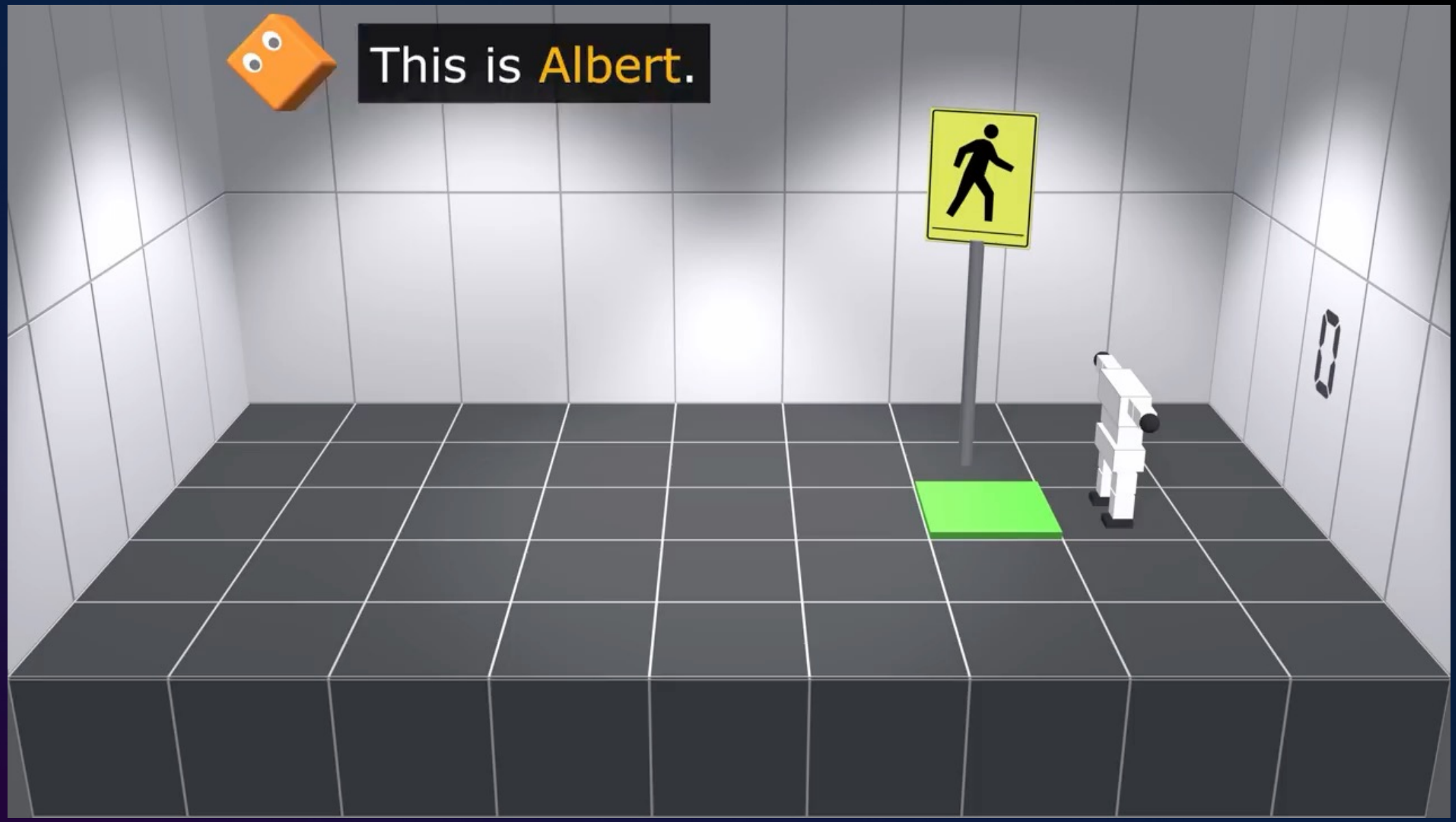
RECAP

Introduction to Re-inforcement Learning

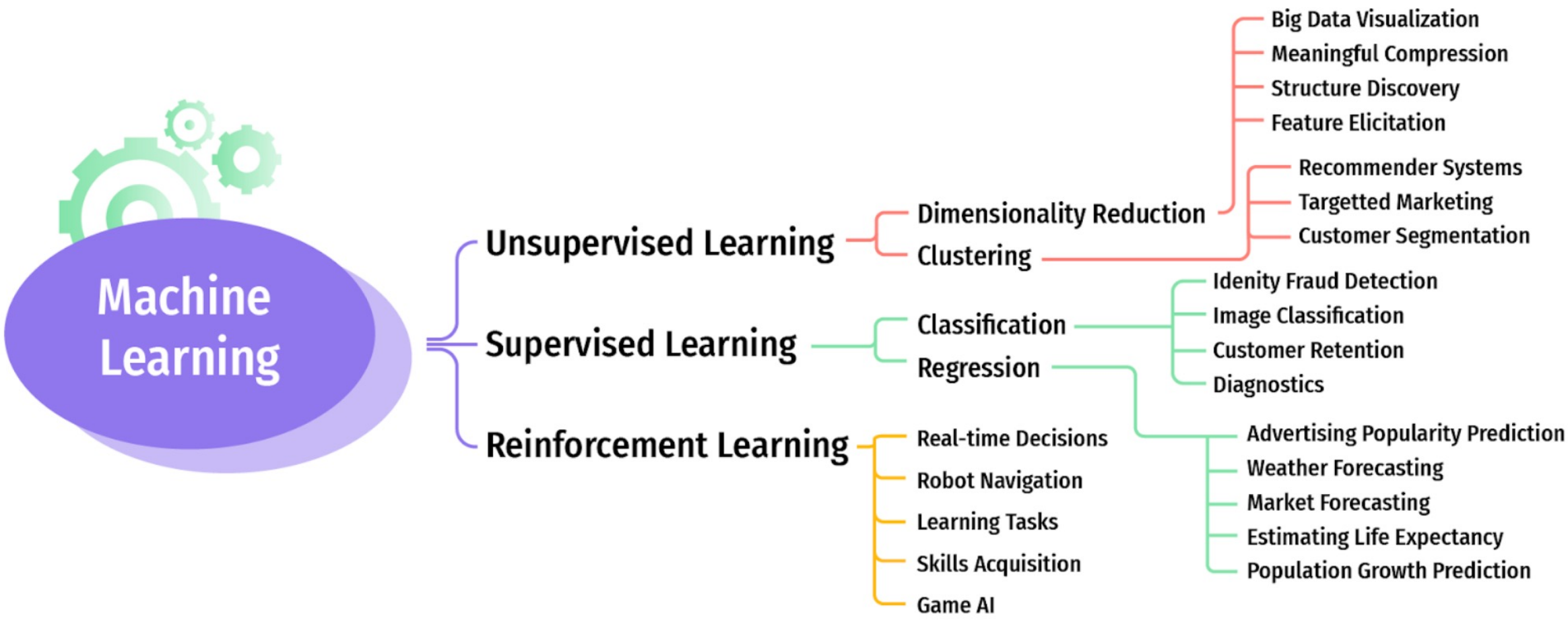
Irshad Chohan

Principal Solutions Architect
AWS India

Let’s watch a video



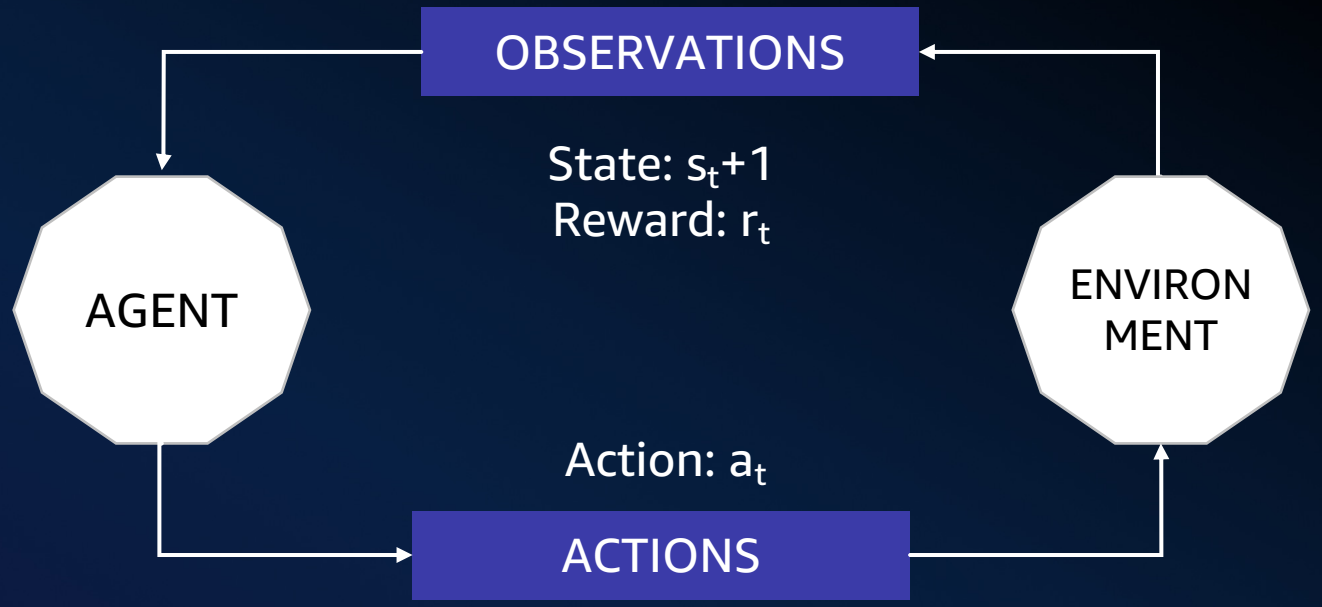
3 fields



What is Reinforcement Learning

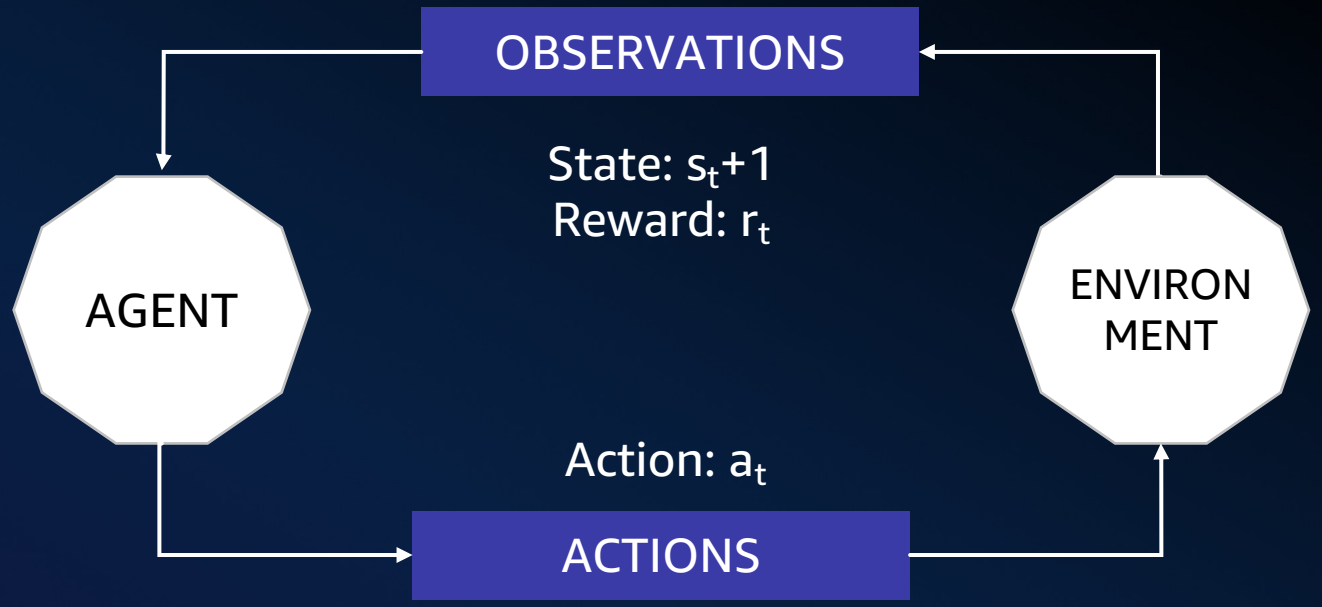
What is Reinforcement Learning

AGENT: Take actions
ENVIRONMENT: the world
ACTION: a move by agent
OBSERVATION: of the env. After taking actions.
STATE: a situation of the env.
Reward: award/punishment for taking actions.



Markov Decision Process

AGENT: Take actions
ENVIRONMENT: the world
ACTION: a move by agent
OBSERVATION: of the env. After taking actions.
STATE: a situation of the env.
Reward: award/punishment for taking actions.



RL algorithms

Value based learning : Find optimal Q(s,a) / Q(s)

Policy based learning: Find optimal π(s)

Bellman equation:

$$V^{\pi}(s) = \sum_{a} \pi(s,a) \sum_{s'} P^{a}_{ss'} \left[r^{a}_{ss'} + \gamma V^{\pi}(s') \right] = \sum_{a} \pi(s,a) Q^{\pi}(s,a)$$

Value functions

state value function: $V^{\pi}(s)$

expected return when starting in *s* and following π

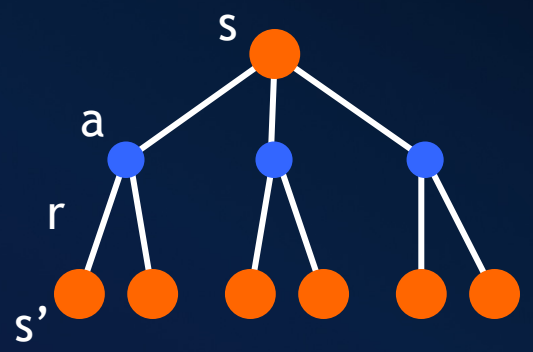
state-action value function: $Q^{\pi}(s,a)$

expected return when starting in *s*, performing *a,* and following π

useful for finding the optimal policy

can estimate from experience

pick the best action using $Q^{\pi}(s,a)$



Bellman equation

$$V^{\pi}(s) = \sum_{a} \pi(s,a) \sum_{s'} P^{a}_{ss'} \left[r^{a}_{ss'} + \gamma V^{\pi}(s') \right] = \sum_{a} \pi(s,a) Q^{\pi}(s,a)$$

Now, agenda

- Defining an RL problem
  - Markov Decision Processes
- Solving an RL problem
  - Dynamic Programming
  - Monte Carlo methods
  - Temporal-Difference learning

Now, agenda

Defining an RL problem

- ~~Markov Decision Processes~~

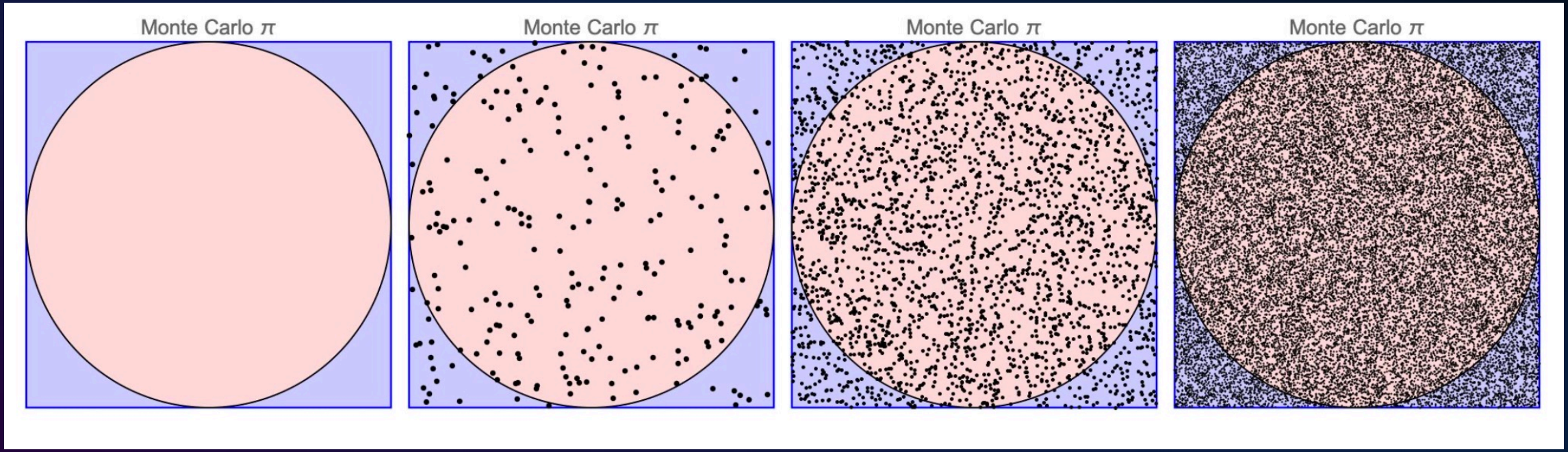
Solving an RL problem

- ~~Dynamic Programming~~
- Monte Carlo methods
- Temporal-Difference learning

Let’s start

What is Monte Carlo method?

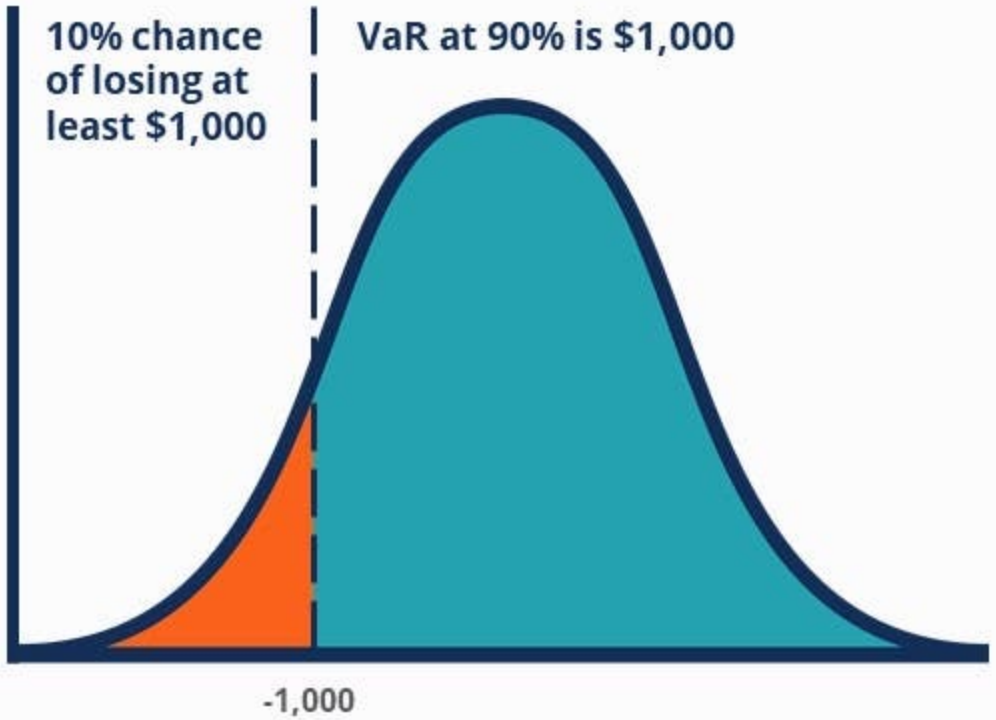
Monte Carlo methods are ways of solving the reinforcement learning problem based on averaging sample returns.



Value At Risk

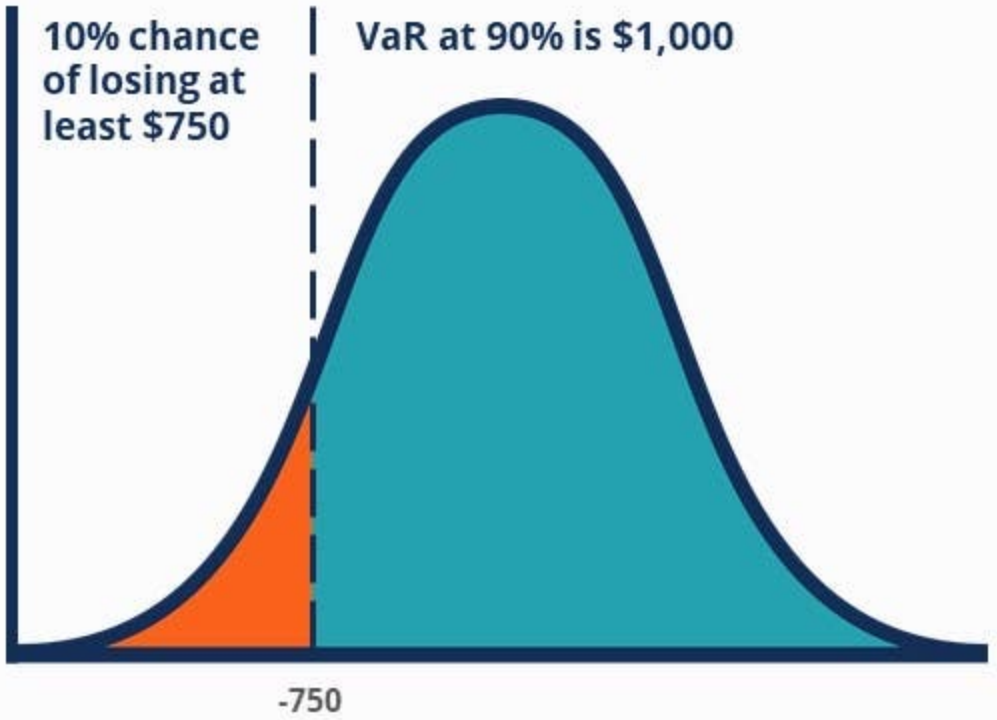
Project A

Expected Profit: $10,000



Project B

Expected Profit: $10,000



Workshop: Finding value of Pi

https://github.com/irshadc/reinforcement-learning-notebooks

Model free vs Model-based methods

- **Model-based**: methods using a model to plan action before they are taken.
- **Model-free**: methods without a model. The algorithm tries to learn to associate actions and respective returns.

Monte Carlo (MC) methods neither estimate nor use $p(s', r \mid s, a)$, so they are model-free. Instead, they learn from **experience** — sequences of states, actions and rewards. By using given agent trajectories, they average approximate expected values and use GPI to obtain optimal policies.

Monte Carlo methods

Don’t need full knowledge of environment

- just experience, or
- simulated experience

but similar to Dynamic Programing

- policy evaluation, policy improvement

averaging sample returns

- defined only for episodic tasks

Monte Carlo in RL

It is a method for estimating Value-action(Value|State, Action) or Value function(Value|State) using some sample runs from the environment for which we are estimating the Value function.

V-function estimation

- **The first-visit MC method**: only returns of the first visits to a state are considered

- **The every-visit MC method**: returns during all visits to a state are considered.

First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:

Append G to $Returns(S_t)$

$V(S_t) \leftarrow \text{average}(Returns(S_t))$

Monte Carlo policy evaluation

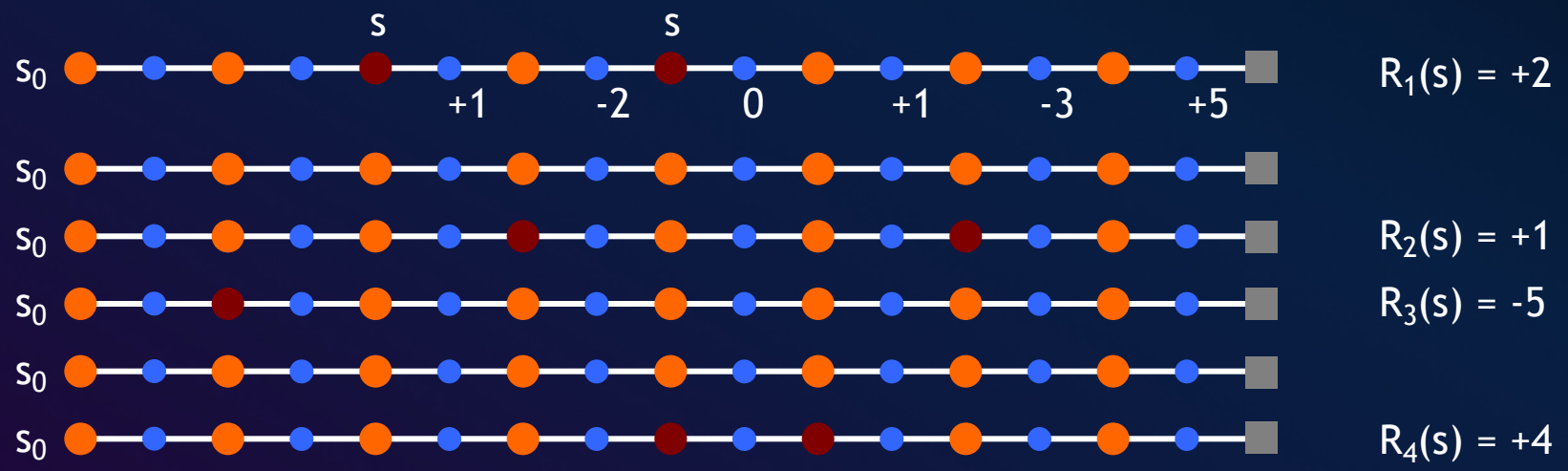
want to estimate $V^{\pi}(s)$

= expected return starting from s and following π

estimate as average of observed returns in state s

first-visit MC

average returns following the first visit to state s

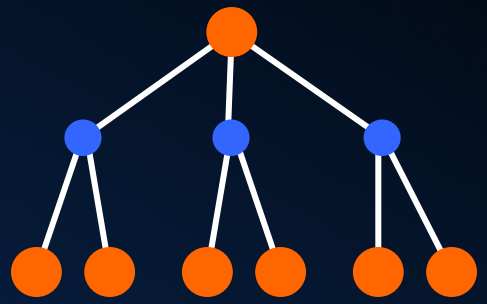


$V^{\pi}(s) \approx (2 + 1 - 5 + 4)/4 = 0.5$

Monte Carlo control

V^π not enough for policy improvement

need exact model of environment



estimate $Q^\pi(s,a)$ $\pi'(s) = \arg\max_a Q^\pi(s,a)$

MC control $\pi_0 \rightarrow^E Q^{\pi_0} \rightarrow^I \pi_1 \rightarrow^E Q^{\pi_1} \rightarrow^I \ldots \rightarrow^I \pi^* \rightarrow^E Q^*$

update after each episode

non-stationary environment

$$V(s) \leftarrow V(s) + \alpha\,[R - V(s)]$$

a problem

greedy policy won't explore all actions

Maintaining exploration

deterministic/greedy policy won't explore all actions

- don't know anything about the environment at the beginning
- need to try all actions to find the optimal one

maintain exploration

- use *soft* policies instead: $\pi(s,a)>0$ (for all s,a)

ε-greedy policy

- with probability 1-ε perform the optimal/greedy action
- with probability ε perform a random action

- will keep exploring the environment
- slowly move it towards greedy policy: ε -> 0

Simulated experience

5-card draw poker

s_0: A♣, A♦, 6♠, A♥, 2♠

a_0: discard 6♠, 2♠

s_1: A♣, A♦, A♥, A♠, 9♠ + dealer takes 4 cards

return: +1 (probably)

DP

list all states, actions, compute P(s,a,s')

P([A♣,A♦,6♠,A♥,2♠], [6♠,2♠], [A♠,9♠,4]) = 0.00192

MC

all you need are sample episodes

let MC play against a random policy, or itself, or another algorithm

Summary of Monte Carlo

- don’t need model of environment
  - averaging of sample returns
  - only for episodic tasks
- learn from sample episodes or simulated experience
- can concentrate on “important” states
  - don’t need a full sweep
- need to maintain exploration
  - use soft policies

Workshop time!

Quiz time::

In Monte Carlo methods, what is the primary requirement for updating value estimates?

a) Full knowledge of the environment's model

b) Access to the environment's reward function

c) Simulated experiences of the full environment

d) Completing an entire episode before updating

What is the main difference between Q-learning and Monte Carlo methods?

a) Q-learning requires episodes to be completed, but Monte Carlo does not

b) Q-learning is model-based, while Monte Carlo is model-free

c) Q-learning updates after every step, whereas Monte Carlo updates after an episode

d) Monte Carlo learns optimal policy faster than Q-learning

Which of the following is TRUE about the exploration-exploitation trade-off in Q-learning?

a) Exploration always leads to better performance

b) Exploitation means trying new actions for better rewards

c) Exploration helps the agent discover better strategies, while exploitation leverages known strategies

d) Q-learning does not face the exploration-exploitation trade-off

How do Monte Carlo methods estimate the value of a state?

a) By averaging the rewards obtained from all visits to that state

b) By calculating the immediate reward after each step

c) By using the maximum reward observed from the state

d) By performing a Bellman backup

Which of the following is a key limitation of Monte Carlo methods?

a) They require a complete model of the environment

b) They can only be used for deterministic environments

c) They rely on completing full episodes, which can be impractical in some tasks

d) They cannot handle stochastic rewards

Importance Sampling

Importance Sampling

Importance sampling is a technique of estimating the expected value of *f(x)* where *x* has a data distribution ***p***. However, Instead of sampling from ***p***, we calculate the result from sampling ***q***.

$$\mathbb{E}_p[f(x)] = \mathbb{E}_q\left(\frac{f(\boldsymbol{X})p(\boldsymbol{X})}{q(\boldsymbol{X})}\right)$$

Difference between these two function?

$$\int p(\mathbf{x}) f(\mathbf{x}) d\mathbf{x}$$

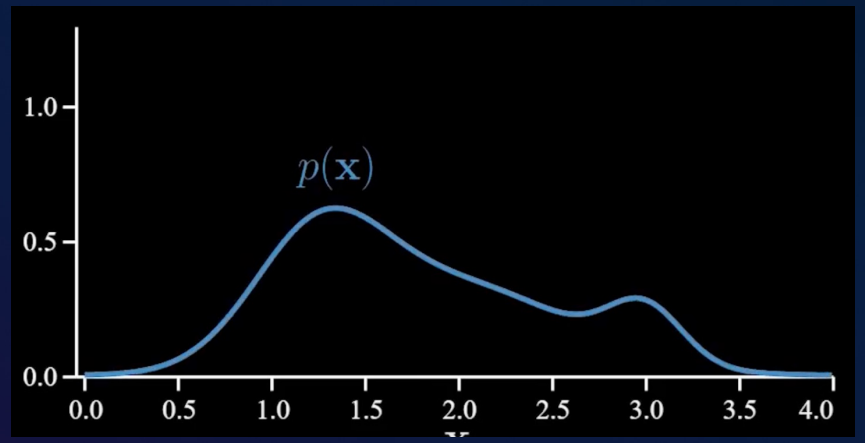
Continuous space $= \mathbb{E}_p[f(\mathbf{x})]$

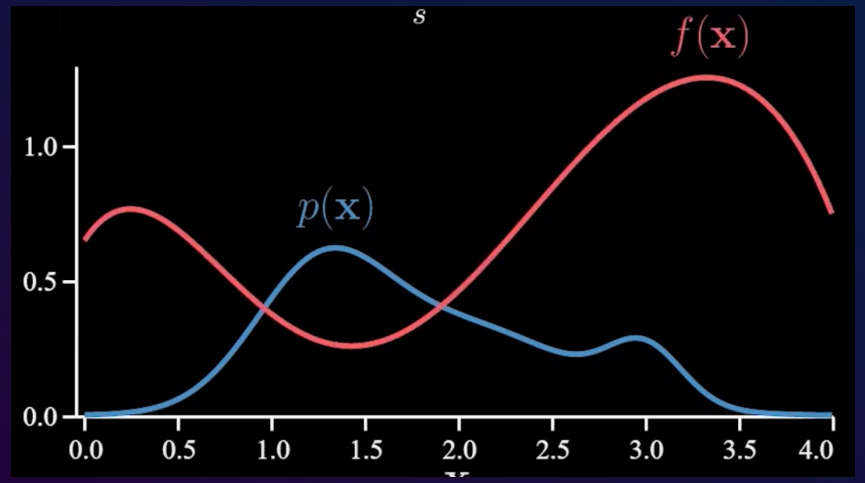
If $\mathbf{x}$ is discrete:

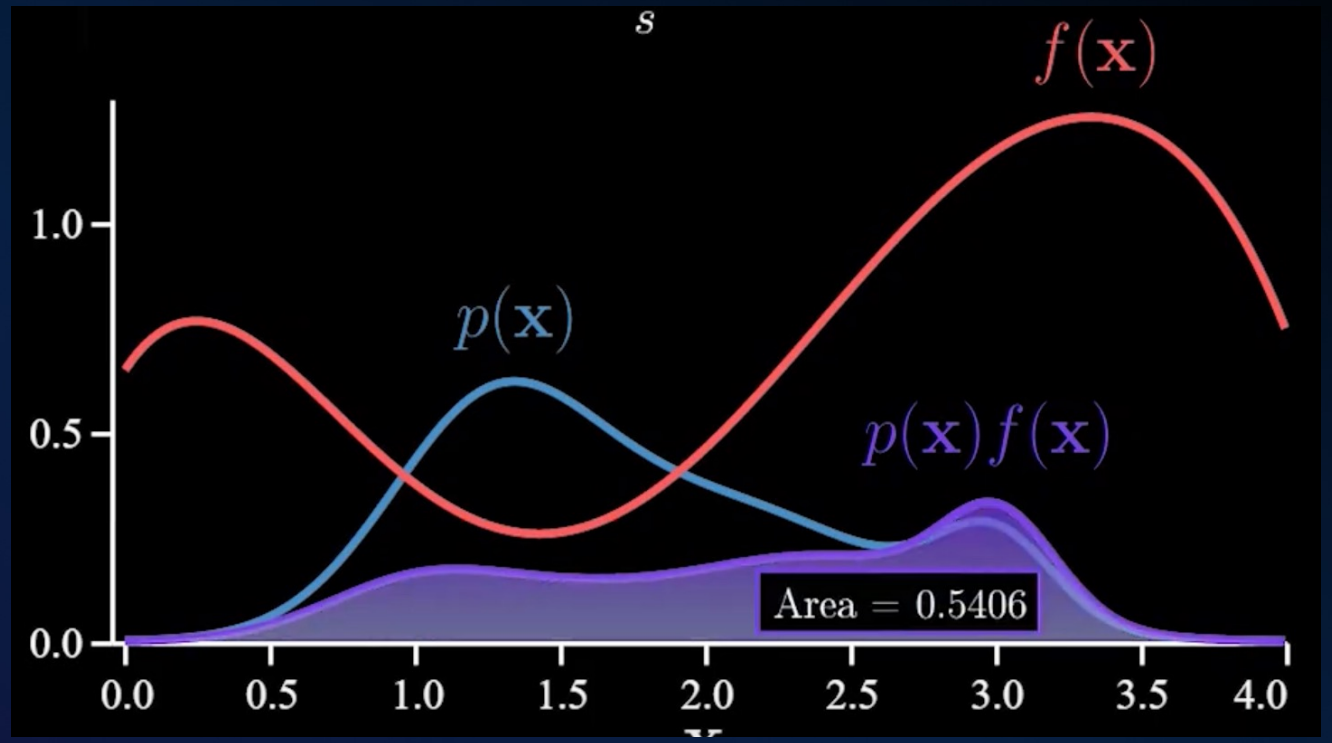
$$\sum_{\mathbf{x}} p(\mathbf{x}) f(\mathbf{x})$$

Discreet space

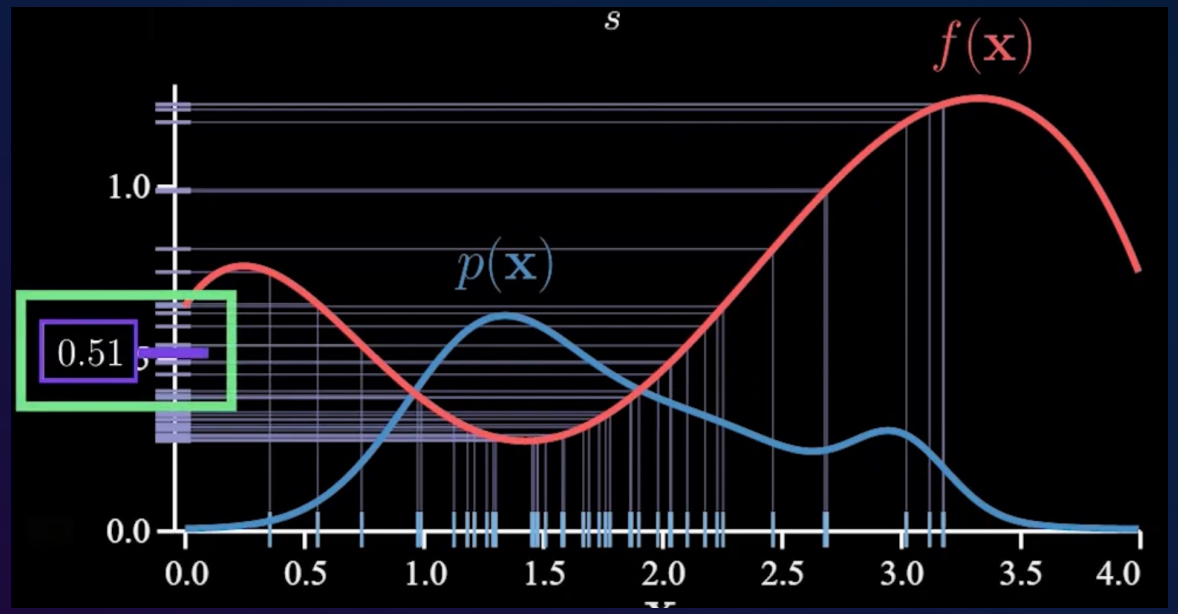
Let’s understand…



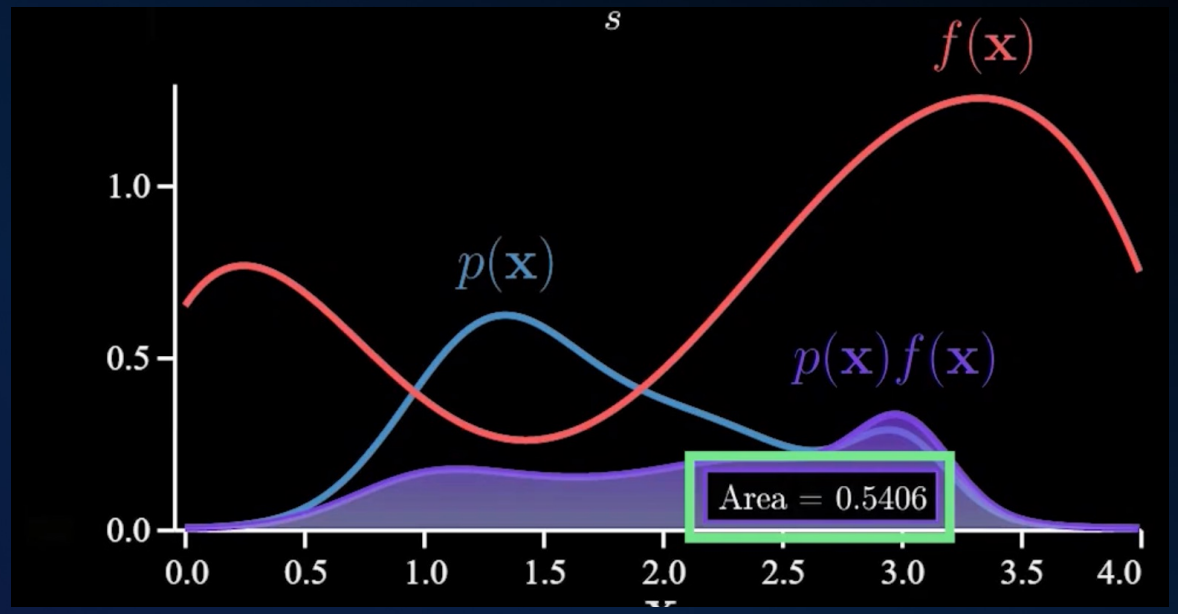




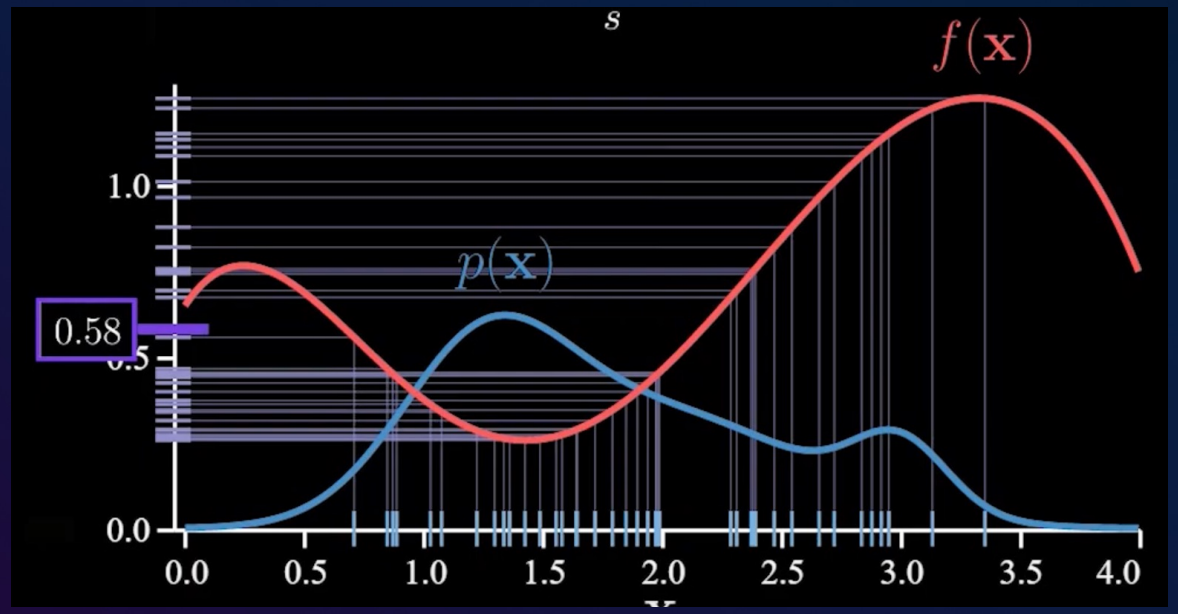
Monte Carlo vs Actual integration product area



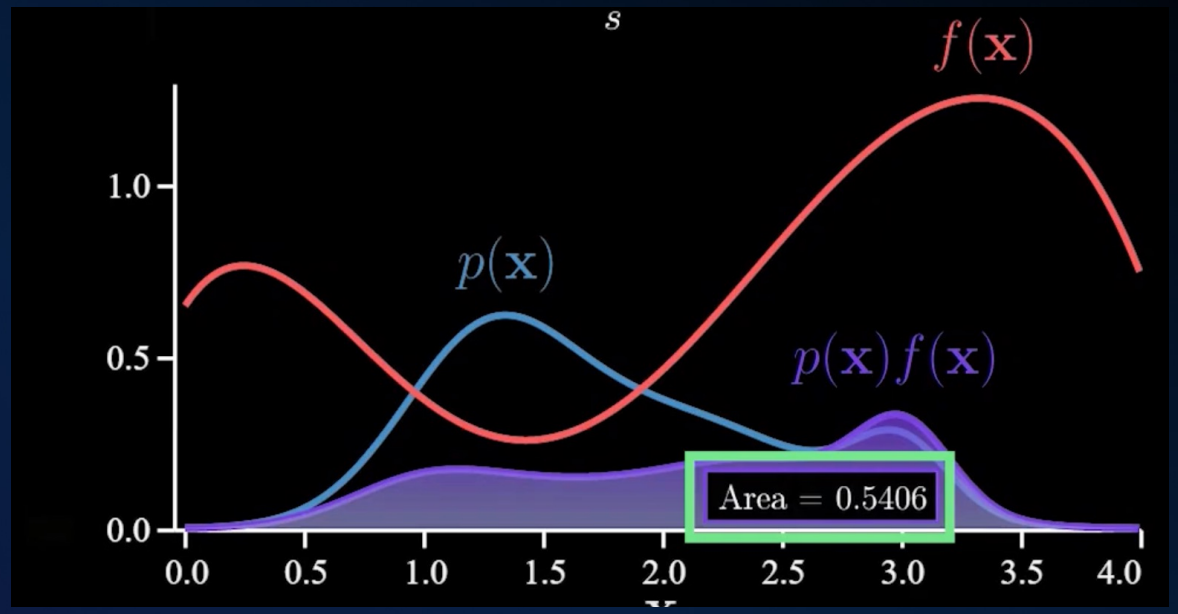
≈



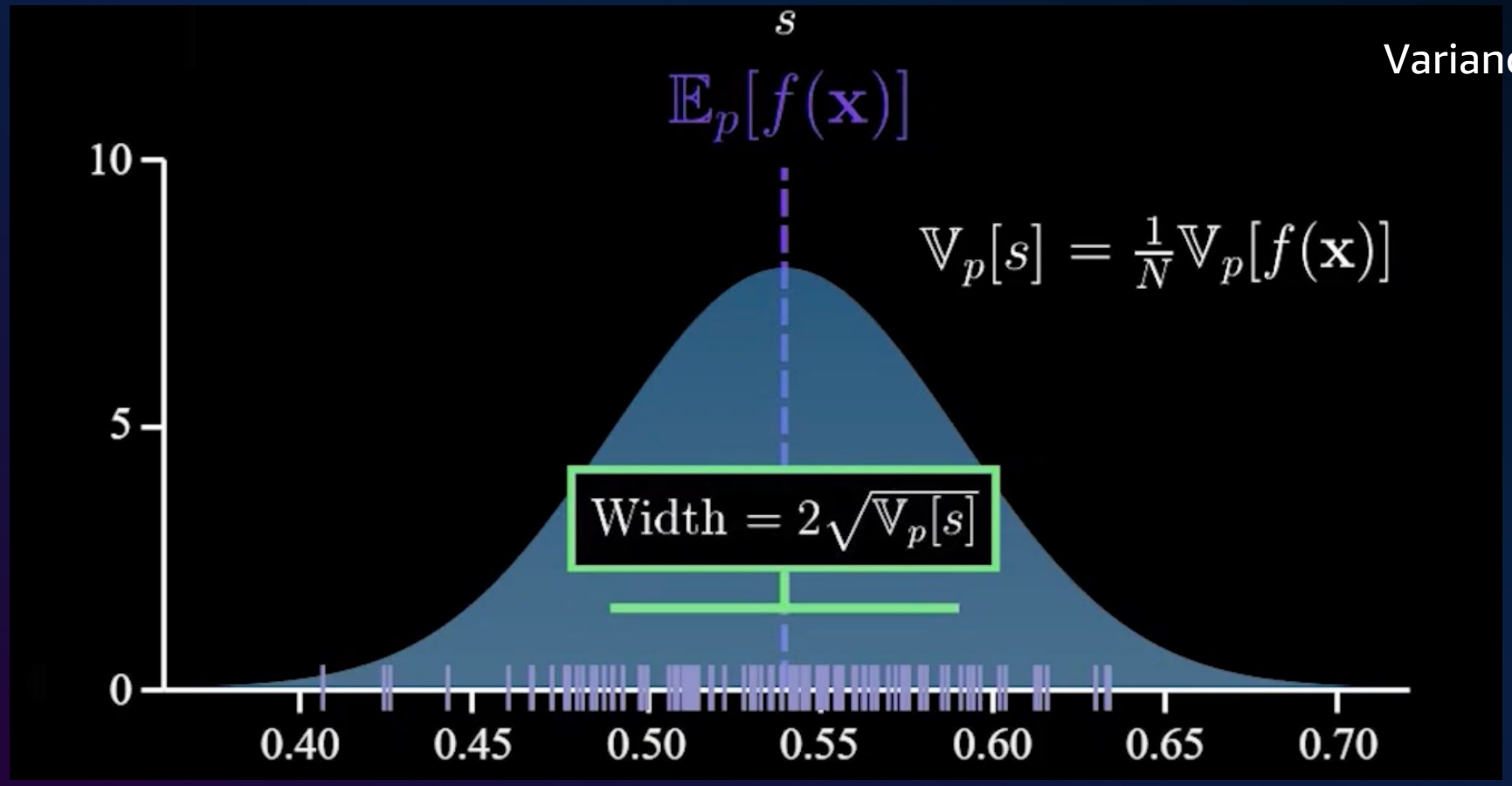
Monte Carlo vs Actual integration product area



≈



Plotting random distribution MC value multiple times

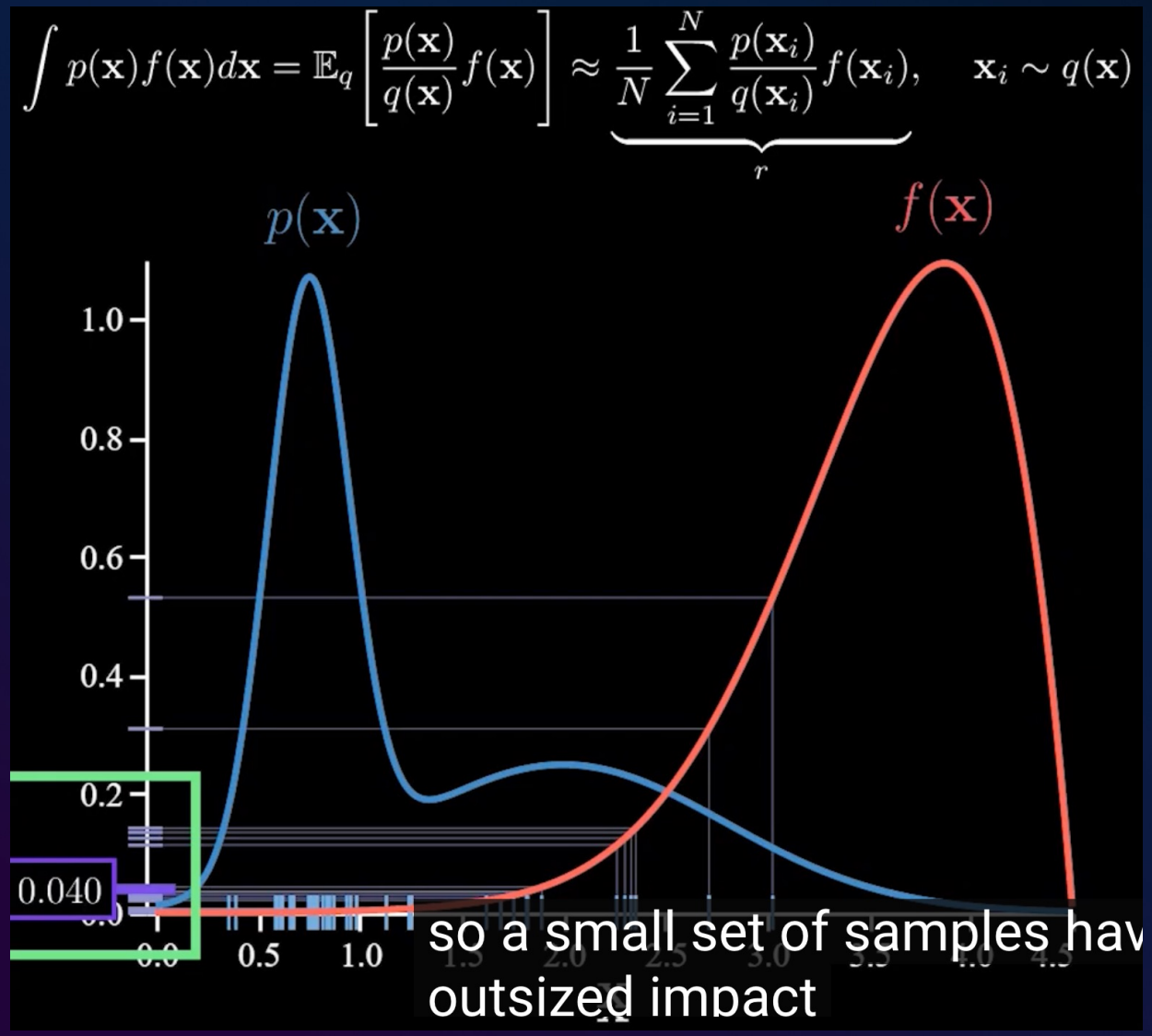


Variance of s (width of distribution) matters.

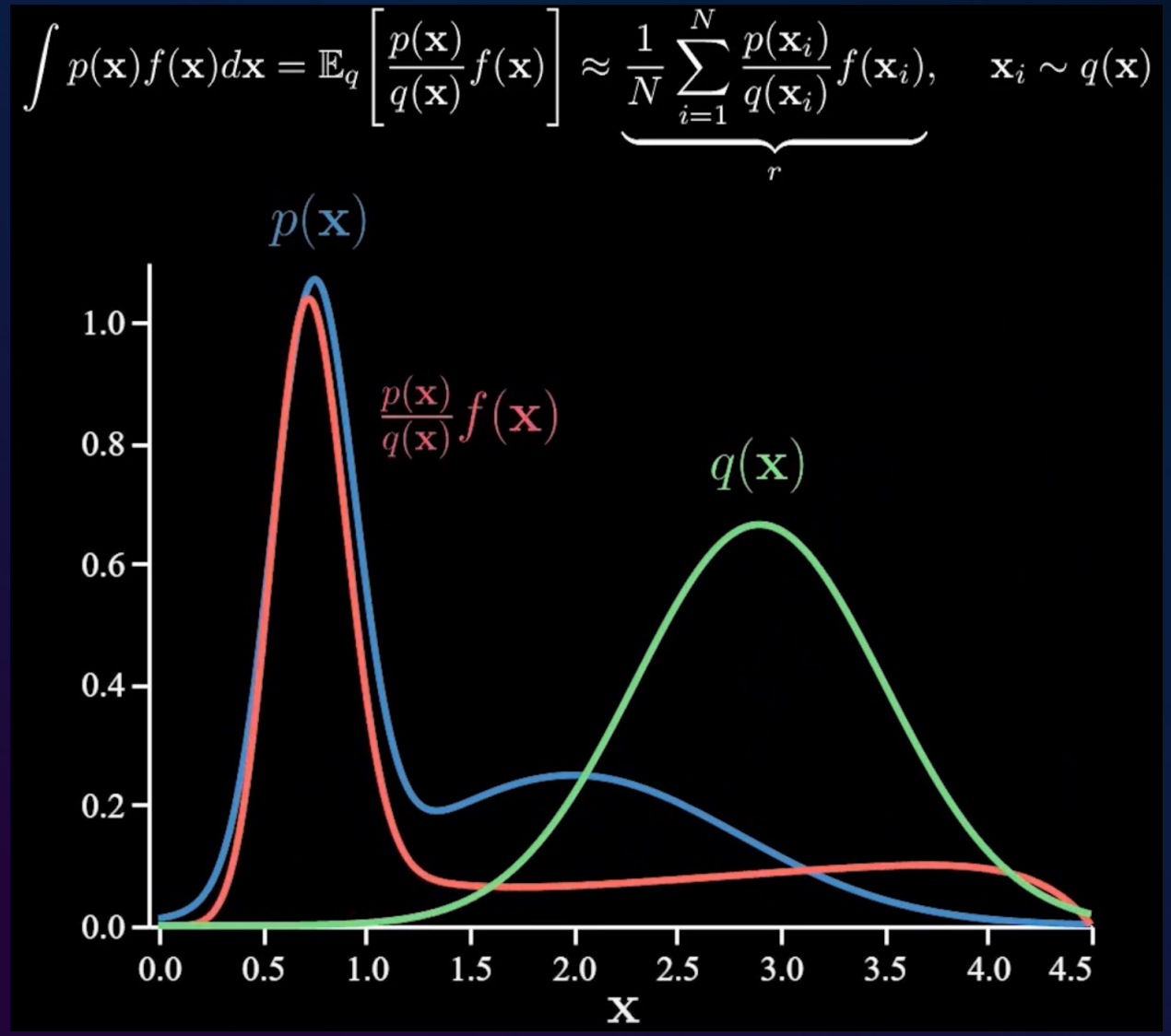
Optimal Importance Sampling

what will be the best sampling distribution in minimizing the variance of our estimation?

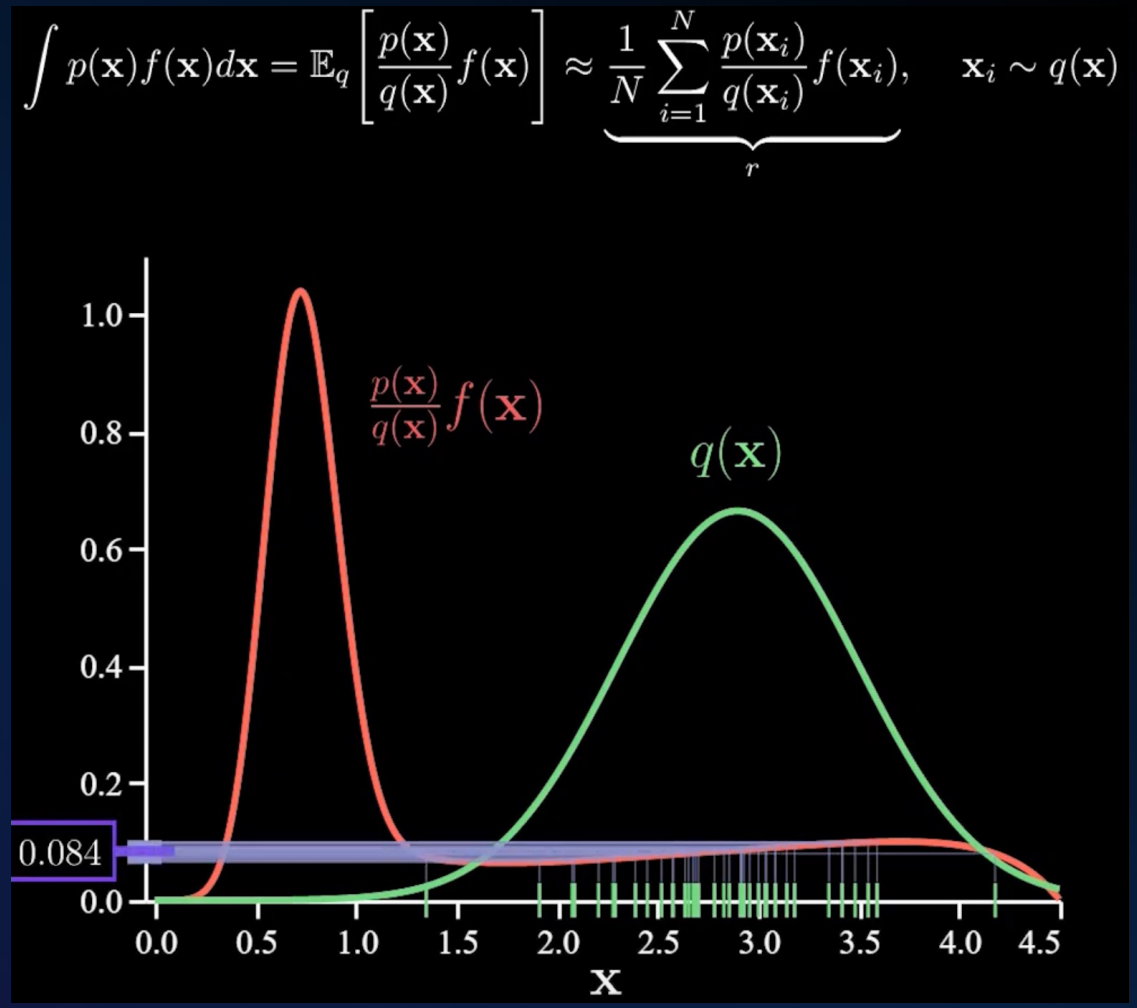
$$\mathbb{E}_p[f(x)] = \mathbb{E}_q\left(\frac{f(\boldsymbol{X})p(\boldsymbol{X})}{q(\boldsymbol{X})}\right)$$



$$\int p(\mathbf{x}) f(\mathbf{x}) d\mathbf{x} = \mathbb{E}_q \left[\frac{p(\mathbf{x})}{q(\mathbf{x})} f(\mathbf{x}) \right] \approx \underbrace{\frac{1}{N} \sum_{i=1}^{N} \frac{p(\mathbf{x}_i)}{q(\mathbf{x}_i)} f(\mathbf{x}_i)}_{r}, \quad \mathbf{x}_i \sim q(\mathbf{x})$$



$$\int p(\mathbf{x}) f(\mathbf{x}) d\mathbf{x} = \mathbb{E}_q \left[\frac{p(\mathbf{x})}{q(\mathbf{x})} f(\mathbf{x}) \right] \approx \underbrace{\frac{1}{N} \sum_{i=1}^{N} \frac{p(\mathbf{x}_i)}{q(\mathbf{x}_i)} f(\mathbf{x}_i)}_{r}, \quad \mathbf{x}_i \sim q(\mathbf{x})$$



When is Importance Sampling used?

- $p(\mathbf{x})$ is difficult to sample from.
- We can evaluate $p(\mathbf{x})$.
- $q(\mathbf{x})$ is easy to evaluate and sample from.
- We can choose $q(\mathbf{x})$ to be high where $|p(\mathbf{x})f(\mathbf{x})|$ is high.

Use-cases

Forrest tree counting

Game AI

Fraud detection

Resources

Deep learning meta-book: d2l.ai

Reinforcement Learning book: http://incompleteideas.net/book/RLbook2020.pdf

Course from MIT: http://introtodeeplearning.com/

AWS MLU Youtube Channel

MLU-Explain

HTML Demo for GridWorld [1], [2]

Thank you!