David Olson

CS 470

Value Iteration Homework

Initialization

Rewards:

R(Pizza) = 10

R(Salad) = 3

R(Poke-in-the-eye) = -10

R(everything-else) = -0.4

Utilities:

U(Pizza) = 10

U(Salad) = 3

U(Poke-in-the-eye) = -10

U(everything-else) = 1.0

Square Types:

Squares that were “blocked” (could not enter) were marked as such so they would not be considered in the value iteration algorithm or as a state that an adjacent square could move into

Neighbors:

Neighbor squares representing up, down, left, and right were stored. These neighbors are the square itself if that direction is a block or out of bounds

Salad, Pizza, Poke:

These squares did not have their utilities changed, but were referenced by other squares in that process

Directions:

All directions were trivially initialized to UP

Results:

\*U stands for Utility, D for direction

**Before Iterations:**

Reward: 3 block block block Reward: 10

U: 1.0 D: UP U: 1.0 D: UP U: 1.0 D: UP U: 1.0 D: UP U: 1.0 D: UP

U: 1.0 D: UP U: 1.0 D: UP block U: 1.0 D: UP Reward: -10

U: 1.0 D: UP U: 1.0 D: UP U: 1.0 D: UP U: 1.0 D: UP U: 1.0 D: UP

U: 1.0 D: UP U: 1.0 D: UP U: 1.0 D: UP U: 1.0 D: UP U: 1.0 D: UP

**1 Iteration:**

Reward: 3 block block block Reward: 10

U: 2.0 D: UP U: 1.3 D: LEFT U: 0.8 D: LEFT U: 0.6 D: RIGHT U: 5.8 D: UP

U: 1.3 D: UP U: 0.8 D: LEFT block U: -0.4 D: UP Reward: -10

U: 0.8 D: UP U: 0.6 D: DOWN U: 0.6 D: UP U: 0.4 D: RIGHT U: -0.4 D: UP

U: 0.6 D: DOWN U: 0.5 D: DOWN U: 0.5 D: RIGHT U: 0.5 D: RIGHT U: 0.4 D: DOWN

**10 Iterations:**

Reward: 3 block block block Reward: 10

U: 3.4 D: RIGHT U: 4.2 D: RIGHT U: 5.1 D: RIGHT U: 5.8 D: RIGHT U: 6.9 D: UP

U: 2.7 D: UP U: 3.4 D: UP block U: 3.2 D: UP Reward: -10

U: 2.1 D: UP U: 2.5 D: UP U: 1.9 D: LEFT U: 2.2 D: UP U: 0.2 D: LEFT

U: 1.5 D: UP U: 1.8 D: UP U: 1.4 D: UP U: 1.5 D: UP U: 0.8 D: LEFT

Code:

**Main.Java**

*/\*\*  
 \* Created by David on 12/3/2016.  
 \*/***public class** Main {  
  
 **static** Node[][] *squares* = **new** Node[5][5];  
 **static** String[] *directions* = **new** String[]{**"UP "**, **"DOWN "**, **"LEFT "**, **"RIGHT"**};  
  
 **public static void** main(String[] args) {  
  
 *//set up the grid and nodes* **for**(**int** i = 0; i < 5; i++){  
 **for**(**int** j=0; j<5; j++){  
 **double** utility = 1;  
 **double** reward = -.4;  
 String type = **"normal"**;  
  
 **if**(i==0){  
 **if**(j==0) {  
 reward = 3;  
 utility = 3;  
 } **else if**(j == 4){  
 reward = 10;  
 utility = 10;  
 } **else** {  
 type = **"block"**;  
 }  
  
 } **else if**(i == 2){  
 **if**(j==2){  
 type = **"block"**;  
 } **else if**(j ==4){  
 reward = -10;  
 utility = -10;  
 }  
 }  
  
 *squares*[i][j] = **new** Node(utility, reward, type);  
 }  
 }  
  
 *//set up the neighbor nodes* **for**(**int** i = 0; i < 5; i++) {  
 **for** (**int** j = 0; j < 5; j++) {  
 Node upNeighbor = *setUpNeighbor*(i,j);  
 Node downNeighbor = *setDownNeighbor*(i,j);  
 Node leftNeighbor = *setLeftNeighbor*(i,j);  
 Node rightNeighbor = *setRightNeighbor*(i,j);  
 *squares*[i][j].setNeighbors(**new** Node[]{upNeighbor, downNeighbor,

leftNeighbor, rightNeighbor});  
 }  
 }  
  
 *printGrid*();  
  
 *//iterate through value iteration 10 times* **for**(**int** x = 0; x < 10; x++) {  
 **for** (**int** i = 0; i < 5; i++) {  
 **for** (**int** j = 0; j < 5; j++) {  
 Node n = *squares*[i][j];  
  
 *//don't do value iteration for blocked squares, and terminal*

*points* **if** (!n.getType().equals(**"block"**) && !((i==0)&&(j==0))

&&!((i==0)&&(j==4)) &&!((i==2)&&(j==4))) {  
 **double** max\_Value = Double.***MIN\_VALUE***;  
 **int** direction = 0;  
 **double**[] fourActions = **new double**[]{n.upSummation(),

n.downSummation(),n.leftSummation(),

n.rightSummation()};  
  
 **for** (**int** k = 0; k < 4; k++) {  
 **if** (fourActions[k] > max\_Value) {  
 max\_Value = fourActions[k];  
 direction = k;  
 }  
 }  
  
 n.setUtility(max\_Value + n.getReward());  
 n.setDirection(direction);  
 }  
 }  
 }  
  
 *printGrid*();  
 }  
 }  
  
 **static** Node setUpNeighbor(**int** i , **int** j){  
 **if**(i == 0){  
 **return** *squares*[i][j];  
 }  
 **if**(*squares*[i-1][j].getType().equals(**"block"**)){  
 **return** *squares*[i][j];  
 }  
  
 **return** *squares*[i-1][j];  
 }  
  
 **static** Node setDownNeighbor(**int** i , **int** j){  
 **if**(i == 4){  
 **return** *squares*[i][j];  
 }  
 **if**(*squares*[i+1][j].getType().equals(**"block"**)){  
 **return** *squares*[i][j];  
 }  
  
 **return** *squares*[i+1][j];  
 }  
  
 **static** Node setLeftNeighbor(**int** i , **int** j){  
 **if**(j == 0){  
 **return** *squares*[i][j];  
 }  
 **if**(*squares*[i][j-1].getType().equals(**"block"**)){  
 **return** *squares*[i][j];  
 }  
  
 **return** *squares*[i][j-1];  
 }  
 **static** Node setRightNeighbor(**int** i , **int** j){  
 **if**(j == 4){  
 **return** *squares*[i][j];  
 }  
 **if**(*squares*[i][j+1].getType().equals(**"block"**)){  
 **return** *squares*[i][j];  
 }  
  
 **return** *squares*[i][j+1];  
 }  
  
 **static void** printGrid(){  
 **for**(**int** i = 0; i < 5; i++) {  
 **for** (**int** j = 0; j < 5; j++) {  
 Node n = *squares*[i][j];  
 **if**(n.getType().equals(**"block"**)){  
 System.***out***.print(**" block "**);  
 } **else if**(i==0 && j==0) {  
 System.***out***.print(**" Reward: 3 "**);  
 } **else if**(i==0 && j==4) {  
 System.***out***.print(**" Reward: 10 "**);  
 } **else if**(i==2 && j==4) {  
 System.***out***.print(**" Reward: -10 "**);  
 } **else** {  
 System.***out***.print(**"U: "** + String.*format*(**"%.1f"**, n.getUtility())

+ **" "**);  
 System.***out***.print(**"D: "** + *directions*[n.getDirection()] + **"**

**"**);  
 }  
 }  
  
 System.***out***.println(**""**);  
 }  
  
 System.***out***.println(**"\n"**);  
 }  
}

**Node.java**

*/\*\*  
 \* Created by David on 12/3/2016.  
 \*/***public class** Node {  
 **double utility**;  
 Node[] **neighbors**;  
 **double reward**;  
 String **type**;  
 **int direction**;  
  
 **public** Node(**double** utitlity, **double** reward, String type){  
 **this**.**utility** = utitlity;  
 **this**.**neighbors** = **neighbors**;  
 **this**.**reward** = reward;  
 **this**.**type** = type;  
 }  
  
 **double** getUtility(){**return utility**;}  
 **void** setUtility(**double** utility){**this**.**utility** = utility;}  
 **double** getReward(){**return reward**;}  
 String getType(){**return type**;}  
 **void** setNeighbors(Node[] neighbors){**this**.**neighbors** = neighbors;}  
 Node[] getNeighbors(){**return this**.**neighbors**;}  
 **void** setDirection(**int** direction){**this**.**direction** = direction;}  
 **int** getDirection(){**return direction**;}  
  
  
 **double** upSummation(){  
 **return** .7 \* **neighbors**[0].getUtility()  
 + .1 \* **neighbors**[1].getUtility()  
 + .1 \* **neighbors**[2].getUtility()  
 + .1 \* **neighbors**[3].getUtility();  
 }  
  
 **double** downSummation(){  
 **return** .1 \* **neighbors**[0].getUtility()  
 + .7 \* **neighbors**[1].getUtility()  
 + .1 \* **neighbors**[2].getUtility()  
 + .1 \* **neighbors**[3].getUtility();  
 }  
  
 **double** leftSummation(){  
 **return** .1 \* **neighbors**[0].getUtility()  
 + .1 \* **neighbors**[1].getUtility()  
 + .7 \* **neighbors**[2].getUtility()  
 + .1 \* **neighbors**[3].getUtility();  
 }  
  
 **double** rightSummation(){  
 **return** .1 \* **neighbors**[0].getUtility()  
 + .1 \* **neighbors**[1].getUtility()  
 + .1 \* **neighbors**[2].getUtility()  
 + .7 \* **neighbors**[3].getUtility();  
 }  
}