

Program 2

Muaz Alhaider and Zakariya Ahmed

November 11, 2020

Contents

1	Modules	1
1.1	Transitional Probabilty	1
1.2	Prediction	3
1.3	Evidence Contional Probabilty	4
1.4	Filtering	5
1.5	Smoothing	5
2	Results	7
3	Screenshots	7

1 Modules

1.1 Transitional Probabilty

$Transitional\ Probabilty(Pos_i, Dir) = (Pos_{from\ left} : Drift(Left), Pos_{from\ straight} : Drift(Straight)),$

$Transitional\ Probabilty(Pos_i, Dir) = (Pos_{from\ left} : Drift(Left), Pos_{from\ straight} : Drift(Straight)),$

Left, *Right*, and *Straight* are all positions defiend in relation to what *Dir* is:

if *Dir* is EAST, then *Left* is SOUTH, *Right* is NORTH, and Straight is EAST.

Here, transitional probability has two forms: one which is all paths that converge to a point, and another where they diverge from a point. Prediction (as the name implies) wants all the possible paths to a point, which is why we include *Pos_{from behind}*. Smoothing however, does not require that, which is why it's not include.

```
@enum SquareType OPEN CLOSED
@enum Direction WEST NORTH EAST SOUTH
@enum DriftType STRAIGHT LEFT RIGHT
const Drift=Dict{STRAIGHT=>.7, LEFT=>.15, RIGHT=>.15}
const AllDirects =(WEST, NORTH, EAST, SOUTH)

# Gets the transitional probability of positions going into a point,
# or transitional probabilities from the point
# the first is used for prediction, the other for smoothing
function transprob( grid::Array{Array{Float64,1},1}, pos::Tuple{Int64, Int64}, dir::Direction,
arr = []

function _gen_parts(straightDir::Direction , leftDir::Direction, rightDir::Direction,
parent_pos = []
behind = move(pos, behindDir)
straight = move(pos, straightDir)
left = move(pos, leftDir)
right = move(pos, rightDir)

if( !notblocked(grid, straight)) # Bounce
push!(parent_pos, (pos,Drift[STRAIGHT]*grid[pos[1]][pos[2]], Drift[STRAIGHT] ))
elseif (getforward) # If this is smoothing, want the probability in front
push!(parent_pos, (straight,Drift[STRAIGHT]*grid[straight[1]][straight[2]], Drift[STRAIGHT] ))
end
if( notblocked(grid, behind) && !getforward) # Prob of square behind current to move to
push!(parent_pos, (behind,Drift[STRAIGHT]*grid[behind[1]][behind[2]], Drift[STRAIGHT] ))
end
if(notblocked(grid, left)) # Get probability of the left pos coming to current
push!(parent_pos, (left,Drift[LEFT]*grid[left[1]][left[2]], Drift[LEFT] ))
else #Bounce from left
push!(parent_pos, (pos,Drift[LEFT]*grid[pos[1]][pos[2]], Drift[LEFT] ))
end
if(notblocked(grid, right)) #Get probability of right pos coming to current
```

```

push!(parent_pos, (right,Drift[RIGHT]*grid[right[1]][right[2]],Drift[RIGHT]))
else #Bounce
push!(parent_pos, (pos,Drift[RIGHT]*grid[pos[1]][pos[2]], Drift[RIGHT]))
end
end

if(dir==WEST)
arr=_gen_parts(WEST, SOUTH, NORTH, EAST, grid, pos)
end
if(dir==NORTH)
arr=_gen_parts(NORTH, EAST, WEST, SOUTH, grid, pos)
end
if(dir==SOUTH)
arr=_gen_parts(SOUTH, EAST, WEST, NORTH, grid, pos)
end
if(dir==EAST)
arr=_gen_parts(EAST, SOUTH, NORTH, WEST, grid, pos)
end
arr
end

```

1.2 Prediction

$$Prediction(Grid, Direction) = \left\{ pos_i \in Grid \mid \sum_{(Pos_j, DriftProb)}^{Transition\ Probability(pos_i, direction)} DriftProb \cdot P(Pos_j) \right\}$$

Prediction(Grid, Direction) (as the name implies) attempts to predict where the agent will be given previous infoamtion. It does this by transforming the grid by the equation $\sum_{(Pos_j, DriftProb)}^{Transition\ Probability(pos_i, direction)} DriftProb \cdot P(Pos_j)$. This gets the probabily of an agent drifting (or if direction is straight, accurantly going to) a point, and what is the probabtily the agent would be at the point Pos_j .

```

# Prediction is the sum of possiple transitional probabilties
# that can reach pos_i, * P(pos).j
function predict(grid::Array{Array{Float64,1}}, dir::Direction)
tmp_grid = deepcopy(grid)
for row in 1:6
for col in 1:5

```

```

if(notblocked(grid, (row,col)))
val=sum([x[2] for x in transprob(grid, (row, col), dir)])
tmp_grid[row][col]=val
end
end
end
tmp_grid

end

```

1.3 Evidence Contional Probabilty

$$Evidence\ Contional\ Probability(Pos_i, Evidence) = \prod_{dir=W}^{Directions} Sense(evidence[pos_i dir], actual[pos_i + dir])$$

This is the equation we use to get the evidence contional probability: it's the product of each the evidencne's value at a direction times what's actually in the value of the direction. So if *Left* has open, but evidencne says it's closed, it's 0.2. Taking the prodcut of all direction's sensed value and actual value, it will result in the Evidencne Contional Probabilty at Pos_i given *Evidence*

```

# Given posistion and evidencne return probabilty of being in that posistion
const Sense=Dict(OPEN=>Dict(OPEN=>.8, CLOSED=>.2), CLOSED=>Dict(OPEN=>.25, CLOSED=>.75)

function evidence_Probabilty(grid::Array{Array{Float64,1},1}, pos::Tuple{Int64, Int64})
prod = 1
for i in 1:4
tmp_pos = move(pos,AllDirects[i])
block = notblocked(grid, tmp_pos)
if (block)
prod*= Sense[OPEN][evidence[i]]
else
prod*= Sense[CLOSED][evidence[i]]
end
end
prod
end

```

1.4 Filtering

$$pos_{s+1,i} = \frac{Evidence\ Conditional\ Probabtily(pos_{s,i},evidecne) \cdot P(pos_{s,i})}{\sum_{pos}^{all\ posistions} Evidence\ Conditional\ Probabtily(pos_{s,i},evidecne) \cdot P(pos_{s,i})}$$

$$Filtering(Grid, Evidence) = \left\{ pos_i \in Grid \mid \frac{P(pos_i) \cdot Evidence\ Conditional\ Probability(pos_i, Evidence)}{\sum_{pos}^{all\ posistions} P(pos_i) \cdot Evidence\ Conditional\ Probabtily(pos_i, evidecne)} \right\}$$

Filtering is a transformation upon the grid: each value gets transformed by the expression $\frac{P(pos_i) \cdot Evidence\ Conditional\ Probability(pos_i, Evidence)}{\sum_{pos}^{all\ posistions} P(pos_i) \cdot Evidence\ Conditional\ Probabtily(pos_i, evidecne)}$, whcih for purposes of making it easier to talk about, will be expressed as *Filter Step*($pos_i, Evidence$). *Filter Step* is conditional probabiltly of each point times what the point was previously, and then dividng it by the sum of all points on the grid. This operatoin is $O(n)$, although more accurately it's $O(2n)$ because there's a minimal of iterating through each value twice.

```
# Get the evidecne contional probabiltly of each posistoin*Pos(s_i)
# Then divide each posistion with the evidnece conditonal probabiltly
function filter(grid::Array{Array{Float64,1},1}, evidence::Tuple{SquareType, SquareType})
tmp_grid = deepcopy(grid)
for row in 1:6
for col in 1:5
if(notblocked(grid, (row,col)))
tmp_grid[row][col]*=evidence_Probabiltly(grid, (row, col), evidence)
end
end
end
total_sum = sum(sum(tmp_grid))
# println("SUM: ", total_sum)
tmp_grid / total_sum
end
```

1.5 Smoothing

SmoothingPart(*Grid*, *Previous*, *Direction*, *Evidence*) =

$$\{pos_i \in Grid \mid P(Pos_i) \cdot \sum_{(Pos_j, DriftProb)}^{Transiton\ Probability(pos_i, direction)}$$

$$P(Pos_j) \cdot DriftProb \cdot Evidence\ Conditional\ Probabtily(Grid, Pos_j, Evidence)\}$$

$$\text{Smoothing}(\text{Grid}, \text{Previous}, \text{Direction}, \text{Evidence}) = \left\{ \text{pos}_i \in \text{Grid} \mid \frac{\text{SmoothPart}(\text{Grid}, \text{Previous}, \text{Direction}, \text{Evidence})}{\sum_{\text{pos}}^{\text{All Positons}} \text{Smoothpart}(\text{Grid}, \text{Prevoius}, \text{Direction}, \text{Evidence})} \right\}$$

Smoothing invovles

```

# Get the transitional probabilty of a point going OUT, not in
# an it's conditional probabilty, with it's initial probabilty
# returns 2 things: B at pos, and B*p(s)
function smoothpart( last_grid::Array{Array{Float64,1}}, Bgrid::Array{Array{Float64,1}}
parent_pos=transprob(grid, pos, dir, true )
x=0
# for i in parent_pos
for (tmp_pos, prob, drift) in parent_pos
# tmp_pos = i[1]
# prob = i[2]
# drift = i[3]
y=evidence_Probabilty(grid, tmp_pos, evidence)* Bgrid[tmp_pos[1]][tmp_pos[2]]* drift
x+=y
end

(x,  x *last_grid[pos[1]][pos[2]])
end

# Get the smoothing part for each posistion in grid
# Then divide the whoel grid by the sum of it's parts
function smooth( grid::Array{Array{Float64,1}}, last_grid::Array{Array{Float64,1}}, Bg
SP = deepcopy(grid)
B = deepcopy(Bgrid)
for row in 1:6
for col in 1:5
if(notblocked(grid, (row,col)))
val=smoothpart(last_grid, Bgrid, evidence, dir, (row,col))
B[row][col] = val[1]
SP[row][col] = val[2]
else
B[row][col] = 0
SP[row][col] = 0
end
end
end
end

```

```

# println("SUM: ", sum(sum(SP)))
SP/=sum(sum(SP))
# print_grid(SP); println(); print_grid(B); println()
(SP, B)
end

```

2 Results

The code outputs the following:

julia SUBMIT.jl

3 Screenshots

```

Initial Location Probabilities
1.17  4.17  4.17  4.17
4.17  ##### 4.17  4.17
4.17  ##### 4.17  4.17
4.17  ##### 4.17  4.17
4.17  ##### 4.17  4.17
4.17  ##### 4.17  4.17

Filtering after Evidence [0, 0, 0]
1.62  1.62  1.62  1.62
1.62  ##### 1.62  1.62
1.62  ##### 1.62  1.62
1.62  ##### 1.62  1.62
1.62  ##### 1.62  1.62
1.62  ##### 1.62  1.62

Prediction after Action W
2.76  1.62  4.12  2.76
1.62  ##### 18.55  1.62
1.62  ##### 22.15  1.62
1.62  ##### 22.15  1.62
1.62  ##### 22.15  1.62
1.62  ##### 22.15  1.62

Filtering after Evidence [1, 1, 0, 1]
1.23  1.9  4.82  0.84  0.1
0.16  ##### 1.23  0.84
0.16  ##### 1.23  0.84
0.16  ##### 1.23  0.84
0.16  ##### 1.23  0.84
0.16  ##### 1.23  0.84

Prediction after Action W
1.14  1.54  3.79  1.64  0.23
0.16  ##### 0.8  0.13
0.16  ##### 42.33  0.13
0.16  ##### 0.88  0.6
0.16  ##### 0.88  0.6
0.16  ##### 0.88  0.6

Filtering after Evidence [1, 1, 0, 1]
1.55  1.25  1.87  0.48  0.83
0.01  ##### 0.11  0.83
0.01  ##### 33.52  0.83
0.01  ##### 0.12  0.83
0.01  ##### 0.12  0.83

Last position Smoothing with Evidence [1, 1, 0, 1] and north
1.59  0.54  2.12  0.14  0.83
0.06  ##### 0.27  0.83
0.01  ##### 0.18  0.83
0.01  ##### 0.18  0.83
0.01  ##### 0.18  0.83
0.01  ##### 0.18  0.83

[1] @zakabanda 1:Zaki the Ahmed 2:Muaz Alhaider-

```

```

1.62 1.62 1.62 5.2 1.62
1.62 ##### 5.2 5.2
1.62 ##### 0.25 18.63 5.2
1.62 ##### 5.2 5.2
1.62 ##### 6.82 18.63 5.2
1.62 1.62 5.2 5.2 1.62

Prediction after Action W
2.76 1.62 4.12 2.7 1.82
1.62 ##### 18.35 1.82
1.62 ##### 12.15 5.2 1.56
1.62 ##### 12.26 1.56
1.62 ##### 13.8 5.2 1.82
2.76 4.12 4.66 4.43 1.82

Filtering after Evidence [1, 1, 0, 1]
1.23 1.9 4.82 0.84 0.1
0.16 ##### 2.25 0.33
0.16 ##### 53.26 0.43 0.84
0.16 ##### 8.82 0.86
0.16 ##### 16.33 0.43 0.83
1.23 4.82 1.45 1.38 0.1

Prediction after Action W
1.14 2.55 1.75 0.83 0.23
0.16 ##### 0.8 0.33
0.16 ##### 55.33 18.27 0.1
0.16 ##### 0.88 0.6
1.13 ##### 16.8 1.39 0.14
1.21 4.88 0.93 0.23 0.22

Filtering after Evidence [1, 1, 0, 1]
1.55 1.25 1.87 0.48 0.82
0.01 ##### 0.11 0.83
0.01 ##### 81.92 0.37 0.8
0.01 ##### 0.12 0.81
0.09 ##### 7.3 0.12 0.8
0.8 7.81 0.15 0.83 0.83

Last position Smoothing with Evidence [1, 1, 0, 1] and north
1.59 0.94 2.12 0.14 0.83
0.86 ##### 0.17 0.8
0.01 ##### 84.08 0.15 0.8
0.01 ##### 0.13 0.8
0.91 ##### 6.85 0.87 0.8
0.17 2.12 0.44 0.87 0.8

Second Last position smoothing with Evidence [1, 1, 0, 1] And west
0.81 0.94 0.95 1.94 0.86
0.16 ##### 0.19 0.13
0.01 ##### 1.55 88.99 0.11
0.01 ##### 0.89 0.83
0.86 ##### 8.72 1.86 0.83
0.78 0.49 2.16 0.52 0.82
[END]
[AI] @rakabanda 1:Zaki the Aband 2:Mas Alhaidar-

```

"rakib@uammi" 13:53 11-Nov-20