UNIVERSITY OF CAMBRIDGE

Machine Learning and the Physical World

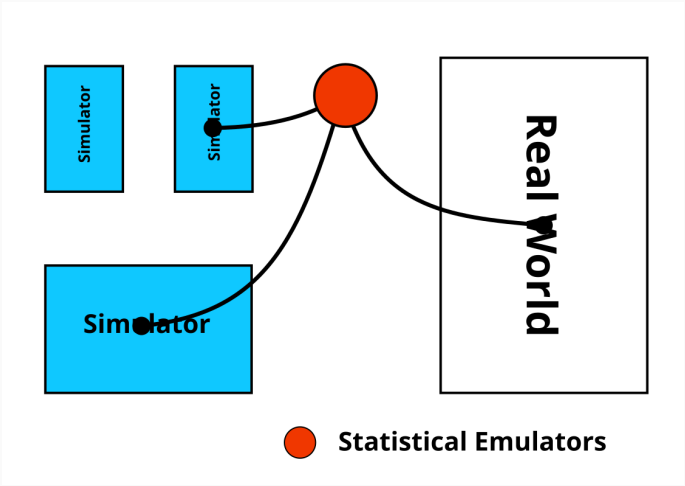
Lecture 9 : Reinforcement Learning

Carl Henrik Ek - che29@cam.ac.uk

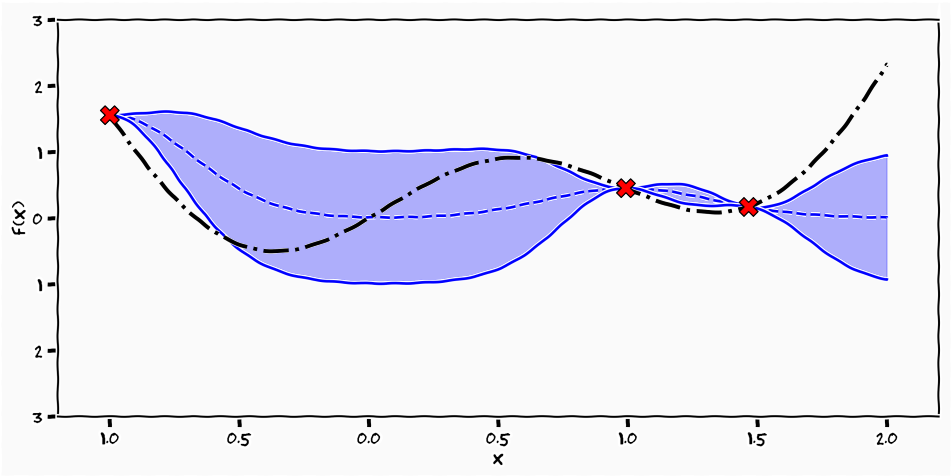
10th of November, 2025

http://carlhenrik.com

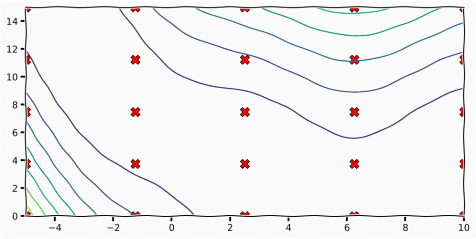
Simulation and Emulation



Emulators



Branin Function

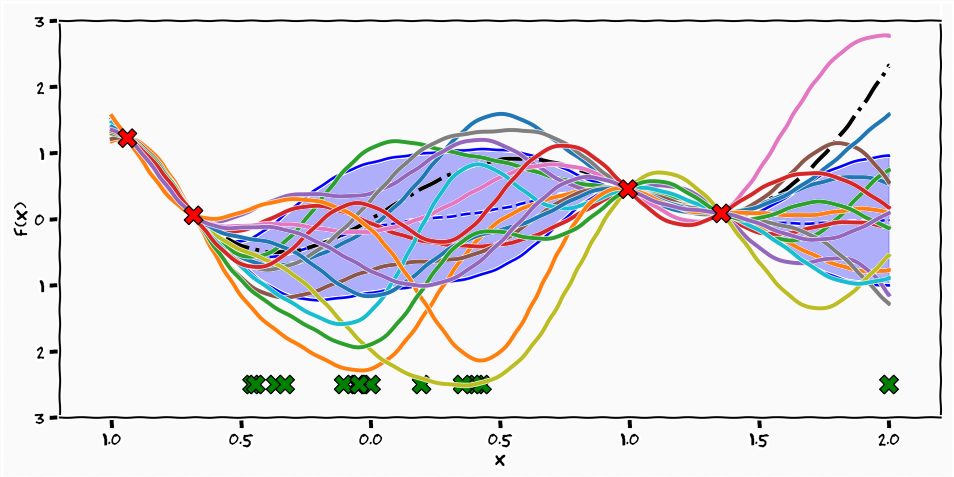


Fit a GP surrogate

$$k(\mathbf{x}_i, \mathbf{x}_j) = k_{\text{Exp}}(\mathbf{x}_i, \mathbf{x}_j) + k_{\text{White}}(\mathbf{x}_i, \mathbf{x}_j) + k_{Bias}$$

$$\mathbb{E}[f(U)] \approx \mathbb{E}[p(\mathbf{f} \mid \mathbf{Y}, \mathbf{X}, \mathbf{X}_{\text{samples}}) = 70.25$$

Surrogate Uncertainty



Data Acquisition

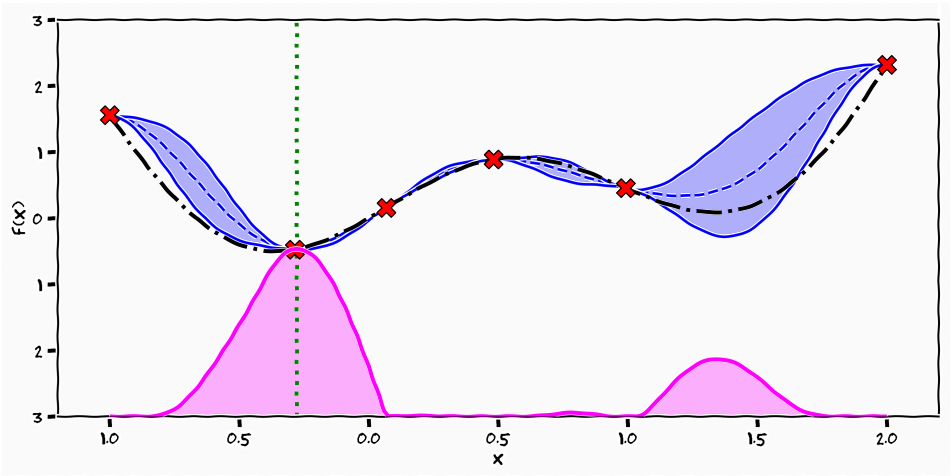
- Define a function that defines the utility of observing each location

$$u(x, f(x^{(*)}), \mathcal{M}_n)$$

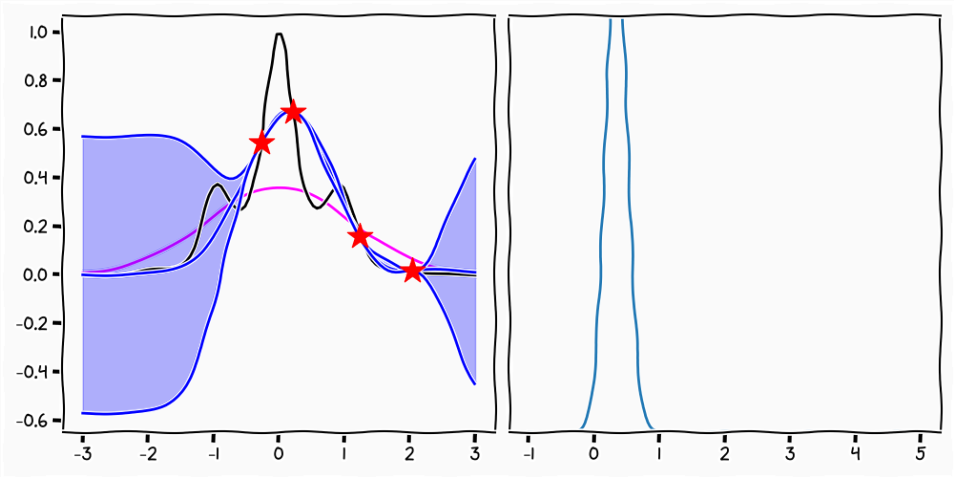
- Define the acquisition function as the expected utility

$$\begin{aligned}\alpha(x; \{x_i, y_i\}_{i=1}^n, \mathcal{M}_n) &= \mathbb{E}_{p(f)}[u(x)] \\ &= \int u(x, f(x^{(*)}), \mathcal{M}_n) p(f \mid \{x_i, y_i\}_{i=1}^n) \mathrm{d}f\end{aligned}$$

Optimisation



Quadrature



Today

November 10, 2025

Reinforcement Learning

Supervised Learning predict output from input

$$\mathcal{D} = \{y_i, x_i\}_{i=1}^{N}$$

$$p(\mathbf{y}|\mathbf{x}) = \int p(\mathbf{y}|\mathbf{x}, \theta) p(\theta) \mathrm{d}\theta$$

Supervised Learning predict output from input

$$\mathcal{D} = \{y_i, x_i\}_{i=1}^N$$

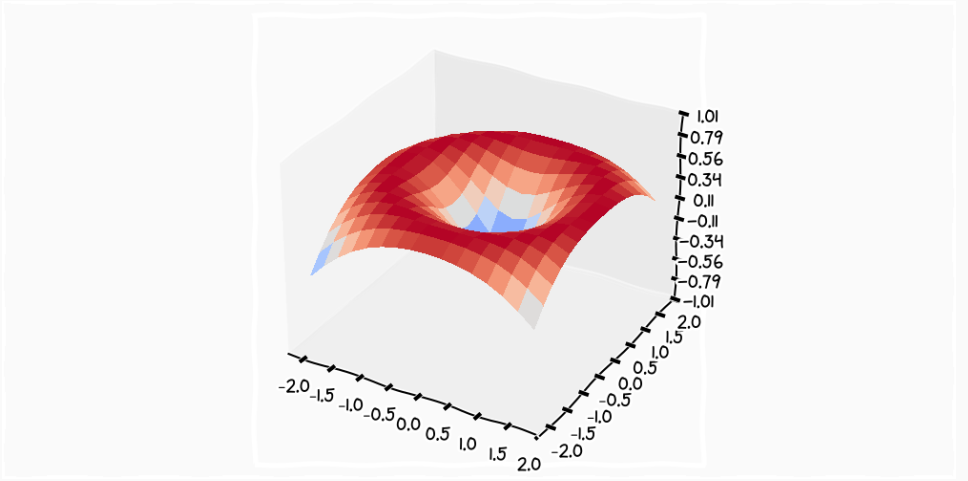
$$p(\mathbf{y}|\mathbf{x}) = \int p(\mathbf{y}|\mathbf{x}, \theta) p(\theta) \mathrm{d}\theta$$

Unsupervised Learning model the data

$$\mathcal{D} = \{y_i\}_{i=1}^N$$

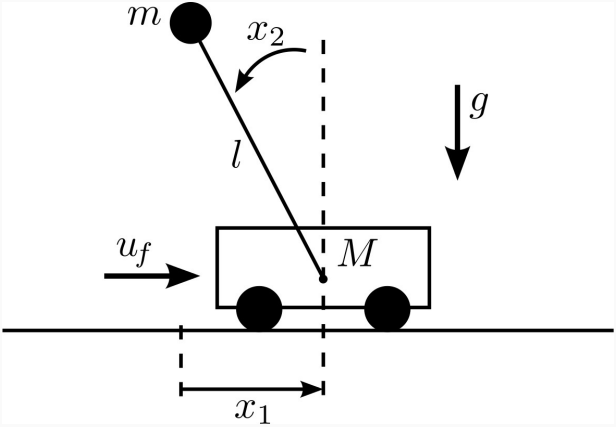
$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{x}, \theta) p(\mathbf{x}) p(\theta) \mathrm{d}\mathbf{x} \mathrm{d}\theta$$

Structure


1.01
0.79
0.56
0.34
0.11
−0.11
−0.34
−0.56
−0.79
−1.01
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

Reinforcement Training

Inverted Pendlum



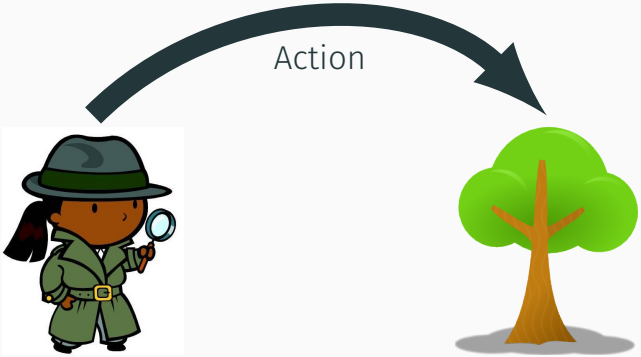
Reinforcement Learning

Can we learn without specifying how the task should be achieved by providing, rewards (positive) and punishment (negative)?

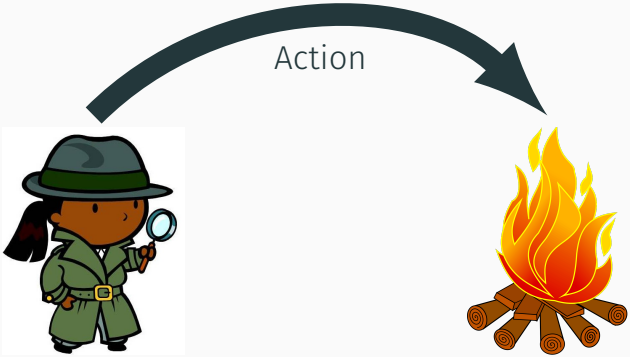
Formalism

Formalism

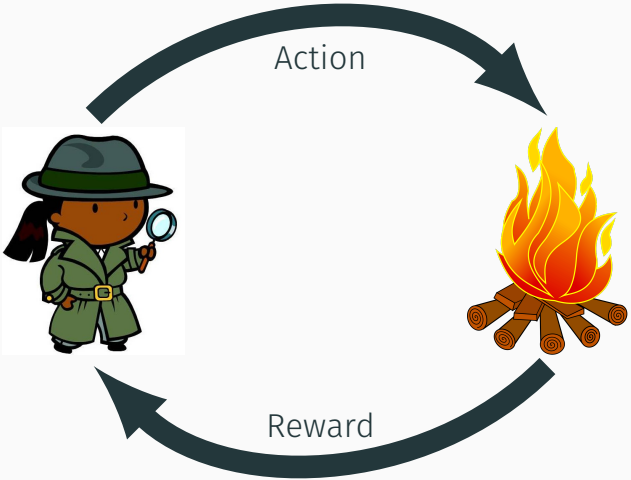
Formalism



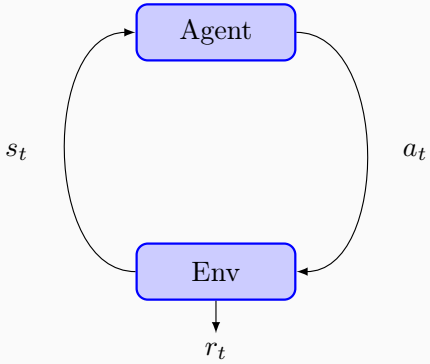
Formalism



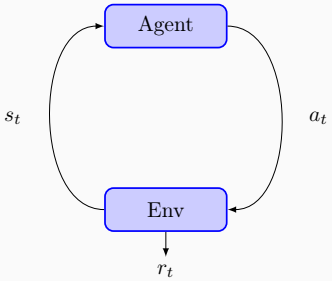
Formalism



The Agent and the Environment



Reinforcement Learning



$$\text{argmax}_{a_0, a_1, \ldots} \sum_{t=0}^{\infty} r_t$$

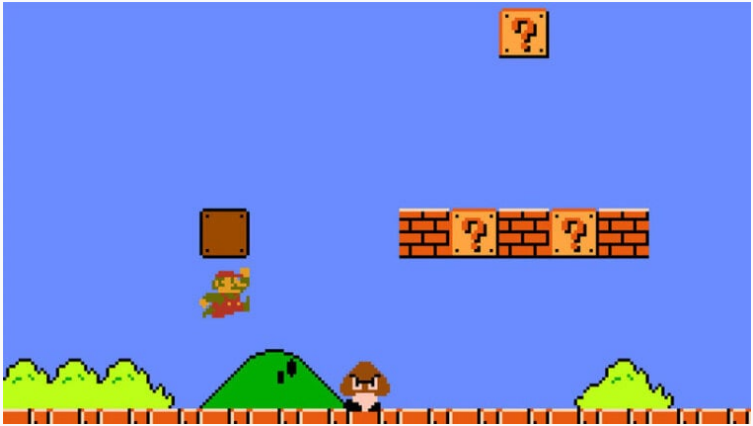
- the agent takes actions in the environment
- actions change the state of the environment
- the state produces a reward

How is this different?

$$p(\sum_{t=0}^{T} r_t \mid s_0, a_0, a_1, \ldots, a_t)$$

Lets just collect lots and lots of data and turn it into a regression problem

Super Mario



When to do RL?

- When the number of possible sequences of input is very large

When to do RL?

- When the number of possible sequences of input is very large
- When acquiring data is hard and expensive

When to do RL?

- When the number of possible sequences of input is very large
- When acquiring data is hard and expensive
- When its hard to define the ”structure” of the problem explicitly as a cost

Markov Decision Processes

Definition (Markov Decision Process)

A Markov Decision Process is a 4-tuple (S, A, T, R)

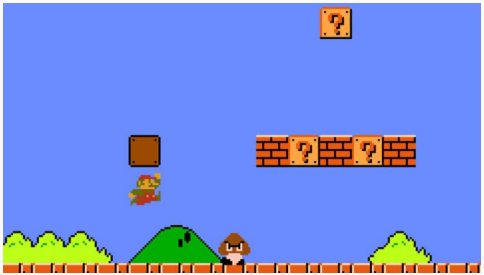
- S is the State space
- A is the Action space
- T the state transition function
- R the reward function

Markov Decision Process

$$T(s_t, a_t) = p(s_{t+1} \mid s_t, a_t)$$

- RL provides solutions to Markov Decision Processes
- We must define our state space in such a way that the environment forms an MDP

State Space



State Space

- position and velocity of Mario
- the score
- number of coins collected
- time remaining
- Goomba position & velocity, state
- ....

Action Space

Action Space

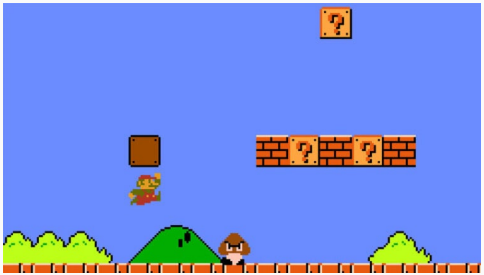
- The controller has

$$a, b, \uparrow, \downarrow, \leftarrow, \rightarrow$$

- Can press certain buttons together

$$A = \underbrace{\{0, 1, \ldots, 8\}}_{\text{directions}} \times \underbrace{\{0, 1, 2, 3\}}_{\text{buttons}}$$

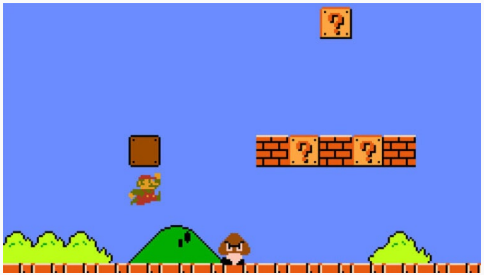
Transition Function



Transition Function

$$T : S \times A \to \Delta S$$

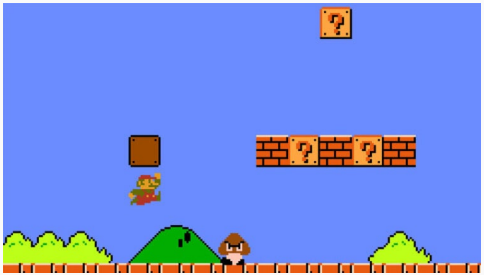
Reward Function



Reward Function

- beating the level $(0, 1)$

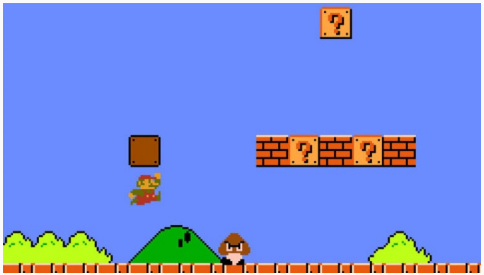
Reward Function



Reward Function

- beating the level $(0, 1)$
- total score

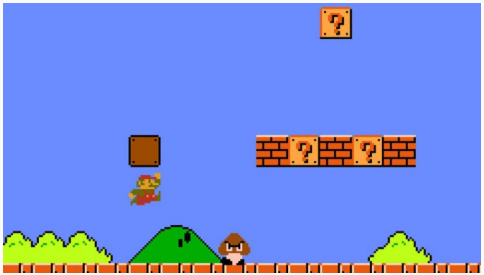
Reward Function



Reward Function

- beating the level $(0, 1)$
- total score
- beating the level $(0, 1) + 0.01 \times \text{score}$

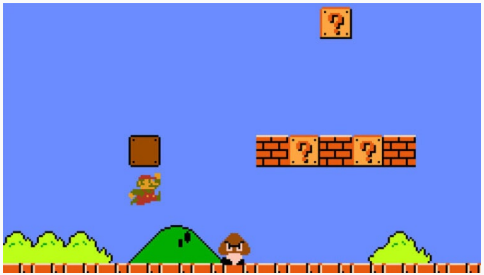
Reward Function



Reward Function

- beating the level $(0, 1)$
- total score
- beating the level $(0, 1) + 0.01 \times$ score
- beating the level/time

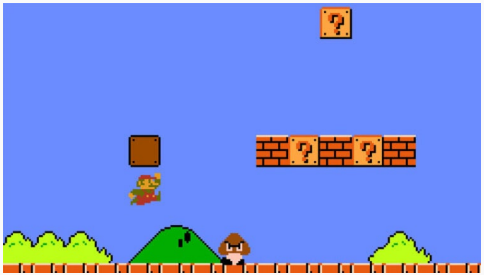
Reward Function



Reward Function

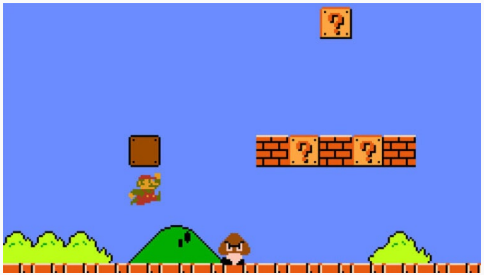
- beating the level $(0, 1)$
- total score
- beating the level $(0, 1) + 0.01 \times$ score
- beating the level/time
- ....

Markov Decision Processes



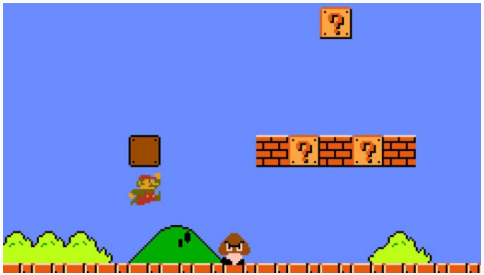
- think carefully about constructing the environment

Markov Decision Processes



- think carefully about constructing the environment
- think carefully about how to structure reward signal

Markov Decision Processes



- think carefully about constructing the environment
- think carefully about how to structure reward signal
- now we have a problem to approach with RL

Reinforcement Learning

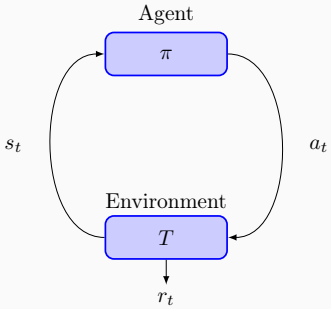
$$G = \sum_{t=0}^{\infty} r_t$$

- Task of RL is to design an agent to maximise return G

$$G = \sum_{t=0}^{\infty} \gamma^t r_t$$

- Task of RL is to design an agent to maximise return G
- We generally have a preference of cookies now compared to cookies later which we introduce by $\gamma \in (0, 1]$ called the discount factor

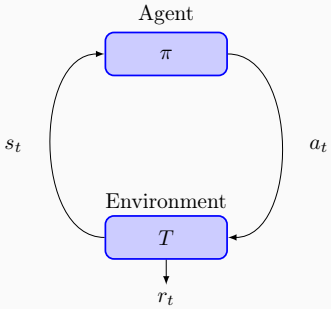
Agent



- The agent acts according to a policy π

$$\pi : S \to A$$

Agent



- The agent acts according to a policy π

$$\pi : S \to A$$

- Goal is to learn the optimal policy

$$\pi_* = \max_\pi \sum_{t=0}^{\infty} \gamma^t r_t$$

The Learning Problem

- We do not know the dynamics of the environment

$$p(s_{t+1} \mid s_t, a_t)$$

The Learning Problem

- We do not know the dynamics of the environment

$$p(s_{t+1} \mid s_t, a_t)$$

- We do not know the policy

$$p(a_t \mid s_t)$$

The Learning Problem

- We do not know the dynamics of the environment

$$p(s_{t+1} \mid s_t, a_t)$$

- We do not know the policy

$$p(a_t \mid s_t)$$

- We not might even know the reward

$$p(r_t \mid s_t)$$

The Learning Problem

- We do not know the dynamics of the environment

$$p(s_{t+1} \mid s_t, a_t)$$

- We do not know the policy

$$p(a_t \mid s_t)$$

- We not might even know the reward

$$p(r_t \mid s_t)$$

- We have no data to learn from

What do we want to learn?

- we want to interact with the environment as little as possible

What do we want to learn?

- we want to interact with the environment as little as possible
- we only want to know the dynamics up to a level where the policy will act

What do we want to learn?

- we want to interact with the environment as little as possible
- we only want to know the dynamics up to a level where the policy will act
- we only need to understand how to act to get a high reward

What do we want to learn?

- we want to interact with the environment as little as possible
- we only want to know the dynamics up to a level where the policy will act
- we only need to understand how to act to get a high reward
- learning the dynamic and policy are just proxies

1. initially we do not have any data

1. initially we do not have any data
2. we usually start off with an initial experiment

1. initially we do not have any data
2. we usually start off with an initial experiment
3. when we get data we can update our dynamics and policy

1. initially we do not have any data
2. we usually start off with an initial experiment
3. when we get data we can update our dynamics and policy
4. how to acquire data from the knowledge that we have?

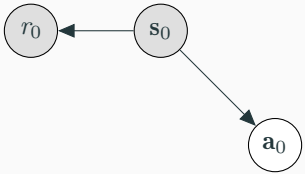
Model Based Reinforcement Learning

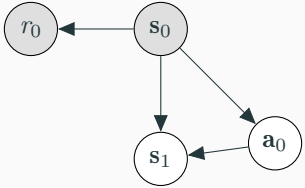
$$p(s_{t+1} \mid s_t, a_t)$$
$$p(a_t \mid s_t)$$

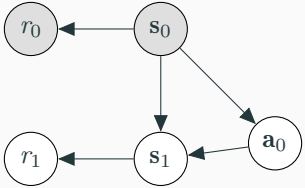
- Model the dynamics and the policy directly
- All the benefits of having uncertainty over each specific component
  - bias with prior knowledge
  - efficient exploration
- Just normal statistical inference :-)

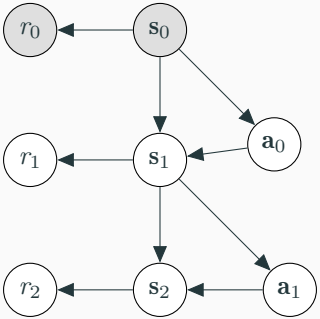

$\mathbf{s}_0$

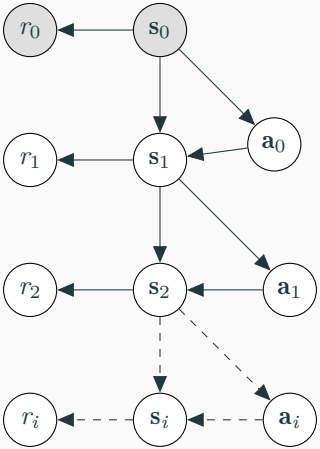

r_0
$\mathbf{s}_0$


r_0
$\mathbf{s}_0$
$\mathbf{a}_0$

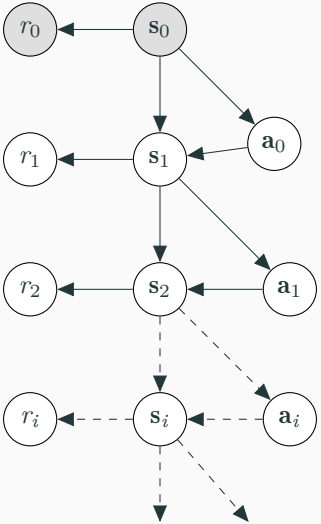

r_0
$\mathbf{s}_0$
$\mathbf{a}_0$
$\mathbf{s}_1$


r_0
$\mathbf{s}_0$
$\mathbf{a}_0$
r_1
$\mathbf{s}_1$

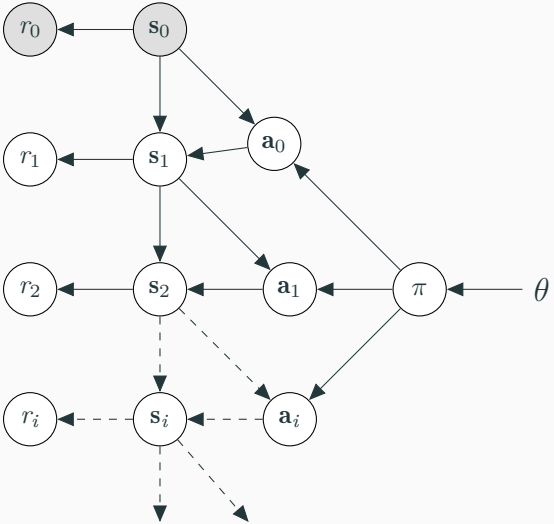

r_0
$\mathbf{s}_0$
$\mathbf{a}_0$
r_1
$\mathbf{s}_1$
r_2
$\mathbf{s}_2$
$\mathbf{a}_1$


r_0
$\mathbf{s}_0$
r_1
$\mathbf{s}_1$
$\mathbf{a}_0$
r_2
$\mathbf{s}_2$
$\mathbf{a}_1$
r_i
$\mathbf{s}_i$
$\mathbf{a}_i$

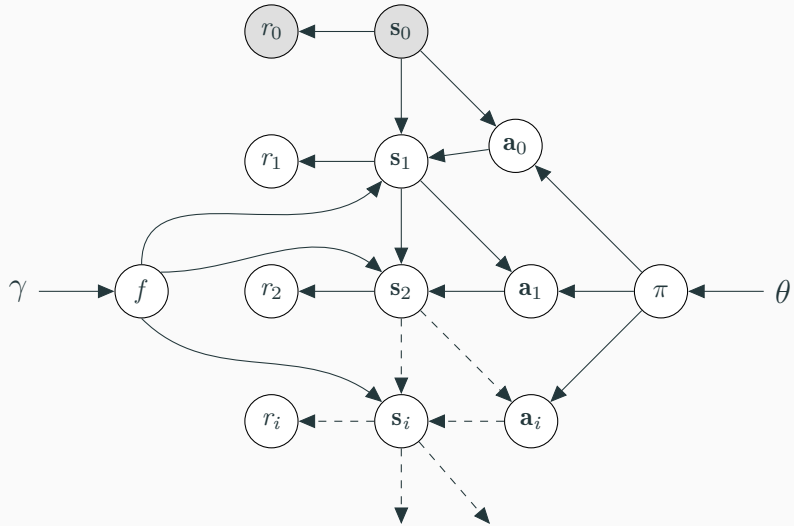
Reinforcement Learning



Reinforcement Learning



Reinforcement Learning



$$
\begin{aligned}
p(\mathbf{s}_{1,\ldots,T}, a_{0,\ldots,T-1}, r_{0,\ldots,T}, f, \pi, \gamma, \theta | \mathbf{s}_0) = \\
p(\mathbf{s}_T | a_{T-1}, s_{T-1}, f) p(a_{T-1} | s_{T-1}, \pi) \ldots \\
\vdots \\
p(\mathbf{s}_2 | a_1, s_1, f) p(a_1 | s_1, \pi) \\
p(\mathbf{s}_1 | a_0, s_0, f) p(a_0 | s_0, \pi) \\
p(f | \gamma) p(\pi | \theta) p(\gamma) p(\theta)
\end{aligned}
$$

- if we want to learn dynamics and policy we need to marginalise out f, π, θ and γ
- very very hard problem as uncertainty propagated a long way

$$\mathbb{E}[r_t] = \int \int p(r_t \mid s_{t+1}) p(s_t + 1 \mid s_t, a_t) p(a_t \mid s_t) \mathrm{d}a_t \mathrm{d}s_{t+1}$$

$$\pi_* = \max_\pi \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid a_t \sim p(a_t \mid s_t)\right]$$

- We want to avoid explicitly learning a policy and a reward model
- Directly maximise the expected reward

Q-Learning

- Define value function V,

$$V_\pi(s_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

Q-Learning

- Define value function V,

$$V_\pi(s_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

- Lets move out the first action and define the function,

$$\tilde{V}_\pi(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

Q-Learning

- Define value function V,

$$V_\pi(s_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

- Lets move out the first action and define the function,

$$\tilde{V}_\pi(s_0, a_0) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t \sim \pi(s_t)\right]$$

- This we will refer to as the Q-function,

$$Q_\pi(s_0, a_0) = \tilde{V}_\pi(s_0, a_0)$$

Q-Learning

- How should we pick our policy?

Q-Learning

- How should we pick our policy?
- The greedy policy is optimal (Bellman Optimality Equation)

$$\pi_*(s) = \text{argmax}_{a \in A}\, Q_*(s, a)$$

Q-Learning

- How should we pick our policy?
- The greedy policy is optimal (Bellman Optimality Equation)

$$\pi_*(s) = \text{argmax}_{a \in A}\, Q_*(s, a)$$

- The optimal Q function,

$$Q_*(s, a) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t \sim \pi_*(s_t)\right]$$

Lets re-arrange things

$$Q_*(s, a) = \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t \sim \pi_*(s_t)\right]$$

Lets re-arrange things

$$
\begin{aligned}
Q_*(s, a) &= \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t \sim \pi_*(s_t)\right] \\
&= \mathbb{E}[r_0 \mid a_0] + \gamma \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid a_{t+1} \sim \pi_*(s_{t+1})\right]
\end{aligned}
$$

Lets re-arrange things

$$
\begin{aligned}
Q_*(s,a) &= \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t \sim \pi_*(s_t)\right] \\
&= \mathbb{E}[r_0 \mid a_0] + \gamma \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid a_{t+1} \sim \pi_*(s_{t+1})\right] \\
&= \mathbb{E}[r_0 \mid a_0] + \gamma \cdot V_*(s_1)
\end{aligned}
$$

Lets re-arrange things

$$
\begin{aligned}
Q_*(s,a) &= \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t \sim \pi_*(s_t)\right] \\
&= \mathbb{E}[r_0 \mid a_0] + \gamma \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid a_{t+1} \sim \pi_*(s_{t+1})\right] \\
&= \mathbb{E}[r_0 \mid a_0] + \gamma \cdot V_*(s_1) \\
&= \mathbb{E}[r_0 \mid a_0] + \gamma \cdot Q_*(s_1, \pi_*(s_1))
\end{aligned}
$$

Lets re-arrange things

$$
\begin{aligned}
Q_*(s,a) &= \mathbb{E}[r_0 \mid a_0] + \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid a_t \sim \pi_*(s_t)\right] \\
&= \mathbb{E}[r_0 \mid a_0] + \gamma \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid a_{t+1} \sim \pi_*(s_{t+1})\right] \\
&= \mathbb{E}[r_0 \mid a_0] + \gamma \cdot V_*(s_1) \\
&= \mathbb{E}[r_0 \mid a_0] + \gamma \cdot Q_*(s_1, \pi_*(s_1)) \\
&= \mathbb{E}[r_0 \mid a_0] + \gamma \operatorname{argmax}_{a \in A} Q(s_1, a)
\end{aligned}
$$

Learning the Q

$$\theta_* = \text{argmin}_\theta \left(\mathrm{Q}_\theta(\mathrm{s},\mathrm{a})\text{-}\left(\mathbb{E}[\mathrm{r}_0 \mid \mathrm{a}_0] + \gamma \,\text{argmax}_{\mathrm{a}_1 \in \mathrm{A}}\, \mathrm{Q}_\theta(\mathrm{s}_1,\mathrm{a})\right)\right)^2$$

$$= \text{argmin}_\theta \left(\underbrace{\left(Q_\theta(s,a)\text{-}\gamma \,\text{argmax}_{a_1 \in A}\, Q_\theta(s_1,a)\right)}_{\Delta \mathrm{Q}} \text{-}\mathbb{E}[\mathrm{r}_0 \mid \mathrm{a}_0] \right)^2$$

- Can we learn a θ such that we can approximate the optimal Q?
- For any $\{s_0, s_1, a_0, r\}$ we will receive an error

Data Acquisition

- Same strategy as with BO but much harder to specify your utility

Data Acquisition

- Same strategy as with BO but much harder to specify your utility
- Acquisition functions to balance Exploration and Exploitation

Data Acquisition

- Same strategy as with BO but much harder to specify your utility
- Acquisition functions to balance Exploration and Exploitation
- Model based RL

Data Acquisition

- Same strategy as with BO but much harder to specify your utility
- Acquisition functions to balance Exploration and Exploitation
- Model based RL
  - we have parametrised our ignorance, how does this relate to the problem?

Data Acquisition

- Same strategy as with BO but much harder to specify your utility
- Acquisition functions to balance Exploration and Exploitation
- Model based RL
  - we have parametrised our ignorance, how does this relate to the problem?
- Model free RL

Data Acquisition

- Same strategy as with BO but much harder to specify your utility
- Acquisition functions to balance Exploration and Exploitation
- Model based RL
  - we have parametrised our ignorance, how does this relate to the problem?
- Model free RL
  - much harder

Data Acquisition

- Same strategy as with BO but much harder to specify your utility
- Acquisition functions to balance Exploration and Exploitation
- Model based RL
  - we have parametrised our ignorance, how does this relate to the problem?
- Model free RL
  - much harder
  - ϵ-greedy

Data Acquisition

- Same strategy as with BO but much harder to specify your utility
- Acquisition functions to balance Exploration and Exploitation
- Model based RL
  - we have parametrised our ignorance, how does this relate to the problem?
- Model free RL
  - much harder
  - ϵ-greedy
    - with probablity $1 - \epsilon$ take the action that maximises the Q

Data Acquisition

- Same strategy as with BO but much harder to specify your utility
- Acquisition functions to balance Exploration and Exploitation
- Model based RL
  - we have parametrised our ignorance, how does this relate to the problem?
- Model free RL
  - much harder
  - ϵ-greedy
    - with probablity $1 - \epsilon$ take the action that maximises the Q
    - with probability ϵ take a random action

Challenge of RL

- Model based RL
  - allows us to input our knowledge into the system
  - allows for (more) principled acquisition strategies
  - very challenging inference task

Challenge of RL

- Model based RL
  - allows us to input our knowledge into the system
  - allows for (more) principled acquisition strategies
  - very challenging inference task
- Model free RL
  - simple to implement and relatively cheap
  - hard to device principle acquisition strategies
  - often very data inefficient

Summary

Summary

- Reinforcement learning is often explained as a thing in itself

Summary

- Reinforcement learning is often explained as a thing in itself
- It is not, its just a very specific, recursive probabilistic model

Summary

- Reinforcement learning is often explained as a thing in itself
- It is not, its just a very specific, recursive probabilistic model
- We apply RL to problems where it is infeasible to acquire all data at the same time

Summary

- Reinforcement learning is often explained as a thing in itself
- It is not, its just a very specific, recursive probabilistic model
- We apply RL to problems where it is infeasible to acquire all data at the same time
- Just as BO the data-acquisition therefore has to be part of the inference mechanism

Summary

- Reinforcement learning is often explained as a thing in itself
- It is not, its just a very specific, recursive probabilistic model
- We apply RL to problems where it is infeasible to acquire all data at the same time
- Just as BO the data-acquisition therefore has to be part of the inference mechanism
- There are many things where RL is a suitable approach

Summary

- RL raises interesting general

Summary

- RL raises interesting general
- Finite capacity of models

Summary

- RL raises interesting general
- Finite capacity of models
  - How do we forget information no longer relevant?

Summary

- RL raises interesting general
- Finite capacity of models
  - How do we forget information no longer relevant?
  - Requires an understanding of my own knowledge

Summary

- RL raises interesting general
- Finite capacity of models
  - How do we forget information no longer relevant?
  - Requires an understanding of my own knowledge
- Proxy objectives

Summary

- RL raises interesting general
- Finite capacity of models
  - How do we forget information no longer relevant?
  - Requires an understanding of my own knowledge
- Proxy objectives
  - just as BO, learning the dynamic and the policy are just proxies to maximise the reward

Summary

- RL raises interesting general
- Finite capacity of models
  - How do we forget information no longer relevant?
  - Requires an understanding of my own knowledge
- Proxy objectives
  - just as BO, learning the dynamic and the policy are just proxies to maximise the reward
- Hierarchical decomposition of uncertainty

Summary

- RL raises interesting general
- Finite capacity of models
  - How do we forget information no longer relevant?
  - Requires an understanding of my own knowledge
- Proxy objectives
  - just as BO, learning the dynamic and the policy are just proxies to maximise the reward
- Hierarchical decomposition of uncertainty
  - am I uncertain of the reward because I don't know what I should do (policy) but if I act I know exactly what happens (dynamic)?

Summary

- RL raises interesting general
- Finite capacity of models
  - How do we forget information no longer relevant?
  - Requires an understanding of my own knowledge
- Proxy objectives
  - just as BO, learning the dynamic and the policy are just proxies to maximise the reward
- Hierarchical decomposition of uncertainty
  - am I uncertain of the reward because I don't know what I should do (policy) but if I act I know exactly what happens (dynamic)?
  - or am I uncertain of the reward because I know what I should do (policy) but I don't know what will happen if I do (dynamic)?

Summary

- Everyone loves RL till they start working on it

Summary

- Everyone loves RL till they start working on it
- RL has very many moving parts making it hard to understand the effects of different components

Summary

- Everyone loves RL till they start working on it
- RL has very many moving parts making it hard to understand the effects of different components
- The literature is good (Sutton et al., 1998) and bad `arXiv`

Summary

- Everyone loves RL till they start working on it
- RL has very many moving parts making it hard to understand the effects of different components
- The literature is good (Sutton et al., 1998) and bad `arXiv`
- RL is very hard meaning its as much an art as a science to get it to work

Summary

- Everyone loves RL till they start working on it
- RL has very many moving parts making it hard to understand the effects of different components
- The literature is good (Sutton et al., 1998) and bad `arXiv`
- RL is very hard meaning its as much an art as a science to get it to work
- Don't do RL for your project!

Summary

- Everyone loves RL till they start working on it
- RL has very many moving parts making it hard to understand the effects of different components
- The literature is good (Sutton et al., 1998) and bad `arXiv`
- RL is very hard meaning its as much an art as a science to get it to work
- Don't do RL for your project!
- Please don't

eof

References

Sutton, Richard S. and Andrew G. Barto (1998). ***Introduction to Reinforcement Learning***. 1st. Cambridge, MA, USA: MIT Press.