

人工智能课程报告

-----基于规则的梭哈牌专家系统

姓名：朱为开

学号：11300240041

2015.01.05

摘要前言：

本次课程设计主要是构件了梭哈游戏专家系统。主要功能有判断牌类型，梭哈牌比较大小以及判断自己获胜的概率，以及简单的推荐系统（本局可以押注还是直接弃，还是果断上大注之类）。本次设计采用 C/C++编写，由于本次只是做一个简单的试验性，故没有制作图形界面。

问题描述：

梭哈游戏是一种很常见的扑克牌游戏，适合 2~5 人游戏，本次简化了的游戏规则如下：

一副牌（除去大小鬼），两个人玩。依次发给每个人 5 张牌（这里称为“一套”），其中的第 1 张牌对方看不到。发完所有的牌之后，双方把第 1 张牌翻起，再比较大小，牌大的一方为赢家。

“一套”牌的大小分 10 个等级，若双方等级不同，等级高的一方为赢家：

等级	名称	描述	样例
1	大同花顺	A, K, Q, J, 10 且同花色	10S JS QS KS AS
2	小同花顺	5 张同花色并且顺位排列	4D 5D 6D 7D 8D
3	铁支	四张牌都是一样的数字	7S 7C 7D 7H 8D
4	葫芦	三张同数字以及一对同数字	3D 3S 3C KS KH
5	同花	五张牌均是同样花色	3H 7H 10H QH AH
6	顺子	5 张牌呈顺位排列	5C 6D 7C 8H 9S
7	三条	三张牌同一数字	JH JS JD 10S 5C
8	2 对	两对两张牌数字一样	5H 5S 9C 9S AC
9	1 对	一对两张牌数字一样	AS AD JC 10D QS
10	“普通牌”	不属于上面的任一等级	2S 3C 8D 10C 5H

注意：在本游戏中，“A2345”、“JQKA2”等同时含有“A”和“2”的一套牌不算顺子；两张牌面数字相同、花色不同的牌不分大小；牌面数字从大到小依次是 A、K、Q、J、10、9、8、7、6、5、4、3、2。

若双方等级相同：

- 假如双方都拿到小同花顺、顺子或是同花，我们就以数字较大的一方为赢家，若是彼此又是同数字，就再比数字第二大的牌，以此类推。若所有牌一样（指牌面的数字），则为平局；

- 假如双方的牌都是葫芦或三条，就以双方的三条的数字来做胜负之评断；

- 假若双方都是两对，就以各方较大的一对比数字大小，若结果是平手，就再以第二对来分胜负。若第二对依旧平手，则看最后一张单牌。若单牌也一样，则平局；

- 假若双方都是只有一对，就以一对中数字较大的一方为赢家，如果相同，则比较双方单牌中最大的一张；若平手再比较下一张，依此类推；

- 如果双方的牌都没有对、顺子、同花，那就以单张数字较大的牌分输赢，若双方平手就再比较下一张，以此类推。若所有牌一样，则为平局。

由于对方有一张牌是暗置的，用 XX 代替，要求实现功能如下：判断自己牌类型（如是同花顺还是葫芦等），计算自己赢对方可能性并以此推荐行动（实战

中），判断自己牌与对方牌大小（亮牌后）

项目设计：

常见的专家系统由以下 5 个部分组成：知识库，数据库，推理引擎，解释设备，用户界面。数据库即后续我输入的自己的手牌以及（需要时）对手的四张手牌，知识库即一些基本规则。推理引擎以及解释设备就是将数据转换成结果的中间件。具体如下：

首先为了方便，我合并了上述规则中的大同花顺和小同花顺，在代码中将 A 当做 14，K 当做 13，Q 当做 12，J 当做 11 来计算，方便比较大小（不管什么比较 A 总归比 K 还要打，2 最小，另外这样做可以合并大同花顺和小同花顺的比较，在得出牌类型时，对同花顺类型做特殊判断即可。（A2345 这种不可以也是采用这种设计一个原因）。另外代码会事先对牌按大小进行排序（2~A）

规则：

Rule1~9 为判断牌类型规则，共 9 种（合并大小同花顺），后面接的值记为牌优先级。补充概念分类，在比较同类型手牌时，我们将其分成 4 类：

1, 2 为类型（1），4, 9 为类型（2），7, 8 为类型（3），3, 5, 6 为类型（4）

Rule10~Rule15 为比较大小规则，由于后续单个规则比较负责，描述相对繁琐，具体看代码。

Rule16~Rule18 为简单的基于概率的推荐行动设计。

Rule1:

IF 颜色均相同 and 是相邻连续的

THEN 是同花顺 1

Rule2:

IF 是相邻连续的 and 颜色不全相同

THEN 是顺子 2

Rule3:

IF 手牌中有某个点数出现了 4 次

THEN 是铁支 3

Rule4:

IF 颜色相同 and 不是相邻连续的

THEN 是同花 4

Rule5:

IF 手牌中 A 点数出现 3 次 and B 点数出现 2 次（A 不等于 B）

THEN 是葫芦 5

Rule6:

IF 手牌中有某个点数出现了 3 次 and 另外两张牌点数不同

THEN 是三条 6

Rule7:

IF 手牌中有两个点数出现了 2 次

THEN 是 2 对 7

Rule8:

IF 手牌中有一组点数出现 2 次

THEN 是 1 对 8

Rule9:
 IF 不满足 Rule1~Rule8
 THEN 是普通牌 9

Rule10:
 IF A 牌优先级< B 牌优先级
 THEN A 胜 B

Rule11:
 IF A 牌优先级>B 牌优先级
 THEN B 胜 A

Rule12:
 IF A 牌优先级==B 牌优先级 and 属于类型 (1)
 THEN 手牌中点数最大的进行比较, 较大者获胜, 相同平

Rule13:
 IF A 牌优先级==B 牌优先级 and 属于类型 (2)
 THEN 从大到小开始比较手牌中点数, 直到出现大小不同, 大者获胜, 一直相同平

Rule14:
 IF A 牌优先级==B 牌优先级 and 属于类型 (3)
 THEN 都是对子型, 先比较对子大小, 再比较单牌大小, 大者获胜, 直到相同平

Rule15:
 IF A 牌优先级==B 牌优先级 and 属于类型 (4)
 THEN 比较出现次数最多的点数大小, 再比较剩余牌大小, 大者获胜, 直到相同平

Rule16:
 IF 我获胜的概率>0.7
 THEN 大胆押注

Rule17:
 IF 我获胜的概率<0.3
 THEN 选择放弃

Rule18:
 IF 我获胜的概率>=0.3 and 我获胜的概率<=0.7
 THEN 随意押注或者放弃

具体部分代码设计:

首先对牌做预处理, 即按点数拍好序, 并且统计手牌各点数出现次数, 没出现置为 0.

```

3 struct poke {
4     int num;int color;
5 };
6

```

Struct poke 为牌类型, num 为点数, 设置 J 为 11, Q 为 12, K 为 13, A 为 14, 方便比较, color C 为 1, D 为 2, S 为 3, H 为 4.

```

125 int calc() {
126     int i;
127     memset(h,0,sizeof(h));
128     for (i = 1;i <= 5;i++)
129         h[bak[i].num]++;
130     if (csame()) {
131         if (sorted())
132             return 1;
133         else
134             return 4;
135     }
136     else {
137         if (sorted())
138             return 5;
139     }
140     int p = 0 , sum = 0 , sec = 0;
141     for (i = 2;i <= 14;i++) {
142         if (h[i] > sum) { sum = h[i];p = i; }
143         if (sum == 4) return 2;
144     }
145     for (i = 2;i <= 14;i++)
146         if (i == p) continue;
147         else if (h[i] > sec) sec = h[i];
148     if (sum == 3) {
149         if (sec == 2) return 3;
150         else return 6;
151     }
152     else {
153         if (sum == 2) {
154             if (sec == 2) return 7;
155             else return 8;
156         }
157     }
158     return 9;
159 }

```

Int calc()函数为计算牌优先级，将要计算的牌赋给 bak 数组即可，h 数组统计其各点数出现次数

其中 csame 函数判断是否颜色全部相同，sorted 函数判断是否为连续有序，后续计算出牌中点数出现次数最多的两个，出现次数存为 sum 和 sec，若 sum=4 则为 2 类型，若 sum=3，sec=2 则为 3，sum=3，sec=1 则为 6，sum=2，sec=2 则为 7，否则为 8，若 sum = 1 则为普通牌

具体如下：

```

100 bool csame() {
101     int i;
102     for (i = 1; i <= 4; i++)
103         if (bak[i].color != bak[i+1].color) return false;
104     return true;
105 }

118 bool sorted() {
119     int i;
120     for (i = 1; i <= 4; i++)
121         if (bak[i].num != bak[i+1].num+1) return false;
122     return true;
123 }

```

下面是比较部分：

```

237         if (m1 < y1) return 1;
238     else {
239         if (m1 == y1)
240             return check();
241         else
242             return -1;
243     }

```

Check 部分为同类型比较，具体如下：

```

15 bool check()
16 {
17     int i , type , top , s1 , s2;
18     bool ord1[6] , ord2[6];
19     int p1 = 0 , p2 = 0 , q1 = 0 , q2 = 0;
20     if ((m1 == 1) || (m1 == 2)) type = 1;
21     else {
22         if ((m1 == 4) || (m1 == 9)) type = 2;
23         else if ((m1 == 7) || (m1 == 8)) type = 3;
24         else type = 4;
25     }
26     switch (type) {
27     case 1 : {
28         if (b[1].num > bak[1].num) return true;
29         else return false;
30         break;
31     }
32     case 2 : {
33         top = 1;
34         while (b[top].num == bak[top].num) top++;
35         if (b[top].num > bak[top].num) return true;
36         else return false;
37         break;
38     }

```



```

39 case 3 : {
40     memset(ord1,true,sizeof(ord1));
41     memset(ord2,true,sizeof(ord2));
42     p1 = 1;
43     while (b[p1].num != b[p1+1].num) { p1++;ord1[p1] = false; }
44     p2 = 1;
45     while (bak[p2].num != bak[p2+1].num) { p2++;ord2[p2] = false; }
46     if (m1 == 7) {
47         q1 = p1+1;
48         while (b[q1].num != b[q1+1].num) { q1++;ord1[q1] = false; }
49         q2 = p2+1;
50         while (bak[q2].num != bak[q2+1].num) { q2++;ord2[q2] = false; }
51         if (b[p1].num > bak[p2].num) return true;
52         else {
53             if (b[p1].num == bak[p2].num) {
54                 if (b[q1].num > bak[q2].num) return true;
55                 else {
56                     for (i = 1;i <= 5;i++)
57                         if (!ord1[i]) break;
58                     s1 = b[i].num;
59                     for (i = 1;i <= 5;i++)
60                         if (!ord2[i]) break;
61                     s2 = bak[i].num;
62                     if (s1 > s2) return true;
63                     else return false;
64                 }
65             }
66         }
67     }
68     else
69     {
70         if (b[p1].num > bak[p2].num) return true;
71         else {
72             if (b[p1].num == bak[p2].num) {
73                 top = 1;
74                 while (b[top].num == bak[top].num) top++;
75                 if (b[top].num > bak[top].num) return true;
76                 else return false;
77             }
78             else return false;
79         }
80     }
81     break;
82 }
83 case 4 : {
84     s1 = 0;
85     for (i = 2;i <= 14;i++)
86         if (h[i] >= 3) s1 = i;

```

```

87     memset(h,0,sizeof(h));
88     for (i = 1;i <= 5;i++)
89         h[b[i].num]++;
90     s2 = 0;
91     for (i = 2;i <= 14;i++)
92         if (h[i] >= 3) s2 = i;
93     if (s2 > s1) return true;
94     else return false;
95     break;
96 }
97 }
98 }

```

上述可以看做是推理引擎部分，解释部分如下：

```

250     init();
251     memset(hash,true,sizeof(hash));
252     translate();
253     win = 0;
254     int i , j;
255     poke tmp;
256     for (i = 1;i <= 5;i++) { bak[i] = me[i];b[i] = bak[i]; }
257     for (i = 1;i <= 4;i++)
258         for (j = i+1;j <= 5;j++)
259             if ((b[i].num < b[j].num)||
260                 ((b[i].num == b[j].num)&&(b[i].color > b[j].color))) {
261                 tmp = b[i];b[i] = b[j];b[j] = tmp;
262             }
263     sort();
264     ml = calc();
265     work();
266     win = win/total;
267     printf("%d\n",ml);
268     printf("%.4f\n",win);
269     if (win > 0.7) printf("You can put All your money!");
270     if (win < 0.3) printf("Give it up");
271     else printf("Try it!Anything!");
272     return 0;

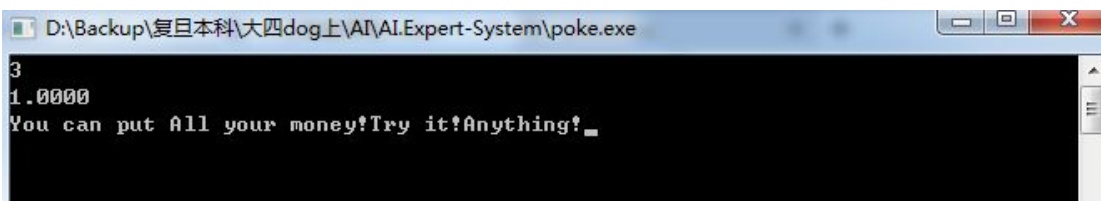
```

Win 为获胜概率，ml 为自己牌的优先级（即类型），由此可以得到我们需要的一些信息，输入格式如下：

XX 3C 5D 4H 6H （别人的牌，XX 代表不知道）

8S 10D 10S 8H 8C（自己的牌）

输出如下：



```

D:\Backup\复旦本科\大四dog上\AI\AI-Expert-System\poke.exe
3
1.0000
You can put All your money!Try it!Anything!_

```


感想与题材由来:

这次做专家系统不像去年，一定程度限定了范围做几何题，本来觉得反而是好事，我可以做我感兴趣的，本来想做动漫或者 nba 相关的，nba 数据都找到了，但突然发现不清楚这个专家系统是用来干啥的，nba 相关，我难道会去问哪个球员什么时候拿多少分这种问题，总感觉这样的专家系统毫无推理过程（即使是基于规则的那种简单的 if else 判断推理，或者简单的 DFS 或者 BFS，这里习惯用前向推理和后向推理），这样的系统总感觉直接交给别人怎么用 sql 就好了，动漫相关则推理部分十分复杂，之间的关系虽然我觉得自己就算不是专家也还可以，但总结出来的规律太难数据化。

这时我陷入了僵局，一直在找话题，后来我借了同学去年的报告，就是做几何题，由于这个挺难的，所以他做的很简单，就是只做全等和相似的证明，而且是非常狭隘的两个确定的三角形之间全等和相似的判断（必须是 ABC 和 DEF 相似或全等），可以认为就是教科书式的教你用公理证明相似全等，做法就是列出一些公理，即 Rule，然后通过 Rule 条件的匹配来做前向推理，然后看是否能找到结论。

然后我想想了，普通的+，-，*，\可以算是规则吗？DFS 或者 BFS 可以认为是对+，-，*，\的前向推理，结论就是值为 24，类别下那么我以前写的 24 点是否可以认为是一个专家系统？然后我觉得这个游戏太 low 了，我想到了自己比较喜欢玩日麻，麻将几向听的判定以及役种的判定涉及了大量的规则（胡牌型大致 3 类，23333，222222，国士无双，役种更是十分丰富），这个做成一个专家系统似乎还是可以，但这个写代码好像一个晚上不好赶出来，然后就想到了扑克，梭哈这种牌型可以看成是麻将役种的简化版，由于可以全部规约成数字比较，不像麻将需要考虑字牌等因素（字牌可以普通用，也可以当和大三元，小三元，大四喜，小四喜，字一色等役种）。

所以最后就决定做这个了，当然如果有时间，还是觉得要把麻将那个好好做个，其实把那个部分做好，就可以做一个简单的麻将游戏或者 app 了。说实话梭哈可以说是上次玩大概已经好几年了，完全称不上高手，麻将相对而言还是个兴趣爱好，可以认为是所谓的“专家”，可以提供一些意见，如果写麻将 AI，想法也比较多，从牌效或者防守等角度，而梭哈只能局限于单纯的比大小了，不过简单也是最终选择做这个原因。