人工智能

人工智能概述

主讲: 赵国亮

内蒙古大学电子信息工程学院

May 28, 2020

目录 I

机器学习

学习是人类获取知识的重要途径和自然智能的重要标志, 机器学习则是机器获取知识的重要途径和人工智能的重要标志. 它是人工智能的一个分支, 探索如何让计算机通过经验学习提高性能——Stuart Russell (伯克利人工智能学家, 友善及安全 AI 的先驱之一).

机器学习

对于某类任务 T 和性能度量 P, 如果计算机程序在 T 上以 P 衡量的性能随着经验 E 而自我完善, 就称这个计算机程序从经验 E 学习—- (Tom Mitchell, “全球机器学习教父”).

普遍认为, 机器学习 (Machine Learning, 常简称为 ML）的处理系统和算法是主要通过找出数据里隐藏的模式进而做出预测的识别模式, 它是人工智能 (Artificial Intelligence, 常简称为 AI）的一个重要子领域.

机器学习常见误解

「机器学习是一个新的领域, 它已经代替了人工智能的地位」.

这种误解是最近机器学习热潮产生的副作用, 大量学生在之前没有接触过人工智能的情况下学习了机器学习课程. 机器学习一直是人工智能的核心话题: 阿兰・图灵在二十世纪五十年代的论文中已经认为学习是通向人工智能最可行的途径. 人工智能最突出的早期成果就是 Arthur Samuel 使用机器学习构建跳棋程序.

「机器不能学习, 它们只能做程序员告诉它的事情」.

这显然是错的, 程序员能够告诉机器如何学习. Samuel 是一个优秀的跳棋玩家, 但他的程序很快就通过学习超过了他.

近年来, 机器学习的很多应用都需要大量数据来进行训练.

代表性观点

① 西蒙 (Simon, 1983): 学习就是系统中的适应性变化, 这种变化使系统在重复同样工作或类似工作时, 能够做得更好.

② 明斯基 (Minsky, 1985): 学习是在人们头脑里 (心理内部) 有用的变化.

③ 迈克尔斯基 (Michalski, 1986): 学习是对经历描述的建立和修改.

机器学习的主要方法

人工智能的关键

学习能力.

监督学习——使用有标签数据进行学习. 典型应用场景: 分类和回归

非监督学习 (Unsupervised learning)

- 使用无标签数据进行学习
- 典型应用场景: 聚类.

半监督学习 (Semi-supervised learning)

- 数据一部分有标签, 无标签数据的数量 » 有标签数据数量
- 典型应用场景: 海量数据分类.

强化学习 (Reinforcement learning)

- 使用无标签但有反馈的数据进行学习
- 典型场景: 策略推理.

- 机器学习的基本定律: 模型的出错率正比于模型复杂程度与样本大小的比.
- 结论: 模型复杂, 需要大样本. 样本小, 简化模型.

- 人工智能 **(1950’s->1980’s)** 到机器学习 **(1980’s->2010’s)** 再到深度学习的脉络 **(2010’s->** 至今**)**

机器学习的重点

以“推理” 为重点- > 以“知识” 为重点.
以“特征” 为重点-> 以“学习” 为重点.

- 传统机器学习: 需要人工提取特征代替原始输入, 有效的特征对模型性能至关重要. 传统机器学习算法的难点在于“特征工程. 特征提取步骤如图 1.

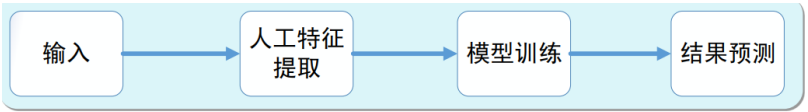


图 1: 特征提取步骤

- 一般性解释

学习是一个有特定目的知识获取和能力增长过程, 其内在行为是获得知识、积累经验、发现规律等, 其外部表现是改进性能、适应环境、实现自我完善等.

- 一般性解释

机器学习就是让机器 (计算机) 来模拟和实现人类的学习功能.

① 认知模拟

主要目的是要通过对人类学习机理的研究和模拟, 从根本上解决机器学习方面存在的种种问题.

② 理论性分析

主要目的是要从理论上探索各种可能的学习方法, 并建立起独立于具体应用领域的学习算法.

③ 面向任务的研究

主要目的是要根据特定任务的要求, 建立相应的学习系统.

• 神经元模型研究

20 世纪 50 年代中期到 60 年代初期, 也被称为机器学习的热烈时期, 最具有代表性的工作是罗森勃拉特 1957 年提出的感知器模型.

① 符号概念获取

20 世纪 60 年代中期到 70 年代初期. 其主要研究目标是模拟人类的概念学习过程. 这一阶段神经学习落入低谷, 称为机器学习的冷静时期.

② 知识强化学习

20 世纪 70 年代中期到 80 年代初期. 人们开始把机器学习与各种实际应用相结合, 尤其是专家系统在知识获取方面的需求, 也有人称这一阶段为机器学习的复兴时期.

③ 连接学习和混合型学习

20 世纪 80 年代中期至今. 把符号学习和连接学习结合起来的混合型学习系统研究已成为机器学习研究的一个新的热点.

• 学习系统

如果一个系统在与环境相互作用时, 能利用过去与环境作用时得到的信息, 并提高其性能, 那么这样的系统就是学习系统.

学习系统

包括环境、学习环节、知识库和执行环节 (图 2).

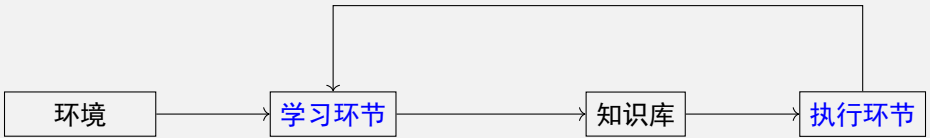


图 2: 学习系统

环境

环境是学习系统所感知到的外界信息集合, 也是学习系统的外界来源. 信息的水平 (一般化程度) 和质量 (正确性) 对学习系统影响较大.

学习环节

对环境提供的信息进行整理、分析归纳或类比, 形成知识, 并将其放入知识库.

知识库

存储经过加工后的信息 (即知识). 其表示形式是否合适非常重要.

执行环节

根据知识库去执行一系列任务, 并将执行结果或执行过程中获得的信息反馈给学习环节. 学习环节再利用反馈信息对知识进行评价, 进一步改善执行环节的行为.

按学习策略分类

可分为记忆学习、传授学习、演绎学习、归纳学习等.

按应用领域分类

① 专家系统学习、机器人学习、自然语言理解学习等.
② 按对人类学习的模拟方式.
③ 符号主义学习、连接主义学习等.

记忆学习

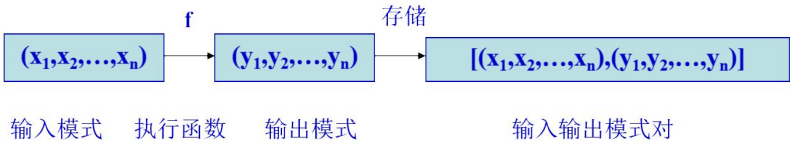
- 记忆学习 (Rote learning) 也叫死记硬背学习, 是一种最基本的学习过程, 它没有足够的能力独立完成智能学习, 但对学习系统来说都是十分重要的一个组成部分, 原因是任何学习系统都必须记住它们所获取的知识, 以便将来使用.

- 记忆学习的基本过程是: 执行元素每解决一个问题, 系统就记住这个问题和它的解, 当以后再遇到此类问题时, 系统就不必重新进行计算, 而可以直接找出原来的解去使用.

若把执行元素比作一个函数 f, 由环境得到的输入模式记为 $(x_1, x_2, \cdots, x_n)$, 由该输入模式经 F 计算后得到的输出模式记为 $(y_1, y_2, \cdots, y_m)$, 则机械学习系统就是要把这一输入输出模式对:

$$[(x_1, x_2, \cdots, x_n), (y_1, y_2, \cdots, y_m)], \tag{1}$$

保存在知识库中, 当以后再需要计算 $f(x_1, x_2, \cdots, x_n)$ 时, 就可以直接从存储器把 $(y_1, y_2, \cdots, y_m)$ 检索出来, 而不需要再重新进行计算.



归纳学习

归纳学习

- 归纳学习是指以归纳推理为基础的学习, 其任务是要从关于某个概念的一系列已知的正例和反例中归纳出一个一般的概念描述.

- 示例学习是归纳学习的一种特例. 它给学习者提供某一概念的一组正例和反例, 学习者归纳出一个总的概念描述, 并使这个描述适合于所有的正例, 排除所有的反例.

- 决策树学习是一种以示例为基础的归纳学习方法, 也是目前最流行的归纳学习方法之一. 在现有的各种决策树学习算法中, 影响较大的是 ID3 算法. 本节

示例学习

按例子的来源分类

① 来源于教师的示例学习.
② 来源于学习者本身的示例学习.

① 学习者明确知道自己的状态, 但完全不清楚所要获取的概念.
② 来源于学习者以外的外部环境的示例学习. 例子的产生是随机的.

按例子的类型分类

① 仅利用正例的示例学习.
这种学习方法会使推出的概念的外延扩大化.

② 利用正例和反例的示例学习

这是示例学习的一种典型方式, 它用正例用来产生概念, 用反例用来防止概念外延的扩大.

示例学习的典型方式

- 示例空间: 向系统提供的示教例子的集合. 研究问题: 例子质量, 搜索方法.
- 解释过程: 是从搜索到的示例中抽象出一般性的知识的归纳过程. 解释方法: 常量转换为变量, 去掉条件, 增加选择, 曲线拟合等.

- 规则空间: 是事务所具有的各种规律的集合. 研究问题: 对空间的要求, 搜索方法.
- 验证过程: 是要从示例空间中选择新的示例, 对刚刚归纳出的规则做进一步的验证和修改.

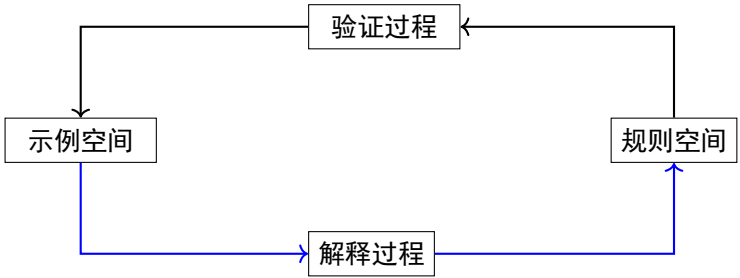


图 4: 示例学习

示例学习的解释方法

是指解释过程从具体示例形成一般性知识所采用的归纳推理方法.

最常用的解释方法有以下 4 种

① 把常量转换为变量: 把示例中的常量换成变量而得到一个一般性的规则.
② 去掉条件: 把示例中的某些无关的子条件舍去.
③ 增加选择: 在析取条件中增加一个新的析取项. 常用的增加析取项的方法有前件析取法和内部析取法两种.
④ 曲线拟合: 对数值问题的归纳可采用最小二乘法进行曲线拟合.

Example

假设例子空间中有以下两个关于扑克牌中 “同花” 概念的示例: 花色 (c_1, 梅花) $\wedge$ 花色 (c_2, 梅花) $\wedge$ 花色 (c_3, 梅花) $\wedge$ 花色 (c_4, 梅花) $\wedge$ 花色 (c_5, 梅花) $\rightarrow$ 同花 (c_1, c_2, c_3, c_4, c_5).

花色 (c_1, 红桃) $\wedge$ 花色 (c_2, 红桃) $\wedge$ 花色 (c_3, 红桃) $\wedge$ 花色 (c_4, 红桃) $\wedge$ 花色 (c_5, 红桃) $\rightarrow$ 同花 (c_1, c_2, c_3, c_4, c_5).

其中, 示例 1 表示 5 张梅花牌是同花, 示例 2 表示 5 张红桃牌是同花.

- 解释过程

Example

对这两个示例, 只要把 “梅花” 和 “红桃” 用变量 x 代换, 就可得到如下一般性的规则:

规则 1: 花色 (c_1, x) $\wedge$ 花色 (c_2, x) $\wedge$ 花色 (c_3, x) $\wedge$ 花色 (c_4, x)$\wedge$ 花色 (c_5, x)$\rightarrow$ 同花 (c_1, c_2, c_3, c_4, c_5).

- 去掉条件: 把示例中的某些无关的子条件舍去.

Example

花色 (c_1, 红桃)$\wedge$ 点数 (c_1, 2) $\wedge$ 花色 (c_2, 红桃)$\wedge$ 点数 (c_2, 3) $\wedge$ 花色 (c_3, 红桃)$\wedge$ 点数 (c_3, 4) $\wedge$ 花色 (c_4, 红桃)$\wedge$ 点数 (c_4, 5) $\wedge$ 花色 (c_5, 红桃)$\wedge$ 点数 (c_5, 6) $\rightarrow$ 同花 (c_1, c_2, c_3, c_4, c_5).

为了学习同花的概念, 除了需要把常量变为变量外, 还需要把与花色无关的 “点数” 子条件舍去. 这样也可得到上述规则 1:

♥ 规则 1: 花色 (c_1, x) $\wedge$ 花色 (c_2, x) $\wedge$ 花色 (c_3, x) $\wedge$ 花色 (c_4, x)$\wedge$ 花色 (c_5, x)$\rightarrow$ 同花 (c_1, c_2, c_3, c_4, c_5).

• 增加选择: 在析取条件中增加一个新的析取项. 包括前件析取法和内部析取法.
♠ 前件析取法: 是通过对示例的前件的析取来形成知识的.

Example

示例的前件有 4 个: ① 点数 (c_1, J)$\rightarrow$ 脸 (c_1).
② 点数 (c_1, Q)$\rightarrow$ 脸 (c_1).
③ 点数 (c_1, K)$\rightarrow$ 脸 (c_1).
将各示例的前件进行析取, 就可得到所要求的规则:
♥ 规则 2: 点数 $(c_1, J) \vee$ 点数 $(c_1, Q) \vee$ 点数 $(c_1, K) \rightarrow$ 脸 (c_1)

内部析取法

是在示例的表示中使用集合与集合的成员关系来形成知识的.

Example

有如下关于 “脸牌” 的示例: ① 点数 $c_1 \in J \rightarrow$ 脸 (c_1).
② 点数 $c_1 \in Q \rightarrow$ 脸 (c_1).
③ 点数 $c_1 \in K \rightarrow$ 脸 (c_1).
用内部析取法, 可得到如下规则:
♥ 规则 3: 点数 $(c_1) \in \{J, Q, K\} \rightarrow$ 脸 (c_1).

- 曲线拟合: 对数值问题的归纳可采用曲线拟合法.

假设示例空间中的每个示例 (x, y, z) 都是输入 x, y 与输出 z 之间关系的三元组.

Example

有下 3 个点: $(0, 2, 7), (6, -1, 10), (-1, -5, -16)$, 用最小二乘法进行曲线拟合, 可得 x, y, z 之间关系的规则如下:

♥ 规则 4: $z = 2x + 3y + 1$.

注

在上述前三种方法中,

♠ 方法 (1) 是把常量转换为变量;

♠ 方法 (2) 是去掉合取项 (约束条件);

♠ 方法 (3) 是增加析取项.

它们都是要扩大条件的适用范围.

从归纳速度上看, 方法 (1) 的归纳速度快, 但容易出错; 方法 (2) 归纳速度慢, 但不容易出错. 因此, 在使用方法 (1) 时应特别小心.

Example

对示例 4、示例 5 及示例 6, 若使用方法 (1) , 则会归纳出如下的错误规则:
规则 5: (错误) 点数 $(c_1, x) \rightarrow$ 脸 (c_1),
它说明, 归纳过程是很容易出错的.

决策树

决策树是一种由节点和边构成的用来描述分类过程的层次数据结构.

决策树的根接点表示分类的开始, 叶节点表示一个实例的结束, 中间节点表示相应实例中的某一属性, 而边则代表某一属性可能的属性值.

在决策树中, 从根节点到叶节点的每一条路径都代表一个具体的实例, 并且同一路径上的所有属性之间为合取关系, 不同路径 (即一个属性的不同属性值) 之间为析取关系. 决策树的分类过程就是从这棵树的根接点开始, 按照给定的事例的属性值去测试对应的树枝, 并依次下移, 直至到达某个叶节点为止.

决策树

图 5 是一个非常简单的用来描述对鸟类进行分类的决策树.

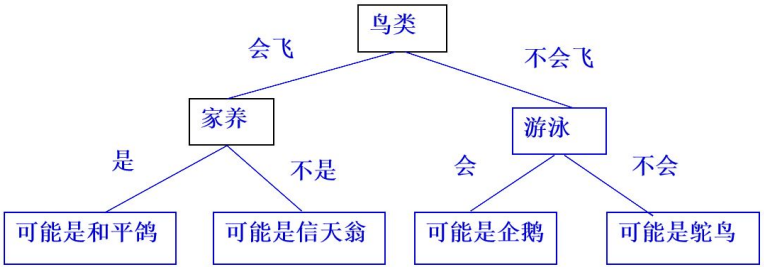


图 5: 对鸟类进行分类的决策树

在该图中: 根节点包含各种鸟类, 叶节点是所能识别的各种鸟的名称;

中间节点是鸟的一些属性, 边是鸟的某一属性的属性值;

从根节点到叶节点的每一条路径都描述了一种鸟, 它包括该种鸟的一些属性及相应的属性值.

决策树还可以表示成规则的形式. 上图 5 的决策树可表示为如下规则集:

表 1: 决策树规则集

IF	鸟类会飞	AND	是家养的	THEN	该鸟类是和平鸽
IF	鸟类会飞	AND	不是家养的	THEN	该鸟类是信天翁
IF	鸟类不会飞	AND	会游泳	THEN	该鸟类是企鹅
IF	鸟类不会飞	AND	不会游泳	THEN	该鸟类是鸵鸟

决策树学习过程实际上是一个构造决策树的过程. 当学习完成后, 就可以利用这棵决策树对未知事物进行分类.

ID3 算法

D3 算法是昆兰 (J. R. Quinlan) 于 1979 年提出的一种以信息熵 (entropy) 的下降速度作为属性选择标准的一种学习算法. 其输入是一个用来描述各种已知类别的例子集, 学习结果是一棵用于进行分类的决策树.

- ID3 算法的数学基础
  先引入信息熵和条件熵的数学概念

信息熵

信息熵是对信息源整体不确定性的度量. 假设 X 为信源, x_i 为 X 所发出的单个信息, $P(x_i)$ 为 X 发出 x_i 的概率, 则信息熵可定义为:

$$\begin{aligned} H(X) &= -P(x_1)\log P(x_1) - P(x_2)\log P(x_2) - \cdots - P(x_r)\log P(x_r) \\ &= -\sum_{i=1}^{k} P(x_i)\log P(x_i). \end{aligned} \tag{2}$$

其中, k 为信源 X 发出的所有可能的信息类型, 对数可以是以各种数为底的对数, 在 ID3 算法中, 我们取以 2 为底的对数.

♦信息熵反应的是信源每发出一个信息所提供的平均信息量.

条件熵

条件熵是收信者在收到信息后对信息源不确定性的度量. 若假设信源为 X, 收信者收到的信息为 Y, $P(x_i/y_j)$ 为当 Y 为 y_j 时 X 为 x_i 的条件概率, 则条件熵可定义为:

$$H(X|Y) = -\sum_{i}^{k}\sum_{j}^{k} P\left(x_i|y_j\right) \log P\left(x_i|y_j\right). \tag{3}$$

它表示收信者收到 Y 后对 X 不确定性的估计.

- ID3 算法及举例

ID3 算法的学习过程

- 首先以整个例子集作为决策树的根节点 S, 并计算 S 关于每个属性的期望熵(即条件熵);
- 然后选择能使 S 的期望熵为最小的一个属性对根节点进行分裂, 得到根节点的一层子节点;
- 接着再用同样的方法对这些子节点进行分裂, 直至所有叶节点的熵值都下降为 0 为止.

这时, 就可得到一棵与训练例子集对应的熵为 0 的决策树, 即 ID3 算法学习过程所得到的最终决策树. 该树中每一条从根节点到叶节点的路径, 都代表了一个分类过程, 即决策过程.

Example

用 ID3 算法完成下述学生选课
假设将决策 y 分为以下 3 类:

表 2: 选课决策

y_1:	必修 AI
y_2:	选修 AI
y_3:	不修 AI

Example

做出这些决策的依据有以下 3 个属性:

表 3: 选课决策

x_1:	学历层次	$x_1 = 1$	研究生,	$x_1 = 2$	本科
x_2:	专业类别	$x_2 = 1$	电信类,	$x_2 = 2$	机电类
x_3:	学习基础	$x_3 = 1$	修过 AI,	$x_3 = 2$	未修 AI.

表 4 给出了一个关于选课决策的训练例子集 S. 在该表中, 训练例子集 S 的大小为. ID3 算法是依据这些训练例子, 以 S 为根节点, 按照信息熵下降最大的原则来构造决策树的.

表 4: 关于选课决策的训练例子集

序号	属性值			决策方案
y_i	x_1	x_2	x_3	
1	1	1	1	y_3
2	1	1	2	y_1
3	1	2	1	y_3
4	1	2	2	y_2
5	2	1	1	y_3
6	2	1	2	y_2
7	2	2	1	y_3
8	2	2	2	y_3

- 解: 首先对根节点, 其信息熵为:

$$H(S) = -\sum_{i=1}^{3} P(y_i) \log_2 P(y_i). \tag{4}$$

其中, 为可选的决策方案数, 且有

$$P(y_1) = \frac{1}{8}, P(y_2) = \frac{2}{8}, P(y_3) = \frac{5}{8}. \tag{5}$$

即有:

$$H(S) = -\left(\frac{1}{8}\right) \log_2 \left(\frac{1}{8}\right) - \left(\frac{2}{8}\right) \log_2 \left(\frac{2}{8}\right) - \left(\frac{5}{8}\right) \log_2 \left(\frac{5}{8}\right) = 1.2988. \tag{6}$$

按照 ID3 算法

需要选择一个能使 S 的期望熵为最小的一个属性对根节点进行扩展, 因此我们需要先计算 S 关于每个属性的条件熵:

$$H(S|x_i) = \sum_t \frac{|S_t|}{|S|} H(S_i), \tag{7}$$

其中, t 为属性 x_i 的属性值, S_t 为 $x_i = t$ 时的例子集, $|S|$ 和 $|S_i|$ 分别是例子集 S 和 S_i 的大小.

下面先计算 S 关于属性 x_1 的条件熵:

$$H(S|x_i) = \sum_t \frac{|S_t|}{|S|} H(S_i). \tag{8}$$

在表 7-1 中, x_1 的属性值可以为 1 或 2. 当 $x_1 = 1$ 时, $t = 1$ 时, 有:

$$S_1 = \{1, 2, 3, 4\}. \tag{9}$$

当 $x_1 = 2$ 时, $t = 2$ 时, 有:

$$S_2 = \{5, 6, 7, 8\}, \tag{10}$$

其中, S_1 和 S_2 中的数字均为例子集 S 中的各个例子的序号, 且有 $|S| = 8, |S_1| = |S_2| = 4$.

由 S_1 可知:

$$P_{s_1}(y_1) = 1/4, P_{s_1}(y_2) = 1/4, P_{s_1}(y_3) = 2/4. \tag{11}$$

则有:

$$\begin{aligned} H(S_1) &= -P_{s1}(y_1)\log_2 P_{s1}(y_1) - P_{s_1}(y_2)\log_2 P_{s1}(y_2) - P_{s_1}(y_3)\log_2 P_{s_1}(y_3) \\ &= -(1/4)\log_2(1/4) - (1/4)\log_2(1/4) - (2/4)\log_2(2/4) = 1.5. \end{aligned} \tag{12}$$

再由 S_2 可知:

$$P_{s_2}(y_1) = \frac{0}{4}, P_{s_2}(y_2) = \frac{1}{4}, P_{s_2}(y_3) = \frac{3}{4}. \tag{13}$$

则有:

$$\begin{aligned} H(S_2) &= -P_{s_2}(y_2)\log_2 P_{s_2}(y_2) - P_{s_2}(y_3)\log_2 P_{s_2}(y_3) \\ &= -\frac{1}{4}\log_2\left(\frac{1}{4}\right) - \left(\frac{3}{4}\right)\log_2\left(\frac{3}{4}\right) = 0.8113. \end{aligned} \tag{14}$$

将 $H(S_1)$ 和 $H(S_2)$ 代入条件熵公式, 有:

$$\begin{aligned} H(S|x_1) &= (|S_1|/|S|)H(S_1) + (|S_2|/|S|)H(S_2) \\ &= \left(\frac{4}{8}\right) \times 1.5 + \left(\frac{4}{8}\right) \times 0.8113 = 1.1557. \end{aligned} \tag{15}$$

同理可得

$$H(S|x_2) = 1.1557, H(S|x_3) = 0.75. \tag{16}$$

可见, 应该选择属性 x_3 对根节点进行扩展.

用 x_3 对 S 扩展后所得到的得到部分决策树如图 6 所示.

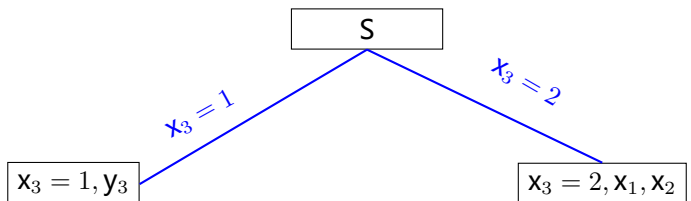


图 6: 部分决策树

在该树中, 节点 “$x_3 = 1, y_3$” 表示当 x_3 的属性值为 1 时, 得到决策方案 y_3. 由于 y_3 已是具体的决策方案, 故该节点的信息熵为 0, 已经为叶节点.
节点 “$x_3 = 2, x_1, x_2$” 的含义是 “当 x_3 的属性值为 2 时, 还需要考虑属性 x_1, x_2”, 它是一个中间节点, 还需要继续扩展.

至于节点 “$x_3 = 2, x_1, x_2$”, 其扩展方法与上面的过程类似. 通过计算可知, 该节点对属性 x_1 和 x_2, 其条件熵均为 1. 由于它对属性 x_1 和 x_2 的条件熵相同, 因此可以先选择 x_1, 也可以先选择 x_2, 本例是先选择 x_2.

依此进行下去, 可得到如图 7 所示的最终的决策树. 在该决策树中, 各节点的含义与图 6 类似.

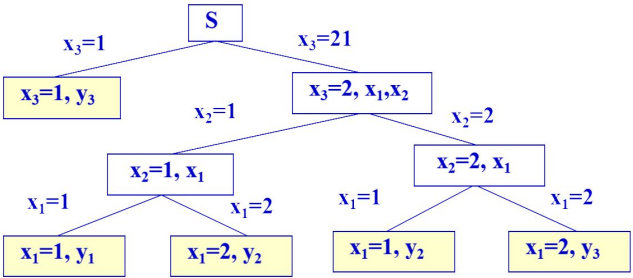

S
x_3=1
x_3=21
x_3=1, y_3
x_3=2, x_1,x_2
x_2=1
x_2=2
x_2=1, x_1
x_2=2, x_1
x_1=1
x_1=2
x_1=1
x_1=2
x_1=1, y_1
x_1=2, y_2
x_1=1, y_2
x_1=2, y_3

图 7: 最终决策树

解释学习涉及三个不同的空间: 例子空间, 概念空间和概念描述空间. 三个空间及它们之间的关系如图 8 所示.

概念描述空间是所有概念描述的集合, 其中的概念描述可分为两大类, 一类是可操作的, 另一类是不可操作的. 所谓可操作是指一个概念描述能有效的用于识别相应概念的例子. 否则是不可操作的.

解释学习的任务就是要把不可操作的概念描述转化为可操作的概念描述.

概念空间是学习过程能够描述的所有概念的集合. 例子空间是用于问题求解的例子集合

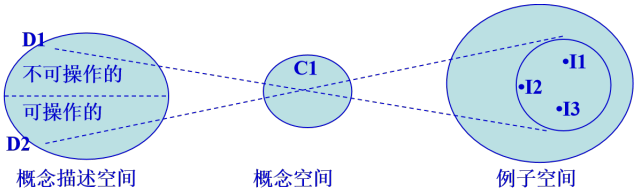


图 8: 解释学习的三个空间

解释学习的模型

模型组成说明:

① EXL 为学习系统
② KB 为领域知识库, 它是不同概念描述之间进行转换所使用的规则集合;
③ PS 为执行系统;
④ D1 是输入的概念描述, 一般为不可操作的;
⑤ D2 是学习结束时输出的概念描述, 它是可操作的.

执行过程: 先由 EXL 接受输入的概念描述 D1, 然后再根据 KB 中的知识对 D1 进行不同描述的转换, 并由 PS 对每个转换结果进行测试, 直到被 PS 所接受, 即为可操作的概念描述 D2 为止; 最后输出 D2.

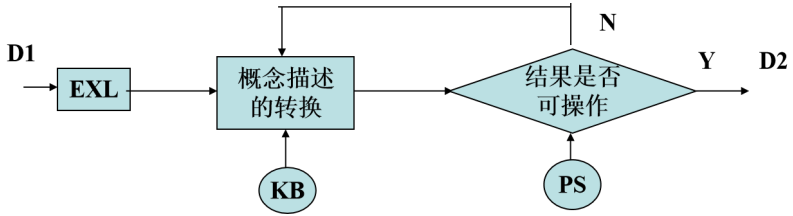


图 9: 解释学习模型

解释泛化学习方法-1 的提出者

主要讨论米切尔等人提出的解释泛化学习方法.

解释泛化学习方法-1 的基本思想:

先对某一情况建立一个解释结构, 然后在对此解释结构进行概括, 获取一般性控制知识.

解释泛化学习方法-1 的其一般性描述

- 目标概念 GC (Goal Concept);
  - ① 训练实例 TE (Training Example);
  - ② 领域理论 DT (Domain Theory);
  - ③ 操作性标准 OC (Operationality Criterion).
- 求出: 满足 OC 的关于 GC 的充分概念描述, 其中: 目标概念 GC 是要学习概念的描述.

解释泛化学习方法-1

- 训练实例 TE 为学习系统提供的一个实例;
- 领域理论 DT 是相关领域的事实和规则, 即为背景知识;
- 操作性标准 OC 用于指导学习系统对用来描述目标的概念进行舍取等的控制性知识.

解释泛化学习方法-2

本节主要讨论米切尔等人提出的解释泛化学习方法.

解释泛化学习方法-2

解释泛化学习的基本思想: 先对某一情况建立一个解释结构, 然后在对此解释结构进行概括, 获取一般性控制知识.

解释泛化学习方法-2

其一般性描述为: 已知:

- 目标概念 GC (Goal Concept);
  - ① 训练实例 TE (Training Example);
  - ② 领域理论 DT (Domain Theory);
  - ③ 操作性标准 OC (Operationality Criterion).
- 求出: 满足 OC 的关于 GC 的充分概念描述.

其中: 目标概念 GC 是要学习概念的描述.

解释泛化学习方法-2

- 训练实例 TE 是为学习系统提供的一个实例;
- 领域理论 DT 是相关领域的事实和规则, 即为背景知识;
- 操作性标准 OC 用于指导学习系统对用来描述目标的概念进行舍取等的控制性知识.

其任务是要证明提供给系统的训练实例为什么是目标概念的一个实例. 为此, 系统应从目标开始反向推理, 根据知识库中已有的事实和规则分解目标, 直到求解结束. 一旦得到解, 便完成了该问题的证明, 同时也获得了一个解释结构.

Example

假设要学习的目标是 “一个物体 x 可以安全地放置在另一个物体 y 的上面”. 即目标概念:

$$\text{Safe-to-Stack}(x, y).$$

训练实例 (是一些描述物体 obj_1 与 obj_2 的事实):

Example

On	(obj1 ,obj2)	物体 1 在物体 2 的上面
Isa	(obj1 , book)	物体 1 是书
Isa	(obj2 , table)	物体 2 是桌子
Volume	(obj1 , 1)	物体 1 的体积是 1
Density	(obj1 , 0.1)	物体 1 的密度是 0.1

Example

领域知识是把一个物体安全地放置在另一个物体上面的准则:

$$\neg \text{ Fragile } (y) \rightarrow \text{Safe-to-stack } (x, y).$$

1) 如果 y 不是易碎的, 则 x 可以安全地放到 y 的上面;

$$\text{Lighter } (x, y) \rightarrow \text{Safe-to-stack } (x, y).$$

2) 如果 x 比 y 轻, 则 x 可以安全地放到 y 的上面;

$$\text{Volume } (p, v) \wedge \text{Density } (p, d) \wedge \text{Product } (v, d, w) \rightarrow \text{Weight } (p, w).$$

3) 如果 p 的体积是 v、密度是 d、v 乘以 d 的积是 w, 则 p 的重量是 w;

$$\text{Is-a (p, table)} \rightarrow \text{Weight (p, 5)}.$$

4) 如果 p 是桌子, 则 p 的重量是 5;

$$\text{Weight } (p_1, w_1) \wedge \text{Weight } (p_2, w_2) \wedge \text{Smaller } (w_1, w_2) \rightarrow \text{Lighter } (p_1, p_2).$$

5) 如果 p_1 的重量是 w_1, p_2 的重量是 w_2, w_1 比 w_2 小, 则 p_1 比 p_2 轻.

其证明过程是一个由目标引导的逆向推理.

最终得到的解释树就是例 10 的解释结构 (如图 10).

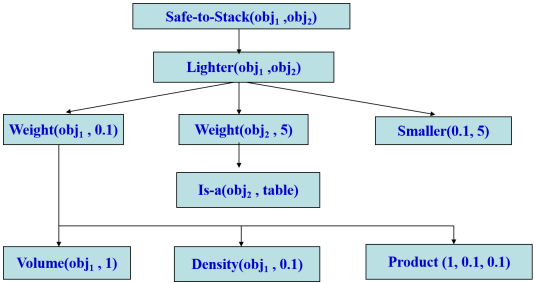


图 10: 逆向推理的解释树

这一步的主要任务是对上一步得到的解释结构进行概括化处理, 从而得到关于目标概念的一般性知识.

进行概括化处理的常用方法是把常量转换为变量, 即把某些具体数据换成变量, 并略去某些不重要的信息, 只保留求解所必须的那些关键信息即可.

对上图 10 的解释结构进行概括化处理以后所得到的概括化解释结构如下:

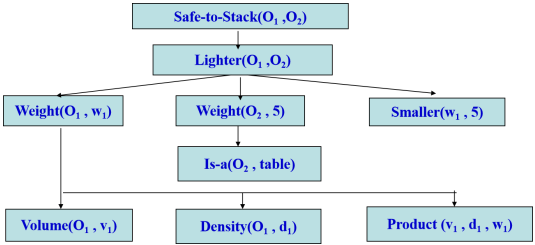

Safe-to-Stack(O1 ,O2)
Lighter(O1 ,O2)
Weight(O1 , w1)
Weight(O2 , 5)
Smaller(w1 , 5)
Is-a(O2 , table)
Volume(O1 , v1)
Density(O1 , d1)
Product (v1 , d1 , w1)

图 11: 概括化解释结构

将该解释结构中所有的叶节点的合取作为前件, 顶点的目标概念做为后件, 略去解释结构的中间部件, 就可得到概括化的一般性知识:

$$\begin{aligned} &\mathsf{Volume}(O_1, v_1) \wedge \mathsf{Density}(O_1, d_1) \wedge \mathsf{Product}(v_1, d_1, w_1) \\ &\qquad \wedge \mathsf{Is-a}(O_2, \mathsf{table}) \wedge \mathsf{Smaller}(w_1, 5) \\ &\rightarrow \mathsf{Safe\text{-}to\text{-}stack}(O_1, O_2). \end{aligned} \tag{17}$$

浅层神经学习

神经生理学研究表明, 人脑的神经元既是学习的基本单位, 同是也是记忆的基本单位. 目前, 关于人脑学习和记忆机制的研究有两大学派:

① 化学学派: 认为人脑经学习所获得的信息是记录在某些生物大分子之上的. 例如, 蛋白质、核糖核酸、神经递质, 就像遗传信息是记录在 DNA (脱氧核糖核酸) 上一样.

② 突触修正学派: 认为人脑学习所获得的信息是分布在神经元之间的突触连接上的.

浅层神经学习

按照突触修正学派的观点, 人脑的学习和记忆过程实际上是一个在训练中完成的突触连接权值的修正和稳定过程. 其中, 学习表现为突触连接权值的修正, 记忆则表现为突触连接权值的稳定.

突触修正假说已成为人工神经网络学习和记忆机制研究的心理学基础, 与此对应的权值修正学派也一直是人工神经网络研究的主流学派. 突触修正学派认为, 人工神经网络的学习过程就是一个不断调整网络连接权值的过程.

所谓学习规则可简单地理解为学习过程中联结权值的调整规则. 按照学习规则, 神经学习可分为 Hebb 学习、纠错学习、竞争学习及随机学习等.

Hebb 学习的基本思想

如果神经网络中某一神经元同另一直接与它连接的神经元同时处于兴奋状态, 那么这两个神经元之间的连接强度将得到加强, 反之应该减弱.

Hebb 学习对连接权值的调整可表示为:

$$w_{ij}(t+1) = w_{ij}(t) + \eta \left[x_i(t) x_j(t) \right], \tag{18}$$

其中, $w_{ij}(t+1)$ 表示对时刻 t 的权值修正一次后所得到的新的权值; η 是一正常量, 也称为学习因子, 它取决于每次权值的修正量; $x_i(t)$、$x_j(t)$ 分别表示 t 时刻第 i 个和第 j 个神经元的状态.

Hebb 学习规则在人工神经网络学习中的影响比较大, 但不符合生物机理. 例如习惯化.

纠错学习

是一种有导师的学习过程, 其基本思想: 利用神经网络的期望输出与实际输出之间的偏差作为连接权值调整的参考, 并最终减少这种偏差.

最基本的误差修正规则

连接权值的变化与神经元希望输出和实际输出之差成正比. 其联结权值的计算公式为:

$$w_{ij}(t+1) = w_{ij}(t) + \eta \left[d_j(t) - y_j(t) \right] x_i(t), \tag{19}$$

其中, $w_{ij}(t)$ 表示时刻 t 的权值;

$w_{ij}(t+1)$ 表示对时刻 t 的权值修正一次后所得到的新的权值;
η 是一正常量, 也称为学习因子;
$y_j(t)$ 为神经元 j 的实际输出;
$d_j(t)$ 为神经元 j 的希望输出;
$d_j(t) - y_j(t)$ 表示神经元 j 的输出误差;
$x_i(t)$ 为第 i 个神经元的输入.

竞争学习的基本思想

网络中某一组神经元相互竞争对外界刺激模式响应的权力, 在竞争中获胜的神经元, 其连接权会向着对这一刺激模式竞争更为有利的方向发展.

随机学习的基本思想

结合随机过程、概率和能量 (函数) 等概念来调整网络的变量, 从而使网络的目标函数达到最大 (或最小). 他不仅可以接受能量函数减少 (性能得到改善) 的变化, 而且还可以以某种概率分布接受使能量函数增大 (性能变差) 的变化.

单层感知器学习

是一种基于纠错学习规则, 采用迭代的思想对连接权值和阈值进行不断调整, 直到满足结束条件为止的学习算法.

感知器学习

X(k) 和 W(k) 分别表示学习算法在第 k 次迭代时输入向量和权值向量, 为方便, 把阈值 θ 作为权值向量 W(k) 中的第一个分量, 对应地把 “-1” 固定地作为输入向量 X(k) 中的第一个分量. 即 W(k) 和 X(k) 可分别表示如下:

$$\mathrm{X}(\mathrm{k}) = [-1, \mathrm{x}_1(\mathrm{k}), \mathrm{x}_2(\mathrm{k}), \cdots, \mathrm{x}_\mathrm{n}(\mathrm{k})], \tag{20}$$

$$\mathrm{W}(\mathrm{k}) = [\theta(\mathrm{k}), \mathrm{w}_1(\mathrm{k}), \mathrm{w}_2(\mathrm{k}), \cdots, \mathrm{w}_\mathrm{n}(\mathrm{k})]; \tag{21}$$

单层感知器学习是一种有导师学习, 它需要给出输入样本的期望输出.
假设一个样本空间可以被划分为 A, B 两类. 其功能函数的定义为: 对属于 A 类输入样本, 其功能函数的输出为 +1, 否则其输出为-1. 对应地也可将期望输出定义为: 当输入样本属于 A 类时, 其期望输出为 +1, 否则为-1.

单层感知器学习算法描述

(1) 设 $t = 0$, 初始化连接权和阈值. 即给 $w_i(0)$, $(i = 1, 2, \cdots, n)$ 及 $\theta(0)$ 分别赋予一个较小的非零随机数, 作为初值, 其中, $w_i(0)$ 是第 0 次迭代时输入向量中第 i 个输入的连接权值; $\theta(0)$ 是第 0 次迭代时输出节点的阈值;
(2) 提供新的样本输入 $x_i(t)$, $(i = 1, 2, \cdots, n)$ 和期望输出 $d(t)$;

(3) 计算网络的实际输出:

$$y(t) = f\left(\sum_{i=1}^{n} w_i(t)x_i(t) - \theta(t)\right), i = 1, 2, \ldots, n. \tag{22}$$

(4) 若 $y(t) = 1$, 不需要调整连接权值, 转 (6). 否则, 转 (5) 调整连接权值

$$w_i(t+1) = w_i(t) + \eta[d(t) - y(t)]x_i(t), i = 1, 2, \ldots, n. \tag{23}$$

其中, η 是一个增益因子, 用于控制修改速度, 其值如果太大, 会影响 $w_i(t)$ 的收敛性; 如果太小, 又会使 $w_i(t)$ 的收敛速度太慢;

(5) 判断是否满足结束条件：

① 若满足, 算法结束;

② 否则, 将 t 值加 1, 转 (2) 重新执行. 这里的结束条件一般是指 $w_i(t)$ 对一切样本均稳定不变.

③ 如果输入的两类样本是线性可分的, 则该算法就一定会收敛. 否则, 该算法将不收敛.

Example

用单层感知器实现逻辑 “与” 运算.

- 解: 根据 “与” 运算的逻辑关系, 可将问题转换为:

输入向量: $\mathsf{X}_1 = [0, 0, 1, 1], \mathsf{X}_2 = [0, 1, 0, 1]$; 输出向量: $\mathsf{Y} = [0, 0, 0, 1]$.
为减少算法的迭代次数, 设初始连接权值和阈值取值如下:

$$\mathsf{w}_1(0) = 0.5, \mathsf{w}_2(0) = 0.7, \theta(0) = 0.6, \tag{24}$$

并取增益因子 $\eta = 0.4$.

向量格式的输入

$$\mathsf{X} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}, \mathsf{Y} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

算法的学习过程

设两个输入为 $\mathsf{x}_1(0) = 0$ 和 $\mathsf{x}_2(0) = 0$, 其期望输出为 $\mathsf{d}(0) = 0$, 实际输出为:

$$\begin{aligned} \mathsf{y}(0) &= \mathsf{f}(\mathsf{w}_1(0)\mathsf{x}_1(0) + \mathsf{w}_2(0)\mathsf{x}_2(0) - \theta(0)) \\ &= \mathsf{f}(0.5 \times 0 + 0.7 \times 0 - 0.6) = \mathsf{f}(-0.6) = 0. \end{aligned}$$

实际输出与期望输出相同, 不需要调节权值.

算法的学习过程

再取下一组输入: $x_1(0) = 0$ 和 $x_2(0) = 1$, 期望输出 $d(0) = 0$, 实际输出:

$$\begin{aligned} y(0) &= f(w_1(0)x_1(0) + w_2(0)x_2(0) - \theta(0)) \\ &= f(0.5 \times 0 + 0.7 \times 1 - 0.6) = f(0.1) = 1. \end{aligned}$$

算法的学习过程

实际输出与期望输出不同, 需要调节权值, 其调整如下:

$$\theta(1) = \theta(0) + \eta(\mathsf{d}(0) - \mathsf{y}(0)) \times (-1) = 0.6 + 0.4 \times (0-1) \times (-1) = 1$$

$$\mathsf{w}_1(1) = \mathsf{w}_1(0) + \eta(\mathsf{d}(0) - \mathsf{y}(0))\mathsf{x}_1(0) = 0.5 + 0.4 \times (0-1) \times 0 = 0.5$$

$$\mathsf{w}_2(1) = \mathsf{w}_2(0) + \eta(\mathsf{d}(0) - \mathsf{y}(0))\mathsf{x}_2(0) = 0.7 + 0.4 \times (0-1) \times 1 = 0.3.$$

算法的学习过程

取下一组输入: $x_1(1) = 1$ 和 $x_2(1) = 0$, 其期望输出为 $d(1) = 0$, 实际输出为:

$$\begin{aligned} y(1) &= f(w_1(1)x_1(1) + w_2(1)x_2(1) - \theta(1)) \\ &= f(0.5 \times 1 + 0.3 \times 0 - 1) = f(-0.51) = 0. \end{aligned}$$

实际输出与期望输出相同, 不需要调节权值.

算法的学习过程

再取下一组输入: $x_1(1) = 1$ 和 $x_2(1) = 1$, 其期望输出为 $d(1) = 1$, 实际输出为:

$$\begin{aligned} y(1) &= f(w_1(1)x_1(1) + w_2(1)x_2(1) - \theta(1)) \\ &= f(0.5 \times 1 + 0.3 \times 1 - 1) \\ &= f(-0.2) = 0. \end{aligned}$$

算法的学习过程

实际输出与期望输出不同, 需要调节权值, 其调整如下:

$$\theta(2) = \theta(1) + \eta(\mathsf{d}(1) - \mathsf{y}(1)) \times (-1) = 1 + 0.4 \times (1 - 0) \times (-1) = 0.6$$
$$\mathsf{w}_1(2) = \mathsf{w}_1(1) + \eta(\mathsf{d}(1) - \mathsf{y}(1))\mathsf{x}_1(1) = 0.5 + 0.4 \times (1 - 0) \times 1 = 0.9$$
$$\mathsf{w}_2(2) = \mathsf{w}_2(1) + \eta(\mathsf{d}(1) - \mathsf{y}(1))\mathsf{x}_2(1) = 0.3 + 0.4 \times (1 - 0) \times 1 = 0.7.$$

算法的学习过程

取下一组输入: $x_1(2) = 0$ 和 $x_2(2) = 0$, 其期望输出为 $d(2) = 0$, 实际输出为:

$$y(2) = f(0.9 \times 0 + 0.7 \times 0 - 0.6) = f(-0.6) = 0.$$

实际输出与期望输出相同, 不需要调节权值.

算法的学习过程

再取下一组输入: $x_1(2) = 0$ 和 $x_2(2) = 1$, 期望输出为 $d(2) = 0$, 实际输出为:

$$y(2) = f(0.9 \times 0 + 0.7 \times 1 - 0.6) = f(0.1) = 1.$$

算法的学习过程

实际输出与期望输出不同, 需要调节权值, 其调整如下:

$$\begin{aligned}\theta(3) &= \theta(2) + \eta(\mathsf{d}(2) - \mathsf{y}(2)) \times (-1) = 0.6 + 0.4 \times (0-1) \times (-1) = 1,\\ \mathsf{w}_1(3) &= \mathsf{w}_1(2) + \eta(\mathsf{d}(2) - \mathsf{y}(2))\mathsf{x}_1(2) = 0.9 + 0.4 \times (0-1) \times 0 = 0.9,\\ \mathsf{w}_2(3) &= \mathsf{w}_2(2) + \eta(\mathsf{d}(2) - \mathsf{y}(2))\mathsf{x}_2(2) = 0.7 + 0.4 \times (0-1) \times 1 = 0.3.\end{aligned}$$

实际上, 由上一章关于与运算的阈值条件可知, 此时的阈值和连接权值以满足结束条件, 算法可以结束.

检验

① 对输入: “0 0” 有 $y = f(0.9 \times 0 + 0.3 \times 0 - 1) = f(-1) = 0$;
② 对输入: “0 1” 有 $y = f(0.9 \times 0 + 0.3 \times 0.1 - 1) = f(-0.7) = 0$;
③ 对输入: “1 0” 有 $y = f(0.9 \times 1 + 0.3 \times 0 - 1) = f(-0.1) = 0$;
④ 对输入: “1 1” 有 $y = f(0.9 \times 1 + 0.3 \times 1 - 1) = f(0.2) = 0$.

BP 网络学习过程是一个对给定训练模式, 利用传播公式, 沿着减小误差的方向不断调整网络连接权值和阈值的过程. 需要用到以下几个符号:

① O_i: 节点 i 的输出;
② I_j: 接点 j 的输入;
③ w_{ij}: 从节点 i 到节点 j 的连接权值;
④ θ_j: 节点 j 的阈值;
⑤ y_k: 输出层上节点 k 的实际输出;
⑥ d_k: 输出层上节点 k 的期望输出.

显然, 对隐含节点 j 有:

$$I_j = \sum_i w_{ij} O_i,\ O_j = f\left(I_j - \theta_j\right).$$

在 BP 算法学习过程中, 可以采用如下公式计算各输出节点的误差:

$$e = \frac{1}{2}\sum_k \left(d_k - y_k\right)^2.$$

连接权值的修改由下式计算:

$$w_{jk}(t+1) = w_{jk}(t) + \Delta w_{jk},$$

其中, $w_{jk}(t)$ 和 $w_{jk}(t+1)$ 分别是时刻 t 和 $t+1$ 时, 从节点 j 到节点 k 的连接权值; Δw_{jk} 是连接权值的变化量.

为了使连接权值能沿着 e 的梯度变化方向逐渐改善, 网络逐渐收敛, BP 算法按如下公式计算 Δw_{jk}:

$$\Delta w_{jk} = -\eta \frac{\partial e}{\partial w_{jk}},$$

其中, η 为增益因子, $\frac{\partial e}{\partial w_{jk}}$ 由下式计算:

$$\frac{\partial e}{\partial w_{jk}} = \frac{\partial e}{\partial I_k} \frac{\partial I_k}{\partial w_{jk}}.$$

由于

$$I_k = \sum_j w_{jk} O_j.$$

故有

$$\frac{\partial I_k}{\partial w_{jk}} = \frac{\partial}{\partial w_{jk}} \sum_j w_{jk} O_j = O_j.$$

令局部梯度

$$\delta_k = \frac{\partial e}{\partial I_k}.$$

故有

$$\Delta w_{jk} = -\eta \frac{\partial e}{\partial w_{jk}} = -\eta \delta_k O_j.$$

计算时, 需要区分节点 k 是输出层上的节点, 还是隐含层上的节点. 如果节点 k 是输出层上的节点, 则有 $O_k = y_k$, 因此

$$\delta_k = \frac{\partial e}{\partial I_k} = \frac{\partial e}{\partial y_k}\frac{\partial y_k}{\partial I_k}.$$

由于

$$\begin{aligned}\frac{\partial e}{\partial y_k} &= -\left(d_k - y_k\right),\\ \frac{\partial y_k}{\partial I_k} &= f'\left(I_k\right).\end{aligned}$$

所以

$$\delta_k = -(d_k - y_k) f'(l_k)$$
$$\Delta w_{jk} = \eta (d_k - y_k) f'(l_k) O_j.$$

如果节点 k 不是输出层上的节点, 则它是隐含层上的节点的, 此时:

$$\delta_k = \frac{\partial e}{\partial l_k} = \frac{\partial e}{\partial O_k}\frac{\partial O_k}{\partial l_k} = \frac{\partial e}{\partial O_k} f'(l_k),$$

其中, $\frac{\partial e}{\partial O_k}$ 是一个隐函数求导问题, 略去推导过程, 其结果为:

$$\frac{\partial e}{\partial O_k} = \sum_m \delta_m w_{km}.$$

所以

$$\delta_k = f'(I_k) \sum_m \delta_m w_{km}.$$

说明: 低层节点的 δ 值是通过上一层节点的 δ 值来计算的. 这样, 我们就可以先计算出输出层上的 δ 值, 然后把它返回到较低层上, 并计算出各较低层上节点的 δ 值.

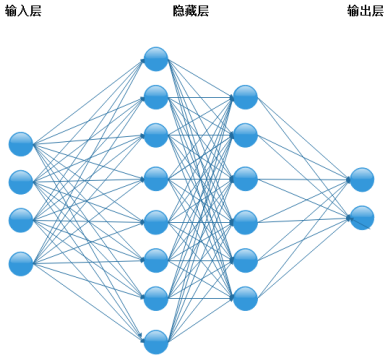


图 12: 多层感知机

算法实现过程

(1) 初始化网络及学习参数, 将各节点的连接权值、阈值赋予 $[-1, 1]$ 区间的一个随机数;

(2) 提供训练模式, 即从训练模式集合中选出一个训练模式送入网络;

(3) 正向传播过程, 即对给定输入模式, 计算输出模式, 并将其与期望模式比较, 若有误差则执行 (4), 否则返回 (2), 提供下一个训练模式.

(4) 反向传播过程, 即从输出层反向计算到第一隐含层, 按以下方式逐层修正各单元的连接权值:

① 计算同一层单元的误差;

② 按下式修正连接权值和阈值:

♠ 对连接权值, 修正公式为:

$$w_{jk}(t+1) = w_{jk}(t) + \eta \delta_k O_j. \tag{25}$$

对阈值, 可按照连接权值的学习方式进行, 只是要把阈值设想为神经元的连接权值, 并假定其输入信号总为单位值 1 即可.

反复执行上述修正过程, 直到满足期望的输出模式为止.

(5) 返回第 (2) 步, 对训练模式集中的每一个训练模式重复第 (2) 到第 (3) 步, 直到训练模式集中的每一个训练模式都满足期望输出为止.

Hopfield 网络学习的过程实际上是一个从网络初始状态向其稳定状态过渡的过程. 而网络的稳定性又是通过能量函数来描述的. 这里主要针对离散 Hopfield 网络讨论其能量函数和学习算法.

离散 Hopfield 网络的能量函数可定义为:

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} w_{ij}v_i v_j + \sum_{i=1}^{n}\theta_i v_i \tag{26}$$

式中, n 是网络中的神经元个数, w_{ij} 是第 i 个神经元和第 j 个神经元之间的连接权值, 且有 $w_{ij} = w_{ji}$; v_i 和 v_j 分别是第 i 个神经元和第 j 个神经元的输出; θ_i 是第 i 个神经元的阈值.

以证明, 当一神经元 k 的状态由 “0” 变为 “1” 时, 网络能量的 □ 变化为:

$$\Delta E = -\left(\sum_{\substack{j=1 \\ j\neq k}}^{n} w_{kj} v_j - \theta_k\right). \tag{27}$$

此时, 由于神经元 k 的状态由 “0” 变为 “1”, 因此有

$$\sum_{\substack{j=1 \\ j\neq k}}^{n} w_{kj}v_j - \theta_k > 0; \tag{28}$$

即 $\Delta E < 0$.

同理可证

若神经元 k 的状态由 “1” 变为 “0” 时, 网络能量函数的变化为:

$$\Delta E = \sum_{j=1}^{n} w_{kj}v_j - \theta_k. \tag{29}$$

此时, 由于神经元 k 的状态由 “1” 变为 “0”, 因此有

$$\sum_{\substack{j=1 \\ j\neq k}}^{n} w_{kj}v_j - \theta_k < 0; \tag{30}$$

即 $\Delta E < 0$.

可见, 无论神经元的状态由 “0” 变为 “1”, 还是由 “1” 变为 “0”, 都总有 $\delta E < 0$. 它说明离散 Hopfield 网络在运行中, 其能量函数总是在不断降低的, 最终将趋于稳定状态.

Example

如图所示的三个节点的 Hopfield 网络, 若给定的初始状态为:

$$\mathsf{v}_0 = \{1, 0, 1\}.$$

各节点之间的联结权值为:

$$\mathsf{w}_{12} = \mathsf{w}_{21} = 1, \mathsf{w}_{13} = \mathsf{w}_{31} = -2, \mathsf{w}_{23} = \mathsf{w}_{32} = 3.$$

Example

各节点的阈值为

$$\theta_1 = -1, \theta_2 = 2, \theta_3 = 1.$$

请计算在此状态下的网络能量.

- 解:

$$
\begin{aligned}
\mathsf{E} &= -\frac{1}{2}\left(\mathsf{w}_{12}\mathsf{v}_1\mathsf{v}_2 + \mathsf{w}_{13}\mathsf{v}_1\mathsf{v}_3 + \mathsf{w}_{21}\mathsf{v}_2\mathsf{v}_1 + \mathsf{w}_{23}\mathsf{v}_2\mathsf{v}_3 + \mathsf{w}_{31}\mathsf{v}_3\mathsf{v}_1 + \mathsf{w}_{32}\mathsf{v}_3\mathsf{v}_2\right) \\
&\quad + \theta_1\mathsf{v}_1 + \theta_2\mathsf{v}_2 + \theta_3\mathsf{v}_3 \\
&= -\left(\mathsf{w}_{12}\mathsf{v}_1\mathsf{v}_2 + \mathsf{w}_{13}\mathsf{v}_1\mathsf{v}_3 + \mathsf{w}_{23}\mathsf{v}_2\mathsf{v}_3\right) + \theta_1\mathsf{v}_1 + \theta_2\mathsf{v}_2 + \theta_3\mathsf{v}_3 \\
&= -(1 \times 1 \times 0 + (-2) \times 1 \times 1 + 3 \times 0 \times 1) + (-1) \times 1 + 2 \times 0 + 1 \times 1 \\
&= 2.
\end{aligned}
$$

(1) 设置互连权值

$$w_{ij} = \begin{cases} \sum_{s=1}^{m} x_i^s x_j^s, & i \neq j \\ 0, & i = j, i \geq 1, j \leq n \end{cases} \tag{31}$$

其中, x_{is} 为 S 型样例 (即记忆模式) 的第 i 个分量, 它可以为 1 或 0 (或-1), 样例类别数为 m, 节点数为 n.

(2) 对未知类别的样例初始化

$$y_i(i) = x_i, 1 \leq i \leq n, \tag{32}$$

其中, $y_i(t)$ 为节点 i 时刻 t 的输出, $y_i(0)$ 是节点的初值; x_i 为输入样本的第 i 个分量.

(3) 迭代运算

$$y_i(t+1) = f\left(\sum_{i=1}^{n} w_{ij} y_i(t)\right), \quad 1 \le j \le n, \tag{33}$$

其中, 函数 f 为阈值型. 重复这一步骤, 直到新的迭代不能再改变节点的输出为止, 即收敛为止.

这时, 各节点的输出与输入样例达到最佳匹配. 否则转第 (2) 步继续.

• 深度学习

模仿人脑处理信息的过程——分层处理机制. 底层捕捉输入的“简单” 特征, 高层通过组合底层特征从而形成更加复杂抽象化的特征. 深度学习由简到繁, 自动实现特征提取.

深度学习结构

如图 13.

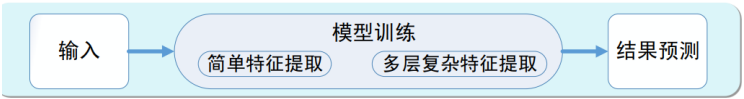


图 13: 深度学习结构

工业 4.0 时代的机器学习

机器学习属于数据科学范畴, 被视为一种涵盖数据处理所有方面的研究, 它是人工智能的一个分支, 赋予机器自主学习的能力, 而不需要人工干预. 机器学习使计算机不需要人工编程, 就可以自动化地执行任务.

机器学习的流程

① 数据收集: 这是机器学习流程的第一步, 也是最重要的一步. 计算机根据问题陈述收集相关数据, 这些被收集的数据被称为训练数据, 它们应当准确完整, 以便解决问题;

② 数据预处理: 预处理是为了将采集到的不完整、不一致和错误的数据转换为可行的数据, 以便更好地拟合机器学习模型. 在除去了数据集的问题后, 数据的特征将被提取出来, 并用于模型训练;

机器学习的流程

③ 模型构建: 选择适当的机器学习技术来获得预期结果的过程. 机器学习的模型种类很多, 总体可分为有监督学习、无监督学习和强化学习三种, 不同的技术适合处理的问题不同;

④ 模型训练与测试: 在确定选择哪种模型之后, 预处理的数据集会被分为训练集和测试集两个部分, 分别满足训练和测试两种要求. 模型训练指利用机器学习技术将模型评估的误差降到最小, 当模型训练完成后, 它就被用于测试数据集中进行测试工作, 以评估模型的效率和准确性, 测试的结果会由一些评价指标进行反映;

机器学习的流程

⑤ 性能评估: 利用交叉验证、参数调整和多种机器学习算法, 尝试得到效果更好的算法, 或者使用组合方法将多个算法的结果组合起来;

⑥ 模型执行: 执行模型输出结果, 以便在未来利用模型完成机器学习任务;
由于不同的 VUCA 战略所要实现的目标不同, 这就涉及到了一些具体的机器学习算法.

注

1989 年, 哈佛大学的莱斯利・维利昂特 (Leslie Valiant, 计算学习理论奠基人、2010 年 ACM 图灵奖得主）和他的学生迈克尔・肯斯 (Michael Kearns, 后来担任贝尔实验室人工智能研究部主任）提出了一个公开问题: “弱可学习性是否等价于强可学习性？”

问题的初略表述

如果一个机器学习任务存在着比“随机猜测” 略好一点的“弱学习算法”, 那么是否就必然存在着准确率任意高 (与该问题的理论上限任意接近）的“强学习算法”？

注

直觉上这个问题的答案大概是“否定” 的, 因为我们在现实任务中通常很容易找到比随机猜测稍好一点的算法 (比方说准确率达到 51%)、却很难找到准确率很高的算法 (比方说达到 95%）.

注

1990 年, 麻省理工学院的罗伯特・夏柏尔 (Robert Schapire) 在著名期刊 Machine Learning 上发表构造性的论文, 证明了这个问题的答案是“YES”！也就是说, 夏柏尔给出了一个过程, 直接按这个过程进行操作就能将弱学习算法提升成强学习算法. 过程的要点是考虑一系列“基学习器”, 让“后来者” 重点关注“先行者” 容易出错的部分, 然后再将这些基学习器结合起来.

注

遗憾的是, 这个过程仅具备理论意义, 并非一个能付诸实践的实用算法, 因为它要求知道一些实践中难以事先得知的信息, 比方说在解决一个问题之前, 先要知道这个问题的最优解有多好.

后来夏柏尔到了新泽西的贝尔实验室工作, 在这里遇见加州大学圣塔克鲁兹分校毕业的约夫・弗洛恩德 (Yoav Freund). 凑巧的是, 弗洛恩德曾经研究过多学习器的结合. 两人开始合作.

注

在 1995 年欧洲计算学习理论会议 (注: 该会议已经并入 33rd Annual Conference on Learning Theory (COLT)) 发表了一个实用算法, 完整版于 1997 年在 Journal of Computer and System Sciences 发表. 这就是著名的 AdaBoost.
夏柏尔和弗洛恩德因这项工作获 2003 年“哥德尔奖”、2004 年“ACM 帕里斯·卡内拉基斯 (Paris Kanellakis) 理论与实践奖”.

AdaBoost 的算法仅需“十来行代码”. 算法和算法修改版能应用于诸多类型的任务.

Example

在人脸识别领域被誉为“第一个实时人脸检测器”、获得 2011 年龙格-希金斯 (Longuet-Higgins) 奖的维奥拉-琼斯 (Viola-Jones) 检测器就是基于 AdaBoost 研制的, AdaBoost 为何未发生过拟合.
夏柏尔和弗洛恩德很快意识到 1997 理论中的问题. 1998 年, 他们与后来领导伯克利著名的西蒙斯计算理论研究所的彼得・巴特莱特 (Peter Bartlett) 和李伟上 (Wee Sun Lee) 合作发表了一个新的基于“间隔 (margin)” 的理论.

Example

“间隔” 是机器学习中一个非常重要的概念. 大致来说, 如图 14 所示, 假定我们用一个划分超平面把不同类别的样本分开, 那么某个样本点与超平面的“距离” 就是这个样本点相对该超平面的“间隔”. 考虑所有样本点相对这个超平面的“最小间隔”, 就定义出了“超平面的间隔”.

机器学习中著名的支持向量机 SVM 就是通过优化技术来求解出间隔最大的划分超平面, 换一个角度看, 就是试图使样本点相对超平面的“最小间隔” 尽可能大. 在夏柏尔等人的新理论中, AdaBoost 的泛化误差界包含一个关于间隔的阈值项 θ, 并且 θ 出现在分母上; 这意味着间隔越大, 泛化误差就可能会越小.

Example

这个理论漂亮地解释了 AdaBoost 为什么没有发生过拟合: 这是因为即便训练误差达到 0, 间隔仍有可能增大. 如图 14, 超平面 B 已经把两类训练样本点完全分开, 其训练误差为 0; 继续训练可能找到超平面 A, 训练误差仍为 0, 但是 A 的间隔比 B 更大, 所以泛化误差可以进一步减小.

AdaBoost 后来衍生出一个庞大的算法家族, 统称 Boosting, 是机器学习中“集成学习” 的主流代表性技术.

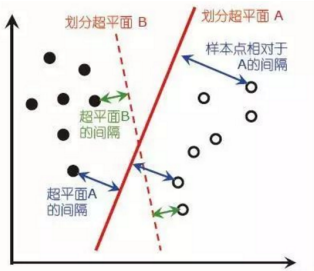


图 14: 学习系统

即便在当今这个 “深度学习时代”, Boosting 仍发挥着重要作用.

Example

经常在许多数据分析竞赛中打败深度神经网络而夺魁的 XGBoost, 就是 Boosting 家族中 GradientBoost 算法的一种高效实现.

定理

Schapire, 1998 对任意 $\delta > 0$ 和 $\theta > 0$, 以至少是 $1 - \delta$ 的概率在大小是 m 的的样本 S 上随机选取样本, 每个候选分类器 $f \in \mathcal{C}(\mathcal{H})$ 满足下面的给定界:

$$\Pr_D[yf(\mathbf{x}) < 0] \leq \Pr_S[yf(\mathbf{x}) \leq \theta] + O\left(\frac{1}{\sqrt{m}}\left(\frac{\ln m \ln |\mathbf{H}|}{\theta^2} + \ln \frac{1}{\delta}\right)^{1/2}\right). \tag{34}$$

定理

Breiman, 1999 对任意 $\delta > 0$, 以至少是 $1 - \delta$ 的概率在 m 大小的样本 S 随机选取样本, 每个候选分类器 $\mathsf{f} \in \mathcal{C}(\mathcal{H})$ 满足下面的给定界:

$$\Pr[\mathsf{yf}(\mathbf{x}) < 0] \leq \mathsf{R}\left(\ln(2\mathsf{m}) + \ln\frac{1}{\mathsf{R}} + 1\right) + \frac{1}{\mathsf{m}}\ln\frac{|\mathcal{H}|}{\delta}. \tag{35}$$

其中 $\theta = \hat{\mathsf{y}}_1\mathsf{f}(\hat{\mathbf{x}}_1) > 4\sqrt{\frac{2}{|\mathbf{H}|}}, \mathsf{R} = \frac{32\ln 2|\boldsymbol{\mu}|}{\mathsf{m}\theta^2} \leq 2\mathsf{m}.$

对任意 $\delta > 0$, 以至少是 $1 - \delta$ 的概率在 $\mathsf{m} \geq 5$ 大小的样本 S 随机选取样本, 每个候选分类器 $\mathsf{f} \in \mathcal{C}(\mathcal{H})$ 满足下面的给定界:

$$\Pr[\mathsf{yf}(\mathbf{x}) < 0] \leq \frac{2}{\mathsf{m}} + \inf_{\theta \in (0,1]} \left[\Pr[\mathsf{yf}(\mathbf{x}) < \theta] + \frac{7\mu + 3\sqrt{3\mu}}{3\mathsf{m}} + \sqrt{\frac{3\mu}{\mathsf{m}} \Pr[\mathsf{yf}(\mathbf{x}) < \theta]} \right], \tag{36}$$

其中 $\mu = \frac{8}{\theta^2} \ln \mathsf{m} \ln(2|\mathcal{H}|) + \ln \frac{2|\mathcal{H}|}{\delta}$.

众所周知, 以支持向量机为代表的一大类统计学习方法都在试图最大化 “最小间隔”, 而这个新理论揭示: 若能最大化 “平均间隔” 同时最小化 “间隔方差”, 得到的学习器会更好?

周志华的博士生张腾等开始了这方面的探索. 2014 年左右建立起 “最优间隔分布学习机 (Optimal margin Distribution Machine, ODM)” 这个新的算法族,

ODM 算法族包括二分类、多分类、聚类、半监督等学习算法.

2013 年的工作引起了很多反响, 如在 2014 年国际人工智能大会 (AAAI) 上, 国际人工智能学会主席、卡内基梅隆大学机器学习系主任曼纽拉・维罗索 (Manuela Veloso) 教授的 Keynote 报告将它作为人工智能领域的重要进展介绍, 称其使 “间隔理论复兴”, 为 “学习算法设计带来了新洞察”.

虽然 2013 理论相应的泛化误差界在当时是最紧致的, 但今后会不会有人基于其他的间隔物理量获得更紧的界, 导致我们关于 “AdaBoost 为何未发生过拟合” 的答案和 “最大化平均间隔同时最小化间隔方差” 的算法指导思想被推翻呢?

六年后, 在 2019 年底的 NeurIPS 会议上, 丹麦奥胡斯大学的阿兰・格洛隆德 (Allan Grønlund)、卡斯柏・拉森 (Kasper G. Larsen)、莱尔・卡玛 (Lior Kamma)、亚历山大・马塞厄森 (Alexander Mathiasen) 与加州大学伯克利分校的杰拉尼・纳尔逊 (Jelani Nelson) 合作发表了一篇论文.

纳尔逊是美国总统奖和斯隆研究奖得主, 拉森在 STOC 和 FOCS 曾两获最佳学生论文奖, 是理论计算机科学界的新星, 卡玛则毕业于以色列魏兹曼研究所. 理论计算机科学家出手机器学习理论问题, 是近年来的一个重要趋势.

这篇论文最终证明了 2013 年给出的已经几乎是最紧的泛化误差上界, 至多再改进一个 log 因子, 并且这个上界已经与下界匹配, 理论上不可能有更好的结果!

了解相关内容的参考文献, ODM 算法.