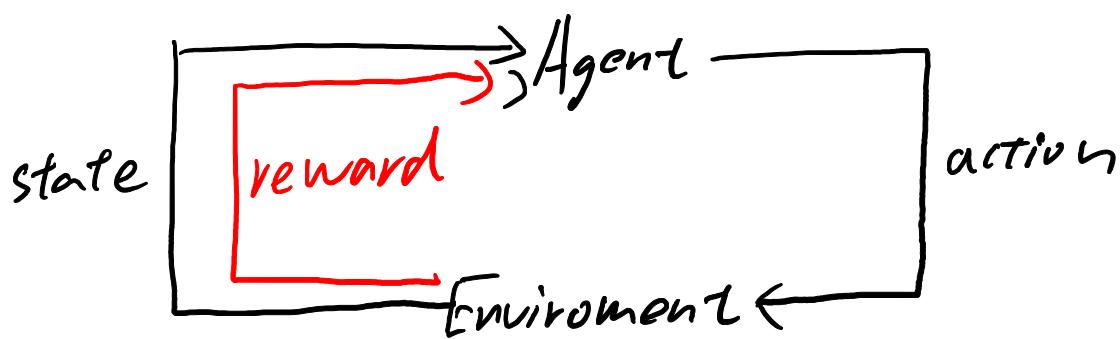


- Supervised learning: fit + predict
- Unsupervised : fit + transform
- RL has no specific obj, but to max reward



- RL is a balance of collecting data and keep an accurate measurement of significances
explore & exploit

- Three strategies for explore & exploit

- Epsilon-greedy

- Pseudo code:

- Initial high value $\theta \gg$ true mean

- $p = \text{random}()$

if $p < \epsilon_{PS}$:

do random actions

else:

do the current best action j which has highest mean among all possible action

update action

- UCB1

• Chernoff-bound:

$$P(|\bar{x} - \mu| \geq \epsilon) \leq 2 \exp(-2\epsilon^2/N)$$

$$\Rightarrow X_{UCB-j} = \bar{x}_j + \sqrt{2 \frac{\ln N}{N_j}}$$

N is total # of actions used so far

N_j is # of time action j is used

This encourage taking actions that haven't been taken many times.

• Pseudo code:

for n in range(N):

$$j = \operatorname{argmax} \left[\text{actions.mean} + \sqrt{2 \frac{\ln n}{N_{action}}} \right]$$

do action j
update action j

- Bayesian / Thompson sampling

- $\bar{X}_j \sim N(\mu, \sigma^2/N_j)$, Assume each action is Gaussian
 $\max p(\theta|X)$, use data X to guide us to find best θ in parameter space

$$P(\theta|X) = P(x|\theta) \cdot P(\theta)$$

posterior = likelihood \times prior

- different to posterior & prior in ML, which is

$$P(y|X) = P(x|y) \cdot P(x)$$

$\max P(y|X)$: MAP

$\max P(x|y)$: ML

- Special pair of $P(x|\theta)$, $P(\theta)$ are needed

$$\cdot X \sim N(\mu, \lambda), \mu \sim N(\mu_0, \lambda_0^{-1})$$

$$\lambda = \lambda_0 + eN$$

$$\mu = \frac{\mu_0 \lambda_0 + e \sum x_n}{\lambda_0 + eN}$$

Randomness controls sampling

More times an action is taken, the more it will converge to true stat

- Pseudo code

Initialize $\mu = 0$, $\lambda_0 = 1$, $\ell = 1$

for n in range(N):

 take a sample from each action

$$\text{sample} = \frac{N(0, 1)}{\lambda} + \mu$$

 find the max of those sample

 update: $\lambda = \lambda + \ell n$

$$\mu = \frac{\lambda \mu + \ell \sum x_n}{\lambda + \ell N}$$

- Non stationary Bayesian

- Components of RL

- Agent : thing that plays the game
- Environment : things agent interact with
- State : specific env agent senses
- Action : things agent can do to effect the state
- Reward : consequences of changing state
 - how good the action is
 - just a number
 - instantaneous
- $S(t), A(t) \rightarrow R(t+1), S(t+1)$
- Terminal state : when to finish, bad & good
- Unstable system \Rightarrow infinite states \Rightarrow hard to deal

- Rewards:

- Needs to define how to give rewards
- Tell agent what you want to achieve, not how to achieve.
- Value function : assign value to current state to reflect future.

- Assign correct credit to past actions we care about which "path" to get reward too
- Value: possible future reward

Value func: $E[\text{all future rewards} \mid S_t]$

$$V(S_t) \leftarrow V(S_t) + \alpha(V(S_{t+1}) - V(S_t))$$

- We explore to update Value functions !!!
- Playing the game more, more likely we get to true probability
- Order is important !
- If $V(S_{t+1}) < V(S_t)$, update doesn't help

