条目枚举器用与Prolog-EBG相似的方法计算最弱前像并构造一个Horn子句
- 条目枚举器输出的是Prolog-EBG输出的子句的超集，存在下面的特点：
  - 训练样例的用途：使算法更关注覆盖实际出现的样例分布的生成规则，比尝试枚举棋盘的所有可能条目更可能得到更小、更相关的规则集
  - Prolog-EBG不会学习到一个超出隐含在领域理论中的知识的假设

知识级的学习（2）

- Prolog-EBG不会学习到一个超出隐含在领域理论中的知识的假设，但这不是分析学习或演绎学习的固有缺陷
  - 能够找到一个B不涵蕴h，但B∧D涵蕴h的例子，如
    - B=(∀x) IF ((PlayTennis=Yes)←(Humidity=x)) THEN ((PlayTennis=Yes)←(Humidity≤x))
    - D=Humidity=0.3
    - h=(PlayTennis=Yes)←(Humidity≤0.3)
- 知识级学习被用来称这类型的学习：学到的假设涵蕴的预测不能被单独的领域理论涵蕴
- 由断言集合Y涵蕴的所有预测的集合常称为Y的演绎闭包，知识级学习中B的演绎闭包是B+h演绎闭包的真子集

搜索控制知识的基于解释的学习（？？？）

小结

- 纯粹的归纳学习方法寻找一个假设以拟合训练数据，而纯粹的分析学习方法搜寻一个假设拟合学习器的先验知识并覆盖训练样例
- 基于解释的学习是分析学习的一种形式，其中学习器处理每个新训练样例的方法是：
  - 按照领域理论解释该样例中观察到的目标值
  - 分析此解释，确定解释成立的一般条件
  - 改进假设，合并这些一般条件
- Prolog-EBG是基于解释的学习算法，它使用一阶Horn子句来表示其领域理论和学到的假设，在Prolog-EBG中，解释即是Prolog证明，而从解释中抽取的假设是此证明的最弱前像

小结（2）

- Prolog-EBG这样的分析学习方法建立游泳的中间特征，它是分析单独训练样例的一个副产品，这种生成特征的分析途径补充了如反向传播这样的归纳方法中基于统计方法的中间特征生成
- Prolog-EBG不会产生能扩展其领域理论的演绎闭包的假设，但其他演绎学习过程具备这个能力
- 可应用正确且完整的领域理论的一类重要问题是大的状态空间的搜索问题，如Prodigy和Soar这样的系统已显示了基于解释的学习方法的效用，它们自动获取有效的搜索规则以加速后续的问题求解
- 纯粹的演绎推理的一个缺点是：它输出的假设的正确性只在领域理论正确时才能保证

补充读物

- Fikes et al.1972，通过对Abstrips中的算子的分析学习宏算子
- Soloway1977，在学习中使用明确的先验知识
- Dejong1981, Mitchell1981, Winston et al.1983, Silver1983讨论了基于解释的学习
- Ram & Leake1995，给出了关于目的和鲜艳知识在人类和记起学习中的作用的综述
- Laird et al.1986提出的Soar系统和Carbonell et al.1990描述的Prodigy系统是使用基于解释的学习来学习问题求解的两个最成熟的系统

补充读物（2）

- 对人类学习的实验性研究支持了这样一个猜想，即人类的学习是基于解释的
  - Ahn et al.1987和Qin et al.1992概述了支持人类应用基于解释的学习过程这一猜想的证据
  - Wisniewski & Medin1995描述了对人类学习 的实验性研究，它建议在先验知识和观察数据之间进行丰富的相互作用以影响学习过程
  - Kotovsky & Baillargeon1994描述的实验说明，即使11个月大的婴儿在学习时也是基于其先验知识的
- Van Harmelen & Bundy1988提供了基于解释的学习中执行的分析与Prolog程序中使用的几类优化方法的关系

机器学习

第2章 概念学习和一般到特殊序

提纲

- 概念学习
  - 给定某一类别的若干正例和反例，从中获得该类别的一般定义。
- 搜索的观点
  - 在预定义的假设空间中搜索假设，使其与训练样例有最佳的拟合。
  - 利用假设空间的偏序结构
- 算法收敛到正确假设的条件
- 归纳学习的本质，从训练数据中泛化的理由

简介

- 许多机器学习涉及到从特殊训练样例中得到一般概念。
- 概念，可被看作一个对象或事件集合，它是从更大的集合中选取的子集，或在这个较大集合中定义的布尔函数。
- 概念学习问题的定义
  - 给定一个样例集合以及每个样例是否属于某个概念的标注，怎样推断出该概念的一般定义。又称从样例中逼近布尔函数。
  - 概念学习是指从有关某个布尔函数的输入输出训练样例中推断出该布尔函数。

概念学习任务

- 一个例子
  - 目标概念，Aldo进行水上运动的日子，表示为布尔函数EnjoySport
  - 任务目的，基于某天的各属性，预测EnjoySport的值
  - 一个样例集，每个样例表示为属性的集合

概念学习任务（2）

表2-1 目标概念EnjoySport的训练样例

Example	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	Yes
4	Sunny	Warm	High	Strong	Cool	Change	Yes

概念学习任务（3）

- 表示假设的形式
  - 一个简单的形式，实例的各属性约束的合取式
  - 令每个假设为6个约束（或变量）的向量，每个约束对应一个属性可取值范围，为
    - ？任意本属性可接受的值
    - 明确指定的属性值
    - ϕ 不接受任何值
  - 假设的例子
    - <?, Cold, High, ?, ?, ?>
    - <?, ?, ?, ?, ?, ?>　　// 所有的样例都是正例
    - < ϕ, ϕ, ϕ, ϕ, ϕ, ϕ>　　// 所有的样例都是反例

概念学习任务（4）

EnjoySport概念学习任务

- 已知
  - 实例集X
    - 每个实例x由6个属性描述，每个属性的取值范围已确定
  - 假设集H
    - 每个假设h描述为6个属性的取值约束的合取
  - 目标概念c
    - 一个布尔函数，变量为实例
  - 训练样例集D
    - 目标函数（或目标概念）的正例和反例
- 求解
  - H中的一假设h，使对于X中任意x，h(x)=c(x)

术语定义

- 实例x
- 实例集X
- 概念
- 目标概念c
- 训练样例x
- 训练样例集D
- 正例，目标概念成员
- 反例，非目标概念成员
- 假设h
- 假设集H

机器学习的目标就是寻找一个假设h，使得对所有的h，都有h(x)=c(x)

归纳学习假设

- 什么是归纳学习？
  - 从特殊的样例得到普遍的规律
- 归纳
  - 只能保证输出的假设能与训练样例相拟合
- 归纳假设的一个基本假定
  - 对于未见实例最好的假设就是与训练数据最佳拟合的假设
- 归纳学习假设
  - 任一假设如果在足够大的训练样例集中很好地逼近目标函数，它也能在未见实例中很好地逼近目标函数。

作为搜索的概念学习

- 概念学习可以看作一个搜索的过程
  - 搜索范围：假设的表示所隐含定义的整个空间
  - 搜索目标：能够最好地拟合训练样例的假设
- 当假设的表示形式选定后，那么就隐含地为学习算法确定了所有假设的空间
- 例子EnjoySport的假设空间

假设的一般到特殊序

- 假设的一般到特殊序关系
  - 考虑下面两个假设
    - h_1=<sunny, ?, ?, Strong, ?, ?>
    - h_2=<Sunny, ?, ?, ?, ?, ?>
  - 任何被h_1划分为正例的实例都会被h_2划分为正例，因此h_2比h_1更一般。
- 利用这个关系，无需列举所有假设，就能在无限的假设空间中进行彻底的搜索

假设的一般到特殊序（2）

- 关系“更一般”的精确定义
  - 任给实例x和假设h，说x满足h，当且仅当 $h(x)=1$
  - 令h_j和h_k是在X上定义的布尔函数，称h_j比h_k更一般，当且仅当 $(\forall x \in X)[(h_k(x)=1) \rightarrow (h_j(x)=1)]$
  - 记为h_j more_general_than_or_equal_to h_k，或 $h_j \geq_g h_k$

假设的一般到特殊序（3）

- “更一般”的严格情形
  - $h_j >_g h_k$，当且仅当，$(h_j \geq_g h_k) \wedge \neg (h_k \geq_g h_j)$
- “更特殊”关系的定义
  - $h_j \leq_g h_k$，当且仅当，$h_k \geq_g h_j$
- 以EnjoySport为例说明上面的定义
- 偏序的特点（区别于全序），全序上的搜索可以是二分法，偏序的搜索比无序简单，比全序复杂。
- 这个偏序关系的定义与目标概念无关

Find-S：寻找极大特殊假设

- 使用more_general_than偏序的搜索算法
  - 从H中最特殊假设开始，然后在假设覆盖正例失败时将其一般化

表2-3 Find-S算法

1. 将h初始化为H中最特殊假设
2. 对每个正例x
   - 对h的每个属性约束a_i
     如果x满足a_i
     那么不做任何处理
     否则将h中a_i替换为x满足的另一个更一般约束
3. 输出假设h

Find-S：寻找极大特殊假设（2）

- Find-S算法在例子EnjoySport上的应用
  - h←<ϕ, ϕ, ϕ, ϕ, ϕ, ϕ>
  - h←<Sunny, Warm, Normal, Strong, Warm, Same>
  - h←<Sunny, Warm, ?, Strong, Warm, Same>
  - 遇到反例，h不变（因为h已经能够正确地识别反例）
  - h←<Sunny, Warm, ?, Strong, ?, ?>

Find-S：寻找极大特殊假设（3）

- Find-S算法演示了一种利用more_general_than偏序来搜索假设空间的方法，沿着偏序链，从较特殊的假设逐渐转移到较一般的假设。因此，每一步得到的假设都是在那一点上与训练样例一致的最特殊的假设。
- Find-S的重要特点：对以属性约束的合取式描述的假设空间H，保证输出为H中与正例一致的最特殊的假设。
- 存在的问题
  - 是否收敛到了正确的目标概念？
  - 为什么要用最特殊的假设？
  - 训练样例是否相互一致？
  - 如果有多个极大特殊假设怎么办？

变型空间和候选消除算法

- 候选消除算法概说
  - 概念学习的另一种方法，候选消除算法（candidate-elimination）
  - Find-S算法的不足，输出的假设只是H中能够拟合训练样例的多个假设中的一个
  - 候选消除算法输出与训练样例一致的所有假设的集合
  - 候选消除算法在描述这一集合时不需要明确列举所有成员
  - 利用more_general_than偏序结构，可以维护一个一致假设集合的简洁表示
  - 候选消除算法的应用，化学质谱分析、启发式搜索的控制规则
  - 候选消除算法的缺点，容错性能差

变型空间和候选消除算法（2）

- “一致”的定义
  - 一个假设h与训练样例集合D一致，当且仅当对D中每一个样例<x,c(x)>都有h(x)=c(x)，即
    $Consistent(h,D)\Leftrightarrow(\forall<x,c(x)>\in D)h(x)=c(x)$
  - “一致”与“满足”的关系
- 变型空间（version space）
  - 与训练样例一致的所有假设组成的集合
  - 表示了目标概念的所有合理的变型
- 关于H和D的变型空间，记为$VS_{H,D}$，是H中与训练样例D一致的所有假设构成的子集
  $VS_{H,D}=\{h\in H|Consistent(h,D)\}$

变型空间和候选消除算法（3）

- 列表后消除算法

  表示变型空间的一种方法是列出其所有成员

  – 变型空间←包含H中所有假设的列表
  – 对每个训练样例<x,c(x)>，从变型空间中移除所有h(x)≠c(x)的假设
  – 输出Version Space中的假设列表
- 优点
  – 保证得到所有与训练数据一致的假设
- 缺点
  – 非常繁琐地列出H中的所有假设，大多数实际的假设空间无法做到

变型空间和候选消除算法（4）

- 变型空间的更简洁表示
  - 变型空间被表示为它的极大一般和极大特殊的成员
  - 这些成员形成了一般和特殊边界的集合，这些边界在整个偏序结构中划分出变型空间
  - 以EnjoySport为例

变型空间和候选消除算法（5）

- 形式化定义
  - 极大一般
  - 极大特殊
  - 关于假设空间H和训练数据D的**一般边界**G，是在H中与D相一致的极大一般成员的集合
  - 关于假设空间H和训练数据D的**特殊边界**S，是在H中与D相一致的极大特殊成员的集合

变型空间和候选消除算法（6）

变型空间表示定理，令X为一任意的实例集合，H为X上定义的布尔假设的集合。令c: $X \rightarrow \{0,1\}$为X上定义的任一目标概念，并令D为任一训练样例集合$\{<x,c(x)>\}$。对所有的X, H, c, D以及良好定义的S和G：

$$VS_{H,D}=\{h \in H | (\exists s \in S)(\exists g \in G)(g \geq_g h \geq_g s)\}$$

证明：只需证明：1）每一个满足上式右边的h都在$VS_{H,D}$中，2）$VS_{H,D}$的每个成员都满足都满足等式右边。…

变型空间和候选消除算法（7）

- 候选消除算法
  - 初始化G和S
  - 如果d是一个正例
    - 从G中移去所有与d不一致的假设
    - 对S中每个与d不一致的假设s
      - 从S中移去s
      - 把s的所有的极小泛化式h加入到S中，其中h满足
        - » h与 d一致，而且G的某个成员比h更一般
  - 如果d是一个反例
    - 从S中移去所有与d不一致的假设
    - 对G中每个与d不一致的假设g
      - 从G中移去g
      - 把g的所有的极小特殊化式h加入到G中，其中h满足
        - » h与d一致，而且S的某个成员比h更特殊
      - 从G中移去所有这样的假设：G中某个假设更特殊

变型空间和候选消除算法（8）

- 算法举例

变型空间和候选消除的说明

- 候选消除算法收敛到正确的假设
  - 训练样例中没有错误
  - H中确实包含描述目标概念的正确假设
- 如果样例中存在错误
  - 如果给定足够的训练数据，我们会发现S和G边界收敛得到一个空的变型空间
- 如果目标概念不能由假设表示方式所描述
  - 相似情况出现

变型空间和候选消除（2）

- 下一步需要什么样的训练样例
  - 一般来说，概念学习的最优查询策略，是产生实例以满足当前变型空间中大约半数的假设。这样，变型空间的大小可以在遇到每个新样例时减半，正确的目标概念就可在只用$\log_2|VS|$次实验后得到。

变型空间和候选消除（3）

- 怎样使用不完全学习概念
  - 虽然图2-3的变型空间中仍包含多个假设，即目标概念还未学习到，但是仍然有可能对新样例进行一定可信度的分类。
  - 表2-6的例子

归纳偏置

- 有关候选消除算法的几个问题
  - 如果目标概念不在假设空间中怎么办？
  - 是否可设计一个包含所有假设的空间来解决这一困难？
  - 假设空间的大小对于算法推广到未见实例的能力有什么影响？
  - 假设空间的大小对所需训练样例的数量有什么影响？

归纳偏置（2）

- 一个有偏的假设空间
  - 在EnjoySport这个例子中，假设空间限制为只包含属性值的合取。（有偏）
  - 这一限制，导致假设空间不能够表示最简单的析取形式的目标概念。

归纳偏置（3）

- 无偏的学习器
  - 为了保证目标概念在假设空间中，需要提供一个假设空间，它能表达所有的可教授概念。换言之，它能表达实例集X的所有子集。
- 问题：为什么2.3节中合取假设空间只能表示973个假设？

归纳偏置（4）

- EnjoySport的无偏形式
  - 带来的问题：概念学习算法无法从训练样例中泛化。
  - 要想获得单个目标概念，就必须提供X中所有实例作为训练样例
  - 使用2.6.3节讨论的部分学习的无效

归纳偏置（5）

- 无偏学习的无用性
  - 归纳学习的一个基本属性：学习器如果不对目标概念的形式做预先的假定，它从根本上无法对未见实例进行分类
  - 归纳学习需要的预先假定，称为归纳偏置

归纳偏置（6）

- 归纳偏置的精确定义
  - $(D_c \wedge x_i) \Rightarrow L(x_i, D_c)$
  - 需要在$D_c \wedge x_i$上附加怎样的前提，以使$L(x_i, D_c)$ 能够演绎派生。
  - L的归纳偏置定义为前提集合B，使所有的新实例满足：

    $(B \wedge D_c \wedge x_i) \Rightarrow L(x_i, D_c)$
  - 考虑对于实例集合X的概念学习算法L。令c为X上定义的任一概念，并令D_c为c的任意训练样例集合，$L(x_i, D_c)$ 表示经过D_c训练后L赋予实例x_i的分类。L的归纳偏置是最小断言集合B，它使任意目标概念c和相应的训练样例D_c满足：

    $\forall x_i \in X[(B \wedge D_c \wedge x_i) \Rightarrow L(x_i, D_c)]$

归纳偏置（6）

- 候选消除算法的归纳偏置
  - $\{c \in H\}$
- 3个有偏程度不同的归纳学习算法
  - 机械式
  - 候选消除算法
  - Find-S
- 一种算法的有偏性越强，它的归纳能力越强，可以分类更多的未见实例。
- 某些归纳偏置隐含在学习器中，有些表示为断言集合，可由学习器操作。

小结

- 主要内容
  - 概念学习可看作搜索预定义潜在假设空间的过程
  - 假设的一般到特殊偏序结构可以定义在任何概念学习问题中，这种结构便于假设空间的搜索
  - Find-S算法使用一般到特殊序，在偏序结构的一个分支上执行一般到特殊搜索，寻找一个与样例一致的最特殊假设
  - 候选消除算法利用一般到特殊序，通过渐进地计算极大特殊假设集合和极大一般假设集合发现变型空间
  - 候选消除算法缺少健壮性，第10章描述了几种基于一般到特殊序关系的概念学习算法，它们能够处理有噪声的数据和目标概念无法在假设空间中表示的情况
  - 归纳学习算法隐含了归纳偏置，候选消除算法的偏置是：目标概念可以在假设空间中找到。输出的假设和对新实例的分类可由归纳偏置和训练样例演绎推出

补充读物

- Bruner et al.1957和Hunt & Hovland1963研究了概念学习以及一般到特殊的偏序
- Winston1970的博士论文将概念学习看作是包含泛化和特殊化操作的搜索过程
- Simon & Lea1973将学习的过程看作是在假设空间中搜索的过程
- Mitchell1977, 1982提出变型空间和候选消除算法
- Haussler1988证明，一般边界的大小随训练样例的数目成指数增长
- Mitchell1979扩展了候选消除算法，以处理可预见的有限数量的误分类样例
- Sebag1994, 1996展示了一种被称为析取变型空间的方法来从有噪声数据中学习析取概念
- ...

机器学习

第12章 归纳和分析学习的结合

概述

- 纯粹的归纳学习方法通过在训练样例中寻找经验化的规律来形成一般假设
- 纯粹的分析方法使用先验知识演绎推导一般假设
- 本章考虑将归纳和分析的机制结合起来的方法，并获得两者的优点：有先验知识时获得更高的泛化精度和依赖训练数据克服先验知识的不足
- 所得到的结合的方法比纯粹的归纳方法和分析方法的性能都要高

动机

- 归纳学习寻找拟合训练数据的一般假设，分析学习寻找拟合先验知识的一般假设，同时使它覆盖训练数据
- 归纳方法和分析方法对假设的论证方法有根本区别，因此优缺点互为补充，将它们结合起来有可能得到更强有力的学习方法
- 纯粹的分析学习方法的优缺点
  - 优点：可用先验知识从较少的数据中更精确地泛化以引导学习
  - 缺点：当先验知识不足或不正确时，可能产生误导
- 纯粹的归纳学习方法的优缺点
  - 优点：不需要显示的先验知识，主要基于训练数据学习规律
  - 缺点：训练数据不足时，会失败，会被其中隐式的归纳偏置所误导

表12-1 纯粹的分析学习和纯粹的归纳学习的比较

	归纳学习	分析学习
目标	拟合数据的假设	拟合领域理论的假设
论证	统计推理	演绎推理
优点	需要很少先验知识	从稀少的数据中学习
缺陷	稀少的数据，不正确的偏置	不完美的领域理论

动机（2）

- 图12-1
  - 概述了学习问题的分布范围，它随着可获得的先验知识和训练数据不同而变化
  - 在一个极端，有大量的训练数据，但没有先验知识
  - 在另一个极端，有很强的先验知识，但训练数据很少
  - 多数实际学习问题位于这两个极端之间，通常可以从近似的先验知识开始
- 本章考虑的问题是：
  - 什么样的算法，使用近似的先验知识结合可用数据来形成一般的假设

动机（3）

- 即使使用最纯粹的归纳学习算法，仍有机会基于特定学习任务的先验知识来选择设计方案
- 通常设计者将领域特定的知识嵌入到学习算法中，但我们感兴趣的是一个系统能将先验知识和训练数据作为显示的输入给学习器
- 概括而言，我们感兴趣的是领域无关算法，这种算法使用显示输入的领域相关的知识，这种算法具备以下的属性：
  - 如果没有领域理论，它至少能像纯粹的归纳方法一样有效学习
  - 如果没有完美的领域理论，它至少能像纯粹的分析方法一样有效学习
  - 如果领域理论和训练数据都不完美，它应能结合两者的长处，比单纯的归纳或分析方法的性能要好
  - 它应能处理训练数据中未知程度的差错
  - 它应能处理领域理论中未知程度的差错
- 这里列出的期望目标很难达到，目前没有算法能以一般化的方式满足所有这些约束

学习的归纳-分析途径

- 本章考虑的学习问题
  - 已知
    - 一个训练样例集合D，可能包含差错
    - 一个领域理论B，可能包含差错
    - 候选假设的空间H
  - 求解
    - 一个最好地拟合训练样例和领域理论的假设
- 最好地拟合训练样例和领域理论的确切定义

$$\arg\min_{h \in H} k_D error_D(h) + k_B error_B(h)$$

学习的归纳-分析途径（2）

- 确定先验知识和数据权值的一种解决方法是使用贝叶斯观点
  - 贝叶斯定律描述了怎样计算给定训练数据D时假设h的后验概率
  - 贝叶斯定律基于观察到的数据D以及先验知识计算后验概率，以P(h), P(D)和P(D|h)的形式表示
  - 我们可以把P(h), P(D)和P(D|h)看作是某种形式的背景知识
  - 贝叶斯理论可看作一种为领域理论加权的方法，它与观察到的数据D一起，赋予h的后验概率为P(h|D)
  - 贝叶斯公式提供了为先验知识和观察到数据的贡献加权的方法
- 但是，贝叶斯公式隐含假定了关于P(h), P(D), P(D|h)概率分布的完美知识
- 贝叶斯公式没有提供将这些近似已知的概率分布与观察数据结合起来的方法

假设空间搜索

- 大多数学习任务可以刻画为假设空间上的搜索任务,而决定这个搜索任务的4个参数是：
  - 假设空间H
  - 搜索的初始假设h0
  - 定义单个搜索步的搜索算子集合O
  - 指定搜索目标的判据G
- 本章探索了3种方法，它们用先验知识来改变纯归纳方法执行的搜索
  - 使用先验知识推导出搜索起步的初始假设：Kbann
  - 使用先验知识来改变假设空间搜索的目标：Ebnn
  - 使用先验知识改变可用的搜索步：Focl

使用先验知识得到的初始假设

- KBANN技术：一种使用先验知识的方法是将假设初始化为完美拟合领域理论，然后按照需要归纳地精化初始假设以拟合训练数据
- 这种技术的动机是：如果领域理论是正确的，初始假设将正确分类所有训练样例，而无需再修正；如果初始假设不能完美地分类训练样例，那么它需要被归纳精华，以改进它在训练样例上的拟合度
- 在纯粹归纳的反向传播算法中，权值一般被初始化为小的随机值，KBANN的含义是：即使领域理论是近似正确的，将网络初始化为拟合领域理论，比初始化为随机值有更好的近似开端

KBANN算法

- KBANN假定领域理论用一组命题形式的非递归的Horn子句来表示，输入和输出如下：
  - 已知：
    - 一组训练样例
    - 由非递归命题型Horn子句组成的领域理论
  - 求解：
    - 一个拟合训练样例的被领域理论偏置的人工神经网络
- KBANN算法包含两个阶段
  - 创建一个完美拟合领域理论的人工神经网络
  - 使用反向传播算法来精化初始网络以拟合训练样例

表12-2 KBANN算法

KBANN(Domain_Theory, Training_Examples)

Domain_Theory：非递归命题型Horn子句集

Training_Examples：目标函数的<input, output>对的集合

- 分析步：创建一个等价于领域理论的初始网络
  - 对每个实例属性创建一个网络输入
  - 对Domain_Theory的每个Horn子句，创建如下的网络单元
    - 连接此单元的输入到此子句的先行词测试的属性
    - 对子句的每个非负先行词，赋予权值W给对应的sigmoid单元输入
    - 对子句的每个负先行词，赋予权值-W给对应的sigmoid单元输入
    - 设置此单元的阈值w0为-(n-0.5)W，其中n为子句的非负先行词的数目
  - 在网络单元之间增加附加的连接，连接深度为i的每个网络单元到深度为i+1的所有网络单元的输入层上，赋予这些附加的连接为接近0的随机权值
- 归纳步：精化此初始网络
  - 应用反向传播算法来调整初始网络权值以拟合Training_Examples

举例

- 表12-3 Cup学习任务
  - 领域理论
  - 训练样例
- 在KBANN算法的第一步，构建一个与领域理论一致的初始网络，见图12-2
  - 对领域理论中每个Horn子句建立一个sigmoid单元
  - 对该Horn子句的每个先行词，建立其对应的Sigmoid单元作为输入
  - 对于每个对应于非负先行词的输入，权值被设置为某正常量W，对每个对应于负先行词的输入，权值为-W
  - 单元的阈值权w0设为-(n-0.5)W，其中n为非负先行词的数目
  - 附加许多输入到每个阈值单元，它们的权值设置为近似0，从而允许网络能够学习到超出领域理论的依赖关系
- 在KBANN算法的第二步，使用训练样例和反向传播算法来精化网络权值
- 图12-3在归纳步发现了全新的依赖关系

KBANN算法说明

- KBANN的好处和局限
  - 好处：在给定近似正确领域理论时，能够比反向传播有更高的泛化精度，特别是在训练数据稀少时
  - 局限：只能使用命题领域理论，如果给予很不精确的领域理论，KBANN也可能被误导，从而其泛化精度变得低于反向传播

使用先验知识改变搜索目标

- 将先验知识合并到梯度下降中需最小化的误差判据，这样网络需要拟合的是训练数据和领域理论的组合函数
- TangentProp算法
  - TangentProp算法接受的领域知识被表示为对应于其输入变换的目标函数的导数
  - 例如，对每个实例xi描述为一个实数，那么每个训练样例的形式可能是<xi, f(xi), $\frac{\partial f(x)}{\partial x}\Big|_{x_i}$>
  - 图12-5，基于3个训练样例学习目标函数f，通过拟合训练值f(xi)的同时拟合相应的导数，学习器能够实现更好的泛化
  - 概括而言，包含训练导数的效果是为了克服反向传播算法中的归纳偏置，将其替换为所希望的导数的显示输入信息
  - ？？？P248-P249

TangentProp举例

- Simard et al.提供了TangentProp的泛化精度与纯归纳反向传播之间的比较结果
  - 针对任务是为单个数字0到9的图像做标注
  - 给予TangentProp的先验知识是：数字的分类不因图像的水平和垂直平移而改变
  - 表12-4，显示TangentProp的泛化精度高于纯反向传播算法

TangentProp的说明

- TangentProp使用的先验知识形式为目标函数对应其输入变换的所希望的导数
- TangentProp通过使一个指标函数最小化来结合先验知识和观察到的训练数据，这个指标函数同时度量了网络对应训练样例值的误差和网络对应于导数的误差
- μ值决定了网络在中个误差中拟合这两部分的程度，它由设计者选择
- TangentProp的不足：对于先验知识中的错误健壮性不强，而且不能预先知道训练导数中的错误出现程度，因而不能很好地选择常量μ以确定拟合训练值和训练导数的相对重要程度

TangentProp的说明（2）

- TangentProp和反向传播的搜索方法比较
  - TangentProp通过改变梯度下降最小化的指标函数来影响假设搜索，相当于改变了搜索目标
  - 如果训练样例和先验知识都正确，并且目标函数可用ANN精确表示，那么满足TangentProp指标的权向量集合将为满足反向传播指标的权向量集合的子集，一些不正确的假设会被TangentProp剔除掉
- 对目标函数的训练导数拟合的另一种方法是，简单地将观察到的训练样例附近的附加训练样例综合起来，使用已知的训练导数来估计这些附近的实例的训练值

EBNN算法

- EBNN是基于解释的神经网络，它用两种方式改进了TangentProp算法
  - 它不依靠用户提供训练导数，而是对每个训练样例自行计算训练导数，计算方法是通过用一套给定的领域理论来解释每个训练样例
  - 涉及了如何确定学习过程中归纳和分析部分相对重要程度的问题，μ的值是对每个训练样例独立选择的，它基于一个启发式规则，考虑领域理论能否精确预测特定样例的训练值
    - 因此，对于那些能由领域理论正确解释的训练样例，学习的分析成分被强化，而对不能正确解释的样例，分析成分被弱化

EBNN算法（2）

- EBNN的输入包括：
  - 形式为<xi, f(xi)>的一组训练样例，不包含训练导数
  - 一组领域理论，表示为一组预先训练过的神经网络，而不是KBANN采用的Horn子句
- EBNN的输出：一个能逼近目标函数f的新的神经网络，学习到的网络能够拟合训练样例以及从领域理论中抽取的f的训练导数
- 图12-7，图的上面部分显示的是目标函数Cup的领域理论，每一方块表示领域理论中一个神经网络，图的下面是要学习的新神经网络，称为目标网络

EBNN算法（3）

- EBNN通过执行TangentProp算法来学习目标网络
  - EBNN把接收到的输入训练值<xi, f(xi)>和从领域理论中计算出的导数提供给TangentProp
- EBNN计算训练导数的方法
  - ？？？

EBNN算法的说明

- 概括地说，EBNN算法使用的领域理论被表示为一组预先学习到的神经网络，然后领域理论与训练样例一起训练其输出假设
- 对每个训练样例，EBNN使用其领域理论来解释它，然后从此解释中抽取训练导数
- 对实例的每个属性计算出一个训练导数，以描述按照领域理论，目标函数值是怎样被属性值的微小变化影响的
- Prolog-EBG与EBNN的区别
  - EBNN能够处理不完美的领域知识，而Prolog-EBG不能
  - Prolog-EBG学习到逐渐增长的Horn子句集，而EBNN学习到固定大小的神经网络

使用先验知识来扩展搜索算子

- FOCL是纯归纳算法FOIL的一个扩展，它们的区别在于：搜索单个Horn子句的一般到特殊过程中候选假设生成的方法
  - FOIL生成每个候选特化式是通过加入一个新文字到子句前件中得到的
  - FOCL使用同样的方法产生候选特化式，但还基于领域理论生成了附加的特化式
- 操作型文字和非操作型文字
  - 操作型文字：当一个文字可被用于描述一个输出假设
  - 非操作型文字：只出现在领域理论中作为中间特征但不是实例的原子属性的文字

FOCL算法

- FOCL使用下面两种算子扩展当前假设h
  - 对不是h一部分的每个操作型文字，创建h的一个特化式，方法是加入文字到前件中
  - 按照领域理论，创建一个操作型的且是目标概念的逻辑充分条件，将这组文字加入到h的当前前件中，最后修剪h的前件，移去对于训练数据不需要的文字，具体过程如下：
    - 首先选择一条领域理论子句，它的头部匹配目标概念，如果有多个这样的子句，选择其中子句体关于训练样例有最高信息增益的
    - 所选子句的前件形成了目标概念的一个逻辑充分条件，在这些充分条件中，再次使用领域理论，每个非操作型文字被替换掉，将子句前件代入到子句后件中
    - 这个展开的过程持续到充分条件被表述为操作型文字，如果有多个可选的展开，那么用贪婪法选择有最大信息增益的一个
    - 最后，修剪充分条件。对表达式中的每个文字，除非文字的移除会降低训练样例上的分类精度，否则它被移去
- 上面的讨论很容易扩展到一阶表示中，类似第10章

FOCL的说明

- 图12-9比较了FOCL和纯归纳的FOIL执行的假设空间搜索
  - FOCL中领域理论推举的特化式对应FOIL搜索中的一个宏步，其中多个文字在一步中被加入
- FOCL使用语法生成候选特化式的同时，还在搜索中每一步使用领域理论驱动生成候选特化式
- 领域理论的使用方式是使学习器偏置：优先选择最相似于领域理论涵蕴的操作型的逻辑充分条件的Horn子句
- 结合分析和归纳的方法，还没有一个在大范围的问题领域中被彻底测试或证明，仍是一个很活跃的研究领域

小结

- 结合归纳和分析学习的方法可以获得两者（归纳和分析）的优点，减小样本复杂度，并且否决不正确的先验知识
- 结合的方法通过领域理论影响假设空间的搜索来实现，主要有：1）影响初始假设；2）影响当前假设的搜索算子集合；3）影响搜索目标
- KBANN影响初始假设，使用领域理论建立初始神经网络，然后使用反向传播算法进行精化
- TangentProp使用先验知识被表示为目标函数所希望的导数，并改变了指标函数

小结（2）

- EBNN使用领域理论改变神经网络搜索的假设空间的目标。领域理论由预先学习的神经网络组成，需要的导数信息来自神经网络
- FOCL使用领域理论来扩展搜索中考虑的候选集，结合了FOIL的贪婪的、一般到特殊的搜索策略以及分析方法中的规则链分析推理
- 还有很多算法试图结合归纳和分析学习，比如第6章讨论的学习贝叶斯网的方法提供了另一种途径

机器学习

第8章 基于实例的学习

概述

- 已知一系列的训练样例，许多学习方法为目标函数建立起明确的一般化描述，
- 基于实例的学习方法只是简单地把训练样例存储起来，从这些实例中泛化的工作被推迟到必须分类新的实例时
- 每当学习器遇到一个新的查询实例，它分析这个新实例与以前存储的实例的关系，并据此把一个目标函数值赋给新实例

概述（2）

- 基于实例的学习方法包括：
  - 假定实例可以表示成欧氏空间中的点
    - 最近邻法
    - 局部加权回归法
  - 对实例采用更复杂的符号表示
    - 基于案例的推理
- 基于实例的学习方法有时被称为消极学习法，它把处理工作延迟到必须分类新的实例时
- 这种延迟的学习方法有一个优点：不是在整个实例空间上一次性地估计目标函数，而是针对每个待分类新实例作出局部的和相异的估计

简介

- 基于实例的学习方法的学习过程只是简单地存储已知的训练数据，当遇到新的查询实例时，一系列相似的实例从存储器中取出，用来分类新的查询实例
- 与其他方法相比，基于实例的学习方法的一个关键差异是：可以为不同的待分类查询实例建立不同的目标函数逼近
- 许多技术不建立目标函数在整个实例空间上的逼近，只建立局部逼近，并将其用于与新实例邻近的实例
- 这样做的好处是：有时目标函数很复杂，但具有不太复杂的局部逼近描述

简介（2）

- 基于案例的学习（基于实例的学习的一种）使用复杂的符号表示法来描述实例，也按照这种方式确定邻近实例
- 基于实例的方法的不足：
  - 分类新实例的开销可能很大。
    - 几乎所有的计算都发生在分类时，而不是在第一次遇到训练样例时。如何有效地索引训练样例是一个重要的问题
  - 当从存储器中检索相似的训练样例时，一般考虑实例的所有属性，如果目标概念仅依赖于很多属性中的几个，那么真正最“相似”的实例之间可能相距甚远

简介（3）

- k-近邻算法和它的几个变体
- 局部加权回归法，这是一种建立目标函数的局部逼近的学习方法，被看作k-近邻算法的一般形式
- 径向基函数网络，它为基于实例的学习算法和神经网络学习算法提供了一个有趣的桥梁
- 基于案例的推理，这是一种使用符号表示和基于知识的推理的方法
- 消极学习方法和积极学习方法之间的差异

k-近邻算法

- k-近邻算法是最基本的基于实例的学习方法
- k-近邻算法假定所有的实例对应于n维空间Rn中的点，任意的实例表示为一个特征向量 $<a_1(x), ..., a_n(x)>$
- 根据欧氏距离定义实例的距离。两个实例x_i和x_j的距离$d(x_i,x_j)$定义为

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} \left(a_r(x_i) - a_r(x_j)\right)^2}$$

- 在最近邻学习中，目标函数值可以是离散的也可以是连续的，本节先考虑离散的情况。

k-近邻算法（2）

- 考虑离散目标函数f: $R^n \rightarrow V$，$V=\{v_1,...,v_s\}$
- 表8-1逼近离散值函数f: $R^n \rightarrow V$的k-近邻算法
  - 训练算法
    - 将每个训练样例<x,f(x)>加入到列表training_examples
  - 分类算法
    - 给定一个要分类的查询实例x_q
      - 在training_examples中选出最靠近x_q的k个实例，并用$x_1...x_k$表示
      - 返回 $\hat{f}(x_q) \leftarrow \arg\max_{v \in V} \sum_{i=1}^{k} \delta(v, f(x_i))$
      - 其中 $\delta(a,b) = \begin{cases} 1 & a = b \\ 0 & a \neq b \end{cases}$

k-近邻算法（3）

- 表8-1的算法返回值是对$f(x_q)$的估计，它是距离x_q最近的k个训练样例中最普遍的f值，结果与k的取值相关。
- 图8-1图解了一种简单情况下的k-近邻算法，实例是二维空间中的点，目标函数具有布尔值，1-近邻算法把x_q分类为正例，5-近邻算法把x_q分类为反例
- k-近邻算法不形成关于目标函数f的明确的一般假设，仅在需要时计算每个新查询实例的分类，但依然可以问：k-近邻算法隐含的一般函数是什么？
- 图8-1中右图画出了1-近邻算法在整个实例空间上导致的决策面形状。这种图称为训练样例集合的Voronoi图

k-近邻算法（4）

- 离散的k-近邻算法作简单修改后可用于逼近连续值的目标函数。即计算k个最接近样例的平均值，而不是计算其中的最普遍的值，为逼近f: $R^n \rightarrow R$，计算式如下：

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

距离加权最近邻算法

- 对k-近邻算法的一个改进是对k个近邻的贡献加权，越近的距离赋予越大的权值，比如：

$$\hat{f}(x_q) \leftarrow \arg\max_{v \in V} \sum_{i=1}^{k} w_i \delta(v, f(x_i)) \qquad w_i = \frac{1}{d(x_q, x_i)^2}$$

- 为了处理查询点x_q恰好匹配某个训练样例x_i，从而导致$d(x_q,x_i)^2$为0的情况，令这种情况下的$\hat{f}(x_q)$等于$f(x_i)$，如果有多个这样的训练样例，我们使用它们占多数的分类
- 也可以用类似的方式对实值目标函数进行距离加权，用下式替代表8-1中的计算式，w_i的定义与前相同

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$$

距离加权最近邻算法（2）

- k-近邻算法的所有变体都只考虑k个近邻用以分类查询点，如果使用按距离加权，那么可以允许所有的训练样例影响x_q的分类，因为非常远的实例的影响很小
- 考虑所有样例的唯一不足是会使分类运行得更慢
- 如果分类一个新实例时，考虑所有的训练样例，我们称为全局法；如果仅考虑靠近的训练样例，称为局部法
- 当式子8.4应用于全局法时，称为Shepard法

对k-近邻算法的说明

- 距离加权的k-近邻算法对训练数据中的噪声有很好的健壮性，通过取k个近邻的加权平均，可以消除孤立的噪声样例的影响
- k-近邻的归纳偏置是：一个实例的分类x_q与在欧氏空间中它附近的实例的分类相似
- k-近邻方法的一个实践问题：维度灾害
  - 许多学习方法，比如决策树方法，选择部分属性作出判断，而k-近邻方法中实例间的距离是根据实例的所有属性计算的
  - 实例间距离会被大量的不相关属性所支配，可能导致相关属性的值很接近的实例相距很远

对k-近邻算法的说明（2）

- 维度灾害的解决方法：对属性加权，相当于按比例缩放欧氏空间中的坐标轴，缩短对应不太相关的属性的坐标轴，拉长对应更相关属性的坐标轴
- 每个坐标轴应伸展的数量可以通过交叉验证的方法自动决定，具体做法如下：
  - 假定使用因子z_j伸展第j个根坐标轴，选择各个z_j的值，使得学习算法的真实分类错误率最小化
  - 这个真实错误率可以使用交叉验证来估计
  - 可以多次重复这个处理过程，使得加权因子的估计更加准确
- 另一种更强有力的方法是从实例空间中完全消除最不相关的属性，等效于设置某个缩放因子为0

对k-近邻算法的说明（3）

- k-近邻算法的另外一个实践问题：如何建立高效的索引。
- k-近邻算法推迟所有的处理，直到接收到一个新的查询，所以处理每个新查询可能需要大量的计算
- 已经开发了很多对存储的训练样例进行索引的方法，以便能高效地确定最近邻
- kd-tree把实例存储在树的叶结点内，邻近的实例存储在同一个或附近的节点内，通过测试新查询x_q的选定属性，树的内部节点把查询x_q排列到相关的叶结点

术语解释

- 来自统计模式识别领域的术语
  - 回归：逼近一个实数值的目标函数
  - 残差：逼近目标函数时的误差 $\hat{f}(x) - f(x)$
  - 核函数：一个距离函数，用来决定每个训练样例的权值，就是使$w_i=K(d(x_i,x_q))$的函数K

局部加权回归

- 前面描述的最近邻方法可以被看作在单一的查询点$x=x_q$上逼近目标函数f(x)
- 局部加权回归是上面方法的推广，它在环绕x_q的局部区域内为目标函数f建立明确的逼近
- 局部加权回归使用附近的或距离加权的训练样例来形成对f的局部逼近
- 例如，使用线性函数、二次函数、多层神经网络在环绕x_q的邻域内逼近目标函数
- 局部加权回归的名称解释
  - 局部：目标函数的逼近仅仅根据查询点附近的数据
  - 加权：每个训练样例的贡献由它与查询点间的距离加权得到
  - 回归：表示逼近实数值函数的问题

局部加权回归（2）

- 给定一个新的查询实例x_q，局部加权回归的一般方法是：
  - 建立一个逼近$\hat{f}$，使$\hat{f}$拟合环绕x_q的邻域内的训练样例
  - 用$\hat{f}$计算$\hat{f}(x_q)$的值
  - 删除$\hat{f}$的描述

局部加权线性回归

- 使用如下形式的线性函数来逼近x_q邻域的目标函数f

$$\hat{f}(x) = w_0 + w_1 a_1(x) + \ldots + w_n a_n(x)$$

- 第4章我们讨论了梯度下降方法，在拟合以上形式的线性函数到给定的训练集合时，它被用来找到使误差最小化的系数w0...wn，当时我们感兴趣的是目标函数的全局逼近，即

$$E = \frac{1}{2}\sum_{x \in D}(f(x) - \hat{f}(x))^2$$

得到的梯度下降训练法则是

$$\Delta w_j = \eta \sum_{x \in D}(f(x) - \hat{f}(x)) a_j(x)$$

局部加权线性回归（2）

- 三种重新定义误差准则E，以着重于拟合局部训练样例，记为$E(x_q)$
  - 只对在k个近邻上的误差平方最小化

$$E_1(x_q) = \frac{1}{2} \sum_{x \in x_q \text{的} k \text{个近邻}} (f(x) - \hat{f}(x))^2$$

  - 使整个训练样例集合D上的误差平方最小化，但对每个训练样例加权，权值为关于相距x_q距离的某个递减函数K

$$E_2(x_q) = \frac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2 K(d(x_q, x))$$

  - 综合1和2

$$E_2(x_q) = \frac{1}{2} \sum_{x \in x_q \text{的} k \text{个近邻}} (f(x) - \hat{f}(x))^2 K(d(x_q, x))$$

局部加权线性回归（3）

- 准则2或许最令人满意，但所需的计算量随着训练样例数量线性增长
- 准则3很好地近似了准则2，并且具有如下优点：计算开销独立于训练样例总数，仅依赖于最近邻数k
- 对应准则3的梯度下降法则是

$$\Delta w_i = \eta \sum_{x \in x_q \text{的} k \text{个近邻}} K(d(x_q, x))(f(x) - \hat{f}(x))a_j(x)$$

局部加权回归的说明

- 大多数情况下，通过一个常量、线性函数或二次函数来局部逼近目标函数，更复杂的函数形式不太常见，原因是：
  - 对每个查询实例用更复杂的函数来拟合，其代价十分高昂
  - 在足够小的实例空间子域上，使用这些简单的近似已能很好地模拟目标函数

径向基函数

- 径向基函数是另一种实现函数逼近的方法，它与距离加权回归和人工神经网络都有着紧密联系
- 待学习的假设是一个以下形式的函数

$$\hat{f}(x) = w_0 + \sum_{u=1}^{k} w_u K_u(d(x_u, x))$$

x_u是X中一个实例，核函数$K_u(d(x_u,x))$被定义为随距离$d(x_u,x)$的增加而减小，k是用户提供的常量，用来指定要包含的核函数的数量

- 尽管$\hat{f}(x)$是对f(x)的全局逼近，但来自每个$K_u(d(x_u,x))$项的贡献被局部化到点x_u附近的区域
- 一种很常见的做法是选择每个核函数Ku(d(xu,x))为高斯函数（表示正态分布的函数）

$$K_u(d(x_u, x)) = e^{-\frac{1}{2\sigma_u^2} d^2(x_u, x)}$$

径向基函数（2）

- Hartman et al.1990指出，式子8.8的函数形式能够以任意小的误差逼近任何函数，只要以上高斯核的数量足够大，并且可以分别指定每个核的宽度
- 图8-2径向基函数网络（RBF），式子8.8给出的函数可以看作是描述了一个两层的网络，第一层计算不同的$K_u(d(x_u,x))$，第二层计算第一层单元值的线性组合
- 给定了目标函数的训练样例集合，一般分两个阶段来训练RBF网络
  - 决定隐藏单元的数量k，并通过定义核函数中心点和方差来定义每个隐藏单元
  - 使用式子8.5给出的全局误差准则来训练权值w_u，使网络拟合训练数据程度最大化

径向基函数（3）

- 已经提出了几种方法来选取适当的隐藏单元或核函数的数量
  - 为每一个训练样例<x_i, $f(x_i)$>分配一个高斯核函数，中心点设为x_i，所有高斯函数的宽度可被赋予同样的值
    - RBF网络学习目标函数的全局逼近，其中每个训练样例<x_i,$f(x_i)$>都只在x_i的邻域内影响$\hat{f}(x_i)$的值
    - 这种核函数的一个优点是允许RBF网络精确地拟合训练数据
    - 对于任意m个训练样例集合，合并m个高斯核函数的权值$w_0 \ldots w_m$，可以被设置为使得对于每个训练样例<x_i,$f(x_i)$>都满足$\hat{f}(x_i) = f(x_i)$

径向基函数（4）

- 选取一组数量少于训练样例数量的核函数，这种方法更有效，特别是训练样例数量巨大时
  - 核函数分布在整个实例空间X上，它们中心之间有均匀的间隔
  - 或者也可以非均匀地分布核函数中心，特别是在实例本身在X上非均匀分布的时候
    - 可以随机选取训练样例的一个子集作为核函数的重心，从而对实例的基准分布进行采样
    - 或者可以标识出实例的原始聚类，然后以每个聚类为中心加入一个核函数
      - » 把训练实例拟合到混合高斯，6.12.1节讨论的EM算法提供了一种从k个高斯函数的混合中选择均值，以最佳拟合观察到实例的方法

径向基函数（5）

- 总而言之，用多个局部核函数的线性组合表示的径向基函数网络提供了一种目标函数的全局逼近
- 仅当输入落入某个核函数的中心和宽度所定义的区域时，这个核函数的值才是不可忽略的
- RBF网络可以被看作目标函数的多个局部逼近的平滑线性组合
- RBF网络的一个优点是，与反向传播算法训练的前馈网络相比，它的训练更加高效，这是因为RBF网络的输入层和输出层可以被分别训练

基于案例的推理

- k-近邻算法和局部加权回归具有三个共同的关键特性：
  - 消极学习方法
  - 通过分析相似的实例来分类新的查询实例，而忽略与查询极其不同的实例
  - 实例表示为n维欧氏空间中的实数点
- 基于案例的推理（CBR）满足前2个原则，但不满足第3个
- CBR使用更丰富的符号描述来表示实例，用来检索实例的方法也更加复杂

基于案例的推理（2）

- CBR已被用于解决很多问题
  - 根据数据库中存储的以前的设计图纸，来进行机械设备的总体设计
  - 根据以前的裁决来对新的法律案件进行推理
  - 通过对以前的相似问题的解决方案的复用或合并，解决规划和调度问题

基于案例的推理（3）

- 一个例子：CADET系统采用基于案例的推理来辅助简单机械设备的总体设计（图8-3）
  - 条件
    - 使用一个数据库，其中包含大约75个以前的设计或设计片断
    - 内存中每一个实例是通过它的结构和定性的功能来表示的，新的设计问题通过所要求的功能和结构来表示
  - 方法
    - 给定新设计问题的功能说明，CADET从它的案例库中搜索存储的案例，使它的功能描述和新设计问题相匹配
      - 如果发现了一个精确的匹配，表明某个存储案例精确地实现了所要求的功能，那么可以返回这个案例作为新设计问题的建议方案
      - 否则，CADET可能找到匹配所需功能的不同子图的案例
        - » 在两个功能图间搜索同构子图，以发现一个案例的某部分，使它匹配更多的案例
        - » 加工原始的功能说明图，产生等价的子图以匹配更多的案例

基于案例的推理（4）

- 通过检索匹配不同子图的多个案例，有时可以拼接得到整个设计
- 但是，从多个检索到的案例产生最终方案的过程可能很复杂
  - 为了合并存储案例中检索到的部分，可能需要从头设计系统的各个部分，也可能需要回溯以前的设计子目标，从而丢弃前面检索到的案例
- CADET合并和自适应已检索到案例并形成最终设计的能力有限，它主要依赖用户来做自适应阶段的处理

基于案例的推理（5）

- CADET的问题框架
  - 在CADET中每个存储的训练样例描绘了一个功能图以及实现该功能的结构
  - 实例空间定义为所有功能图的空间，目标函数f映射到实现这些功能的结构
  - 每个存储训练样例<x,f(x)>是一个序偶，描述某个功能图x和实现x的结构f(x)
  - 系统通过学习训练样例，以输出满足功能图查询输入x_q的结构$f(x_q)$

基于案例的推理（6）

- CADET系统区别于k-近邻方法的一般特征
  - 实例（或称案例）可以用丰富的符号描述表示，因此可能需要不同于欧氏距离的相似性度量
  - 检索到的多个案例可以合并形成新问题的解决方案，合并案例的过程与k-近邻方法不同，依赖于知识推理而不是统计方法
  - 案例检索、基于知识的推理、问题求解是紧密耦合在一起的
- 概括而言，基于案例的推理是一种基于实例的学习方法，在这个方法中
  - 实例可以是丰富的关系的描述
  - 案例检索和合并过程可能依赖于知识推理和搜索密集的问题求解方法
- 一个研究课题：改进索引案例的方法
  - 句法相似度量仅能近似地指出特定案例与特定问题的相关度，而不能捕捉其他难点，比如多个设计片断的不兼容性
  - 发现这些难点后，可以回溯，并且可用来改进相似性度量

对消极学习和积极学习的评价

- 本章考虑了三种消极学习方法：k-近邻、局部加权回归、基于案例的推理
- 本章考虑了一种积极学习方法：学习径向基函数网络的方法
- 消极方法和积极方法的差异：
  - 计算时间的差异
    - 消极算法在训练时需要较少的计算，但在预测新查询的目标值时需要更多的计算时间
  - 对新查询的分类的差异（归纳偏置的差异）
    - 消极方法在决定如何从训练数据D中泛化时考虑查询实例x_q
    - 积极方法在见到x_q之前，就完成了泛化
  - 核心观点：消极学习可以通过很多局部逼近的组合表示目标函数，积极学习必须在训练时提交单个的全局逼近

对消极学习和积极学习的评价（2）

- 使用多个局部逼近的积极方法，可以产生与消极方法的局部逼近同样的效果吗？径向基函数网络是对这个目标的尝试
- RBF学习方法是在训练时提交目标函数全局逼近的积极方法，然而，一个RBF网络把这个全局函数表示为多个目标局部核函数的线性组合
- RBF学习方法创建的局部逼近不能达到像消极学习方法中那样特别针对查询点

小结

- 基于实例的学习方法推迟处理训练样例，直到必须分类一个新查询实例时才进行。它们不必形成一个明确的假设来定义整个实例空间上的完整目标函数，而是对每个查询实例形成一个不同的目标函数局部逼近
- 基于实例的方法的优点：通过一系列不太复杂的局部逼近来模拟复杂目标函数，而且不会损失训练样例中蕴含的任何信息
- 基于实例的方法的主要的实践问题：
  - 分类新实例的效率
  - 难以选择用来检索相关实例的合适的距离度量
  - 无关特征对距离度量的负作用

小结（2）

- k-近邻假定实例对应于n维欧氏空间中的点，一个新查询的目标函数值是根据k个与其最近的训练样例的值估计得到
- 局部加权回归法是k-近邻方法的推广，为每个查询实例建立一个明确的目标函数的局部逼近，逼近方法可以基于常数、线性函数、二次函数等这类简单的函数形式，也可以基于核函数
- 径向基函数网络是一类由空间局部化核函数构成的人工神经网络，可被看作是基于实例的方法和神经网络方法的结合
- 基于案例的推理使用复杂的逻辑描述而不是欧氏空间中的点来表示实例。给定实例的符号描述，已经提出了许多方法将训练样例映射成新实例的目标函数值

补充读物

- k-近邻算法
  - Cover & Hart1967提出了早期的理论结果
  - Duda & Hart1973提供了一个很好的概述
  - Bishop1995讨论了k-近邻算法以及它与概率密度估计的关系
  - Atkeson et al.1997对局部加权回归方法给出了一个非常好的纵览
  - Atkeson et al.1997b给出了这些方法在机器人控制方面的应用
- 径向基函数
  - Bishop1995提供了一个全面讨论
  - Powell1987和Poggio & Girosi1990给出了其他论述
- 基于案例的推理
  - Kolodner1993提供了基于案例的推理的一般介绍

机器学习

第7章 计算学习理论

概述

- 本章从理论上刻画了若干类型的机器学习问题中的困难和若干类型的机器学习算法的能力
- 这个理论要回答的问题是：
  - 在什么样的条件下成功的学习是可能的？
  - 在什么条件下某个特定的学习算法可保证成功运行？
- 这里考虑两种框架：
  - 可能近似正确（PAC）
    - 确定了若干假设类别，判断它们能否从多项式数量的训练样例中学习得到
    - 定义了一个对假设空间复杂度的自然度量，由它可以界定归纳学习所需的训练样例数目
  - 出错界限框架
    - 考查了一个学习器在确定正确假设前可能产生的训练错误数量

简介

- 机器学习理论的一些问题：
  - 是否可能独立于学习算法确定学习问题中固有的难度？
  - 能否知道为保证成功的学习有多少训练样例是必要的或充足的？
  - 如果学习器被允许向施教者提出查询，而不是观察训练集的随机样本，会对所需样例数目有怎样的影响？
  - 能否刻画出学习器在学到目标函数前会有多少次出错？
  - 能否刻画出一类学习问题中固有的计算复杂度？

简介（2）

- 对所有这些问题的一般回答还未知，但不完整的学习计算理论已经开始出现
- 本章阐述了该理论中的一些关键结论，并提供了在特定问题下一些问题的答案
- 主要讨论在只给定目标函数的训练样例和候选假设空间的条件下，对该未知目标函数的归纳学习问题
- 主要要解决的问题是：需要多少训练样例才足以成功地学习到目标函数以及学习器在达到目标前会出多少次错

简介（3）

- 如果明确了学习问题的如下属性，那么有可能给出前面问题的定量的上下界
  - 学习器所考虑的假设空间的大小和复杂度
  - 目标概念须近似到怎样的精度
  - 学习器输出成功的假设的可能性
  - 训练样例提供给学习器的方式
- 本章不会着重于单独的学习算法，而是在较宽广的学习算法类别中考虑问题：
  - 样本复杂度：学习器要收敛到成功假设，需要多少训练样例？
  - 计算复杂度：学习器要收敛到成功假设，需要多大的计算量？
  - 出错界限：在成功收敛到一个假设前，学习器对训练样例的错误分类有多少次？

简介（4）

- 为了解决这些问题需要许多特殊的条件设定，比如
  - “成功”的学习器的设定
    - 学习器是否输出等于目标概念的假设
    - 只要求输出的假设与目标概念在多数时间内意见一致
    - 学习器通常输出这样的假设
  - 学习器如何获得训练样例
    - 由一个施教者给出
    - 由学习器自己实验获得
    - 按照某过程随机生成

简介（5）

- 7.2节介绍可能近似正确（PAC）学习框架
- 7.3节在PAC框架下，分析几种学习算法的样本复杂度和计算复杂度
- 7.4节介绍了假设空间复杂度的一个重要度量标准，称为VC维，并且将PAC分析扩展到假设空间无限的情况
- 7.5节介绍出错界限模型，并提供了前面章节中几个学习算法出错数量的界限，最后介绍了加权多数算法

可能学习近似正确假设

- 可能近似正确学习模型（PAC）
  - 指定PAC学习模型适用的问题
  - 在此模型下，学习不同类别的目标函数需要多少训练样例和多大的计算量
- 本章的讨论将限制在学习布尔值概念，且训练数据是无噪声的（许多结论可扩展到更一般的情形）

问题框架

- X表示所有实例的集合，C代表学习器要学习的目标概念集合，C中每个目标概念c对应于X的某个子集或一个等效的布尔函数c: $X \rightarrow \{0,1\}$
- 假定实例按照某概率分布D从X中随机产生
- 学习器L在学习目标概念时考虑可能假设的集合H。在观察了一系列关于目标概念c的训练样例后，L必须从H中输出某假设h，它是对c的估计
- 我们通过h在从X中抽取的新实例上的性能来评估L是否成功。新实例与训练数据具有相同的概率分布
- 我们要求L足够一般，以至可以从C中学到任何目标概念而不管训练样例的分布如何，因此，我们会对C中所有可能的目标概念和所有可能的实例分布D进行最差情况的分析

假设的错误率

- 为了描述学习器输出的假设h对真实目标概念的逼近程度，首先要定义假设h对应于目标概念c和实例分布D的真实错误率
- h的真实错误率是应用h到将来按分布D抽取的实例时的期望的错误率
- 定义：假设h的关于目标概念c和分布D的真实错误率为h误分类根据D随机抽取的实例的概率

$$error_D(h) \equiv \Pr_{x \in D}[c(x) \neq h(x)]$$

假设的错误率（2）

- 图7-1：h关于c的错误率是随机选取的实例落入h和c不一致的区间的概率
- 真实错误率紧密地依赖于未知的概率分布D
  - 如果D是一个均匀的概率分布，那么图7-1中假设的错误率为h和c不一致的空间在全部实例空间中的比例
  - 如果D恰好把h和c不一致区间中的实例赋予了很高的概率，相同的h和c将造成更高的错误率
- h关于c的错误率不能直接由学习器观察到，L只能观察到在训练样例上h的性能
- 训练错误率：指代训练样例中被h误分类的样例所占的比例
- 问题：h的观察到的训练错误率对真实错误率产生不正确估计的可能性多大？

PAC可学习性

- 我们的目标是刻画出这样的目标概念，它们能够从合理数量的随机抽取训练样例中通过合理的计算量可靠地学习
- 对可学习性的表述
  - 一种可能的选择：为了学习到使$error_D(h)=0$的假设h，所需的训练样例数
    - 这样的选择不可行：首先要求对X中每个可能的实例都提供训练样例；其次要求训练样例无误导性
  - 可能近似学习：首先只要求学习器输出错误率限定在某常数ε范围内的假设，其次要求对所有的随机抽取样例序列的失败的概率限定在某常数δ范围内
    - 只要求学习器可能学习到一个近似正确的假设

PAC可学习性（2）

- PAC可学习性的定义
  - 考虑定义在长度为n的实例集合X上的一概念类别C，学习器L使用假设空间H。当对所有$c \in C$，X上的分布D，ε和δ满足$0<\varepsilon, \delta<1/2$，学习器L将以至少$1-\delta$输出一假设$h \in H$，使$error_D(h) \leq \varepsilon$，这时称C是使用H的L可PAC学习的，所使用的时间为$1/\varepsilon$，$1/\delta$，n以及size(c)的多项式函数
- 上面定义要求学习器L满足两个条件
  - L必须以任意高的概率（$1-\delta$）输出一个错误率任意低（ε）的假设
  - 学习过程必须是高效的，其时间最多以多项式方式增长
- 上面定义的说明
  - $1/\varepsilon$和$1/\delta$表示了对输出假设要求的强度，n和size(c)表示了实例空间X和概念类别C中固有的复杂度
  - n为X中实例的长度，size(c)为概念c的编码长度

PAC可学习性（3）

- 在实践中，通常更关心所需的训练样例数，
  - 如果L对每个训练样例需要某最小处理时间，那么为了使c是L可PAC学习的，L必须从多项式数量的训练样例中进行学习
  - 实际上，为了显示某目标概念类别C是可PAC学习的，一个典型的途径是证明C中每个目标概念可以从多项式数量的训练样例中学习到，且处理每个样例的时间也是多项式级
- PAC可学习性的一个隐含的条件：对C中每个目标概念c，假设空间H都包含一个以任意小误差接近c的假设

有限假设空间的样本复杂度

- PAC可学习性很大程度上由所需的训练样例数确定
- 随着问题规模的增长所带来的所需训练样例的增长称为该学习问题的样本复杂度
- 我们把样本复杂度的讨论限定于一致学习器（输出完美拟合训练数据的学习器）
- 能够独立于特定算法，推导出任意一致学习器所需训练样例数的界限
- 变型空间：能正确分类训练样例D的所有假设的集合。 $VS_{H,D} = \{h \in H \mid (\forall < x, c(x) > \in D(h(x) = c(x))\}$

有限假设空间的样本复杂度（2）

- 变型空间的重要意义是：每个一致学习器都输出一个属于变型空间的假设
- 因此界定任意一个一致学习器所需的样例数量，只需要界定为保证变型中没有不可接受假设所需的样例数量
- 定义：考虑一假设空间H，目标概念c，实例分布D以及c的一组训练样例S。当$VS_{H,D}$中每个假设h关于c和D错误率小于ε时，变型空间被称为c和D是ε-详尽的。 $(\forall h \in VS_{H,D}) error_D(h) < \varepsilon$

有限假设空间的样本复杂度（3）

- ε-详尽的变型空间表示与训练样例一致的所有假设的真实错误率都小于ε
- 从学习器的角度看，所能知道的只是这些假设能同等地拟合训练数据，它们都有零训练错误率
- 只有知道目标概念的观察者才能确定变型空间是否为ε-详尽的
- 但是，即使不知道确切的目标概念或训练样例抽取的分布，一种概率方法可在给定数目的训练样例之后界定变型空间为ε-详尽的

有限假设空间的样本复杂度（4）

- 定理7.1（变型空间的ε-详尽化）
  - 若假设空间H有限，且D为目标概念c的一系列m>=1个独立随机抽取的样例，那么对于任意0=<ε<=1，变型空间$VS_{H,D}$不是ε-详尽的概率小于或等于：$|H|e^{-\varepsilon m}$
- 证明：
  - 令$h_1,...,h_k$为H中关于c的真实错误率大于ε的所有假设。当且仅当k个假设中至少有一个恰好与所有m个独立随机抽取样例一致时，不能使变型空间ε-详尽化。
  - 任一假设真实错误率大于ε，且与一个随机抽取样例一致的可能性最多为1-ε，因此，该假设与m个独立抽取样例一致的概率最多为$(1-\varepsilon)^m$
  - 由于已知有k个假设错误率大于ε，那么至少有一个与所有m个训练样例都不一致的概率最多为（当$0 \le \varepsilon \le 1$，则$1-\varepsilon \le e^{\varepsilon}$）

$$k(1-\varepsilon)^m \le |H|(1-\varepsilon)^m \le |H|e^{-\varepsilon m}$$

有限假设空间的样本复杂度（5）

- 定理7.1基于训练样例的数目m、允许的错误率ε和H的大小，给出了变型空间不是ε-详尽的概率的上界
- 即它对于使用假设空间H的任意学习器界定了m个训练样例未能将所有“坏”的假设（错误率大于ε的假设）剔除出去的概率
- 利用上面的结论来确定为了减少此“未剔除”概率到一希望程度δ所需的训练样例数
  - 由$|H|e^{-\varepsilon m} \le \delta$
  - 解出m，得到$m \ge \frac{1}{\varepsilon}(\ln|H| + \ln(1/\delta))$

有限假设空间的样本复杂度（6）

- 式子7.2提供了训练样例数目的一般边界，该数目的样例足以在所期望的值δ和ε程度下，使任何一致学习器成功地学习到H中的任意目标概念
- 训练样例的数目m足以保证任意一致假设是可能（可能性为1- δ）近似（错误率为ε）正确的
- m随着1/ε线性增长，随着1/δ和假设空间的规模对数增长
- 上面的界限可能是过高的估计，主要来源于|H|项，它产生于证明过程中在所有可能假设上计算那些不可接受的假设的概率和
- 在7.4节讨论一个更紧凑的边界以及能够覆盖无限大的假设空间的边界

不可知学习和不一致假设

- 如果学习器不假定目标概念可在H中表示，而只简单地寻找具有最小训练错误率的假设，这样的学习器称为不可知学习器
- 式7.2基于的假定是学习器输出一零错误率假设，在更一般的情形下学习器考虑到了有非零训练错误率的假设时，仍能找到一个简单的边界
- 令S代表学习器可观察到的特定训练样例集合，$error_S(h)$表示h的训练错误率，即S中被h误分类的训练样例所占比例
- 令h_{best}表示H中有最小训练错误率的假设，问题是：多少训练样例才足以保证其真实错误率$error_D(h_{best})$不会多于$\varepsilon+error_S(h_{best})$？（上一节问题是这个问题的特例）

不可知学习和不一致假设（2）

- 前面问题的回答使用类似定理7.1的证明方法，这里有必要引入一般的Hoeffding边界
- Hoeffding边界刻画的是某事件的真实概率及其m个独立试验中观察到的频率之间的差异
- Hoeffding边界表明，当训练错误率$error_S(h)$在包含m个随机抽取样例的集合S上测量时，则

$$\Pr[error_D(h) > error_S(h) + \varepsilon] \le e^{-2m\varepsilon^2}$$

- 上式给出了一个概率边界，说明任意选择的假设训练错误率不能代表真实情况，为保证L寻找到的最佳的假设的错误率有以上的边界，我们必须考虑这|H|个假设中任一个有较大错误率的概率

$$\Pr[(\exists h \in H)(error_D(h) > error_S(h) + \varepsilon)] \le |H| e^{-2m\varepsilon^2}$$

不可知学习和不一致假设（3）

- 将上式左边概率称为δ，问多少个训练样例m才足以使δ维持在一定值内，求解得到 $m \geq \frac{1}{2\varepsilon^2}(\ln|H| + \ln(1/\delta))$
- 式7.3是式7.2的一般化情形，适用于当最佳假设可能有非零训练错误率时，学习器仍能选择到最佳假设h∈H的情形。

布尔文字的合取是PAC可学习的

- 我们已经有了一个训练样例数目的边界，表示样本数目为多少时才足以可能近似学习到目标概念，现在用它来确定某些特定概念类别的样本复杂度和PAC可学习性
- 考虑目标概念类C，它由布尔文字的合取表示。布尔文字是任意的布尔变量，或它的否定。问题：C是可PAC学习的吗？
- 若假设空间H定义为n个布尔文字的合取，则假设空间|H|的大小为3^n，得到关于n布尔文字合取学习问题的样本复杂度

$$m \geq \frac{1}{\varepsilon}\left(n \ln 3 + \ln(1/\delta)\right) = \frac{1}{0.1}\left(10 \ln 3 + \ln(1/0.05)\right) = 140$$

布尔文字的合取是PAC可学习的（2）

- 定理7.2：布尔合取式的PAC可学习性
  - 布尔文字合取的类C是用Find-S算法PAC可学习的
- 证明：
  - 式7.4显示了该概念类的样本复杂度是n、$1/\delta$和$1/\varepsilon$的多项式级，而且独立于size(c)。为增量地处理每个训练样例，Find-S算法要求的运算量根据n线性增长，并独立于$1/\delta$、$1/\varepsilon$和size(c)。因此这一概念类别是Find-S算法PAC可学习的。

其他概念类别的PAC可学习性

- 无偏学习器（无归纳偏置）
  - 考虑一无偏概念类C，它包含与X相关的所有可教授概念，X中的实例定义为n个布尔值特征，则有

$$|H| = |C| = 2^{|X|} = 2^{2^n}$$

$$m \geq \frac{1}{\varepsilon}\left(2^n \ln 2 + \ln(1/\delta)\right)$$

  - 无偏的目标概念类在PAC模型下有指数级的样本复杂度

其他概念类别的PAC可学习性（2）

- K项DNF和K-CNF概念
  - 某概念类有多项式级的样本复杂度，但不能够在多项式时间内被学习到
  - 概念类C为k项析取范式（k项DNF）的形式
  - k项DNF：$T_1 \vee ... \vee T_k$，其中每一个T_i为n个布尔属性和它们的否定的合取
  - 假定H=C，则|H|最多为3^{nk}，代入式7.2，得到 $m \geq \frac{1}{\varepsilon}(nk \ln 3 + \ln(1/\delta))$
  - 因此，k项DNF的样本复杂度为1/δ、1/ε、n和k的多项式级
  - 但是计算复杂度不是多项式级，该问题是NP完全问题（等效于其他已知的不能在多项式时间内解决的问题）
  - 因此，虽然k项DNF有多项式级的样本复杂度，它对于使用H=C的学习器没有多项式级的计算复杂度

其他概念类别的PAC可学习性（3）

- 令人吃惊的是，虽然k项DNF不是PAC可学习的，但存在一个更大的概念类是PAC可学习的
- 这个更大的概念类是K-CNF，它有每样例的多项式级时间复杂度，又有多项式级的样本复杂度
- K-CNF：任意长度的合取式$T_1 \wedge ... \wedge T_j$，其中每个$T_i$为最多k个布尔变量的析取
- 容易证明K-CNF包含了K项DNF，因此概念类k项DNF是使用H=K-CNF的一个有效算法可PAC学习的

无限假设空间的样本复杂度

- 式子7.2用|H|刻画样本复杂度有两个缺点：
  - 可能导致非常弱的边界
  - 对于无限假设空间的情形，无法应用
- 本节考虑H的复杂度的另一种度量，称为H的Vapnik-Chervonenkis维度（简称VC维或VC(H)）
- 使用VC维代替|H|也可以得到样本复杂度的边界，基于VC维的样本复杂度比|H|更紧凑，另外还可以刻画无限假设空间的样本复杂度

打散一个实例集合

- VC维衡量假设空间复杂度的方法不是用不同假设的数量|H|，而是用X中能被H彻底区分的不同实例的数量
- S是一个实例集，H中每个h导致S的一个划分，即h将S分割为两个子集$\{x \in S|h(x)=1\}$和$\{x \in S|h(x)=0\}$
- 定义：一实例集S被假设空间H打散，当且仅当对S的每个划分，存在H中的某假设与此划分一致
- 如果一实例集合没有被假设空间打散，那么必然存在某概念可被定义在实例集之上，但不能由假设空间表示
- H的这种打散实例集合的能力是其表示这些实例上定义的目标概念的能力的度量

Vapnik-Chervonenkis维度

- 打散一实例集合的能力与假设空间的归纳偏置紧密相关
- 无偏的假设空间能够打散所有实例组成的集合X
- 直观上，被打散的X的子集越大，H的表示能力越强
- 定义：定义在实例空间X上的假设空间H的Vapnik-Chervonenkis维，是可被H打散的X的最大有限子集的大小
- 如果X的任意有限大的子集可被H打散，则VC(H)=∞

Vapnik-Chervonenkis维度（2）

- 对于任意有限的H，$VC(H) <= \log_2|H|$
- VC维举例
  - 假定实例空间X为实数集合，而且H为实数轴上的区间的集合，问VC(H)是多少？
    - 只要找到能被H打散的X的最大子集，首先包含2个实例的集合能够被H打散，其次包含3个实例的集合不能被H打散，因此VC(H)=2
  - 实例集合S对应x、y平面上的点，令H为此平面内所有线性决策面的集合，问H的VC维是多少？
    - 能够找到3个点组成的集合，被H打散，但无法找到能够被H打散的4个点组成的集合，因此VC(H)=3
    - 更一般地，在r维空间中，线性决策面的VC维为r+1

Vapnik-Chervonenkis维度（3）

- 假定X上每个实例由恰好3个布尔文字的合取表示，而且假定H中每个假设由至多3个布尔文字描述，问VC(H)是多少？
  - 找到下面3个实例的集合
    - $instance_1$: 100
    - $instance_2$: 010
    - $instance_3$: 001
  - 这三个实例的集合可被H打散，可对如下任意所希望的划分建立一假设：如果该划分要排除$instance_i$，就将文字$\neg l_i$加入到假设中
  - 此讨论很容易扩展到特征数为n的情况，n个布尔文字合取的VC维至少为n
  - 实际就是n，但证明比较困难，需要说明n+1个实例的集合不可能被打散

样本复杂度和VC维

- 使用VC维作为H复杂度的度量，就有可能推导出该问题的另一种解答，类似于式子7.2的边界，即（Blumer el al. 1989）

$$m \geq \frac{1}{\varepsilon}\left(4\log_2(2/\delta) + 8VC(H)\log_2(13/\varepsilon)\right)$$

- 定理7.3：样本复杂度的下界（Ehrenfeucht et al. 1989）
  - 考虑任意概念类C，且VC(C)>=2，任意学习器L，以及任意0<ε<1/8，0<δ<1/100。存在一个分布D以及C中一个目标概念，当L观察到的样例数目小于下式时：

$$\max\left[\frac{1}{\varepsilon}\log(1/\delta), \frac{VC(C)-1}{32\varepsilon}\right]$$

    L将以至少δ的概率输出一假设h，使$error_D(h)>\varepsilon$

样本复杂度和VC维（2）

- 定理7.3说明，若训练样例的数目太少，那么没有学习器能够以PAC模型学习到任意非平凡的C中每个目标概念
- 式子7.7给出了保证充足数量的上界，而定理7.3给出了下界

神经网络的VC维

- 本节给出一般性的结论，以计算分层无环网络的VC维。这个VC维可用于界定训练样例的数量，该数达到多大才足以按照希望的ε和δ值近似可能正确地学习一个前馈网络
- 考虑一个由单元组成的网络G，它形成一个分层有向无环图
  - 分层有向无环图的特点：
    - 节点可划分成层，即所有第l层出来的有向边进入到第l+1层节点
    - 没有有向环，即有向弧形成的回路
  - 这样网络的VC维的界定可以基于其图的结构和构造该图的基本单元的VC维

神经网络的VC维（2）

- 定义一些术语
  - G表示神经网络
  - n是G的输入数目
  - G只有1个输出节点
  - G的每个内部单元Ni最多有r个输入，并实现一布尔函数$c_i:R^r \rightarrow \{0,1\}$，形成函数类C
- 定义C的G-合成：网络G能实现的所有函数的类，即网络G表示的假设空间，表示成C_G

神经网络的VC维（3）

- 定理7.4分层有向无环网络的VC维（Kearns & Vazirani 1994）
  - 令G为一分层有向无环图，有n个输入节点和s≥2个内部节点，每个至少有r个输入，令C为VC维为d的R^r上的概念类，对应于可由每个内部节点s描述的函数集合，令C_G为C的G合成，对应于可由G表示的函数集合，则$VC(C_G)<=2ds\log(es)$

神经网络的VC维（4）

- 假定要考虑的分层有向无环网络中单个节点都是感知器，由于单独的r输入感知器VC维为r+1，代入定理7.4和式子7.7，得到

$$VC(C_G^{perceptrons}) \le 2(r+1)s\log(es)$$

$$m \ge \frac{1}{\varepsilon}(4\log(2/\delta) + 16(r+1)s\log(es)\log(13/\varepsilon))$$

- 上面的结果不能直接应用于反向传播的网络，原因有两个：
  - 此结果应用于感知器网络，而不是sigmoid单元网络
  - 不能处理反向传播中的训练过程

学习的出错界限模型

- 计算学习理论考虑多种不同的问题框架
  - 训练样例的生成方式（被动观察、主动提出查询）
  - 数据中的噪声（有噪声或无噪声）
  - 成功学习的定义（必须学到正确的目标概念还是有一定的可能性和近似性）
  - 学习器所做得假定（实例的分布情况以及是否$C \subseteq H$）
  - 评估学习器的度量标准（训练样例数量、出错数量、计算时间）

学习的出错界限模型（2）

- 机器学习的出错界限模型
  - 学习器的评估标准是它在收敛到正确假设前总的出错数
  - 学习器每接受到一个样例x，先预测目标值c(x)，然后施教者给出正确的目标值
  - 考虑的问题是：在学习器学习到目标概念前，它的预测会有多少次出错
  - 下面讨论中，只考虑学习器确切学到目标概念前出错的次数，确切学到的含义是$\forall x\ h(x)=c(x)$

Find-S算法的出错界限

- Find-S算法的一个简单实现
  - 将h初始化为最特殊假设$l_1 \wedge \neg l_1 \wedge ... \wedge l_n \wedge \neg l_n$
  - 对每个正例x
    - 从h中移去任何不满足x的文字
  - 输出假设h
- 计算一个边界，以描述Find-S在确切学到目标概念c前全部的出错次数
  - Find-S永远不会将一反例错误地划分为正例，因此只需要计算将正例划分为反例的出错次数
  - 遇到第一个正例，初始假设中2n个项半数被删去，对后续的被当前假设误分类的正例，至少有一项从假设中删去
  - 出错总数至多为n+1

Halving算法的出错界限

- 学习器对每个新实例x做出预测的方法是：从当前变型空间的所有假设中取多数票得来
- 将变型空间学习和用多数票来进行后续预测结合起来的算法称为Halving算法
- Halving算法只有在当前变型空间的多数假设不能正确分类新样例时出错，此时变型空间至少可减少到它的一半大小，因此出错界线是$\log_2|H|$
- Halving算法有可能不出任何差错就确切学习到目标概念，因为即使多数票是正确的，算法仍将移去那些不正确、少数票假设
- Halving算法的一个扩展是允许假设以不同的权值进行投票（如贝叶斯最优分类器和后面讨论的加权多数算法）

最优出错界限

- 问题：对于任意概念类C，假定H=C，最优的出错边界是什么？
- 最优出错边界是指在所有可能的学习算法中，最坏情况下出错边界中最小的那一个
- 对任意学习算法A和任意目标概念c，令$M_A(c)$代表A为了确切学到c，在所有可能训练样例序列中出错的最大值
- 对于任意非空概念类C，令$M_A(C)=\max_{c\in C} M_A(c)$
- 定义：C为任意非空概念类，C的最优出错界限定义为Opt(C)是所有可能学习算法A中$M_A(C)$的最小值

$$Opt(C)=\min_{A\in\text{学习算法}} M_A(C)$$

最优出错界限（2）

- 非形式地说，Opt(C)是C中最难的那个目标概念使用最不利的训练样例序列用最好的算法时的出错次数
- Littlestone1987证明了

$$VC(C) \le Opt(C) \le M_{Halving}(C) \le \log_2 |C|$$

加权多数算法

- Halving算法的更一般形式称为加权多数算法
- 加权多数算法通过在一组预测算法中进行加权投票来作出预测，并通过改变每个预测算法的权来学习
- 加权多数算法可以处理不一致的训练数据，因为它不会消除与样例不一致的假设，只是降低其权
- 要计算加权多数算法的出错数量边界，可以用预测算法组中最好的那个算法的出错数量

加权多数算法（2）

- 加权多数算法一开始将每个预测算法赋予权值1，然后考虑训练样例，只要一个预测算法误分类新训练样例，它的权被乘以某个系数β，0<=β<1。
  - 如果β=0，则是Halving算法
  - 如果β>0，则没有一个预测算法会被完全去掉。如果一算法误分类一个样例，那么它的权值变小

加权多数算法（3）

- a_i代表算法池A中第i个预测算法，w_i代表与a_i相关联的权值
- 对所有i，初始化$w_i \leftarrow 1$
- 对每个训练样例<x, c(x)>做：
  - 初始化q_0和q_1为0
  - 对每个预测算法a_i
    - 如果$a_i(x)=0$，那么$q_0 \leftarrow q_0+w_i$
    - 如果$a_i(x)=1$，那么$q_1 \leftarrow q_1+w_i$
  - 如果$q_1>q_0$，那么预测c(x)=1
  - 如果$q_0>q_1$，那么预测c(x)=0
  - 如果$q_0=q_1$，那么对c(x)随机预测为0或1
  - 对每个预测算法a_i
    - 如果$a_i(x)\neq c(x)$，那么$w_i \leftarrow \beta w_i$

加权多数算法（4）

- 定理7.5：加权多数算法的相对误差界限
  - 令D为任意的训练样例序列，令A为任意n个预测算法集合，令k为A中任意算法对样例序列D的出错次数的最小值。那么使用β=1/2的加权多数算法在D上出错次数最多为：$2.4(k+\log_2 n)$
- 证明：
  - 可通过比较最佳预测算法的最终权和所有算法的权之和来证明。令a_j表示A中一算法，并且它出错的次数为最优的k次，则与a_j关联的权w_j将为$(1/2)^k$。A中所有n个算法的权的和$W=\sum_{i=1}^{n} w_i$，W的初始值为n，对加权多数算法的每次出错，W被减小为最多$\frac{3}{4}W$，其原因是加权投票占少数的算法最少拥有整个权W的一半值，而这一部分将被乘以因子1/2。令M代表加权多数算法对训练序列D的总出错次数，那么最终的总权W最多为$n(3/4)^M$
  - 由$\left(\frac{1}{2}\right)^k \le n\left(\frac{3}{4}\right)^M$，得 $M \le \frac{(k+\log_2 n)}{-\log_2(\frac{3}{4})} \le 2.4(k+\log_2 n)$
- 意义：加权多数算法的出错数量不会大于算法池中最佳算法出错数量，加上一个随着算法池大小对数增长的项，再乘以一常数因子

小结

- 可能近似正确模型（PAC）针对的算法从某概念类C中学习目标概念，使用按一个未知但固定的概念分布中随机抽取的训练样例，它要求学习器可能学习到一近似正确的假设，而计算量和训练样例数都只随着1/δ、1/ε、实例长度和目标概念长度的多项式级增长
- 在PAC学习模型的框架下，任何使用有限假设空间H的一致学习器，将以1-δ的概率输出一个误差在ε内的假设，所需的训练样例数m满足

$$m \geq \frac{1}{\varepsilon}\left(\ln(1/\delta) + \ln|H|\right)$$

小结（2）

- 不可知学习考虑更一般的问题：学习器不假定目标概念所在的类别，学习器从训练数据中输出H中有最小误差率的假设。学习保证以概率1-δ从H中最可能的假设中输出错误率小于ε的假设，需要的随机抽取的训练样例数目m满足

$$m \geq \frac{1}{2\varepsilon^2}\left(\ln(1/\delta) + \ln|H|\right)$$

- 学习器考虑的假设空间的复杂度对所需样例的数目影响很大，而衡量假设空间复杂度的一个有用的度量是VC维。VC维是可被H打散的最大实例集的大小

小结（3）

- 在PAC模型下以VC(H)表示的足以导致成功学习的训练样例数目的上界和下界分别是：

$$m \geq \frac{1}{\varepsilon}\left(4\log_2(2/\delta) + 8VC(H)\log_2(13/\varepsilon)\right) \qquad m \geq \max\left[\frac{1}{\varepsilon}\log(1/\delta), \frac{VC(C)-1}{32\varepsilon}\right]$$

- 另一种学习模式称为出错界限模式，用于分析学习器在确切学习到目标概念之前会产生的误分类次数
  - Halving算法在学习到H中的任意目标概念前会有至多$\log_2|H|$次出错
  - 对任意概念类C，最坏情况下最佳算法将有Opt(C)次出错，满足$VC(C)<=Opt(C)<=\log_2|C|$

小结（4）

- 加权多数算法结合了多个预测算法的加权投票来分类新的实例，它基于这些预测算法在样例序列中的出错来学习每个算法的权值。加权多数算法产生的错误界限可用算法池中最佳预测算法的出错数来计算

补充读物

- 计算学习理论中许多早期的工作针对的问题是：学习器能否在极限时确定目标概念
- Gold1967给出了极限模型下的确定算法
- Angluin1992给出了一个好的综述
- Vapnik1982详细考察了一致收敛
- Valiant1984给出了PAC学习模型
- Haussler1988讨论了ε-详尽变型空间
- Bluer et al.1989给出了PAC模型下的一组有用的结论
- Kearns & Vazirani1994提供了计算学习理论中许多结论的优秀的阐述
- 会议：计算学习理论年会COLT
- 杂志：机器学习的特殊栏目

机器学习

第3章 决策树学习

概论

- 决策树学习是应用最广的归纳推理算法之一
- 是一种逼近离散值函数的方法
- 很好的健壮性
- 能够学习析取表达式
- ID3, Assistant, C4.5
- 搜索一个完整表示的假设空间
- 归纳偏置是优先选择较小的树
- 决策树表示了多个if-then规则

提纲

- 决策树定义
- 适用问题特征
- 基本ID3算法
- 决策树学习的归纳偏置
- 训练数据的过度拟合
- 更深入的话题

决策树表示法

- 决策树
  - 通过把实例从根节点排列到某个叶子节点来分类实例。
  - 叶子节点即为实例所属的分类
  - 树上每个节点说明了对实例的某个属性的测试
  - 节点的每个后继分支对应于该属性的一个可能值
- 图3-1
- 决策树代表实例属性值约束的合取的析取式。从树根到树叶的每一条路径对应一组属性测试的合取，树本身对应这些合取的析取。

决策树学习的适用问题

- 适用问题的特征
  - 实例由“属性-值”对表示
  - 目标函数具有离散的输出值
  - 可能需要析取的描述
  - 训练数据可以包含错误
  - 训练数据可以包含缺少属性值的实例
- 问题举例
  - 根据疾病分类患者
  - 根据起因分类设备故障
  - 根据拖欠支付的可能性分类贷款申请
- 分类问题
  - 核心任务是把样例分类到各可能的离散值对应的类别

基本的决策树学习算法

- 大多数决策树学习算法是一种核心算法的变体
- 采用自顶向下的贪婪搜索遍历可能的决策树空间
- ID3是这种算法的代表

基本的决策树学习算法（2）

- ID3的思想
  - 自顶向下构造决策树
  - 从“哪一个属性将在树的根节点被测试”开始
  - 使用统计测试来确定每一个实例属性单独分类训练样例的能力
- ID3的过程
  - 分类能力最好的属性被选作树的根节点
  - 根节点的每个可能值产生一个分支
  - 训练样例排列到适当的分支
  - 重复上面的过程

表3-1 用于学习布尔函数的ID3算法概要

- ID3(Examples, Target_attribute, Attributes)
- 创建树的root节点
- 如果Examples都为正,返回label=+的单节点树root
- 如果Examples都为反,返回label=-的单节点树root
- 如果Attributes为空，那么返回单节点root，label=Examples中最普遍的Target_attribute值
- 否则开始
  - A←Attributes中分类examples能力最好的属性
  - root的决策属性←A
  - 对于A的每个可能值vi
    - 在root下加一个新的分支对应测试A=vi
    - 令$Examples_{vi}$为Examples中满足A属性值为vi的子集
    - 如果$Examples_{vi}$为空
      - 在这个新分支下加一个叶子节点，节点的label=Examples中最普遍的Target_attribute值
      - 否则在新分支下加一个子树ID3（ $Examples_{vi}$,Target_attribute,Attributes-{A}）
- 结束
- 返回root

最佳分类属性

- 信息增益
  - 用来衡量给定的属性区分训练样例的能力
  - ID3算法在增长树的每一步使用信息增益从候选属性中选择属性
- 用熵度量样例的均一性
  - 熵刻画了任意样例集的纯度
  - 给定包含关于某个目标概念的正反样例的样例集S，那么S相对这个布尔型分类的熵为

    $\text{Entropy}(S)=-p+\log_2 p+ - p-\log_2 p-$
  - 信息论中对熵的一种解释，熵确定了要编码集合S中任意成员的分类所需要的最少二进制位数
  - 更一般地，如果目标属性具有c个不同的值，那么S相对于c个状态的分类的熵定义为

    $$\text{Entropy}(S)=\sum_{i=1}^{c} - p_i \log_2 p_i$$

最佳分类属性（2）

- 用信息增益度量期望的熵降低
  - 属性的信息增益，由于使用这个属性分割样例而导致的期望熵降低

$$Gain(S,A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{S} Entropy(S_v)$$

  - Gain(S,A)是在知道属性A的值后可以节省的二进制位数
  - 例子

ID3算法举例

- 表3-2
  - …
  - 继续这个过程，直到满足以下两个条件中的一个
    - 所有的属性已经被这条路经包括
    - 与这个节点关联的所有训练样例都具有相同的目标属性值

决策树学习中的假设空间搜索

- 观察ID3的搜索空间和搜索策略，认识到这个算法的优势和不足
  - 假设空间包含所有的决策树，它是关于现有属性的有限离散值函数的一个完整空间
  - 维护单一的当前假设（不同于第二章的变型空间候选消除算法）
  - 不进行回溯，可能收敛到局部最优
  - 每一步使用所有的训练样例，不同于基于单独的训练样例递增作出决定，容错性增强

决策树学习的归纳偏置

- ID3的搜索策略
  - 优先选择较短的树
  - 选择那些信息增益高的属性离根节点较近的树
  - 很难准确刻画ID3的归纳偏置
- 近似的ID3的归纳偏置
  - 较短的树比较长的树优先
  - 近似在于ID3得到局部最优，而不一定是全局最优
  - 一个精确具有这个归纳偏置的算法，BFS-ID3
- 更贴切近似的归纳偏置
  - 较短的树比较长的树优先，信息增益高的属性更靠近根节点的树优先

限定偏置和优选偏置

- ID3和候选消除算法的比较
  - ID3的搜索范围是一个完整的假设空间，但不彻底地搜索这个空间
  - 候选消除算法的搜索范围是不完整的假设空间，但彻底地搜索这个空间
  - ID3的归纳偏置完全是搜索策略排序假设的结果，来自搜索策略
  - 候选消除算法完全是假设表示的表达能力的结果，来自对搜索空间的定义

限定偏置和优选偏置

- 优选偏置
  - ID3的归纳偏置是对某种假设胜过其他假设的一种优选，对最终可列举的假设没有硬性限制
- 限定偏置
  - 候选消除算法的偏置是对待考虑假设的一种限定
- 通常优选偏置比限定偏置更符合归纳学习的需要
- 优选偏置和限定偏置的结合
  - 考虑第1章的例子

为什么短的假设优先

- ID3的归纳偏置的哲学基础
- 奥坎姆剃刀
  - 优先选择拟合数据的最简单的假设
- 科学上的例子
  - 物理学家优先选择行星运动的简单假设
  - 简单假设的数量远比复杂假设的数量少
  - 简单假设对训练样例的针对性更小，更像是泛化的规律，而不是训练样例的另一种描述

为什么短的假设优先

- 奥坎姆剃刀的困难
  - 我们反问，使用上页的推理，应该优先选择包含恰好17个叶子节点和11个非叶子节点的决策树？
  - 假设的规模由学习器内部使用的特定表示决定
- 从生物进化的观点看内部表示和奥坎姆剃刀原则

决策树学习的常见问题

- 决策树学习的实际问题
  - 确定决策树增长的深度
  - 处理连续值的属性
  - 选择一个适当的属性筛选度量标准
  - 处理属性值不完整的训练数据
  - 处理不同代价的属性
  - 提高计算效率
- 针对这些问题，ID3被扩展成C4.5

避免过度拟合数据

- 过度拟合
  - 对于一个假设，当存在其他的假设对训练样例的拟合比它差，但事实上在实例的整个分布上表现得却更好时，我们说这个假设过度拟合训练样例
  - 定义：给定一个假设空间H，一个假设h∈H，如果存在其他的假设h'∈H，使得在训练样例上h的错误率比h'小，但在整个实例分布上h'的错误率比h小，那么就说假设h过度拟合训练数据。
  - 图3-6的例子

避免过度拟合数据（2）

- 导致过度拟合的原因
  - 一种可能原因是训练样例含有随机错误或噪声
  - 当训练数据没有噪声时，过度拟合也有可能发生，特别是当少量的样例被关联到叶子节点时，很可能出现巧合的规律性，使得一些属性恰巧可以很好地分割样例，但却与实际的目标函数并无关系。

避免过度拟合数据（3）

- 避免过度拟合的方法
  - 及早停止树增长
  - 后修剪法
- 两种方法的特点
  - 第一种方法更直观
  - 第一种方法中，精确地估计何时停止树增长很困难
  - 第二种方法被证明在实践中更成功

避免过度拟合数据（4）

- 避免过度拟合的关键
  - 使用什么样的准则来确定最终正确树的规模
- 解决方法
  - 使用与训练样例截然不同的一套分离的样例，来评估通过后修剪方法从树上修建节点的效用。
  - 使用所有可用数据进行训练，但进行统计测试来估计扩展（或修剪）一个特定的节点是否有可能改善在训练集合外的实例上的性能。
  - 使用一个明确的标准来衡量训练样例和决策树的复杂度，当这个编码的长度最小时停止树增长。

避免过度拟合数据（5）

- 方法评述
  - 第一种方法是最普通的，常被称为训练和验证集法。
  - 可用数据分成两个样例集合：
    - 训练集合，形成学习到的假设
    - 验证集合，评估这个假设在后续数据上的精度
  - 方法的动机：即使学习器可能会被训练集合误导，但验证集合不大可能表现出同样的随机波动
  - 验证集合应该足够大，以便它本身可提供具有统计意义的实例样本。
  - 常见的做法是，样例的三分之二作训练集合，三分之一作验证集合。

错误率降低修剪

- 将树上的每一个节点作为修剪得候选对象
- 修剪步骤
  - 删除以此节点为根的子树，使它成为叶结点
  - 把和该节点关联的训练样例的最常见分类赋给它
  - 反复修剪节点，每次总是选取那些删除后可以最大提高决策树在验证集合上的精度的节点
- 继续修剪，直到进一步的修剪是有害的为止
- 数据分成3个子集
  - 训练样例，形成决策树
  - 验证样例，修剪决策树
  - 测试样例，精度的无偏估计
- 如果有大量的数据可供使用，那么使用分离的数据集合来引导修剪

规则后修剪

- 从训练集合推导出决策树，增长决策树直到尽可能好地拟合训练数据，允许过度拟合发生
- 将决策树转化为等价的规则集合，方法是为从根节点到叶节点的每一条路径创建一条规则
- 通过删除任何能导致估计精度提高的前件来修剪每一条规则
- 按照修剪过的规则的估计精度对它们进行排序，并按这样的顺序应用这些规则来分类后来的实例

规则后修剪（2）

- 例子
  - 图3-1的最左一条路径
  - if (outlook=sunny)∧(Humidity=High) then PlayTennis=No
  - 考虑删除先行词(outlook=sunny)和(Humidity=High)
  - 选择使估计精度有最大提升的步骤
  - 考虑修剪第二个前件

规则后修剪（3）

- 规则精度估计方法
  - 使用与训练集不相交的验证集
  - 基于训练集合本身
    - 被C4.5使用，使用一种保守估计来弥补训练数据有利于当前规则的估计偏置
    - 过程
      - 先计算规则在它应用的训练样例上的精度
      - 然后假定此估计精度为二项式分布，并计算它的标准差
      - 对于一个给定的置信区间，采用下界估计作为规则性能的度量
    - 评论
      - 对于大的数据集，保守预测非常接近观察精度，随着数据集合的减小，离观察精度越来越远
      - 不是统计有效（此概念第5章介绍），但是实践中发现有效

规则后修剪（4）

- 把决策树转化成规则集的好处
  - 可以区分决策节点使用的不同上下文
  - 消除了根节点附近的属性测试和叶节点附近的属性测试的区别
  - 提高了可读性

合并连续值属性

- ID3被限制为取离散值的属性
  - 学习到的决策树要预测的目标属性必须是离散的
  - 树的决策节点的属性也必须是离散的
- 简单删除上面第2个限制的方法
  - 通过动态地定义新的离散值属性来实现，即先把连续值属性的值域分割为离散的区间集合

合并连续值属性（2）

- 例子，Temperature应该定义什么样的基于阈值的布尔属性
  - 选择产生最大信息增益的阈值
  - 按照连续属性排列样例，确定目标分类不同的相邻实例
  - 产生一组候选阈值，它们的值是相应的A值之间的中间值
  - 可以证明产生最大信息增益的c值位于这样的边界中（Fayyad1991）
  - 通过计算与每个候选阈值关联的信息增益评估这些候选值
- 方法的扩展
  - 连续的属性分割成多个区间，而不是单一阈值的两个空间

属性选择的其他度量标准

- 信息增益度量存在一个内在偏置，偏向具有较多值的属性
- 避免方法，其他度量，比如增益比率
- 增益比率通过加入一个被称作分裂信息的项来惩罚多值属性，分裂信息用来衡量属性分裂数据的广度和均匀性

  $$\text{SplitInformation}(S,A)=-\sum_{i=1}^{c}\frac{|S_i|}{|S|}\log_2\frac{|S_i|}{|S|}$$

  $$\text{GainRatio}(S,A)=\frac{Gain(S,A)}{SplitInformation(S,A)}$$
- 分裂信息项阻碍选择值为均匀分布的属性
- 问题，当某个$S_i \approx S$。解决方法：采用一些启发式规则，比如仅对增益高过平均值的属性应用增益比率测试

属性选择的其他度量标准（2）

- 基于距离的度量
  - 定义了数据划分间的一种距离尺度
  - 计算每个属性产生的划分与理想划分间的距离
  - 选择最接近完美划分的属性
  - Lopez de Mantaras定义了这个距离度量，证明了它不偏向有大量值的属性
- 此外
- Mingers实验，不同的属性选择度量对最终精度的影响小于后修剪得程度和方法的影响

缺少属性值的训练样例

- 例子，医学领域
- 经常需要根据此属性值已知的实例来估计这个缺少的属性值
- 为了评估属性A是否是决策节点n的最佳测试属性，要计算决策树在该节点的信息增益Gain(S,A)。假定<x,c(x)>是S中的一个训练样例，并且其属性A的值A(x)未知

缺少属性值的训练样例（2）

- 处理缺少属性值的
  - 一种策略是赋给它节点n的训练样例中该属性的最常见值
  - 另一种策略是赋给它节点n的被分类为c(x)的训练样例中该属性的最常见值
  - 更复杂的策略，为A的每个可能值赋予一个概率，而不是简单地将最常见的值赋给A(x)

处理不同代价的属性

- 实例的属性可能与代价相关
- 优先选择尽可能使用低代价属性的决策树，仅当需要产生可靠的分类时才依赖高代价属性
- 通过引入一个代价项到属性选择度量中，可以使ID3算法考虑属性代价
- Tan和Schlimmer的例子

小结和补充读物

- 决策树学习为概念学习和学习其他离散值的函数提供了一个实用的方法
- ID3算法
  - 贪婪算法
  - 从根向下推断决策树
  - 搜索完整的假设空间
  - 归纳偏置，较小的树
- 过度拟合问题
- ID3算法的扩展

小结和补充读物（2）

- Hunt
- Quinlan
- Mingers

附录

- C4.5 is a software extension of the basic ID3 algorithm designed by Quinlan to address the following issues not dealt with by ID3:
  - Avoiding overfitting the data
  - Determining how deeply to grow a decision tree.
  - Reduced error pruning.
  - Rule post-pruning.
  - Handling continuous attributes.
  - e.g., temperature
  - Choosing an appropriate attribute selection measure.
  - Handling training data with missing attribute values.
  - Handling attributes with differing costs.
  - Improving computational efficiency.

机器学习

第5章 评估假设

概述

- 对假设的精度进行评估是机器学习中的基本问题
- 本章介绍用统计方法估计假设精度，主要解决以下三个问题：
  - 已知一个假设在有限数据样本上观察到的精度，怎样估计它在其他实例上的精度？
  - 如果一个假设在某些数据样本上好于另一个，那么一般情况下该假设是否更准确？
  - 当数据有限时，怎样高效地利用这些数据，通过它们既能学习到假设，还能估计其精度？
- 统计的方法，结合有关数据基准分布的假定，使我们可以用有限数据样本上的观察精度来逼近整个数据分布上的真实精度

动机

- 对学习到的假设进行尽可能准确地性能评估十分重要
  - 为了知道是否可以使用该假设
  - 是许多学习方法的重要组成部分
- 当给定的数据集有限时，要学习一个概念并估计其将来的精度，存在两个很关键的困难：
  - 估计的困难
    - 使用与训练样例和假设无关的测试样例
  - 估计的方差
    - 即使假设精度在独立的无偏测试样例上测量，得到的精度仍可能与真实精度不同。
    - 测试样例越少，产生的方差越大
- 本章讨论了对学到的假设的评估、对两个假设精度的比较、两个学习算法精度的比较

学习问题的框架

- 有一所有可能实例的空间X，其中定义了多个目标函数，我们假定X中不同实例具有不同的出现频率。一种合适的建模方式是，假定存在一未知的概率分布D，它定义了X中每一实例出现的概率。
- 学习任务是在假设空间上学习一个目标概念，训练样例的每一个实例按照分布D独立地抽取，然后连同正确的目标值提供给学习器。

评估假设的问题

- 给定假设h和包含若干按D分布抽取的样例的数据集，如何针对将来按同样分布抽取的实例，得到对h的精度最好估计
- 这一精度估计的可能的误差是多少

样本错误率和真实错误率

- 定义：假设h关于目标函数f和数据样本S的样本错误率（标记为$error_s(h)$）

$$error_s(h) = \frac{1}{n}\sum_{x \in S} \delta(f(x), h(x))$$

$$n = |S|$$

$$\delta(f(x), h(x)) = \begin{cases} 1 & f(x) \neq h(x) \\ 0 & otherwise \end{cases}$$

- 定义：假设h关于目标函数f和分布D的真实错误率（标记为$error_D(h)$）

$$error_D(h) = \Pr_{x \in D}[f(x) \neq h(x)]$$

样本错误率和真实错误率（2）

- 我们想知道的是假设的真实误差，因为这是在分类未来样例时可以预料到的误差。
- 我们所能测量的只是样本错误率，因为样本数据是我们知道的。
- 本节要考虑的问题是：样本错误率在何种程度上提供了对真实错误率的估计？

离散值假设的置信区间

- 先考虑离散值假设的情况，比如：
  - 样本S包含n个样例，它们的抽取按照概率分布D，抽取过程是相互独立的，并且不依赖于假设h
  - n>=30
  - 假设h在这n个样例上犯了r个错误
- 根据上面的条件，统计理论可以给出以下断言：
  - 没有其他信息的话，真实错误率$error_D(h)$最可能的值是样本错误率$error_S(h)=r/n$
  - 有大约95%的可能性，真实错误率处于下面的区间内：$error_S(h) \pm 1.96\sqrt{\frac{error_S(h)(1-error_S(h))}{n}}$

举例说明

- 数据样本S包含n=40个样例，并且假设h在这些数据上产生了r=12个错误，这样样本错误率为 $error_S(h)=12/40=0.3$
- 如果没有更多的信息，对真实错误率$error_D(h)$的最好的估计即为0.3
- 如果另外收集40个随机抽取的样例S’，样本错误率$error_{S'}(h)$将与原来的$error_S(h)$存在一些差别
- 如果不断重复这一实验，每次抽取一个包含40样例的样本，将会发现约95%的实验中计算所得的区间包含真实错误率
- 将上面的区间称为$error_D(h)$的95%置信区间估计

置信区间表达式的推广

- 常数1.96是由95%这一置信度确定的
- 定义z_N为计算N%置信区间的常数（取值见表5-1），计算$error_D(h)$的N%置信区间的一般表达式（公式5.1）为：

$$error_S(h) \pm z_N \sqrt{\frac{error_S(h)(1-error_S(h))}{n}}$$

- 可以求得同样情况下的68%置信区间，从直觉上可以看出68%置信区间要小于95%置信区间，因为减小了要求$error_D(h)$落入的概率

置信区间表达式的推广（2）

- 公式5.1只能应用于离散值假设，它假定样本S抽取的分布与将来的数据抽取的分布相同，并且假定数据不依赖于所测试的假设
- 公式5.1只提供了近似的置信区间，这一近似在至少包含30个样例，并且$error_S(h)$不太靠近0或1时很接近真实情况
- 判断这种近似是否接近真实的更精确规则是：

$$n \cdot error_S(h)(1 - error_S(h)) \geq 5$$

统计学中的基本定义和概念

- 随机变量
- 某随机变量Y的概率分布
- 随机变量Y的期望值或均值
- 随机变量的方差
- Y的标准差
- 二项分布
- 正态分布
- 中心极限定理
- 估计量
- Y的估计偏差
- N%置信区间

错误率估计和二项比例估计

- 样本错误率和真实错误率之间的差异与数据样本大小的依赖关系如何？
- 给定从总体中随机抽取的某些样本的观察比例，估计某个属性在总体的比例
- 此处，我们感兴趣的属性是：假设h对实例错误分类

错误率估计和二项比例估计（2）

- 测量样本错误率相当于在作一个有随机输出的实验
- 从分布D中随机抽取n个独立的实例，形成样本S，然后测量样本错误率$error_S(h)$
- 将实验重复多次，每次抽取大小为n的不同的样本S_i，得到不同的$error_{S_i}(h)$，取决于S_i的组成中的随机差异
- $error_{S_i}(h)$被称为一随机变量，一般情况下，可以将随机变量看成一个有随机输出的实验。随机变量值即为随机实验的观察输出

错误率估计和二项比例估计（3）

- 设想要运行k个这样的随机实验，得到k个随机变量值，以图表的形式显示观察到的每个错误率值的频率
- 当k不断增长，该图表将呈现如表5-3所显示的分布，称为二项分布

二项分布

- 有一非均质硬币，要估计在抛硬币时出现正面的概率p
- 投掷硬币n次并计算出现正面的次数r，那么p的一个合理估计是r/n
- 如果重新进行一次实验，生成一个新的n次抛硬币的集合，出现正面的次数r可能与前不同，得到对p的另一个估计
- 二项分布描述的是对任一可能的r值，这个正面概率为p的硬币抛掷n次恰好出现r次正面的概率

二项分布（2）

- 从抛掷硬币的随机样本中估计p与在实例的随机样本上测试h以估计$error_D(h)$是相同的问题
- 一次硬币抛掷对应于从D中抽取一个实例并测试它是否被h误分类
- 一次随机抛掷出现正面的概率p对应于随机抽取的实例被误分类的概率$error_D(h)$
- 二项分布给出了一个一般形式的概率分布，无论用于表示n次硬币出现正面的次数还是在n个样例中假设出错的次数
- 二项分布的具体形式依赖于样本大小n以及概率p或$error_D(h)$

应用二项分布的条件

- 有一基本实验，其输出可被描述为一随机变量Y，随机变量Y有两种取值
- 在实验的任一次尝试中Y=1的概率为常数p，它与其他实验尝试无关，因此Y=0的概率为1-p
- p为预先未知，面临的问题是如何估计
- 基本实验的n次独立尝试按序列执行，生成一个独立同分布的随机变量序列
- 随机变量R表示n次实验中出现Y_i=1的次数，它取特定值r的概率由二项分布给出

$$\Pr(R=r)=\frac{n!}{r!(n-r)!}p^r(1-p)^{n-r}$$

均值

- 期望值是重复采样随机变量得到的值的平均
- 定义：考虑随机变量Y可能的取值为$y_1 \cdots y_n$，Y的期望值E[Y]定义如下：

$$E[Y] = \sum_{i=1}^{n} y_i \Pr(Y = y_i)$$

- 如果随机变量Y服从二项分布，那么可得E[Y]=np

方差

- 方差描述的是概率分布的宽度或散度，描述了随机变量与其均值之间的差有多大
- 定义：随机变量Y的方差Var[Y]定义如下：

$$Var[Y] = E[(Y - E(Y))^2]$$

描述了从Y的一个观察值估计其均值E[Y]的误差平方的期望

- 随机变量Y的标准差σ_Y

$$\sigma_Y = \sqrt{E[(Y - E[Y])^2]}$$

- 若随机变量Y服从二项分布，则方差和标准差分别为：

Var[Y]=np(1-p)

$$\sigma_Y = \sqrt{np(1-p)}$$

估计量、偏差和方差

- 回到问题：我们得出了随机变量$error_S(h)$服从二项分布，那么$error_S(h)$和$error_D(h)$之间可能的差异是多少？
- 用5.2式定义的二项分布，可得

  $error_S(h)=r/n$

  $error_D(h)=p$
- 统计学中将$error_S(h)$称为$error_D(h)$的一个估计量
- 估计量是用来估计总体的某一参数的随机变量，最关心的是它平均来说是否能产生正确估计

估计量、偏差和方差（2）

- 估计偏差衡量估计量的期望值同真实参数值之间的差异
- 定义：针对任意参数p的估计量Y的估计偏差是：E[Y]-p
- 如果估计偏差为0，称Y为p的无偏估计量，在此情况下，由多次重复实验生成的Y的多个随机值的平均将收敛于p
- 由于$error_S(h)$服从二项分布，因此$error_S(h)$是$error_D(h)$的一个无偏估计量

估计量、偏差和方差（3）

- 对估计偏差的补充说明：
  - 要使$error_S(h)$是$error_D(h)$的无偏估计，假设h和样本S必须独立选取
  - 估计偏差不能与第2章介绍的学习器的归纳偏置相混淆
- 估计量的另一重要属性是它的方差，给定多个无偏估计量，选取其中方差最小的
- 由方差的定义，所选择的应为参数值和估计值之间期望平方误差最小的

估计量、偏差和方差（4）

- 一个例子
  - n=40个随机样例
  - r=12个错误
  - $error_S(h)$的标准差
- 一般地，若在n个随机选取的样本中有r个错误，$error_S(h)$的标准差是：

$$\sigma_{error_S(h)} = \frac{\sigma_r}{n} = \sqrt{\frac{p(1-p)}{n}}$$

近似地

$$\sigma_{error_S(h)} = \sqrt{\frac{error_S(h)(1-error_S(h))}{n}}$$

置信区间

- 通常描述某估计的不确定性的方法是使用置信区间，真实的值以一定的概率落入该区间中，这样的估计称为置信区间估计
- 定义：某个参数p的N%置信区间是一个以N%的概率包含p的区间
- 由于估计量$error_S(h)$服从二项分布，这一分布的均值为$error_D(h)$，标准差可由式5.9计算，因此，为计算95%置信区间，只需要找到一个以$error_D(h)$为中心的区间，它的宽度足以包含该分布全部概率的95%
- 这提供了一个包围$error_D(h)$的区间，使$error_S(h)$有95%机会落入其中，同样它也指定了$error_D(h)$有95%的机会落入包围$error_S(h)$的区间的大小

置信区间（2）

- 对于二项分布，计算置信区间很烦琐，多数情况下，计算它的近似值
- 对于足够大的样本，二项分布可以由正态分布来近似，而正态分布的置信区间容易得到
- 如果随机变量Y服从均值为μ，标准差为σ的一个正态分布，那么Y的任一观察值y有N%的机会落入下面的区间 $\mu \pm z_N \sigma$
- 相似地，均值μ有N%的机会落入下面的区间 $y \pm z_N \sigma$

置信区间（3）

- 式子5.1的三步推导过程
  - $error_S(h)$遵从二项分布，其均值为$error_D(h)$，标准差如式5.9所示
  - 对于足够大的样本n，二项分布非常近似于正态分布
  - 式5.11告诉我们如何根据正态分布的均值求出N%置信区间
- 式子5.1的推导中有两个近似
  - 估计$error_S(h)$的标准差，我们将$error_D(h)$近似为$error_S(h)$
  - 用正态分布近似二项分布
- 统计学的一般规则表明，这两个近似在n>=30或np(1-p)>=5时工作得很好，对于较小的n值，最好使用列表的形式给出二项分布的具体值

双侧和单侧边界

- 上述的置信区间是双侧的，有时用到单侧边界
- 例如问题“$error_D(h)$至多为U的概率”，在只要限定h的最大错误率，而不在乎真实错误率是否小于估计错误率时，很自然提出这种问题
- 由于正态分布关于其均值对称，因此，任意正态分布上的双侧置信区间能够转换为相应的单侧区间，置信度为原来的两倍（见图5-1b）
- 由一个有下界L和上界U的100(1-α)%置信区间，可得到一个下界为L且无上界的100(1- α/2)%置信区间，也得到一个有上界U且无下界的100(1- α/2)%置信区间

推导置信区间的一般方法

- 前面介绍的是针对一特定情况推导置信区间估计：基于独立抽取的n个样本，估计离散值假设的$error_D(h)$
- 本节介绍的方法是在许多估计问题中用到的通用的方法
- 基于大小为n的随机抽取样本的均值，来估计总体均值的问题

通用的过程的步骤

- 确定基准总体中要估计的参数p，例如$error_D(h)$
- 定义一个估计量Y（如$error_S(h)$），它的选择应为最小方差的无偏估计量
- 确定控制估计量Y的概率分布D_Y，包括其均值和方差
- 通过寻找阈值L和U确定N%置信区间，以使这个按D_Y分布的随机变量有N%机会落入L和U之间

中心极限定理

- 考虑如下的一般框架
  - 在n个独立抽取的且服从同样概率分布的随机变量$Y_1 ... Y_n$中观察试验值
  - 令μ代表每一变量Y_i服从的未知分布的均值，并令σ代表标准差，称这些变量Y_i为独立同分布随机变量
  - 为了估计Yi服从的分布的均值μ，我们计算样本的均值 $\bar{Y}_n = \frac{1}{n}\sum_{i=1}^{n} Y_i$
  - 中心极限定理说明在$n \to \infty$时，$\bar{Y}_n$所服从的概率分布为一正态分布，而不论Yi本身服从什么样的分布
  - $\bar{Y}_n$服从的分布均值为μ，而标准差为$\frac{\sigma}{\sqrt{n}}$

中心极限定理（2）

- 定理5.1（中心极限定理）考虑独立同分布的随机变量$Y_1...Y_n$的集合，它们服从一任意的概率分布，均值为μ，有限方差为σ^2，定义样本均值为$\bar{Y}_n = \frac{1}{n}\sum_{i=1}^{n} Y_i$，当$n \to \infty$时，式子$\frac{\bar{Y}_n - \mu}{\frac{\sigma}{\sqrt{n}}}$服从正态分布，均值为0且标准差为1
- 中心极限定理说明在不知道独立的Yi所服从的基准分布的情况下，我们可以得知样本均值的分布形式，说明了怎样使用$\bar{Y}$的均值和方差来确定独立的Yi的均值和方差
- 中心极限定理说明了任意样本均值的估计量服从的分布在n足够大时可以近似为正态分布

两个假设错误率间的差异

- 问题：
  - 考虑某离散目标函数的两个假设h_1和h_2，h_1在一拥有n_1个随机抽取的样例的样本S_1上测试，h_2在一拥有n_2个从相同分布中抽取的样例的样本S_2上测试，要估计这两个假设的真实错误率间的差异

    $d=error_D(h_1)-error_D(h_2)$

两个假设错误率间的差异（2）

- 使用5.4节中描述的四个步骤来推导d的置信区间估计
  - 确定待估计的参数，如上所述的d
  - 定义一估计量，$\hat{d} = error_{S_1}(h_1) - error_{S_2}(h_2)$
  - $\hat{d}$ 是d的无偏估计量，即E[$\hat{d}$]=d。$\hat{d}$由于对于较大的n1和n2，$error_{S1}(h_1)$和$error_{S2}(h_2)$都近似遵从正态分布，两个正态分布的差仍为正态分布，方差为两个正态分布的方差的和

    $$\sigma_{\hat{d}}^2 \approx \frac{error_{S_1}(h_1)(1-error_{S_1}(h_1))}{n_1} + \frac{error_{S_2}(h_2)(1-error_{S_2}(h_2))}{n_2}$$
  - 现在知道了$\hat{d}$服从均值为d、方差为σ2的正态分布，因此d的N%置信区间是$\hat{d} \pm z_N \sigma$

两个假设错误率间的差异（3）

- 上面分析的是h_1和h_2在相互独立的数据样本上测试的情况，如果在同一个样本上测试h_1和h_2，那么也可以使用公式5.13计算置信区间
- 这种情况下的方差通常小于式子5.12给出的方差，这是因为单个样本消除了两个样本组合带来的随机差异，这样，由式子5.13给出的置信区间一般来说偏于保守，但结果是正确的

假设检验

- 有时感兴趣的是某个特定猜想正确的概率，而不是对某参数的置信区间估计。比如：$error_D(h_1)>error_D(h_2)$的可能性有多大？
- 例子，假定分别用大小为100的独立样本S1和S2测量h1和h2的样本错误率为0.30和0.20，给定$\hat{d}=0.10$，问$error_D(h_1)>error_D(h_2)$的概率是多少？d>0的概率是多少？
- 概率Pr(d>0)等于$\hat{d}$对d的过高估计不大于0.1的概率，也就是这个概率为$\hat{d}$落入单侧区间

  $\hat{d}<d+0.10=\mu_{\hat{d}}+0.10$的概率

假设检验（2）

- 对于$\hat{d}$落入单侧区间$\hat{d}<\mu_{\hat{d}}+0.10$的概率，可以通过计算分布在该区间的概率质量来确定$\hat{d}$落入这个单侧区间的概率
- 将区间$\hat{d}<\mu_{\hat{d}}+0.10$用允许偏离均值的标准差的数目来重新表示，根据式5.12可得$\sigma_{\hat{d}}\approx 0.061$，所以这一区间可近似表示为$d<\mu_{\hat{d}}+\frac{0.10}{\sigma_{\hat{d}}}\cdot\sigma_{\hat{d}}=\mu_{\hat{d}}+1.64\sigma_{\hat{d}}$
- 查表5-1知，关于均值的1.64标准差对应置信度90%的双侧区间，因此这个单侧区间具有95%的置信度
- 因此给定观察$\hat{d}=0.1$，$error_D(h_1)>error_D(h_2)$的概率约为95%。使用统计学术语表述为：接受$error_D(h_1)>error_D(h_2)$假设的置信度是95%

学习算法比较

- 有时感兴趣的是比较两个学习算法的性能，而不是两个具体的假设本身
  - 如何近似地检验多个学习算法？
  - 如何确定两个算法之间的差异在统计上是有意义的？
- 假定有L_A和L_B两个算法，要确定为了学习一特定目标函数f，平均来说那个算法更好
- 定义“平均”的一种合理方法是，从一基准实例分布中抽取包含n个样例的训练集合，在所有这样的集合中测量两个算法的平均性能，即

$$\underset{S \subset D}{E}[error_D(L_A(S)) - error_D(L_B(S))]$$

学习算法比较（2）

- 在实际的学习算法比较中，我们只有一个有限的样本D0，把它分割成训练集合S0和测试集合T0，使用下式比较两个学习到的假设的准确度

$$error_{T_0}(L_A(S_0)) - error_{T_0}(L_B(S_0))$$

- 上式与5.14有两个重要的不同
  - 使用$error_{T0}(h)$来近似$error_D(h)$
  - 错误率的差异测量是在一个训练集合S0上，而不是在从分布D中抽取的所有样本S上计算的期望值
- 改进5.15式的一种方法是，将数据D0多次分割为不相交的训练和测试集合，然后在其中计算这些不同的实验的错误率的平均值，见表5-5

$\bar{\delta}$

学习算法比较（3）

- 表5-5返回的 可被用作对公式5.14的一个估计，更合适的说法是把 看作下式的估计

$$\underset{S\subset D_0}{E}[error_D(L_A(S)) - error_D(L_B(S))]$$

- 估计式5.17的近似的N%置信区间可表示成 $\bar{\delta} \pm t_{N,k-1} s_{\bar{\delta}}$，其中 $t_{N,k-1}$是一常数，其意义类似于前面的z_N，第一个下标表示所需的置信度，第二个下标表示自由度，常记作v，它与生成随机变量的值时独立的随机事件数目相关。而 代表 所服从的概率分布的标准差的估计，定义如下

$$s_{\bar{\delta}} = \sqrt{\frac{1}{k(k-1)} \sum_{i=1}^{k} (\delta_i - \bar{\delta})^2}$$

- 注意当自由度$v \to \infty$时，$t_{N,v}$的值趋向常数z_N。

学习算法比较（4）

- 这里描述的比较学习算法的过程要在同样的测试集合上测试两个假设，这与5.5节中描述的比较两个用独立测试集合评估过的假设不同。
- 使用相同样本来测试假设被称为配对测试，配对测试通常会产生更紧密地置信区间，因为在配对测试中任意的差异都来源于假设之间的差异。
- 若假设在分开的数据样本上的测试，两个样本错误率之间的差异也可能部分来源于两个样本组成的不同。

配对t测试

- 本节讨论给定固定数据集时比较两个学习算法的过程，并论证公式5.18和5.19
- 为了理解5.18中的置信区间，考虑一下的估计问题
  - 给定一系列独立同分布的随机变量Y1…Yk的观察值
  - 要估计这些Yi所服从的概率分布的均值μ
  - 使用的估计量为样本均值 $\bar{Y} = \frac{1}{k}\sum_{i=1}^{k} Yi$

配对t测试（2）

- 这一基于样本均值估计分布均值的问题非常普遍（比如，早先的用$error_S(h)$估计$error_D(h)$）
- 由式5.18和5.19描述的t测试应用于该问题的一特殊情形，即每个单独的Yi都遵循正态分布
- 考虑表5-5比较学习算法的过程的一个理想化形式，假定不是拥有固定样本数据D0，而是从基准实例分布中抽取新的训练样例，使每一次循环需要的训练集Si和测试集Ti是从基准实例分布中抽取
- 这一理想化方法能很好地匹配上面的估计问题，该过程所测量的δi对应独立同分布的随机变量Yi，其分布的均值μ对应两学习算法错误率的期望差异。

配对t测试（3）

- 测试集Ti至少包含30个样例，因此，单独的δi将近似遵循正态分布，因此，我们也要求Yi服从近似的正态分布，样本均值$\bar{Y}$也遵循正态分布
- 由此，可以考虑使用前面的计算置信区间的表达式。然而，该公式要求我们知道这个分布的标准差，但这个标准差未知
- t测试正好用于这样的情形，即估计一系列独立同正态分布的随机变量的样本均值
- 当k趋近于无穷时，t分布趋近于正态分布，即$t_{N,k-1}$趋近于正态分布，因为样本规模k增加时，$s_{\bar{Y}}$收敛到真实的标准差，并且当标准差确切已知时可使用z_N。

实际考虑

- 上面的讨论说明了在使用样本均值来估计一个包含k个独立同正态分布的随机变量的样本均值时，可使用式5.18来估计置信区间
- 这个结论假定对于目标函数的样例可进行无限存取，实际问题是随机变量之间并不独立，因为它们基于从有限子集中抽取的相互重叠的训练样例
- 当只有一个有限的数据样本可用时，有几种重叠采用的方法。
  - 表5-5描述了k-fold方法
  - 随机抽取至少有30个样例的测试集合，剩余样例组成训练集合，重复这一过程直到足够的次数

实际考虑（2）

- 随机方法的好处是能够重复无数次，以减少置信区间到需要的宽度，而k-fold方法受限于样例的总数
- 随机方法的缺点是，测试集合不再被看作是从基准实例分布中独立抽取，而k-fold交叉验证生成的测试集合是独立的，因为一个实例只在测试集合中出现一次
- 概括而言，统计学模型在数据有限时很少能完美地匹配学习算法验证中的所有约束。然而，它们确实提供了近似的置信区间。

小结

- 统计理论提供了一个基础，从而基于在数据样本S上的观察错误率，估计真实错误率。
- 估计置信区间的问题可通过一待估计的参数以及相对应的估计量来完成。由于估计量是一个随机变量，它可由其服从的概率分布来描述。置信区间的计算可通过确定该分布下包含所需概率质量的区间来描述。
- 估计假设精度中的一种可能误差为估计偏差。如果Y为对某参数p的估计量，Y的估计偏差为Y的期望值和p之间的差

小结（2）

- 估计产生误差的第二种原因是估计方差。即使对于无偏估计，估计量的观察值也可能在各实验中不同，估计量分布的方差描述了该估计与真实值的不同有多大。方差在数据样本增大时降低。
- 比较两个学习算法效果的问题在数据和时间无限时是一个相对容易的估计问题，但在资源有限时要困难得多。本章描述的一种途径是在可用数据的不同子集上运行学习算法，在剩余数据上测试学到的假设，然后取这些结果的平均值。
- 本章考虑的多数情况中，推导置信区间需要多个假定和近似。近似计算分布的方差，以及假定实例从一固定不变的概率分布中生成。

补充读物

- Billingsley et al.提供了对统计学的一个很简明的介绍，详尽讨论本章涉及的一些问题
- DeGroot、Casella & Berger、Duda & Hart在数值模式识别领域提出了对这些问题的解决方法
- Segre et al.、Etzioni & Etzioni、Gordon & Segre讨论了评估学习算法的统计意义测试
- Geman et al.讨论了在同时最小化偏差和最小化方差之间作出的折中。Dietterich讨论了在不同训练测试数据分割下使用配对差异t测试带来的风险

机器学习

第4章 人工神经网络（ANN）

概述

- 人工神经网络提供了一种普遍且实用的方法从样例中学习值为实数、离散值或向量的函数
- 反向传播算法，使用梯度下降来调节网络参数以最佳拟合由输入-输出对组成的训练集合
- 人工神经网络对于训练数据中的错误健壮性很好
- 人工神经网络已被成功应用到很多领域，例如视觉场景分析，语音识别，机器人控制

简介

- 神经网络学习对于逼近实数值、离散值或向量值的目标函数提供了一种健壮性很强的方法
- 对于某些类型的问题，如学习解释复杂的现实世界中的传感器数据，人工神经网络是目前知道的最有效的学习方法
- 反向传播算法
- 成功例子，学习识别手写字符，学习识别口语，学习识别人脸

生物学动机

- ANN受到生物学的启发，生物的学习系统是由相互连接的神经元组成的异常复杂的网络。
- ANN由一系列简单的单元相互密集连接构成的，其中每一个单元有一定数量的实值输入，并产生单一的实数值输出
- 人脑的构成，大约有10^{11}个神经元，平均每一个与其他10^{4}个相连
- 神经元的活性通常被通向其他神经元的连接激活或抑制
- 最快的神经元转换时间比计算机慢很多，然而人脑能够以惊人的速度做出复杂度惊人的决策
- 很多人推测，生物神经系统的信息处理能力一定得益于对分布在大量神经元上的信息表示的高度并行处理

生物学动机（2）

- ANN系统的一个动机就是获得这种基于分布表示的高度并行算法
- ANN并未模拟生物神经系统中的很多复杂特征
- ANN的研究分为两个团体
  - 使用ANN研究和模拟生物学习过程
  - 获得高效的机器学习算法，不管这种算法是否反映了生物过程
- 本书属于后一个研究团体

神经网络表示

- ALVINN系统
  - Pomerleau 1993
  - 使用一个学习到的ANN以正常的速度在高速公路上驾驶汽车
  - ANN的输入是一个30x32像素的网格，输出是车辆行进的方向
  - 每个节点对应一个网络单元的输出，而从下方进入节点的实线为其输入
  - 隐藏单元，输出仅在网络内部，不是整个网络输出的一部分
  - 每个输出单元对应一个特定的驾驶方向，这些单元的输出决定哪一个方向是被最强烈推荐的

神经网络表示（2）

- ALVINN是很多ANN的典型结构，所有单元分层互连形成一个有向无环图
- 通常，ANN图结构可以有很多种类型
  - 无环或有环
  - 有向或无向
- 本章讨论以反向传播算法为基础的ANN方法
- 反向传播算法假定网络是一个固定结构，对应一个有向图，可能包含环
- ANN学习就是为图中每一条边选取权值
- 大多数实际应用与ALVINN相似

适合神经网络学习的问题

- 训练集合为含有噪声的复杂传感器数据，例如来自摄像机和麦克风
- 需要较多符号表示的问题，例如决策树学习的任务，能够取得和决策树学习大体相当的结果
- 反向传播算法是最常用的ANN学习技术

反向传播算法适合问题的特征

- 实例是用很多“属性-值”对表示的
- 目标函数的输出可能是离散值、实数值或者由若干实数属性或离散属性组成的向量
- 训练数据可能包含错误
- 可容忍长时间的训练
- 可能需要快速求出目标函数值
- 人类能否理解学到的目标函数是不重要的

本章余后部分提纲

- 讨论训练单个单元的学习算法
- 介绍组成神经网络的几种主要单元
  - 感知器（perceptron）
  - 线性单元（liner unit）
  - sigmoid单元（sigmoid unit）
- 给出训练多层网络的反向传播算法
- 考虑几个一般性问题
  - ANN的表征能力
  - 假设空间搜索的本质特征
  - 过度拟合问题
  - 反向传播算法的变体
- 例子，利用反向传播算法训练识别人脸的ANN

感知器

- 一种类型的ANN系统是以感知器为基础
- 感知器以一个实数值向量作为输入，计算这些输入的线性组合，如果结果大于某个阈值，就输出1，否则输出-1

$$o(x_1,...,x_n)=\begin{cases}1 & w_0+w_1x_1+...+w_nx_n>0\\ -1 & otherwise\end{cases}$$

其中每个w_i是一个实数常量，或叫做权值，用来决定输入x_i对感知器输出的贡献率。特别地，-w0是阈值。

感知器（2）

- 两种简化形式，附加一个常量输入$x_0=1$，前面的不等式写成

$$\sum_{i=0}^{n} w_i x_i > 0$$

或写成向量形式

$$\vec{w}\cdot\vec{x} > 0$$

- 为了简短起见，把感知器函数写为

$$o(\vec{x}) = \mathrm{sgn}(\vec{w}\cdot\vec{x})$$

其中，$\mathrm{sgn}(y) = \begin{cases} 1 & y > 0 \\ -1 & otherwise \end{cases}$

感知器（3）

- 学习一个感知器意味着选择权$w_0,\ldots,w_n$的值。所以感知器学习要考虑的候选假设空间H就是所有可能的实数值权向量的集合 $H = \{\vec{w} \mid \vec{w} \in R^{n+1}\}$

感知器的表征能力

- 可以把感知器看作是n维实例空间（即点空间）中的超平面决策面
- 对于超平面一侧的实例，感知器输出1，对于另一侧的实例，输出-1
- 这个决策超平面方程是 $\vec{w} \cdot \vec{x} = 0$
- 可以被某个超平面分割的样例集合，称为线性可分样例集合

感知器的表征能力（2）

- 单独的感知器可以用来表示很多布尔函数
- 表示m-of-n函数
- 感知器可以表示所有的原子布尔函数：与、或、与非、或非
- 然而，一些布尔函数无法用单一的感知器表示，例如异或

感知器的表征能力（3）

- 因为所有的布尔函数都可表示为基于原子函数的互连单元的某个网络，因此感知器网络可以表示所有的布尔函数。事实上，只需要两层深度的网络，比如表示析取范式
- 注意，要把一个AND感知器的输入求反只要简单地改变相应输入权的符号
- 因为感知器网络可以表示大量的函数，而单独的单元不能做到这一点，所以我们感兴趣的是学习感知器组成的多层网络

感知器训练法则

- 虽然我们的目的是学习由多个单元互连的网络，但我们还是要从如何学习单个感知器的权值开始
- 单个感知器的学习任务，决定一个权向量，它可以使感知器对于给定的训练样例输出正确的1或-1
- 我们主要考虑两种算法
  - 感知器法则
  - delta法则
- 这两种算法保证收敛到可接受的假设，在不同的条件下收敛到的假设略有不同
- 这两种算法提供了学习多个单元构成的网络的基础

感知器法则

- 算法过程
  - 从随机的权值开始
  - 反复应用这个感知器到每个训练样例，只要它误分类样例就修改感知器的权值
  - 重复这个过程，直到感知器正确分类所有的训练样例
- 感知器训练法则

$$w_i \leftarrow w_i + \Delta w_i$$

其中

$$\Delta w_i = \eta(t - o)x_i$$

感知器法则（2）

- 为什么这个更新法则会成功收敛到正确的权值呢？
  - 一些例子
  - 可以证明（Minskey & Papert 1969）
    - 如果训练样例线性可分，并且使用了充分小的η
    - 否则，不能保证

梯度下降和delta法则

- delta法则克服感应器法则的不足，在线性不可分的训练样本上，收敛到目标概念的最佳近似
- delta法则的关键思想是，使用梯度下降来搜索可能的权向量的假设空间，以找到最佳拟合训练样例的权向量
- delta法则为反向传播算法提供了基础，而反向传播算法能够学习多个单元的互连网络
- 对于包含多种不同类型的连续参数化假设的假设空间，梯度下降是必须遍历这样的空间的所有算法的基础

梯度下降和delta法则（2）

- 把delta训练法则理解为训练一个无阈值的感知器

$$o(\vec{x}) = \vec{w} \cdot \vec{x}$$

- 指定一个度量标准来衡量假设相对于训练样例的训练误差

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

- 第6章给出了选择这种E定义的一种贝叶斯论证，在一定条件下，使E最小化的假设就是H中最可能的假设

可视化假设空间

- 图4-4
  - 根据E的定义，误差曲面是一个抛物面，存在一个单一全局最小值
- 梯度下降搜索从一个任意的初始权向量开始，然后沿误差曲面最陡峭下降的方向，以很小的步伐反复修改这个向量，直到得到全局的最小误差点

梯度下降法则的推导

- 如何发现沿误差曲面最陡峭下降的方向？
  - 通过计算E相对向量 $\vec{w}$的每个分量的导数，这个向量导数被称为E对于$\vec{w}$ 的梯度，记作 $\nabla E(\vec{w})$
  - 当梯度被解释为权空间的一个向量时，它确定了使E最陡峭上升的方向，所以这个向量的反方向给出了最陡峭下降的方向
- 梯度训练法则

$$\vec{w} \leftarrow \vec{w} + \Delta\vec{w}$$

其中，

$$\Delta\vec{w} = -\eta \nabla E(\vec{w})$$

梯度下降法则的推导（2）

- 需要一个高效的方法在每一步都计算这个梯度

$$\frac{\partial E}{\partial w_i} = \sum_{d \in D} (t_d - o_d)(-x_{id})$$

- 梯度下降权值更新法则

$$\Delta w_i = \eta \sum_{d \in D} (t_d - o_d) x_{id}$$

梯度下降法则的推导（3）

- 表4-1，训练线性单元的梯度下降算法

Gradient-Descent(training_examples, η)

training_examples中每个训练样例形式为序偶<$\vec{x}$,t>，$\vec{x}$是输入值向量，t是目标输出值，η是学习速率

- 初始化每个w_i为某个小的随机值
- 遇到终止条件之前，做以下操作
  - 初始化每个Δw_i为0
  - 对于训练样例training_examples中的每个<$\vec{x}$,t>，做
    - 把实例$\vec{x}$输入到此单元，计算输出o
    - 对于线性单元的每个权增量Δw_i，做

      $\Delta w_i \leftarrow \Delta w_i + \eta(t-o)x_i$
  - 对于线性单元的每个权w_i，做

    $w_i \leftarrow w_i + \Delta w_i$

梯度下降法则的推导（4）

- 梯度下降算法如下
  - 选取一个初始的随机权向量
  - 应用线性单元到所有的训练样例，根据公式4.7计算每个权值的w更新权值
- 因为误差曲面仅包含一个全局的最小值，所以无论训练样例是否线性可分，算法都会收敛到具有最小误差的权向量，条件是使用足够小的学习速率
- 算法的一种常用改进方法是随着梯度下降步数的增加逐渐减小学习速率

梯度下降的随机近似

- 梯度下降是一种重要的通用学习范型，它是搜索庞大假设空间或无限假设空间一种策略
- 梯度下降应用于满足以下条件的任何情况
  - 假设空间包含连续参数化的假设
  - 误差对于这些假设参数可微
- 梯度下降的主要实践问题
  - 有时收敛过程可能非常慢
  - 如果在误差曲面上有多个局部极小值，那么不能保证找到全局最小值

梯度下降的随机近似（2）

- 随机梯度下降（或称增量梯度下降）
  - 根据某个单独样例的误差增量计算权值更新，得到近似的梯度下降搜索（随机取一个样例）
  - 对表4-1算法的修改
  - 可以看作为每个单独的训练样例定义不同的误差函数
  - 在迭代所有训练样例时，这些权值更新的序列给出了对于原来误差函数的梯度下降的一个合理近似
  - 通过使下降速率的值足够小，可以使随机梯度下降以任意程度接近于真实梯度下降

梯度下降的随机近似（2）

- 标准梯度下降和随机梯度下降之间的关键区别
  - 标准梯度下降是在权值更新前对所有样例汇总误差，而随机梯度下降的权值是通过考查每个训练样例来更新的
  - 在标准梯度下降中，权值更新的每一步对多个样例求和，需要更多的计算（？）
  - 标准梯度下降，由于使用真正的梯度，标准梯度下降对于每一次权值更新经常使用比随机梯度下降大的步长
  - 如果标准误差曲面有多个局部极小值，随机梯度下降有时可能避免陷入这些局部极小值中
- 实践中，标准和随机梯度下降方法都被广泛应用

梯度下降的随机近似（3）

- delta法则（增量法则），又称LMS法则、Adaline法则、Windrow-Hoff法则
- 公式4.10与4.4.2节的感知器法则的相似和区别
- delta法则可以学习非阈值线性单元的权，也可以用来训练有阈值的感知器单元。
- 如果非阈值输出能够被训练到完美拟合这些值，那么阈值输出也会完美拟合它们
- 即使不能完美地拟合目标值，只要线性单元的输出具有正确的符号，阈值输出就会正确拟合目标值
- 尽管这个过程会得到使线性单元输出的误差最小化的权值，但这些权值不能保证阈值输出的误差最小化（？）

感知器学习小结

- 感知器法则和delta法则的关键差异
  - 前者根据阈值化的感知器输出的误差更新权值
  - 后者根据输入的非阈值化线性组合的误差来更新权值
- 这个差异带来不同的收敛特性
  - 前者经过有限次的迭代收敛到一个能理想分类训练数据的假设，条件是训练样例线性可分
  - 后者可能经过极长的时间，渐近收敛到最小误差假设，但无论训练样例是否线性可分都会收敛

感知器学习小结（2）

- 学习权向量的第3种方法是线性规划
- 线性规划是解线性不等式方程组的一种通用的有效方法
- 这种方法仅当训练样例线性可分时有解
- Duda和Hart给出了一种更巧妙的适合非线性可分的情况的方法
- 更大的问题是，无法扩展到训练多层网络，而delta法则可以很容易扩展到多层网络

多层网络和反向传播算法

- 多层网络能够表示种类繁多的非线性曲面
- 图4-5描述了一个典型的多层网络和它的决策曲面

可微阈值单元

- 使用什么类型的单元来构建多层网络？
- 多个线性单元的连接仍产生线性函数，而我们希望构建表征非线性函数的网络
- 感知器单元可以构建非线性函数，但它的不连续阈值使它不可微，不适合梯度下降算法
- 我们需要的单元满足的条件
  - 输出是输入的非线性函数
  - 输出是输入的可微函数
- Sigmoid单元，类似于感知器单元，但基于一个平滑的可微阈值函数

可微阈值单元（2）

- 图4-6
- sigmoid单元先计算它的输入的线性组合，然后应用到一个阈值上，阈值输出是输入的连续函数

$$o = (\vec{w} \cdot \vec{x})$$

其中

$$\sigma(y) = \frac{1}{1+e^{-y}}$$

可微阈值单元（3）

- sigmoid函数
  - 也称logistic函数
  - 挤压函数
  - 输出范围是0到1
  - 单调递增
  - 导数很容易用函数本身表示
- sigmoid函数的变型
  - 其他易计算导数的可微函数
  - 增加陡峭性
  - 双曲正切函数

反向传播算法

- 用来学习多层网络的权值
- 采用梯度下降方法试图最小化网络输出值和目标值之间的误差平方
- 网络的误差定义公式，对所有网络输出的误差求和

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} \sum_{k \in outpus} (t_{kd} - o_{kd})^2$$

反向传播算法（2）

- 反向传播算法面临的学习任务
  - 搜索一个巨大的假设空间，这个空间由网络中所有的单元的所有可能的权值定义，得到类似图4-4的误差曲面
  - 在多层网络中，误差曲面可能有多个局部极小值，梯度下降仅能保证收敛到局部极小值
  - 尽管有这个障碍，已经发现对于实践中很多应用，反向传播算法都产生了出色的结果

反向传播算法（3）

- 表4-2包含两层sigmoid单元的前馈网络的反向传播算法

BackPropagation(training_examples, η, n_{in}, n_{out}, n_{hidden})

training_examples是序偶$<\vec{x},\vec{t}>$的集合，$\vec{x}$是网络输入值向量，$\vec{t}$是目标输出值。η是学习速率，n_{in}是网络输入的数量，n_{hidden}是隐藏层单元数，n_{out}是输出单元数，从单元i到单元j的输入表示为x_{ji}，单元i到单元j的权值表示为w_{ji}。

- 创建具有n_{in}个输入，n_{hidden}个隐藏，n_{out}个输出单元的网络
- 初始化所有的网络权值为小的随机值
- 在遇到终止条件前
  - 对于训练样例training_examples中的每个$<\vec{x},\vec{t}>$：
    - 把输入沿网络前向传播
      - 把实例$\vec{x}$输入网络，并计算网络中每个单元u的输出o_u
    - 使误差沿网络反向传播
      - 对于网络的每个输出单元k，计算它的误差项$\delta_k \leftarrow o_k(1-o_k)(t_k-o_k)$
      - 对于网络的每个隐藏单元h，计算它的误差项$\delta_h \leftarrow o_h(1-o_h)$
      - 更新每个网络权值$w_{ji} \leftarrow w_{ji}+\Delta w_{ji}$，其中$\Delta w_{ji}=\eta\delta_j x_{ji}$

反向传播算法（4）

- 表4-2给出的反向传播算法适用于包含两层sigmoid单元的分层前馈网络，并且每一层的单元与前一层的所有单元相连。
- 表4-2是反向传播算法的增量梯度下降（或随机梯度下降）版本
- 使用的符号做了如下扩展
  - 网络中每个节点被赋予一个序号，这里的节点要么是网络的输入，要么是网络中某个单元的输出
  - x_{ji}表示节点i到单元j的输入，w_{ji}表示对应的权值
  - δ_n表示与单元n相关联的误差项。

表4-2的算法解释

- 从建立一个具有期望数量的隐藏单元和输出单元的网络并初始化所有的网络的权值为小的随机数开始
- 给定一个固定的网络结构，算法的主循环就对训练样例进行反复的迭代
- 对于每一个训练样例，它应用目前的网络到这个样例，计算出对这个样例网络输出的误差，然后更新网络中所有的权值
- 对这样的梯度下降步骤进行迭代，直到网络的性能达到可接受的精度为止

反向传播算法的梯度下降法则

- 表4-2的梯度下降权更新法则与delta训练法则相似
- 类似delta法则，依照以下三者来更新每一个权
  - 学习速率η
  - 该权值涉及的输入值x_{ji}
  - 该单元的输出误差
- 不同于delta法则的地方
  - delta法则中的误差项被替换成一个更复杂的误差项δ_j

反向传播算法的误差项

- 输出单元k的误差项
  - δ_k与delta法则中的(t_k-o_k)相似，但乘上了sigmoid挤压函数的导数$o_k(1-o_k)$。
- 隐藏单元h的误差项
  - 因为训练样例仅对网络的输出提供了目标值t_k，所以缺少直接的目标值来计算隐藏单元的误差值
  - 采取以下的间接方法计算隐藏单元的误差项：对受隐藏单元h影响的每一个单元的误差δ_k进行加权求和，每个误差δ_k权值为w_{kh}，w_{kh}就是从隐藏单元h到输出单元k的权值。这个权值刻画了隐藏单元h对于输出单元k的误差应负责的程度。

表4-2的算法解释（2）

- 表4-2的算法随着每个训练样例的出现而递增地更新权，这一点与梯度下降的随机近似算法一致
- 要取得误差E的真实梯度，需要在修改权值之前对所有训练样例的$\delta_j x_{ji}$值求和
- 在典型的应用中，权值的更新迭代会被重复上千次
- 有很多终止条件可以用来停止这个过程
  - 迭代的次数到了一个固定值时停止
  - 当在训练样例上的误差降到某个阈值以下
  - 在分离的验证样例集合上的误差符合某个标准
- 终止条件很重要，太少的迭代无法有效地降低误差，太多的迭代会导致对训练数据的过度拟合

增加冲量项

- 因为反向传播算法的应用如此广泛，所以已经开发出了很多反向传播算法的变体
- 修改权值更新法则，使第n次迭代时的权值的更新部分地依赖于发生在第n-1次迭代时的更新，比如
  - $\Delta w_{ji}(n)=\eta\delta_j x_{ji}+\alpha\Delta w_{ji}(n-1)$
- 右侧第一项就是表4-2中的权值更新法则，第二项被称为冲量项
- 梯度下降的搜索轨迹就像一个球沿误差曲面滚下，冲量使球从一次迭代到下一次迭代时以同样的方向滚动
- 冲量有时会使这个球滚过误差曲面的局部极小值或平坦区域
- 冲量也具有在梯度不变的区域逐渐增大搜索步长的效果，从而加快收敛

学习任意的无环网络

- 表4-2的算法可以简单地推广到任意深度的前馈网络
- 第m层的单元r的δ_r值由更深的第m+1层δ值根据下式计算$\delta_r = o_r(1-o_r)\sum_{s\in m+1层} w_{sr}\delta_s$
- 将这个算法推广到任何有向无环结构也同样简单，而不论网络中的单元是否被排列在统一的层上，计算任意内部单元的δ的法则是：$\delta_r = o_r(1-o_r)\sum_{s\in Downstream(r)} w_{sr}\delta_s$，Downstream(r)是在网络中单元r的直接下游单元的集合，即输入中包括r的输出的所有单元

反向传播法则的推导

- 随机梯度下降算法迭代处理训练样例，每次处理一个，对于每个训练样例d，利用关于这个样例的误差E_d的梯度修改权值

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}}$$

$$E_d(\vec{w}) = \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2$$

符号说明

- x_{ji}，单元j的第i个输入
- w_{ji}，与x_{ji}相关联的权值
- net_j，单元j的输入的加权和
- o_j，单元j计算出的输出
- t_j，单元j的目标输出
- σ，sigmoid函数
- outputs，网络最后一层的输出单元的集合
- Downstream(j)，单元j的输出到达的单元的集合

随机梯度下降法则的推导

$\frac{\partial E_d}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j}\frac{\partial net_j}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j}x_{ji}$，分情况讨论$\frac{\partial E_d}{\partial net_j}$的推导

- 输出单元

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j}\frac{\partial o_j}{\partial net_j}$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j}\frac{1}{2}\sum_{k\in outputs}(t_k - o_k)^2$$

$$= \frac{\partial}{\partial o_j}\frac{1}{2}(t_j - o_j)^2$$

$$= \frac{1}{2}2(t_j - o_j)\frac{\partial(t_j - o_j)}{\partial o_j}$$

$$= -(t_j - o_j)$$

$$\frac{\partial o_j}{\partial net_j} = \frac{\partial \sigma(net_j)}{\partial net_j} = o_j(1-o_j)$$

$$\frac{\partial E_d}{\partial net_j} = -(t_j - o_j)o_j(1-o_j)$$

$$\Delta w_{ji} = -\eta\frac{\partial E_d}{\partial w_{ji}} = \eta(t_j - o_j)o_j(1-o_j)x_{ji}$$

随机梯度下降法则的推导（２）

- 隐藏单元

$$
\begin{aligned}
\frac{\partial E_d}{\partial net_j} &= \sum_{k \in Downstream(j)} \frac{\partial E_d}{\partial net_k} \frac{\partial net_k}{\partial net_j} \\
&= \sum_{k \in Downstream(j)} -\delta_k \frac{\partial net_k}{\partial net_j} \\
&= \sum_{k \in Downstream(j)} -\delta_k \frac{\partial net_k}{\partial o_j} \frac{\partial o_j}{\partial net_j} \\
&= \sum_{k \in Downstream(j)} -\delta_k w_{kj} \frac{\partial o_j}{\partial net_j} \\
&= \sum_{k \in Downstream(j)} -\delta_k w_{kj} o_j (1 - o_j) \\
&= -o_j (1 - o_j) \sum_{k \in Downstream(j)} \delta_k w_{kj}
\end{aligned}
$$

$$
\Delta w_{ji} = -\eta x_{ji} o_j (1 - o_j) \sum_{k \in Downstream(j)} \delta_k w_{kj}
$$

收敛性和局部极小值

- 对于多层网络，误差曲面可能含有多个不同的局部极小值，梯度下降可能陷入这些局部极小值中的任何一个
- 对于多层网络，反向传播算法仅能保证收敛到误差E的某个局部极小值，不一定收敛到全局最小误差
- 尽管缺乏对收敛到全局最小误差的保证，反向传播算法在实践中仍是非常有效的函数逼近算法

收敛性和局部极小值（2）

- 网络的权越多，误差曲面的维数越多，也就越可能为梯度下降提供更多的逃逸路线
- 考虑随着训练中迭代次数的增加网络权值的演化方式
  - 如果把网络的权值初始化为接近于0的值，那么在早期的梯度下降步骤中，网络将表现为一个非常平滑的函数，近似为输入的线性函数，这是因为sigmoid函数本身在权值靠近0时接近线性
  - 仅当权值增长一定时间后，它们才会到达可以表示高度非线性网络函数的程度，可以预期在这个能表示更复杂函数的权空间区域存在更多的局部极小值
  - 但是当权到达这一点时，它们已经足够靠近全局最小值，即便它是这个区域的局部最小值也是可以接受的

收敛性和局部极小值（3）

- 用来缓解局部极小值问题的启发式规则
  - 为梯度更新法则加一个冲量，可以带动梯度下降过程，冲过狭窄的局部极小值（原则上，也可能冲过狭窄的全局最小值）
  - 使用随机的梯度下降而不是真正的梯度下降。随机近似对于每个训练样例沿一个不同的误差曲面有效下降，这些不同的误差曲面通常有不同的局部极小值，这使得下降过程不太可能陷入一个局部极小值
  - 使用同样的数据训练多个网络，但用不同的随机权值初始化每个网络。如果不同的训练产生不同的局部极小值，那么对分离的验证集合性能最好的那个网络将被选中，或者保留所有的网络，输出是所有网络输出的平均值

前馈网络的表征能力

- 布尔函数：任何布尔函数可以被具有两层单元的网络准确表示，尽管在最坏情况下所需隐藏单元的数量随着网络输入数量的增加成指数级增长。
  - 考虑下面的通用方案：对于每一个可能的输入向量，创建不同的隐藏单元，并设置它的权值使当且仅当这个特定的向量输入到网络时该单元被激活，这样就产生了一个对于任意输入仅有一个单元被激活的隐藏层，然后把输出单元实现为一个仅由所希望的输入模式激活的或门。

前馈网络的表征能力（2）

- 连续函数：每个有界的连续函数可以由一个两层的网络以任意小的误差逼近。这个结论适用于在隐藏层使用sigmoid单元、在输出层使用（非阈值）线性单元的网络。所需的隐藏单元数量依赖于要逼近的函数。
- 任意函数：任意函数可以被一个有三层单元的网络以任意精度逼近。两个隐藏层使用sigmoid单元，输出层使用线性单元，每层所需单元数不确定。
  - 证明方法：首先说明任意函数可以被许多局部化函数的线性组合逼近，这些局部化函数的值除了某个小范围外都为0；然后说明两层的sigmoid单元足以产生良好的局部逼近
- 注意：梯度下降从一个初始值开始，因此搜索范围里的网络权向量可能不包含所有的权向量

假设空间搜索和归纳偏置

- 反向传播算法的假设空间是n个网络权值形成的n维欧氏空间。这个空间是连续的，与决策树学习和其他基于离散表示的方法的假设空间不同
- 假设空间的连续性以及误差E关于假设的连续参数可微，导致了一个定义良好的误差梯度，为最佳假设的搜索提供了一个非常有用的结构。
- 精确地刻画出反向传播学习的归纳偏置是有难度的，它依赖于梯度下降搜索和权空间覆盖可表征函数空间的方式的相互作用性
- 把这一偏置粗略地刻画为在数据点之间平滑插值。如果给定两个正例，它们之间没有反例，反向传播算法会倾向于把这两点之间的点也标记为正例

隐藏层表示

- 反向传播算法的一个迷人特性是：它能够在网络内部的隐藏层发现有用的中间表示
  - 训练样例仅包含网络输入和输出，权值调节的过程可以自由地设置权值，来定义任何隐藏单元表示，这些隐藏单元表示在使误差E达到最小时最有效。
  - 引导反向传播算法定义新的隐藏层特征，这些特征在输入中没有明确表示出来，但能捕捉输入实例中与学习目标函数最相关的特征
- 多层网络在隐藏层自动发现有用表示的能力是ANN学习的一个关键特性。允许学习器创造出设计者没有明确引入的特征。
- 网络中使用的单元层越多，就可以创造出越复杂的特征

泛化、过度拟合和停止判据

- 权值更新算法的终止条件
  - 一种选择是，对训练样例的误差降低至某个预先定义的阈值之下
    - 这不是一个好的策略，因为反向传播算法容易过度拟合训练样例，降低对于其他未见实例的泛化精度
- 泛化精度：网络拟合训练数据外的实例的精度
- 图4-9，尽管在训练样例上的误差持续下降，但在验证样例上测量到的误差先下降，后上升。
  - 因为这些权值拟合了训练样例的“特异性”，而这个特异性对于样例的一般分布没有代表性。ANN中大量的权值参数为拟合这样的“特异性”提供了很大的自由度

过度拟合

- 为什么过度拟合发生在迭代的后期，而不是早期？
  - 设想网络的权值是被初始化为小随机值的，使用这些几乎一样的权值仅能描述非常平滑的决策面
  - 随着训练的进行，一些权值开始增长，以降低在训练数据上的误差，同时学习到的决策面的复杂度也在增加
  - 如果权值调整迭代次数足够多，反向传播算法可能会产生过度复杂的决策面，拟合了训练数据中的噪声和训练样例中没有代表性的特征

过度拟合解决方法

- 权值衰减
  - 它在每次迭代过程中以某个小因子降低每个权值，这等效于修改E的定义，加入一个与网络权值的总量相应的惩罚项，此方法的动机是保持权值较小，从而使学习过程向着复杂决策面的反方向偏置
- 验证数据
  - 一个最成功的方法是在训练数据外再为算法提供一套验证数据，应该使用在验证集合上产生最小误差的迭代次数，不是总能明显地确定验证集合何时达到最小误差

过度拟合解决方法（2）

- 一般而言，过度拟合是一个棘手的问题
- 交叉验证方法在可获得额外的数据提供验证集合时工作得很好，但是小训练集合的过度拟合问题更为严重
- k-fold交叉方法
  - 把训练样例分成k份，然后进行k次交叉验证过程，每次使用不同的一份作为验证集合，其余k-1份合并作为训练集合。
  - 每个样例会在一次实验中被用作验证样例，在k-1次实验中被用作训练样例
  - 每次实验中，使用上面讨论的交叉验证过程来决定在验证集合上取得最佳性能的迭代次数，然后计算这些迭代次数的均值 $\bar{i}$
  - 最后，运行一次反向传播算法，训练所有m个实例并迭代 $\bar{i}$ 次

举例：人脸识别

- 训练样例
  - 20个不同人的摄影图像
  - 每个人大约32张图像
    - 不同的表情
      - 快乐、沮丧、愤怒、中性
    - 不同的方向
      - 左、右、正前、上
    - 不同的穿戴
      - 是否带眼镜
  - 共624幅灰度图像
    - 分辨率为120x128，每个像素使用0（黑）到255（白）的灰度值描述
- 任务：学习图像中人脸的朝向

人脸识别——设计要素

- 输入编码
  - ANN的输入必然是图像的某种表示，那么设计的关键是如何编码这幅图像
  - 例如，可以对图像进行预处理，分解出边缘、亮度一致的区域或其他局部图像特征，然后把这些特征输入网络，问题是导致每幅图像有不同数量的特征参数，而ANN具有固定数量的输入单元
  - 把图像编码成固定的30x32像素的亮度值，每个像素对应一个网络输入，把范围是0到255的亮度值按比例线性缩放到0到1的区间内，以使网络输入和隐藏单元、输出单元在同样的区间取值。

人脸识别——设计要素（2）

- 输出编码
  - ANN必须输出4个值中的一个来表示输入图像中人脸的朝向
  - 可以使用单一的输出单元来编码这4种情况
  - 这里使用4个不同的输出单元，每一个对应4种可能朝向中的一种，取具有最高值的输出作为网络的预测值。称为1-of-n输出编码
- 选择1-of-n的原因
  - 为网络表示目标函数提供了更大的自由度
  - 最高值输出和次高值输出间的差异可以作为对网络预测的置信度

人脸识别——设计要素（3）

- 输出单元的目标值
  - 一个显而易见的方法，<1,0,0,0>...
  - 这里使用的方法，<0.9,0.1,0.1,0.1>...
  - 避免使用0和1作为目标值的原因
    - sigmoid单元对于有限权值不能产生这样的输出
    - 如果企图训练网络来准确匹配目标值0和1，梯度下降将会迫使权值无限增长
    - 0.1和0.9是sigmoid单元在有限权值情况下可以完成的

人脸识别——设计要素（4）

- 网络结构图
  - 网络包含多少个单元以及如何互连？
  - 最普遍的结构是分层网络，一层的每个单元向前连接到下一层的每一个单元
  - 目前采用了包含两层sigmoid单元的标准结构
  - 隐藏单元的数量
    - 3个，达到90%的精度，训练时间约5分钟
    - 30个，提高1~2个百分点，训练时间约1个小时
  - 实践发现，需要某个最小数量的隐藏单元来精确地学习目标函数，并且超过这个数量的多余的隐藏单元不会显著地提高泛化精度
  - 如果没有使用交叉验证，那么增加隐藏单元数量经常会增加过度拟合训练数据的倾向，从而降低泛化精度

人脸识别——设计要素（5）

- 学习算法的其他参数
  - 学习速率设定为0.3，冲量设定为0.3
  - 赋予这两个参数更低的值会产生大体相当的泛化精度，但需要更长的训练时间
  - 如果赋予更高的值，训练将不能收敛到一个具有可接受误差的网络
  - 适用完全的梯度下降
  - 输出单元的权值被初始化为小的随机值
  - 输入单元的权值被初始化为0
  - 训练的迭代次数的选择可以通过分割可用的数据为训练集合和验证集合来实现
  - 最终选择的网络是对验证集合精度最高的网络
  - 最终报告的精度是在没有对训练产生任何影响的第三个集合——测试集合上测量得到的

学习到的隐藏层表示

- 图中紧挨人脸图像下的4个矩形，每个矩形描绘了网络中4个输出单元中的一个权值，每个矩形中的4个小方形表示和这个输出单元关联的4个权值
- 隐藏单元的权值显示在输出单元的下边，每个隐藏单元接受所有30x32个像素输入。与这些输入关联的30x32个权值被显示在它们对应的像素的位置
- 针对每一个训练样例，梯度下降迭代100次后的网络权值显示在图的下部。
  - 如果一个人的脸是转向他的右面，那么他的亮度高的皮肤会大致与这个隐藏单元中的较大正值对齐，同时他的亮度低的头发会大致与负权值对齐，这导致此单元输出一个较大的值，同样的图像会使第3个隐藏单元输出一个接近0的值。

其他可选的误差函数

- 为权值增加一个惩罚项
  - 把一个随着权向量幅度增长的项加入到E中，这导致梯度下降搜寻较小的权值向量，从而减小过度拟合的风险，等价于使用权衰减策略

$$E(\vec{w}) = \frac{1}{2}\sum_{d\in D}\sum_{k\in outputs}(t_{kd} - o_{kd})2 + \gamma\sum_{i,j} w_{ji}^2$$

- 对误差增加一项目标函数的斜率或导数
  - 某些情况下，训练信息中不仅有目标值，而且还有关于目标函数的导数

$$E(\vec{w}) = \frac{1}{2}\sum_{d\in D}\sum_{k\in outputs}\left[(t_{kd} - o_{kd})^2 + \mu\sum_{j\in inputs}\left(\frac{\partial t_{kd}}{\partial x_d^j} - \frac{\partial o_{kd}}{\partial x_d^j}\right)^2\right]$$

其他可选的误差函数（2）

- 使网络对目标值的交叉熵最小化
  - 比如根据借贷申请者的年龄和存款余额，预测他是否会还贷，目标函数最好以申请者还贷的概率的形式输出，而不是输出明确的0和1。在这种情况下，可以证明最小化交叉熵的网络可以给出最好的概率估计。交叉熵定义如下：

$$-\sum_{d\in D} t_d \log o_d + (1-t_d)\log(1-o_d)$$

  - 第6章讨论了何时及为什么最可能的网络假设就是使交叉熵最小化的假设，并推导了相应的sigmoid单元的梯度下降权值调整法则，还描述了在什么条件下最可能的假设就是使误差平方和最小化的假设。

其他可选的误差函数（3）

- 通过权值共享改变有效误差函数
  - 把与不同单元或输入相关联的权“捆绑在一起”，强迫不同的网络权值取一致的值，通常是为了实施人类设计者事先知道的某个约束
  - 约束了假设的潜在空间，减小了过度拟合的风险
  - 实现方法，首先在共享权值的每个单元分别更新各个权值，然后取这些权值的平均，再用这个平均值替换每个需要共享的权值。
  - 被共享的权值比没有共享的权值更有效地适应一个不同的误差函数

其他可选的误差最小化过程

- 梯度下降是搜寻使误差函数最小化的假设的最通用的方法之一，但不是最高效的
- 不妨把权值更新方法看作是要决定这样两个问题：
  - 选择一个改变当前权值向量的方向（梯度的负值）
  - 选择要移动的距离（学习速率）
- 线搜索，每当选定了一条确定权值更新方向的路线，那么权更新的距离是通过沿这条线寻找误差函数的最小值来选择的
- 共轭梯度，进行一系列线搜索来搜索误差曲面的最小值，这一系列搜索的第一步仍然使用梯度的反方向，在后来的每一步中，选择使误差梯度分量刚好为0并保持为0的方向
- 像共轭梯度这样的方法对最终网络的泛化误差没有明显的影响，唯一可能的影响是，不同的误差最小化过程会陷入不同的局部最小值

递归网络

- 递归网络是有如下特征的人工神经网络
  - 适用于时序数据
  - 使用网络单元在时间t的输出作为其他单元在时间t+1的输入
  - 递归网络支持在网络中使用某种形式的有向环
- 考虑一个时序预测任务
  - 根据当天的经济指标x(t)，预测下一天的股票平均市值y(t+1)
  - 训练一个前馈网络预测输出y(t+1)，图4-11a

递归网络

- 预测y(t+1)时，考虑任意过去的时间窗内的信息，图4-11b
- 图4-11b那样的递归网络可以使用反向传播算法的简单变体来训练
- 把递归网络拷贝成几份，用不同拷贝间的连接替换掉反馈环，这个大的网络不再包含回路，所以可以直接使用反向传播算法来训练
- 实践中，我们仅保留一份递归网络和权值集合的拷贝，在训练了展开的网络后，可以取不同拷贝中权值的平均值作为最终网络的对应的权值
- 实践中，递归网络比没有反馈环的网络更难以训练，泛化的可靠性也不如后者，然而它们仍因较强的表征力而保持着重要性

动态修改网络结构

- 动态增长或压缩网络单元和单元间连接的数量
- 从一个不包含隐藏单元的网络开始，然后根据需要增加隐藏单元来增长网络，直到训练误差下降到某个可接受的水平
  - 级联相关算法，每当加入一个新的隐藏单元，它的输入包括所有原始的网络输入和已经存在的隐藏单元的输出，网络以这种方式增长，积聚隐藏单元，直到网络的残余误差下降到某个可接受的水平
  - 由于每一步仅有一层网络在被训练，级联相关算法显著减少了训练时间
  - 算法的一个实际困难是，因为算法可以无限制地增加单元，很容易过度拟合训练数据。

动态修改网络结构

- 从一个复杂的网络开始修剪掉某些连接
  - 判断某个权是否无关紧要的一种方法是看它的值是否接近0
  - 在实践中更加成功的方法是考虑这个权值的一个小的变化对误差的影响（连接的显著性）
  - 最不显著的连接被拆除，重复这个过程，直到遇到某个终止条件为止（最优脑损伤法）
- 一般而言，动态修改网络结构的方法能否稳定地提高反向传播算法的泛化精度还有待研究

小结

- 人工神经网络为学习实数值和向量值函数提供了一种实际的方法，对于连续值和离散值的属性都可以使用，并且对训练数据中的噪声具有很好的健壮性。
- 反向传播算法是最常见的网络学习算法
- 反向传播算法考虑的假设空间是固定连接的有权网络所能表示的所有函数的空间
- 包含3层单元的前馈网络能够以任意精度逼近任意函数，只要每一层有足够数量的单元。即使是一个实际大小的网络也能够表示很大范围的高度非线性函数
- 反向传播算法使用梯度下降方法搜索可能假设的空间，迭代减小网络的误差以拟合训练数据

小结（2）

- 梯度下降收敛到训练误差相对网络权值的局部极小值。只要训练误差是假设参数的可微函数，梯度下降可用来搜索很多连续参数构成的假设空间
- 反向传播算法能够创造出网络输入中没有明确出现的特征。
- 交叉验证方法可以用来估计梯度下降搜索的合适终止点，从而最小化过度拟合的风险
- 其他ANN学习算法，递归网络方法训练包含有向环的网络，级联相关算法改变权和网络结构

补充读物

- 本书其他与ANN学习相关的章节
  - 第6章给出了选择最小化误差平方和的贝叶斯论证，以及在某些情况下，用最小化交叉熵代替最小化误差平方和的方法
  - 第7章讨论了为可靠学习布尔函数所需要的训练实例数量的理论结果，以及某些类型网络的VC维
  - 第5章讨论了过度拟合和避免过度拟合的方法
  - 第12章讨论了使用以前知识来提高泛化精度的方法

补充读物（2）

- 发展历程
  - McCulloch & Pitts
  - Widrow & Hoff
  - Rosenblatt
  - Minsky & Papert
  - Rumelhart & McClelland; Parker
- 教科书
  - Duda & Hart
  - Windrow & Stearns
  - Rumelhart & McClelland

机器学习

第10章 学习规则集合

概述

- 对学习到的假设，最具有表征力的和最能为人类所理解的表示方法之一是if-then规则的集合
- 本章探索若干能学习这样的规则集合的算法
- 其中，最重要的是学习包含变量的规则集合，或称一阶Horn子句集合
- 由于一阶Horn子句集合可被解释为逻辑编程语言Prolog中的程序，学习的过程常被称为归纳逻辑编程
- 本章考察了多种学习规则集合的途径，其中一种是基于机器定理证明器中演绎算子的逆转

简介

- 在许多情况下，有必要学习一个由若干if-then规则共同定义的目标函数，比如
  - 决策树
  - 遗传算法
- 本章我们讨论一组不同的算法，它们直接学习规则集合，与前面算法有两点关键的不同
  - 可学习包含变量的一阶规则集合（一阶子句的表达能力比命题规则要强得多）
  - 使用序列覆盖算法，一次学习一个规则，以递增的方式形成最终的规则集合

简介（2）

- 一阶规则集合的例子

  if Parent(x,y) then Ancestor(x,y)

  if Parent(x,z) $\wedge$ Ancestor(z,y) then Ancestor(x,y)
  - 这个规则集合很紧凑地描述了一个递归函数，它很难用决策树或其他命题的方法来表示
- Prolog程序就是一阶规则的集合，因此一个可以学习这种规则集合的通用算法，可被看作是从样例中自动推导出Prolog程序的算法
- 一阶表示的学习系统在实践中的应用
  - 在质谱仪中学习哪一个化学药品能粘合碎片
  - 学习哪一个化学亚结构会产生诱导有机体突变的放射性物质
  - 学习有限单元网以分析物理结构中的应力

内容安排

- 先介绍能够学习命题规则集的算法（命题规则可看作不含变量的一阶规则），算法搜寻假设空间学习析取规则集合
- 将上面算法扩展到一阶规则
- 讨论归纳逻辑的两种通用途径以及归纳和演绎推理的基本关系

序列覆盖算法

- 序列覆盖算法
  - 学习一个规则，移去它覆盖的数据，再重复这一过程
- 假定已有一个子程序Learn-One-Rule，它的输入是一组正例和反例，输出是单个规则，它能够覆盖许多正例而覆盖很少的反例
- 我们要求输出的规则有较高的精确度，但不必有较高的覆盖度

序列覆盖算法（2）

- 序列覆盖算法的过程
  - 在所有可用训练样例上执行Learn-One-Rule
  - 再移去由其学到的规则覆盖的正例
  - 重复上面的过程，直到规则集覆盖正例达到希望的程度
- 序列覆盖算法按次序学习到一组规则，它们共同覆盖了全部正例
- 规则集中的规则可排序，分类新实例时可先应用精度最高的规则

表10-1 学习析取规则集的序列覆盖算法（CN2）

Sequential-Covering(Target_attribute, Attributes, Examples, Threshold)

- Learned_rules←{}
- Rule←Learn-One-Rule(Target_attribute, Attributes, Examples)
- 当Performance(Rule, Examples)>Threshold
  - Learned_rules←Learned_rules+Rule
  - Examples←Examples-{被Rule正确分类的样例}
  - Rule←Learn-One-Rule(Target_attribute, Attributes, Examples)
- Learned_rules←按照在Examples上的Performance排序的Learned_rules
- 返回Learned_rules

序列覆盖算法（3）

- 序列覆盖算法将问题化简为一系列简单的问题，执行的是一种贪婪搜索，它不能保证找到能覆盖样例的最小或最佳规则集
- 下面重点讨论Learn-One-Rule的设计，我们希望算法能够得到较高精度的规则集，但不必覆盖所有的正例

一般到特殊的柱状搜索

- 一种方法是，将假设空间搜索过程设计为与ID3算法中相似的方式，但在每一步只沿着最有希望的分支进行，即产生最佳性能的属性-值对，而不是用增长子树的办法覆盖所选属性的所有可能值
- 与ID3类似，可定义最佳分支，它覆盖的样例有最低的熵
- 与其他贪婪算法一样，上面算法的缺陷是，它的每一步都可能做出次优的选择
- 用柱状搜索来减小风险，即每一步保留k个最佳候选分支，每一步对k个候选分支进行处理，然后再将结果集削减至k个最可能成员

表10-2 Learn-One-Rule的一种实现：一般到特殊柱状搜索

Learn-One-Rule(Target_attribute, Attributes, Examples, k)

- 初始化Best_hypothesis为最一般的假设ϕ
- 初始化Candidate_hypotheses为集合{Best_hypothesis}
- 当Candidate_hypotheses不空，做以下操作
  - 生成下一个更特殊的候选假设
    - All_constraints←所有形式为(a=v)的约束集合，其中，a为Attributes的成员，v为出现在当前Examples集合中的a的值
    - New_candidate_hypotheses←
      - 对Candidate_hypotheses中每个h，循环
        - » 对All_constraints中每个c，循环
          通过加入约束c创建一个h的特化式
    - New_candidate_hypotheses中移去任意重复的、不一致的或非极大特殊化的假设
  - 更新Best_hypothesis
    - 对New_candidate_hypotheses中所有h做以下操作
      - 如果 Performance(h,Examples,Target_attribute)>Performance(Best_hypothesis,Examples,Target_attribute)
        则Best_hypothesis←h
  - 更新Candidate_hypotheses
    - Candidate_hypotheses←Candidate_hypotheses中k个Performance最佳成员
- 返回如下形式的一个规则
  - “如果Best_hypothesis”，则prediction
  - 其中，prediction为在与Best_hypothesis匹配的Examples中最频繁的Target_attribute值

表10-2 Learn-One-Rule的一种实现：一般到特殊柱状搜索（2）

Performance(h, Examples, Target_attribute)

- h_examples←与h匹配的Examples子集
- 返回-Entropy(h_examples)，Entropy是关于Target_attribute的熵

对表10-2的Learn-One-Rule算法的说明

- 算法主循环中考虑的每个假设都是属性-值约束的合取
- 每个合取假设对应于待学习规则的候选前件集合，它由其覆盖的样例的熵来评估
- 搜索算法不断特化候选假设，直到得到一个极大特殊假设，它包含所有可用的属性
- 规则的后件在算法的最后一步产生，在其前件确定后，算法构造出的规则后件用于预测在前件所能覆盖的样例中最常见的目标属性值
- 尽管使用了柱状搜索，贪婪搜索仍可能产生次优的规则

序列覆盖算法的几种变型

- 只学习覆盖正例的规则，对该规则没有覆盖的实例默认地赋予其反例分类
  - 正例在整个群体中所占比例很小，所以规则集只标定正例的类别，而对其他样例默认分类为反例，这样规则集更简洁易懂
  - 这一方法对应于Prolog中的失败否定策略，其中不能证明为真的表达式都默认为假
  - 需要修改Learn-One-Rule算法
    - 增加输入变量，指定感兴趣的目标值
    - Performance使用假设覆盖正例的比例

序列覆盖算法的几种变型（2）

- AQ算法
  - AQ明确地寻找覆盖特定目标值的规则，然后，对每个目标值学习一个析取规则集
  - AQ算法学习单个规则的方法也不同于Learn-One-Rule，当它对每个规则执行一般到特殊柱状搜索时，他围绕单个正例来进行
  - 搜索中只考虑被某正例满足的属性，以得到逐渐特殊的假设，每次学一个新规则时，它从那些未覆盖的样例中也选择一个新的正例，以指引新析取项的搜索

学习规则集：小结

- 串行与并行的差异
  - 序列学习算法（CN2）每次学习一个规则，而ID3每一步并行学习整个析取项的集合，ID3可称为并行覆盖算法
  - ID3在每一搜索步中根据它对数据产生的划分选择不同的属性，CN2选择的是不同的属性-值对
  - 为了学习到n个规则的集合，每个规则前件包含k个属性值测试，CN2需要nk次基本搜索步，而ID3独立选择次数要少得多
  - CN2需要较大数量的训练数据

学习规则集：小结（2）

- 搜索方向的差异
  - Learn-One-Rule的搜索方向是从一般到特殊，而其他算法是从特殊到一般
  - 从一般到特殊的一个优点是只有一个极大一般假设可作为搜索起始点
  - 而多数假设空间中有很多特殊假设，因此有许多极大特殊假设，从特殊到一般的算法难以确定搜索的起点

学习规则集：小结（3）

- 生成再测试与样例驱动搜索的差异
  - 样例驱动搜索算法包括：Find-S、候选消除、AQ算法、Gigol
  - 样例驱动算法中，对假设的生成或修正是由单独的训练样例驱动的，而且结果是一个已修正的假设，它对此单个样例的性能得到改善
  - Learn-One-Rule是生成再测试搜索
  - 生成再测试搜索方法中，后续的假设的生成只基于假设表示的语法，然后基于这些假设在全部样例上的性能来进行选择
  - 生成再测试的一个优点是搜索中每一步的选择都基于在许多样例上的假设性能，因此噪声数据的影响被最小化

学习规则集：小结（4）

- 规则的后修剪和后修剪的方法
  - Learn-One-Rule也有可能形成过度拟合，解决的方法也可以是后修剪
- 性能函数的定义
  - 相对频率：令n代表规则所匹配的样例数目，n_c代表其中它能正确分类的数目，则规则性能的相对频率估计为
    $$\frac{n_c}{n}$$
  - 精度的m-估计：令p为从整个数据集中随机抽取的样例与该规则赋予的分类相同的先验概率，令m为权，或称对此先验概率p进行加权的等效样例数目
    $$\frac{n_c + mp}{n + m}$$
  - 熵：令S为匹配规则前件的样例集合，熵衡量的是该样例集合中目标函数的均一性
    $$Entropy(S) \equiv \sum_{i=1}^{c} -p_i \log_2 p_i$$

学习一阶规则

- 本节考虑带有变量的规则，即一阶Horn子句，它们比命题规则有强得多的表征能力
- 一阶规则的归纳学习通常被称为归纳逻辑编程，因为这一过程可看作从样例中自动推导出Prolog程序
- 命题规则过于特殊了，不能描述属性值之间的实质关系，对今后的分类几乎不起作用，一阶规则能够表示更一般的规则
- 一阶Horn子句还可指定前件中的变量不出现在后件中的规则，这种变量可以被存在量词或全称量词修饰
- 还可能在规则的后件和前件中使用相同的谓词描述递归的规则

术语

- 所有的一阶表达式由常量、变量、谓词符号以及函数符号组成
- 谓词和函数的区别是谓词只能取真或假（特殊的函数），而函数的取值可为任意常量
- 通常用小写符号表示函数，大写符号表示谓词
- 术语：
  - 项：任意常量、任意变量、应用到任意项上的任意函数
  - 文字：应用到项上的任意谓词或其否定，如果包含否定符号，称为负文字，否则称为正文字，不包含任何变量的称为基本文字
  - 子句：多个文字的任意析取，其中所有的变量假定是全称量化的
  - Horn子句：至多包含一个正文字的子句，Horn子句的前件被称为子句体或子句先行词，后件被称为子句头或子句推论
  - 置换：将文字L的某些变量替换为某些项的函数θ，记为Lθ。如果置换θ使得L1θ=L2θ，那么θ称为L1和L2的合一置换

学习一阶规则集：FOIL

- FOIL学习的规则类似Horn子句，但有两个不同：
  - 比Horn子句更受限，因为文字不允许包含函数符号
  - 比Horn子句更有表征力，因为规则体中的文字可以是负文字

表10-4 基本的FOIL算法

FOIL(Target_predicate, Predicates, Examples)

- Pos←Examples中Target_predicate为True的成员
- Neg←Examples中Target_predicate为False的成员
- Learned_rules←{}
- 当Pos不空，学习NewRule
  - NewRule←没有前件的谓词Target_predicate规则
  - NewRule←Neg
  - 当NewRuleNeg不空，增加新文字以特化NewRule
    - Candidate_literature←对NewRule生成候选新文字，基于Predicate
    - Best_literal← $\arg\max_{L \in Candidate\_literals} Foil\_Gain(L, NewRule)$
    - 把Best_literal加入到NewRule的前件
    - NewRuleNeg←NewRuleNeg中满足NewRule前件的子集
  - Learned_rules←Learned_rules+NewRule
  - Pos←Pos-{被NewRule覆盖的Pos成员}
- 返回Learned_rules

FOIL算法的解释

- FOIL外层循环中每次加入一条新的规则到其析取式假设Learned_rules中去
- 每个新规则的效果是通过加入一个析取项泛化当前的析取假设
- 这是假设空间的特殊到一般的搜索过程，它开始于最特殊的空析取式，在假设足够一般以至覆盖所有正例时终止
- FOIL内层循环执行的是细粒度较高的搜索，以确定每个新规则的确切定义
- 内层循环在另一假设空间中搜索，它包含文字的合取，以找到一个合取式形成规则的前件
- 内层循环执行一般到特殊的爬山搜索，开始于最一般的前件，然后增加文字以使规则特化直到避开所有的反例

FOIL算法的解释（2）

- FOIL与前面的序列覆盖和Learn-One-Rule算法之间有两个最实质的不同，来源于算法对一阶规则处理的需求
  - 在学习每个新规则的一般到特殊搜索中，FOIL使用了不同的细节步骤来生成规则的候选特化式，这一不同是为了处理规则前件中含有的变量
  - FOIL使用的性能度量FOIL_Gain不同于表10-2中的熵度量，这是为了区别规则变量的不同约束，以及由于FOIL只搜寻覆盖正例的规则

FOIL中的候选特化式的生成

- FOIL生成多个不同的新文字，每个可被单独地加到规则前件中
- 更精确地，假定当前规则为：

  $P(x_1,...,x_k) \leftarrow L_1...L_n$

  FOIL生成该规则的候选特化式的方法是考虑符合下列形式的新文字L_{n+1}
  - $Q(v_1,...,v_r)$，其中Q为在Predicates中出现的任意谓词名，v_i既可为新变量，也可为规则中已有的变量，至少有一个是当前规则中已有的
  - $Equal(x_j,x_k)$，其中x_j和x_k为规则中已有的变量
  - 以上两种文字的否定

FOIL中的候选特化式的生成（2）

- 举例：待学习的规则的目标文字是GrandDaughter(x,y)，即

  GrandDaughter(x,y)←
  - 生成下列候选文字
    - Equal(x,y)
    - Female(x)
    - Female(y)
    - Father(x,y)
    - Father(y,x)
    - Father(x,z)
    - Father(y,z)
    - Father(z,y)
    - 上面文字的否定
  - 假定FOIL贪婪地选择了Father(y,z)作为最有希望的文字，得到一个较特殊的规则

    GrandDaughter(x,y)←Father(y,z)

FOIL中的候选特化式的生成（3）

- 生成进一步特化该规则的候选文字
  - Female(z)
  - Equal(z,x)
  - Equal(x,z)
  - Father(z,w)
  - Father(w,z)
  - 上面文字的否定
- 如果FOIL选择了Father(z,x)，然后又选择了Female(y)，将得到下面的规则，它只覆盖正例，因此终止了进一步搜索该规则的特化式的过程

  GrandDaughter(x,y)←Father(y,z)∧Father(z,x)∧Female(y)
- FOIL移去被该规则覆盖的所有样例，如果还有未覆盖的正例，算法将开始搜索下一个规则

引导FOIL的搜索（选择文字）

- 要在每一步中从候选文字中选择最有希望的文字，FOIL在训练数据上测量规则的性能
- 在此过程中，考虑当前规则中每个变量的可能的约束
- FOIL使用评估函数以估计增加新文字的效用，它基于加入新文字前后的正例和反例的约束数目
- 考虑某个规则R和一个可能被加到R的规则体的候选文字L，令R’为加入文字L到规则R后生成的规则，则

$$Foil\_Gain(L,R) = t\left(\log_2 \frac{p_1}{p_1+n_1} - \log_2 \frac{p_0}{p_0+n_0}\right)$$

- 按照信息论理论，Foil_Gain(L,R)可看作，为了编码R的所有正例约束的分类所需的全部位数由于L带来的减少

学习递归规则集

- 如果在表10-4的算法输入参数Predicates中包含目标谓词（即规则头的谓词），那么FOIL在生成候选文字时必须考虑它，这允许产生递归的规则
- 递归规则是否能被FOIL发现，取决于它在FOIL的贪婪搜索中是否比其他候选评分更高
- 避免在学习规则集中产生无限递归

FOIL小结

- FOIL扩展了CN2的序列覆盖算法，执行一般到特殊的搜索，每步增加一个新的文字到规则前件中，新文字可是规则前件或后件中已有的变量，或为新变量
- FOIL在每一步使用Foil_Gain函数在候选新文字中进行选择，如果新文字可指向目标谓词，那么FOIL可能学习到递归规则
- 在训练数据无噪声的情况下，FOIL可持续地增加新文字到规则中，直到不覆盖任何反例
- 为处理有噪声数据，搜索的终止需要在规则精度、覆盖度和复杂性之间做出折中
- FOIL使用最小描述长度的方法终止规则增长，新的文字只在它们的描述长度短于它们所解释的数据的描述长度时才被加入

作为逆演绎的归纳

- 归纳逻辑编程，基于一个简单的事实：归纳是演绎的逆过程
- 一般来说，机器学习涉及的是如何建立能解释观察数据的理论，给定某些数据D和一些不完整的背景知识B，学习过程被描述为生成一个假设h，它与B一起解释了D，即

$$(\forall < x_i, f(x_i) > \in D)(B \wedge h \wedge x_i) \Rightarrow f(x_i)$$

- 基于背景知识扩展谓词集合的过程，称为建设性归纳
- 可以利用演绎推理的逆过程，以使归纳泛化的过程自动化

作为逆演绎的归纳（2）

- 我们感兴趣的是设计一个逆涵蕴算子O(B,D)，它使用训练数据D和背景知识B作为输入，输出一个满足式子10.2的假设h
- 满足式子10.2的假设很多，一般依赖最小描述长度准则来进行选择
- 逆演绎的归纳方法有许多有吸引力的特点：
  - 包含了一种普遍的学习定义方法，即寻找某个一般概念，它与给定的训练样例相拟合
  - 通过引入背景知识，可以对一个假设何时可被称作“拟合”训练数据进行更充分的定义
  - 通过引入背景知识B，该公式要求学习算法使用这一背景信息来引导h的搜索，而不是只搜索语法上合法的假设空间

作为逆演绎的归纳（3）

- 不能处理有噪声的数据，不允许在观察到实例xi和其目标值f(xi)中出现差错的可能性，这样的差错可能产生对h的不一致约束，但是多数形式逻辑框架完全没有处理不一致断言的能力
- 一阶逻辑语言的表征力太强，而且满足式子10.2的假设很多，以至于假设空间的搜索在一般情形下难以执行
- 尽管直觉上背景知识有助于限制假设的搜索，但在多数ILP系统中，搜索的复杂度随着背景知识的增加而增高

逆归纳

- 自动演绎的一般方法是Robinson提出的归纳规则，归纳规则是一阶逻辑中一个合理且完备的演绎推理规则
- 算法Gigol通过反转归纳规则来形成逆涵蕴算子
- 命题归纳规则是 $\dfrac{P \vee L \quad \neg L \vee R}{P \vee R}$
- 归纳算子
  - 给定初始子句C1和C2，从子句C1中寻找一个文字L，并且¬L出现在C2中
  - 通过合并C1和C2中的除了L和¬L的所有文字，形成归纳式C，即C=(C1-{L})∪(C2-{¬L})
- 给定两个子句，归纳算子首先确定文字L是否以正文字出现在一个子句中，以负文字出现在另一个子句中，然后使用上面的归纳规则，见图10-2

逆归纳（2）

- 归纳算子根据初始子句C1和C2，得到归纳式C，逆涵蕴算子O(C,C1)根据C和C1得到另一个初始子句C2
- 逆归纳是不确定的，即可能有多个子句C2，使得C1和C2产生C，在其中进行选择的一个启发式方法是偏好更短的子句，即假定C2和C1没有共同的文字
- 逆归纳算子
  - 给定初始子句C1和C，寻找一个文字L，它出现在子句C1中但不出现在C中
  - 通过包含下列的文字，形成第二个子句C2=(C-(C1-{L}))∪{¬L}

逆归纳（3）

- 可以基于逆归纳这样的逆涵蕴算子开发出规则学习算法来
  - 一种策略是使用序列覆盖算法，循环地以这种方法学习Horn子句集
  - 每次循环中，算法选择没有被以前学习到的子句覆盖的一个训练样例<xi,f(xi)>，然后应用归纳规则来生成满足$(B \wedge hi \wedge xi) \Rightarrow f(xi)$的候选假设hi
  - 这是一个样例驱动的搜索，因为每个候选假设的建立是为了覆盖一特定样例
  - 如果存在多个候选假设，那么选择的策略是选取在其他样例上也有最高精度的假设
- Gigol程序使用了结合这种序列覆盖算法的逆归纳，通过它与用户交互，获得训练样例并引导它在可能的归纳推理步骤的巨大空间中的搜索，
- Gigol使用了一阶表示而不是命题表示

一阶归纳

- 归纳规则可以很容易地扩展到一阶表示，与命题归纳的关键不同是：这个过程基于合一置换操作
- 定义置换为变量到项的任意映射，符号Wθ表示应用置换θ到表达式W上的结果
- 置换的例子
  - L=Father(x,Bill)
  - θ={x/Bob, y/z}
  - Lθ=Father(Bob,Bill)
- 合一置换：如果L1θ=L2θ，则θ为两个文字L1和L2的合一置换

一阶归纳（2）

- 合一置换的意义在于可用来推广命题的归纳规则，在一阶归纳中，从子句C1中寻找一文字L1和在C2中寻找一文字L2，使得存在L1和¬L2的合一置换，则归纳式计算方法如下

  $C=(C1-\{L1\})\theta\cup(C2-\{L2\})\theta$
- 表10-7一阶形式的归纳规则
  - 寻找C1中的文字L1，C2中的文字L2，以及置换θ，使得$L1\theta=\neg L2\theta$
  - 通过包含C1θ和C2θ中，除了L1θ和¬L2θ以外的文字，形成归纳式C，见式子10.3

一阶归纳（3）

- 一阶形式的归纳的举例
  - C1=White(x)←Swan(x), C2=Swan(Fred)
  - 首先表示成子句形式C1=White(x)∨¬Swan(x)
  - 找到C1中的文字L1=Swan(x)和C2中的文字L2=Swan(Fred)，存在它们的合一置换θ={x/Fred}
  - 因此归纳结果C=White(Fred)∪ϕθ=White(Fred)

一阶情况的逆归纳

- 式子10.3中的合一置换θ可被为唯一地分解为θ1和θ2，即θ=θ1θ2，θ1包含涉及C1中变量的所有置换，θ2包含涉及C2中变量的所有置换
- 由于C1和C2是全称量化陈述，可以使用不同的变量名，因此上面的分解是合理的
- 使用这种分解，式子10.3可重新表达为：$C=(C1-\{L1\})\theta1\cup(C2-\{L2\})\theta2$
- 使用归纳规则的定义$L2=\neg L_1\theta_1\theta_2^{-1}$，解出C2为$C_2=(C-(C_1-\{L_1\}\theta_1)\theta_2\cup\{\neg L_1\theta_1\theta_2^{-1}\}$

一阶情况的逆归纳（2）

- 式子10.4表示的逆涵蕴算子是非确定的，在应用过程中，一般可找到待归纳的子句C1和置换θ1和θ2的多种选择，每组不同的选择都产生一个不同的C2

一阶情况的逆归纳（3）

- 图10-3显示了此逆归纳规则应用在一个简单例子上的多个步骤
  - 训练数据D=GrandChild(Bob, Shannon)
  - 背景知识B={Father(Shanon,Tom), Father(Tom, Bob)}
  - 学习目标谓词GrandChild(y,x)的规则
    - 第一步
      - 设置结论C为训练样例GrandChild(Bob, Shannon)
      - 从背景知识中选择子句C1=Father(Shannon, Tom)
      - 选择逆置换θ_1^{-1}={}，θ_2^{-1}={Shannon/x}
      - 得到C2=GrandChild(Bob,x)∨¬Father(x,Tom)
    - 第二步
      - 设置结论为上步得到的C=C2
      - 从背景知识中选择C1=Father(Tom, Bob)
      - 得到GrandChild(y,x)∨¬Father(x,z)∨¬Father(z,y)

逆归纳小结

- 逆归纳提供了一种一般的途径以自动产生满足式子10.2的假设
- 与FOIL方法的差异
  - 逆归纳是样例驱动，FOIL是生成再测试
  - 逆归纳在每一步只考虑可用数据中的一小部分，FOIL考虑所有的可用数据
  - 看起来基于逆归纳的搜索更有针对性且更有效，但实际未必如此

泛化、θ-包容于和涵蕴

- 考虑如下的定义
  - more_general_than：给定两布尔值函数hj(x)和hk(x)，称$h_j \geq_g h_k$，当且仅当$(\forall x)h_k(x) \rightarrow h_j(x)$（参见第2章）
  - θ-包容于：考虑两个子句Cj和Ck，称Cj θ-包容于Ck，当且仅当存在一个置换θ，使得$C_j\theta \subseteq C_k$
  - 涵蕴：考虑两个子句Cj和Ck，子句Cj被称为涵蕴Ck，当且仅当Ck从Cj中演绎派生

泛化、θ-包容于和涵蕴（2）

- 三个定义之间的关系
  - 泛化和θ-包容于的关系：如果$h1 \geq_g h2$，则子句C1:c(x)←h1(x) θ-包容于子句C2:c(x)←h2(x)
  - θ-包容于是涵蕴的一种特殊形式，即如果子句A θ-包容于子句B，则A⇒B，然而反之不一定成立
  - 因此，泛化是θ-包容于的一种特殊情况，而θ-包容于又是涵蕴的特殊情况
- 通过泛化和特化假设来搜索假设空间比用一般的逆涵蕴算子来搜索更局限
- 遗憾的是，逆涵蕴这种最一般的形式可产生无法处理的搜索
- 中间的θ-包容于提供了位于泛化和涵蕴之间的一种概念和方法

Progol

- 在实践中，逆归纳容易导致候选假设的组合爆炸
- 另一种途径（Progol的思想）是：
  - 只使用逆涵蕴来生成一个最特殊假设，它与背景信息一起涵蕴观察的数据
  - 然后，这个最特殊假设可用于确定假设空间的一般到特殊搜索边界，只考虑比此边界更一般的假设

Prolog（2）

- Progol使用的算法概述
  - 用户指定使用一个受限的一阶表示语言为假设空间H，
  - Progol使用序列覆盖法来从H中学习一组覆盖数据的表达式
  - Progol在这个由最一般假设和第2步中得到的特殊边界hi所界定的假设空间中执行了一般到特殊搜索，在此假设集合中，它寻找有最小描述长度的假设

小结

- 序列覆盖算法学习析取的规则集，方法是先学习单个精确的规则，然后移去被此规则覆盖的正例，再在剩余样例上重复这一过程
- 序列覆盖算法提供了一个学习规则集的有效的贪婪算法，可作为由顶向下的决策树学习算法的替代算法，决策树算法可被看作并行覆盖，与序列覆盖相对应
- 在序列覆盖算法中，已研究了多种方法以学习单个规则，这些方法的不同在于它们考察规则前件空间的策略不同
- 一阶规则集提供了一种表征能力很强的表示，学习一阶Horn子句的问题常被称为归纳逻辑编程的问题

小结（2）

- 学习一阶规则集的方法是将CN2中的序列覆盖算法由命题形式扩展到一阶表示，体现在FOIL程序中，它可学习包括简单递归规则集在内的一阶规则集
- 学习一阶规则集的另一个方法是逆归纳，通过运用熟知的演绎推理的逆算子来搜索假设
- Cigol使用的逆归纳是归纳算子的逆转，而归纳是普遍用于机器定理证明的一种推理规则，Progol结合了逆涵蕴策略和一般到特殊策略来搜索假设空间

补充读物

- Winston1970，学习如“arch”这样的概念的网络式描述
- Banerji1964,1969，将逻辑表示用于学习问题的研究
- Michalski，AQ算法
- Plotkin1970，θ-包容于的定义，对归纳和演绎之间的关系进行了形式化
- Vere1975，研究了学习的逻辑表示问题
- Buchanan1976，Meta-Dendral程序可学习分子结构中可在质谱仪中被分割的部分的关系描述
- Mitchell1979，候选消除变型空间算法被用于化学结构的关系描述

补充读物（2）

- Horn子句的研究
  - Shapiro1983，MIS
  - Sammut & Banerji1986，MARVIN
  - Quinlan1990，FOIL
  - Dzerosk1991，mFOIL
  - Pazzani et al.1991，FOCL
  - De Raedt & Bruynooghe1993，CLAUDIEN
  - Grobelnik1992，MARKUS
  - Muggleton & Buntine1988，逆涵蕴

补充读物（3）

- 归纳逻辑编程
  - Lavrac & Dzeroski，一个可读性很强的教材
  - Wrobel1996，一个好的综述
  - Bratko & Muggleton1995，概述了ILP在一些重要问题上的应用

机器学习

第9章 遗传算法

概述

- 遗传算法是一种大致基于模拟进化的学习方法
- 假设通常被描述为二进制位串，也可以是符号表达式或计算机程序
- 搜索合适的假设从若干初始假设的群体或集合开始
- 当前群体的成员通过模拟生物进化的方式来产生下一代群体，比如随机变异和交叉
- 每一步，根据给定的适应度评估当前群体中的假设，而后使用概率方法选出适应度最高的假设作为产生下一代的种子
- 遗传算法已被成功用于多种学习任务和最优化问题中，比如学习机器人控制的规则集和优化人工神经网络的拓扑结构和学习参数
- 本章主要介绍了基于位串描述假设的遗传算法和基于计算机程序描述假设的遗传编程

动机

- 遗传算法（GA）是一种受生物进化启发的学习方法，它不再是从一般到特殊或从简单到复杂地搜索假设，而是通过变异和重组当前已知的最好假设来生成后续的假设
- 每一步，更新被称为当前群体的一组假设，方法是使用当前适应度最高的假设的后代替代群体的某个部分
- 这个过程形成了假设的生成测试的柱状搜索，其中若干个最佳当前假设的变体最有可能在下一步被考虑

动机（2）

- 遗传算法的普及和发展得益于下面的因素
  - 在生物系统中，进化被认为是一种成功的自适应方法，具有很好的健壮性
  - 遗传算法搜索的假设空间中，假设的各个部分相互作用，每一部分对总的假设适应度的影响难以建模
  - 遗传算法易于并行化
- 本章内容安排
  - 描述了遗传算法，举例演示了它的用法，分析了它搜索的空间的特性
  - 描述了遗传算法的一个变体：遗传编程，这个方法中，整个计算机程序向着某个适应度准则进化
  - 介绍了一些有关生物进化的课题（遗传算法和遗传编程是进化计算领域中的两种普遍方法），比如鲍德温效应，它描述了个体的学习能力与整个群体进化速度之间的相互作用

遗传算法

- 遗传算法研究的问题是搜索候选假设空间并确定最佳假设
- 最佳假设被定义为使适应度最优的假设
  - 适应度是为当前问题预先定义的数字度量，比如：
    - 如果学习任务是在给定一个未知函数的输入输出训练样例后逼近这个函数，适应度可被定义为假设在训练数据上的精度
    - 如果是学习下国际象棋的策略，适应度可被定义为该个体在当前群体中与其他个体对弈的获胜率

遗传算法（2）

- 遗传算法具有以下的共同结构：
  - 算法迭代更新一个假设池，这个假设池称为群体
  - 在每一次迭代中，根据适应度评估群体中的所有成员，然后用概率方法选取适应度最高的个体产生新一代群体
  - 在被选中的个体中，一部分保持原样地进入下一代群体，其他被用作产生后代个体的基础，其中应用交叉和变异这样的遗传方法

表9-1 遗传算法原型

- GA(Fitness, Fitness_threshold, p, r, m)

  Fitness：适应度评分函数

  Fitness_threshold：指定终止判据的阈值

  p：群体中包含的假设数量

  r：每一步中通过交叉取代群体成员的比例

  m：变异率

  - 初始化群体：P←随机产生的p个假设
  - 评估：对于P中每个假设h，计算Fitness(h)
  - 当 $\lfloor \max_h Fitness(h) \rfloor$ <Fitness_threshold，产生新一代P_S，做：
    - 选择：用概率方法选择P的(1-r)p个成员加入PS，概率公式是

$$\Pr(h_i) = \frac{Fitness(h_i)}{\sum_{j=1}^{p} Fitness(h_j)}$$

    - 交叉：按概率从P中选择rp/2对假设，对于每对假设<h1,h2>，应用交叉算子产生两个后代，把所有的后代加入PS
    - 变异：使用均匀的概率从PS中选择m%的成员，应用变异算子
    - 更新：P←PS
    - 评估：对于P中每个h计算Fitness(h)
  - 从P中返回适应度最高的假设

遗传算法（3）

- 算法的每一次迭代以3种方式产生新一代群体
  - 直接从当前群体中选择
  - 在选中的个体中进行交叉操作
  - 在新群体上进行变异操作
- 遗传算法执行一种随机的、并行柱状的搜索，根据适应度函数发现好的假设

表示假设

- 遗传算法中的假设常常被表示成二进制位串，这便于用变异和交叉遗传算子来操作
- 把if-then规则编码成位串
  - 首先使用位串描述单个属性的值约束
    - 比如考虑属性Outlook，它的值可以取以下3个中的任一个：Sunny、Overcast、Rain，因此一个明显的方法是使用一个长度为3的位串，每位对应一个可能值，若某位为1，表示这个属性可以取对应的值
  - 多个属性约束的合取可以很容易地表示为对应位串的连接
  - 整个规则表示可以通过把描述规则前件和后件的位串连接起来

表示假设（2）

- 位串的特点
  - 表示规则的位串对假设空间中的每个属性有一个子串，即使该属性不被规则的前件约束。
  - 得到一个固定长度的规则位串表示，其中特定位置的子串描述对应特定属性的约束
- 规则集的表示：单个规则的位串表示连接起来
- 有必要让每个句法合法的位串表示一个有意义的假设
- 假设也可以用符号描述来表示，而不是位串，比如计算机程序

遗传算子

- 遗传算法使用一系列算子来决定后代，算子对当前群体中选定的成员进行重组
- 表9-1列出了用来操作位串的典型遗传算法算子，它们是生物进化中的遗传过程的理想化形式
- 最常见的算子是交叉和变异
- 交叉：
  - 从两个双亲串中通过复制选定位产生两个新的后代，每个后代的第i位是从它的某个双亲的第i位复制来的
  - 双亲中的哪一个在第i位起作用，由另一个称为交叉掩码的位串决定：
    - 单点交叉：前n位来自第一个双亲，余下的位来自第二个双亲
    - 两点交叉：用一个双亲的中间片断替换第二个双亲的中间片断
    - 均匀交叉：合并了从两个双亲以均匀概率抽取的位

遗传算子（2）

- 变异：
  - 从单一双亲产生后代，对位串产生随机的小变化，方法是选取一个位，然后取反
  - 变异经常是在应用交叉之后
- 其他算子
  - 使规则特化的算子
  - 直接泛化

适应度函数和假设选择

- 适应度函数定义了候选假设的排序准则
  - 如果学习任务是分类的规则，那么适应度函数中会有一项用来评价每个规则对训练样例集合的分类精度，也可包含其他的准则，比如规则的复杂度和一般性
- 选择假设的概率计算方法
  - 适应度比例选择（或称轮盘赌选择），选择某假设的概率是通过这个假设的适应度与当前群体中其他成员的适应度的比值得到的
  - 锦标赛选择，先从当前群体中随机选取两个假设，再按照事先定义的概率p选择适应度较高的假设，按照概率1-p选择适应度较低的假设
  - 排序选择，当前群体中的假设按适应度排序，某假设的概率与它在排序列表中的位置成比例，而不是与适应度成比例

举例

- 遗传算法可以被看作是通用的最优化方法，它搜索一个巨大的候选假设空间，根据适应度函数查找表现最好的假设
- 遗传算法尽管不能保证发现最优的假设，但一般能够发现具有较高适应度的假设
- 遗传算法的广泛应用
  - 电路布线
  - 任务调度
  - 函数逼近
  - 选取人工神经网络的拓扑结构

Gabil系统

- Dejong et al.1993的Gabil系统：遗传算法在概念学习方面的应用
- 学习以命题规则的析取集合表示的布尔概念
- 在对几个概念学习问题的实验中发现，在泛化精度方面Gabil与其他学习算法大体相当
- Gabil使用的算法就是表9-1描述的典型算法

Gabil系统（2）

- Gabil的具体实现
  - 表示：每个假设对应一个命题规则的析取集，按照9.2.1节描述的方法编码
  - 遗传算子：
    - 变异：使用表9-2中的标准变异算子
    - 交叉：表9-2中的两点交叉算子的一个扩展，这种方法保证了产生的位串表示的规则集是良定义的
  - 适应度函数：每个规则集的适应度是根据它在训练数据上的分类精度计算的，$Fitness(h)=(correct(h))^2$

Gabil系统的扩展

- 增加两个新算子
  - AddAlternative，它泛化对某个特定属性的约束，方法是把这个属性对应的子串中的一个0改为1，使用概率为0.01
  - DropCondition，把一个特定属性的所有位都替换为1，使用概率为0.60
- 两种使用新算子的方法
  - 对每一代群体中的每个假设以同样的概率应用
  - 对假设的位串进行扩展，使其包含另外两位以决定是否可以对该假设应用这两个新算子

假设空间搜索

- 遗传算法采用一种随机化的柱状搜索来寻找最大适应度的假设，与前面章节的搜索算法有很大不同
  - 梯度下降搜索从一个假设平滑移动到另一个非常相似的假设
  - 遗传算法的移动可能非常突然，使用和双亲根本不同的后代替换双亲假设
  - 遗传算法的搜索不太可能像梯度下降方法那样陷入局部最小值的问题
- 遗传算法应用中的一个难题：拥挤问题
  - 拥挤：群体中某个体适应度大大高于其他个体，因此它迅速繁殖，以至于此个体和与它相似的个体占据了群体的绝大部分
  - 拥挤降低了群体的多样性，从而减慢了进化的进程

假设空间搜索（2）

- 降低拥挤的策略
  - 使用锦标赛选择或排序选择，而不用适应度比例轮盘赌选择
  - 适应度共享，根据群体中与某个体相似的个体数量，减小该个体的适应度
  - 对可重组生成后代的个体种类进行限制，比如受到生物进化的启示
    - 通过只允许最相似的个体重组，可以在群体中促成相似的个体聚类，形成亚种
    - 按空间分布个体，只允许相邻的个体重组

群体进化和模式理论

- 模式理论（Holland1975）提供了一种用数学方法刻画遗传算法中群体进化的过程
- 模式是由若干0、1和*组成的任意串，*表示任意符号
- 模式理论根据每个模式的实例数量来刻画遗传算法中群体的进化
  - 令m(s,t)表示群体中的模式s在时间t的实例数量
  - 模式理论根据m(s,t)和模式、群体及其他属性来描述m(s,t+1)的期望值

群体进化和模式理论（2）

- 遗传算法中群体的进化依赖于几个步骤，即选择、重组和变异。
  - 符号表示：
    - f(h)表示位串个体h的适应度
    - n为群体中个体的总数量
    - $\bar{f}(t)$ 表示在时间t群体中所有个体的平均适应度
    - $\hat{u}(s,t)$ 表示在时间t群体中模式s的实例的平均适应度
    - $h \in s \cap p_t$ 表示个体h既是模式s的一个代表，又是时间t群体的一个成员

群体进化和模式理论（3）

- E(m(s,t+1))的推导，仅考虑选择的情况

$$\Pr(h) = \frac{f(h)}{\sum_{i=1}^{n} f(h_i)} \qquad \hat{u}(s,t) = \frac{\sum_{h \in s \cap p_t} f(h)}{m(s,t)}$$

$$\Pr(h \in s) = \sum_{h \in s \cap p_t} \frac{f(h)}{n\bar{f}(t)} = \frac{\hat{u}(s,t)m(s,t)}{n\bar{f}(t)}$$

$$E[m(s,t+1)] = n \Pr(h \in s) = \frac{\hat{u}(s,t)m(s,t)}{\bar{f}(t)}$$

- 式子9.3表明，在t+1代中，模式s的实例期望数量与这个模式的实例在时间t内的平均适应度成正比，与群体的所有成员的平均适应度成反比，适应度高的模式的影响力会随着时间增加

群体进化和模式理论（4）

- 同时考虑交叉和变异，仅考虑单点交叉和可能造成的负面影响

$$E[m(s,t+1)] \geq \frac{\hat{u}(s,t)}{\bar{f}(t)} m(s,t)\left(1 - p_c \frac{d(s)}{l-1}\right)(1-p_m)o(s)$$

最左一项描述了选择步的影响，中间一项描述了单点交叉算子的影响，最后一项描述了代表模式s的任意个体在应用了变异算子后还代表s的概率

- 模式理论不完备的是：无法考虑交叉和变异的正面影响
- 其他一些新的理论分析：基于马尔可夫链模型和统计力学模型

遗传编程

- 遗传编程是进化计算的一种形式，其中进化群体中的个体是计算机程序而不是位串
- 遗传编程中的程序一般被表示为程序的解析树，每个函数调用被表示为树的一个节点，函数的参数通过它的子节点给出
- 遗传编程算法维护多个个体组成的群体，在每一步迭代中，它使用选择、交叉、变异产生新一代个体
- 群体中某个体程序的适应度一般通过在训练数据上执行这个程序来决定
- 交叉操作：在一个双亲程序中随机选择一个子树，然后用另一个双亲的子树替代这个子树

遗传编程举例

- 学习堆砌图9-3所示的子块的算法（ Koza1992）
  - 开发一个通用的算法来把字块堆叠成单个栈，拼出单词，无论这些字块初始的结构如何
  - 可执行的动作是每次只允许移动一个字块，栈中最上面的字块可以被移到桌面上，或者桌面上的字块移到栈顶
- 原子函数包含3个端点参数
  - CS，当前栈，即栈顶字块的名字，栈空时为F
  - TB，最上面的正确字块，即该字块和它以下的字块均为正确顺序的字块
  - NN，下一个所需字块，即为了拼成单词 “universal”，栈内紧邻TB之上的所需字块的名字，当不再需要时为F

遗传编程举例（2）

- 原子函数
  - MS x：移动到栈，如果字块x在桌面上，把x移动到栈顶，并且返回T；否则什么也不做，返回F
  - MT x：移动到桌面，如果字块x是在栈中某个位置，把栈顶的字块移动到桌面，并且返回T；否则返回F
  - EQ x y：如果x等于y，返回T；否则F
  - NOT x：如果x=F，返回T；否则F
  - DU x y：反复执行表达式x直到表达式y返回T
- Koza提供了166个训练问题，表示不同的初始字块结构，任意给定程序的适应度就是它解决的训练问题的数量
- 群体被初始化为300个随机程序的集合，经过10代后，系统发现了下面的程序解决所有的166个问题

(EQ (DU (MT CS) (NOT CS)) (DU (MS NN) (NOT NN)))

遗传编程说明

- 遗传编程把遗传算法扩展到对完整的计算机程序的进化
- 尽管遗传算法要搜索巨大的假设空间，但已经证实相当数量的应用中遗传编程产生了很好的结果
- 遗传编程在更复杂的任务中的应用
  - 电子滤波电路的设计
  - 蛋白质分子片断的分析
- 在大多数情况下，表示方法的选择和适应度函数的选择对遗传编程的性能是至关重要的，一个活跃的研究领域是自动发现和合并子程序，改善最初的原子函数集合，从而允许系统动态地改变用以构建个体的原子

进化和学习模型

- 有关进化系统的一个有趣问题：单一个体生命期间的学习与整个物种较长时期内由进化促成的学习，它们的关系是什么？
- 拉马克进化
  - 拉马克在19世纪末提出，多代的进化直接受到个别生物体在它们生命期间的经验的影响
  - 目前的科学证据不支持拉马克模型
  - 但近来的计算机研究表明，拉马克过程有时可以提高计算机遗传算法的效率

进化和学习模型（2）

- 鲍德温效应
  - 通过个体学习改变进化进程的机制
  - 基于以下现象
    - 如果一个物种在一个变化的环境中进化，那么进化的压力会支持有学习能力的个体，这种学习能力可以使个体在其生命期间执行一种小的局部搜索，以最大化它的适应度，相反，不学习的个体的适应度完全取决于它的遗传结构，会处于相对劣势
    - 具备学习能力的个体可以通过学习克服遗传代码中的“丢失的”或“并非最优的”特性，从而支持更加多样化的基因池，然后促进适应性更加快速地进化
  - 提供了一种间接的机制，使个体的学习可以正面影响进化速度

进化和学习模型（3）

- 人们一直努力开发研究鲍德温效应的计算模型
  - Hinton & Nowlan1987对一个简单神经网络的群体进行试验
  - 在一个网络个体的“生命期”间，它的一些权值是固定的，而其他权是可被训练的
  - 在群体进化初期的各代中，具有很多可训练权值的个体占据较大的比例
  - 随着进化的进行，群体向着遗传给定权值和较少依赖个体学习权值的方向进化，正确的固定权值的数量趋于增长

并行遗传算法

- 遗传算法很自然地适合并行实现
- 粗粒度并行方法
  - 把群体细分成相对独立的个体群，称为类属，然后为每个类属分配一个不同的计算节点，在每个节点进行标准的GA搜索
  - 类属之间的通信和交叉发生的频率与类属内相比较低，类属之间的交换通过迁移来进行，也就是某些个体从一个类属复制或交换到其他类属
  - 这个过程模拟了以下的生物进化方式，即自然界中异体受精可能发生在分离的物种子群体之间
  - 这种方法的一个好处是减少了非并行GA经常碰到的拥挤问题
- 细粒度并行方法
  - 给每个个体分配一个处理器，然后相邻的个体间发生重组

小结

- 遗传算法用一种随机化的并行爬山搜索来发现使预先定义的适应度函数最优的假设
- 遗传算法基于对生物进化的模拟，维护一个由竞争假设组成的多样化群体，在每一次迭代中，选出群体中适应度最高的成员来产生后代，替代群体中适应度最差的成员
- 遗传算法阐明了如何把学习过程看成最优化过程的一个特例，学习任务就是根据预先定义的适应度函数发现最优假设
- 遗传编程是遗传算法的变体，在遗传编程中，被操作的假设是计算机程序而不是位串

补充读物

- 在计算机科学发展的早期，人们就开始探索基于进化的计算方法（ Box1957、Bledsoe1964）
- Rechenberg1965,1973、Schwefel1975,1977,1995开发的进化策略用来优化工程设计中的数字参数
- Folgel & Owens & Walsh1966开发了进化编程，作为进化有限状态机的一种方法
- Koza1992介绍了遗传编程，把遗传算法的搜索策略应用到由计算机程序组成的假设中
- K. Dejong和他的学生开发了使用GA学习规则集的方法，每个规则集是竞争假设组成的群体的一个成员
- Holland和他的学生设定每个规则是群体的一个成员，群体本身是一个规则集

补充读物（2）

- Mitchell1996和Goldberg1986给出了有关遗传算法的两本教材
- Koza1992关于遗传编程的专著是对遗传算法扩展到操作计算机程序的标准参考书
- 会议
  - 遗传算法国际会议（主要会议）
  - 自适应行为仿真会议
  - 人工神经网络和遗传算法国际会议
  - IEEE进化计算国际会议
- 杂志
  - 进化计算杂志
  - 机器学习

机器学习

第1章 引言

什么是机器学习

什么是机器学习

计算机程序如何随着经验积累自动提高性能

系统自我改进的过程

历史

成功应用

学习识别人类讲话

学习驾驶车辆

学习分类新的天文结构

学习对弈西洋双陆棋

相关学科

- 人工智能
- 计算复杂性理论
- 控制论
- 信息论
- 统计学

学习问题的标准描述

- 定义
  - 如果一个计算机针对某类任务T的用P衡量的性能根据经验E来自我完善，那么我们称这个计算机程序在从经验E中学习，针对某类任务T，它的性能用P来衡量。
- 西洋跳棋学习问题的解释
  - E，和自己下棋
  - T，参与比赛
  - P，比赛成绩（或赢棋能力，击败对手的百分比）
- 手写识别学习问题
- 机器人驾驶学习问题

学习问题的标准描述（2）

- 定义太宽泛
  - 甚至包括了以非常直接的方式通过经验自我提高的计算机程序
- 实际的机器学习问题往往比较复杂
  - 定义一类问题
  - 探索解决这类问题的方法
  - 理解学习问题的基本结构和过程

设计一个学习系统

- 基本设计方法和学习途径

  （以西洋跳棋为例）

  – 选择训练经验
  – 选择目标函数
  – 选择目标函数的表示
  – 选择函数逼近算法
  – 最终设计

设计一个学习系统

- 西洋跳棋学习问题
  - 任务T，下西洋跳棋
  - 性能标准P，击败对手的百分比
  - 训练经验E，和自己进行训练对弈
- 学习系统需要选择
  - 要学习的知识的确切类型
  - 对于这个目标知识的表示
  - 一种学习机制

选择训练经验

- 第一个关键属性，训练经验能否为系统的决策提供直接或间接的反馈
- 第二个重要属性，学习器在多大程度上控制样例序列
- 第三个重要属性，训练样例的分布能多好地表示实例分布，通过样例来衡量最终系统的性能

选择目标函数

- 目标函数ChooseMove
  - ChooseMove: B→M，接受合法棋局集合中的棋盘状态作为输入，并从合法走子集合中选择某个走子作为输出
- 问题转化
  - 我们把提高任务T的性能P的问题转化（或简化）为学习像ChooseMove这样某个特定的目标函数

选择目标函数（2）

- ChooseMove的评价
  - 学习问题很直观地转化成这个函数
  - 这个函数的学习很困难，因为提供给系统的是间接训练经验
- 另一个目标函数V
  - 一个评估函数，V: B→R，它为任何给定棋局赋予一个数值评分，给好的棋局赋予较高的评分
  - 优点，学习简单
- V的应用
  - 根据V能够轻松地找到当前棋局的最佳走法。

选择目标函数（3）

- V的设计，对于集合B中的任意棋局b，V(b)定义如下
  - 如果b是一最终的胜局，那么V(b)=100
  - 如果b是一最终的负局，那么V(b)=-100
  - 如果b是一最终的和局，那么V(b)=0
  - 如果b不是最终棋局，那么V(b)=V(b’)，其中b’是从b开始双方都采取最优对弈后可达到的终局

选择目标函数（4）

- 上面设计的缺陷
  - 递归定义
  - 运算效率低
  - 不可操作
- 简评
  - 学习任务简化成发现一个理想目标函数V的可操作描述。
  - 通常要完美地学习这样一个V的可操作的形式是非常困难的。
  - 一般地，我们仅希望学习算法得到近似的目标函数V’，因此学习目标函数的过程常称为函数逼近。

选择目标函数的表示

- 函数的表示
  - 一张大表，对于每个唯一的棋盘状态，表中有唯一的表项来确定它的状态值
  - 规则集合
  - 二项式函数
  - 人工神经网络

选择目标函数的表示（2）

- 重要的权衡过程
  - 一方面，我们总希望选区一个非常有表现力的描述，以最大可能地逼近理想的目标函数
  - 另一方面，越有表现力的描述需要越多的训练数据，使程序能从它表示的多种假设中选择

选择目标函数的表示（3）

- 一个简单的表示法，对于任何给定的棋盘状态，函数V可以通过以下棋盘参数的线性组合来计算。
  - x1，黑子的数量
  - x2，红子的数量
  - x3，黑王的数量
  - x4，红王的数量
  - x5，被红子威胁的黑子数量
  - x6，被黑子威胁的红子数量

选择目标函数的表示（4）

- 目标函数
  - $V(b)=w0+w1x1+w2x2+\ldots+w6x6$
  - 其中，w0…w6是权值，表示不同棋局特征的相对重要性
- 至此，问题转化为学习目标函数中的系数（即权值）

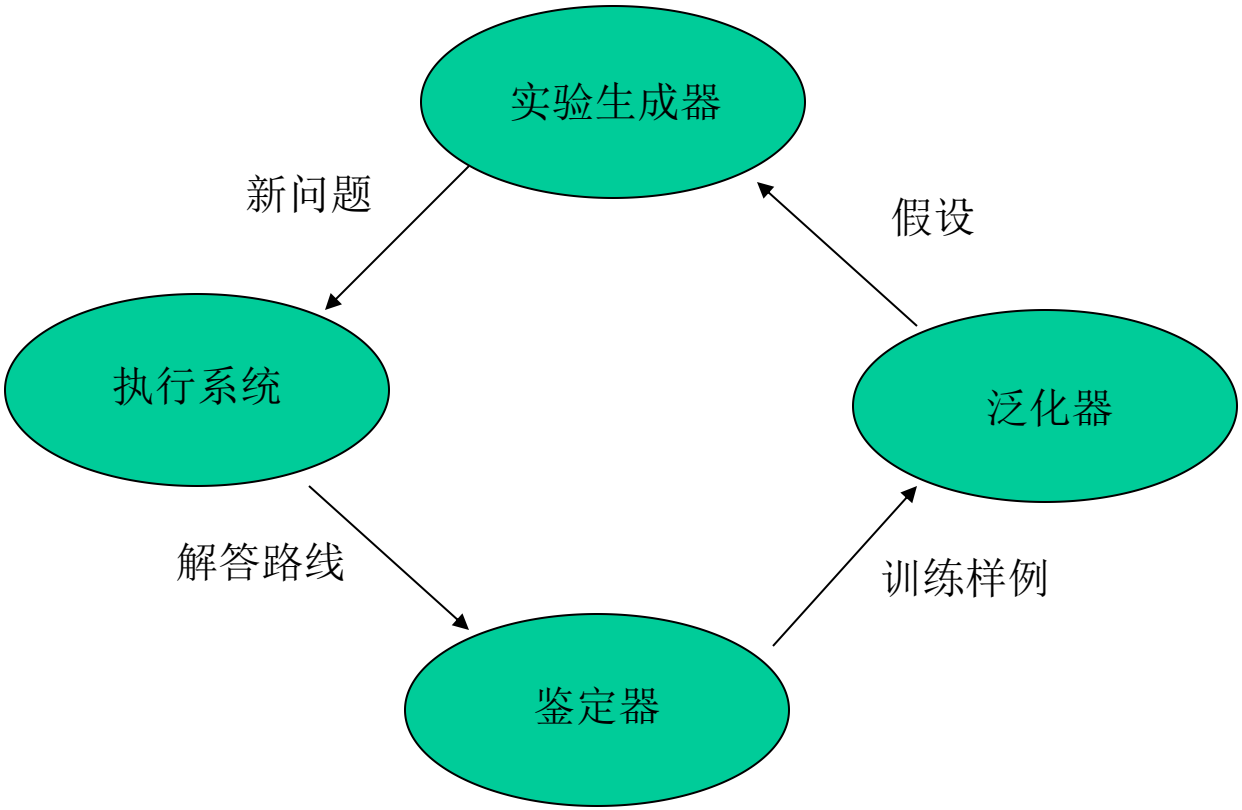
选择函数逼近算法

- 每个训练样例表示成二元对
  - <b，Vtrain(b)>
  - b是棋盘状态，Vtrain(b)是训练值
  - 比如，<<x1=0,x2=0,x3=1,x4=0,x5=0,x6=0>,100>
- 训练过程
  - 从学习器可得到的间接训练经验中导出上面的训练样例
  - 调整系数wi，最佳拟合这些训练样例

选择函数逼近算法（2）

- 估计训练值
  - 困难处
  - 一个简单的方法，Vtrain(b)=V’(Successor(b))
- 调整权值
  - 最佳拟合的定义，比如误差平方和最小
  - 寻找算法，比如最小均方法，LMS Least Mean Squares

最终设计



最终设计（2）

- 执行系统
  - 用学会的目标函数来解决给定的任务
- 鉴定器
  - 以对弈的路线或历史记录作为输入，输出目标函数的一系列训练样例。
- 泛化器
  - 以训练样例为输入，产生一个输出假设，作为它对目标函数的估计。
- 实验生成器
  - 以当前的假设作为输入，输出一个新的问题，供执行系统去探索。

西洋跳棋学习的更多讨论

- 图1-2
- 第13章理论上的保证
  - 这种学习技术是否确保发现一个非常接近的近似。
- 更复杂的目标函数
- 其他学习算法
  - 最近邻算法，存储训练样例，寻找保存的最接近的情形来匹配新的情况
  - 遗传算法，产生大量候选的西洋跳棋程序，让它们相互比赛，保留最成功的程序并进一步用模拟进化的方式来培育或变异它们
  - 基于解释的学习，分析每次成败的原因

机器学习的一个观点

- 一个有效的观点
  - 机器学习问题归结于搜索问题
- 本书给出了对一些基本表示定义的假设空间的搜索算法
- 通过搜索策略和搜索空间的内在结构来刻画学习方法

机器学习的问题

- 存在什么样的算法能从特定的训练数据学习一般的目标函数呢？如果提供了充足的训练数据，什么样的条件下，会使特定的算法收敛到期望的函数？哪个算法对哪些问题和表示的性能最好？
- 多少训练数据是充足的？怎样找到学习到假设的置信度与训练数据的数量及提供给学习器的假设空间特性之间的一般关系？
- 学习器拥有的先验知识是怎样引导从样例进行泛化的过程的？当先验知识仅仅是近似正确时，它们会有帮助吗？
- 关于选择有效的后验训练经验，什么样的策略最好？这个策略的选择会如何影响学习问题的复杂性。
- 怎样把学习任务简化为一个或多个函数逼近问题？换一种方式，系统该试图学习哪些函数？这个过程本身能自动化吗？
- 学习器怎样自动地改变表示法来提高表示和学习目标函数的能力？

全书内容简介

- 第2章，基于符号和逻辑表示的概念学习
- 第3章，决策树
- 第4章，人工神经网络
- 第5章，统计和估计理论的基础概念
- 第6章，贝叶斯理论
- 第7章，计算学习
- 第8章，基于实例的学习
- 第9章，遗传算法
- 第10章，规则学习
- 第11章，基于解释的学习
- 第12章，近似知识与现有数据的结合
- 第13章，增强学习

小结和补充读物

- 定义
- 涉及学科
- 完整过程
- 搜索的观点
- 相关杂志、会议、国际组织

机器学习

第13章 增强学习

概述

- 增强学习要解决的问题：一个能够感知环境的自治agent，怎样通过学习选择能达到其目标的最优动作
- 当agent在其环境中做出每个动作，施教者提供奖励或惩罚信息，agent从这个非直接的回报中学习，以便后续动作产生最大的累积回报
- 本章介绍一个称为Q学习的算法，它可从有延迟的回报中获取最优控制策略
- 增强学习与动态规划算法有关，后者常被用于解决最优化问题

简介

- 考虑一个可学习的机器人，它可以观察环境的状态并能做出一组动作改变这些状态，学习的任务是获得一个控制策略，以选择能达到目的的行为
- 本章关心的是：机器人怎样在环境中做实验并根据回报函数成功学习到控制策略
- 图13-1，学习控制策略以使累积回报最大化这个问题很普遍，它是一个通过学习来控制序列过程的问题，比如
  - 生产优化问题：选择一系列生产动作，使生产出的货物减去其成本达到最大化
  - 出租车调度：选择出租车运载乘客，其中回报函数为乘客等待的时间和车队的整体油耗

简介（2）

- 在第11章，已经接触到了通过学习来控制序列过程的问题，用基于解释的方法学习规则，以控制问题求解中的搜索
- 本章考虑的问题不同于第11章，因为考虑的问题中，行为可能有非确定性的输出，而且学习器缺少描述其行为输出的领域理论
- 学习控制策略类似前面讨论过的函数逼近问题，这里待学习的目标函数是控制策略$\pi: S \rightarrow A$，它在给定当前状态S集合中的s时，从集合A中输出一个合适的动作a

简介（3）

- 增强学习问题与普通函数逼近问题有几个重要的不同：
  - 延迟回报：施教者只在机器人执行其序列动作时提供一个序列立即回报值，因此面临一个时间信用分配的问题：确定最终回报的生成应归功于序列中哪一个动作
  - 探索：学习器面临一个权衡过程，是选择探索未知的状态和动作，还是选择利用它已经学习过、会产生高回报的状态和动作
  - 部分可观察状态：机器人的传感器只能感知环境的部分状态
  - 终生学习：使得有可能使用先前获得的经验或知识在学习新任务时减小样本复杂度

学习任务

- 本节我们把学习序列控制策略的问题更精确地形式化，有多种可选择的形式化方法，比如
  - 机器人的行为是确定性或非确定性的
  - 机器人可以预测或不能预测每一个行为所产生的状态
  - 机器人由外部专家通过示例最优动作序列来训练或必须通过执行自己选择的动作来训练
  - …

学习任务（2）

- 我们基于马尔科夫决策过程定义学习控制策略问题的一般形式
  - 设机器人可感知到其环境的不同状态集合S，可执行的动作集合A
  - 在每个离散时间步t，机器人感知到当前状态st，选择当前动作at，环境给出回报rt=r(st,at)，并产生后继状态st+1=δ(st,at)
  - 注意：回报函数和后继状态函数只依赖于当前状态和动作，这里先考虑它们为确定性的情形
- 定义：策略π从初始状态st获得的累积值为

$$V^{\pi}(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots$$
$$= \sum_{i=0}^{\infty} \gamma^i r_{t+i}$$

学习任务（2）

- 上面定义的量又称为折算累积回报，还有其他一些整体回报的定义：有限水平回报、平均回报
- 定义：学习控制策略的任务是，要求机器人学习到一个策略π，使得对于所有状态s，$V^{\pi}(s)$为最大，表示为

$$\pi^{*} = \arg\max_{\pi} V^{\pi}(s), (\forall s)$$

最优策略的值函数$V^{\pi^{*}}(s)$记作$V^{*}(s)$

- 图13-2，对上面定义的示例

Q学习

- 机器人在任意的环境中直接学习最优策略很难，因为没有形式为<s,a>的训练样例
- 训练数据是立即回报函数，容易学习一个定义在状态和动作上的数值评估函数，然后实现最优策略
- 很明显，可以将V*作为待学习的评估函数，由于状态s下的最优动作是使立即回报r(s,a)加上立即后继状态的V*值最大的动作a，即

$$\pi^*(s) = \arg\max_a [r(s,a) + \gamma V^*(\delta(s,a))]$$

因此，如果具有回报函数和状态转移函数的完美知识，那么就可以计算出任意状态下的最优动作

- 但在实际问题中，无法知道回报函数和状态转移函数的完美知识

Q函数

- 对于无法知道回报函数和状态转移函数完美知识的情形，我们使用评估函数Q
- 评估函数Q的定义：

$$Q(s,a) = r(s,a) + \gamma V^*(\delta(s,a))$$

- 式子13.3可以重写为：

$$\pi^*(s) = \arg\max_a Q(s,a)$$

- 因此只需对当前状态的Q值做出反应，就可以选择到全局最优化的动作序列

Q函数（2）

- 注意到 $V^*(s) = \max_{a'} Q(s,a')$
- 式子13.4可重写为

$$Q(s,a) = r(s,a) + \gamma \max_{a'} Q(\delta(s,a),a')$$

- 这个Q函数的递归定义提供了迭代逼近Q算法的基础
- 用 $\hat{Q}$ 表示实际Q的估计，算法中用一个大表存储所有状态-动作对的 $\hat{Q}$ 值
- 一开始所有表项填充为初始的随机值，然后利用下式更新表项，直到这些值收敛

$$\hat{Q}(S,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a') \qquad s' = \delta(s,a)$$

表13-1 在确定性回报和动作假定下的Q学习算法

Q学习算法

- 对每个s,a初始化表项 $\hat{Q}(s,a)=0$
- 观察当前状态s，一直重复做：
  - 选择一个动作a并执行它
  - 接收到立即回报r
  - 观察新状态s’
  - 对 $\hat{Q}(s,a)$ 按照下式更新表项 $\hat{Q}(s,a)\leftarrow r+\gamma\max_{a'}\hat{Q}(s',a')$
  - s←s’

Q算法举例

- 图13-3，单个状态转移的Q学习
- 吸收目标状态：没有移出的状态
- 情节：在每个情节中，机器人从某个随机选择的状态开始执行动作直到到达吸收目标状态
- 训练过程包含一系列情节
- $\hat{Q}$ 值的演化过程
  - 因为初始的 $\hat{Q}$ 值都为0，算法不会改变任何 $\hat{Q}$ 表项，直到它恰好到达目标状态并且收到非零回报，这导致通向目标状态的转换的 $\hat{Q}$ 值被精化
  - 在下一个情节中，如果经过这些与目标状态相邻的状态，其非0的 $\hat{Q}$ 值会导致与目的相差两步的状态中值的变化，依次类推，最终得到一个 $\hat{Q}$ 表

Q算法举例（2）

- 上面的Q学习算法有两个特点：
  - $\hat{Q}$值在训练中不会下降
  - $\hat{Q}$值保持在0和真实Q值区间内
- 因此上面的Q学习算法会收敛到正确的Q函数
- 定理13.1（确定性马尔科夫决策过程中的Q学习的收敛性）
  - 考虑一个Q学习，在一个有有界回报的确定性MDP中，Q学习使用式子13.7的训练规则，将表$\hat{Q}(s,a)$初始化为任意有限值，并且使用折算因子，令$\hat{Q}_n(s,a)$表示在第n次更新后的值，如果那么对所有的s和a，当$n\rightarrow\infty$时$\hat{Q}_n(s,a)$收敛到Q(s,a)

Q算法举例（3）

– 证明：

- 令s’=δ(s,a)，则

$$
\begin{aligned}
|\hat{Q}_{n+1}(s,a)-Q(s,a)| &= |(r+\gamma \max_{a'} \hat{Q}_n(s',a'))-(r+\gamma \max_{a'} Q(s',a')| \\
&= \gamma |\max_{a'} \hat{Q}_n(s',a')-\max_{a'} Q(s',a')| \\
&\le \gamma \max_{a'} |\hat{Q}_n(s',a')-Q(s',a')| \\
&\le \gamma \max_{s'',a'} |\hat{Q}_n(s'',a')-Q(s'',a')|
\end{aligned}
$$

- 令 $\Delta_n = \max_{s,a} |\hat{Q}_n(s,a)-Q(s,a)|$ ， 则

$$\Delta_{n+1} \le \gamma \Delta_n$$

$$\lim_{n\to\infty} \Delta_n = 0$$

实验策略

- 表13-1的算法的一个明显的策略是，对于状态s，选择使 $\hat{Q}(s,a)$ 最大化的动作
- 上面策略的风险是过度束缚到在早期训练中有高 $\hat{Q}$ 值的动作，而不能探索到其它可能有更高值的动作
- 在Q学习中，通常使用概率的途径来选择动作，有较高 $\hat{Q}$ 值的动作被赋予较高的概率，但所有动作的概率都非0，其中一种方法是

$$P(a_i \mid s) = \frac{k^{\hat{Q}(s,a_i)}}{\sum_j k^{\hat{Q}(s,a_j)}}$$

更新序列

- 定理13.1说明，Q学习不需要用最优动作进行训练，就可以收敛到最优策略
- 可以改变训练转换样例的序列，以改进训练效率而不危及最终的收敛性
  - 图13-1的例子，以逆序方式执行更新，那么在第一个情节后，agent会沿着通向目标的路径对每个转换更形Q估计，这个过程要求更多的内存来存储整个情节，但会在更少的循环次数内收敛
- 第二个策略是存储过去的状态-动作转换，以及相应收到的立即回报，然后周期性地在其上重新训练
  - 重放旧的转换相比于从环境中获得新转换的程度取决于两种操作的开销

非确定性回报和动作

- 考虑非确定性情况，即回报函数和转换函数有概率的输出，比如西洋双陆棋，输出的动作具有固定的概率性
- 把处理确定性情况的方法扩展到非确定性的情况，一种一般化的方法是引入期望值 $V^{\pi}(s_t) = E[\sum_{i=0}^{\infty} \gamma^i r_{t+i}]$
- 把非确定性情况下的Q(s,a)简单地重定义为在确定性情况下定义的量的期望值

$$
\begin{aligned}
Q(s,a) &= E[r(s,a) + \gamma V^*(\delta(s,a))] \\
&= E[r(s,a)] + \gamma E[V^*(\delta(s,a))] \\
&= E[r(s,a)] + \gamma \sum_{s'} P(s'|s,a) V^*(s') \\
&= E[r(s,a)] + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q(s',a')
\end{aligned}
$$

非确定性回报和动作（2）

- 前面对确定性情形推导的训练法则不能够在非确定性条件下收敛
- ???

时间差分学习

- Q学习算法是时间差分算法中的特例，时间差分算法的学习过程是减小机器人在不同的时间做出估计间的差异
- 基于单步、两步、多步前瞻计算的训练值

$$Q^{(1)}(s_t, a_t) = r_t + \gamma \max_a \hat{Q}(s_{t+1}, a)$$

$$Q^{(2)}(s_t, a_t) = r_t + \gamma r_{t+1} + \gamma^2 \max_a \hat{Q}(s_{t+2}, a)$$

$$Q^{(n)}(s_t, a_t) = r_t + \gamma r_{t+1} + \ldots + \gamma^{n-1} r_{t+n-1} + \gamma^n \max_a \hat{Q}(s_{t+n}, a)$$

时间差分学习（2）

- Sutton使用的合并从不同前瞻距离中获得的估计的TD(λ)法

$$Q^{\lambda}(s_t,a_t)=(1-\lambda)[Q^{(1)}(s_t,a_t)+\lambda Q^{(2)}(s_t,a_t)+\lambda^2 Q^{(3)}(s_t,a_t)+...]$$

- 一个等价的递归定义

$$Q^{\lambda}(s_t,a_t)=r_t+\gamma[(1-\lambda)\max_a \hat{Q}(s_t,a)+\lambda Q^{\lambda}(s_{t+1},a_{t+1})]$$

- 如果选择λ=0，则得到原来的训练估计$Q^{(1)}$，它只考虑$\hat{Q}$估计中的单步差异，当λ增大时，算法重点逐渐转移到更远的前瞻步
- TD(λ)方法的动机是，在某些条件下，如果考虑更远的前瞻，训练会更有效
  - 如果机器人遵循最优策略选择动作，λ=1时将提供对真实Q值的完美估计，而不能$\hat{Q}$多么不准确

从样例中泛化

- Q学习中，只有每个可能的状态-动作被无限频繁的访问，学习过程才会收敛，执行效率很低
- 比较实际的系统通常在Q学习中合并使用其他章讨论的函数逼近方法
- 把反向传播算法结合到Q学习中，方法是用神经网络替代查找表，把每个　更新作为训练样例，例如
  - 把状态s和动作a编码为网络输入，并使用式子13.7和13.10训练网络
- 一旦引入了泛化函数逼近器，增强学习将不能收敛

与动态规划的联系

- 像Q学习这样的增强学习方法，与用于解决马尔科夫决策过程的动态规划方法有紧密的关系
- Q学习的新颖之处在于它假定不具有环境的完美知识，因此不能在内部模拟的状态空间移动中学习，而必须在现实世界的移动中观察后果
- 我们主要考虑机器人为收敛到一个可接受的策略必须执行的真实世界动作的数量，而不是计算迭代次数，这是因为在外部世界中执行动作的时间和费用开销比计算开销更值得关注
- 在真实环境中移动学习并观察结果的系统通常称为在线系统，而主要通过模型模拟动作的学习的被称为离线系统

与动态规划的联系（2）

- Bellman等式形成了解决MDP的动态规划方法的基础，形式如下

$$(\forall s \in S) V^*(s) = E[r(s, \pi(s)) + \gamma V^*(\delta(s, \pi(s)))]$$

- Bellman证明了最优策略满足上式，且满足上式的任何策略为最优策略
- 动态规划早期的工作包括Bellman-Ford最短路径算法，它基于节点邻居的距离，通过不断更新每个图节点到终点的估计距离来学习图中的路径，图的各边以及目标节点已知的假定，等价于移动函数和回报函数已知的假定

小结

- 增强学习解决自治机器人学习控制策略的问题，目标是使机器人从任意起始状态收到的总回报最大化
- 本章介绍的增强学习算法适合一类被称为马尔科夫决策过程的问题，即应用任意动作到任意状态上的输出只取决于当前动作和状态
- Q学习是增强学习的一种形式，其中机器人学习的是一组状态和动作上的估计函数，它被定义为最大期望折算积累回报，算法可用在不具备动作怎样影响环境的先验知识情况下
- 在一定假定下，Q学习具备收敛性，实践中需要大量的训练循环
- Q学习是更广泛的时间差分算法中的一种，时间差分算法通过不断减小在不同时间内产生的估计间的差异来学习
- 增强学习与动态规划有密切联系，关键差异是，动态规划假定拥有状态转移函数和回报函数的知识，而Q学习假定缺乏这些知识

补充读物

- Samuel1959，西洋双陆棋学习程序
- Bellman1958，Ford & Fulkerson1962，最短路径算法
- Holland1986，学习分类系统的组桶式方法
- Barto et al.1983，时间信用分配方法
- Sutton1988，Dayan1992，TD(λ)方法
- Watkin1989，Q学习
- McCallum1995和Littman1996，增强学习的扩展
- ...