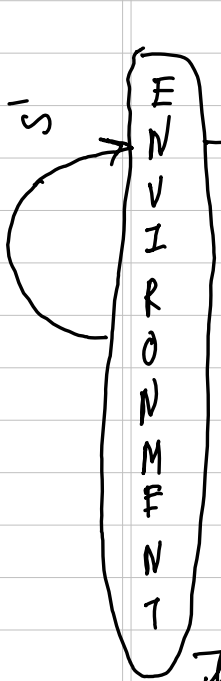
Model in RL

A model is anything that agent can use to predict the environments behaviour

Given an state & action, a model predicts the next state & next reward

stochastic → several possible states with probability (distribution model)

sample model → give only 1 state with highest probability



Next state (s') after (A) is taken is determined by the environment.

But if the environment is not available, this work is done by model

Model can just replace environment.

Planning in RL

Planning is any computational process that takes a model as input and produces or improves policy for interacting with modelled environment

Model $\xrightarrow{\text{Planning}}$ Policy

Learning is when we actually interact with the Environment

Environment $\xrightarrow{\text{Learning}}$ Policy

Planning uses simulated experiances generated by model

Learning uses real experiances

Model based (D.P., huristic search) rely on Planning
Model free (MC, TDL, etc) rely on learning

Learning	Planning
Real experiance	Simulated experiance
Model free	Model based
May be slower depending on environment	faster

In many cases, learning algorithm can be used for planning.

eg Planning & learning where everything is the same except environment is replaced by model

Model free vs Model based

Model based methods work well when the model is known.

In such cases (toy problems) the model and environment are the same

But in real life cases, if the knowledge about the exact environment is not known then model ≠ env.

This may be due to incorrect assumptions

(D.P.) So when $P(s'|s,a)$ & $R(s,a)$ are estimates and not true values, model ≠ env.

Also in some applications like recommendation systems and financial markets, the environment may change over time.

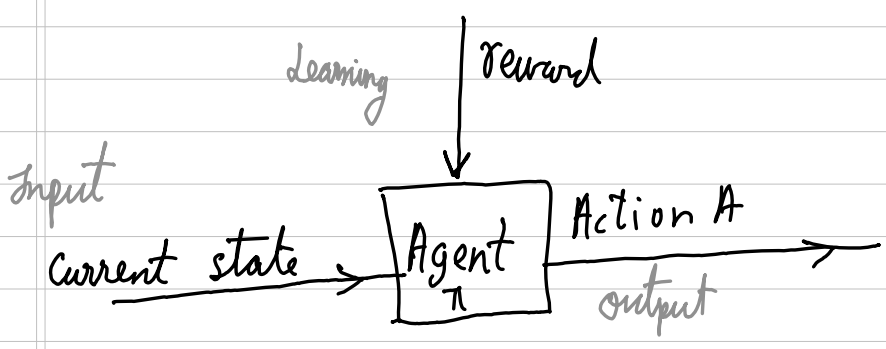
In such cases, the model used is just an estimate or approximation of the environment.

Model free methods on other hand learn directly from the environment

Model may be incorrect because →

1. Environment is stochastic and only a limited amount of experience is available
2. Model has learned using a function approximator that failed to generalize
3. Environment has changed and new behaviour has not been observed yet.

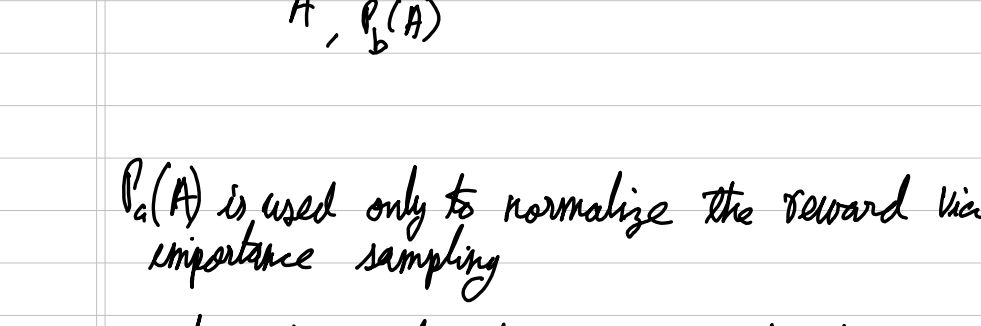
Planning Vs learning



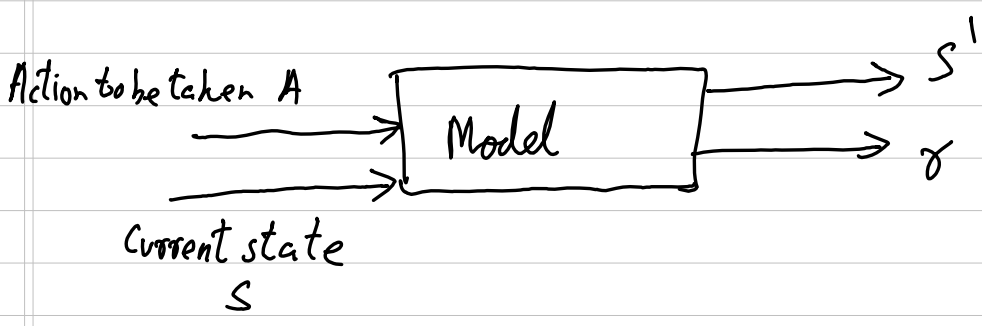
Agent is a black box with input S output A. γ is used for learning

S & γ need to be supplied by environment or someone else

In learning RL →

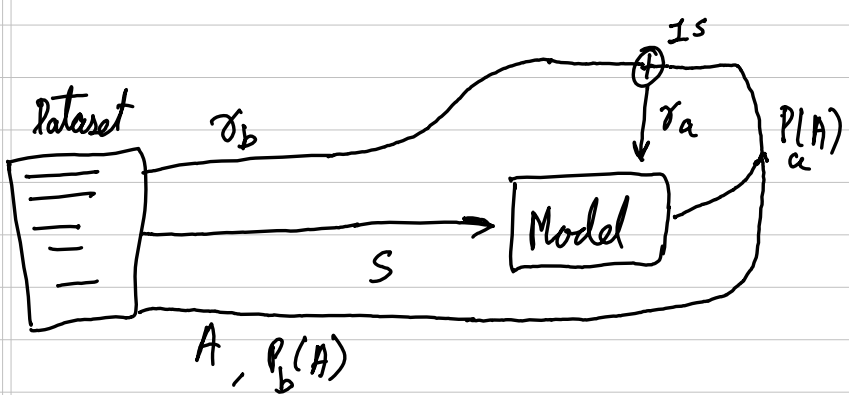


In Planning RL →



A model is used to <u>simulate</u> the environment

Note → for off policy historical learning
There is no environment, just dataset



$P_a(A)$ is used only to normalize the reward via importance sampling

Action has already been predecided by b

Direct & Indirect RL

* what we can do with real experience

① Model learning → Improve model to match it with the environment. (to cope environment changes)

② Direct RL → Improve value functions and policy from environment (learning)

③ Indirect RL → Improve value function and policy from model (Planning)

Direct methods are not affected by bias in model design

But real world interactions may be costly

With a model, agent can "Imagine" ie. simulate transitions between states without needing to interact directly with the environment

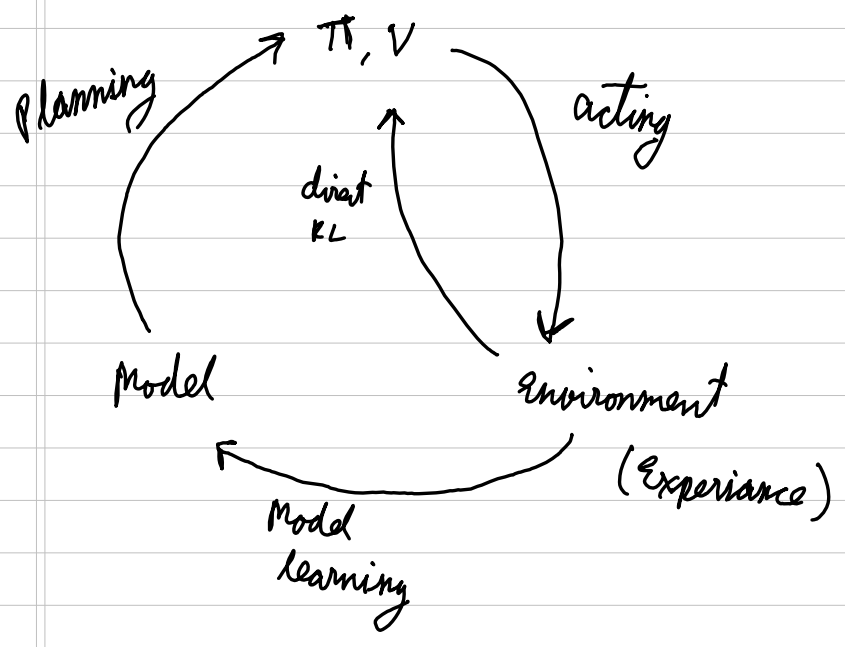
If all these 3 methods are combined, then there will be increase in efficiency

Agent will learn from real environment as well as simulated models
So the agent can update policy multiple times on single real update → faster learning

Dyna - Q

Simple Architecture integrating major functions of an online planning agent

It combines model free & model based techniques.



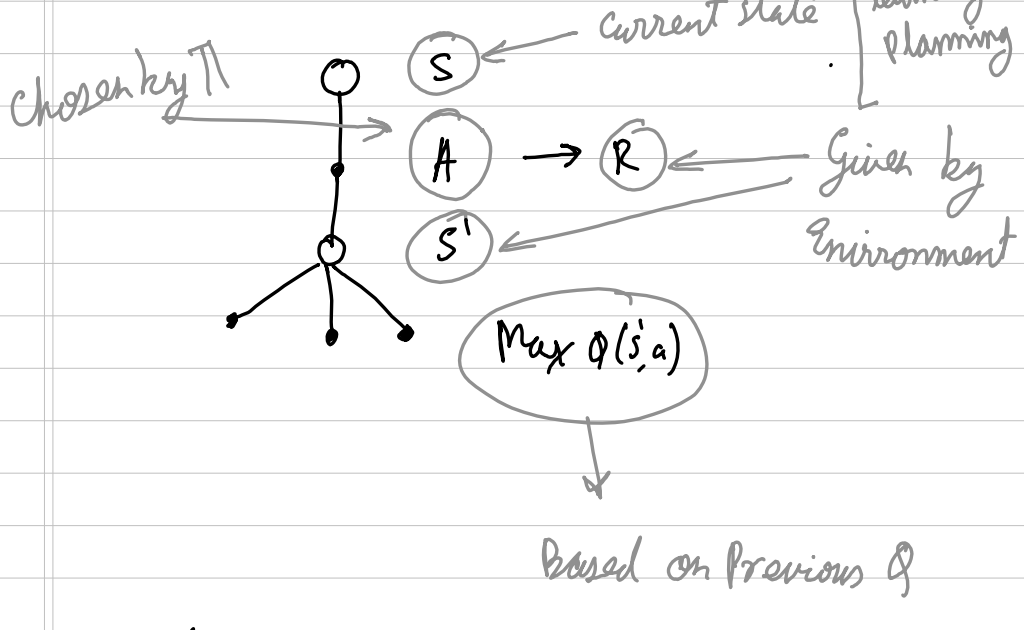
step 1) Take Action & learn from real Experiance

Agent selects action based on π (ε-greedy)

Take action observe reward & new state

Update π using Q learning

current state is the state as returned by the last learning phase and NOT planning phase



step 2) Update the Model

record S' & R to update P(S, S', a) & R(S, a)

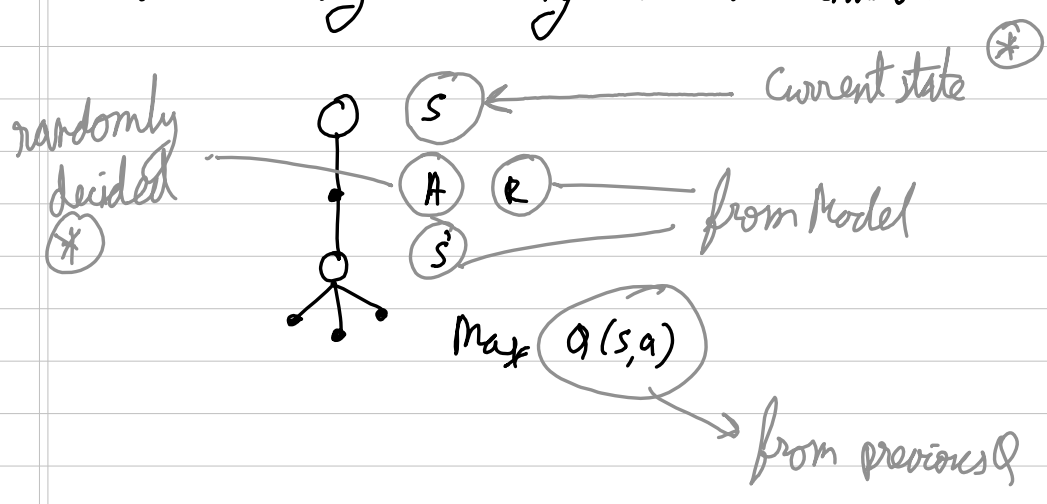
(a is action taken)

step 3) Simulate experiance (Planning)

Agent generates number of simulated experiances using Model

for each simulated experiance, update Q-value as if it were a real experiance

This step allows the agent to make rapid progress by learning from hypothetical experiances without actually interacting with the environment



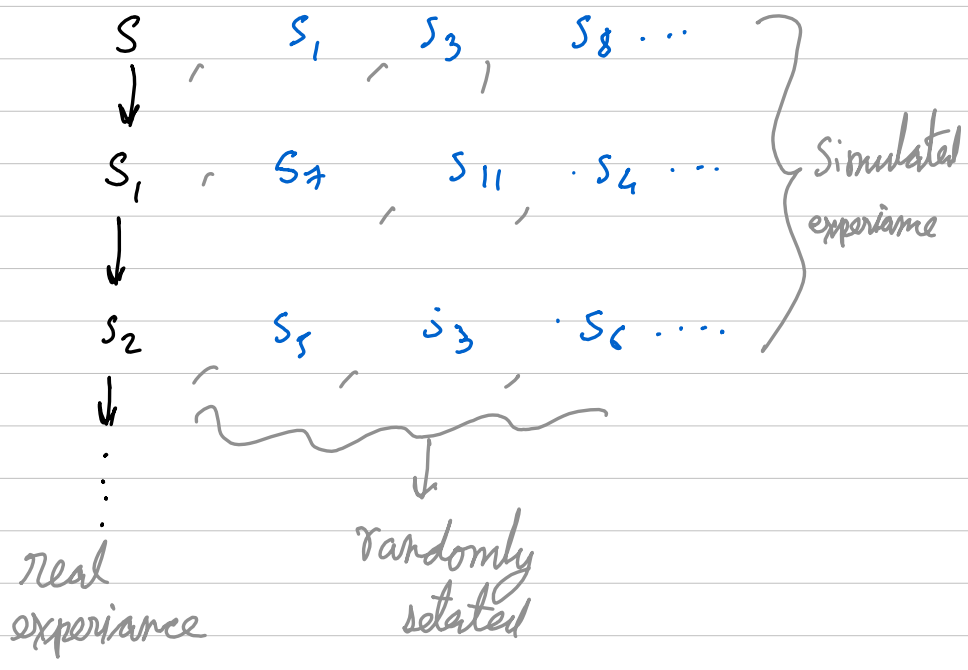
* → Randomly selected from past state action pairs (S, a)

Repeat Planning N times

Then go back to step 1

Note → The current state for learning remains same as it was after the most recent real (environmental) experiance, not the state from any of the planning steps.

The planning steps only modify the Q values and not agents current state



Dyna Q +

Designed to improve exploration

for Non stationary environments where r, P changes over time

Introduces exploration bonus to agent to revisit the states and actions that it has not visited recently in simulated experiance

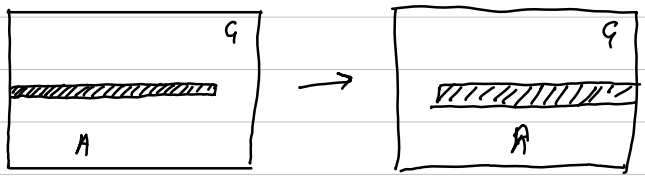
Model keeps track of each state - action pair of how many time steps have elapsed since it was last tried in a real interaction.

Special bonus is assigned $r \rightarrow r + k\sqrt{\tau}$

(k: Parameter; τ: Transition not tried for τ rewards in real world)

Has cost for revisit, but worth it for changing environment

eg Grid world barrier moves



In such cases, revisiting states is helpful

Prioritized Sweeping

Instead of updating random state action pairs during planning, prioritized sweeping selectively updates only those pairs with highest potential for learning.

These state action pairs are prioritized based on how much their Q values could change

By focusing only on these high impact updates, prioritized sweeping achieves faster learning with fewer updates.

TD error → $\left| \gamma + \gamma \max_a Q(s', a') - Q(s, a) \right|$

estimated goodness — current goodness

High T.D. error means updating the Q value could significantly improve the agents knowledge

Once a (s, a) is visited,

calculate TD error for (s, a)

Calculate TD error for Predessors

Add all to priority queue with priority = TD error (Thresholding can be done)

In planning phase, instead of randomly selecting actions, take actions from priority queue

For each predessors, when they come in planning phase for update, calculate the Q of their predessors too.

A predessor state are the states that could transition into the current state under some action.

Priority sweeping has been found to dramatically increase the speed of convergence.