人工智能基础
期末 CheatSheet

自 74 班 陈奕凡 2017011621

2020 年 1 月 6 日

第 1 章 产生式系统的搜索策略

· 宽度优先搜索算法：
h(n)=0,g(n)=depth(n),f(n)=g(n)+h(n)
算法对 Open 表的操作规则是先进先出，可使用数据结构中的队列来实现。
（1）把起始节点放到 Open 表中，如果该起始节点为一目标节点，则求得一个解，算法结束。
（2）如果 Open 是个空表，则没有解，失败退出；否则继续。
（3）把第一个节点（节点 n）从 Open 表移出（先进先出），并把它放入 Closed 的扩展节点表中。
（4）扩展节点 n。如果 n 没有后继节点，则转向步骤（2）。
（5）把 n 的所有后继节点放到 Open 表的末端 (末端的节点比前端的节点后移出)，并提供从这些后继节点回到 n 的指针。
（6）如果 n 的任一个后继节点是一个目标节点，则找到一个解，成功退出；否则转向步骤（2）。

· 深度优先搜索算法： h 很大，g=0
算法对 Open 表的操作规则是先进后出，可以使用数据结构中的栈来实现。
（1）把起始节点 S 放到未扩展节点 Open 表中。如果此节点为一目标节点，则得到一个解。
（2）如果 Open 为空表，则失败退出。
（3）把 Open 中第一个节点（先进后出）n 从 Open 表移到 Closed 表。
（4）扩展节点 n，产生其全部后裔，并把它们放入 Open 表的前端（末端的节点比前端的节点后移出）。如果没有后裔，则转向步骤（2）。
（5）如果后继节点中有任一个为目标节点，则求得一个解，成功退出；否则，转向步骤（2）。

· 等费用搜索算法
宽度优先 +Open(小顶堆) 排序
（1）把起始节点 S 放到未扩展节点表 Open 中。如果此起始节点为一目标节点，则求得一个解；否则令 g(S)=0。
（2）如果 Open 是个空表，则没有解而失败退出。
（3）从 Open 表中选择一个节点 m，使其 g(m) 为最小。如果有几个节点都合格，那么就要选择一个目标节点作为节点 m（要是有目标节点的话）；否则，就从中任选一个作为节点 m。把节点 m 从 Open 表移至扩展节点表 Closed 中。
（4）如果节点 m 为目标节点，则求得一个解。
（5）扩展节点 m。如果 m 没有后继节点，则转向步骤（2）。
（6）对于节点 m 的每个后继节点 n，计算 g(n)=g(m)+c(n,m)，并把所有后继节点 n 放进 Open 表。提供回到节点 m 的指针。
（7）转向步骤（2）。

· 爬山法：(贪婪)g=0,h(n) 为与目标节点的距离
（1）把起始节点 S 放到未扩展节点表 Open 中。如果此起始节点为一目标节点，则求得一个解；否则计算 h(S)。
（2）如果 Open 是个空表，则没有解而失败退出。
（3）从 Open 表中选择一个节点 m，使其 h(m) 为最小。如果有几个节点都合格，那么就要选择一个目标节点作为节点 m（要是有目标节点的话）；否则就从中任选一个作为节点 m。把节点 m 从 Open 表移至扩展节点表 Closed 中。
（4）如果节点 m 为目标节点，则求得一个解。
（5）扩展节点 m。如果 m 没有后继节点，则转向步骤（2）。
（6）对于节点 m 的每个后继节点 n，计算 h(n)，并把所有后继节点 n 放进 Open 表。提供回到节点 m 的指针。
（7）转向步骤（2）。爬山法不总是能够找到最优解。

· A 算法： g 已走过的路,h* 与目标距离, 估计 f(n)=g+h
（1）把起始节点 S 放入 Open 表, 记 f(S)=h(S), 令 Closed 为空表
（2）若 Open 为空表，则宣告失败。
（3）选取 Open 表中具有最小 f 值的节点为最佳节点 BestNode，并把它放入 Closed 表。
（4）若 BestNode 为一目标节点，则成功求得一解。
（5）若 BestNode 不是目标, 则扩展之, 产生后继节点 Successor
（6）对每个 Successor 进行下列过程：
a. 建立从 Successor 返回 BestNode 的指针。
b. 计算 g(Successor)=g(BestNode)+c(Successor,BestNode)。
c. 如果 Successor 在 Open 表中，则称 Open 表中该节点为 Old
d. 比较新旧路径代价。如果 g(Successor)<g(old)，则用 Successor 替换 Old 节点。
e. 若与 Old 节点的代价低或一样，则停止扩展节点。
f. 若 Successor 不在 Open 表中，则看其是否在 Closed 表中。
g. 若 Successor 在 Closed 表中，则转向 c。
h. 若 Successor 既不在 Open 表中，又不在 Closed 表中，则把它放入 Open 表中，然后转向（7）。
（7）计算该节点 f 值。
（8）转向（2）

· A* 算法： 估计 h*(n) 的下界 h(n)
A* 算法具有完备性、可采纳性、最优性
-完备性 (completeness): 如果一个问题有解，算法就一定能够找到它，称这种算法是完备的。
-可采纳性 (Admisibility): 只要存在代价最小的解，算法就能找到它。称这种算法是可采纳的。
-最优性：对于可采纳的算法

$$A_1 : f_1(n) = g_1(n) + h_1(n)$$

$$A_2 : f_2(n) = g_2(n) + h_2(n)$$

如果：$h_1(n) < h_2(n) < h^*(n)$，就有: 由 A_1 展开的节点数目至少与 A_2 一样多

· 计算复杂度：

$$f(x) = O(g(x)) \Leftrightarrow \exists A, B > 0, A \le |\frac{f(x)}{g(x)}| \le B$$

· P 问题： 有多项式时间算法解决的判定问题。
· NP 问题： 对问题的一个猜想存在多项式时间算法来验证的判定问题。
· 多项式可简化： 问题 A 对于问题 B 是多项式可简化的：A 的算法中要调用 B 作为子程序，如果把 B 的子程序作为一个步骤，A 的算法是多项式时间算法。记为：A∝B。
· NP 完全问题： 判定问题 $P_1 \propto NP$；对所有其它判定问题 Pm NP，有 Pm P1，则 P1 是 NP 完全的。NP 完全问题是 NP 中最难的问题。
· NP 难题： 如果 $P_1 \propto P_2, P_1 \propto NP$ 完全问题，则 P_2 是 NP 难

· 与或图代价表示
1. 若 n 是 N 的一个元素，则 k(n,N)=0；
2. 若 n 是一个外向连接符指向后继结点 $\{n_1, ..., n_i\}$（与后继结点），并设该连接符的代价值为 Cn，则

$$k(n, N) = C_n + k(n_1, N) + ... + k(n_i, N)$$

或者：

$$k(n, N) = \max k(n_i, N) + c(n_i, n)$$

3. 若 n 有 m 个或后继结点：$\{n_1, ..., n_i\}$，则

$$k(n, N) = \min k(n_i, N) + c(n_i, n)$$

对于具有最小代价值的解图，其值也用 $h^*(n_0)$ 标记。

· 与或图的启发式搜索算法 AO*
1) 初始节点为 s，建立一个搜索图 G，开始时图 G 只包括 s，计算 h(s)，若 s 是叶结点，则标记能解。
2) 重复下面循环，直到 s 已被标记为可解节点：
3)Begin
4) 根据连接符标记（指针）找出一个待扩展的局部解图 G'
5) 选 G' 中的任一个非终结点 n 作为当前结点。
6) 扩展节点 n，生成其后继结点集 $\{n_j\}$，并且把它们作为 n 的后继添加到 G 中。对每一个不曾在 G 中出现过的节点 n_j，计算 $h(n_j)$。把其中属于终节点的后继节点标记为可解节点。
7) 建立含 n 的单一结点集合 S。
8) 重复下面循环，直到 S 为空
9)Begin
10) 从 S 中删除一个节点 m，该节点在 G 中的后继不出现在 S 中。
11)(i) 修改 m 的代价值 h(m)：
对 m 指向结点集 $\{n_{1i}, n_{2i}, ..., n_{ki}\}$ 的每一个连接符 i 分别计算 h_i：
$h_i(m) = C_i + h(n_{1i}) + .. + h(n_{ki})$
$h(m) = \min h_i(m)$，即对 m 的 i 个连接符，取代价最小的那个值作为该节点的代价 h(m)。
(ii) 把指针加到 $h_i(m)$ 最小的连接符上，或把指针修改到 $h_i(m)$ 最小的连接符上。
(iii) 若该连接符的所有子结点都是可解的，则 m 也标为可解。
12) 如果 m 可解或修正后的代价值与原来估算的代价值不同，则把 m 的所有父辈结点插入到 S 中。
13)end
14)end

· 与或图的宽度优先搜索：
1. 当找到一个可解节点 A 时，向上回溯确定，A 的祖先是否可解，直到不能再回溯为止。然后去掉 open 表中以最高可解节点为祖先的节点。
2. 当找到一个不可解节点 B 时，向上回溯确定，B 的祖先是否不可解，直到不能再回溯为止。然后去掉 open 表中以最高不可解节点为祖先的节点。

· 博弈树搜索： 极小极大搜索过程
$\alpha - \beta$ 剪枝：极大值层的下界值称为 α，极小值层的上界值称为 β。
1.α 剪枝： 若任一极小值层结点的 β 值小于或等于它任一先辈极大值层结点的 α 值，即 α（先辈层）$\ge \beta$（后继层），则可中止该极小值层中这个 MIN 结点以下的搜索过程。这个 MIN 结点最终的倒推值就确定为这个 β 值。
2.β 剪枝： 若任一极大值层结点的 α 值大于或等于它任一先辈极小值层结点的 β 值，即 α（后继层）$\ge \beta$（先辈层），则可以中止该极大值层中这个 MAX 结点以下的搜索过程。这个 MAX 结点的最终倒推值就确定为这个 α 值。

第 2 章 谓词逻辑

· 公式的分类
-重言式/永真式：命题永远为真
-矛盾式/永假式：命题永远为假
-可满足式：至少有一个成真赋值

· 常见易错的基本等值式
-摩根律：$\sim(A \vee B) \Leftrightarrow \sim A \wedge \sim B$; $\sim(A \wedge B) \Leftrightarrow \sim A \vee \sim B$
-吸收律：$A \vee (A \wedge B) \Leftrightarrow A$; $A \wedge (A \vee B) \Leftrightarrow A$
-蕴含等值式：$A \to B \Leftrightarrow \sim A \vee B$
-等价等值式：$A \leftrightarrow B \Leftrightarrow (A \to B) \wedge (B \to A)$
• 不含任何联结词的公式为原子公式。
• 原子或原子的否定形式称为文字。任何文字的析取式为子句。
• 仅由有限个文字构成的合取式称为**简单合取式**。
• 仅由有限个文字构成的析取式称为**简单析取式**。
• 仅由有限个简单析取式构成的合取式称为**合取范式**
• 仅由有限个简单合取式构成的析取式称为**析取范式**
• 逻辑公式的子句集 S: 合取范式形式的所有子句（元素）的集合。

· 合取范式的求法： (1) 削去对于 $\{\sim, \wedge, \vee\}$ 来说冗余的联结词；(2) 内移或削去否定号；(3) 利用分配率。
· 归结原理/消解原理： 如 $p \vee q$ 和 $r \vee \sim q$ 都为真，则 $p \vee r$ 为真。
· 归结原理/消解原理证明过程： (1) 建立待归结命题公式。首先根据反证法将所求证的问题转化成为命题公式，求证其是矛盾式（永假式）。(2) 求取合取范式。(3) 建立子句集。(4) 归结，对子句集中的子句使用归结规则：a) 归结式作为新子句加入子句集参加归结；b) 归结式为空子句 ，停止。若得到空子句，表示 S 是不可满足的（矛盾），故原命题成立。

****** 量词分配 ******
$(\forall x)(P(x) \wedge Q(x)) \Leftrightarrow (\forall x)P(x) \wedge (\forall x)Q(x)$
$(\exists x)(P(x) \vee Q(x)) \Leftrightarrow (\exists x)P(x) \vee (\exists x)Q(x)$
注意：
$(\exists x)(P(x) \wedge Q(x)) \neq (\exists x)P(x) \wedge (\exists x)Q(x)$
$(\forall x)(P(x) \vee Q(x)) \neq (\forall x)P(x) \vee (\exists x)Q(x)$
$(\forall x)(P(x) \vee Q) \Leftrightarrow (\forall x)P(x) \vee Q$
$(\forall x)(P(x) \wedge Q) \Leftrightarrow (\forall x)P(x) \wedge Q$
$(\forall x)(P(x) \to Q) \Leftrightarrow (\exists x)P(x) \to Q$
$(\forall x)(Q \to P(x)) \Leftrightarrow Q \to (\forall x)P(x)$
$(\exists x)(P(x) \vee Q) \Leftrightarrow (\exists x)P(x) \vee Q$
$(\exists x)(P(x) \wedge Q) \Leftrightarrow (\exists x)P(x) \wedge Q$
$(\exists x)(P(x) \to Q) \Leftrightarrow (\forall x)P(x) \to Q$
$(\exists x)(Q \to P(x)) \Leftrightarrow Q \to (\exists x)P(x)$

· 前束范式： 公式 A 中所有量词都在公式最左边（不含否定词）, 且这些量词辖域都到公式末端，称之为前束范式。
· 化为前束范式的步骤：
第 1 步：消去 →；第 2 步：~ 深入；第 3 步：量词前移；第 4 步：易名、再量词前移

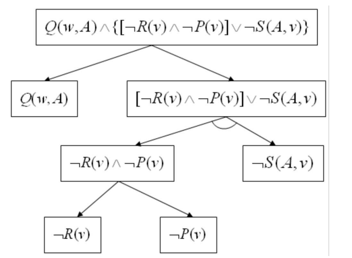
· Skolem 范式
1、先化成前束范式。
2、消去存在量词，即将该量词约束的变量用任意常量或任意变量的函数代替。如果存在量词左边没有任何任意量词，则只将其改写成为常量；如果左边有任意量词的存在量词，消去时该变量改写成为任意量词的函数。
3、略去任意量词，简单地省略掉该量词。

· 子句集 S 的求取过程：
(1) 将谓词公式 G 转换成前束范式；
(2) 消去前束范式中的存在量词，略去其中的任意量词，生成 Skolem 标准型（Skolem 标准型必须满足合取范式）；
(3) 将 Skolem 标准型中的各个子句提出，表示为集合形式。

· 归结过程控制策略：
1. 删除策略：在归结过程中可以随时删除以下子句：
(1) 含有永真式的子句；
(2) 被子句集中其他子句归类的子句。
2．支撑集策略：
设有不可满足子句集 S 的子集 T，如果 S–T 是可满足的，则称 T 是 S 的支撑集。采用支撑集策略时，从开始一直到得到空子句的整个归结过程中，只选取不同时属于 S–T 的子句对，在其间进行归结。就是说，至少有一个子句来自于支撑集 T 或由 T 导出的归结式。

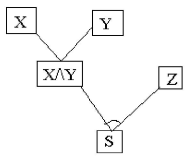
· 事实表达式的与或形变换
方法：消去"→"符号，谓词深入到每一个文字，化成斯柯林范式，删去全称量词，变量更名。
例：
$(\exists u)(\forall v)Q(v, u) \wedge \sim[(R(v) \vee P(v)) \wedge S(u, v)]$
$Q(v, a) \wedge ((\sim R(v) \vee \sim P(v)) \wedge \sim S(a, v))$
$Q(w, a) \wedge ((\sim R(v) \vee \sim P(v)) \wedge \sim S(a, v))$
$Q(w, a) \wedge ((\sim R(v) \vee \sim S(a, v)) \wedge (\sim P(v) \sim S(a, v))$

· 事实表达式的与或树表示

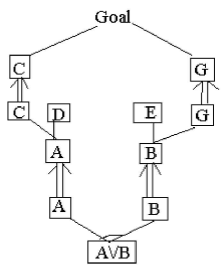


注意：表达式中与或和树中与或相反。
公式的与或图表示有个性质，即由变换该公式得到的子句集可作为此与或图的解图的集合（终止于叶节点）读出；也就是说，**所得到的每个子句是作为解图的各个叶节点上文字的析取。**

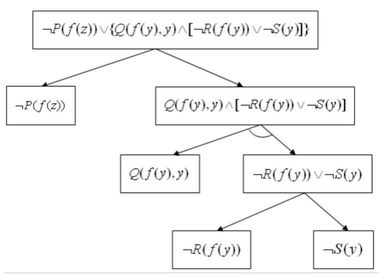
· 正向演绎系统
正向演绎系统使用的规则为 F 规则。形如 L→W，L：文字，W：是公式。规则：$S \to (X \wedge Y) \vee Z$ 可以表示成：



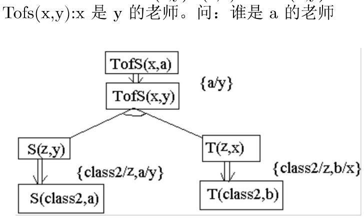
· 推理过程
对于目标公式：限制为文字的析取式。事实 A∨B，目标：C∨G
规则：$A \to C \wedge D$，$B \to E \wedge G$



· 逆向演绎系统（数据库）
B(backward) 规则，W→L，L 为单文字
目标表达式：



推理过程例：事实：F1:S(Class2,a):a 是班级 Class2 中的学生；F2:T(Class2,b):b 是班级 Class2 中的老师。规则：R1：S(z,y)T(z,x)→TofS(x,y)，其中 Tofs(x,y):x 是 y 的老师。问：谁是 a 的老师

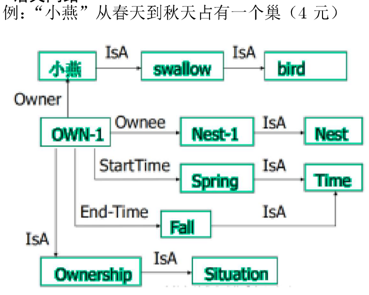


第 3 章 知识表示

· 知识的分类
-事物性知识：事物的概念、类别、性质；-事件性知识：事件的论述、特性、现象、时间、因果；-性能性知识：如何做事情及其做事的技巧（典型的表示方法就是程序）-元知识：关于知识的范围、来源、重要性，力量弱点经验等

· 知识表示方法应具有的性质
-充分性：充分、详尽地表示知识；-正确性：保证推理正确；-有效性：提高推理效率；-灵活性：模块化好，利用知识和灵活运用和增删，修改。；-自然性：表示接近自然语言，易于理解。

· 语义网络
例："小燕"从春天到秋天占有一个巢（4 元）



第 4 章 线性回归

线性回归试图学得 $h(x_i) = wx_i + b$, 使得 $h(x) \approx y_i$
最小二乘法： 最小化均方误差（MSE）：

$$MSE = \frac{1}{n} \sum (h(x_i) - y_i)^2$$

参数估计值：

$$\hat{w} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2} = \frac{\sum_{i=1}^n (x_i - \bar{x}) y_i}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

$$\hat{b} = \bar{y} - \hat{w}\bar{x}$$

似然函数：

$$L(w, b; x, y) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(y_i - wx_i - b)^2}{2\sigma^2})$$

抽样分布（方差已知）：

$$\frac{\hat{w} - w}{\sqrt{\sigma^2 / S_{xx}}} \sim N(0, 1)$$

抽样分布（方差未知）：

$$\frac{\hat{w} - w}{\sqrt{S^2 / S_{xx}}} \sim T_{n-2}$$

确定性系数：

$$r^2 = \frac{\sum_{i=1}^n (\hat{y}_i - \bar{y})^2}{\sum_{i=1}^n (y_i - \bar{y})^2} = \frac{S_{xy}^2}{S_{xx} S_{yy}}$$

· 多元线性回归：

$$w = (w_0, w_1, ..., w_n)^T, X = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1m} \\ 1 & x_{21} & \cdots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{nm} \end{bmatrix}$$

$$y = (y_0, y_1, ..., y_n)^T$$

则 w 估计值为：$w^* = (X^T X)^{-1} X^T y$

· 过拟合与欠拟合： $MSE = Var\ Y_0 + (Bias\ Y_0)^2$，简单模型偏差大，方差小，容易欠拟合，复杂模型方差小，偏差大，容易过拟合。一个粗略的启发是，数据点的数量不应该小于模型的可调节参数的数量的若干倍（比如 5 或 10）。
· 正则化： 加入惩罚项，模型修改为
$\min(y - Xw)^T(y - Xw) + \lambda f(w)$(如$f(w) = \|w\|_2$)

· 模型评估方法
-留出法：直接将数据集 D 划分为两个互斥的集合，其中一个集合作为训练集 S，另一个作为测试集 T，即 $D = S \cup T, S \cap T = \emptyset$。在 S 上训练出模型后，用 T 来评估其测试误差，作为对泛化误差的估计。
-随机抽样（蒙特卡洛方法）将留出法进行多次，每次随机划分训练集和测试集。进行多次单独的模型训练和验证，最后将这些验证结果取平均值，作为此模型的验证误差。与单次验证的留出法相比，这

种方法可以更好地衡量模型的性能。与 k 折交叉验证相比，这种方法能够更好地控制模型训练和验证的次数，以及训练集和验证集的比例。缺点是有些观测值可能从未被选入验证子样本，而有些观测值可能不止一次被选中。（偏差大，方差小）

-交叉验证法：首先将数据集 D 划分为 k 个大小相似的互斥子集。每个子集 Di 都尽可能保持数据分布的一致性，即从 D 中通过分层采样得到。然后，每次用其中 k-1 个子集的并集作为训练集，余下的那个子集作为测试集；这样就可以获得 k 组训练/测试集，从而可进行 k 次训练和测试，最终返回的是这 k 个测试结果的均值。显然，交叉验证法评估结果的稳定性和保真性在很大程度上取决于 k 的取值，为强调这一点，通常把交叉验证法称为“k 折交叉验证”。k 最常用的取值是 10，此时称为 10 折交叉验证，其他常用的 k 值有 5、20 等。

第 5 章 Logistic 回归

· 模型：

$$\pi(x) = p(y=1|x) = \frac{e^{wx+b}}{1+e^{wx+b}} = \frac{1}{1+e^{-(wx+b)}}$$

· 损失函数（交叉熵）：

$$J(w,b) = -\frac{1}{n}\sum_{i=1}^{n}(y_i \log\pi(x_i) + (1-y_i)\log(1-\pi(x_i)))$$

反向传播过程：

$$v = \frac{1}{1-e^{-z}}$$

$$J = y\log v + (1-y)\log(1-v)$$

$$\frac{dJ}{dv} = \frac{y-v}{v(1-v)} \qquad \frac{dv}{dz} = v(1-v)$$

$$\frac{\partial z}{\partial b} = 1 \qquad \frac{\partial z}{\partial w} = x$$

因此

$$\frac{dJ}{db} = y - v \qquad \frac{dJ}{dw} = (y-v)x$$

注意：实际计算中取

$$J = -\frac{1}{n}\sum(y\log v + (1-y)\log(1-v))$$

梯度下降法更新：

$$w = w - \alpha(\frac{dJ}{dw}) \qquad b = b - \alpha(\frac{dJ}{db})$$

· 混淆矩阵

为衡量一个模型的好坏，需要给定一个测试集，用模型对测试集中的每一个样本进行预测，并根据预测结果计算评价分数。混淆矩阵是评判模型结果的指标，多用于判断分类器的优劣。

Binary Classification		Truth	
		Positive	Negative
Decision	Claim Positive	True Positive	False Positive
	Claim Negative	False Negative	True Negative

TP：样本的真实类别是正例，并且模型预测的结果也是正例。TN：样本的真实类别是负例，并且模型将其预测成为负例。FP：样本的真实类别是负例，但是模型将其预测成为正例。FN：样本的真实类别是正例，但是模型将其预测成为负例。

True Positive Rate (TPR) = TP / (TP + FN)

False Positive Rate (FPR) = FP / (TN + FP)

True Negative Rate (TNR) = TN / (TN + FP)

False Negative Rate (FNR) = FN / (TP + FN)

Sensitivity = TPR; 1-Sensitivity = FNR

Specificity = TNR; 1-Specificity = FPR

$$准确率 Accuracy = \frac{TP+TN}{TP+TN+FN+TN}$$

精度 Precision (P) = TP / (TP + FP)

召回率 Recall (R) = TP / (TP + FN)

$$F1-score = \frac{2TP}{2TP+FP+FN} = \frac{2PR}{P+R}$$

Balanced error rate:

$$BER = \frac{1}{2} * [\frac{FP}{FP+TN} + \frac{FN}{FN+TP}]$$

Matthew’s correlation coefficient: MCC=

$$\frac{(TP*TN-FP*FN)}{\sqrt{[(TP+FP)(FP+TN)(TN+FN)(FN+TP)]}}$$

第 6 章 神经网络

· 卷积运算的方式：

（1）Full padding 方式卷积核与图像刚相交时开始做卷积，白色部分填 0。

（2）Same padding 方式进行填充，允许卷积核超出原始图像边界，卷积结果与原图大小一致。

（3）Valid padding 方式不进行任何处理，不允许卷积核超出原始图像边界，卷积结果与原图大小不一致。当卷积核全部在图像里面时，开始做卷积。

· 卷积神经网络

-卷积层：卷积层的功能是对输入数据进行特征提取，其内部包含多个卷积核，组成卷积核的每个元素都对应一个权重系数和一个偏差量，类似于一个前馈神经网络的神经元，在卷积神经网络中激活函数常用 ReLU。每个卷积核都具有第 1 节所描述的卷积的特征。三个超参数控制卷积层输出体积的大小：深度、步长和零填充。深度其实指的就是卷积核的个数，有时又称通道数。步长表示每做完一次卷积运算，卷积核移动的像素数。零填充的大小是第三个超参数。当输入数据为 N*N、卷积核大小为 m*m、步长为 S 以及零填充大小为 P（填充后图像变为 (N+P)*(N+P)）时，输出数据大小为 K*K，计算公式如下：K=(N-m+2P)/S+1. 因此，当该层有 D 个卷积核时，其输出大小应为 K*K*D。

-池化层：池化将特征图中单个点的结果替换为其相邻区域的统计特征，可以降低图像维度，减少参数数量，采用的基本形式是降采样。常采用的池化函数包括最大值池化和平均值池化。池化步长可以控制输出图像的大小。

**** 神经网络要是过拟合了怎么办？**

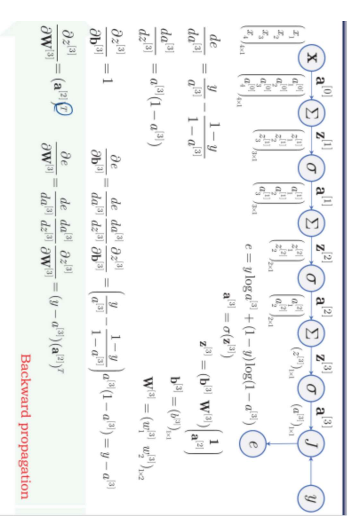
-Dropout　　随机扔掉一些节点

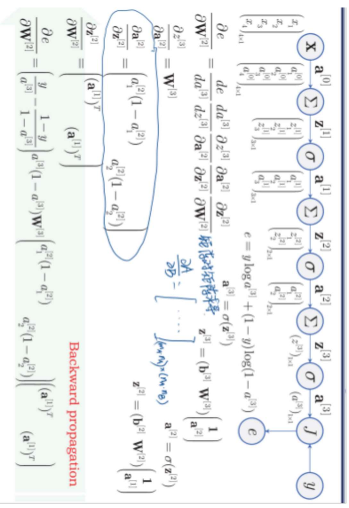
-DropConnect　　随机扔掉一些连接

-Stochastic pooling　　随机选择池化值

-Artificial data　　训练数据重采样

· 多层神经网络反向传播过程





第 7 章 监督学习

· K-Means 聚类

（1）初始化：随机选择中心点 $\mu_1 \sim \mu_k$

（2）为每个样例制定一个类别：

$$y_i = \arg\min \|x_i - \mu_c\|$$

（3）根据指定的类别更新中心点：

$$\mu_k = \frac{1}{n}\sum_{y_i=k} x_i$$

评价：（使用带标签数据）聚类结果好：分类准确率高；不好：准确率低。（使用不带标签数据）聚类结果好：组内距离小、组间距离大；不好：组内距离大、组间距离小。

-平均组内距离

$$a(i) = \frac{1}{|C_i - 1|}\sum_{j\in C_i, i\neq j} d(i,j)$$

-平均组间距离

$$b(i) = \min_{k\neq i}\frac{1}{|C_k|}\sum_{j\in C_k} d(i,j)$$

· 数据的投影

$X = (x_1,...,x_n)T$（$n\times m$ 维）即 X 矩阵的每一行代表一个样本

W 矩阵（$m\times d$ 维）的每一列代表一个正交基向量。

T 矩阵（$n\times d$ 维）的每一行代表坐标变换后的一个样本

注意：样本是去均值化的；正交基向量满足 $w_i^T w_i = 1$, $w_i^T w_j = 0$

· 主成分分析： $X * W = T$

如何投影到一维？最大化类间方差

$$var(Xw) = (Xw)^T(Xw) = w^T X^T Xw$$

$$L(w,\lambda) = w^T X^T Xw + \lambda(1 - w^T w)$$

$$\Longrightarrow 2 * X^T Xw - 2\lambda w = 0$$

因此 w 是 $X^T X$ 关于 λ 的特征向量，最大化方差就是最大化 $X^T X$ 的特征值

投影到第二维：w_2 是 $X^T X$ 第二大特征值对应的特征向量

矩阵 W：按照 $X^T X$ 特征值从大到小排列对应的特征向量

**** 奇异值分解的低秩近似求解矩阵 W：**

将 X 进行奇异值分解 $X = U\sum V^T$，若目标维数为 k，则取 U 的前 k 列为 U'，$\sum$ 左上角 $k\times k$ 小块 $\sum'$，V 的前 k 列 V'，则 $W = V'$，$T = U'\sum'$。

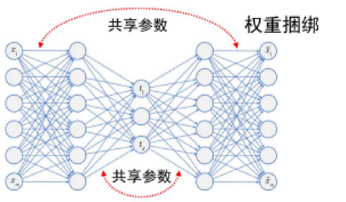
U 的列是 XX^T 的特征向量

V 的列是 $X^T X$ 的特征向量

$\sum$ 的对角线是 XX^T 特征值的平方根

$\sum$ 的对角线是 $X^T X$ 特征值的平方根

· 自编码器： 是一种以无监督方式学习有效数据编码的人工神经网络。自编码器 encoder 部分的目的是学习原始数据的表示，通常用于降维。decoder 部分的目的是尝试从 encoding 中生成尽可能接近其原始输入的表示。



第 8 章 强化学习

· 马尔可夫性： 未来独立与过去，只与现在的状态有关

· 状态转移矩阵 P： 从行转移到列。

$$p_{ss'} = P(S_{t+1} = s'|S_t = s)$$

状态转移：

$$p^{(k+1)} = P * p^{(k)}$$

· 马尔可夫回报过程

状态期望回报

$$r = (r_1, ..., r_n)$$

$$r_s = E(R_{t+1}|S_t = s)$$

累积回报：$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ...$

（为什么要折现？数学上的要求：级数收敛；模型上的要求：将来的不确定性因素大）

状态价值：$v_s = E(G_t|S_t = s)$，可分解成两部分：当前状态的即时回报、后续状态价值的折现。

$$v(s) = r_s + \gamma\sum_{s'\in S} p_{ss'} v(s')$$

· 贝尔曼期望方程：

$$v = r + \gamma Pv \longrightarrow v = (I - Pv)^{-1} r$$

· 行动期望回报

$$r_s^a = E[R_{t+1}|S_t = s, A_t = a]$$

$$\pi(a|s) = P(A_t = a|S_t = s)$$

行动价值函数：（行动价值是后续状态的加权和）

$$q_\pi(s,a) = E(G_t|S_t = s, A_t = a) = r_s^a + \gamma\sum_{s'\in S} p_{ss'}^a v_\pi(s')$$

状态价值函数：（状态价值是同时刻的行动价值加权和）

$$v_\pi(s) = \sum_{a\in A}\pi(a|s) q_\pi(s,a)$$

行动价值的贝尔曼期望方程：

$$q_\pi(s,a) = r_s^a + \gamma_{s'\in S} p_{ss'}^a \sum_{a'\in A}\pi(a'|s') q_\pi(s',a')$$

状态价值的贝尔曼期望方程：

$$v_\pi(s) = \sum_{a\in A}\pi(a|s)(r_s^a + \gamma\sum_{s'\in S} p_{ss'}^a v_\pi(s'))$$

$$v_\pi = r^\pi + \gamma P^\pi v_\pi \Longrightarrow v_\pi = (I - \gamma P^\pi)^{-1} r^\pi$$

· 策略评价： 问题：评价给定策略 π；算法：迭代应用贝尔曼期望方程计算状态价值；过程：生成序列 $v_1 \to v_2 \to ... \to v_\pi$，公式：（当迭代次数足够多时，与矩阵求逆结果相同）

$$v^{(k+1)} = r^\pi + \gamma P^\pi v^{(k)}$$

策略的好坏用状态价值评价：

$$\pi \geq \pi' : v_\pi(s) \geq v_{\pi'}(s), \forall s$$

最优策略：$\pi^* \geq \pi' : v_{\pi^*}(s) \geq v_{\pi'}(s), \forall s \forall \pi'$

最优行动价值：即最优策略下的行动价值（从表达式看出来）

· 策略改进：

$$\pi'(a|s) = \begin{cases} 1 & if\ a = \arg\max_{a\in A} q_\pi(s,a) \\ 0 & otherwise \end{cases}$$

改进的策略一定不比原来的策略差，证明：

$$\begin{aligned} v_\pi(s) &\leqslant q_\pi(s,\pi'(s)) \\ &= \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = \pi'(s)] \\ &= \mathbb{E}_{\pi'}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \\ &\leqslant \mathbb{E}_{\pi'}[R_{t+1} + \gamma q_\pi(S_{t+1},\pi'(S_{t+1})) \mid S_t = s] \\ &= \mathbb{E}_{\pi'}[R_{t+1} + \gamma\mathbb{E}_{\pi'}[R_{t+2} + \gamma v_\pi(S_{t+2})|S_{t+1}] \mid S_t = s] \\ &= \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 v_\pi(S_{t+2}) \mid S_t = s] \\ &\leqslant \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 v_\pi(S_{t+3}) \mid S_t = s] \\ &\ \vdots \\ &\leqslant \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \mid S_t = s] \\ &= v_{\pi'}(s). \end{aligned}$$

· 策略迭代： 交替进行策略评估和策略改进。问题：策略评价计算量很大（需要迭代至收敛），事实上，可以在每次策略评价后进行策略改进。

· 价值迭代：

$$v_{k+1}(s) = \max_a E[R_{t+1} + \gamma v_k(S_{t+1})|S_t = s, A_t = a]$$

$$\max_a\sum_{s',r} p(s',r|s,a)[r + \gamma v_k(s')]$$

行动价值的贝尔曼最优方程：

$$q^*(s,a) = r_s^a + \gamma\sum_{s'\in S}\max_{a'} q^*(s',a')$$

状态价值的贝尔曼最优方程：

$$v^*(s) = \max_a(r_s^a + \gamma\sum_{s'\in S} p_{ss'}^a v^*(s'))$$

· 蒙特卡洛预测

- 首次访问蒙特卡洛
- 根据待评价策略产生观测片段

$$S_1, A_1, R_2, \ldots, S_t, A_t, R_{t+1}, \ldots, S_T, A_T, R_{T+1}$$

$$S_t = s \quad 检查之前是不是出现过$$

- 计算 $G_i = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-t} R_{T+1}$
- 重复 n 次，得到 $G_1, G_2, \ldots, G_n$
- 估计状态价值

$$V(S_t) = \frac{1}{n}\sum_{i=1}^{n} G_i$$

每次访问蒙特卡洛，只要观测片段中出现目标状态，不论是不是出现在最开始，都用来计算平均值，操作上，不检查之前是不是出现过，其他与首次访问相同。

· 定步长蒙特卡洛预测：

$$V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$$

增量式蒙特卡洛预测：

$$V(S_t) \leftarrow V(S_t) + \frac{1}{k}(G_t - V(S_t))$$

· 蒙特卡洛策略迭代： 贪心策略

问题：一旦陷入不好的策略就无法摆脱。改进：ε-贪心策略

$$\begin{cases} 1 - \varepsilon + \dfrac{\varepsilon}{m} & if\ a = \arg\max_{a\in A} Q(s,a) \\ \dfrac{\varepsilon}{m} & otherwise \end{cases}$$

· 动态规划和蒙特卡洛的区别

	动态规划	蒙特卡洛
模型	有模型	无模型
采样	不采样	采样
自举	自举	不自举
终止	无需终止状态	需要终止状态
计算	计算量与状态数相关	与状态数不相关

· 时序差分：

$$V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$

$$Q(S_t,A_t) \leftarrow Q(S_t,A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1},A_{t+1}) - Q(S_t,A_t))$$

特点：从部分序列学习；可用于无终止状态的决策过程；在线学习；仅需获取下一状态的即时回报；马尔科夫环境更有效。

· SARSA 算法：

行动策略选择：ε-贪心算法；

目标策略选择：ε-贪心算法

· Q-Learning 算法：

行动策略选择：ε-贪心算法；

目标策略选择：贪心算法

· 期望 SARSA 算法：

行动策略选择：ε-贪心算法；

目标策略选择：行动的期望

** 具体方法：根据行为策略从 S_t 产生 A_t，获得回报 R_{t+1} 和下一状态 S_{t+1}；根据目标策略从 S_{t+1} 产生 A_{t+1}，计算 (S_{t+1}, A_{t+1}) 的行动价值；更新 (S_t, A_t) 的行动价值。

第 9 章 基于函数逼近的同轨策略预测

· 状态价值近似：

(1) 特征提取：多项式基函数

$$x(s) = (x_1(s), x_2(s), \cdots, x_m(s))^T$$

(2) 函数组：线性函数

$$\hat{v}(s|x,w) = w^T x$$

(3) 优化准则：均方误差最小化

$$\min\sum_{i=1}^{n}(v_\pi(s_i) - \hat{v}(s_i|x,w))^2$$

(4) 优化方法：随机梯度下降

$$w^{(new)} \leftarrow w + \alpha(v_\pi(s) - \hat{v}(s|x,w))x$$

目标如何产生？$v_\pi(s)$

-蒙特卡洛：采样长期回报

$$w^{(new)} \leftarrow w + \alpha(G_t - \hat{v}(s|x,w))x$$

-时序差分：自举下一状态

$$w^{(new)} \leftarrow w + \alpha(R_{t+1} + \gamma\hat{v}(S_{t+1}|x,w) - \hat{v}(s|x,w))x$$

· 行动价值近似

(1) 特征提取：多项式基函数

$$x(s,a) = (x_1(s,a), x_2(s,a), \cdots, x_m(s,a))^T$$

(2) 函数组：线性函数

$$\hat{q}(s,a|x,w) = w^T x$$

(3) 优化准则：均方误差最小化

$$\min\sum_{i=1}^{n}(q_\pi(s_i,a_i) - \hat{q}(s_i,a_i|x,w))^2$$

(4) 优化方法：随机梯度下降

$$w^{(new)} \leftarrow w + \alpha(q_\pi(s,a) - \hat{q}(s,a|x,w))x$$

目标如何产生？$v_\pi(s)$

-蒙特卡洛：采样长期回报

$$w^{(new)} \leftarrow w + \alpha(G_t - \hat{q}(s,a|x,w))x$$

-时序差分：自举下一状态

$$w^{(new)} \leftarrow w + \alpha(r + \gamma\hat{q}(s',a'|x,w) - \hat{q}(s,a|x,w))x$$

· 基于行动价值近似的控制

策略迭代：交替进行策略评价和策略改进

策略评价：增量式行动价值近似

策略改进：ε-贪心

· 批量式价值近似

状态价值的蒙特卡洛经验回放：$w = (X^T X)^{-1} X^T g$

状态价值的时序差分经验回放：

$$w = (X^T(X - \gamma X'))^{-1} X^T r$$