Lecture 8： Review

18th Jun. 2024

第一章 绪论

- 人工智能的定义与发展
  - 什么是人工智能？什么是智能机器？
    - 研究理解和模拟人类智能、智能行为及其规律
    - 能够在各类环境中自主地或交互地执行各种拟人任务的机器
  - 3个主流学派
    - 逻辑学派（符号主义方法）
    - 仿生学派（连接主义方法）
    - 控制论学派（行为主义方法）
- 弱人工智能和强人工智能
  - 一种是希望借鉴人类的智能行为，研制出更好的工具以减轻人类智力劳动，一般称为“弱人工智能”，类似于“高级仿生学”
  - 另一种是希望研制出达到甚至超越人类智慧水平的人造物，具有心智和意识、能根据自己的意图开展行动，一般称为“强人工智能”，实则可谓“人造智能”
- 人工智能的应用与发展
  - AI 领域包括知识表示和推理、规划与学习、机器人学、自然语言处理、机器学习、模式识别、视觉系统 等

第二章 知识表示和推理

- 命题逻辑
  - 什么是命题？
    - 命题就是具有真假意义的陈述句
    - 原子命题：不能分解成更简单的陈述语句
    - 复合命题：有联结词、标点符号和原子命题等复合结构的命题
  - 命题逻辑符号：
    - 命题常元：T、F
    - 命题符号：P、Q、R等
    - 联结词：否定、合取、析取、蕴涵、等价
    - 括号
  - 命题公式
  - 联结词的优先级
  - 合取范式、析取范式
  - 命题演算形式系统PC

- 谓词逻辑
  - 什么是谓词？
    - 用于刻画个体的性质、状态和个体之间关系的语言成分
    - 一个谓词可以分为谓词名和个体两部分
  - 谓词逻辑：
    - 常量符号、变量符号、函数符号、谓词符号、联结词
    - 量词：全称量词∀、存在量词∃
  - 项和公式
    - 项的递归定义、公式的定义
    - 约束变元和自由变元：
      - 通常把位于量词后面的单个谓词或者是用括号括起来的合式公式成为量词的辖域，辖域内与量词中指导变元同名的变元称为约束变元，不受约束的变元称为自由变元
      - 例：$P(x) \wedge \exists x[P(x) \vee Q(x)]$
      - 换名规则

- 谓词逻辑
  - 例：“某些患者喜欢所有医生。没有患者喜欢庸医。所以没有医生是患者。”目的是证明G是F1和F2的逻辑结论

    解：P(x)表示“x是患者”，D(x)表示“x是医生”，Q(x)表示“x是庸医”，L(x, y)表示“x喜欢y”

$$F_1 : (\exists x)(P(x) \wedge (\forall y)(D(y) \rightarrow L(x, y)))$$

$$F_2 : (\forall x)(P(x) \rightarrow (\forall y)(Q(y) \rightarrow \neg L(x, y)))$$

$$G : (\forall x)(D(x) \rightarrow \neg Q(x))$$

- 谓词逻辑
  - 解释　An interpretation is a pair $\Im = \langle D, I \rangle$
    - D is the domain, can be any non-empty set
    - I is a mapping from the set of predicate and function symbols
    - If P is a predicate symbol of arity n, $I(P)$ is an n-ary relation over D, *i.e.*, $I(P) \subseteq D^n$
      - If p is a 0-ary predicate symbol, *i.e.*, a propositional symbol, $I(p) \in \{true, false\}$
    - If f is a function symbol of arity n, $I(f)$ is an n-ary function over D, *i.e.*, $I(f) : D^n \to D$
      - If c is a 0-ary function symbol, *i.e.*, a constant symbol, $I(c) \in D$
  - 项的指派
  - 满足：原子公式、联结词、量词
  - 可满足性
  - 逻辑蕴涵

- 谓词逻辑
  - 10个常用的等价式：双重否定律、交换律、结合律、分配律、德摩根律、吸收率、补余律、联结词化归律、量词转换律、量词分配律
  - 9个永真蕴涵式：化简式、附加式、析取三段论、假言推理、拒取式、假言三段论、二难推理、全称特化、存在特化
  - 谓词逻辑形式系统FC
    - 合取范式、析取范式
    - 前束范式
    - 前束标准型
  - 例2.5（见书本57页）

第二章 知识表示和推理

归结推理

- 我们希望找到一种自动的推理程序来判断“KB逻辑蕴涵α”是否成立
- 对于一个推理程序：
  - 它是合理的（ Sound） 是指：如果该推理程序认为答案为yes， 那么“KB逻辑上蕴涵α”是成立的
  - 它是完备的（ Complete） 是指：如果“KB逻辑上蕴涵α”， 那么该推理程序会认为答案为yes

- 什么是归结原理：
  - 在定理证明系统中， 已知一个公式集F1，F2， …， Fn， 要证明一个公式W (定理)是否成立， 即要证明W是公式集的逻辑推论时，一种证明法就是要证明F1 ∧ F2 ∧…∧ Fn →W为永真式

- 文字：原子公式及其否定
- 子句：任何文字的析取，某个文字本身也都是子句。
- 空子句：不包含任何文字的子句，空子句式永假的，不可满足
- 子句集：由子句构成的集合（子句的合区）

- 归结推理
  - 归结式的定义及性质
    - 对于任意两个子句C1和C2， 若C1中有一个文字L， 而C2中有一个与L成互补的文字$\neg L$， 则分别从C1和C2中删去L和$\neg L$， 并将其剩余部分组成新的析取式。 这个新的子句被称为C1和C2关于L的归结式
    - 例如， P和$\neg P$的归结式为空子句， 记作()、 □或NIL；
      (W, R, Q)和(W, S, $\neg R$)关于R的归结式为(W, Q, S)
    - 定理：两个子句的归结式是这两个子句的逻辑推论，
      如$\{(P, C1), (\neg P, C2)\} \models (C1, C2)$
  - 鲁滨逊归结原理
    - 子句集中子句之间是合取关系， 只要有一个子句不可满足， 则子句集就不可满足
    - 基本思想：
    - 检查子句集 S 中是否包含空子句， 若包含， 则 S 不可满足
    - 若不包含， 在 S 中选择合适的子句进行归结， 一旦归结出空子句，就说明 S 是不可满足的

- 归结推理
  - 命题逻辑的归结原理和过程
    - 命题逻辑中， 若给定前提集F和命题P， 则归结证明过程可归纳如下：

1. 把F转化成子句集表示， 得到子句集S_0;
2. 把命题P的否定式$\neg P$也转化成子句集表示，并将其加到S_0中， 得$S = S_0 \cup S_{\neg P}$，
3. 对子句集S反复应用归结推理规则（ 推导） ，直至导出含有空子句的扩大子句集为止。 即出现归结式为空子句时， 表明已找到矛盾， 证明过程结束。

- 归结推理
  - 命题逻辑的归结原理和过程

例1：设已知前提集为

P……(1)　　$(P \wedge Q) \rightarrow R$……(2)

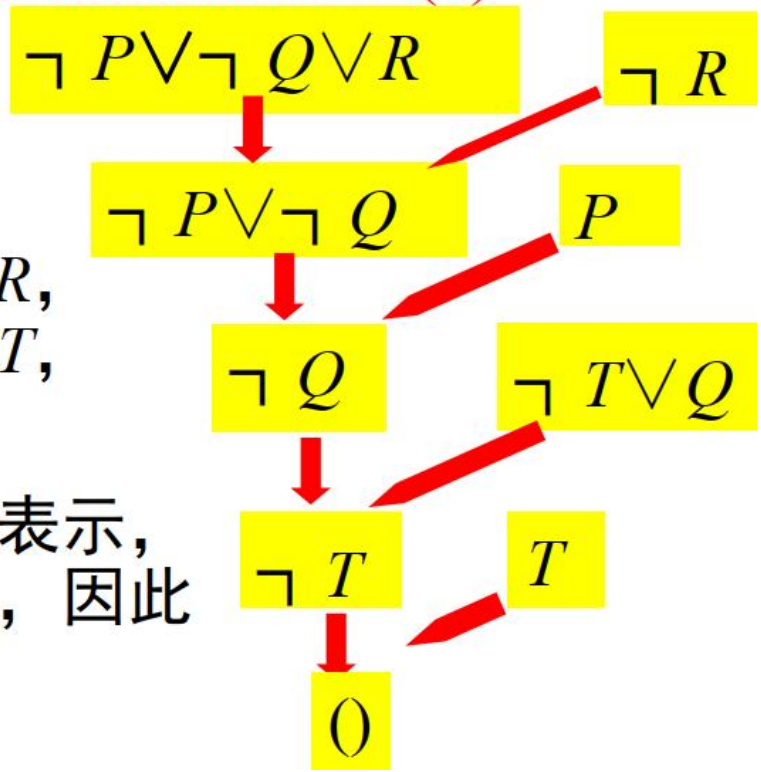
$(S \vee T) \rightarrow Q$…(3)　　T……(4)

求证R。

证明：化成子句集

$S = \{P, \neg P \vee \neg Q \vee R, \neg S \vee Q, \neg T \vee Q, T, \neg R\}$

- 归结可用图的演绎树表示，由于根部出现空子句，因此命题R得证。



- 归结推理
  - 谓词逻辑的归结原理和过程
    - 把合式公式化成子句式
    - 若S中两个子句间有相同互补文字的谓词， 但它们的项不同， 则必须找出对应的不一致项；
    - 进行变量置换， 使它们的对应项一致;
    - 求归结式看能否推导出空子句。

  - 把合式公式化成子句式
    - 消去蕴含符号
    - 把否定符号移到每个谓词前面
    - 变量标准化
    - 消去存在量词（Skolemize）
    - 化为前束式
    - 化为标准型
    - 略去全称量词
    - 消去合取词
    - 子句变量标准化

- 归结推理
  - 合一（Unify）
    - 在谓词逻辑的归结过程中， 寻找项之间合适的变量置换使表达式一致， 这个过程称为合一

- 一个表达式的项可以是常量符号、变量符号或函数式。
- 表达式的例（instance）是指在表达式中用置换项置换变量后而得到的一个特定的表达式。
- 用$\sigma = \{v_1/t_1, v_2/t_2, \ldots, v_n/t_n\}$来表示任一置换（教材表示相反）。$v_i/t_i$是指表达式中的变量$v_i$以项$t_i$来替换，且不允许$v_i$用与$v_i$有关的项$t_i$（但是$t_i$中可以包含其它变量）作置换。为了便于理解，后续记$\sigma = \{v_1 = t_1, v_2 = t_2, \ldots, v_n = t_n\}$。
- 用σ对表达式E作置换后的例简记为$E\sigma$。

- 归结推理
  - 合一（Unify）
    - 例如，$P(x,g(y,z))\{x=y,y=f(a)\}\Rightarrow P(y,g(f(a),z))$
    - 注意：置换是同时进行的，而不是先后进行的。
    - 可以对表达式多次置换，如用θ和σ依次对E进行置换，记为$(E\theta)\sigma$。其结果等价于先将这两个置换合成（组合）为一个置换，即$\theta\sigma$，再用合成置换对E进行置换，即$\mathrm{E}(\theta)\sigma$

令 $\theta=\{x=f(y),y=z\}$，$\sigma=\{x=\mathrm{a},y=\mathrm{b},z=y\}$

- 1. Get $S=\{x=f(\mathrm{b}),y=y,x=\mathrm{a},y=\mathrm{b},z=y\}$
- 2. Delete $x=\mathrm{a}$; Delete $y=\mathrm{b}$
- 3. Delete $y=y$

$\theta\sigma=S=\{x=f(\mathrm{b}),z=y\}$

这样的合成法可使$(E\theta)\sigma=E(\theta\sigma)$，即可结合。
但置换是不可交换的，即$\theta\sigma\neq\sigma\theta$。
空置换$\epsilon=\{\}$也是一个置换，且$\theta\epsilon=\theta$。

归结推理

最一般的合一项（Most General Unifier）

A substitution σ of two formulas f and g is a Most General Unifier (MGU) if

- σ is a unifier.
- For every other unifier θ of f and g there must exist a third substitution λ such that $\theta = \sigma\lambda$.

This says that every other unifier is "more specialized" than σ.

The MGU of a pair of formulas f and g is unique up to renaming.

示例：

- $P(f(x), z)$ and $P(y, a)$
- $\sigma = \{y = f(a), x = a, z = a\}$ is a unifier, but not an MGU
- $\theta = \{y = f(x), z = a\}$ is an MGU
- $\sigma = \theta\lambda$, where $\lambda = \{x = a\}$

- 归结推理
  - 最一般的合一项（Most General Unifier）
    - 计算MGU：

Given two atomic formulas f and g

1. $\sigma = \{\}$; $S = \{f, g\}$
2. If S contains an identical pair of formulas, stop and return σ as the MGU of f and g.
3. Else find the disagreement set $D = \{e_1, e_2\}$ of S
4. If $e_1 = V$ a variable, and $e_2 = t$ a term not containing V (or vice-versa) then let $\sigma = \sigma\{V = t\}$; $S = S\{V = t\}$; Goto 2
5. Else stop, f and g cannot be unified.

Note: to update σ, we must compose σ with $\{V = t\}$.
A common error is to just add $V = t$ to σ.

- 归结推理
  - 示例

已知：

（1） 会朗读的人是识字的，

（2） 海豚都不识字，

（3） 有些海豚是很机灵的。

证明：有些很机灵的东西不会朗读。

解：把问题用谓词逻辑描述如下，

已知：

(1) $\forall x(R(x) \rightarrow L(x))$

(2) $\forall x(D(x) \rightarrow \neg L(x))$

(3) $\exists x(D(x) \wedge I(x))$

求证： $\exists x(I(x) \wedge \neg R(x))$

- 归结推理
  - 示例

前提化简，待证结论取反并化成子句形，求得子句集：

1. $(\neg R(x), L(x))$
2. $(\neg D(y), \neg L(y))$
3. $D(a)$
4. $I(a)$
5. $(\neg I(z), R(z))$

一个可行的证明过程：

6. $R[4,5]\{z = a\}\ R(a)$
7. $R[1,6]\{x = a\}\ L(a)$
8. $R[2,7]\{y = a\}\ \neg D(a)$
9. $R[3,8]\ ()$

- 归结推理
  - 应用归结原理求解问题的步骤：
    - 已知前提 F 用谓词公式表示， 并化为子句集 S
    - 把待求解的问题 P 用谓词公式表示， 并否定P，再与 answer 构成析取式（ ¬ P ∨ answer ）
    - 把（ ¬ P ∨ answer） 化为子句集， 并入到子句集 S中， 得到子句集S’；
    - 对 S’ 应用归结原理进行归结
    - 若得到归结式answer， 则答案就在answer中

第三章 搜索技术

- 搜索
  - 搜索就是找到智能系统的动作序列的过程
  - 搜索算法的输入式给定的问题，输出时表示为动作序列的方案
  - 一旦有了方案，就可以执行该方案所给出的动作了
  - 求解问题包括：
    - 目标表示
    - 搜索
    - 执行
  - 盲目搜索：
    - 只是可以区分出哪个时目标状态
    - 一般是按预定的搜索策略进行搜索
    - 没有考虑到问题本身的特性，这种搜索具有很大的盲目性，效率不高，不便于复杂问题的求解
  - 启发式搜索：
    - 在搜索过程中加入了与问题有关的启发式信息，用于指导搜索朝着最有希望的方向前进，加速问题的求解并找到最优解

第三章 搜索技术

- 搜索
  - 盲目搜索：
    - 宽度优先搜索
    - 深度优先搜索
    - 迭代加深搜索
  - 宽度优先搜索
    - 沿着树的宽度遍历树的节点，它从深度为0的层开始，直到最深的层次。它可以很容易地用队列实现
    - 宽度优先搜索算法原理：
      - 如果当前的节点不是目标节点，则把当点节点的子孙以任意顺序增加 到队列的后面，并把队列的前端元素定义为current。
      - 如果目标发现，则算法终止。
    - 宽度优先搜索是一种盲目搜索， 时间和空间复杂度都比较高，当目标节点距离初始节点较远时会产生许多无用的节点，搜索效率低
    - 宽度优先搜索中，时间需求是一个很大的问题，特别是当搜索的深度比较大时，尤为严重，但是空间需求是比执行时间更严重的问题

- 搜索
  - 深度优先搜索算法原理：
    - 深度优先搜索生成节点并与目标节点进行比较是沿着树的最大深度方向进行的，只有当上次访问的节点不是目标节点，而且没有其他节点可以生成的时候，才转到上次访问节点的父节点。
    - 转移到父节点后，该算法会搜索父节点的其他的子节点。
    - 深度优先搜索也称为回溯搜索，它总是首先扩展树的最深层次上的某个节点， 只是当搜索遇到一个死亡节点（非目标节点而且不可扩展），搜索方法才会返回并扩展浅层次的节点。
    - 上述原理对树中的每一节点是递归实现的(实现该递归用栈)。
    - 深度优先搜索的优点是比宽度优先搜索算法需要较少的空间，该算法只需要保存搜索树的一部分，它由当前正在搜索的路径和该路径上还没有完全展开的节点标志所组成。
    - 深度优先搜索的存储器要求是深度约束的线性函数

- 搜索
  - 迭代加深搜索算法原理：
    - 有界深度优先搜索过程总体上按深度优先算法方法进行，但对搜索深度需要给出一个深度限制dm，当深度达到了dm的时候，如果还没有找到解答，就停止对该分支的搜索，换到另外一个分支进行搜索。
    - 策略说明：
      - 深度限制dm很重要。当问题有解，且解的路径长度小于或等于dm时，则搜索过程一定能够找到解，但是和深度优先搜索一样这并不能保证最先找到的是最优解。
      - 但是当dm取得太小，解的路径长度大于dm时，则搜索过程中就找不到解，即这时搜索过程甚至是不完备的
      - 深度限制dm不能太大。当dm太大时，搜索过程会产生过多的无用节点，既浪费了计算机资源，又降低了搜索效率
      - 有界深度搜索的主要问题是深度限制值dm的选取

- 搜索
  - 迭代加深搜索算法原理：
    - 改进方法：
      - 先任意给定一个较小的数作为dm，然后按有界深度算法搜索，若在此深度限制内找到了解，则算法结束；如在此限制 内没有找到问题的解，则增大深度限制dm，继续搜索。
  - 宽度优先搜索需要指数数量的空间，深度优先搜索的空间复杂度和最大搜索深度呈线性关系。
  - 迭代加深搜索对一棵深度受控的树采用深度优先的搜索。它结合了宽度优先和深度优先搜索的优点。和宽度优先搜索一样，它是最优的，也是完备的。但对空间要求和深度优先搜索一样是适中的。

- 搜索
  - 启发式搜索：
    - 如果在选择节点时能充分利用与问题有关的特征信息，估计出节点的重要性，就能在搜索 时选择重要性较高的节点，以便求得最优解
    - A*搜索
    - 迭代加深A*搜索
  - 评估函数
    - 用来评估节点重要性的函数
    - f(x)=g(x)+h(x)
    - 评估函数f(x)定义为从初始节点S_0出发，约束地经过节点x到达目标节点S_g的所有路径中最小路径代价的估计值
    - g(x)——从初始节点S_0到节点x的实际代价；
    - h(x)——从x到目标节点S_g的最优路径的评估代价，它体现了问题的启发式信息，其形式要根据问题的特性确定，h(x)称为启发式函数。

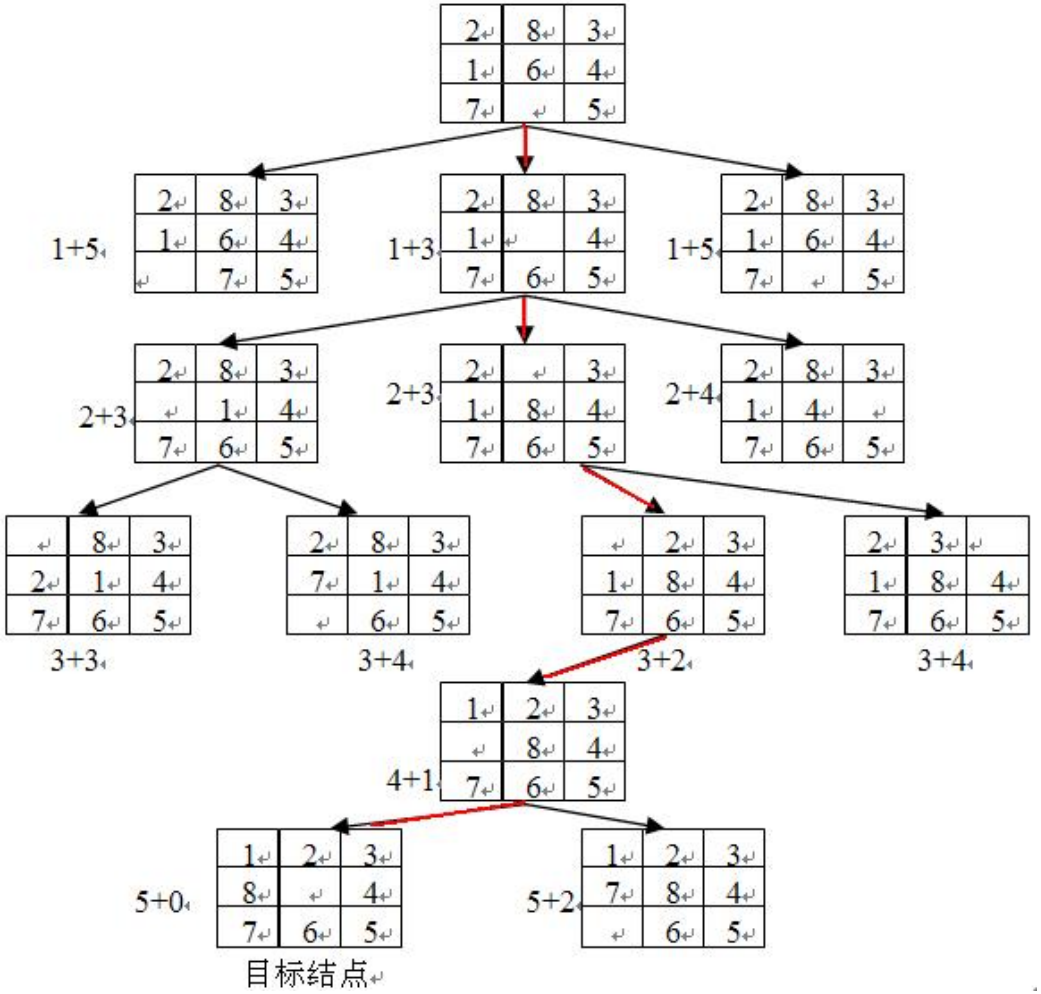
第三章 搜索技术

- 搜索
  - 最好优先搜索算法：
    - （最好优先）搜索是从最有希望的节点开始，并且生成其所有的子节点
    - 计算每个节点的性能（合适性），
    - 选择最有希望的节点进行扩展，而不是仅仅从当前节点所生成的子节点中进行选择。
    - 如果在早期选择了一个错误的节点，最好优先搜索就提供了一个修改的机会
    - 最好优先搜索算法并没有显式地给出如何定义启发式函数，
    - 它不能保证当从起始节点到目标节点的最短路径存在时，一定能够找到它。
  - A*算法就是对启发式函数加上限制后得到的一种启发式搜索算法

- 搜索
  - A*算法
    - 评估函数f*： $f^*(n)=g^*(n)+h^*(n)$
    - $g^*(n)$为起始节点到节点n的最短路径的代价
    - $h^*(n)$是从n到目标节点的最短路径的代价
    - 这样$f^*(n)$就是从起始节点出发通过节点n到达目标节点的最佳路径的总代价的估值。
    - 把估价函数$f(n)$和 $f^*(n)$相比较， $g(n)$是对$g^*(n)$的估价。 $h(n)$是对$h^*(n)$的估价。
    - 有了$g^*(n)$和$h^*(n)$的定义，如果对最好优先的启发式搜索算法中的$g(n)$和$h(n)$做如下的 限制：
      - $g(n)$是对$g^*(n)$估计，且$g(n)>0$
      - $h(n)$是$h^*(n)$的下界，即对任意节点n均有$h(n) \le h^*(n)$

      则称这样得到的算法为A*算法。
    - $h(n) \le h^*(n)$的限制十分重要，它保证A*算法能够找到最优解。

- 搜索
  - A*算法求解八数码问题



- 搜索
  - 迭代加深A*算法
    - 由于A*算法把所有生成的节点保存在内存中，所以A*算法在耗尽计算时间之前一般早已经把空间耗尽了。
    - 迭代加深搜索算法，它以深度优先的方式在有限制的深度内搜索目标节点。
    - 在每个深度上，该算法在每个深度上检查目标节点是否出现，如果出现则停止，否则深度加1继续搜索。
    - 而A*算法是选择具有最小估价函数值的节点扩展。
    - 迭代加深A*搜索算法IDA*是上述两种算法的结合。这里启发式函数用做深度的限制，而不是选择扩展节点的排序。

- 博弈
  - 极大极小过程
    - 极大极小过程是考虑双方对弈若干步之后，从可能的走法中选一步相对好的走法来走，即在有限的搜索深度范围内进行求解。
    - 需要定义一个静态估价函数f，以便对棋局的态势做出评估。
    - 这个函数可以根据棋局的态势特征进行定义。假定对弈双方分别为MAX和MIN，规定：
      - 有利于MAX方的态势：f(p)取正值
      - 有利于MIN方的态势：f(p)取负值
      - 态势均衡的时候：f(p)取零

      其中p代表棋局。

- 博弈
  - MINMAX基本思想

(1) 当轮到MIN走步的节点时，MAX应考虑最坏的情况（即f(p)取极小值）。

(2) 当轮到MAX走步的节点时，MAX应考虑最好的情况（即f(p)取极大值）。

(3) 评价往回倒推时，相应于两位棋手的对抗策略，交替使用（1）和（2）两种方法传递倒推值。

- 博弈
  - MINMAX基本思想
  - 图3-18 所示是向前看两步，共四层的博弈树，用□表示MAX，用○表示MIN，端节点上的数字表示它对应的估价函数的值。在MIN处用圆弧连接，用0表示其子节点取估值最小的格局。
  - 图中节点处的数字，在端节点是估价函数的值，称它为静态值，在MIN处取最小值， 在MAX处取最大值，最后MAX选择箭头方向的走步。

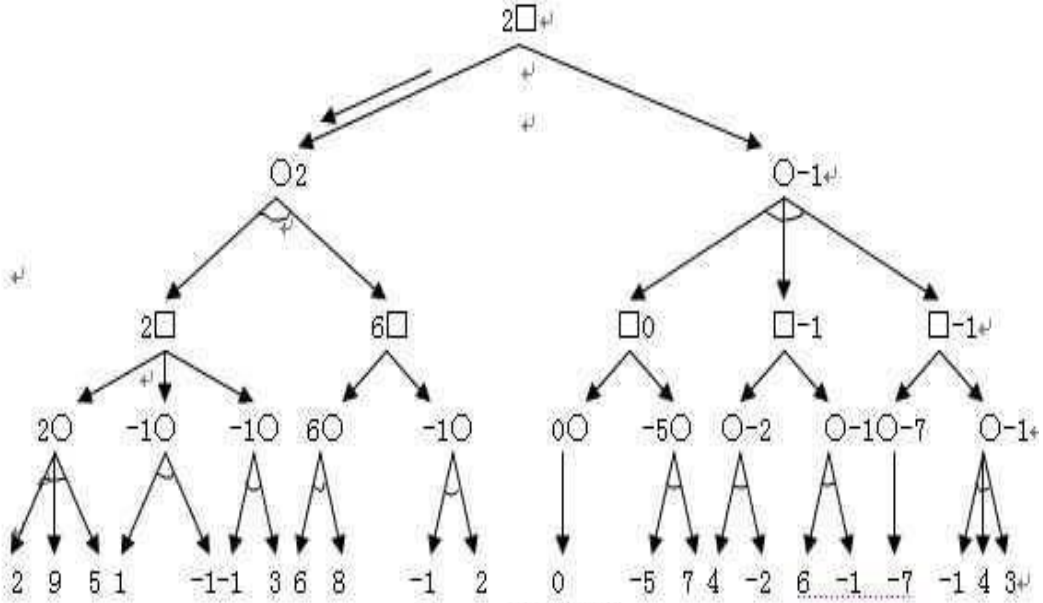
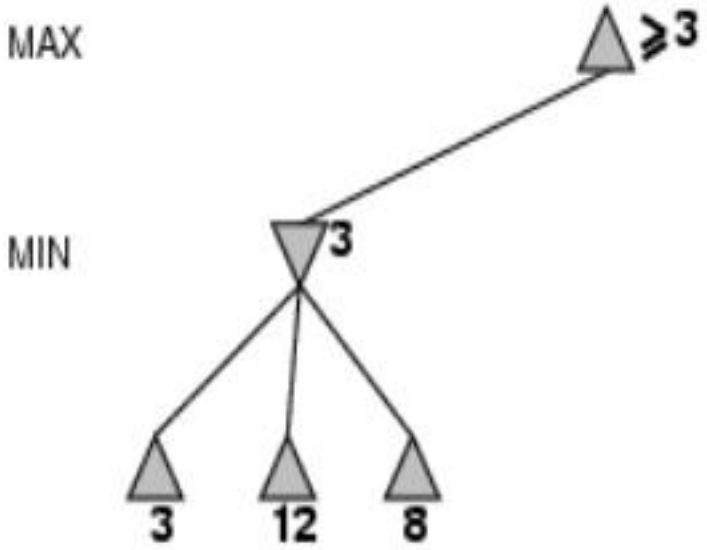


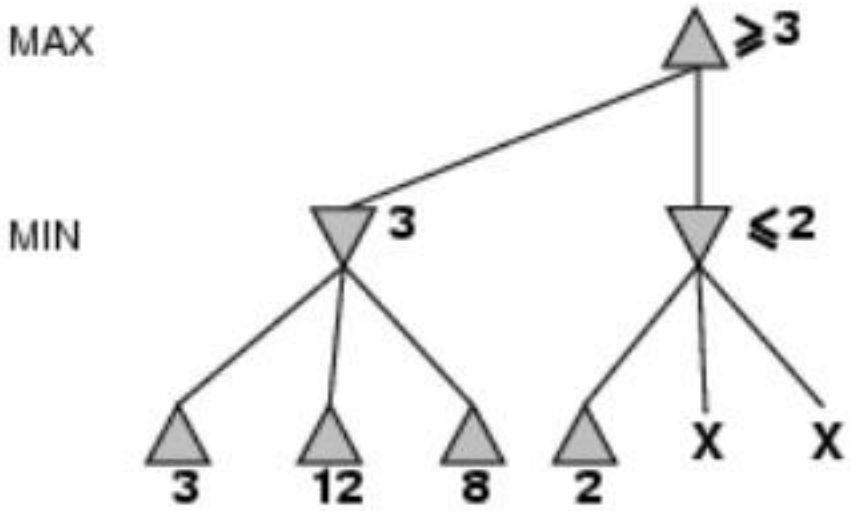
图3-18 四层博弈树

- 博弈
  - $\alpha-\beta$基本思想
    - 极大极小方法是把搜索树的生成和估值这两个过程完全分开。只有在已经生成树之后才开始进行估值，这一分离导致了低效率的策略
    - $\alpha-\beta$剪枝技术是极大极小方法的改进，是一种提高博弈树搜索效率的方法
    - $\alpha-\beta$剪枝技术是一种边生成节点，边计算估值和倒推值的方法，从而剪去某些分枝。
  - $\alpha-\beta$的定义和计算
    - 对于一个“与”节点，它取当前子节点中的最小的倒推值作为它的倒推值的上界,称此值为β值。也就是说β值可以等于其后继节点当前最 小的最终倒推值。“与”节点的β值是永远不会增加的。
    - 对于一个“或”节点，它取当前子节点中得最大的倒推值作为它的倒推值的下界，称此值为α值。也就是说α值可以等于其后继节点当前最 大的最终倒推值。“或”节点的α值是永远不会减少的。

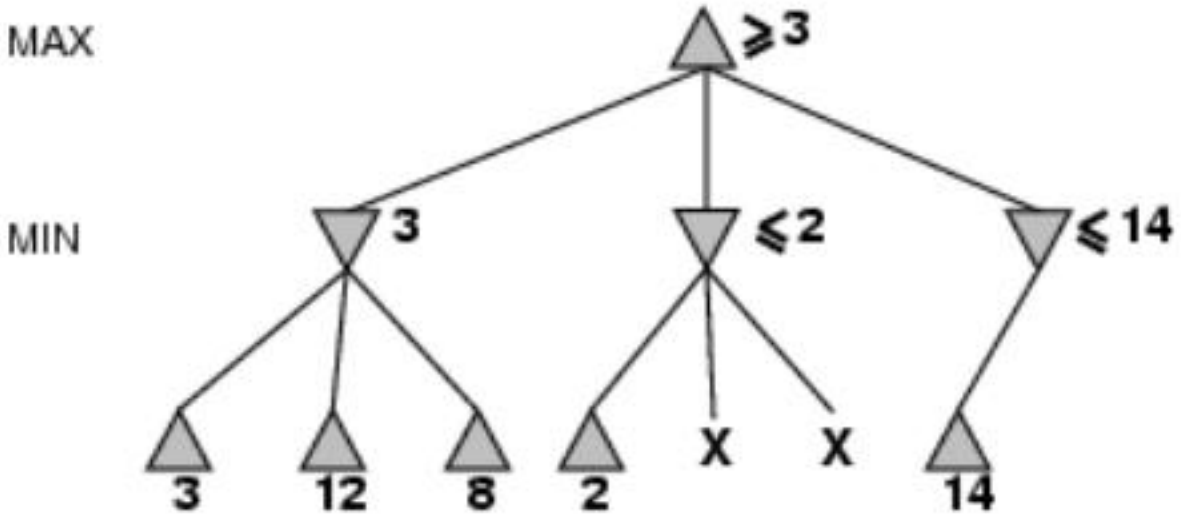
- 博弈
  - $\alpha-\beta$过程



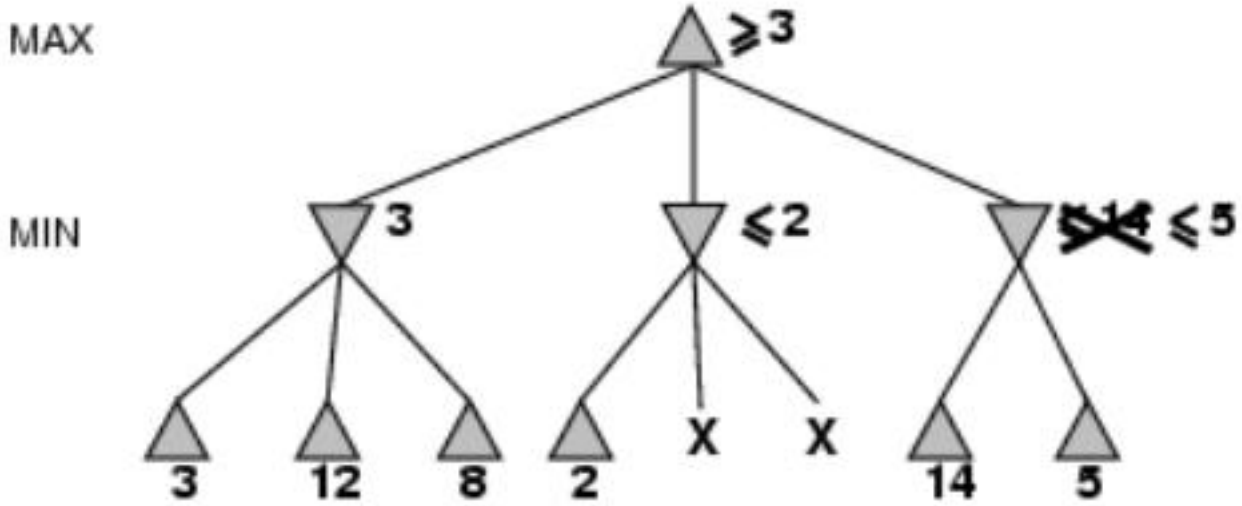
- 博弈
  - $\alpha-\beta$过程



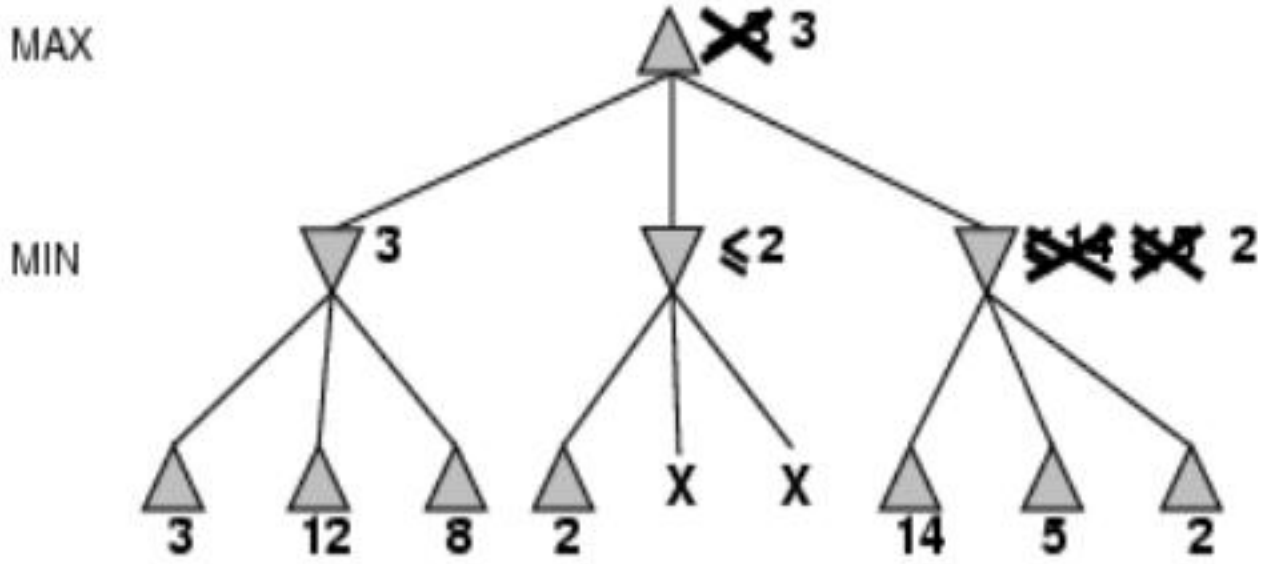
- 博弈
  - $\alpha-\beta$过程



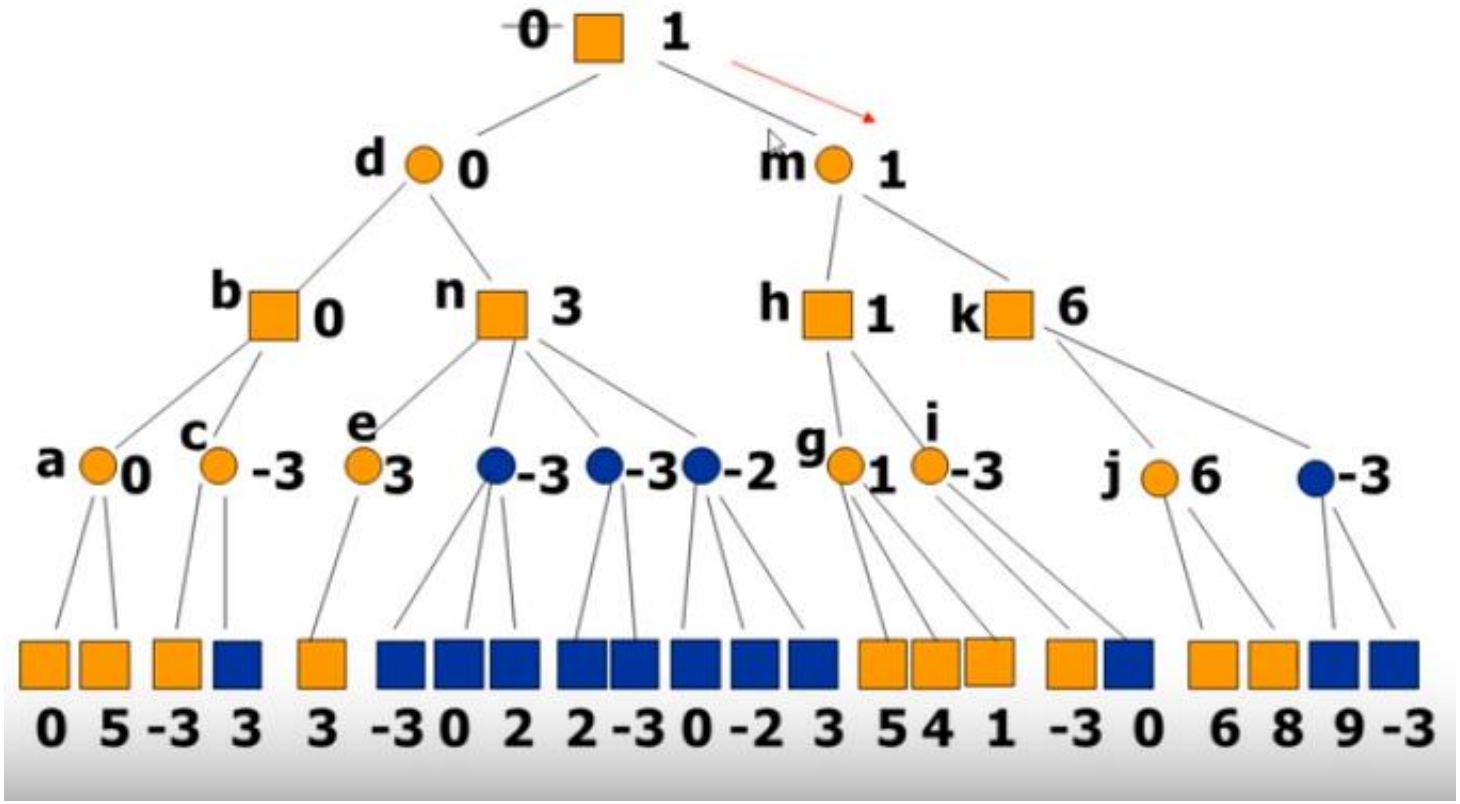
- 博弈
  - $\alpha-\beta$过程



- 博弈
  - $\alpha-\beta$过程



- 博弈
  - $\alpha-\beta$一般规律
    - 任何“或”节点x的α值如果不能降低其父节点的β值,则对节点x以下的分枝可以停止搜索, 并使x的倒推值为α。这种技术称为β剪枝。
    - 任何“与”节点x的β值如果不能升高其父节点的α值,则对节点x以下的分枝可以停止搜索, 并使x的倒推值为β。这种技术称为α剪枝。
    - 要进行$\alpha-\beta$剪枝,至少必须使某一部分的搜索 树生长到最大深度.因为α和β值必须以某个 端节点的静态估值为依据.因此采用$\alpha-\beta$过程都要使用某种深度优先的搜索方法.


0 1
d 0
m 1
b 0
n 3
h 1
k 6
a 0
c -3
e 3
-3
-3
-2
g 1
i -3
j 6
-3
0 5 -3 3 3 -3 0 2 2 -3 0 -2 3 5 4 1 -3 0 6 8 9 -3

- 高级搜索
  - 爬山法搜索—局部搜索
  - 模拟退火算法
  - 遗传算法

  - 模拟退火算法
    - 固体退火是将固体加热，使其温度达到充分高，随着温度的升高，固体内部的粒子逐渐变为无序状态，然后再将固体徐徐降温，在降温的过程中，固体内部的粒子逐渐趋于有序状态，在每个温度都达到平衡，直至常温，固体达到基础状态
    - 类比优化搜索过程，在一定温度下，搜索从一个状态随机地变化到另一个状态，随着温度的不断降低，直至最低温度，搜索过程以接近 1 的概率停留在最优解。
    - 是对爬山算法的优化改进，爬山算法在搜索过程中只接受更优的邻近解，直至搜索不到更优解为止，算法简单易实现，但是却极有可能停止在局部最优解

- 高级搜索
  - 模拟退火算法
    - 固体退火是将固体加热，使其温度达到充分高，随着温度的升高，固体内部的粒子逐渐变为无序状态，然后再将固体徐徐降温，在降温的过程中，固体内部的粒子逐渐趋于有序状态，在每个温度都达到平衡，直至常温，固体达到基础状态
    - 算法引入一个温度参数，当搜索到较差的邻近解时，利用温度参数和目标函数值之差共同确定一个概率参数，利用此概率参数决定是否接受较差的邻近解，概率参数P的定义为

$$P = e^{-\frac{\delta}{t}}$$

    - 其中，δ为邻近解与当前解的目标函数之差，t为温度参数，该参数对应于固体退火过程中的温度，随着搜索过程的不断推进而不断减小，直至算法达到终止条件。
    - 在温度下降足够慢时，算法找到全局最优解的概率接近 1

- 高级搜索
  - 遗传算法
    - 固体退火是将固体加热，使其温度达到充分高，随着温度的升高，固体内部的粒子逐渐变为无序状态，然后再将固体徐徐降温，在降温的过程中，固体内部的粒子逐渐趋于有序状态，在每个温度都达到平衡，直至常温，固体达到基础状态
    - 第1步：随机产生初始种群，个体数目一定，每个个体表示为染色体的基因编码
    - 第2步：计算个体的适应度，并判断是否符合优化准则，若符合，输出最佳个体及其代表的最优解，并结束计算；否则转向第3 步；
    - 第3步：依据适应度选择再生个体，适应度高的个体被选中的概率高，适应度低的个体可能被淘汰；
    - 第4步：按照一定的交叉概率和交叉方法，生成新的个体；
    - 第5步：按照一定的变异概率和变异方法，生成新的个体；
    - 第6步：由交叉和变异产生新一代的种群，返回到第2步。

第四章 高级搜索

- 高级搜索
  - 利用模拟退火和遗传算法解决TSP问题

概率基础

如果$Pr(B|A) = Pr(B)$，则B和A独立

如果$Pr(B|A,C) = Pr(B|A)$，则给定A的前提下，B和C条件独立

乘法/链式法则

$$Pr(A,B)=Pr(A)Pr(B|A)=Pr(B,A)=Pr(B)Pr(A|B)$$

$$Pr(A,B_1,B_2,B_3)=Pr(A)Pr(B_1|A)Pr(B_2|A,B_1)Pr(B_3|A,B_1,B_2)$$

加法法则

$$Pr(A) = \sum_B Pr(A,B) = \sum_{i=1}^{n} Pr(A,B_i)$$

$$= \sum_{i=1}^{n} Pr(A|B_i)Pr(B_i)$$

- 贝叶斯定理

后验概率

$$Pr(B|A) = \frac{Pr(A,B)}{Pr(A)}$$

先验概率 似然度

$$= \frac{Pr(B)Pr(A|B)}{Pr(A)}$$

标准化常量

$$= \frac{Pr(B)Pr(A|B)}{Pr(A,B)+Pr(A,B^c)}$$

第五章 不确定性知识表示和推理

概率基础

- 联合概率分布
- 边缘概率分布
- 最大后验概率状态

- 联合概率分布：$p(I,D,G)$

I	D	G	Prob.
i^0	d^0	g^1	0.126
i^0	d^0	g^2	0.168
i^0	d^0	g^3	0.126
i^0	d^1	g^1	0.009
i^0	d^1	g^2	0.045
i^0	d^1	g^3	0.126
i^1	d^0	g^1	0.252
i^1	d^0	g^2	0.0224
i^1	d^0	g^3	0.0056
i^1	d^1	g^1	0.06
i^1	d^1	g^2	0.036
i^1	d^1	g^3	0.024

- 边缘概率：$p(I)=\sum_{D,G} p(I,D,G)$

$$\Rightarrow \begin{cases} p(I=i^0)=0.6 \\ p(I=i^1)=0.4 \end{cases}$$

$$p(i^0)=\sum_{D,G} p(i^0,D,G)$$
$$=p(i^0,d^0,g^1)+p(i^0,d^0,g^2)+p(i^0,d^0,g^3)$$
$$+p(i^0,d^1,g^1)+p(i^0,d^1,g^2)+p(i^0,d^1,g^3)$$

- 最大后验概率状态：$\{I^*,D^*,G^*\}=\arg\max_{I,D,G} p(I,D,G)$

$$\Rightarrow \begin{cases} I^*=i^1 \\ D^*=d^0 \\ G^*=g^1 \end{cases}$$

朴素贝叶斯

- 思想：朴素贝叶斯假设，又称条件独立性假设

对于特征$X=(x_1,x_2,...,x_n)$，满足$x_i \perp x_j \mid y\ (i \neq j)$

$$p(X\mid y)=p(x_1,x_2,...,x_n\mid y)=\prod_{j=1}^{n}p(x_j\mid y)$$

- Motivation：简化运算
- 条件独立假设，用于分类的特征在分类模型确定的条件下是条件独立的
- 做法：根据贝叶斯定理来估计每个类别的后验概率。

$$p(y\mid x)=\frac{p(x,y)}{p(x)}=\frac{p(x\mid y)p(y)}{p(x)}=\frac{p(x\mid y)p(y)}{\sum_i p(x\mid y_i)p(y_i)}\propto p(x\mid y)p(y)$$

- 朴素贝叶斯法的目标是找到

$$y=\arg\max_y p(y\mid x)=\arg\max_y \frac{p(x,\ y)}{p(x)}=\arg\max_y p(x\mid y)p(y)$$

- 朴素贝叶斯
  - 给定一个包含 M 个文本的数据集，其中每个有 K 维特征向量 $X = (x_1, \ldots, x_K)$ 和一个情感标签 e_i，为了预测测试文本，需要估计：

$$
\begin{aligned}
\arg\max_{e_i} p(e_i|X) &= \arg\max_{e_i} \frac{P(X|e_i)p(e_i)}{p(X)} \\
&= \arg\max_{e_i} p(X|e_i)p(e_i) \\
&= \arg\max_{e_i} \prod_{k=1}^{K} p(x_k|e_i)p(e_i) \\
&= \arg\max_{e_i} \sum_{j=1}^{M} \prod_{k=1}^{K} p(x_k|e_i, d_j)p(d_j, e_i)
\end{aligned}
$$

- 朴素贝叶斯
  - 假设现在有一个文本：”*Step by step, we succeed*”。

X	step	by	we	succeed	joy	sad
onehot	1	1	1	1	0.9	0.1
TF	0.4	0.2	0.2	0.2	0.9	0.1
TF-IDF	1.03	0.7	0.6	1.16	0.9	0.1

$$p(x_k|d_j, e_i) = \frac{x_k}{\sum_{k=1}^{K} x_k}$$

X	step	by	we	succeed	joy	sad
onehot	0.25	0.25	0.25	0.25	0.9	0.1
TF	0.4	0.2	0.2	0.2	0.9	0.1
TF-IDF	0.30	0.20	0.17	0.33	0.9	0.1

- 朴素贝叶斯
  - 假设现在有一个文本：”*Step by step, we succeed*”。

Documnt	sentence	joy	sad
train1 (d1)	Step by step, we will succeed.	0.9	0.1
train2 (d2)	We step on shit.	0.3	0.7
test1 (d3)	We succeed.	?	?

X	x_1	x_2	x_3	x_4	x_5	x_6	x_7	emotion	
Document	step	by	we	succeed	on	shit	will	joy	sad
train1 (d1)	0.33	0.17	0.17	0.17	0	0	0.17	0.9	0.1
train2 (d2)	0.25	0	0.25	0	0.25	0.25	0	0.3	0.7
test1 (d3)	0	0	0.5	0.5	0	0	0	?	?

- 朴素贝叶斯
  - 假设现在有一个文本：”*Step by step, we succeed*”。
  - 为了预测文本 $X_3 = (x_3, x_4)$ 的情感 e_i ，我们需要估计：

$$p(e_i|X_3) \propto \sum_{j=1}^{2} p(x_3|e_i, d_j)p(x_4|e_i, d_j)p(d_j, e_i)$$

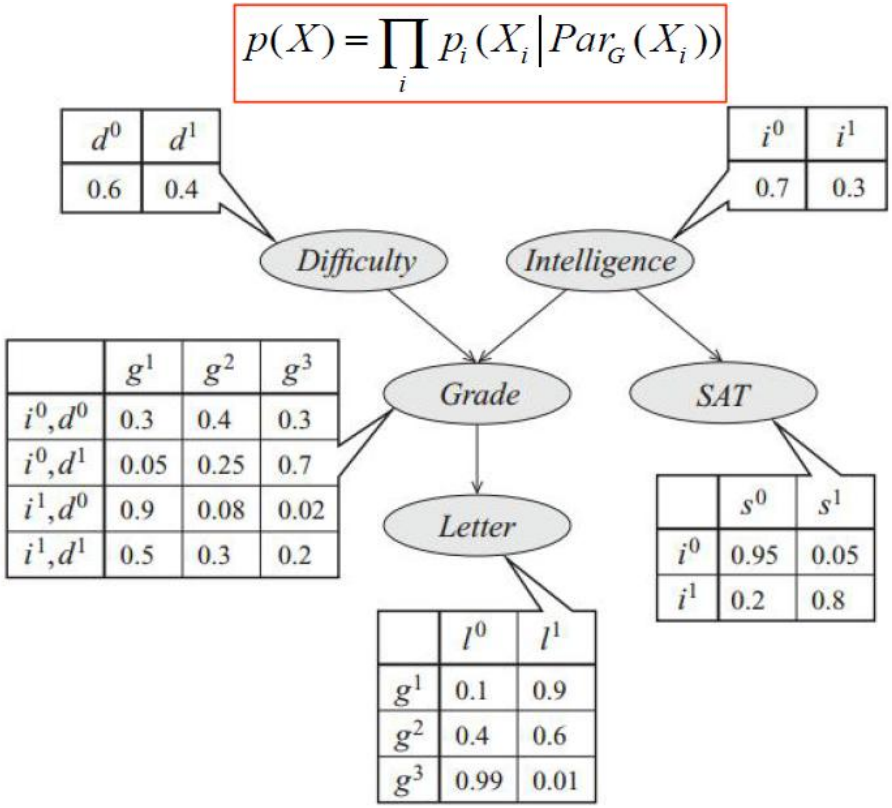
$$\begin{aligned} p(joy|X_3) &\propto p(x_3|joy, d_1)p(x_4|joy, d_1)p(d_1, joy) \\ &+ p(x_3|joy, d_2)p(x_4|joy, d_2)p(d_2, joy) \\ &= 0.17 \times 0.17 \times 0.9 + 0.25 \times 0 \times 0.3 = \mathbf{0.02601} \end{aligned}$$

$$\begin{aligned} p(sad|X_3) &\propto p(x_3|sad, d_1)p(x_4|sad, d_1)p(d_1, sad) \\ &+ p(x_3|sad, d_2)p(x_4|sad, d_2)p(d_2, sad) \\ &= 0.17 \times 0.17 \times 0.1 + 0.25 \times 0 \times 0.7 = \mathbf{0.00289} \end{aligned}$$

概率图模型

令 G 为定义在$\{X_1, X_2, \ldots, X_N\}$ 上的一个贝叶斯网络，其联合概率分布可以表示为各个节点的条件概率分布的乘积：

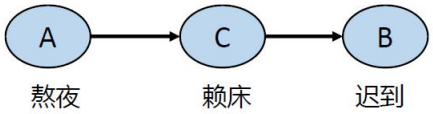
$$p(X)=\prod_i p_i(X_i \mid Par_G(X_i))$$



联合概率分布：

$$p(D,I,G,S,L)$$
$$=P(D)P(I)P(G \mid I,D)P(S \mid I)P(L \mid G)$$
$$p(d^1,i^0,g^1,s^1,l^1)$$
$$=P(d^1)P(i^0)P(g^1 \mid i^0,d^1)P(s^1 \mid i^0)P(l^0 \mid g^1)$$
$$=0.4\times 0.7\times 0.05\times 0.05\times 0.1$$
$$=0.00007$$

独立性判断



联合概率分布链式法则：

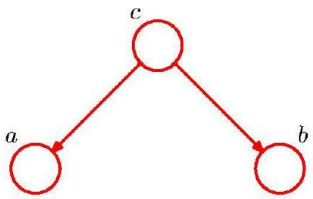
$$p(A,C,B) = P(A)P(C|A)P(B|A,C)$$

条件独立：

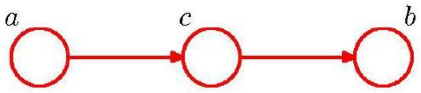
$$P(B|A,C) = P(B|C)$$

$$p(A,C,B) = P(A)P(C|A)P(B|C)$$

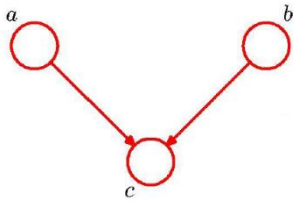
不独立



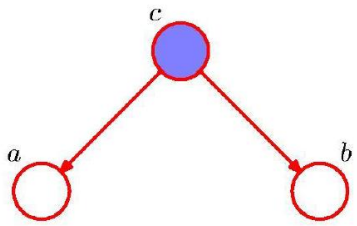
不独立



独立

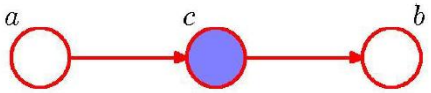


独立



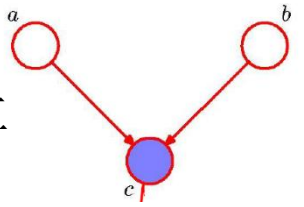
Fork （tail-to-tail）

独立



Chain (head-to-tail)

不独立



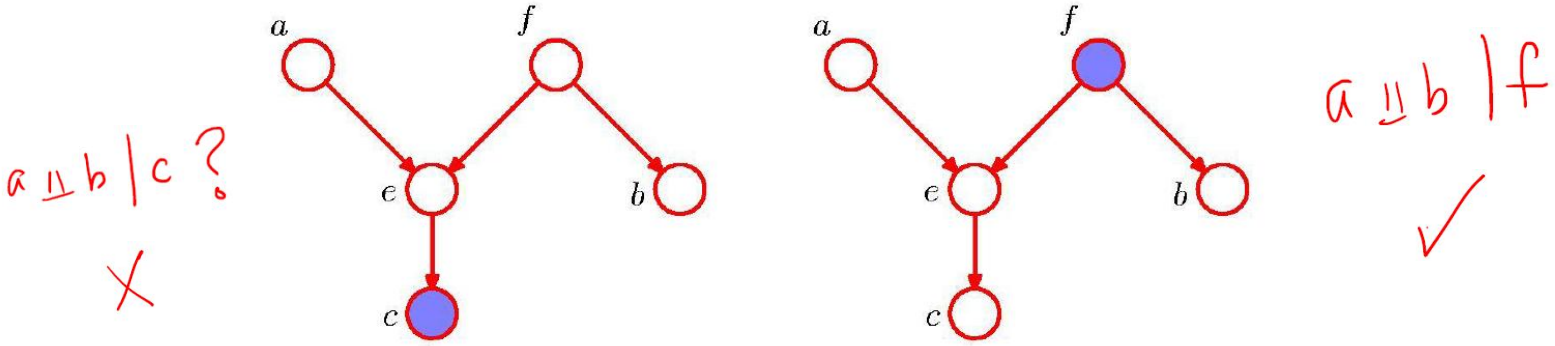
Collider (head-to-head)

独立性判断

- D-分离(Directional separation，D-separation)可用于判断任意两个节点的相关性和独立性。若存在一条路径将这两个节点(直接)连通，则称这两个节点是有向连接(d-connected)的，即这两个节点是相关的；若不存在这样的路径将这两个节点连通，则这两个节点不是有向连接的，则称这两个节点是有向分离的(d-separated)，即这两个节点相互独立。

- 定义：路径p 被限定集Z 阻塞(block)当且仅当：
  - 路径p 含有链结构$A \rightarrow B \rightarrow C$ 或分连结构$A \leftarrow B \rightarrow C$ 且中间节点B 在Z 中，或者
  - 路径p 含有汇连结构$A \rightarrow B \leftarrow C$ 且汇连节点B 及其后代都不在Z 中。

- 定义：若Z 阻塞了节点X 和节点Y 之间的每一条路径，则称给定Z 时，X 和Y 是D-分离，即给定Z 时，X 和Y 条件独立。



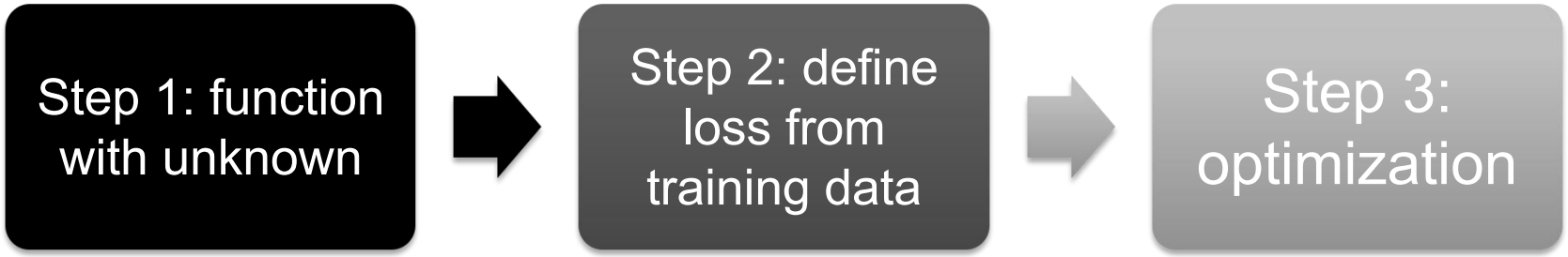
分类

- **Supervised (inductive) learning**
  - Given: training data + desired outputs (labels)
- **Unsupervised learning**
  - Given: training data (without desired outputs)
- **Semi-supervised learning**
  - Given: training data + a few desired outputs
- **Reinforcement learning**
  - Rewards from sequence of actions

监督学习—分类、预测

Training data: $\{(\boldsymbol{x^1}, \hat{y}^1), (\boldsymbol{x^2}, \hat{y}^2), ..., (\boldsymbol{x^N}, \hat{y}^N)\}$

Training:



$y = f_{\boldsymbol{\theta}}(\boldsymbol{x})$

$L(\boldsymbol{\theta})$

$\boldsymbol{\theta}^* = arg\min_{\boldsymbol{\theta}} L$

Testing data: $\{\boldsymbol{x^{N+1}}, \boldsymbol{x^{N+2}}, ..., \boldsymbol{x^{N+M}}\}$

Use $y = f_{\boldsymbol{\theta}^*}(\boldsymbol{x})$ to label the testing data

$\{y^{N+1}, y^{N+2}, ..., y^{N+M}\}$

非监督学习—K-means聚类算法

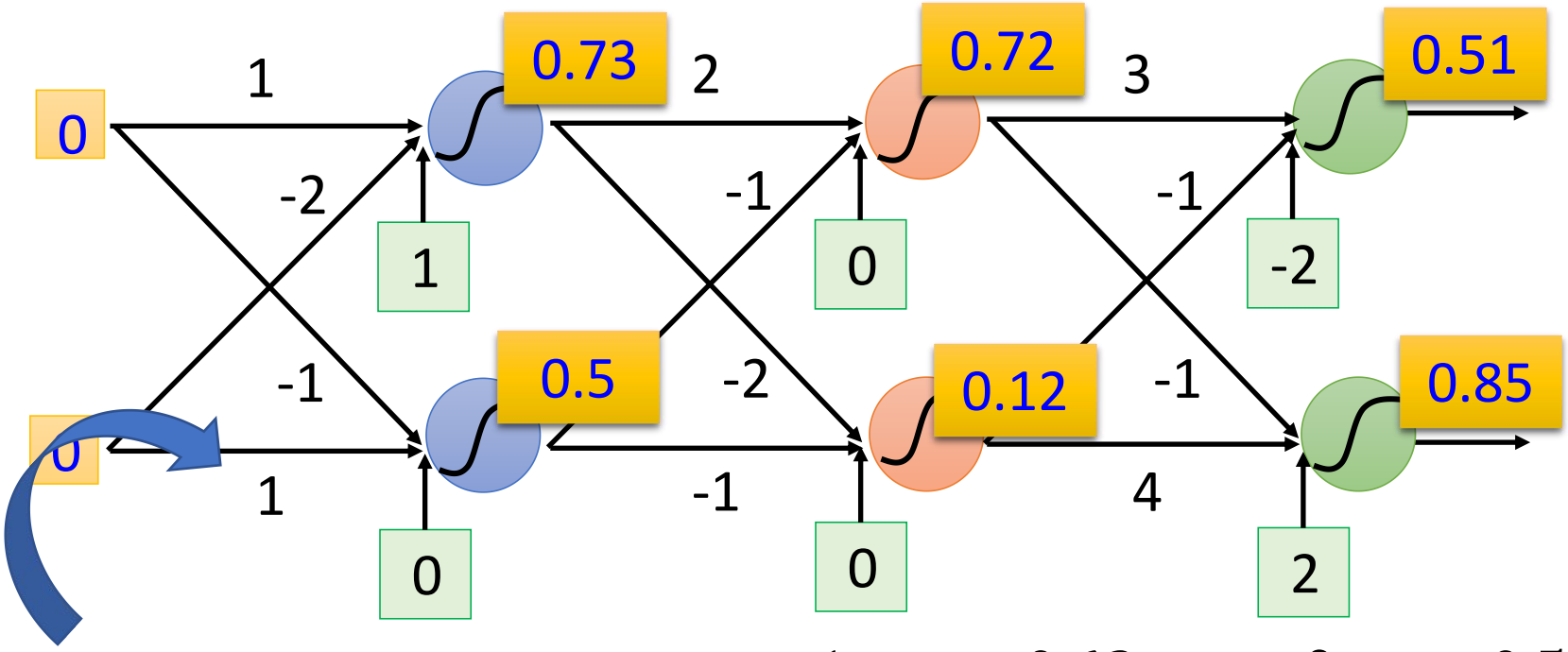
- Partitional clustering approach
- Each cluster is associated with a centroid (center point)
- Each point is assigned to the cluster with the closest centroid
- Number of clusters, K, must be specified
- The basic algorithm is very simple

- 初始点的选择
- 出现空集

1: Select K points as the initial centroids.
2: **repeat**
3: Form K clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: **until** The centroids don't change

https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

- 神经网络



相当于函数功能

$$f\left(\begin{bmatrix}1\\-1\end{bmatrix}\right)=\begin{bmatrix}0.62\\0.83\end{bmatrix}\quad f\left(\begin{bmatrix}0\\0\end{bmatrix}\right)=\begin{bmatrix}0.51\\0.85\end{bmatrix}$$

输入一个向量，输出另一个向量

网络结构相当于定义了函数集空间

- 神经网络

链式法则

Case 1

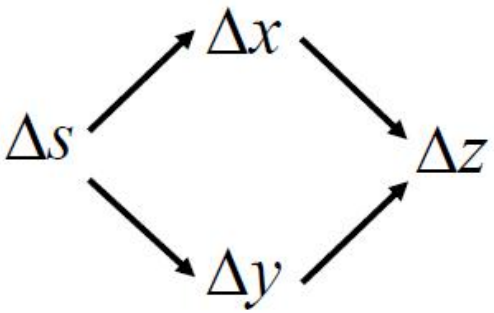
$$y = g(x) \quad z = h(y)$$

$$\Delta x \to \Delta y \to \Delta z$$

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$$

Case 2

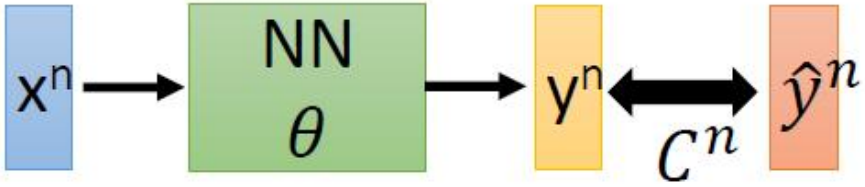
$$x = g(s) \quad y = h(s) \quad z = k(x, y)$$



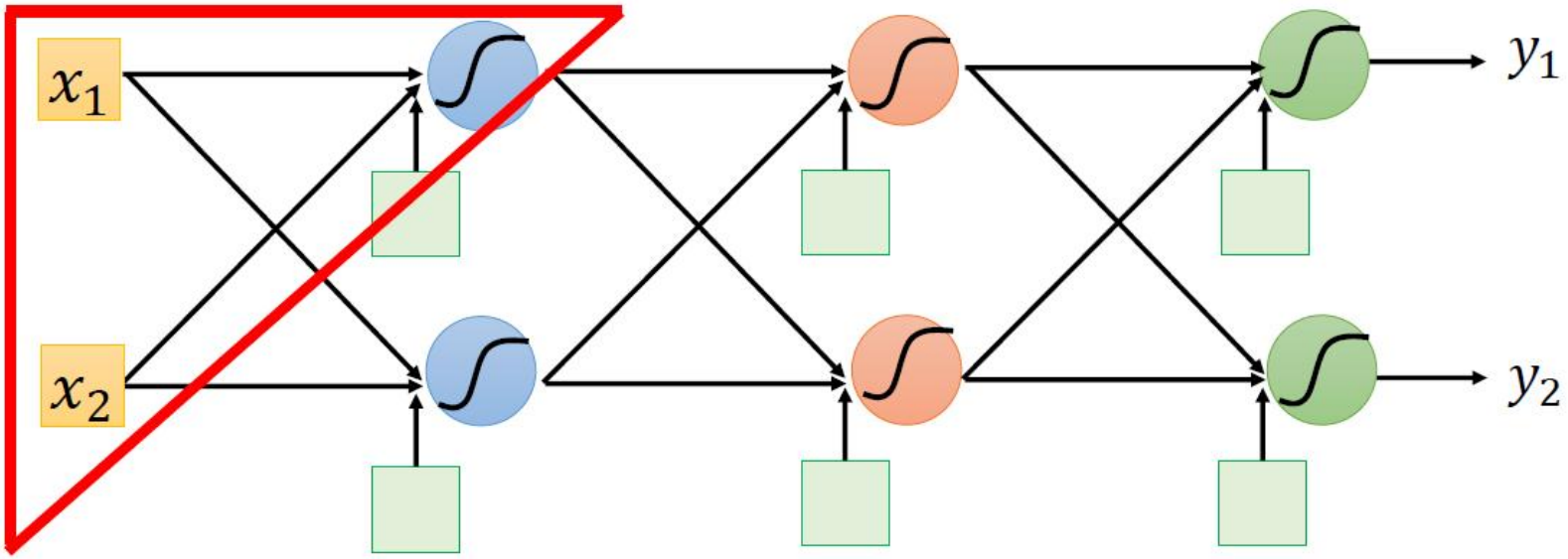
$$\frac{dz}{ds} = \frac{\partial z}{\partial x}\frac{dx}{ds} + \frac{\partial z}{\partial y}\frac{dy}{ds}$$

- 神经网络

BP算法

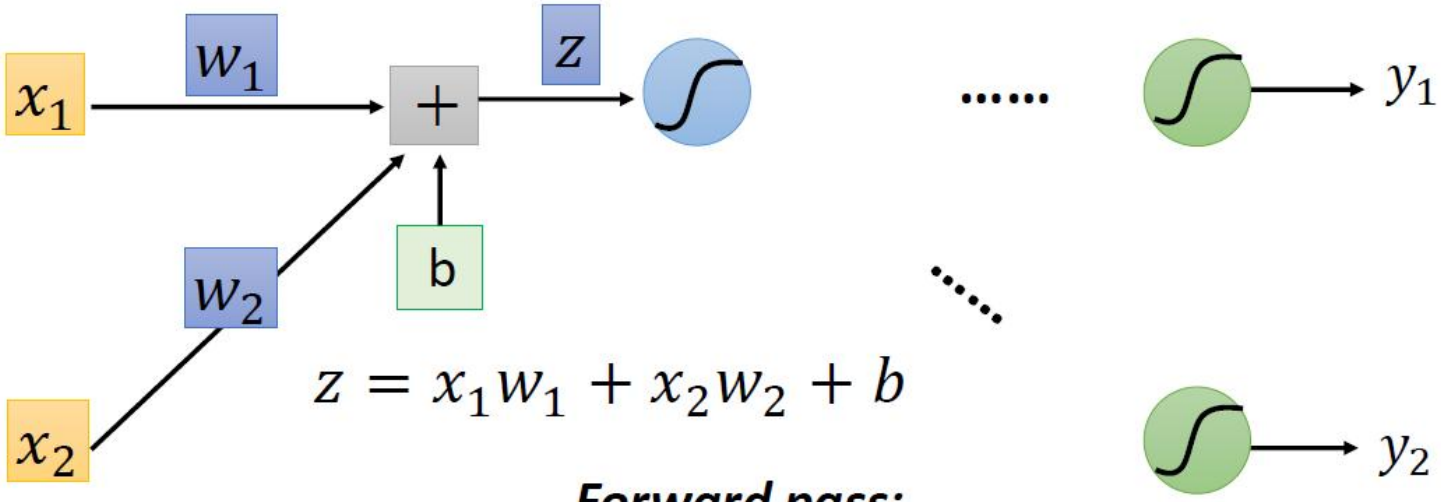


$$L(\theta) = \sum_{n=1}^{N} C^n(\theta) \Rightarrow \frac{\partial L(\theta)}{\partial w} = \sum_{n=1}^{N} \frac{\partial C^n(\theta)}{\partial w}$$



- 神经网络

BP算法



$$z = x_1 w_1 + x_2 w_2 + b$$

Forward pass:

Compute $\partial z / \partial w$ for all parameters

$$\frac{\partial C}{\partial w} = ? \quad \frac{\partial z}{\partial w}\frac{\partial C}{\partial z}$$

(Chain rule)

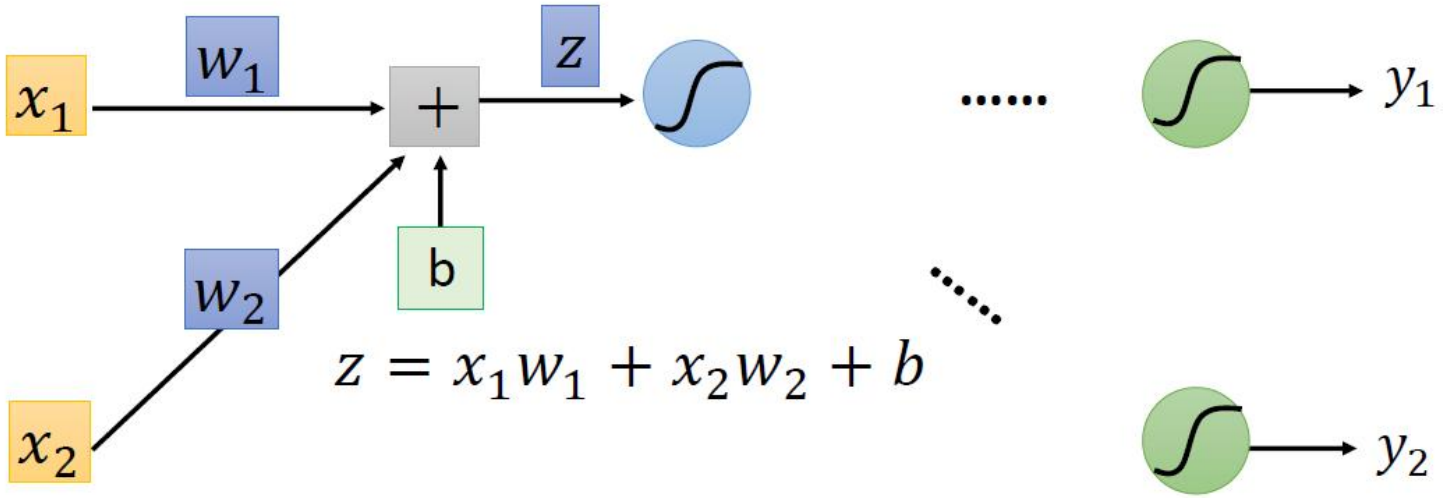
Backward pass:

Compute $\partial C / \partial z$ for all activation function inputs z

- 神经网络

BP算法-前向传递

Compute $\partial z/\partial w$ for all parameters



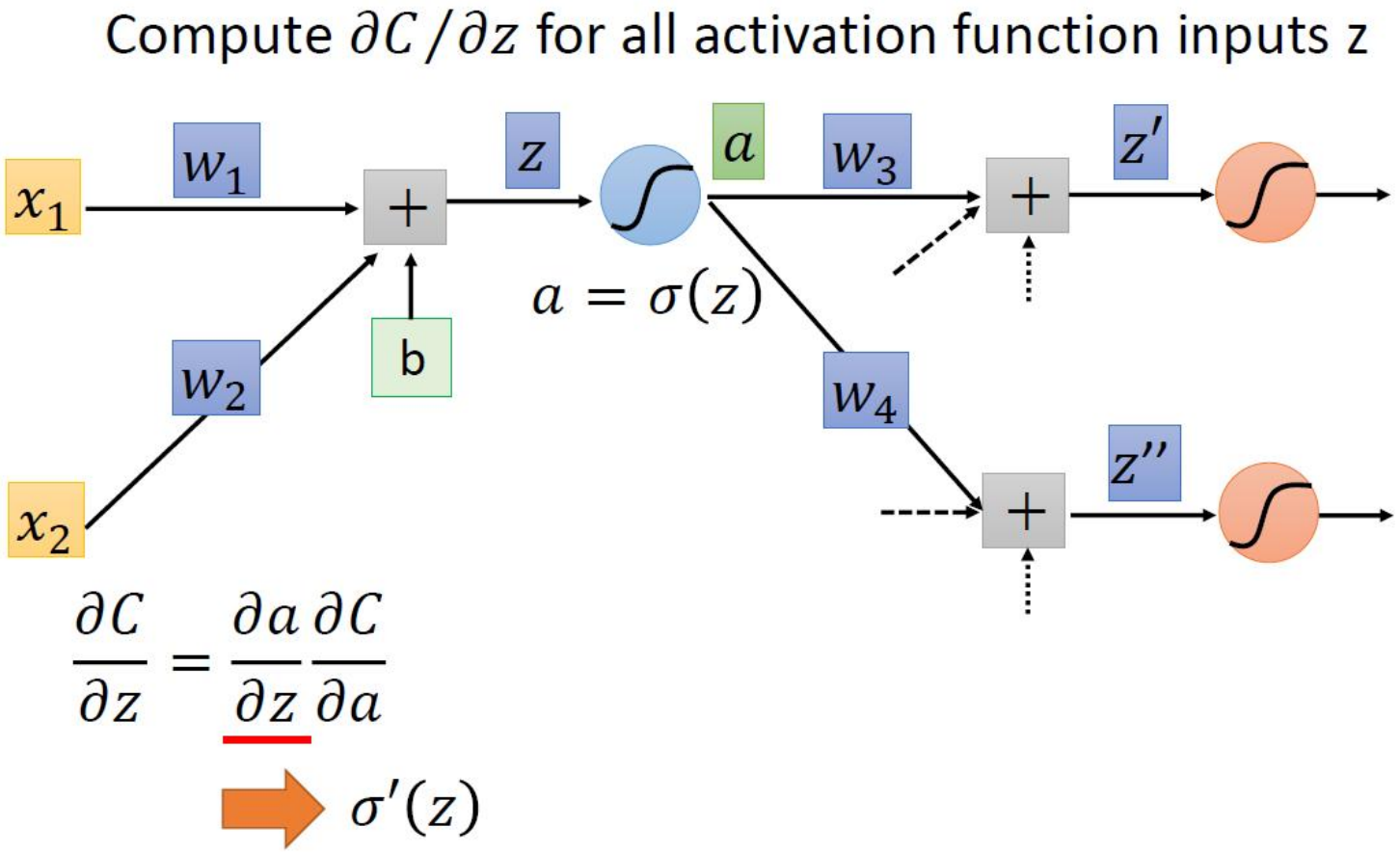
$\partial z/\partial w_1 =?\ x_1$

$\partial z/\partial w_2 =?\ x_2$

The value of the input connected by the weight

- 神经网络

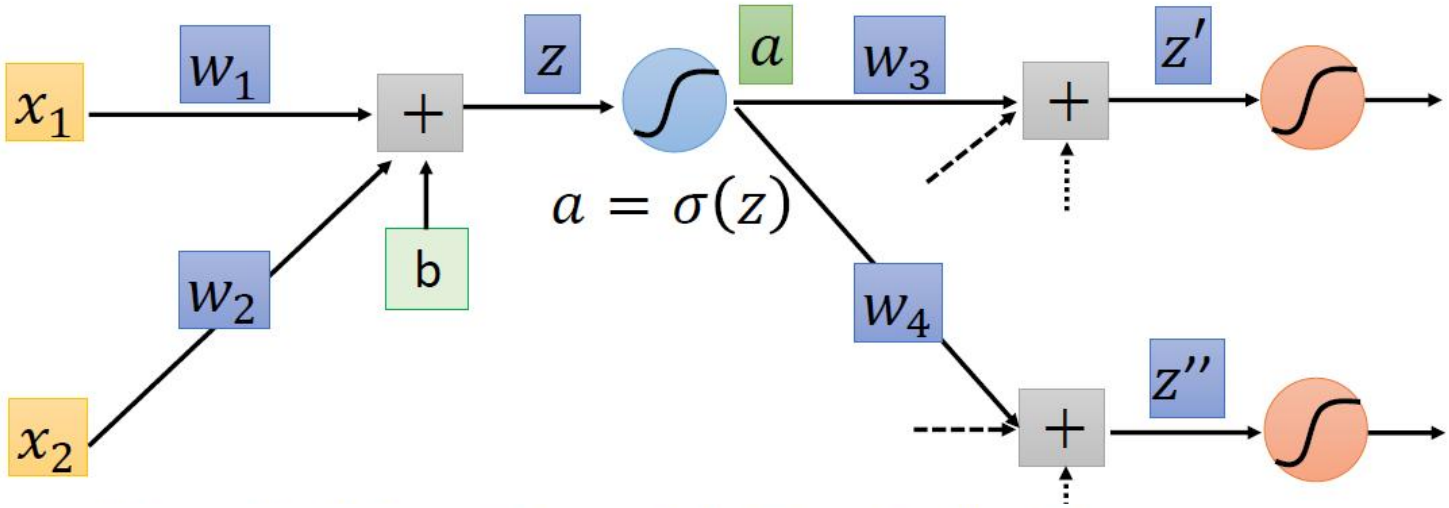
BP算法-反向传递



神经网络

BP算法-反向传递

Compute $\partial C/\partial z$ for all activation function inputs z



$$\frac{\partial C}{\partial z} = \frac{\partial a}{\partial z}\frac{\partial C}{\partial a} \qquad \frac{\partial C}{\partial a} = \frac{\partial z'}{\partial a}\frac{\partial C}{\partial z'} + \frac{\partial z''}{\partial a}\frac{\partial C}{\partial z''} \quad \text{(Chain rule)}$$

$\frac{\partial z'}{\partial a} = w_3$, $\frac{\partial C}{\partial z'}$ = ?, $\frac{\partial z''}{\partial a} = w_4$, $\frac{\partial C}{\partial z''}$ = ?

Assumed it's known

第六章 机器学习

- 深度学习

CNN

Property 1

- 一些特征会比整张图片小很多

Property 2

- 相同特征会出现在不同区域

Property 3

- 像素二次采样不会改变识别的目标信息

Convolution

Max Pooling

Convolution

Max Pooling

可重复多次

Flatten

- 深度学习 CNN

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

需要学习的网络参数

1	-1	-1
-1	1	-1
-1	-1	1

卷积核 1 Matrix

-1	1	-1
-1	1	-1
-1	1	-1

卷积核 2 Matrix

⋮

Property 1 每个卷积核可以识别区域特征(3 x 3).

- 深度学习 CNN

stride(步长)=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

卷积核 1

3 -1

- 深度学习 CNN

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

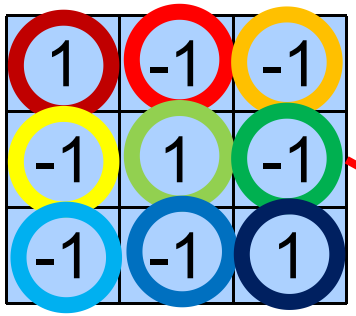
6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

卷积核 1

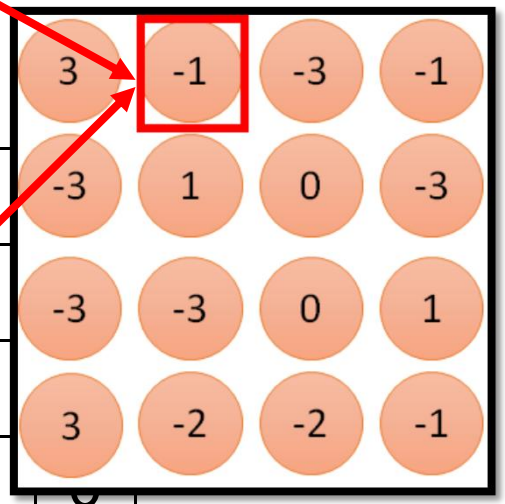
3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

Property 2 相同特征出现在不同区域



CNN

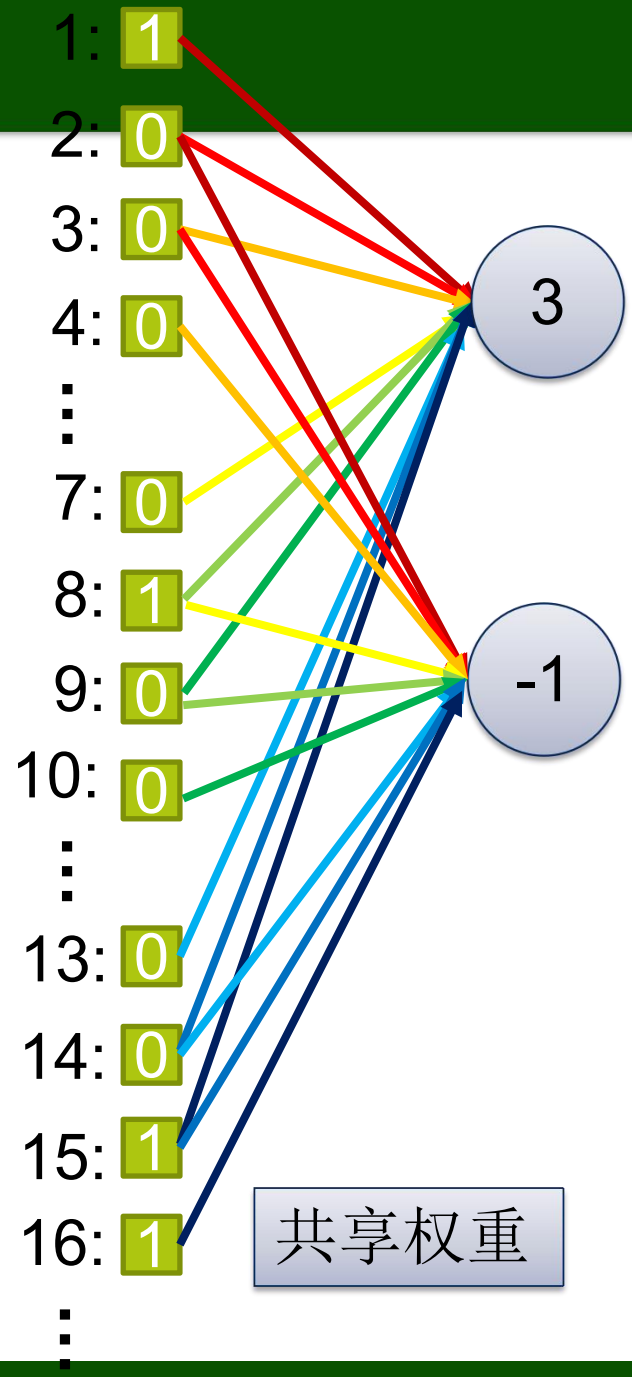
卷积核 1



1	0	0	0	0	
0	1	0	0	1	
0	0	1	1	0	
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

减少更多的参数!



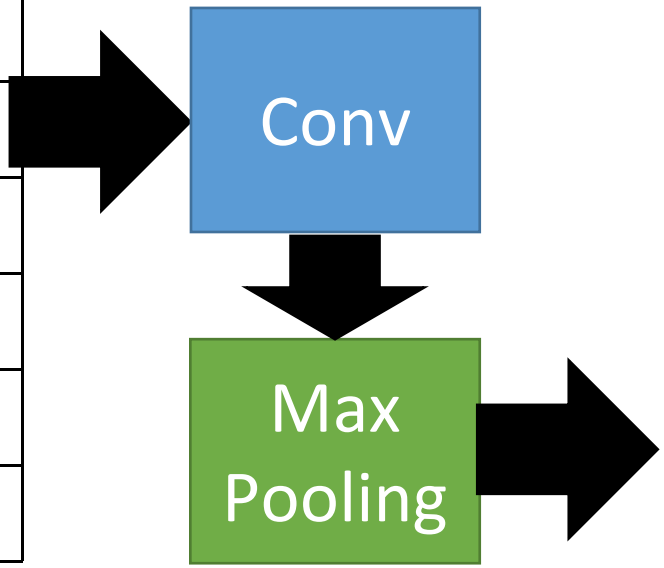
共享权重

- 深度学习 CNN

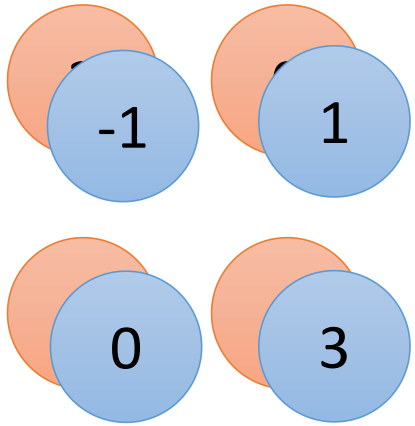
Max Pooling (最大池化)

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image



更少的特征信息



2 x 2 image

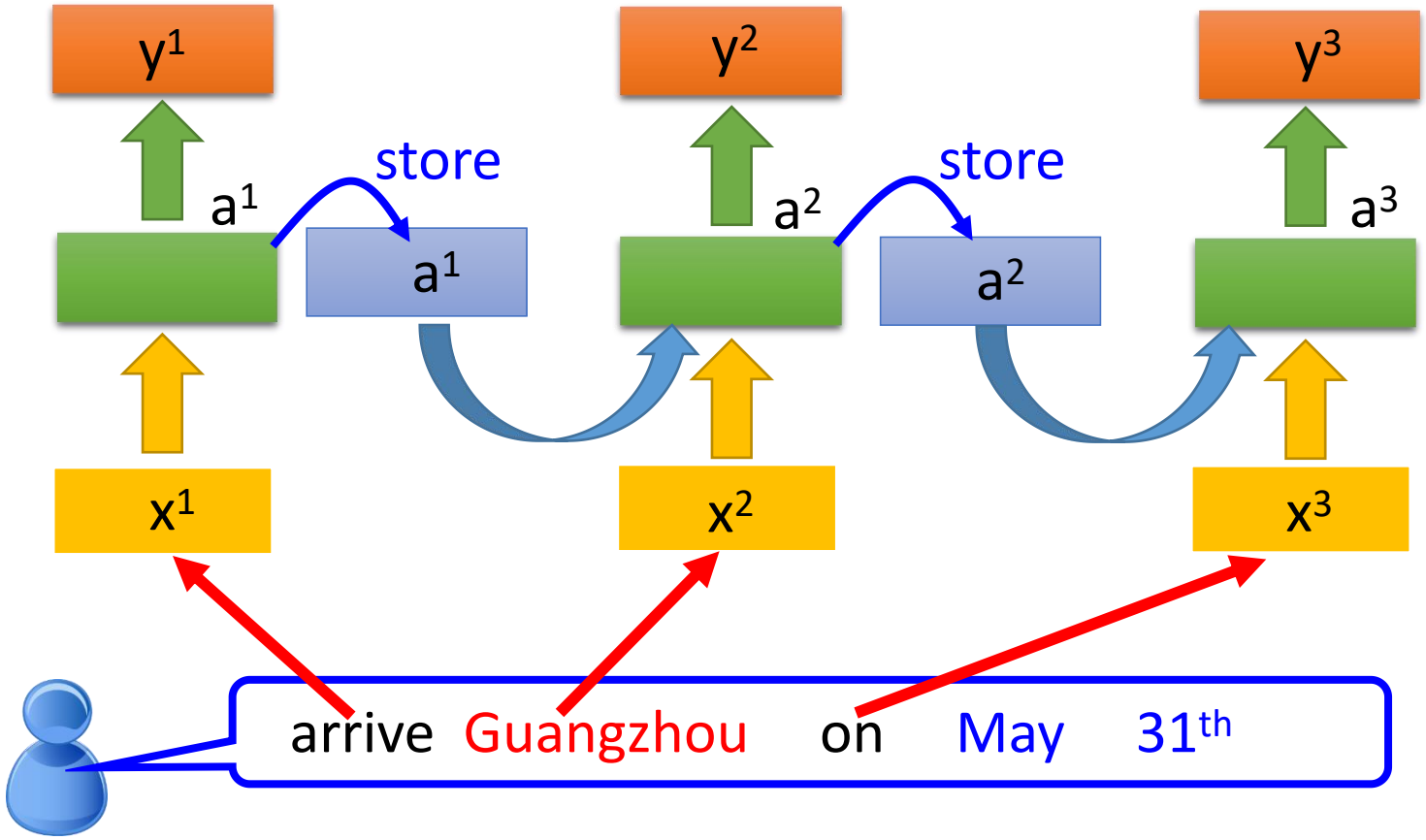
每个卷积核都会生成一个通道(channel)

- 深度学习 RNN

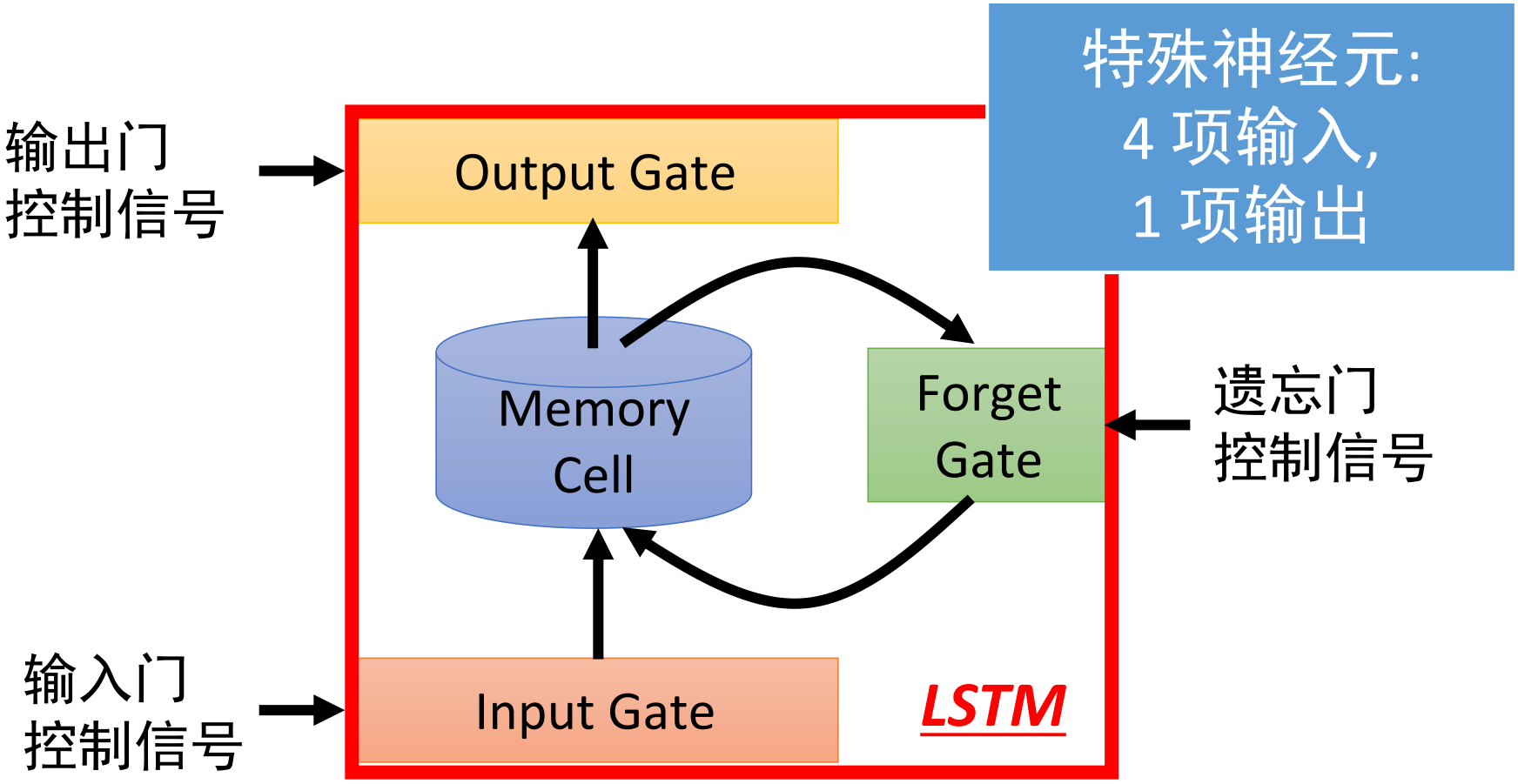
“arrive” 对应每个类别的概率

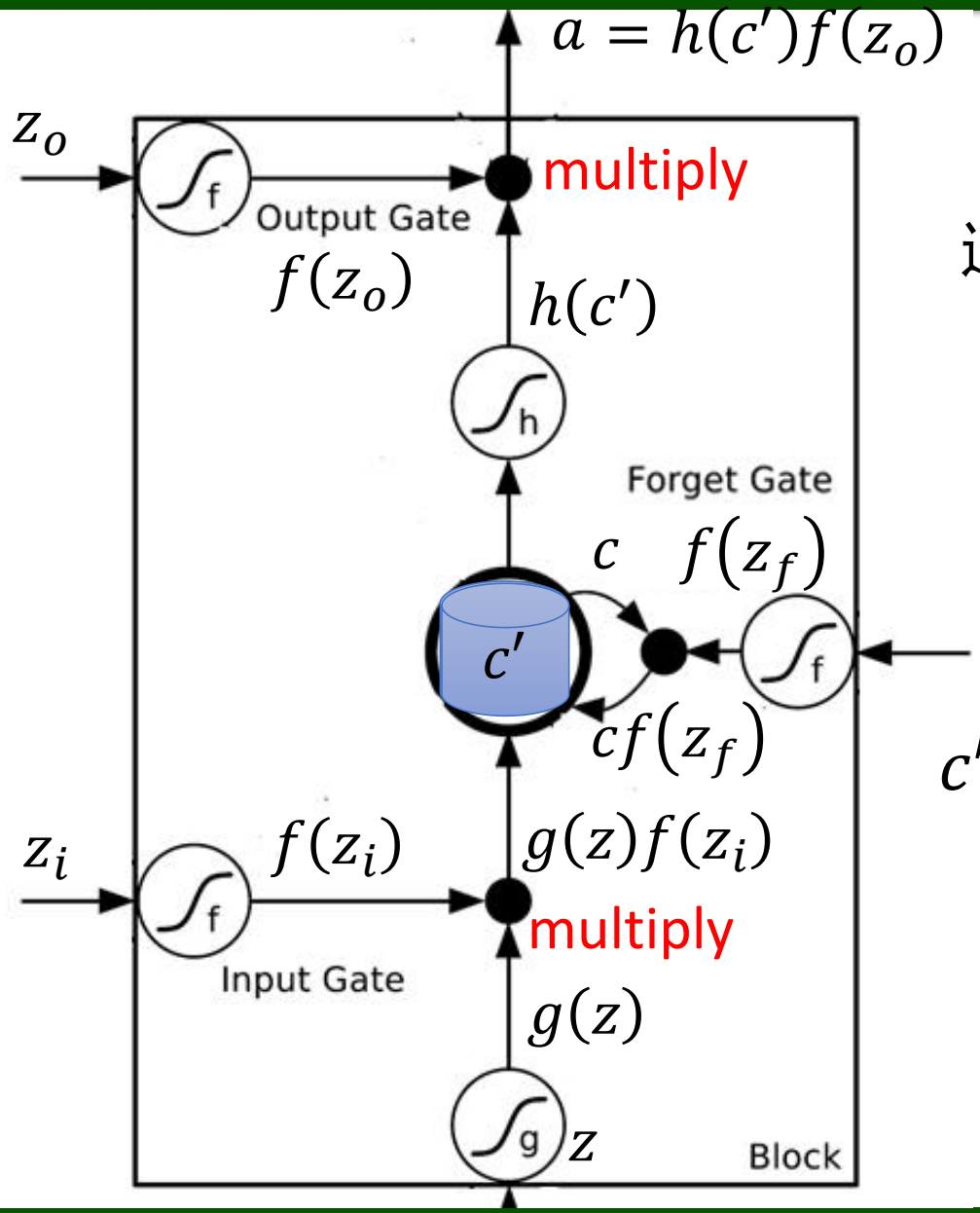
“Guangzhou” 对应每个类别的概率

“on” 对应每个类别的概率



- 深度学习 LSTM


特殊神经元:
4 项输入,
1 项输出
输出门
控制信号
Output Gate
Memory
Cell
Forget
Gate
遗忘门
控制信号
输入门
控制信号
Input Gate
LSTM



通常使用 sigmoid 激活函数

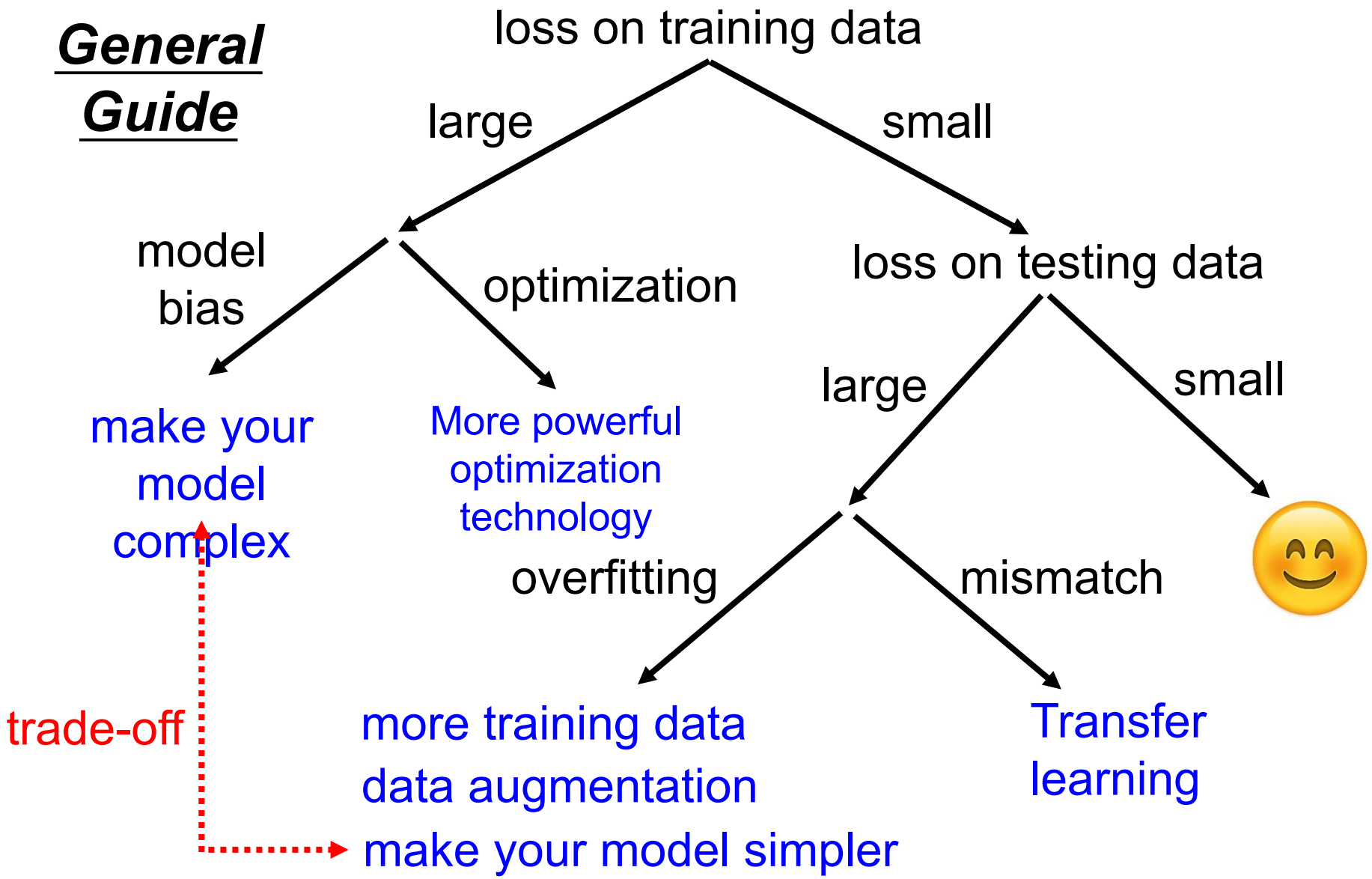
限制在 0 到 1 之间

模拟门的开关

$$c' = g(z)f(z_i) + cf(z_f)$$

第六章 机器学习

- 机器学习研究基础理论问题
  - Framework of ML
  - Small Gradient
  - Optimizer
  - Normalization
- 机器学习研究子问题
  - Attention
  - GAN
  - BERT
  - VAE
  - Attack
  - …..


General Guide
loss on training data
large
small
model bias
optimization
loss on testing data
make your model complex
More powerful optimization technology
large
small
overfitting
mismatch
trade-off
more training data
data augmentation
make your model simpler
Transfer learning