/\*CopyRight Ramin Edjlal\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*2018\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The Magic Table Game Satte Learing Quantum Atamata.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

using System;

using System.IO;

namespace LearningMachine

{

[Serializable]

public class NetworkQuantumLearningKrinskyAtamata : LearningKrinskyAtamata

{

public static String Root = System.IO.Path.GetDirectoryName(Environment.GetCommandLineArgs()[0]);

static void Log(Exception ex)

{

try

{

Object a = new Object();

lock (a)

{

string stackTrace = ex.ToString();

File.AppendAllText(Root + "\\ErrorProgramRun.txt", stackTrace + ": On" + DateTime.Now.ToString()); // path of file where stack trace will be stored.

}

}

catch (Exception t) { Log(t); }

}

int r, m, k;

LearningKrinskyAtamata[,] Netfi;

public NetworkQuantumLearningKrinskyAtamata(int r0, int m0, int k0) : base(r0, m0, k0)

{

Object o = new Object();

lock (o)

{

Netfi = new LearningKrinskyAtamata[m0, k0];

for (int j = 0; j < m0; j++)

for (int k = 0; k < k0; k++)

Netfi[j, k] = new LearningKrinskyAtamata(r0, m0, k0);

r = r0;

m = m0;

k = k0;

}

}

public double LearningAlgorithmRegardNet(int Row, int Column)

{

Object o = new Object();

lock (o)

{

double Hu = 1;

try

{

Netfi[Row, Column].LearningAlgorithmRegard();

Hu = Netfi[Row, Column].alpha[State];

}

catch (Exception t)

{

Log(t);

}

return Hu;

}

}

public int IsRewardActionNet(int Row, int Column)

{

Object o = new Object();

lock (o)

{

if (Netfi[Row, Column].IsReward)

return 1;

return -1;

}

}

public double IsPenaltyActionNet(int Row, int Column)

{

Object o = new Object();

lock (o)

{

if (Netfi[Row, Column].IsPenalty)

return 0;

return -1;

}

}

public double LearningAlgorithmPenaltyNet(int Row, int Column)

{

Object o = new Object();

lock (o)

{

double Hu = 1;

try

{

Netfi[Row, Column].LearningAlgorithmPenalty();

Hu = Netfi[Row, Column].alpha[State];

}

catch (Exception t)

{

Log(t);

}

return Hu;

}

}

public double LearingValue(int Row, int Column)

{

Object o = new Object();

lock (o)

{

double Hu = 1;

try

{

Hu = Netfi[Row, Column].alpha[State];

}

catch (Exception t)

{

Log(t);

}

return Hu;

}

}

}

}