Problem is: robot does not know if it is in E1 or E2.

**Assuming**: some historical data available at the time = 0. E1 & E2.

Verify if making good analogy, how fast convergence to the right experience (validation of theory).

 (If analogies are wrong, it should take more time to converge.)

**Big question:** what is the best way to get best analogy at the beginning?

Initial data:

E1= Believe in Exp1 initial analogy P[E1(0)]

Under E1, the success rate of the Action A is = 95%

E2= Believe in Exp2 initial analogy P[E2(0))

Under E2, the success rate of the Action A is = 65%

A=Observed action success rate, to be collected during the iterations

Smoothing of the observed success rate using NLP: <https://en.wikipedia.org/wiki/Additive_smoothing>

1st iteration

Given observed action success rate as

The believe of the Exp1 is updated as

New believes

The believe of the Exp2 is updated as

New believes

2nd iteration

Given observed action success rate as

The believe of the Exp1 is updated as

New believes

The believe of the Exp2 is updated as

New believes

Ith iteration

Given observed action success rate as

The believe of the Exp1 is updated as

New believes

The believe of the Exp2 is updated as

New believes

Scenarios:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| R | 10 | -100 | -100 | 5 | D100 |
| 10 | 15 | -100 | -100 | 5 | 80 |
| 10 | 15 | 15 | 15 | 15 | 60 |
| 0 | 18 | 20 | 30 | 50 | 55 |
| -10 | -10 | 10 | 10 | 34 | 45 |

Experience1 90%  (Action success 95%)     Localisation (100%)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| R | 10 | -100 | -100 | 0 | D100 |
| 10 | 15 | -100 | -100 | 0 | 80 |
| 10 | 10 | 0 | 0 | 0 | 60 |
| 0 | 18 | 20 | 30 | 50 | 55 |
| -10 | -10 | 10 | 10 | 34 | 45 |

Experience2 10% (Action success 65%)   Localisation (100%)

How to do the Verification?

Unknown Situation 1   (Action success ??%):

Skill: expiernces.

How to select skill:

Updating confidence based on fuzzy reasoning.

S(1,2,3)

S\_policy(1,2,x,y,5) 2 reward -1

**State compression problem**: efficient way of representing state. Absolute position (fast). (Memory of humans)

**Find similarity of states:** how to compare similarity. If similar reward it may be the same situation. Estimate reward and have some estimation. If reward is low, maybe not do so much calculation. If the other way around.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X (state x) | y |  |  |  |  |
| z |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Unknown Situation 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Calculate Q-table using Bellman equation and epsilon greedy algorithm

1. Define formually in maths the 1+1

Decompose and recombine experience (human). Verify and experience repeat.

E = Q table

action1, action2, action3

State 1 : 1 2 2.3

-Table 3

-Circle detection & MDP.

**Research questions:**

\* If newSittuation is either E1 or E2, this meta reasoning using analogy is always better than random just my pure logic. Prove by using simulation and many experiments averaging them.

\* The real interesting problem is if newSittuation is not E1 or E2 but resemblances somewhat to E1 or E2.

\* If newSittuation is a linear combination of E1 and E2, can the algorithm realize this and make better informed decision and create new E3?

**Strategies:**

-Change believe of E1 to E2 and act according to their policies dynamically or select a policy based on initial analogy and stick with it until the end.

State[0] = [map, robotPose, robot\_localization\_state]

E1 = Q\_table\_1

E2 = Q\_table\_2

Def Analogy(State[0], Experience):

Return analogical\_distance 🡪 P[E1(0)]

Iteration 0:

Given State[0], E1 and E2.

P[E1(0)] = Analogy(State[0], E1) [Probab of E1 in t=0]

P[E2(0)] = Analogy(State[0], E2) [Probab of E2 in t=0]

= MDP E1 [Prob of success of action in context E1]

= General given value. [Prob of success of action in any context]

Iteration 1:

Given State[1], E1 and E2.

=