神经网络在俄罗斯方块自动游戏中的应用

结题报告

机械 54 班

卢仁浩 学号：2015010524

朱迪 学号：2015010538

陈诺 学号：2015010533

冉瑶 学号：2015010536

一、设计说明书

1、俄罗斯方块游戏模块

游戏版块有三个部分组成：消息处理、图形界面和方块。

消息处理方面，主要用到了windows CPI函数，借用了网上的代码。图形界面集成于screenn类和stock类。类图如下：

|  |
| --- |
| Screen |
| -plot[20][10]:int  -rect[20][10]:RECT  -rect1[20][10]:RECT |
| +screen()  +prin(HDC hdc):void  +fill(HDC hdc):void  +clearrow():void |

|  |
| --- |
| stock |
| -sort:int -block[4][2]:int  -x:int -maxs:int  -y:int -scr:screen\*  -situation:int |
| +stock() +draw():bool  +clear():void +setXY(int x1,int y1):void  +setS(int s):void +gx():int  +gy():int +gs():int  +gms():int +changeXY(int x1,int y1):void  +move(int x,int y):bool +left():bool  +right():bool +down():bool  +turn():bool +refresh\_block():void  +renew():void |

详细功能说明：

（1）Screen类：

数据成员

int plot[20][10] 存储面板在不同位置如否为实心

RECT rect[20][10]; 存储面板10\*20个矩阵的边线

RECT rect1[20][10]; 存储面板10\*20个矩阵的实心内容

函数成员

screenn() 构造函数，初始化rect和rect1

void prin(HDC hdc) 绘制图形界面的矩形边线

void fill(HDC hdc) 根据plot数组填充实心方块

void clearrow() 消去满了的行

（2）stock类：

数据成员

int sort 存储方块的类型，1代表长条，2代表z形

3代表反z形，4代表L形，5代表反L形

6代表T形，7代表方形

int x,y 代表方块核心块的位置

int situation 代表方块的方向

int block[4][2] 代表每个方块四个小方格的坐标

int maxs 代表每一种方块可能方向的数量

screenn \*scr 为调用screenn类plot数组做准备

函数成员：

stock() 构造函数，初始化方向，位置和随机的类型

bool draw() 将block数组的坐标绘制在plot上并返回1，如果不能绘制，

返回0

void clear() 当方块坐标改变，消除原来坐标在plot上的痕迹

void setXY(int x1,int y1) 重设方块坐标，用于方块触底之后的重新生成

void setS(int s) 重设方块方向，同上用途

int gx() 返回当前x值

int gy() 返回当前y值

int gs() 返回当前s值

int gms() 返回当前maxs值

void changeXY(int x1,int y1)改变方块坐标

bool move(int x,int y); 移动方块，若成功，返回1，若不成功，返回0

bool left() 方块左移

bool right() 方块右移

bool down() 方块下移

bool turn() 方块转向

void refresh\_block() 根据方向和坐标刷新四个小方块的坐标

void renew() 方块定格后的重生

2、神经网络方面

nether：函数成员：

1.构造函数double inn[200]，构造一个200\*1的数组，用于存放输入数据；2.构造函数double w[200]，构造一个200\*1的数组，用于存放随机数；

3.构造函数double outt，构造一个double型变量；

4.函数double cal()，作为神经元的功能函数，实现对输入的200维变量的加权求和，再另加一个偏置而得到活性值，最后将活性值用阈值韩束处理得到范围为0到1的输出值

net：数据成员：

1.double \*inn

2.nether ne1[200]，构造一个nether类，便于使用nether类的成员函数；

3.nether ne2[34] 构造一个nether类，便于使用nether类的成员函数；

函数成员：

1.net()，通过nether类中的函数实现对net1和net2中随机数的赋值；

2.void cal()，功能函数，通过将net1中的输出值作为net2中的输入值，得到神经网络最终需要的输出值；

3.void setinn(double \*p)，给net1和net2中的inn赋值

|  |
| --- |
| nether |
| + inn[200]:double  + w[200]:double  + outt:double=0  + cal():double |

|  |
| --- |
| net |
| -\*inn:double  - |
| + net()  + cal():void  + setinn(double \*p):void |

nether

nether

依赖关系

3、难点和算法

本算法的难点在于三方面：俄罗斯方块类的实现、自我学习和权值修改。

俄罗斯方块现在网络上代码很多，但是真正用到类的方法的并不多。我们这次的大作业运用了类的方法，虽然复杂度会变高，但是整体有序性会变大。

本课题最大的难点还是在于计算机的自我学习过程。

首先我们知道自我学习有很多途径，比如较为普通的枚举，通过列举很多种方案然后选出最佳。但是枚举过于繁琐复杂，在处理大数据的时候就会花费较长的时间，甚至可能导致笔记本的崩溃，所以这个方案是行不通的。因此，我们借助于神经网络，用一种训练和记忆的手段，给出一些输入——输出样本对并训练网络使之尽可能地拟合这些样本，对前一次的数据进行一定的分析，作出决策并反馈，然后进行一次又一次的修正最终得出相对满意度很高的操作。

神经网络权值修改中，利用不同数组储存每一层神经元的输出，还有在枚举判断后得出的最优解数组，利用反向传递误差价值梯度减小原理，由最后的误差传递到第一层，之后正向进行权值修改。修改的参数参考已有的误差计算公式进行逐个计算。

二、用户手册

本程序是结合神经网络的俄罗斯方块游戏，用户可手动进行游戏操作，除此之外，还能利用神经网络系统完成自动游戏功能。

游戏界面为10×20的网格，默认游戏模式为手动模式，该模式下，玩家利用方向键操控方块，若方块堆叠满一行则消除该行，每消除一行得分，分数可叠加，直到任一方块到达网格顶端，游戏结束。自动模式下，程序将调用神经网络系统进行自动游戏，规则与手动模式相同，无需玩家手动干预。

手动模式操作规则：

左方向键控制方块左移；右方向键控制方块右移；

下方向键加速方块下落；上方向键控制方块转向；

空格键暂停游戏。

自动模式操作规则：

手动模式下按P键切换为自动模式。

三、总结报告

此份总结报告分为三部分，一是组队来由和前期准备，二是程序代码编写过程。

* 1. 组队来由和前期准备

早在课程的开始，我们便有着做大作业的打算，想着如此才能够综合运用所学的知识写出自己想要的东西而并不是仅仅局限于考试，同时对于课程的学习也有更大的好处。而且宿舍里有一个对此方面比较精通的人，能够带领大家并在关键时刻给予指导。所以我们查阅了有关资料，海选课题，最终发现俄罗斯方块是一个简单却有很大改造空间的游戏，而人工智能、深度学习在现今又是十分流行，因此决定用神经网络的方法使电脑自我学习，自主完成俄罗斯方块的游戏。

* 1. 程序代码编写过程

首先这是一个艰难、富有挑战性却充满乐趣的过程。我们查阅了很多关于神经网络的资料，同时也对一些必要的算法进行了学习，整个课题的前半段时间，我们都是以理论学习和概念探索为主，再经过了大约两周的资料收集之后，我们开始了代码的编写。

代码编写部分是由小组分工完成，主要是俄罗斯方块游戏部分、神经网络的实现部分和神经网络与俄罗斯方块的连接部分。由于各个部分联系都较为紧密，这就要求我们在编写过程中时刻保持交流，而且分工必须明确。起初程序编写进度十分缓慢，在中期我们集中讨论编写过程中的问题，相互交流了进度和各自部分的难点，之后再一个个攻克这些技术上的困难；后半部分的进度明显加快，三天时间里我们就完成了第一版的所有代码，但在调试的过程中却发现了很多问题，于是我们对照参考书，对程序进行了进一步的修改，最终得出了一个较为完善但仍有优化空间的一个主程序。

四、测试用例

见附件

五、源代码

见附件