亚马逊棋程序设计思路总结

1、比赛模式

（1）开启服务器等待GUI发送棋盘

（2）初始化游戏和神经网路

（3）将新棋盘（board）使用蒙特卡洛树搜索得到下一步策略pi　'此处的策略分为，选皇后点(start)、选落点(end)、选放箭点(arrow)三个概率'

（4）采用随机策略选择以上三点，判断走法是否合理直到第一个合理的走法，随即更新棋盘和判断输赢，接着将棋盘输赢结果发给GUI

2、训练模式

3、函数文档

1：class AmazonsGame()：定义棋盘规则与走法

方法名	含义	参数含义	返回值
init(self, n)	初始化棋盘	n：设置棋盘大小：n*n	无
getInitBoard(self)	得到棋盘	无	返回n*n的二维数组，eg:[……]
getBoardSize(self)	得到棋盘大小	无	返回一个元组,eg:（10，10）
getActionSize(self)	得到行为大小	无	300
getNextState(self, board, player, action)	更新选点,落点,放箭后的新棋盘	board：棋盘:10*10数组 player：当前玩家: 1/2 action：策略走法 eg (67,78,99): 起始坐标,放皇后坐标,箭坐标	返回新棋盘和下次走棋方(0/1)

方法名	含义	参数含义	返回值
getValidMoves(self, board, Ps)	得到所有有效的行为动作	Ps :概率1 * 300 一维数组 eg [0.2， 0.02，0.23，.....] board：棋盘	Ps：将不能到达的点直接置0后归一化返回新的概率 valids：所有可能的动作，eg[(s, e, a), (98, 87, 56).....]
getGameEnded(self, board, player)	判断某方是否结束	board：棋盘 player：当前玩家	player胜返回1; player败返回-1 未结束返回0
get_neighbours(self, board, stone)	得到八个方向上的邻居	board：棋盘 stone某点坐标 eg(2,3)	返回可走邻居的坐标集合 eg{23,54,56.....} 不可走的邻居不在其中
expand_territory(self, board, expanded)	查询能扩展的范围（能到达的点）	board:棋盘;expanded:某点周围八个方向可扩展的点的集合eg {(2, 3), (3,4),(4,5)....}	返回新点周围八个方向可扩展的点的集合
get_territory(self, board, stone)	获得某点的所有可达点	board:棋盘，stone：棋子坐标eg(2,4)	返回所有可到达点的集合 eg{(1,2), (3,4).....}
set_state(self, board, territory, value)	设置能到达点在棋盘上的值	board：棋盘，territory：某点所有可到达点的集合 eg{(1,2), (3,4).....}	返回新棋盘
make_input_board_for_NN(self, board)	将原始棋盘转换成10个棋盘：前8个为每个棋子的封闭计算，第九个为原始棋盘，第10张棋盘（哪些棋子还可以走）	board：原始棋盘	返回10张棋盘三维数组：(10* 10* 10)
is_closed(self, board_arr)	查询各棋子封闭状态	board_arr：10* 10 *10 的三维数组	返回每个点的封闭状态，8个元素的字典 {0：1，1：0，2：1，3：0，4：0.....} 1封闭 0else

方法名	含义	参数含义	返回值
getCanonicalForm(self, board, player)	规范化棋盘	board,：棋盘 player：当前玩家：0/1	当前玩家是白方1 -->返回棋盘;当前玩家是黑棋2 -->将棋子颜色反转 player=1代表是自己
getSymmetries(self, board, pi)	将局面和策略顺时针旋转180度	board：棋盘 pi:策略向量 type:1*300一维数组	返回4个棋盘与策略的元组 eg:[(board, pi),(board, pi),(board, pi),(board, pi)]
stringRepresentation(self, board)	将棋盘转换成字符串	board：棋盘	返回棋盘的字符串

2：class Board()：定义棋盘信息

方法名	含义	参数	返回值
init(self)	初始化棋盘和棋子	无	无
judge_legal_move(self, start, end)	判断起始点走法是否合法	start：起始点 eg:45 end:终止点 eg:67	True if 能到达 False else

3：class Arena()：竞技场

方法名	含义	参数	返回值
playGames(self, num, verbose=False)	玩 num 盘游戏,玩家一和二各先手 num/2 盘	num：总盘数	赢、输、和棋数，eg:1，2，3
playGame(self, verbose=False)	进行一次从头到结束的搜索。	无	返回输赢 True/False

4：class Coach()：训练模式

方法名	含义	参数	返回值
init(self, game, nnet, args)	初始化	game: 游戏规则对象 nnet：神经网路对象 args：字典类型参数	无
executeEpisode(self)	根节点到终结点执行一次	通过上述初始化将棋盘传入	返回 三个元组的列表:[(棋盘 , 策略 , 1/-1(输赢奖励)),(s),(p),(v).....]
learn(self)	自我学习	无	无
	得到训练		

方法名	含义	参数	返回值
getCheckpointFile(self, iteration)	某次迭代的训练数据名称	无	无
loadTrainExamples(self)	从文件中加载训练数据	无	无

5：class MCTS()

方法名	含义	参数	返回值
init(self, game, nnet, args)	初始化	game:游戏对象 nnet:神经网路对象 args:神经网路参数	无
getActionProb(self, canonicalBoard, temp=1)	传入棋盘得到MCTS树搜索新策略	canonicalBoard：棋盘	返回新策略列表：1*300
search(self, canonicalBoard)	MCTS树搜索方法	canonicalBoard：棋盘	返回输赢 1 赢 0 else

4、代码中神经网路中的参数

1、trainExamples: 单个的训练数据：每4个为一组（棋盘各种翻转算一种情况）-------[[board, curPlayer, pi, None],[board, curPlayer, pi, None],[board, curPlayer, pi, None],[board, curPlayer, pi, None]]

2、iterationTrainExamples：迭代数据 ——格式：3个一组：[(棋盘 , 策略 , 1/-1(输赢奖励)),(s,pi,z),(),()……..] ： 真实走三盘（每盘四个trainExamples的数据）保存一次该参数

3、trainExamplesHistory：存储训练的总数据，每20条iterationTrainExamples保存到文件一次

iterationTrainExamples（iterationTrainExamples（trainExamples））

----------20--3-------------------------------4----------