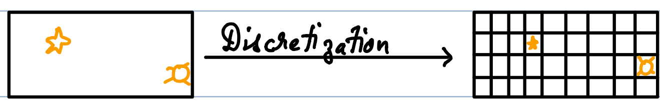
RL in Continuous Space

So far, we have seen Monte Carlo & TD learning methods. The algorithms create a mapping for (state, action) pairs with their corresponding rewards.

A major problem with such an approach occurs when we have continuous state or action space. For eg. Instead of grid world, we have an infinite coordinate space for the agent to move in or a dart throwing agent the actions are angles & force with which the dart needs to be thrown.

Discretization :-

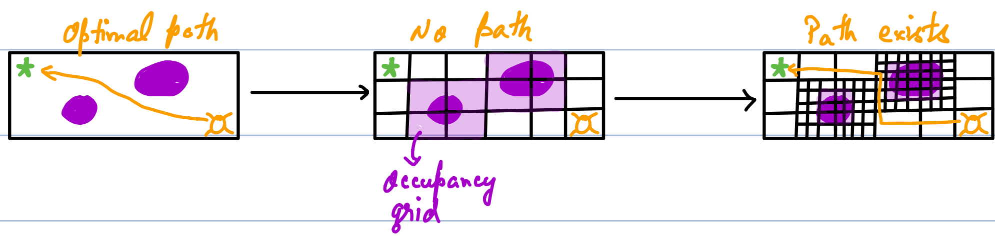
Converting a continuous space into a discrete space by rounding off the values.



Similarly action can also be discretized to use algorithms known so far directly.

Note :- if the occupancy grid is too large it may leave no path for the agent to navigate. To overcome this, we can have smaller grids wherever required.

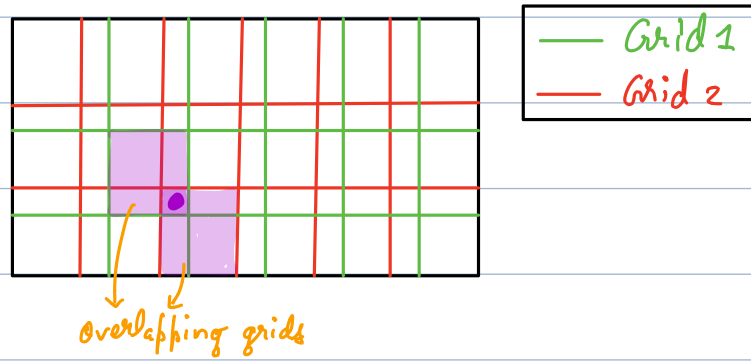
Eg.



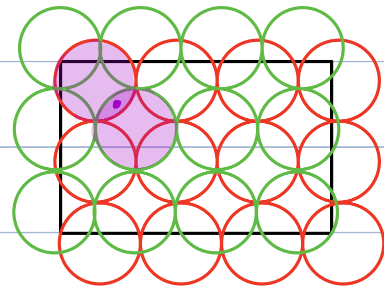
Discretization can be uniform or non-uniform depending on the application.

Tile Coding :-

Similar to discretization, tile coding divides the continuous space into discrete space. But for better generalization, TC uses multiple grids over the same space with small offsets. This allows the same discrete state to lie under multiple tiles.

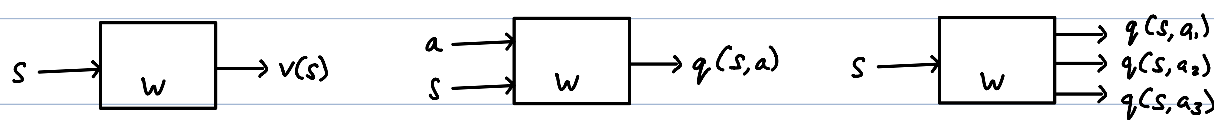


Coarse Coding :- Similar to tile coding but uses circles instead of tiles which help in fine-tuning by calculating the distance of the point from the center of a circle & applying Radial Basis Functions.



Function Approximations :-

Instead of dividing the continuous space into discrete space, another approach is to approximate the value returned for the state, action space.



The intermediate approximation function could be :-

1. Linear Function Approximation :- Dot product $\Rightarrow$ $\tilde{v}(s) = x(s)^T \cdot w$

feature vector of s — weight vector

The weights can then be learned by having the (actual value - output value) loss being backpropagated. $J(w) = E_\pi\left[\left(v_\pi(s) - \tilde{v}(s)\right)^2\right]$

2. Kernel Functions:-

Instead of having linear features, they may be transformed into non-linear features to capture more complex relations.

Eg.

Feature Vector

$$X(S) = \begin{pmatrix} x_1(s) \\ x_2(s) \\ \vdots \\ x_n(s) \end{pmatrix}$$

where

Kernel Functions

$x_1(s) = s$

$x_2(s) = s^2$

$x_3(s) = s^3$

$\vdots$

⊛ Radial Basis Functions can be used as non-linear Kernel Functions.

3. Non-Linear Function Approximation:-

Non-linear functions (Activation functions) are used to introduce non-linear relation.

non-linear function

$$\tilde{v}(s) = f(\, X(s)^T \cdot W)$$

This is what neural networks do.