人工智能导论期末复习

张宇琛

（西安交通大学，电信学院，计算机试验班 81，2181411919）

一、 Summary

1.1 **任老师考察内容**

- Agent 的体系结构；
- 无信息的搜索-BFS,DFS,IDS；
- 有信息的搜索-爬山法，模拟退火，遗传算法；
- 对抗搜索-$\alpha-\beta$ 剪枝；
- 不确定性的量化-条件概率，联合分布，独立性，贝叶斯规则；
- 概率推理-贝叶斯网络；

二、 Agent-**智能体**

2.1 Summary

- 智能体通过执行器和传感器与环境交互
- 智能体函数描述智能体在各种环境下的行为
- 最佳的理性智能体能够最大化期望表现
- 智能体程序实现了一些智能体函数
- PEAS 描述定义了任务的环境
- 不同的环境
  - 可观察性（完全 & 部分）
  - 确定性 & 随机性
  - 上一步的决策是否会影响下一步
  - 静态 & 动态
  - 离散 & 连续
  - 单智能体 & 多智能体（single&multi-agent）
- 不同的智能体种类

  - 简单反射型智能体 (simple reflect agents)
  - 基于模型型智能体 (model-based agetns)（基于当前环境信息，做出对状况的最好猜测）
  - 基于目标型智能体 (goal-based agents)（知道环境还不够，还要加入目标）
  - 基于效用型智能体 (utility-based agents)（目标不止一个）
- 所有的智能体都能通过学习来提高表现

2.2 **智能体函数和程序**

表驱动智能体 (table-lookup agent) 的缺点

1. 表格庞大，占用大量内存
2. 建表耗时长
3. 不能自治（没有学习能力）
4. 即使有学习能力，也需要很长时间来学习表中的条目（对于一个大表来说，更新里面的值是非常麻烦的）

2.3 **理性与完美**

理性 (rational) 意味着使期望最大化，完美 (perfect) 意味着使实际效果最大化
一般通过 PEAS 判断一个智能体是否为理性智能体

2.4 **设计智能体四大要素** (PEAS)

1. 性能度量 (Performance Measure)
2. 环境 (Environment)
3. 执行器 (Actuators)
4. 传感器 (Sensors)

三、 **无信息搜索策略** (Uninformed Searching)

3.1 **定义**

无信息搜索策略指的是除了问题定义中提供的状态信息外没有任何附加信息。

3.2 **无信息搜索算法总结**

1. 广度优先搜索 (Breadth-First) 总是扩展搜索树中深度最浅的结点，算法是完备的，在单位代价的情况下是最优的，但是具有指数级别的空间复杂度；

2. 一致代价搜索 (Uniform-Cost) 扩展的是当前路径代价最小的结点，对于一般性的步骤代价而言是最优的；
3. 深度优先搜索 (Depth-First) 扩展搜索树中深度最深的结点，既不是完备的，也不是最优的，但是具有线性的空间复杂度；
4. 深度受限搜索 (Depth-Limited) 在深度优先搜索的基础上加了深度限制解决无限空间状态问题；
5. 迭代加深的深度优先搜索 (Iterative Deepening) 在不断增加的深度限制上调用 DLS 直到找到目标，是完备的，单位代价的情况下是最优的，时间复杂度可与 BFS 比较，具有线性的空间复杂度；
6. 双向搜索 (Bidrectional) 可以自爱很大程度上降低时间复杂度，但是它并不总是可行的，而且有时需要过多的内存空间。

3.3 BFS

先扩展根结点, 接着扩展根结点的所有后继, 然后再扩展它们的后继, 依此类推。一般地, 在下一层的任何结点扩展之前, 搜索树上本层深度的所有结点都应该已经扩展过 (FIFO 队列)。

时间复杂度为指数级，一般来讲，指数级复杂度不能用无信息搜索的搜索算法求解，除非是规模很小的实例。

BFS 只有在每步代价都一致时是最优的。

3.4 UCS

对 BFS 的改进，对任何单步代价函数都是最优的（使用优先队列），不再扩展深度最浅的结点，而是扩展路径消耗最小的结点。

3.5 DFS

使用 LIFO 队列，总是扩展搜索树当前边缘结点集中最深的结点。

不论是基于图还是树搜索的 DFS 都不是最优的，但是 DFS 相比于 BFS 在空间复杂度上有很大优势。

回溯搜索是 DFS 的一种变形。

3.6 DLS

深度受限搜索通过对 DFS 设置界限来解决 DFS 在无限状态空间搜索时的问题，就是说深度为 l 的结点被当做没有后继来对待，从而解决了无穷路径的问题。

注：DFS 和 DLS 均不完备

3.7 IDS

Iterative Deepening Search 迭代加深的深度优先搜索常与 DFS 结合来确定最好的深度界限。

做法：不断地增大深度限制，从 0 开始，接着 1,2，以此类推，直到找到目标。

3.8 **双向搜索**

同时运行两个搜索，一个从初始状态向前搜索，同时另一个从目标状态向后搜索，希望它们在中间某点相遇。因为 $2b^{\frac{d}{2}} << b^d$。

但是当目标状态是一种抽象描述的时候，很那应用双向搜索。

四、 有信息搜索策略 (Informed Searching)

4.1 **定义**

使用问题本身的定义之外的特定知识进行问题求解。

4.2 **有信息搜索算法总结**

1. 最佳优先搜索 (best-first search)
   - 贪婪最佳优先搜索 (Greedy search) 扩展 h 最低的结点，一般来说既不完备也不是最优的；
   - A* 搜索扩展 g+h 最低的结点，当启发式 h 是可采纳的且一致的时是最优的
   - A* 算法的最优性
2. 内存受限搜索 (Memory Bounded Search)
   - 迭代加深 A*(Iterative Deepening A*)
   - 递归最佳优先搜索 (Recursive Best-First Search)
   - 简化版 MA*(Simplified Memory-Bounded A*)
3. 启发式
   - 启发式的性能
   - 设计可采纳的启发式
     - 从松弛问题 (relax problem) 出发设计
     - 从子问题 (subproblem) 出发设计
     - 从经验 (experience) 中学习启发式

4.3 贪婪最佳优先搜索

试图扩展离目标最近的结点，只用启发式信息，即 $f(n) = h(n)$

贪婪最佳优先搜索与 DFS 类似，即使是有限状态空间，也是不完备的

4.4 A* 搜索

对结点的评估结合了 $g(n)$，即到达此结点已经花费的代价，和 $h(n)$，即从该节点到目标结点所需的代价 $f(n) = g(n) + h(n) =$ 经过结点 n 的最小代价解的估计代价

假设启发式函数满足特定的条件，A* 搜索既是完备的也是最优的，算法与一致代价搜索类似，但是 A* 使用 $g + h$，因此可以理解为 $A^* \approx greedy + Uninform - cost$

保证最优性的条件：可采纳性和一致性

1. $h(n)$ 是一个可采纳启发式，即它从不会过高估计到达目标的代价
2. 一致性（单调性）：只作用于在图搜索中使用 A* 算法。称启发式 $h(n)$ 是一致的如果对于每个结点 n 和通过任一行动 a 生成的 n 的每个后继结点 n'，从结点 n 到达目标的估计代价不大于从 n 到 n' 的单步代价与从 n' 到达目标的估计代价之和，即 $h(n) \leq c(n, a, a') + h(n')$

一致的启发式都是可采纳的，如果 $h(m)$ 是可采纳的，那么 A* 的树搜索版本是最优的；如果 $h(n)$ 是一致的，那么图搜索的 A* 算法是最优的

4.5 迭代加深 A* 算法 (Iterative Deepening A*-IDA*)

A* 算法的最大缺点：在找到解之前就耗尽内存，因此将迭代加深的思想用在启发式搜索上来减少内存需求 IDA* 的特点：

1. 使用 $f(g + h)$ 作为截断值而非深度
2. 初始截断值：$f(s_0) = h(s_0)$
3. 当 $f(n) < cutoff$ 时使用 DFS
4. 每次迭代，截断值取超过上一次迭代截断值的结点中最小的 f 代价值
5. 与 A* 一样是完备且最优的
6. 空间复杂度为 O(bd)
7. 对于每步都是单位代价的问题是最优的

4.5.1 递归最佳优先搜索 (Recursive Best-First Search)

是模仿标准的最佳优先搜索的递归算法，且只占用线性的空间属性：

1. 与 A* 一样是完备且最优的
2. 时间复杂度取决于启发式函数的精确性和最佳路径的改变频率

3. 空间复杂度 O(bd)
4. 可能需要重新扩展已经遗忘的结点来创建最佳路径

问题：IDA* 与 RBFS 使用的内存空间都太小：

- 在每次迭代之间，IDA* 保留一个数字即 current f-limit
- RBFS 占用的内存空间稍多，但也只有线性的空间，不能从中获益

因此应该多使用可用的空间（不要浪费空间）

4.5.2 充分利用内存的算法

1. MA*(内存受限 A* Memory-Bounded A*)
2. SMA*(简化版 MA* Simplified Memory-Bounded A*)
   - 思路：正常运行 A* 直到内存耗尽，然后每次加入新结点时都丢弃一个旧结点 (SMA* 总是丢弃最差的叶结点，即 f 值最大的叶结点)
   - 问题：很多结点有相同的 f 值
   - 解决方法：删除最旧的，扩展最新的

五、 超越经典搜索 (Beyond Classical Search)

5.1 爬山法 (Hill-Climbing)

选择附近情况中状态最好的一个，而不考虑下一步该如何走，因此有时被成为贪婪局部搜索，可以轻松地改变一个坏的状态，但是经常会陷入困境

- 局部极大值：怎样改变都会比当前状况更差；
- 山脊；
- 高原：一块平的区域，可能是一块平的局部极大值，或者是山肩，爬山法会在高原迷路；

不完备的爬山法：

- 最陡上升爬山法（上述）
- 随机爬山法：在上山移动过程中随机地选择下一步，被选中的概率可能随着上山移动的陡峭程度不同而不同
- 首选爬山法：随机地生成后继结点，知道生成一个优于当前结点的后继（适用于后继结点很多时）

完备的概率接近 1 的爬山法：随机重启爬山法
随机生成初始状态来引导爬山法搜索，直到找到目标

5.2 **模拟退火搜索** (Simulated Annealing-SA)

爬山法搜索不下降，因此会困在局部最优解；而纯粹的随机行走是完备的但效率很低，因此需要把爬山法和随机行走相结合，同时得到效率和完备性。SA 没有选择最佳移动，而是选择了随机移动，如果该移动使得情况改善，则接受该移动，否则以一个小于 1 的概率接受该移动。

5.3 **局部束搜索** (Local Beam Search)

在内存中记录 k 个状态，从 k 个随机生成的状态开始，每一步全部的 k 个状态的所有后后继状态全部被生成，如果有一个是目标状态，则算法停止，否则从整个后继列表中选择 k 个最佳的后继。

由于这 k 个状态缺乏多样性，会很快收敛到局部最优解，因此使用随机束搜索，类似随机爬山法，并不是从候选后继集合中选择 k 个最好的后继，而是随机选择 k 个后继状态，其中选择给定后继状态的概率是状态值的递增函数。

5.4 **遗传算法** (Genetic Algorithm-GA)

是随机束搜索的一种变形，通过把两个父状态结合来生成后继，而不是通过单一的修改状态进行。

同随机束搜索一样，遗传算法结合了上山趋势，随机探索和在并行搜索线程之间交换信息。

- 从 k 个随机生成的状态开始，成为种群;
- 每个状态成为个体，用一个有限长度的字符串表示，通常是 0-1 串;
- 每个状态由目标函数/适应度函数给出评估值;
- 对于要配对的每对个体，在字符串中随机选择一个位置作为杂交点;
- 最后在每个位置都会按照某个晓得独立概率随机变异

六、 对抗搜索

对抗搜索问题通常被称为博弈。博弈要求具备在无法计算出最优决策的情况下也要给出某种决策的能力。

- 剪枝允许我们在搜索树中忽略那些不影响最后决定的部分;
- 启发式的评估函数允许在不进行完全搜索的情况下估计某状态的真实效用值

6.1 **游戏的形式化**

- S_0：初始状态，规范游戏开始时的情况;

- PLAYER(s)：定义此时轮到谁行动；
- ACTIONS(s): 返回此状态下的合法移动集合；
- RESULT(s,a)：转移模型，定义行动的结果；
- TERMINAL-TEST(s)：终止测试，游戏结束返回真，否则为假。游戏结束的状态成为终止状态；
- UTILITY(s,p)：效用/目标/收益函数。

6.2 **极小极大算法** (MinMax Search)

对博弈树进行完整的深度优先搜索。时间复杂度为 $O(b^m)$，一次性生成所有后继的算法，空间复杂度为 $O(bm)$，每次生成一个后继的算法的空间复杂度为 $O(m)$。

时间复杂度过大。

6.3 $\alpha-\beta$ **剪枝** ($\alpha-beta$ Pruning)

剪掉不可能影响决策的分支，仍然返回和极小极大算法同样的结果。

- α：到目前为止路径上发现的 MAX 的最佳（即极大值）选择；
- β：到目前为止路径上发现的 MIN 的最佳（即极小值）选择；

当出现一个值比当前路径上的最佳值差（比 MAX 小或者比 MIN 大）时，将该分支剪去。

$\alpha-\beta$ 剪枝的效率受检查后继状态的顺序影响很大，应该首先检查可能的最好后继。因此增加动态行棋的排序方案，如先试图采用以前走过的最好行棋，等等。最好行棋成为绝招，先走绝招成为绝招启发式。使用以前见过的棋局的哈希表一般称为换位表。

$\alpha-\beta$ 剪枝在总是检查最好后继的情况下，只需要检查 $O(b^{\frac{m}{2}})$ 个结点。

6.4 **不完美的实时决策** (Imperfect Real-Time Decisions)

极小极大算法即使使用 $\alpha-\beta$ 剪枝依然要搜索部分空间直到终止状态，这样的搜索深度耗时依然过大。因此将启发式评估函数用于搜索中的状态，把非终止结点变成终止结点，用截断测试 (CUTOFF-TEST) 代替终止测试 (TERMINAL-TEST)。

七、 **不确定性的量化** (Uncertainly)

7.1 **概率相关知识**

1. 条件概率：
   $P(A|B)=\frac{P(AB)}{P(B)}$
   $P(ABC)=P(A)P(B|A)P(C|AB)$

2. 全概率公式：若事件 $B_1, B_2, \ldots, B_n, \ldots$ 构成互斥完备事件组，且 $P(B_i) > 0$，则对任一事件 A 有：
   $P(A) = \sum_j P(AB_j) = \sum_j P(B_j)P(A|B_j)$
3. 贝叶斯定理：$P(Y|X) = P(X|Y)\frac{P(Y)}{P(X)}$
4. 贝叶斯公式：若事件 $B_1, B_2, \ldots, B_n, \ldots$ 构成互斥完备事件群，且 $P(B_i) > 0$，则对任一事件 A($P(a) > 0$) 有：
   $P(B_i|A) = \frac{P(B_i)P(A|B_i)}{\sum_j P(B_j)P(A|B_j)}$

7.2 条件概率

1. 无条件概率 (unconditional probabilities)/先验概率 (prior probabilities)：不知道其他信息的情况下对命题的信念度，eg.P(n)；
2. 条件概率 (conditional probabilities)/后验概率 (posterior probabilities)：提供了一些我们已经知道的信息（证据）。

7.3 完全概率分布

对于任意命题 ϕ，有

$$P(\phi) = \sum_{\omega \in \phi} P(\omega)$$

因此计算任何命题不论是简单命题还是复合命题，只要把他们对应为真的概率相加即可。

完全联合概率分布指定了对随机变量的每种完整赋值的概率。不过完全联合概率分布通常过于庞大, 难以对其进行显式地创建和使用。当完全联合分布可用时, 它可以用于回答查询, 只需简单地将其中对应于查询命题的可能世界的条目相加。

7.4 独立性

影响牙疼的因素：toothache, catch, cavity，现在加入一个天气因素，根据乘法规则，

$$P(toothache, catch, cavity, cloudy) = P(cloudy|toothache, catch, cavity)P(toothache, catch, cavity)$$

显然天气不影响牙病变量，因此

$$P(cloudy|toothache, catch, cavity) = P(cloudy)$$

因此可以推断

$$P(toothache, catch, cavity, cloudy) = P(cloudy)P(toothache, catch, cavity)$$

因此 Weather 中每个条目都有类似的公式，即

$$P(toothache, catch, cavity, weather) = P(toothache, catch, cavity)P(weather)$$

在这个例子中天气是独立于牙病问题的，这种特性被称为独立性 (independence)/边缘独立性 (marginal independence)/绝对独立性 (absolute independence)。

$$P(a \wedge b) = P(a)P(b), P(X, Y) = P(X)P(Y)$$

独立性和条件概率都是减小联合概率分布规模的工具。

随机变量的子集之间的绝对独立性允许将完全联合分布分解成多个更小的联合分布。这样做能够大大降低问题的复杂度，但在实际中绝对独立性很少出现。

7.5 贝叶斯规则及其应用

贝叶斯规则允许我们基于 $P(a|b), P(a), P(b)$ 计算 $P(b|a)$，通常将未知因素 (cause) 带来的结果 (effect) 看做证据，来确定未知因素 cause

$$P(cause|effect) = P(effect|cause)\frac{P(cause)}{P(effect)}$$

贝叶斯规则允许通过已知的条件概率（通常是因果方向的）计算未知的概率。应用有多条证据的贝叶斯规则时，会遇到与完全联合分布同样的规模扩展问题。

朴素贝叶斯 (Naive Bayes) 模型：当单一原因直接影响很多结果时，

$$P(cause|effect_1, effect_2, \ldots, effect_n) = P(cause)\prod_i P(effect_i, cause)$$

八、 贝叶斯网络 (Bayesian Network)

8.1 贝叶斯网络的概念

贝叶斯网络是一个有向图，每个结点都标注了定量的概率信息。

1. 每一个结点对应一个随机变量，这个变量可以是离散的或者连续的；
2. 一组有向边或者箭头连接结点时，如果有从 X 指向 Y 的箭头，则称 X 为 Y 的一个父结点，图中没有又向回路（有向无环图，DAG）；
3. 每个结点 X_i 有一个条件概率分布 $P(X_i|Parents(X_i))$，量化其父结点对其的影响。

8.2 贝叶斯网络的语义

8.2.1 构建贝叶斯网络（从联合概率分布角度理解贝叶斯网络）

链式规则：

$$P(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | x_{i-1}, \ldots, x_1)$$

构建贝叶斯网络的方法：

1. 结点 (Nodes)：确定一组结点 $\{X_1, \ldots, X_n\}$，任何顺序都可以，但应该尽量让原因结点在前；
2. 边 (Links):for i = 1 to n:
   - 从 $X_1, \ldots, X_n$ 中选择 X_i 的父结点的最小集合，使得 $P(X_i|X_{i-1}, \ldots, X_1) = P(X_i|Parents(X_i))$ 得到满足；
   - 在每个父结点和 X_i 之间插入一条边；
   - 写出条件概率表 (conditional probability table, CPDs)$P(X_i|Parents(X_i)‘')$

8.3 包含连续变量的贝叶斯网络

处理连续问题的方法

1. 离散化，可能导致很大的误差和很大的条件概率表 CPTs；
2. 使用一族有限的参数进行指定
   - 连续变量，离散 + 连续的父结点；
   - 离散变量，连续父结点。

混合贝叶斯网络 (Hybrid Bayesian Network) 是同时包含离散随机变量和连续随机变量的网络。

线性高斯分布 (Linear Gaussian Distribution)：子结点服从高斯分布，其均值 μ 随父结点的值线性变化，标准差保持不变

$$P(c|h, subsidy) = N(a_t h + b_t, \sigma^2)(c) = \frac{1}{\sigma_t \sqrt{2\pi}} e^{-\frac{1}{2}(\frac{c-(a_t h+b_t)}{\sigma_t})^2}$$

当离散变量作为连续变量的父结点加入时，网络就定义一个条件高斯 (conditional Gaussian, CG) 分布。

使用高斯积分的概率分布：

$$\Phi(x) = \int_{-\inf}^{x} N(0,1)(x)dx$$

$$P(Buys = true|Cost = c) = \Phi(\frac{-c+u}{\sigma})$$

九、 机器学习基本概念

人工智能的复杂性：

1. 计算复杂性：指数增长；
2. 信息复杂性：需要获取知识

人工智能（机器展示的智能）> 机器学习（从数据中获取知识的方法）> 深度学习（一种利用神经网络实现机器学习的技术）

实现机器学习的三大要素：

1. 任务 T(Task)
   - 是智能系统执行的、实现某种目标的工作，也可以描述为智能系统处理一个样本 (example) 的工作；
   - 样本 (example) 是从对象或事件中收集到的已经量化的特征 (feature) 集合；
   - 特征 (feature) 是从对象或事件中提取的描述某种属性的度量；
   - 几种典型的机器学习任务：
     - 分类 (Classification)eg. 目标识别；
     - 回归 (Regression)eg. 股价预测、年龄估计、距离估计；
     - 转录 (Transcription)eg. 语音、字符识别；
     - 机器翻译 (Machine Translation)；
     - 结构输出 (Structured Output)eg. 语句解析、图像内容解析、场景综合理解；
     - 合成和采样 (Synthesis and Sampling)eg. 图像生成、数据合成；
     - 去噪 (Denoising)；
2. 性能 P(Performance)
   - 用来描述机器学习算法的能力，与具体任务相关；
   - 分类任务：正确率、准确性、0-1loss；
   - 概率估计：K-L Divergence、average log-probability；
   - 对不同任务来说，不一定能精确定义其性能度量；
3. 经验 E(Experience)
   - 是人们知识积累的抽象；
   - 在 ML 算法中，经验就是数据集 (dataset)；
   - 数据集是样本 (example,sample) 的集合，样本也叫数据点 (data point)；
   - 训练数据包含样本数据与标记或目标为监督学习 (Supervised)
     只包含样本数据而无标记或目标为非监督学习 (Umsupervised)。

机器学习是研究算法的学科，这些算法利用经验 E 使得在某种任务 T 中的性能 P 得到提高

泛化能力：

- 一个机器学习算法在先前未观测到的输入数据上执行某种任务的能力称作该算法的泛化能力；
- 机器学习算法追求的最终目标是在测试集上有更小的误差，即更好的性能。

独立同分布假设：

- 理论上，独立同分布的训练集与测试集有相同的误差；
- 而事实上测试误差大于训练误差。

判断机器学习算法效果好坏的两个因素：

1. 训练误差小（否则欠拟合）；
2. 训练误差与测试误差之间的差距小（否则过拟合）。

模型的容量 (Capacity)

- 表示了模型拟合函数的能力；
- 容量低的模型很难拟合训练集；
- 容量高的模型会记住不适用于测试集的训练集性质（过拟合）。

模型学习中参数估计一般准则的最常用的一种是最大似然估计 (Maximum Likelihood Estimation) 构建一个机器学习算法的要素：

- 数据库
- 代价函数
- 优化方法
- 模型

十、 概率图模型与马尔科夫随机场 (Probabilistic Graphical Model and Markov Random Fields)

10.1 概率图模型

1. 概率图包含结点和边，每个结点表示一个/组随机变量，边表示变量之间的概率关系；
2. 概率图以可视化的方式表达了多个随机变量之间的依赖关系，每个概率图是图中所有变量的联合分布；
3. 概率图分为有向图和无向图
   - 无向图中的边没有方向，也称为马尔科夫随机场，有向图称为贝叶斯网络；
   - 一个有向图中，所有变量的联合分布为每个节点以其所有父结点为条件的概率分布乘积；

10.2 随机场基本概念

1. 位点空间 (Site Space): 有限元素集合 $S = \{s_1, s_2, ..., s_m\}$, 表示位点的集合，例如时间、空间坐标集合等;
2. 相空间 (Phase Space)：有限元素集合 $\Lambda = \{\lambda_1, \lambda_2, ..., \lambda_m\}$，表示随机变量的取值空间，例如颜色、类别、几何属性等。
3. 位点空间 S 中每一个位点从相空间 Λ 中取值的随机变量的集合就是空间 S 上的一个随机场：

$$X = \{x(s)|s \in S, x(s \in \Lambda)\}$$

10.3 马尔科夫随机场

五个要素

1. 位点空间 S;
2. 相空间 $Lambda$;
3. 定义在 S 上的近邻系统 $N = \{N_s\}$;
4. 局部特性 $\left\{P^{(s)}\right\}_{s \in S}$;
5. 概率表达;

10.4 吉布斯分布

基本概念

1. 团 (Clique)：任意单位点都是一个团。元素数目大于 1 的子集，若其中任意两点都是近邻，则该子集也是团。所有团的集合为 C;
2. 势 (Potential)：在 Λ^S 上的吉布斯势，是函数 $V_c : \Lambda^S \to R$ 的集合，并且该函数满足
   - 如果 C 不是团，则 $V_c \equiv 0$;
   - 对所有的 $x, x' \in \Lambda^S$，以及所有的 $C \in C$，有 $x(C) = x'(C) \to V_c(x) = V_c(x')$;

吉布斯与马尔科夫的等价性：一个定义在 (S, Λ, N) 上的马尔科夫随机场的联合概率分布 $p(x)$ 是一个吉布斯分布，即

$$p(x) = \frac{1}{Z} e^{-U(x)}$$

U 为能量函数。

十一、 隐马尔科夫模型 (Hidden Markov Models)

11.1 组成要素

1. 状态空间 S。HMM 的状态空间通常是可数的，元素可表示为 i,j,k,...；
2. 输出集 γ。元素 $y_\gamma \in Y$ 对应系统的一个物理输出；
3. 状态转移矩阵 $P = \{p_{ij}\}, p_{ij} = P(j|i), i, j \in S$;
4. 输出概率分布矩阵 $Q = \{q_{sy}\}$，其中 $q_{sy} = P(y|s)$ 是状态 s 下输出 y 的概率，$y \in Y, s \in S$；
5. 初始状态分布 $\pi = \{\pi_i\}, i \in S$。

这五个要素中，状态空间 S 和输出集 γ 对应 HMM 的模型结构，状态转移矩阵、输出概率分布矩阵、初始状态分布对应 HMM 的模型参数，因此使用 $\lambda = (P, Q, \pi)$ 来表示 HMM 的完整参数集。当前不存在好的方法来自动估计模型结构，因此只能利用对问题的了解和直觉判断来设计 HMM 的结构，针对这一结构估计相关参数。

11.2 HMM 的三个基本问题

1. 给定模型参数 λ 和观测序列 $y = (y_1, y_2, ..., y_n)$ 如何才能更有效地计算出 $P(y|\lambda)$?
   - 该问题是评估问题，旨在计算给定模型生成某观测输出序列的概率，eg. 给一段视频，判断能不能进球；
   - 解决方法：前向算法
2. 给定模型参数 λ 和观测序列 $y = (y_1, y_2, ..., y_n)$ 如何才能更有效地计算出最可能生成观测序列 y 的状态序列 $s^* = (s_1, s_2, ..., s_n)$，即 $s^* = argmax P(s|y, \lambda)$?
   - 旨在识别给定模型的隐藏部分，即找出最有可能生成观测输出序列的状态序列
3. 给定多个观测序列 y_i, 我们如何才能找出最优模型参数集 λ 使得对所有的 y_i, $P(y|\lambda)$ 最大?
   - 关系到 HMM 的训练，通过训练我们将得到一个把训练样本似然度最大化的模型；
   - 解决方法：EM(Expectation Maximization) 算法
     (a) 求期望值
     (b) 最大化期望值

十二、 神经网络 (Neural Networks)

12.1 神经网络的基本知识

神经网络的组成

1. 结构 (Architecture)：网络中的变量和它们的拓扑关系；
2. 激励函数 (Activity Rule)：神经元如何根据其他神经元的活动来改变自己的激励值；
3. 学习规则 (Learning Rule)：网络中的权重如何随着实践推进而调整。

12.2 前馈神经网络

基本概念

- 机器学习任务可以表达为一个广义映射 $y = f(x, \theta)$；
- $y = f(x, \theta)$ 可以是线性/非线性模型；
- ϕ 是一个非线性函数，$\phi(x)$ 成为 x 的特征；

构建 ϕ 的三种方式

- 通用映射
- 手动设计
- 自主学习

前馈神经网络 (Feedforward Neural Network) 也叫多层感知机 (Multilayer perceptron)，是一种在模型输出与模型本身之间没有反馈连接的神经网络。

12.3 常见的激活函数

- Sigmoid:$y = \frac{1}{1+e^{-x}}$
- Hyperbolic Tangent:$y = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
- ReLU:$y = max(0, x)$
- Leaky ReLU:$y = max(\alpha x, x)$

十三、 卷积神经网络 (Convolutional Neural Networks)

13.1 池化的作用

1. 特征不变形：池化操作使模型更关注是否存在某些特征而不是特征具体的位置；
2. 特征降维性：减少参数数量，避免过拟合；

13.2 卷积单元的作用

- 局部感知；
- 权值共享；
- 池化降维；

十四、 循环神经网络 (Recurrent Neural Network)

循环神经网络是一种具有从后续层到前面反馈连接或者同层之间神经元连接的神经网络，常用于处理顺序数据。

一般循环神经网络的缺陷

1. 梯度消失：后面时间点的错误信号不能回到足够早的时间点；
2. 难以刻画长期依赖关系；

LSTM 的结构

1. 遗忘门；
2. 输入门；
3. 状态更新；
4. 信息输出

十五、 支持向量机 (Supprot Vector Machines)

定义：SVM 是一类按监督学习 (supervised learning) 方式对数据进行二元分类的广义线性分类器 (generalized linear classifier)

间隔 (margin)：边界在遇到一个数据点前能增加的距离，即到边界最近的数据点到边界的距离。

最大间隔 (maximum margin) 线性分类器为具有最大间隔的线性分类器，也称作线性支持向量机 (Linear SVM, LSVM)。

十六、 自然语言处理 (Natural Language Processing)

记得好像不考。