人工智能：

不属于人工智能：自动控制，计算科学

图灵机的定义：是一个数学模型，是一个七元组

人工智能三大流派：符号主义，连接主义，行为主义

符号主义：把现实的物，映射到代表它的符号，在符号上完成所有的推理、计算等等。图灵测试就是这个思路

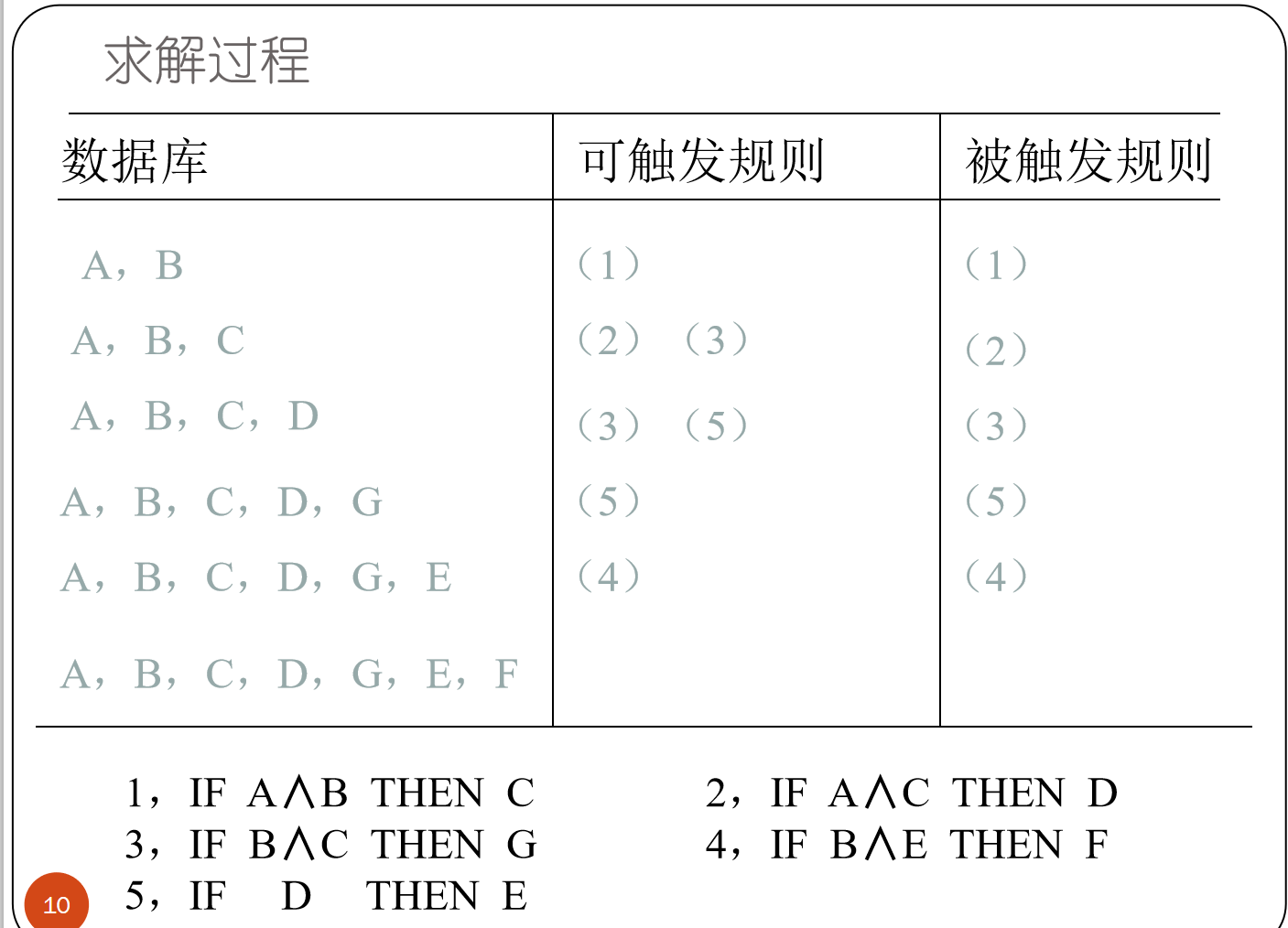
超算主要是浮点运算能力很强，普通计算机计算浮点运算很慢

GPU无法单独工作，必须由CPU进行控制调用才能工作。CPU可单独作用，处理复杂的逻辑运算和不同的数据类型，但当需要大量的处理类型统一的数据时，则可调用GPU进行并行计算。

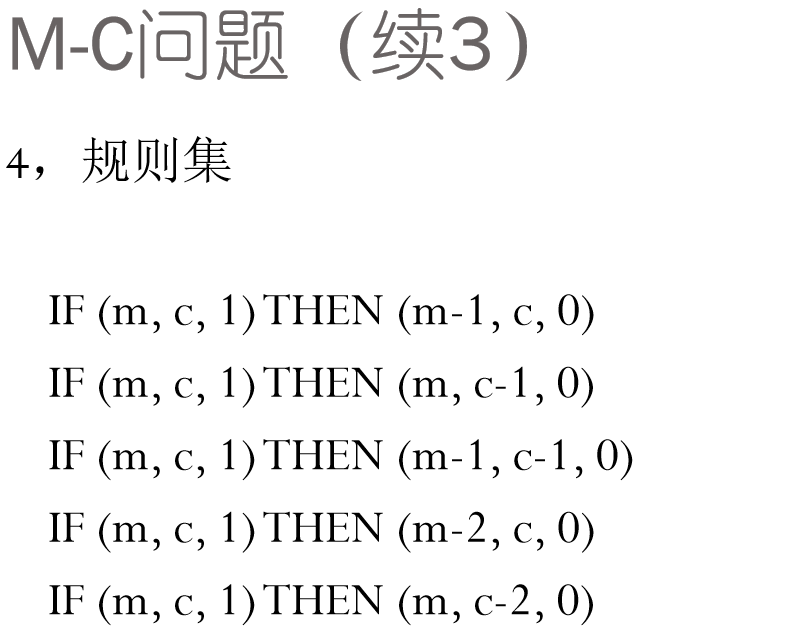
“天河一号”第一次采用GPU承担主要计算量的异构模式

大数据（big data），指无法在一定时间范围内用常规软件工具进行捕捉、管理和处理的数据集合，是需要新处理模式才能具有更强的决策力、洞察发现力和流程优化能力的海量、高增长率和多样化的信息资产。

大数据的5V特点（IBM提出）：Volume（大量）、Velocity（高速）、Variety（多样）、Value（低价值密度）、Veracity（真实性）。



有点像规约，第一个A和B then C，然后把C入栈，然后把和C关联（2）&（3）的入栈，然后二者随便选一个，选了（2）可以规约出D的把D入栈。



产生式特点：

数据驱动，知识的无序性，控制系统与问题无关，数据、知识和控制相互独立。

盲目算法：回溯，宽度优先，深度优先

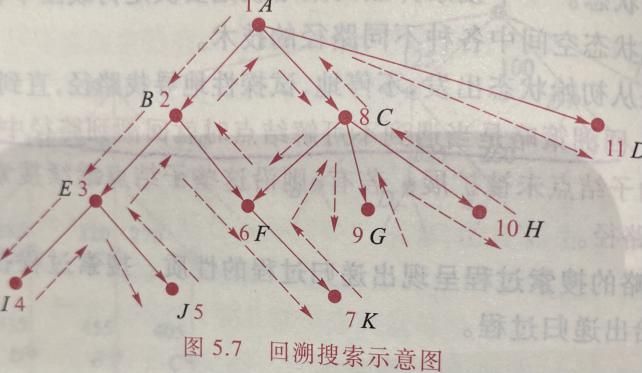
回溯算法：遇到不行的解就掉头。深度优先，但可能会死循环，可以限定深度或者记录路径

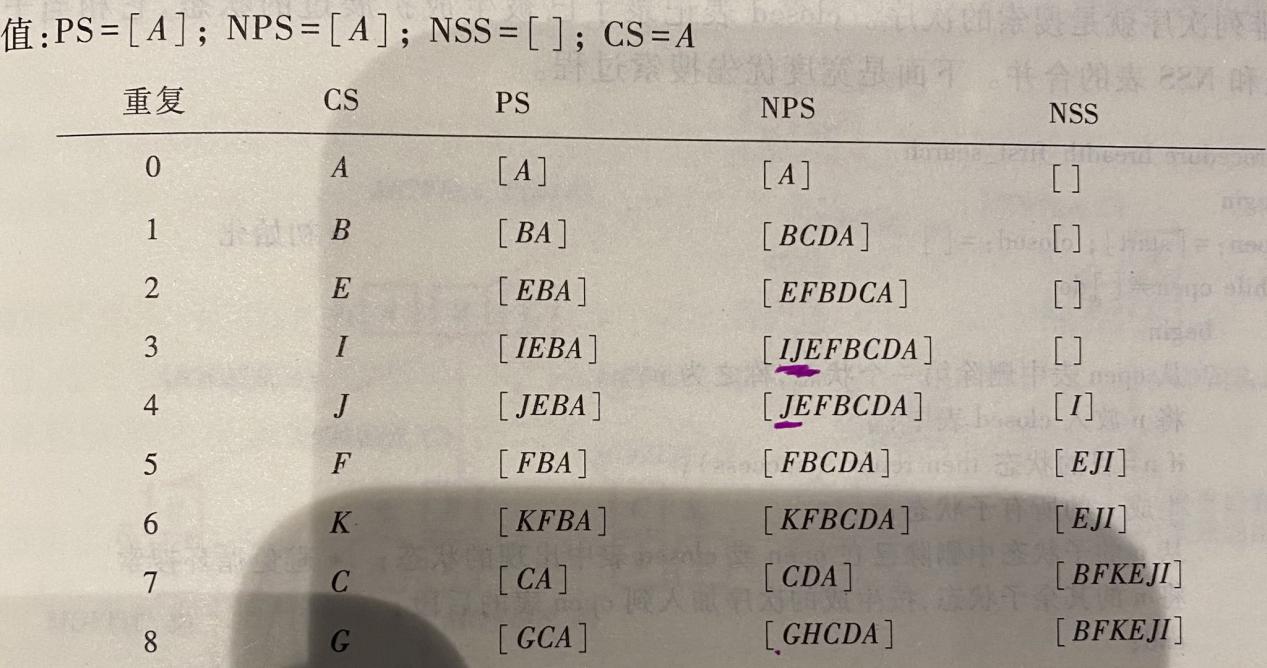
回溯搜索：只保存从当前路径到下一节点的值；

PS （访问过的点）NPS（将要访问的点） NSS（不可访问的点） cs是正在访问的点

Ps = [start] Nps = [start]

NSS = [0]





下一个必考：

图搜索：保留所有路径

八数码是按照空格移动的方向来生成树的，不一定都是第一个根生成四个节点的，若是边上的空格可能会只有三个或者两个方向。

宽度优先：

Open表相当于NPS表，表示未被搜索的状态

Close表相当于PS和NSS的合并表，记录了已经生成并扩展的状态

While　open！＝空

从open删除第一个值

放入close中

如果等于目标状态的话就return　succeed

Else生成这个节点所有的子状态

删除已经在open和close的节点，然后剩下的加入open

End

Open是先进先出的队列

但是宽度优先搜索太费空间，因为要把一层都加入OPEN表

但是总能找到最优的解

深度优先：

While　open！＝空

从open删除第一个值

放入close中

如果等于目标状态的话就return　succeed

If 节点的深度<d就继续；

Else生成这个节点所有的子状态

删除已经在open和close的节点，然后剩下的加入open

End

深度优先的open表是栈的结构先进后出

并且不一定可以找到最优的解

启发式搜索：利用知识来引导搜索，达到减少搜索范围，降低问题复杂度的目的。

启发式搜索的强度：

强：降低搜索工作量，但可能导致找不到最优解

弱：一般导致工作量加大，极限情况下变为盲目搜索，但可能可以找到最优解

启发式搜索：

可以利用启发式搜索的剪枝来减少空间的大小

可以利用搜索消除复杂的搜索过程

启发式搜索的方法的种类：陈述性（描述更精炼具体），过程性（操作算子少而精），控制性（控制策略）

估价函数（从初始点经过n节点到终点的路径最小的代价估计值）：

F(n) = g(n) + h(n)

G(n)是实际从初始点到n节点的代价

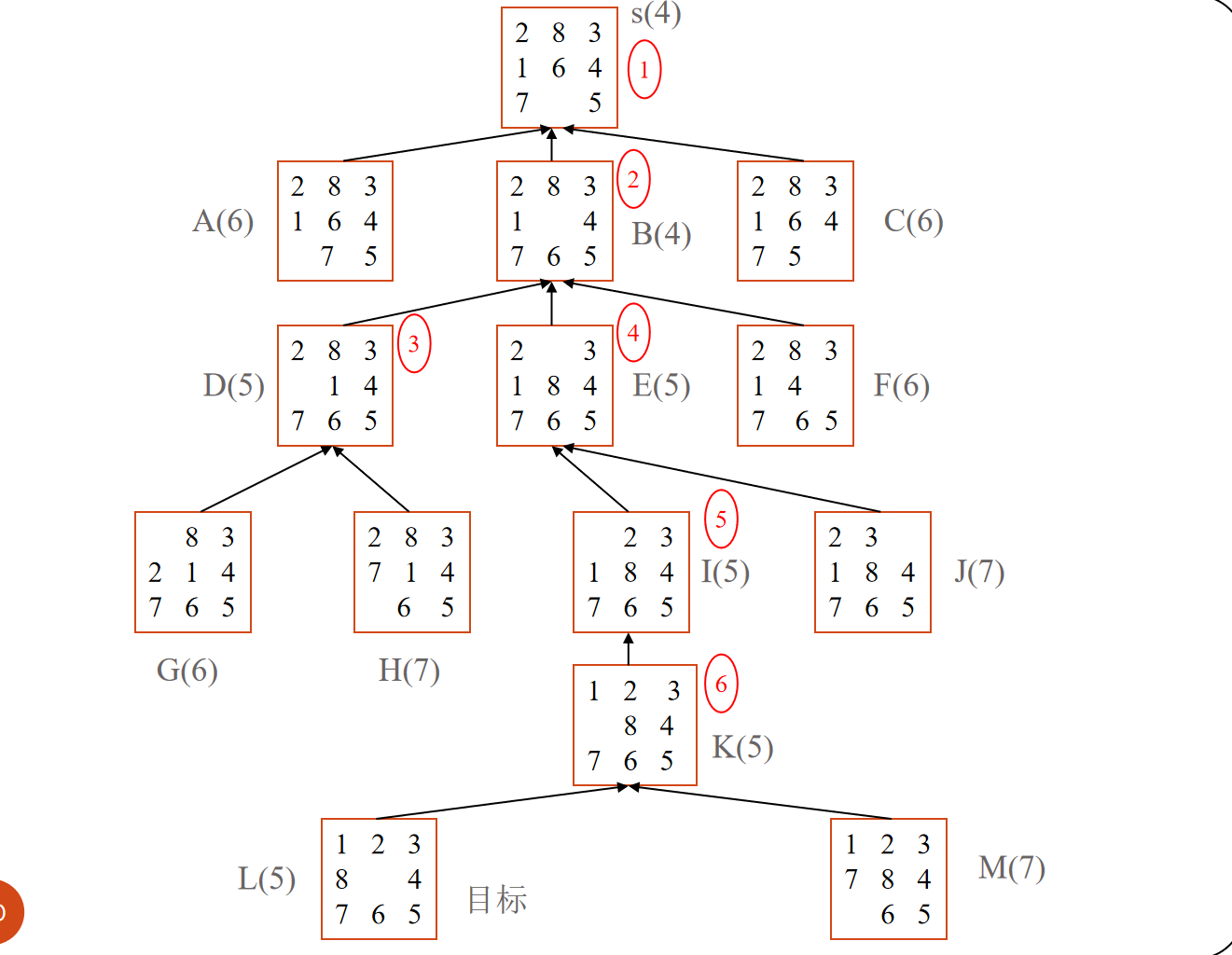
H(n)是从n到终点的估计代价

只希望找到目的节点而不关心代价，就忽略g(n)

当h(n)>>g(n)的时候，有利于提高效率的搜索，但影响效率的完备性

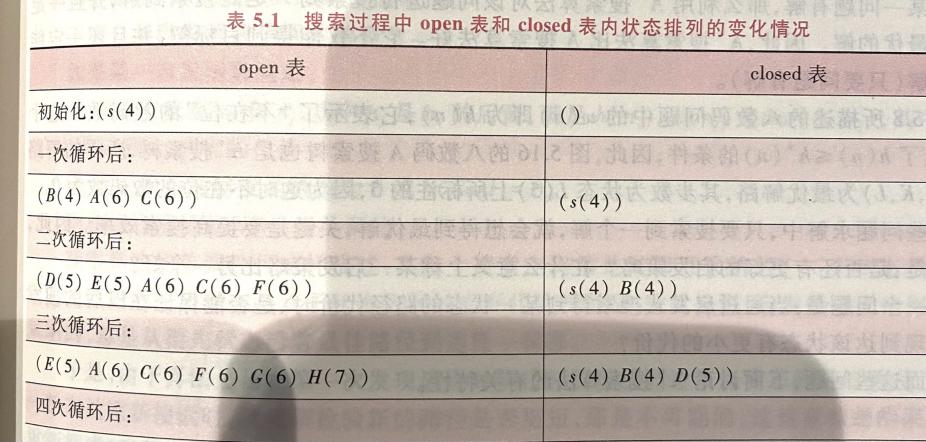
A搜索算法：

G(n)代表状态的深度，h(n)代表不在位数码的个数。

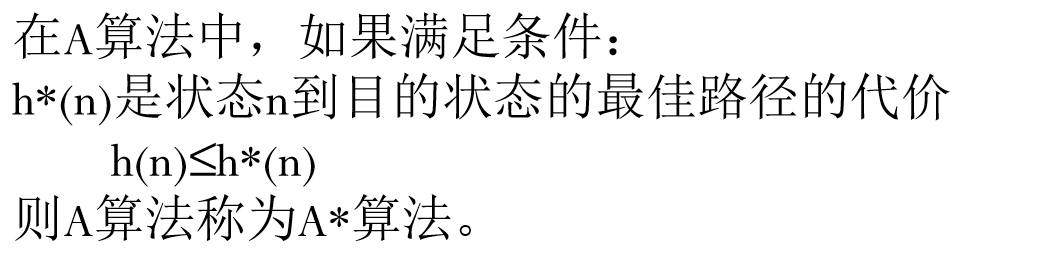


最后close的存储的节点是解的路径

删除open的第一个节点并加到close中，把他的子节点的状态按照f(n)的排序加进来



A\*算法：



A\*算法比A算法好，因为A\*算法一定能找到最优的解，并且一定是只要问题有解

A\*算法的特性：

可采纳性，单调性，信息性

可采纳性：

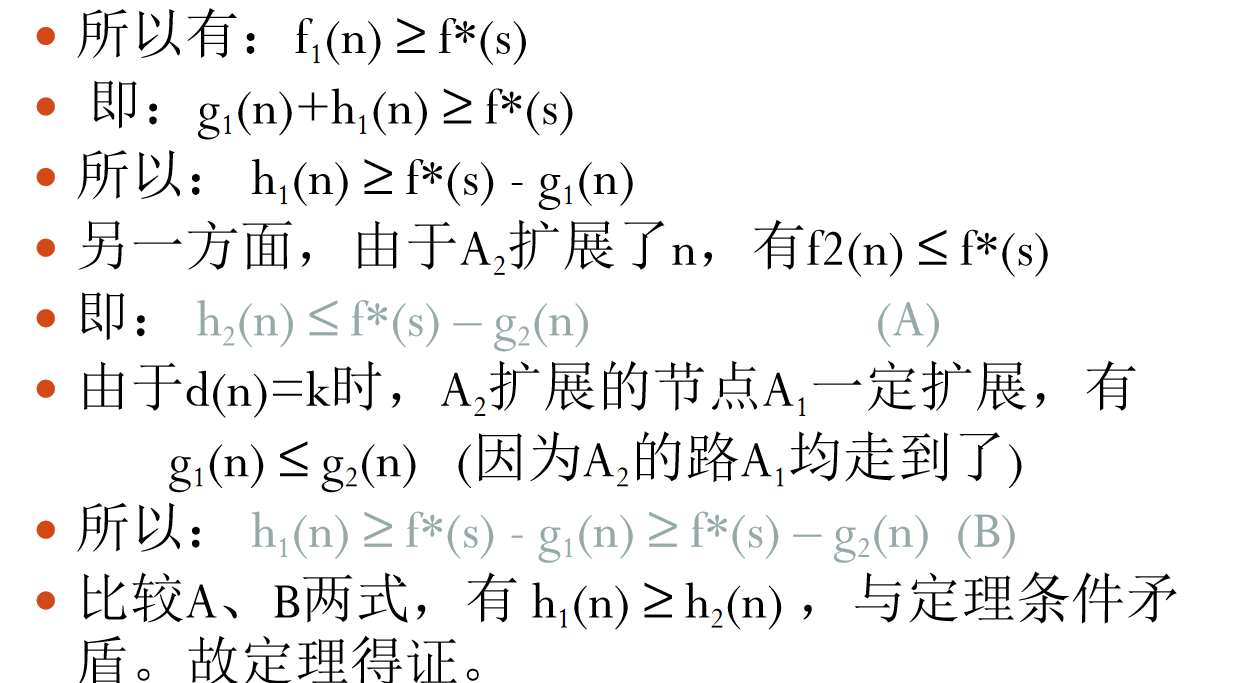
若存在从初始节点s到目标节点t有路径，则A\*必能找到最佳解结束。

信息性：

如果h2(n) > h1(n) (目标节点除外)，则A1扩展的节点数 ≥ A2扩展的节点数

h2的信息比h1的多

证明：用归纳的反证法



推理：

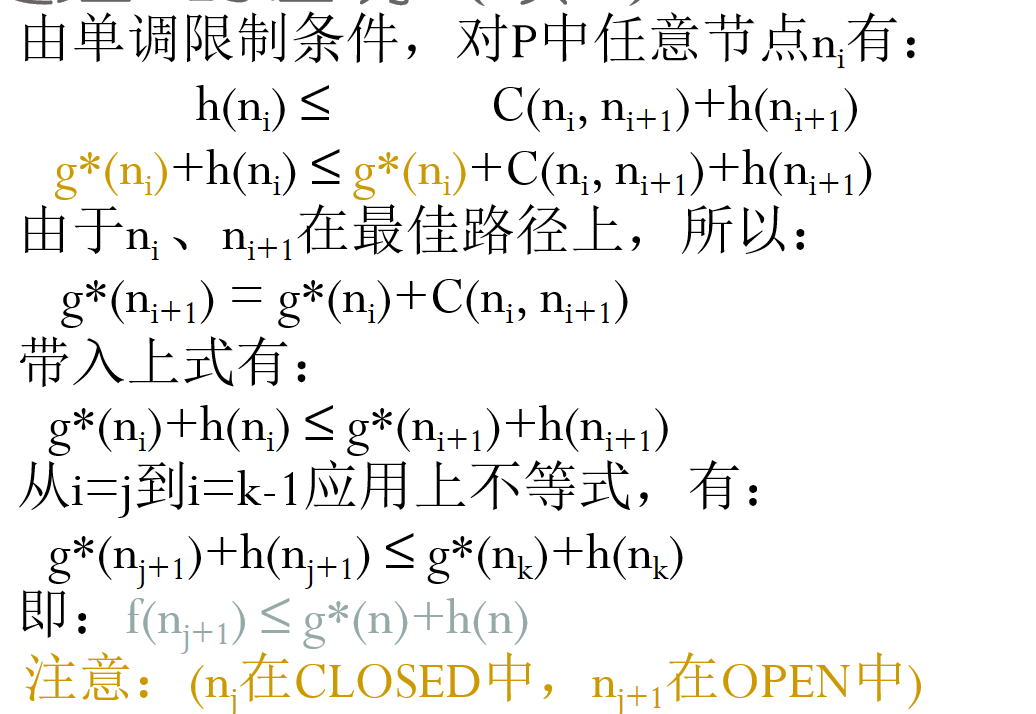
A\*选作扩展的任一节点n，有f(n)≤f\*(s)

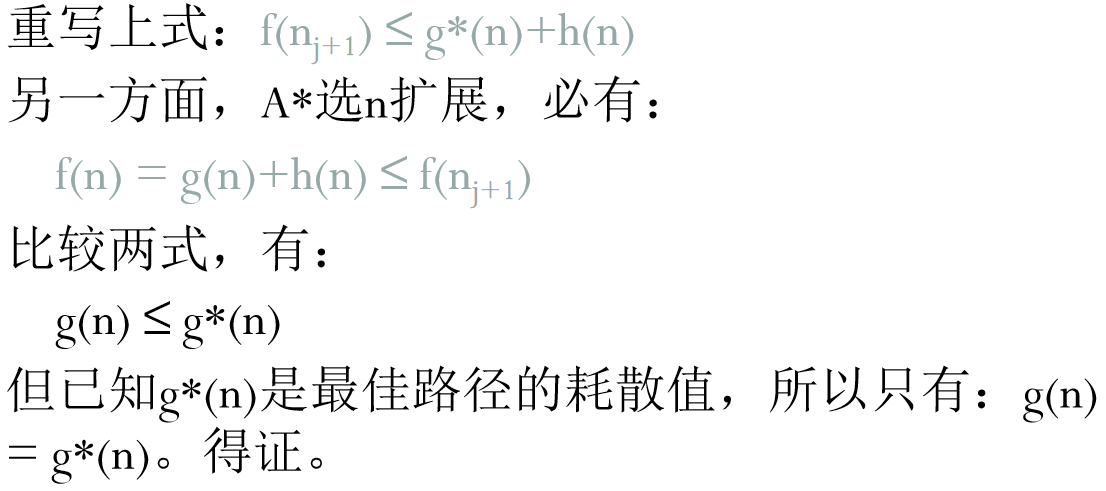
单调性：

启发策略无论是何处都是可采纳的，总是沿着祖先沿着最佳的路线到达另一状态。当A\*选n扩展时，有g(n)=g\*(n)。

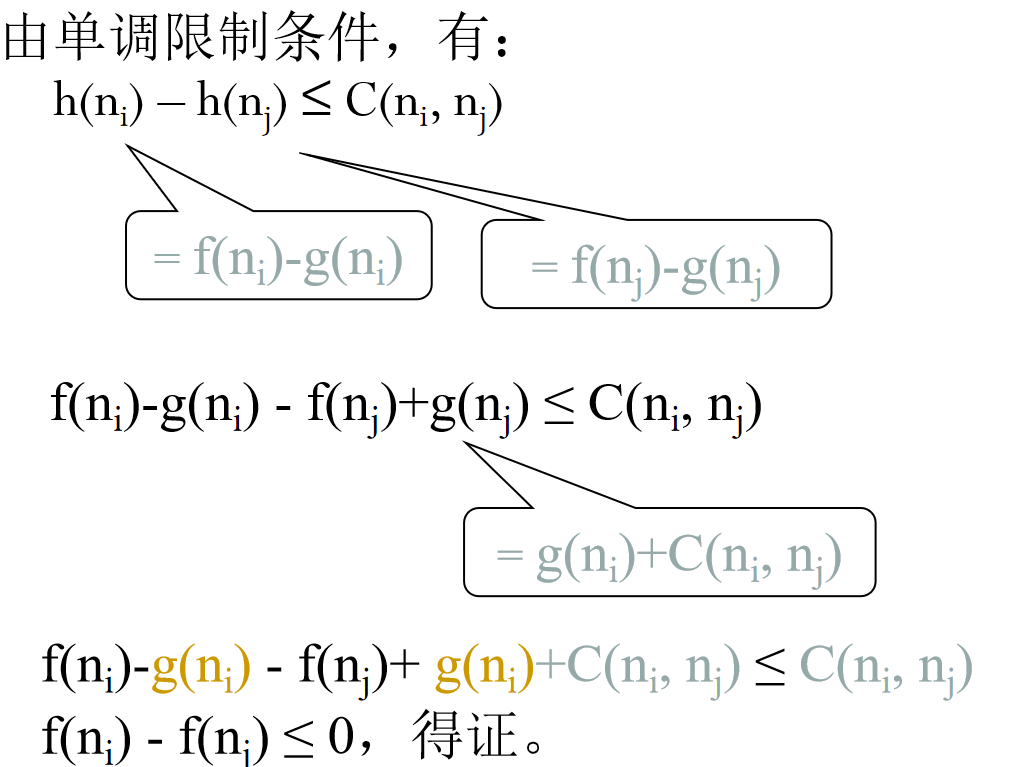
若h(n)是单调的，则由A\*所扩展的节点序列其f值是非递减的。即f(ni) ≤ f(nj)。

证明：

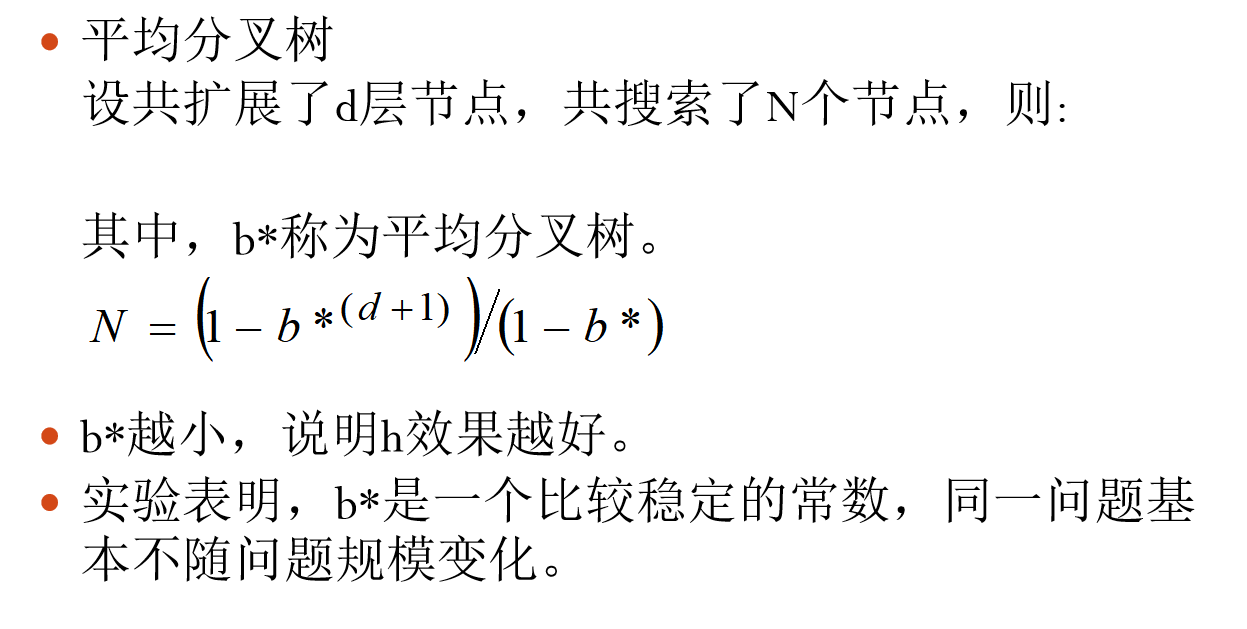


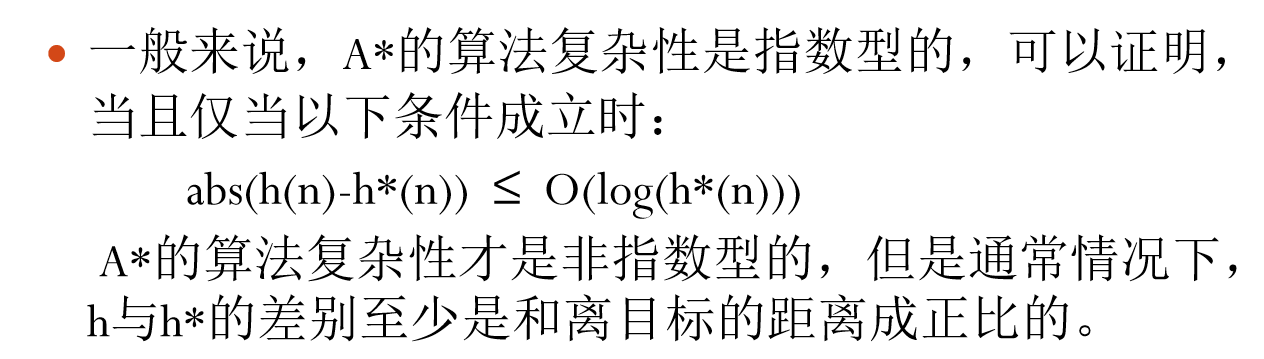


第二行的证明：



一个证明h效果的式子

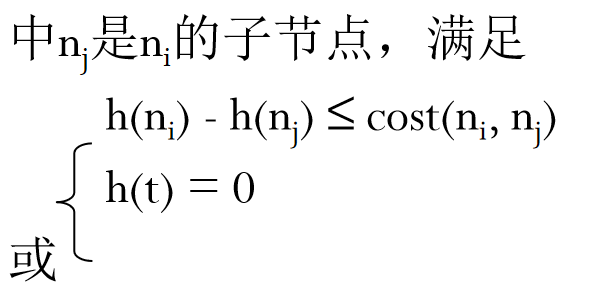




A算法与A\*算法的比较：

因A算法第6步对ml类节点可能要重新放回到OPEN表中，因此可能会导致多次重复扩展同一个节点，导致搜索效率下降。

但是，单调性可以对这个加以限制，即让每一步都是最优的即：当A\*选n扩展时，有g(n)=g\*(n)。

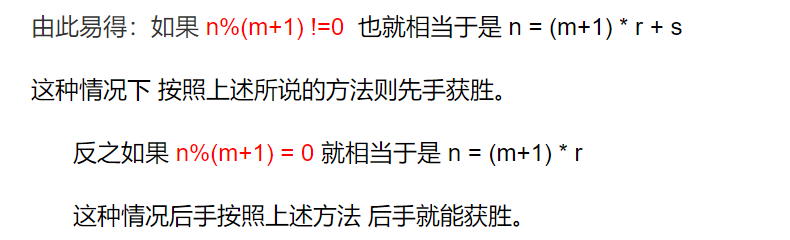


博弈问题：

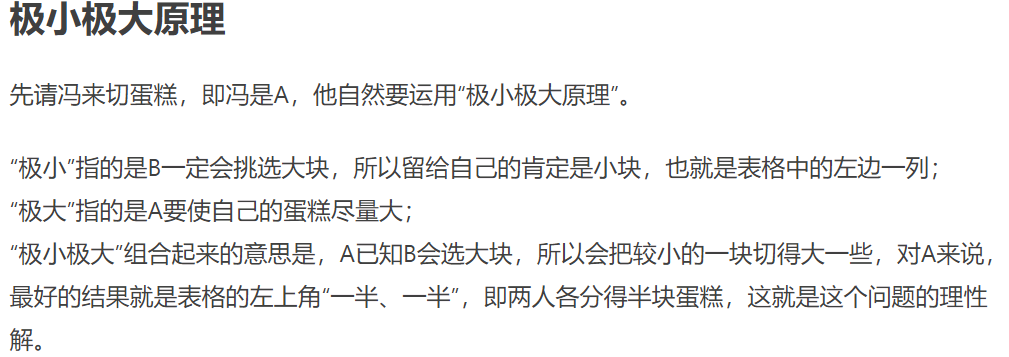
博弈是有限的。即无论两人怎样决策，都会在有限步后决出胜负。

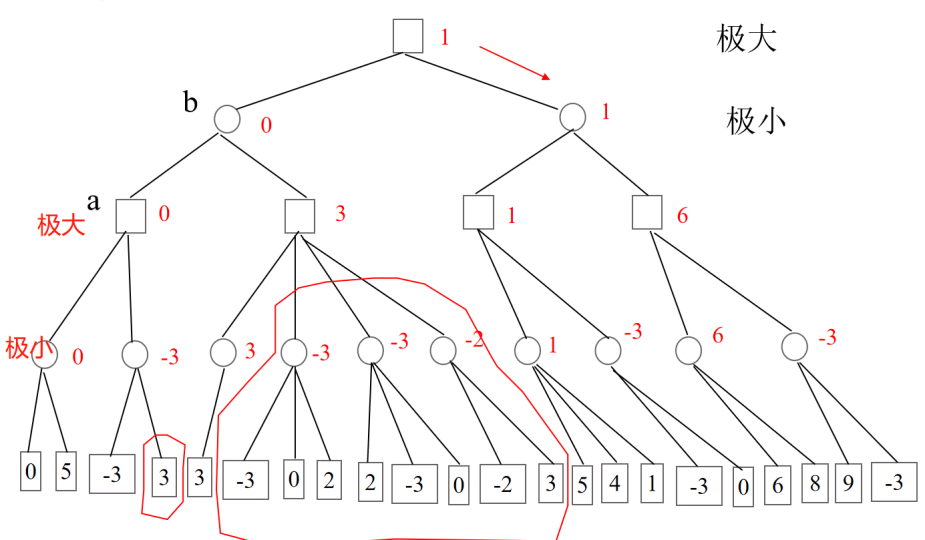
巴什博弈：一共n个，每个人可以拿m个

那么会有下面的结果：



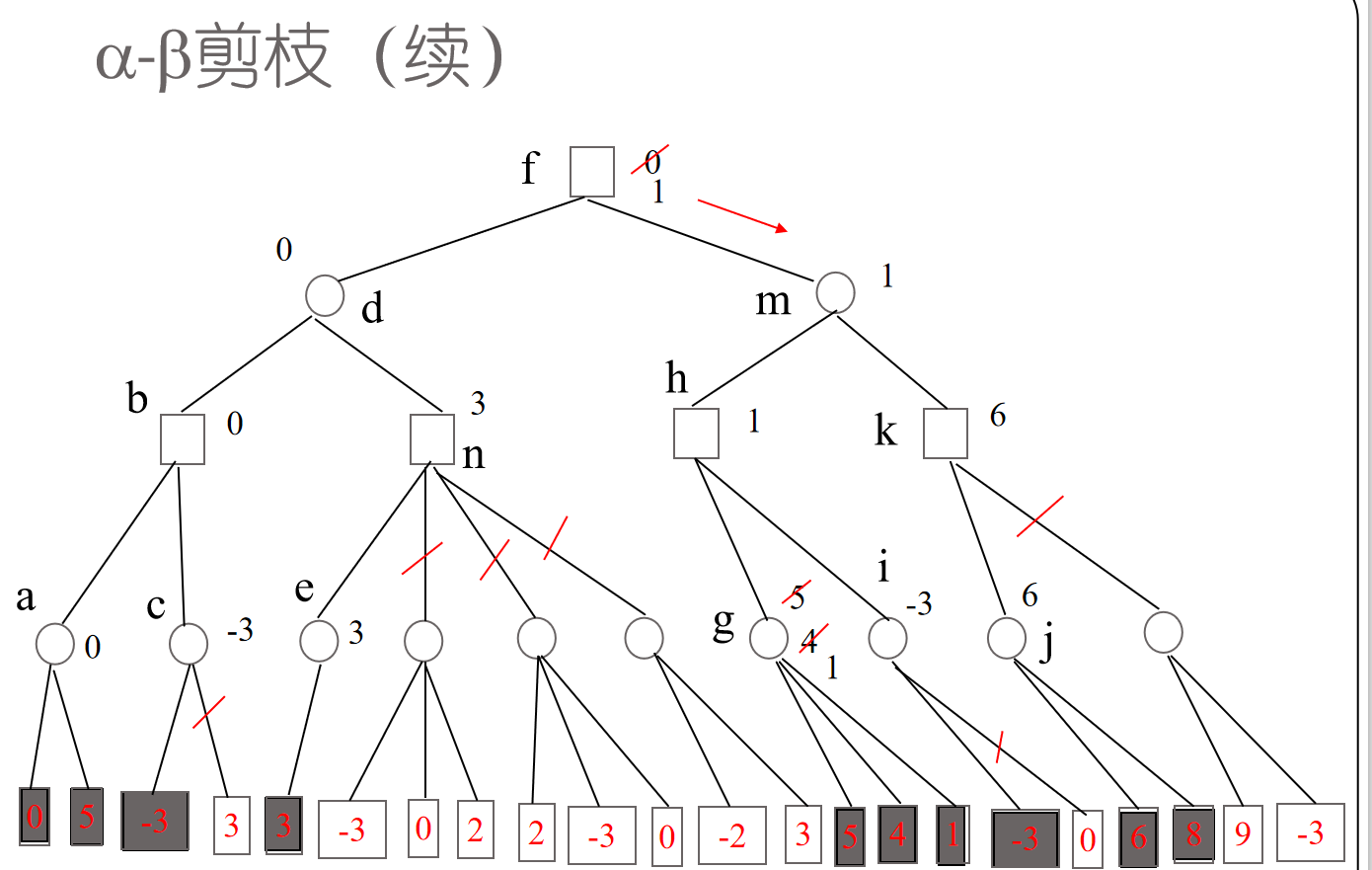
极小极大：





α-β剪枝算法：

因为方块代表我方，所以方块肯定取得是两个圆圈中的最大值，为了让对方失败，圆圈肯定要方块的最小值，如果圆圈中已经有个值不被取，那么别圆圈的子方块的值再被圆圈取也只能小于圆圈，但是方块肯定是要取圆圈的最大值，因此，圆圈就不用再去遍历属于自己的子方块了。有此引发剪枝操作。



剪枝、搜索策略很难改变算法的复杂性

姜桂飞：算法博弈论

主导战略：

•对于玩家i，如果玩x，策略x支配策略y

总比不管其他人怎样都玩y好

但是你不应该严格的采取这个策略

考虑到回报，她也会选择策略“α”。

•主导战略均衡：（α，α）效率低下，囚徒困镜

五个结论：

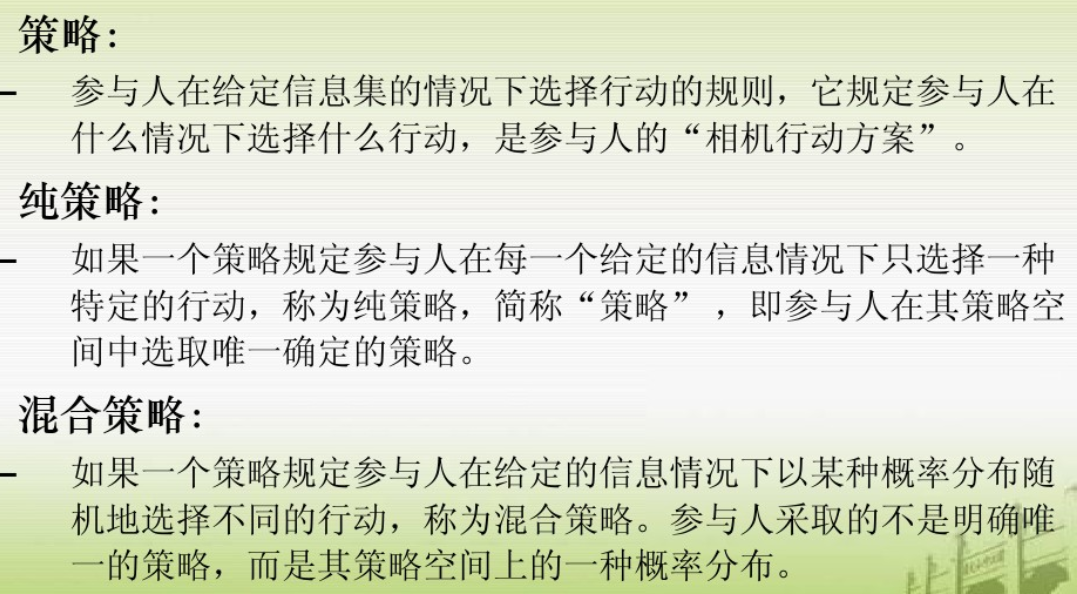
选择严格优势策略（无论对方选择什么，自己的收益都会更高）

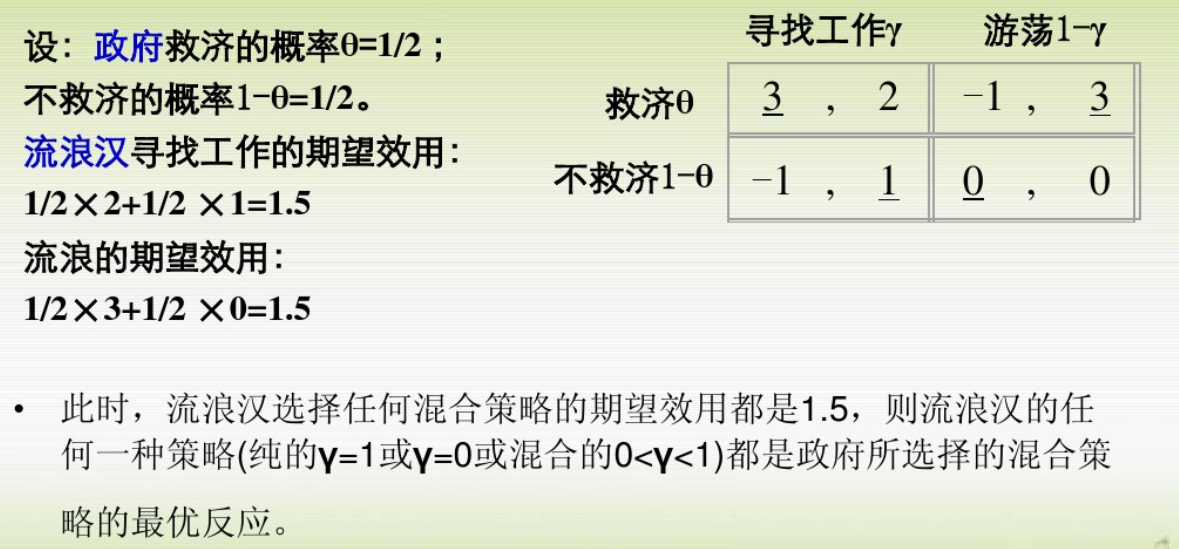
理性选择导致次优结果

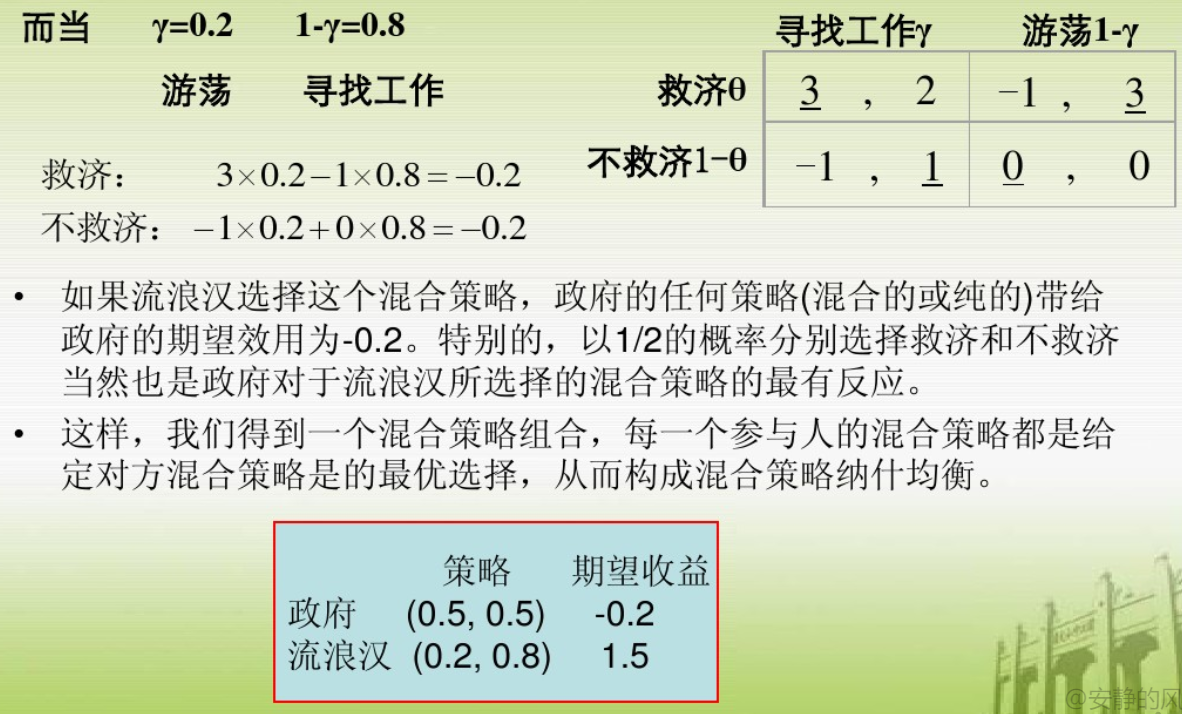
Payoffs matter.苟欲求之，必先知之

策略决策的核心：换位思考，了解别人的举动。

人总是以自己为出发点思考问题。

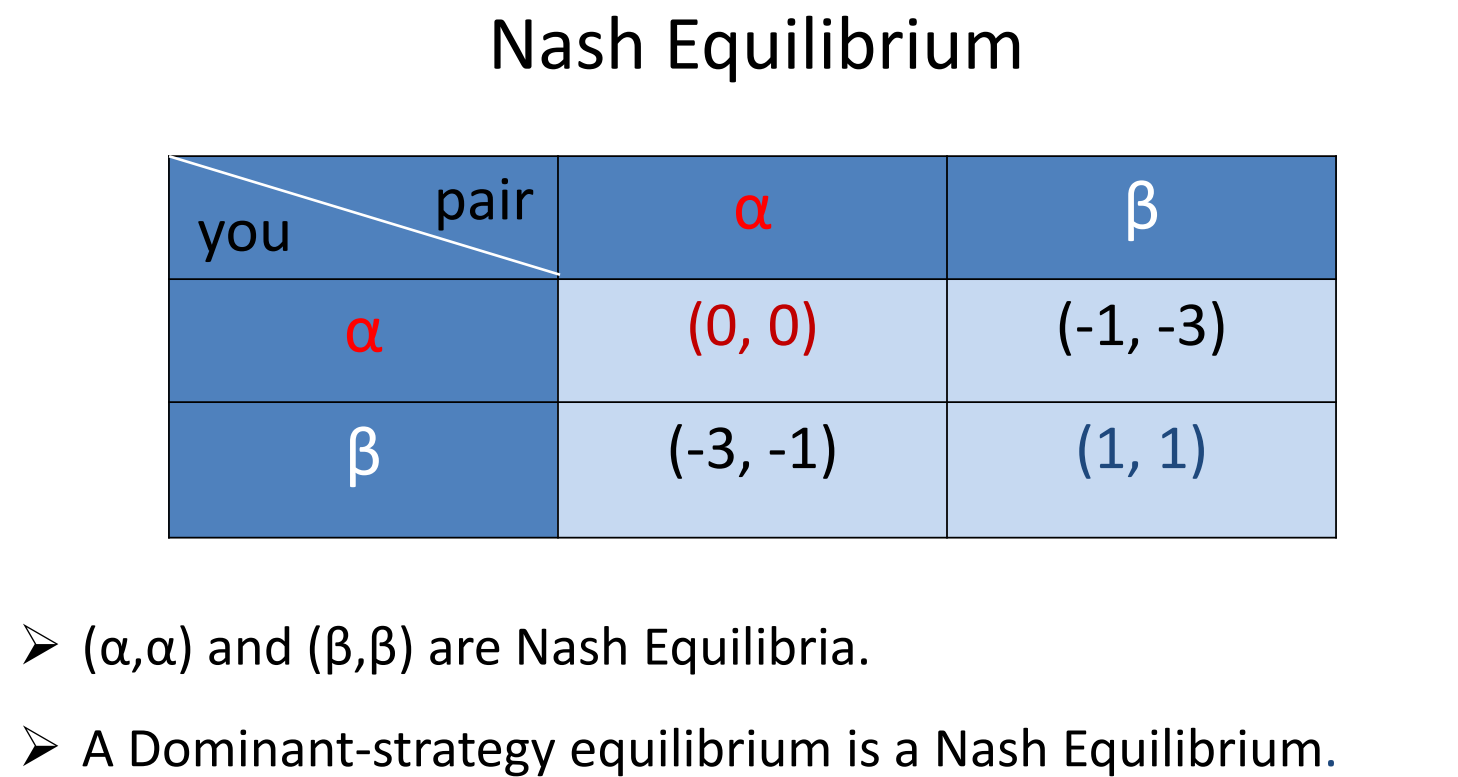


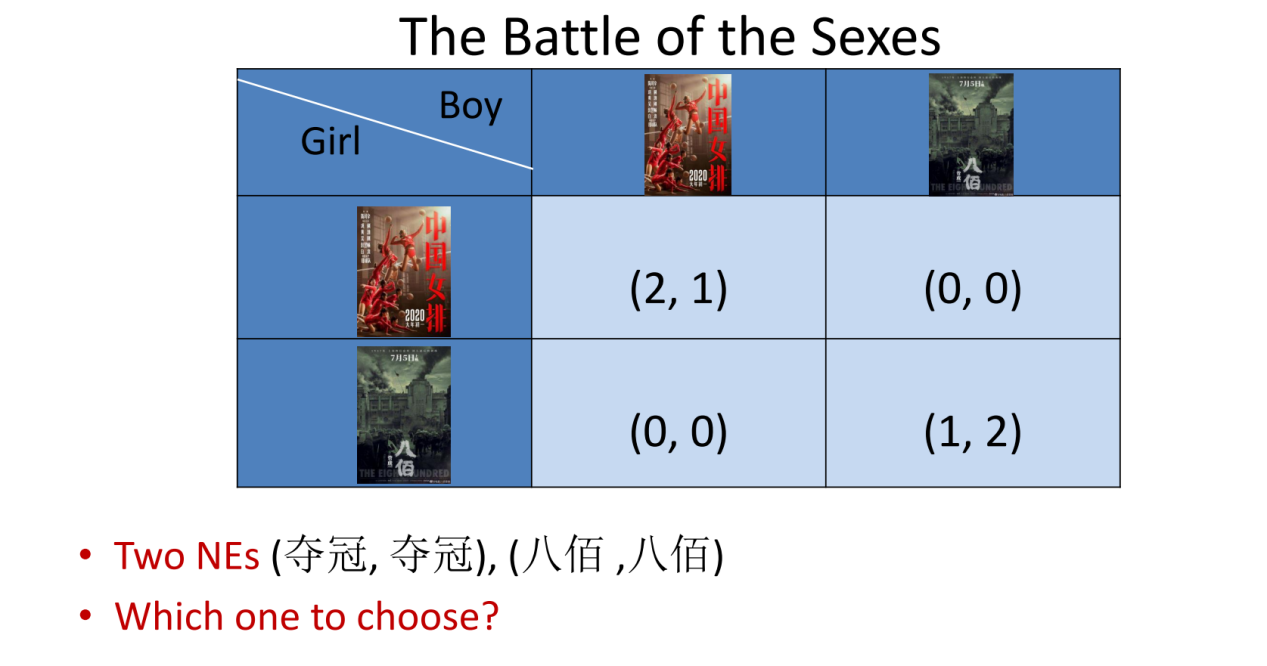




纳什均衡：

一套策略，每个玩家一个，这样没有一个玩家有动力偏离她的策略，因为其他玩家没有偏离





蒙特卡罗树搜索(Monte Carlo Tree Search)并不是一种"模拟人"的算法。而是通过随机的对游戏进行推演来逐渐建立一棵不对称的搜索树的过程。可以看成是某种意义上的强化学习，当然这一点学界还有一些争议。

蒙特卡罗树搜索大概可以被分成四步。选择(Selection)，拓展(Expansion)，模拟(Simulation)，1.Selection（选择）

从根节点开始，根据某种策略依次选择最佳的子节点，直到到达叶子节点。选

择节点的好坏直接影响搜索的好坏，目前广泛采用的策略是uct算法（Upper

Confidence Bound Apply to Tree）

2. Expansion（扩展）

– 扩展叶子节点，将一个或多个可行的落子添加为该叶子节点的子节点。

3. Simulation（模拟）

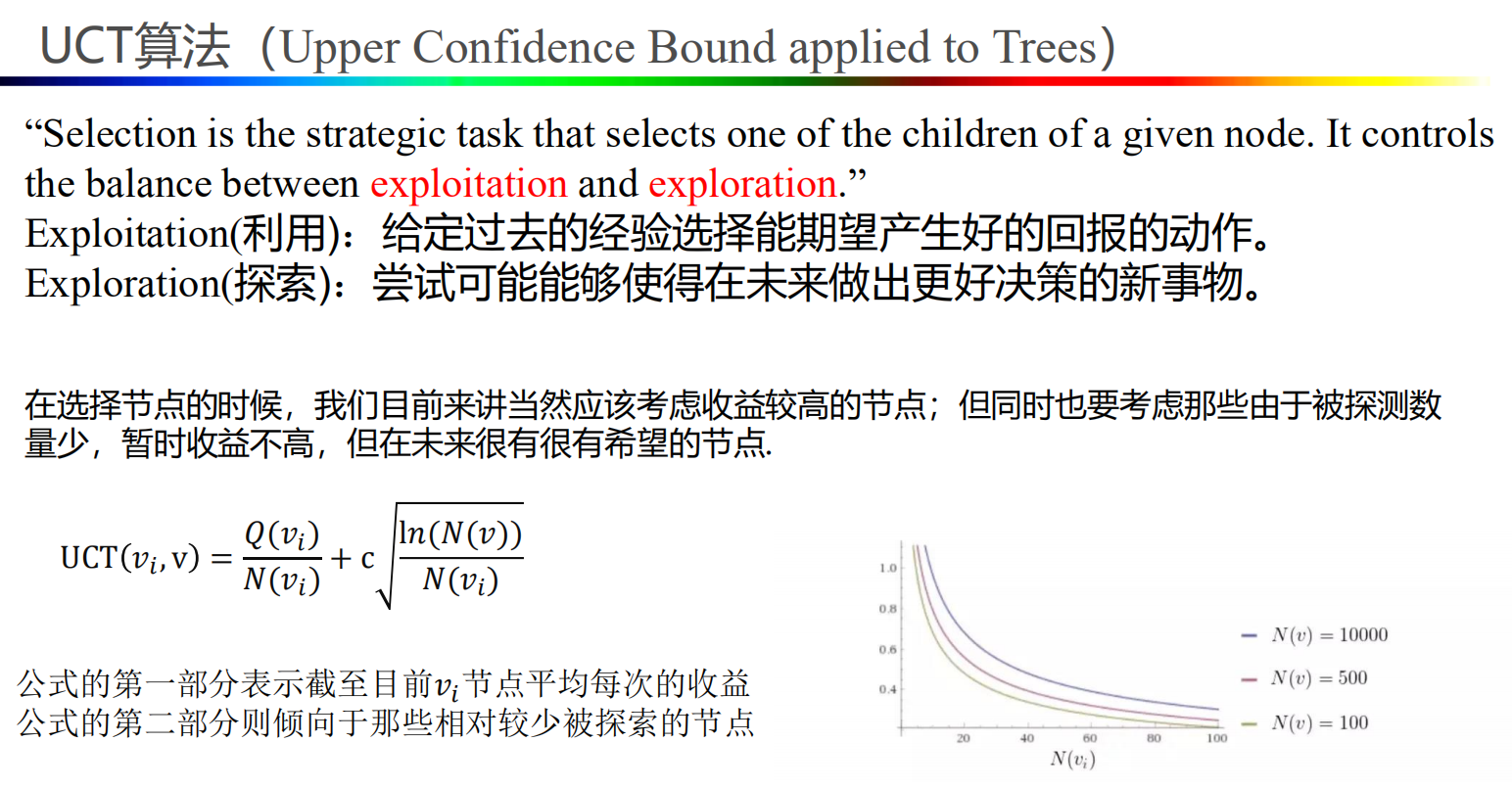
– 根据某种策略（比如围棋中的完全随机落子）从扩展的位置进行到游戏结束。

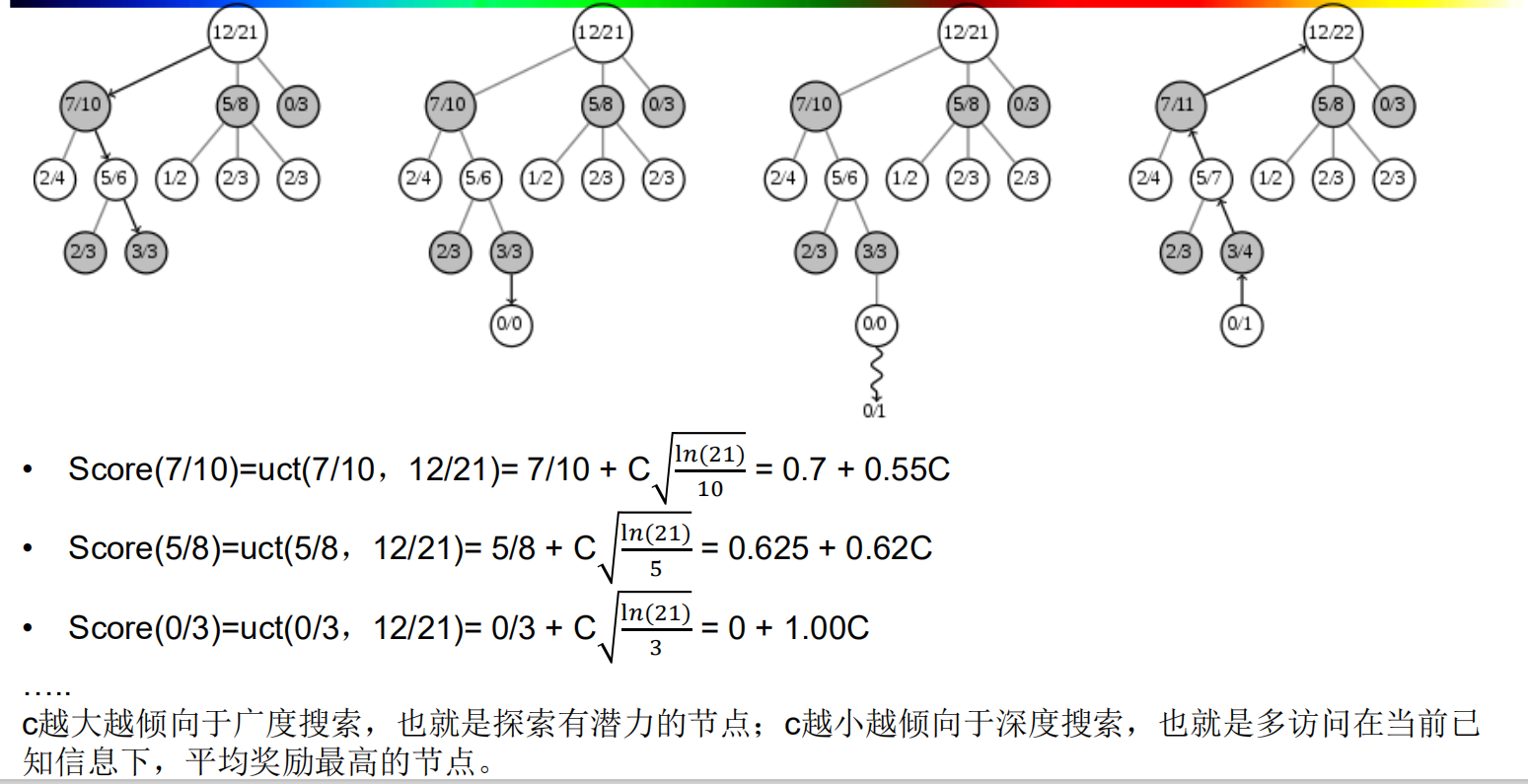
模拟总是会产生一个结果，对于围棋类游戏来说就是获胜、失败或平局，但是

广义上来说模拟的合法结果可以是任意值。

4. Backpropagation（反向传播）

– 将模拟的结果沿着传递路径反向传递回根节点。反向传播(Backpropagation)。





机器学习：

使计算机能模拟人的学习行为，自动的通过学习来获取知识和技能

分类：

监督学习：根据教师给出的正确响应

强化学习：给出奖惩，但不给正确答案

信用分配，游戏，迷宫，平衡木

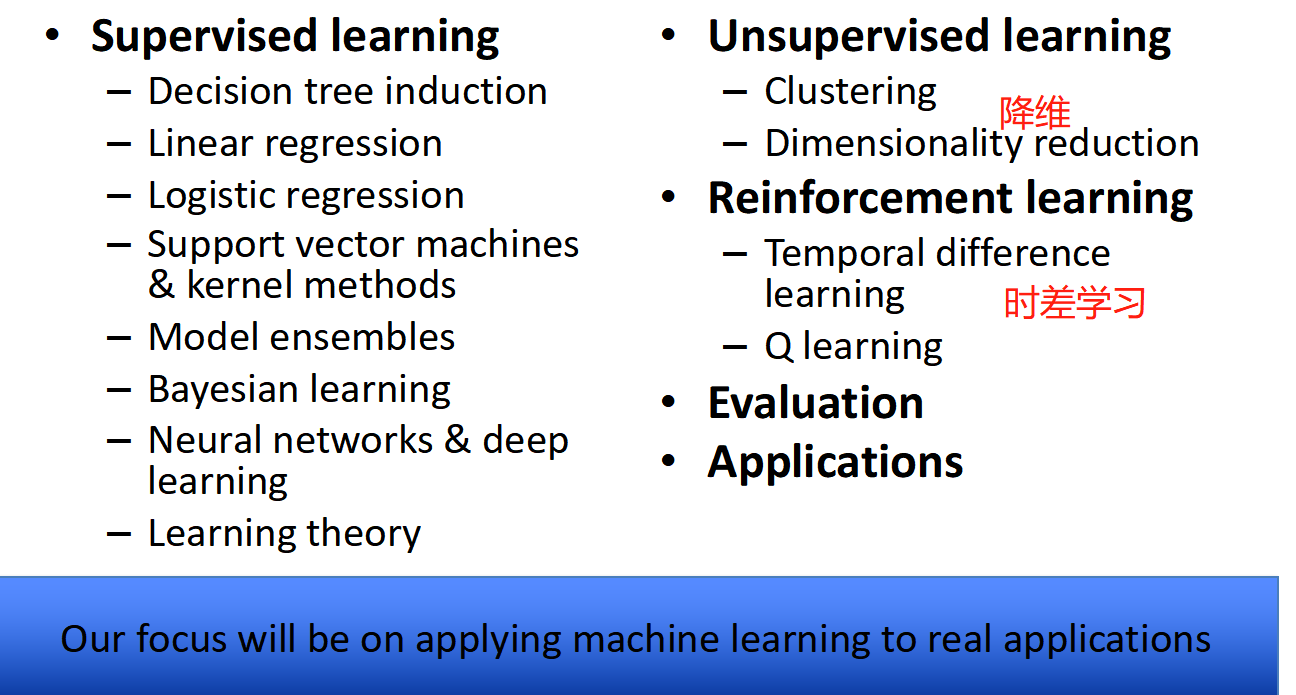
非监督学习：聚类，基因组，管理计算机群，分析社会网，市场划分，天文，信号分离

机械学习：存储足够的信息，通过对知识库的检索得到的知识来求解问题

指导式学习：由外部环境向系统提供指示性建议，转化为知识入库

示例学习：从环境中获取与某概念有关的例子，经归纳得出的一般性概念

迁移学习。做猫狗分类，有少量label data，另外有其他不属于猫狗大量data，有的有label有的没



机器学习=找函数

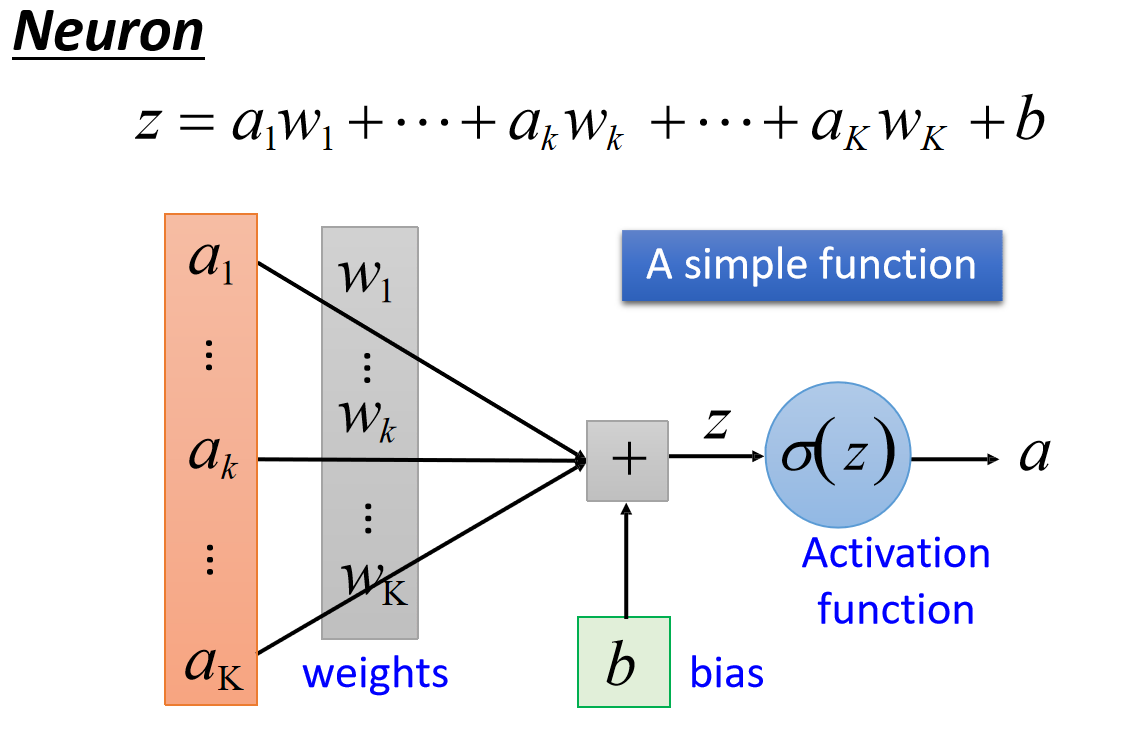
机器学习框架分为training和testing。

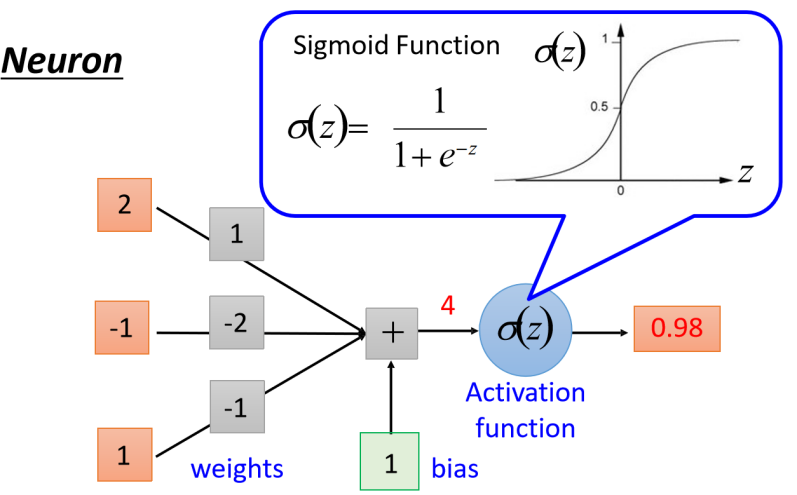
学习部分：准备一个model（a set of function），准备训练数据（training data），经过学习，找出最好的function。

测试部分：找未曾训练过的数据，输入后，能得到正确输出。

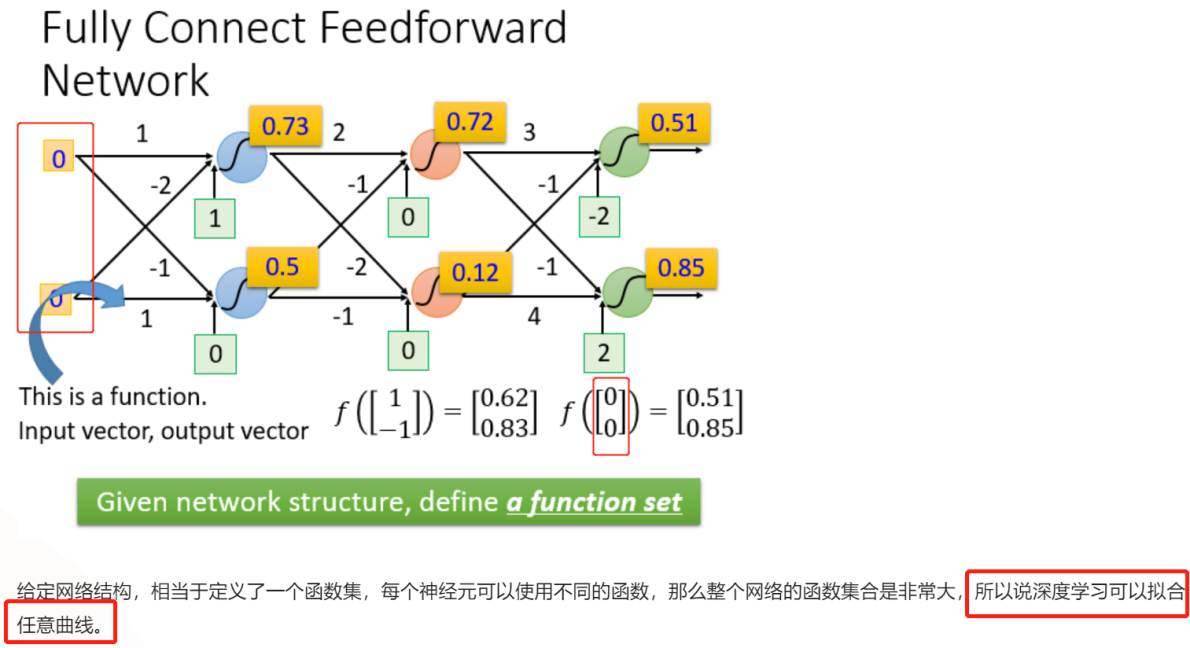
机器学习的三个步骤：定义一个model，能够判断方法好或者不好，能够从model中挑选出最好的方法。

神经网络：

激活函数是sigmoid



每个神经元都有自己的权重和偏差。



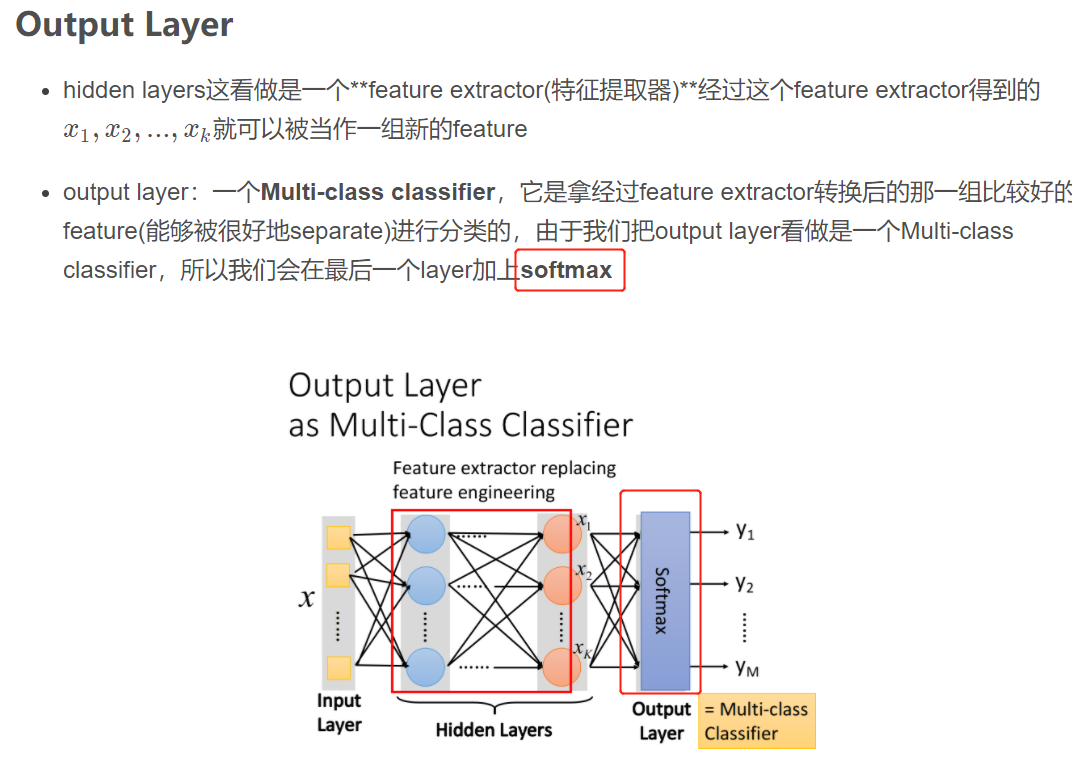
全链接：layer1与layer2之间两两都有连接

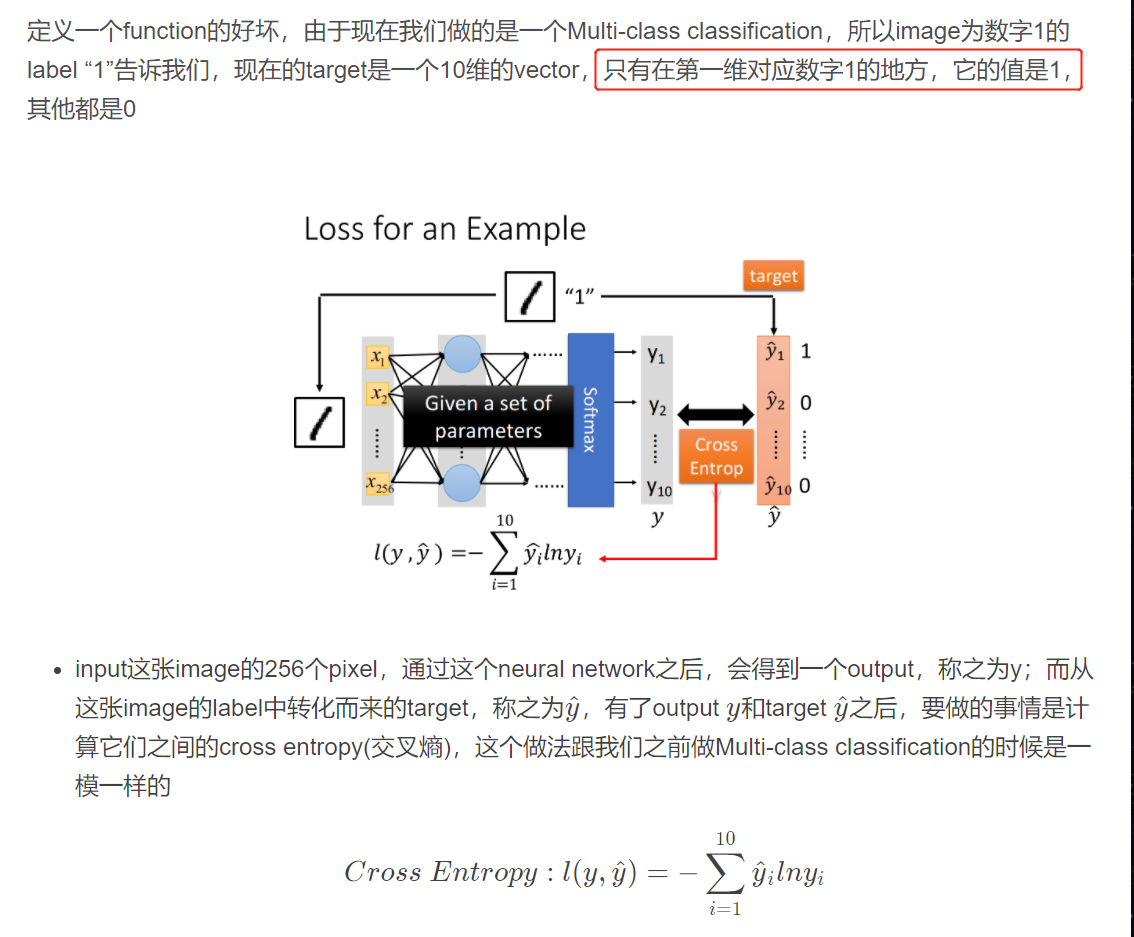
前馈：传递的方向是由后往前传

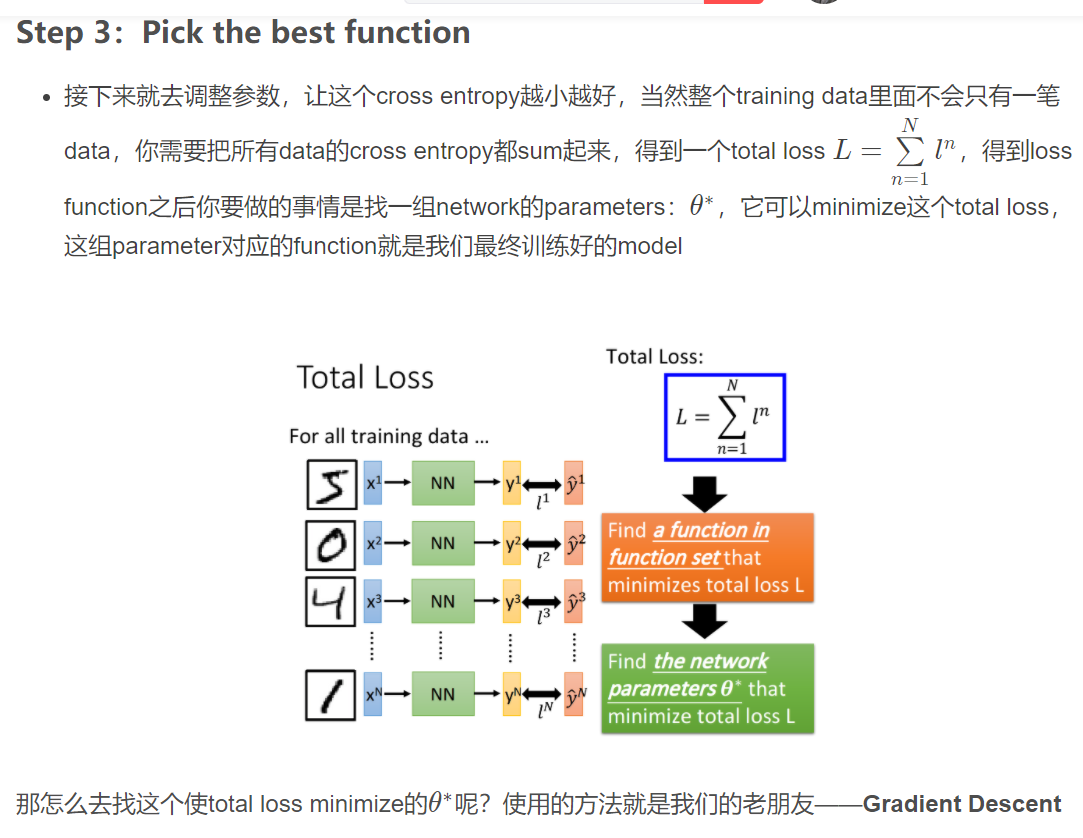
输入层不是有神经元组成的，严格讲不能当作一个layer

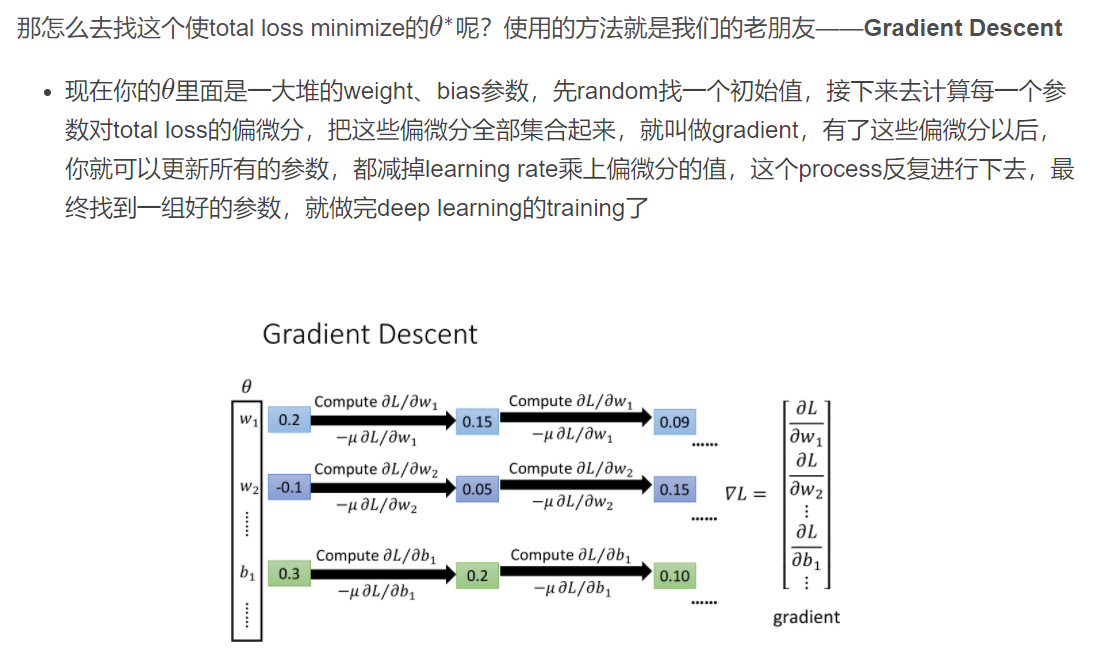
除了输入层和输出层，其他的叫做隐藏层



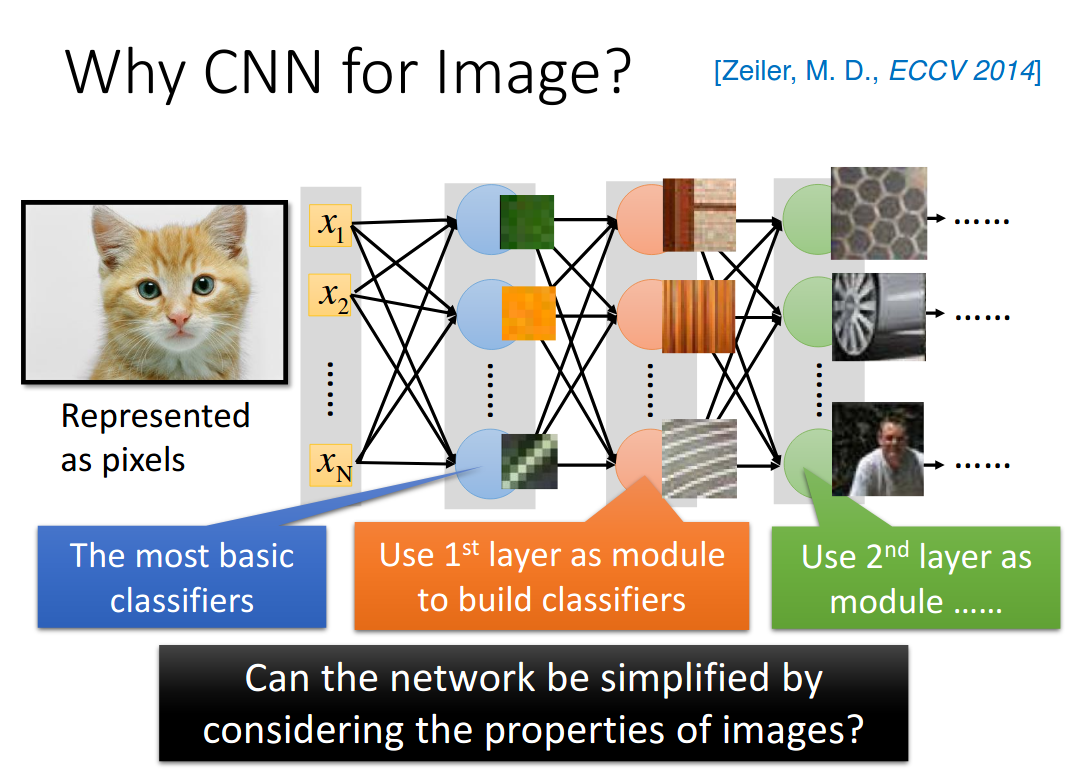








用梯度下降法，不断更新w的值，用w - μ∂L/∂w



第一个layer的neuron，它就是最简单的classifier，它做的事情就是detect有没有绿色出现、有没有黄色出现、有没有斜的条纹出现等等

那第二个layer，它做的事情是detect更复杂的东西，根据第一个layer的output，它如果看到直线横线，就是窗框的一部分；如果看到棕色的直条纹就是木纹；看到斜条纹加灰色的，这个有可能是很多东西，比如说，轮胎的一部分等等

再根据第二个hidden layer的output，第三个hidden layer会做更复杂的事情，比如它可以知道说，当某一个neuron看到蜂巢，它就会被activate；当某一个neuron看到车子，它就会被activate；当某一个neuron看到人的上半身，它就会被activate等等

那现在的问题是这样子：当我们直接用一般的fully connected的feedforward network来做图像处理的时候，往往会需要太多的参数

举例来说，假设这是一张100\*100的彩色图片，它的分辨率才100\*100，那这已经是很小张的image了，然后你需要把它拉成一个vector，总共有100\*100\*3个pixel，把这些加起来input vectot就已经有三万维了；如果input vector是三万维，又假设hidden layer有1000个neuron，那仅仅是第一层hidden layer的参数就已经有30000\*1000个了，这样就太多了

所以，CNN：来简化这个neural network的架构，我们根据自己的知识和对图像处理的理解，一开始就把某些实际上用不到的参数给过滤掉，不用fully connected network，而是用比较少的参数CNN其实是比一般的DNN还要更简单的。

我们可以对一张image做subsampling(二次抽样)，假如你把它奇数行、偶数列的pixel拿掉，image就可以变成原来的十分之一大小，而且并不会影响人对这张image的理解

一定要看下面的链接：

卷积-最大池-卷积-最大池

<https://blog.csdn.net/chairon/article/details/110088441>

下面的链接是RNN（循环神经网络）的

<https://blog.csdn.net/chairon/article/details/112093821>

词嵌入模型：

简单来说，embedding就是用一个低维的向量表示一个物体，可以是一个词，或是一个商品，或是一个电影等等。这个embedding向量的性质是能使距离相近的向量对应的物体有相近的含义，比如 Embedding(复仇者联盟)和Embedding(钢铁侠)之间的距离就会很接近，但 Embedding(复仇者联盟)和Embedding(乱世佳人)的距离就会远一些