1　機械学習と深層学習

1.1　強化学習とは

強化学習 (reinforcement learning) は、一連の行動の最後に評価が与えられるような場合に用いる学習方法です。強化学習は、たとえばゲームの勝敗を通じた戦略知識の獲得などに用いることができます。

教師ありの学習では効率的な学習が可能ですが、実は教師データをどのように構成するのかが大変な難問です。これに対して強化学習の枠組みでは、一連の着手が終了した後に評価を得て、その評価に基づいて学習を進めることができます。ここで、ゲームにおける着手として良かったか悪かったかの評価は、簡単です。それは、ゲームの勝敗をみればよいのです。強化学習では、一連の動作の最後に得られる価値のことを、報酬 (reward) と呼びます。強化学習では、報酬を受けた時、そこに至るまでの過程でとった複数の行動それぞれに対して、報酬を分配して与えることになります。

機械学習と深層学習　小高知宏著　オーム社 52 頁

1.2 強化学習 (Q 学習) の例題　迷路の探索を学習します qlearning.c

```
/*******************************************/
/*          qlearning.c                    */
/*    強化学習（Q 学習）の例題プログラム          */
/*    迷路の探索を学習します                    */
/*使い方                                     */
/*:\Users\odaka\dl\ch2>qlearning             */
/*******************************************/

/*Visual Studio との互換性確保 */
#define _CRT_SECURE_NO_WARNINGS

/*ヘッダファイルのインクルード*/
#include <stdio.h>
#include <stdlib.h>

/* 記号定数の定義                */
#define GENMAX  1000 /*学習の繰り返し回数*/
#define NODENO  15  /*Q値のノード数*/
#define ALPHA 0.1/*学習係数*/
#define GAMMA 0.9/*割引率*/
#define EPSILON 0.3 /*行動選択のランダム性を決定*/
#define SEED 32767 /*乱数のシード*/

/* 関数のプロトタイプの宣言    */
int rand100() ;/*0〜100 を返す乱数関数*/
int rand01() ;/*0 又は 1 を返す乱数関数*/
double rand1() ;/*0〜1 の実数を返す乱数関数*/
void printqvalue(int qvalue[NODENO]);/*Q 値出力*/
int selecta(int s,int qvalue[NODENO]);/*行動選択*/
int updateq(int s,int qvalue[NODENO]);/*Q 値更新*/

/***************/
/*  main() 関数  */
/***************/
int main()
{
 int i;
```

```
 int s;/*状態*/
 int t;/*時刻*/
 int qvalue[NODENO];/*Q 値*/


 srand(SEED);/*乱数の初期化*/

 /*Q値の初期化*/
 for(i=0;i<NODENO;++i)
  qvalue[i]=rand100() ;
 printqvalue(qvalue) ;


 /*学習の本体*/
 for(i=0;i<GENMAX;++i){
  s=0;/*行動の初期状態*/
  for(t=0;t<3;++t){/*最下段まで繰り返す*/
   /*行動選択*/
   s=selecta(s,qvalue) ;

   /*Q 値の更新*/
   qvalue[s]=updateq(s,qvalue) ;
  }
  /*Q 値の出力*/
  printqvalue(qvalue) ;
 }
 return 0;
}

/****************************/
/*       updateq() 関数      */
/*       Q 値を更新する       */
/****************************/
int updateq(int s,int qvalue[NODENO])
{
 int qv ;/*更新される Q 値*/
 int qmax ;/*Q値の最大値*/

 /*最下段の場合*/
 if(s>6){
```

```
  if(s==14)/*報酬の付与*/
   qv=qvalue[s]+ALPHA*(1000-qvalue[s]) ;
  /*報酬を与えるノードを増やす*/
  /*他のノードを追加する場合は*/
  /*下記２行のコメントを外す    */
//  else if(s==11)/*報酬の付与*/
//   qv=qvalue[s]+ALPHA*(500-qvalue[s]) ;
  else/*報酬なし*/
   qv=qvalue[s] ;
 }
 /*最下段以外*/
 else{
  if((qvalue[2*s+1])>(qvalue[2*s+2]))
   qmax=qvalue[2*s+1];
  else qmax=qvalue[2*s+2];
  qv=qvalue[s]+ALPHA*(GAMMA*qmax-qvalue[s]) ;
 }

 return qv ;
}

/****************************/
/*        selecta() 関数     */
/*       行動を選択する      */
/****************************/
int selecta(int olds,int qvalue[NODENO])
{
 int s ;

 /* ε -greedy 法による行動選択*/
 if(rand1()<EPSILON){
  /*ランダムに行動*/
  if(rand01()==0) s=2*olds+1 ;
  else s=2*olds+2 ;
 }
 else{
  /*Q値最大値を選択*/
  if((qvalue[2*olds+1])>(qvalue[2*olds+2]))
   s=2*olds+1;
  else s=2*olds+2;
```

```
 }

 return s ;
}

/****************************/
/*    printqvalue() 関数     */
/*     Q 値を出力する        */
/****************************/
void printqvalue(int qvalue[NODENO])
{
 int i ;

 for (i=1;i<NODENO;++i)
  printf("%d\t",qvalue[i]);

 printf("\n");

}

/****************************/
/*     rand1() 関数          */
/*0〜1 の実数を返す乱数関数  */
/****************************/
double rand1()
{
 /*乱数の計算*/
 return (double)rand()/RAND_MAX ;

}

/****************************/
/*     rand01() 関数         */
/*     0 又は 1 を返す乱数関数 */
/****************************/
int rand01()
{
 int rnd ;

 /*乱数の最大値を除く*/
```

```
 while((rnd=rand())==RAND_MAX) ;
 /*乱数の計算*/
 return (int)((double)rnd/RAND_MAX*2) ;

}

/****************************/
/*     rand100() 関数       */
/*     0〜100 を返す乱数関数 */
/****************************/
int rand100()
{
 int rnd ;

 /*乱数の最大値を除く*/
 while((rnd=rand())==RAND_MAX) ;
 /*乱数の計算*/
 return (int)((double)rnd/RAND_MAX*101) ;

}
```

1.3 qlearning.exe の使い方

```
./qlearning | more
```