根据蒙特卡洛模拟和对手模型生成一个麻将玩家

水上直紀[1]*, 鶴岡慶雅[2]

摘要

预测对手的移动和隐含状态在不完全信息博弈中是十分重要的。这篇文章描述了一个方法，对对手建模并使用蒙特卡洛模拟，用于生成麻将程序。我们将对手的状态分成三个元素考虑，‘听牌与否’，‘听牌种类’，‘打点’，并使用麻将高级别玩家的游戏数据训练这些元素的预测模型。在蒙特卡洛模拟中对手的移动取决于对对手建模得到的概率分布。我们利用麻将网站‘天凤’计算了生成麻将程序的实力。这个麻将程序得到的r值为1718，显著高于一般玩家。

[1]東京大学大学院 工学系研究科電気系工学専攻 博士卒業　[2]東京大学大学院 工学系研究科 准教授　***译者**: 零之审判 北京大学数学科学学院 学士毕业

1. 介绍

在人工智能研究中，不完全信息博弈的游戏提供了许多具有挑战性的重要问题。因为这些游戏要求玩家在现实中处理普遍的问题，他们要面对着隐含信息和偶然性的挑战。麻将是一个在亚洲流行的不完全信息博弈的传统游戏。各种麻将的形式中，我们只讨论在日本流行最广泛的形式——日本麻将。从AI研究的角度来看，日本麻将是具有挑战的项目，因为(a)对局玩家超过两名，(b)不完全信息博弈，和(c)信息集合数量远超过扑克之类的很多卡牌游戏。

德州扑克的二人版本，作为最流行的扑克形式之一，已经被近似纳什均衡策略的离线计算所解决。[1]然而这类尝试并不总是奏效，在具体游戏中会因为计算纳什均衡策略的耗时而被阻碍。另一种生成不完全信息博弈游戏的电脑玩家的方式是使用对手模型。对对手建模，Van der Kleij[2]根据玩家们的游戏记录得到的游戏风格对其聚类。在游戏进程中手牌级别的分布会更新。[3]在电脑斯卡特牌中，Bruo et al.[4]提出了一个方法，可以利用游戏记录在玩家决定从当前位置移动时估计不确定的假定的空间。这个假定空间包含对手手中不可见卡牌的分布。利用一般人类玩家的数据训练并结合游戏空间中高级别特征的推理，他们的斯卡特牌程序达到了人类的专家水平。

这篇论文的工作中，我们利用对手模型预测对手的三个提炼的元素，被称作‘听牌与否’，‘听牌种类’，‘打点’，以此来生成麻将程序。首先，我们使用高级别人类玩家的游戏数据训练这三个元素的预测模型。然后，基于蒙特卡洛模拟和预测模型生成一个麻将电脑玩家。我们展示了该程序与人类玩家和目前最好程序的对战表现。

这篇论文的框架如下。第二章说明了麻将的基本规则和术语。第三章展示单人麻将打牌程序。第四、五、六章描述了‘听牌与否’，‘听牌种类’，‘打点’的预测模型。第七章说明了如何利用蒙特卡洛模拟与预测模型决定打牌。第八章展示了程序的实验结果。第九章阐述了相关工作。最终结果总结在第十章。

2. 日本麻将的基本规则和术语

这一章说明日本麻将（之后我们均简称麻将）的基本规则和术语。麻将是四人桌游，各玩家开局持点25000，竞争得到最高的点数。一次麻将的对局一般包含四或八轮。玩家在一轮中手中的14张牌达到和牌的条件时，即在该轮获胜，并得到相应的点数。每一轮中的每一巡，玩家从牌山里摸牌，或者鸣走他家切出的牌。当玩家摸到自己获胜所需的牌时，称为自摸，并由其他三家支付点数给赢家。玩家可以在他家切出自己获胜所需的牌（铳牌）时宣告和牌，称为荣和，并由切出铳牌的玩家支付点数给赢家。

麻将由136张牌组成，34种不同的牌、每种4张。其中108张为数牌，数牌中包括三套1到9，分别为万字、筒子和索子。数牌之外的为字牌，没有统一的顺序。字牌由风牌（东、南、西、北）和三元牌（白、发和中）组成。

玩家们追求手牌符合和牌条件，通常指四个面子和一个雀头。面子有两种，三张相同牌组成的刻子和相连的数牌组成的顺子。

我们下面说明一些与研究工作尤其相关的麻将术语。

听牌

一个玩家听牌指的是他的手牌仅需加入一张即符合和牌条件。

立直

一个玩家在听牌且没有任何鸣牌的面子时，可以宣告立直。宣告立直后，除非自摸，玩家必须切出摸上的牌。

弃和

一个玩家弃和指其放弃追求胜利的手牌，而避免切出他家获胜所需的牌。与扑克不同，放弃并不需要宣告，因此弃和在麻将中不是操作而是策略。

3. 单人麻将打牌

这一章简单解释‘单人麻将打牌’程序，其在之后的蒙特卡洛模拟中起到重要作用。首先，我们定义单人麻将，指的是麻将游戏中仅有一名玩家。这名玩家的目标是追求符合和牌条件的手牌。

在我们以前的工作[5]中，使用有监督机器学习构建了单人麻将的电脑玩家程序。任给14张牌的组合，程序可以选择最有可能尽快和牌的移动（切牌）。但总选择这样的切牌在四人麻将时不是好的策略，因为没有进行鸣牌的选择。我们随之在程序中加入了是否鸣牌的判断，其同样基于机器学习。这篇论文中，我们称这个程序选择的移动为‘单人麻将打牌’。

单人麻将打牌程序并不是一个强者，因为它永远不弃和，因此经常切出他家的铳牌。这篇文章中，我们将单人麻将打牌程序与蒙特卡洛模拟选择的打牌结合来处理这个问题，其中蒙特卡洛模拟选择时用到对对手建模的三个预测模型。我们处理问题的主要出发点是在快速和牌与减小放铳损失之间的平衡。后面的章节中我们将详述模型。

4. 听牌与否的预测

这一章阐述如何训练预测对手听牌与否的模型。在麻将中，没有听牌的玩家对他人不会造成威胁。因此对听牌与否的准确预测是生成有力的麻将程序的重要要求。

4.1 训练的游戏数据

预测模型中我们采用的是有监督机器学习方法。我们使用天凤网站[1]的‘凤凰桌’的游戏记录[2]作为训练数据。因为天凤玩家中0.1%的高级别玩家才被允许在凤凰桌对战，因此我们认为这些可以代表高级别人类玩家的对战记录。

在游戏记录的每个界面，我们生成一个二元标签：对手听牌与否，并得到其他玩家视野下的各个牌桌的信息特征组成的向量。共有大约$1.77 * 10^7$个记录界面。

4.2 训练模型

我们使用logistic回归进行预测。一个玩家在某一界面（对战状态）下听牌的可能性计算如下：

$$P(p = waiting) = \frac{1}{1 + exp(-\mathbf{w}^T \mathbf{x}_p)}, \qquad (1)$$

其中$\mathbf{x}_p$为代表对手p相关的信息的特征向量，$\mathbf{w}$为这些特征的权向量。表1展示了模型中所使用的特征。这些特征中有对手切出的牌的种类、副露的面子数量、巡目等等，共有6888个特征。

表 1. 用于预测听牌与否的特征

特征类	特征个数
立直	1
副露面子数和切出的手牌数	5×19=95
副露面子数和巡目	4×19=76
副露面子数和手切[3]次数	5×19=95
副露面子数和刚切出的牌	5×37=185
副露面子和切出的牌种类	136×37=5032
切出宝牌[4]与其种类	34
切出红宝牌[5]	1
切出的牌两两组合	37×37=1369

预测模型的训练是最小化以下的式子：

$$L(\mathbf{w}) = -\sum_{i=1}^{N}(c_i P(\mathbf{X_i}) + (1-c_i)(1-P(\mathbf{X_i})) + \frac{\lambda |\mathbf{w}|^2}{N}, \qquad (2)$$

其中N为训练样本数、$\mathbf{X_i}$为第i个训练样本、c_i为表示对手是否听牌的标签（0或1）、λ为正则化系数，用来避免对训练数据的过拟合。我们设定正则化系数为0.01。

[1] http://tenhou.net/

[2] 游戏记录为20/02/2009到31/12/2013间的牌谱

[3] 指玩家切出手中已有的牌

[4] 赢家手牌中有宝牌时，收入点数根据宝牌数量增加

[5] 我们将部分5的数牌染红、将其作为宝牌

我们使用FOBOS算法[6]训练得到权向量。Adagrad用于权更新[7]。更新算式如下：

$$w_{t+1,i} = w_{t,i} - \frac{\eta g_{t,i}}{\sqrt{1+\sum_{k=1}^{t} g_{k,i}^2}}, \qquad (3)$$

其中$w_{t,i}$代表权向量的第i个元素、η为学习率、t为更新的次数、g为目标函数的（随机）梯度。我们设定学习率η为0.01。

4.3 预测准确率

我们使用AUC计算前面描述的预测模型的准确率用于评价。测试集是游戏记录中随机选出的100个界面（均在不同的对局中）。因为目标对手立直时显然听牌，所以我们不选择这样的界面。

表2展示了评价值。其中高级别玩家指的是一名凤凰打者。结果说明了我们的预测模型基本达到了高级别玩家的能力水平。我们同样展示了不使用某种特征训练得到的预测模型的预测水平，比开始的预测模型稍低。

表 2. 是否听牌的预测评价值

预测玩家	AUC
高级别玩家	0.778
预测模型	0.777
-切出的牌	0.772
-副露面子数量	0.770

5. 听牌种类的预测

这一章阐述如何训练预测对手听牌种类的模型。在麻将中，切出对手铳牌的玩家需要支付所有点数，这是致命的。因此对对手听牌种类的准确预测是生成有力的麻将程序的重要要求。

5.1 训练的游戏数据

我们使用与第四章相同的游戏记录。对所有的牌，我们生成一个二元标签：是否是铳牌，并得到其他玩家视野下的各个牌桌的信息特征组成的向量。共有大约$1.77 * 10^7$个记录界面。

5.2 训练模型

一般来说，一个对手有着一或多个铳牌。因此对每种牌都进行二元分类预测（即是否为铳牌），对此我们构建使用logistic回归的34个预测模型。

表 3. 用于预测听牌种类的特征

特征类	特征个数
每张牌玩家能见数量	5×34=170
安全牌[6]	34
手切牌n次	3×34=102
宝牌	34
巡目和牌的数量	18×37=666
立直宣告牌	37
副露面子和切出的牌种类	136×37=5032
副露面子种类的组合	136×136=18496
切出的牌的组合以及其是否手切	37×37×2×2=5476
切出的牌和手切的牌	37×37=1369

表3展示了模型中所使用的特征。这些特征中有对手切出的牌、每种牌的数量等等，共有31416个特征。

我们对预测模型的训练与之前章节对听牌与否的预测模型的训练一致。目标函数为

$$L(\mathbf{w}) = -\sum_{i=1}^{N}(c_i P(\mathbf{X_i}) + (1-c_i)(1-P(\mathbf{X_i})) + \frac{\lambda|\mathbf{w}|}{N}, \qquad (4)$$

其中N为训练样本数、$\mathbf{X_i}$为第i个训练样本、c_i为表示预测的牌是否是铳牌的标签（0或1）、λ为正则化系数，我们使用l_1正则化使权向量稀疏化。

5.3 预测准确率

预测听牌种类的模型的评价应该基于麻将的规则。我们需要预测全部的铳牌，因为切出其任何一张都会输掉该轮对局。因此我们用如下的方式评价预测模型。将玩家手中的牌按照其为铳牌的可能性从小到大排列其顺位，评价值的计算如下：

$$Evaluation\ value = \sum_{i=1}^{N} \frac{Clear_i}{AT_i - WT_i} \qquad (5)$$

其中N为测试集数量、$Clear_i$为铳牌中最小的顺位、AT_i（All Tiles）为玩家手中牌的种类数、WT_i（Winning Tiles）为其中铳牌的种类数。

我们的测试集是游戏记录中随机选出的100个界面（均在不同的对局中）。表4展示了评价值。其中随机指的是随机切牌的玩家。结果说明了预测模型的表现远好于随机切牌，但差于高级别玩家。

[6] 对手已经切出的牌

表 4. 听牌种类的预测评价值

预测玩家	Evaluation value
高级别玩家	0.744
预测模型	0.676
-副露的面子	0.675
-切牌	0.673
随机	0.502

6. 打点的预测

这一章阐述如何训练预测切出对手铳牌时玩家需要支付的点数。对自摸打点的预测模型训练也是类似的。

6.1 训练的游戏数据

对游戏记录中某玩家获胜的界面，我们生成一个训练集，包括打点和其他玩家视野下的各个牌桌的信息特征组成的向量。共有大约$5.92 * 10^7$个记录界面。

6.2 训练模型

在麻将中，手牌的打点基本上随着番数[7]指数增加。因此我们建立线性模型预测训练集中实际打点的自然对数值。表5展示了模型中所使用的特征。这些特征中有对手副露面子中含有的番数、宝牌数，共有26889个特征。

模型对所给的界面预测HS（Hand Score打点）：

$$HS = \mathbf{w}^T \mathbf{x}, \tag{6}$$

其中$\mathbf{x}$是特征向量，$\mathbf{w}$为权向量。

预测模型的训练是最小化以下的目标函数：

$$L(\mathbf{w}) = -\sum_{i=1}^{N}(HS_i - c_i)^2 + \frac{\lambda |\mathbf{w}|^2}{N}, \tag{7}$$

其中N为训练样本数、HS_i为预测值、c_i为实际点数的自然对数值、λ为正则化系数。使用该模型得到的值作为指数得到预测的打点。我们使用FOBOS算法和Adagrad训练得到权向量。我们设定学习率η为0.01和正则化系数λ为0.01。

6.3 预测准确率

我们使用预测值和实际值的误差平方平均评价模型。我们的测试集是游戏记录中随机选出的100个界面（均在不同的对局中）。我们给人类玩家切牌和赢家是谁的主要信息。表6的评价值说明预测模型甚至强于高级别玩家。

[7]番数指和牌时手牌的价值总和

7. 使用对手模型的蒙特卡洛打牌

这一章阐述我们的程序如何使用这些预测模型和蒙特卡洛模拟决定打牌。之前章节的评价值显示了对对手的手牌的三个元素的预测模型，其表现与高级别玩家不分伯仲。我们使用这些模型估计程序支付的期望值。

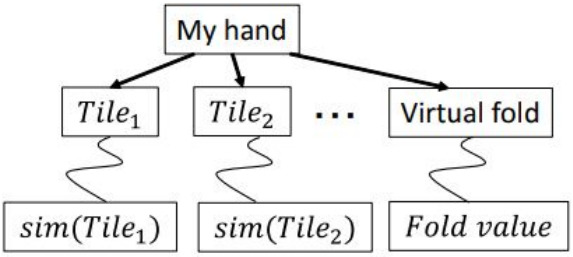

图 1. 蒙特卡洛打牌的概括

我们定义LP（Losing Probability放铳可能性）为对手p已听牌且Tile为其所听的牌的可能性，计算如下：

$$LP(p, Tile) = P(p = waiting) \times P(Tile = winning), \tag{8}$$

其中p为对手、Tile为程序所切的牌。EL（Expected Loss损失期望）为LP乘上HS，计算如下：

$$EL(p, Tile) = LP(p, Tile) \times HS(p, Tile), \tag{9}$$

则程序切出一张牌的收支点数期望计算如下：

$$\begin{aligned} Score = & Sim(Tile) \times \\ & \prod_{p \in opponents} (1 - LP(p, Tile)) - \\ & \sum_{p \in opponents} EL(p, Tile), \end{aligned} \tag{10}$$

其中Sim(Tile)为使用蒙特卡洛模拟计算的和牌收入期望，第二项为程序切出$Tile$的放铳损失期望。该程序对手中的所有牌计算$Score$（Tile），并切出点数最高的牌。我们称之为蒙特卡洛打牌。

现在我们阐述如何实际应用蒙特卡洛模拟计算Sim(Tile)。图2展示了轮到程序和对手切牌时的

[8]与安全牌数字相差为3
[9]断幺指只包含2到8的数牌
[10]指三元牌、玩家的自风和场风牌
[11]只包含一套数牌、字牌
[12]只包含刻子和一个雀头

表 5. 用于预测打点的特征

特征类	特征个数
立直和庄家	2×2=4
副露的番数和可见宝牌	7×8=56
立直或副露面子数和副露的番数和可见宝牌	3×7×8=168
副露面子的种类和副露的番数和可见宝牌	136×7×8=7616
两种副露面子的组合	136×136=18496
副露面子数和副露的番数和可见宝牌	5×7×8=280
立直和切牌是筋牌[8]或全部是断幺牌[9]	3×2×2=12
没有副露役牌[10]和副露役牌、没有副露	3
切牌是宝牌、数字相差1或2、同一套、没有关联	5
手牌可以有断幺和宝牌和可见宝牌	2×2×8=32
手牌可以有一色[11]（混一色或清一色）和副露的番数和宝牌同为该色	5×7×2×2=140
手牌可以有对对[12]和副露的番数和宝牌同为该色	5×7×2=70
副露三元牌数	3
副露风牌数	4

表 6. 打点的预测评价值

预测玩家	误差平方平均
预测模型	0.37
-副露的面子	0.38
-副露的番数	0.38
高级别玩家	0.40

两个流程图。在程序切牌时，程序进行一次单人麻将打牌。在对手切牌时，他们基于一些概率分布和牌或宣告立直。切出的牌的计算则假设玩家从牌山中摸牌并直接切出。我们对每个牌进行相同数量的模拟，且对手的移动和牌山的随机生成是一致的，使得模拟的随机性影响尽可能小。

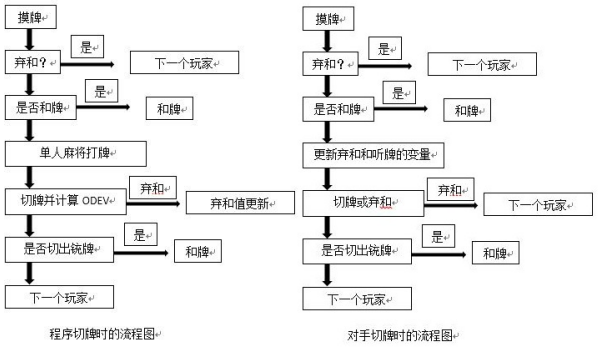

图 2. 每个玩家的流程

7.1 对手切牌

当对手切牌时，不考虑其手牌，假设他们基于一个概率分布决定移动。换句话说，每个对手有两个二元变量代表其是否听牌或弃和。

首先，我们阐述听牌的变量。对手获胜仅当其听牌且从牌山摸到或他家切出所听的牌。听牌变量的初始值设定为第六章预测模型对当前游戏界面得到的预测值。模拟过程中对手玩家从牌山摸牌时，其听牌变量由其他的概率分布决定。我们称之为$WPET$（Waiting Probability at Each Turn每次摸牌的听牌概率），并事前使用单人麻将打牌模型计算。

我们阐述$WPET$的计算方法。首先，我们生成一个牌山。在每一巡，玩家基于单人麻将打牌模型切牌，对手玩家则自模切。当玩家听牌时，游戏结束并记录该巡目。则$WPET$计算如下：

$$WPET_i = \frac{BW_i}{R_i}, \tag{11}$$

其中i为巡目数、BW_i（Becomes Waiting）为该巡目玩家听牌的总次数、R_i（Reach）为游戏进行到该巡目的总次数。

听牌方式有两种，门清听牌和副露听牌。门清时，玩家不能副露。我们进行10^6次游戏来计算$WPET$，图3展示结果。听牌形式基于模拟时事先确定的副露比例，且并在各模拟中不变，该比例为35%，即高级别玩家的平均数据。

接下来阐述弃和的变量。当玩家不弃和时，宣

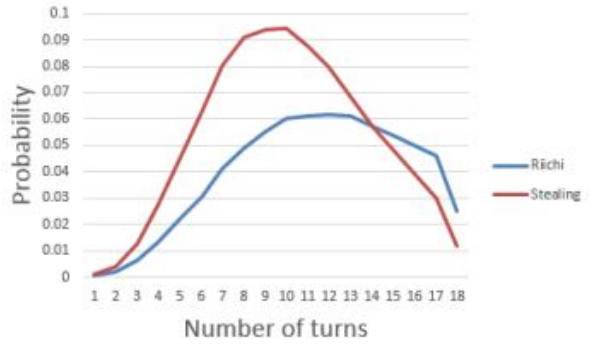

图 3. 各巡目玩家听牌可能性

告立直的玩家能容易获胜，这不是符合实际的模拟。为了解决这个问题，我们让对手玩家决定弃和。

首先，我们说明如何基于概率决定是否弃和。单人麻将打牌和游戏记录里切牌的最大不同即弃和。FP（Folding Probability弃和概率)由游戏记录和单人麻将打牌模型得到，计算如下：

$$FP(situation) = \frac{different_{situation}}{count_{situation}}, \qquad (12)$$

其中$situation$是程序是否听牌和副露或立直玩家数量的组合、$different_{situation}$为单人麻将打牌模型中选择顺位前三的切牌与$situation$下的切牌不一致的数量、$count_{situation}$为$situation$的数量。总情况数量为5739，游戏界面总数量大约为1.28×10^8。

我们说明如何选择弃和。麻将中的弃和与扑克不同，玩家不需要宣告且继续切牌。实际对局中，弃和的玩家通常能避免切出铳牌。为了模仿弃和情形，模拟中弃和的玩家摸到牌后不切出手牌，我们称之为虚拟弃和，其他玩家因此无法荣和弃和玩家。

弃和变量的初始值设定为不弃和，当对手玩家摸牌时决定其更新值。

7.2 程序切牌

我们的程序在模拟中进行单人麻将打牌。然而单人麻将打牌一直进攻而不弃和，导致不合实际的模拟。为了解决这个问题，我们的程序基于$ODEV$（One-Depth Expected Value)决定是否进行虚拟弃和。$ODEV$为搜索程序下一次切牌前游戏树得到的期望值、其不使用蒙特卡洛模拟。使用虚拟弃和可以计算一个期望值（fold value弃和值）和流局[13]可能性（PED）。摸到牌的概率分布和对手的两个二元变量在该值的计算中固定。图4展示了$ODEV$的算法，代码中的变量与游戏树根结点处的一致。

[13]不能在牌山摸牌前没有玩家获胜

Algorithm 1 One-Depth Expected Value

function ODEV
  $p = 1$ ▷ Probability of reaching current states
  $value = 0$
  $PWFW = 0$ ▷ Probability of winning-from-the-wall
  $SWFW = 0$ ▷ Score of winning-from-the-wall
  $SWD = 0$ ▷ Score of winning-by-a-discard about the program
  $P(get\,tiles)$ ▷ Probability of getting the tiles
  foreach $i \in all\,players$ **do** ▷ Opponent players and program
    foreach $j \in all\,kinds\,of\,tiles$ **do**
      $PWFW_i += P(j = winning) \times P(get\,tiles = j)$
      $SWFW_i += HS_{WFW}(i, j) \times P(j = winning) \times P(get\,tiles = j)$
    end for
  end for
  foreach $j \in all\,kinds\,of\,tiles$ **do**
    $SWD += HS(program, j) \times P(j = winning) \times P(get\,tiles = j)$
  end for
  foreach $i \in opponents$ **do**
    $value -= EL(i, Tile)$
  end for
  foreach $i \in opponents$ **do**
    $value -= p \times WAITING(Player_i) \times PWFW_i \times SWFW_i$
    $p \times = (1 - WAITING(Player_i) \times PWFW_i)$
    $value += p \times (1 - FOLD(Player_i)) \times PWFW_{program} \times SWD$
    foreach $k \in all\,players$ **do**
      if $k \neq i$ **then**
        $p \times = (1 - PWFW_k \times (1 - Fold(Player_i)))$
      end if
    end for
  end for
  $value += p \times SWFW_{program}$
  $value -= p \times PED \times fold\,value$
  if $value \leqq 0$ **then**
    return **FOLD**
  else
    return **NOT FOLD**
  end if
end function

function WAITING(Player)
  if $Player.waiting = YES$ **then**
    return **1**
  else
    return **0**
  end if
end function

function FOLD(Player)
  if $Player.Fold = YES$ **then**
    return **1**
  else
    return **0**
  end if
end function

图 4. 算法

7.3 单人麻将打牌和蒙特卡洛打牌的切换

在开放的情形下，蒙特卡洛打牌可能比较糟糕，因为模拟程序的时间限制而且模拟中玩家们需要多次切牌才能和牌。因此该情形单人麻将打牌更合适。我们用如下方式在两种打牌中切换：

$$\begin{cases} One-playerMahjongmoves \\ \quad \textbf{if}(\frac{\sum_{p\in opponents} EL(p,Tile)}{foldvalue} \le \alpha), \\ MonteCarlomoves \\ \quad \textbf{otherwise} \end{cases} \tag{13}$$

其中α为阈值，其调整使得两种打牌结合的程序选择与游戏记录里的一致率最高。

我们使用10000个游戏界面作为运算数据，蒙特卡洛打牌程序往往一秒内得到结果。图5展示了运算结果。Rank n指程序选择顺位前n位的切牌和游戏记录一致的比例，根据运算结果我们设定α为0.2。

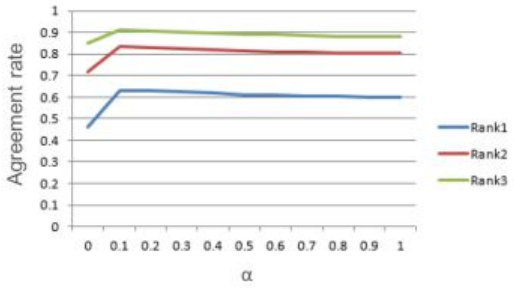

图 5. 各个阈值下的一致率

8. 试验

这一章我们要评价总程序的表现。

8.1 玛塔里酱对战的评价

我们对比程序和玛塔里酱[14]。我们目前所知最强的公开麻将程序即马塔里酱，玛塔里酱由事先组合的静态局面生成评价函数。

对战采用东风局。程序对战三个玛塔里酱，一秒内计算切牌。我们使用*match game type*模式，指一个由随机数生成的复制对局，有相同的牌山和手牌。程序和玛塔里酱分别在复制对局里的对战结果进行对比，这样的对局进行1000次，使用平均顺位进行评价。

表7展示了结果。我们使用bootstrap resampling[8]计算平均顺位的置信区间（p-value≤0.05）。计算t值可知我们的程序和先前的工作[5]得到的平均顺位差异显著（p-value=0.01）。与玛塔里酱的对比也胜出了，但差异并不统计显著（p-value=0.29）。

表8展示了和率和放铳率，结果显示我们程序的表现强于玛塔里酱。

表 7. 顺位分布

	1st	2nd	3rd	4th	平均顺位
我们的程序	0.252	0.256	0.247	0.245	2.48±0.07
玛塔里酱	0.248	0.247	0.250	0.255	2.51±0.07
Mizukami et al.[5]	0.243	0.226	0.222	0.309	2.59±0.07

表 8. 和率与铳率

	和率	铳率
我们的程序	0.222	0.125
玛塔里酱	0.200	0.122
Mizukami et al.[5]	0.228	0.178

8.2 天凤对战的评价

为了试验我们的程序与人类玩家对战的实力，我们在名为天凤的麻将网站运行程序进行对战，规则与玛塔里酱相同。天凤玩家根据其水平在不同的桌对战，共有四种对战桌，级别从高到低分别为‘凤凰’、‘特上’、‘上级’和‘一般’。我们基于R值进行评价。R值的变化与平均顺位负相关，其计算如下：

$$R = R + (50 - Rank \times 20 + \frac{AveR - R}{40}) \times 0.2, \tag{14}$$

其中$Rank$为该局顺位，$AveR$为该局全部玩家的平均R值，R初值为1500。平均顺位进步0.1就能导致R值陡增100。

有两种R值的评价方式，分别是安定R和保证安定R[15]。安定R指玩家一直以现在的顺位分布时计算的R值，保证安定R指玩家之后的战绩随机选择时比安定R低一个范围的R值。这样的评价用于对比人类玩家的战绩。

8.3 评价结果

表10展示了天凤战绩。计算t值可知我们的程序和先前的工作得到的平均顺位差异不显著（p-value=0.22），但我们的程序提升了保证安定R。表9展示了天凤对战的和率和放铳率。

[14] Mattari Mahjong http://homepage2.nifty.com/kmo2/

[15] http://totutohoku.b23.coreserver.jp/hp/SLtotu14.htm

表 9. 和率与铳率

	和率	铳率
我们的程序	0.245	0.131
Mizukami et al.[5]	0.256	0.148

8.4 讨论

我们的程序在对手立直且手牌不行时倾向于弃和。图6展示了我们的程序一次弃和，我们的程序听牌但难以取胜，中是铳牌的概率不小，于是程序弃和并切出8筒。

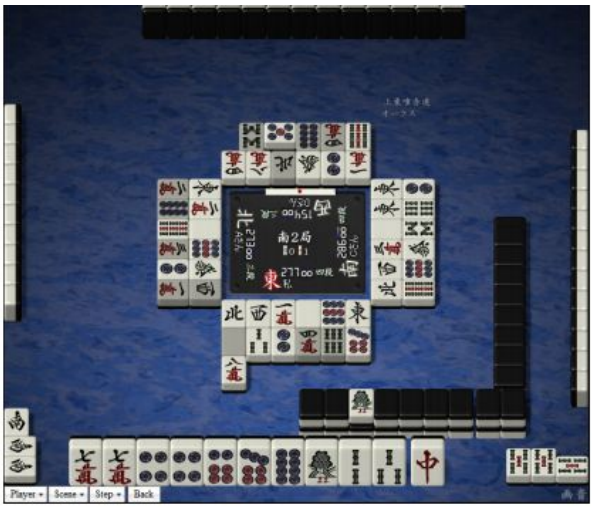

图 6. 弃和情形

我们的程序切牌类似于人类玩家。图7展示了我们的程序一次兜牌，这个情形下想和牌应当切出3或5筒，然而它们可能是铳牌，于是程序切出了7筒，而在该轮最后程序仍然听牌了。

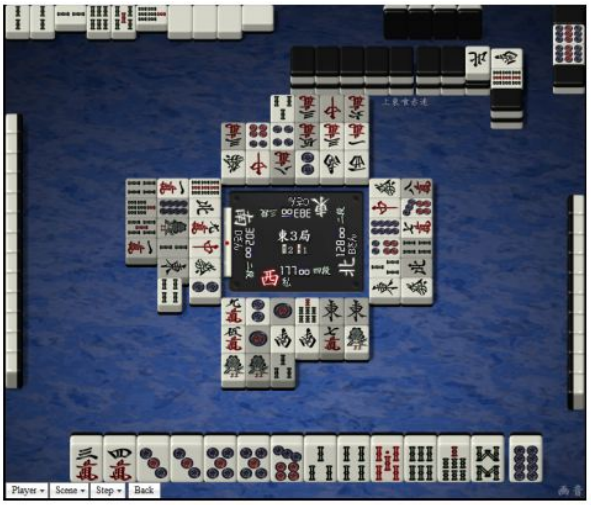

图 7. 兜牌情形

与我们先前的工作对比，程序对战玛塔里酱的成绩提升了，但对战人类玩家的战绩没有提升。可能的原因是玛塔里酱的和牌能力要高于对战的人类玩家，因此实际对战时蒙特卡洛打牌使用更频繁。程序的弃和能力比以前提高了。

当我们程序的手牌难以取胜时，单人麻将打牌是糟糕的选择。单人麻将打牌模型用来计算听牌或弃和的对手比例，对其的提升可以使得程序更强。我们的程序用了一些事先规定，比如听牌的玩家直接宣告立直。为了提升我们的程序，我们需要改进模型使得不管玩家立不立直都能预测。

9. 相关研究

先前的工作中我们定义了单人麻将游戏，指只有一名玩家且不能鸣牌[5]。我们使用人类玩家的游戏记录训练得到单人麻将程序，并分析单人麻将和四麻的差异来扩展单人麻将程序。程序高于人类玩家的平均水平，这些结果说明麻将程序能够鸣牌和弃和是重要的。研究过程的一个问题是我们需要人工标记数据集中玩家弃和的界面，且需要大量的玩家手牌和游戏情形的组合数据。另一个问题是训练需要的数据难以收集，还有程序无法兜牌等问题。

Wagatsuma et al.[9]提出了一个预测对手打点的支持向量机方法。这个方法提取游戏记录中的特征向量并在玩家切牌时预测打点。他们方法的一个问题是一些训练数据集中玩家听牌的界面导致的训练低效，另一个问题是预测准确度的评价只基于人类和程序的一致率。

Miki和Chikayama[10]生成了使用蒙特卡洛树搜索的麻将程序。由于麻将游戏树的搜索中叶结点数量太多，他们用模拟的方式进行搜索，对手的手牌和切牌在模拟中随机选择。尽管没有使用麻将的知识，该程序仍然比仅仅试图减小向听数的简单程序表现更出色。该程序与人类玩家的打牌风格迥异，其经常鸣牌以尽量听牌，因为模拟中对手玩家是不会和牌的。

电脑扑克已经有了许多的研究。CFR+方法使用自我对局的结果计算近似纳什均衡策略，最近已解决了两人有限注德州扑克[1]。长远来看纳什均衡策略是不会失败的，但其不能识别和利用对手的弱点。Van der Kleij[2]使用对手模型和蒙特卡洛树搜索来利用之，对手模型用来预测对手的手牌和选择。他提出了一个根据游戏记录的玩法对玩家聚类的算法。Aaron[3]提出了一个游戏过程中更新对手模型

表 10. 顺位分布

	1st	2nd	3rd	4th	平均顺位	对局数	安定R	保证安定R
我们的程序	0.242	0.281	0.248	0.23	2.46±0.04	2634	1718	1690
Mizukami et al.[5]	0.253	0.248	0.251	0.248	2.49±0.06	1441	1689	1610

来利用的方法，他的程序可以更新当前游戏界面摊牌时的手牌顺位分布。

因为程序与普通玩家的对局数小，所以麻将中对对手建模变得困难。对对手的简单建模来预测其切牌和手牌并不适合麻将。因此我们使用不同的建模方法，用来处理手牌不具体的全体玩家。

10. 总结

这篇论文的工作中，我们提出了利用游戏记录训练对手模型、并由模型的预测结果和蒙特卡洛模拟来打牌的方法。试验结果说明了三个预测模型比高级别玩家能力更高，程序也像人类玩家在弃和与进攻之间达到了好的平衡。在大型麻将对局网站、天凤中，我们的程序对战2643局后R值为1718，基本相当上级桌的一般玩家。

在之后的工作中，我们计划将玩家所持点数纳入计算。我们的程序切牌时没有考虑所持点数，而高级别玩家会根据点数考虑是否弃和或暗听，这些决定能极大影响最终顺位并且是提升实力的关键。模拟中计算的值应当根据对战结果而不同，即当模型根据点数预测结果顺位的分布时，顺位的期望应当作为一个模拟中的奖励。

参考文献

[1] M. Bowling, N. Burch, M. Johanson, and O. Tammelin. Heads-up limit hold'em poker is solved. *Science*, 347(6218):145–149, 2015.

[2] A. Van Der Kleij. Monte carlo tree search and opponent modeling through player clustering in no-limit texas hold' em poker. Master's thesis, University of Groningen, 2010.

[3] A. Davidson. Opponent modeling and search in poker: Learning and acting in a hostile and uncertain environment. Master's thesis, University of Alberta, 2002.

[4] M. Buro, J. R. Long, T. Furtak, and N. R. Sturtevant. Improving state evaluation, inference, and search in trick-based card games. *Proceedings of 21th International Joint Conference on Artificial Intelligence*, pages 1407–1413, 2009.

[5] N. Mizukami, R. Nakahari, A. Ura, M. Miwa, Y. Tsuruoka, and T. Chikayama. Realizing a four-player computer mahjong program by supervised learning with isolated multi-player aspects. *Transactions of Information Processing Society of Japan*, 55(11):1–11, 2014.

[6] J. Duchi and Y. Singer. Efficient online and batch learning using forward backward splitting. *The Journal of Machine Learning Research*, 10:2899–2934, 2009.

[7] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

[8] E. W. Noreen. Computer-intensive methods for testing hypotheses: an introduction. *Computer*, 1989.

[9] A. Wagatsuma, M. Harada, H. Morita, K. Komiya, and Y. Kotani. Estimating risk of discarded tiles in mahjong using svr. *The Special Interest Group Technical Reports of IPSJ. GI,[Game Informatics]*, 2014(12):1–3, 2014.

[10] A. Miki and T. Chikayama. Research on decision making in multiplayer games with imperfect information. Master's thesis, University of Tokyo, 2010.