Reinforcement Learning

João Pedro Fontoura da Silva
Vítor Bandeira Borges

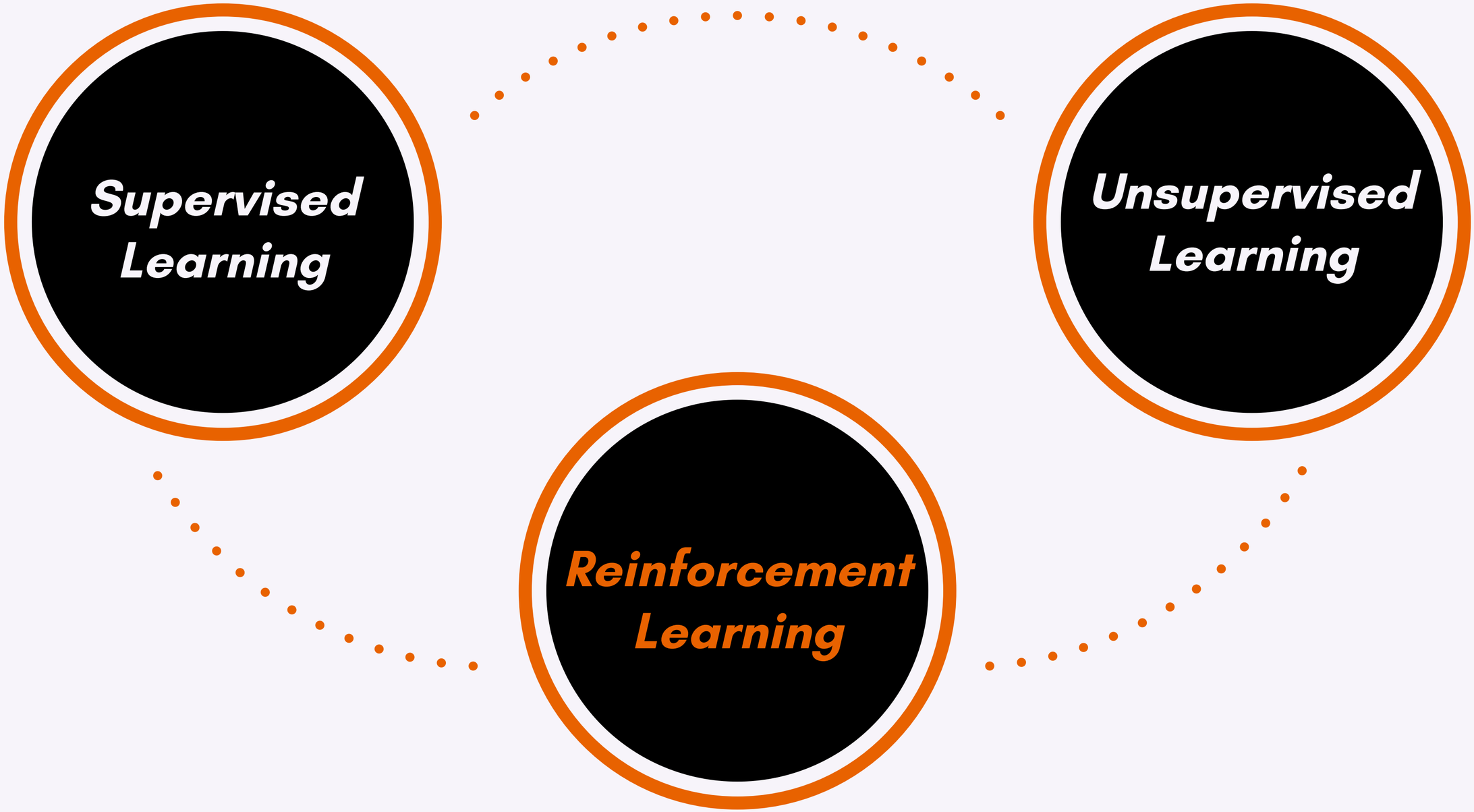
LAMFO/UnB

REINFORCEMENT LEARNING

I.

1. Introdução

- Aprendizado a partir da interação com o ambiente
- Ação → Informação → Reação
- É um método computacional de aprendizado por interação
- Estudamos um agente que toma decisões que afetam o ambiente e que busca maximizar retornos, mas sempre primando por atingir um objetivo explícito


Supervised Learning
Unsupervised Learning
Reinforcement Learning

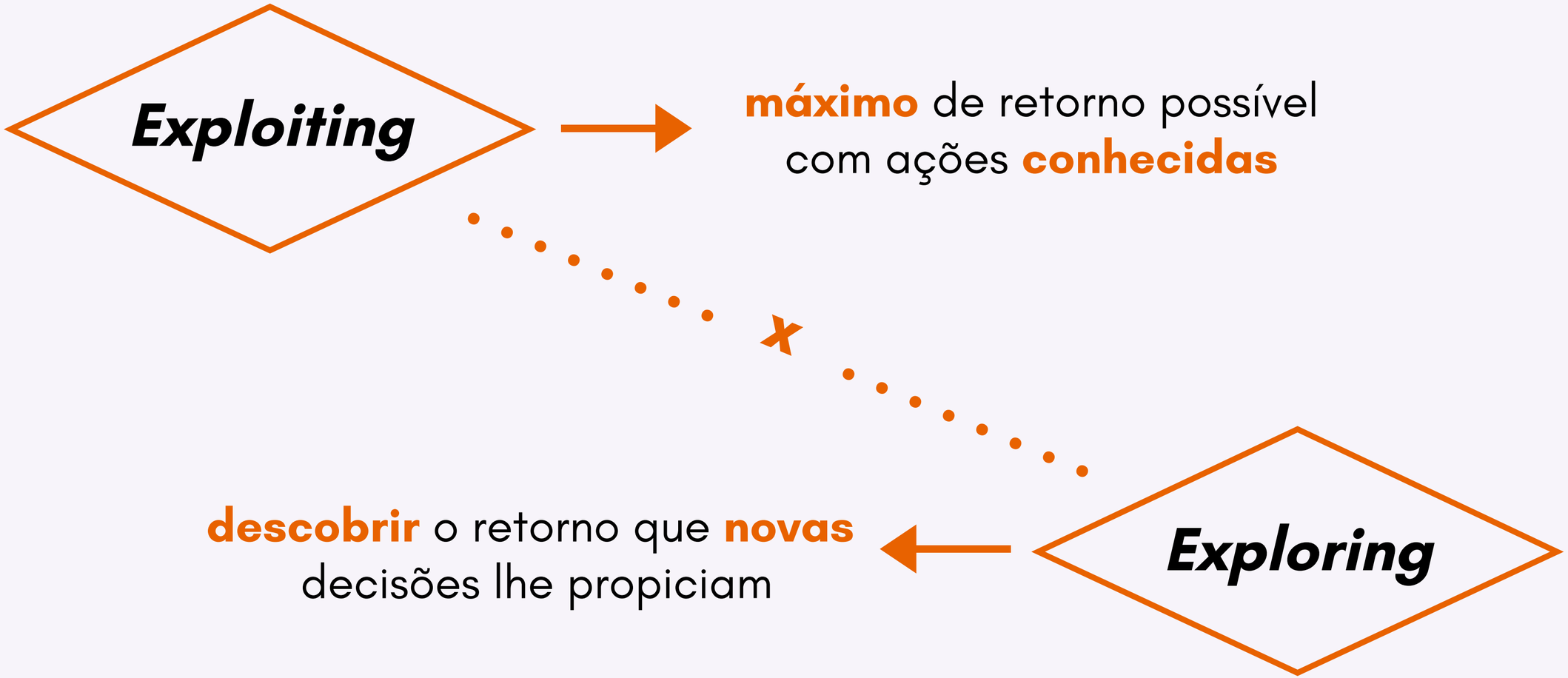
- Supõe a existência de decisões "certas" e "erradas" previamente definidas por um supervisor externo
- O agente deve encontrar uma forma de **generalizar** os inputs a partir dos exemplos de treinamento

- Procura encontrar estruturas ocultas em dados sem classificação prévia

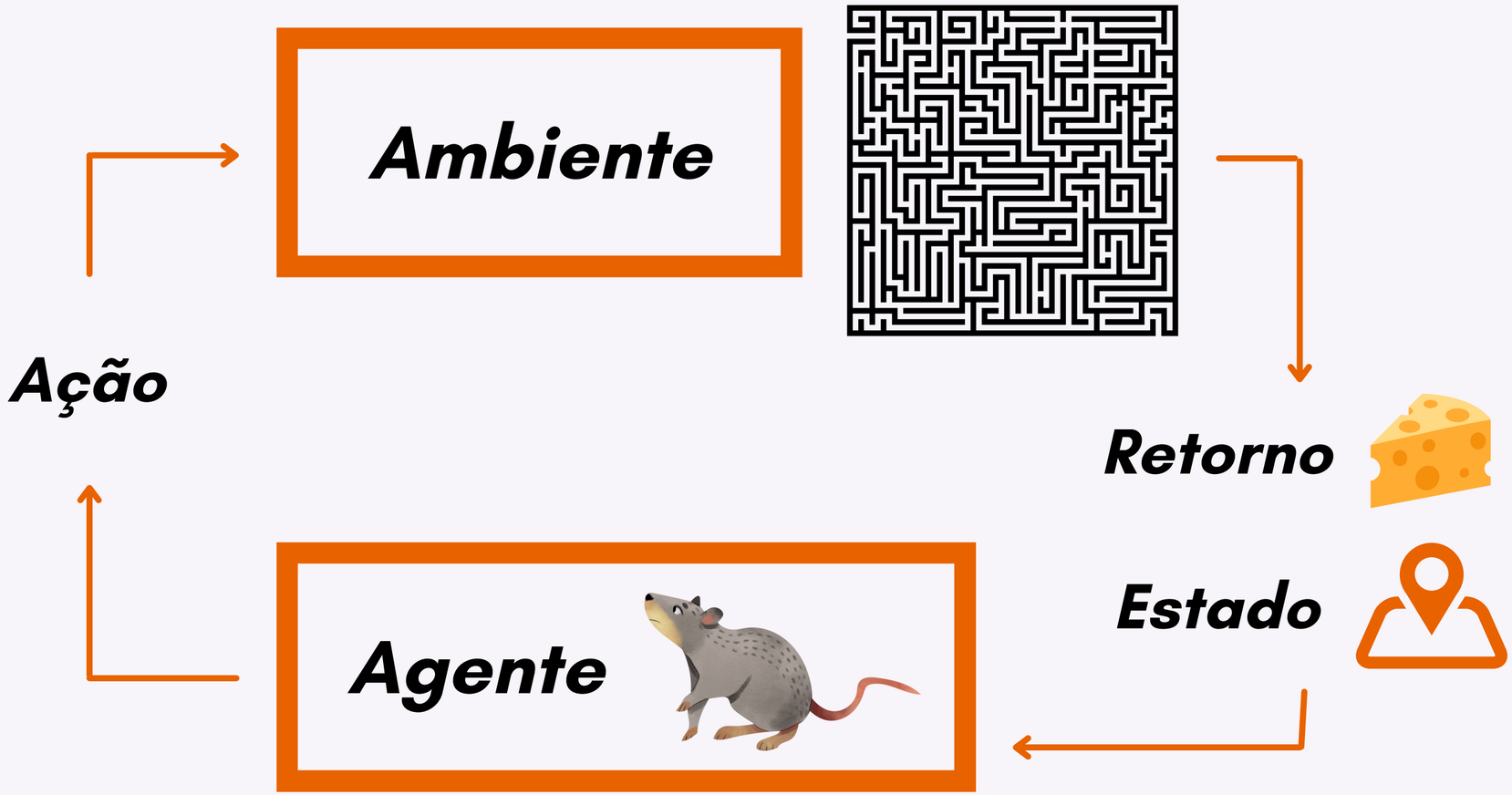
Diferenças:

- **Capacidade de avaliação**, não é um experimento instrutivo
- Esperamos que o agente aprenda a partir de sua própria experiência prévia
- Busca a **maximização** de valores de retorno

Tradeoff em R.L.



2. Conceitos básicos



AGENTE

Toma decisões, interage com o ambiente e processa informações

AMBIENTE

Aquilo que o agente não tem domínio completo sobre

MODELO

Imita o comportamento do ambiente

POLICY

Descreve a tomada de decisão do agente

REWARD SIGNAL

Define eventos bons e ruins imediatos

VALUE FUNCTION

Value é o montante de rewards que pode colher a partir de uma ação

Em métodos de reinforcement learning:

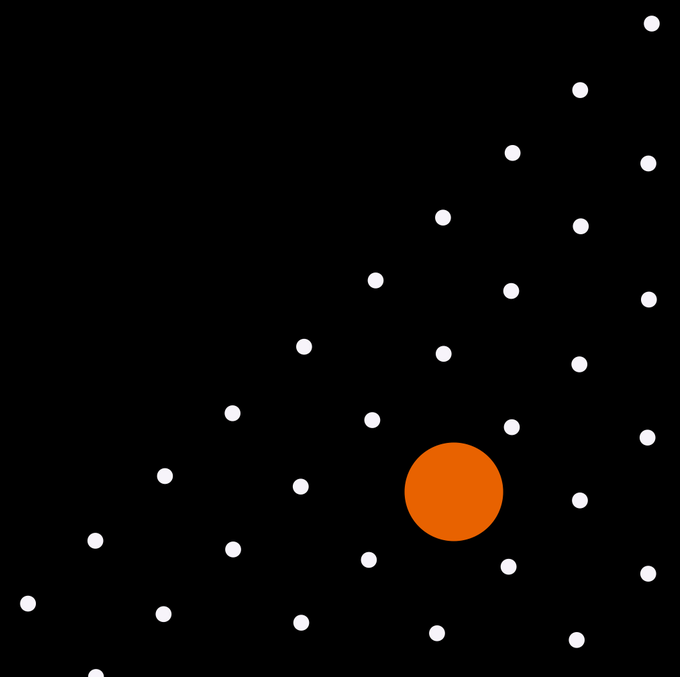
O agente busca encontrar uma ***optimal policy*** que maximize a sua *value function*, esta última definida pelos *reward signals* a cada decisão tomada

The n-armed bandits problem

- Agente se depara com n-máquinas caça-níquel
- Puxar uma alavanca gera um retorno
- Qual alavanca puxar?
- E se puder jogar 10 vezes? 1000 vezes?
- Exploit vs explore

Processos de Decisão de Markov

- Satisfazem a *propriedade de Markov*: os estados devem guardar em si toda (e apenas) informação relevante sobre o passado
- O agente consegue elaborar uma previsão dos estados e retornos futuros e selecionar sua ação seguinte com esse conhecimento



DYNAMIC PROGRAMMING

II.

3. Um método de R.L.

- É um método de otimização matemático e de programação computacional
- Simplificamos um problema complexo ao fracioná-lo em subproblemas mais simples
- A **equação de Bellman** relaciona os subproblemas ao problema original de forma recursiva
- Cuidado com a ***curse of dimensionality!***

FUNÇÃO OBJETIVO

É a função matemática que desejamos maximizar

VARIÁVEL DE ESTADO

Convém toda informação relevante para tomada de decisão

VARIÁVEL DE CONTROLE

Estão sob controle do agente

POLICY FUNCTION

Descreve a tomada de decisão do agente

VALUE FUNCTION

Value é o montante de *rewards* que pode colher a partir de uma ação

OPTIMAL POLICY

Policy function que maximiza a *value function*

Indução retroativa:

- Seja « *s* » a variável de estado, e « *a* » a variável de controle, e β um fator de desconto.
- Escrevemos a relação entre a *value function* de um período, ***V(s)***, e a *value function* do período seguinte, ***V(s')*** na Equação de Bellman:

$$V(s) = \max_{a} \{R(s,a) + \beta\, V(s')\}$$

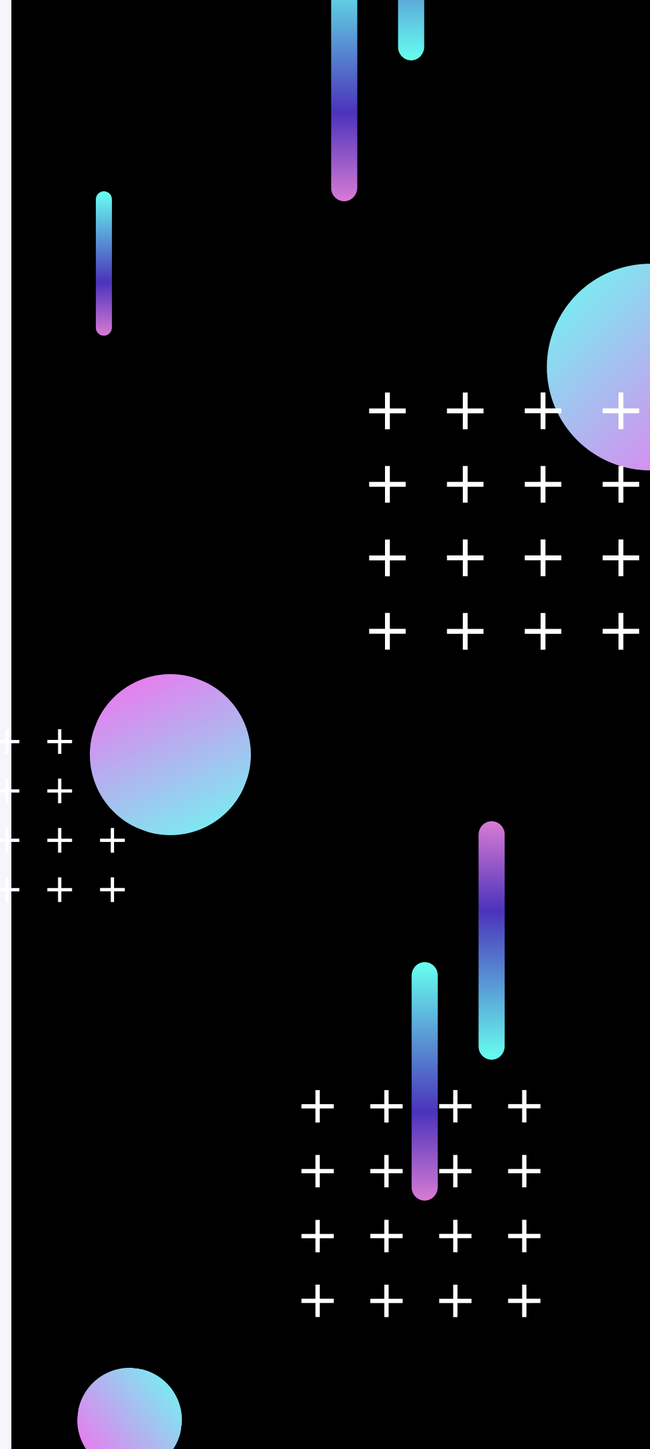
Value function iteration:

Começamos com uma estimativa inicial V^(0) e então usamos a equação de Bellman para obter uma estimativa atualizada de V^(1); repetimos para encontrar V^(2), e assim em diante, até que cada iteração subsequente não mude nossa estimativa:

$$\lim_{n \to \infty} ||V^{(n+1)} - V^{(n)}|| = 0$$

Value function iteration:

- O método percorre todo o set de variáveis estado, executando um ***backup*** completo para cada uma: atualiza os estados baseado nos valores e probabilidade de todos os possíveis estados futuros
- Se o backup não resulta em qualquer mudança no *value*, chegamos à **convergência** que satisfaz a equação de Bellman

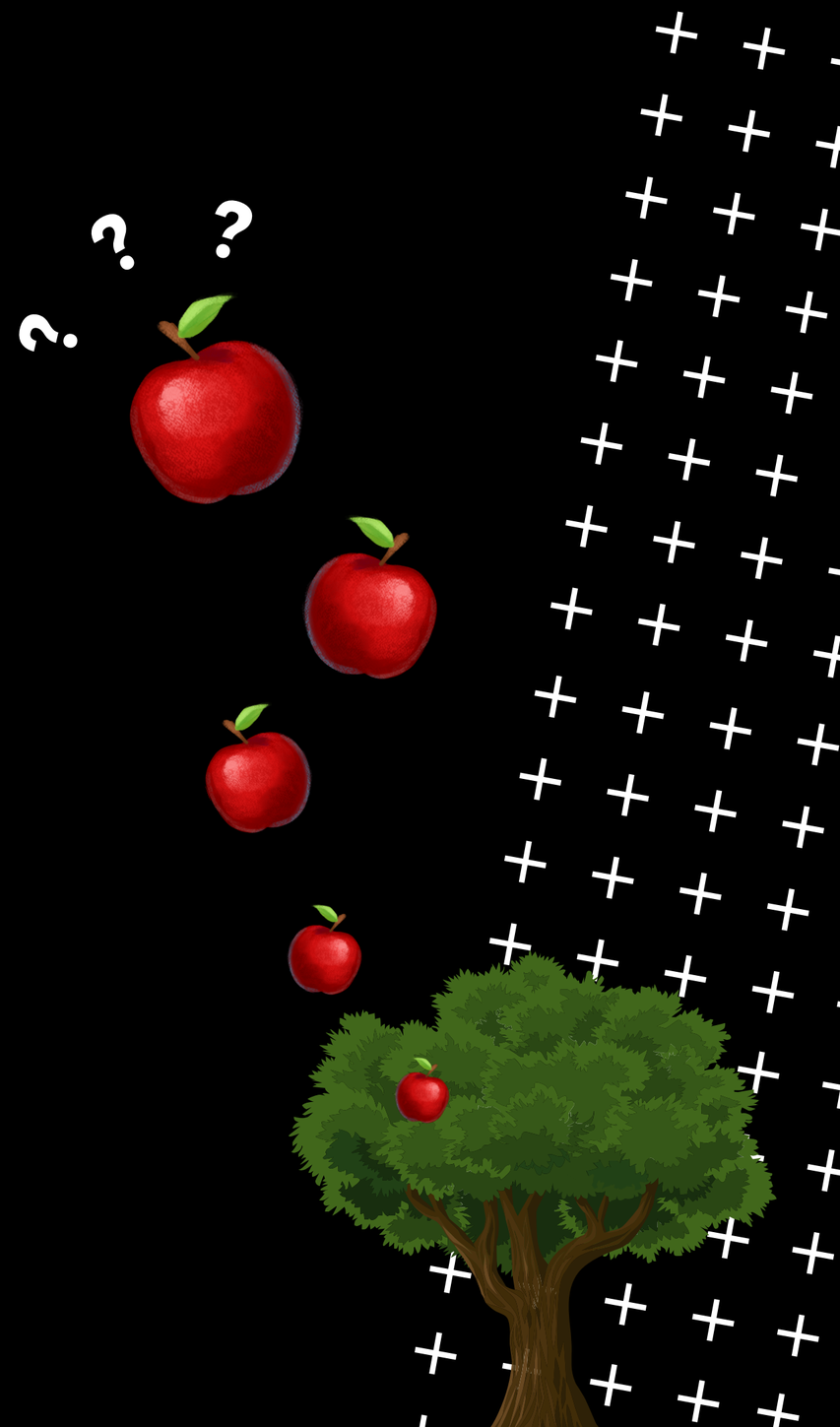


Problema do Planejador Central

- Consumo (c) vs investimento (k)
- Consumir gera retornos imediatos, e invisto para consumir mais no futuro
- Qual a regra de investimento ótimo?
- Posso usar a value function iteration!

$$V(k) = \max_{k'} \{u(c) + \beta V(k')\}$$

$$s.t.\ k' = k^{\alpha} - c$$



Problema do Planejador Central

- Suposição inicial: $V^{(0)}$

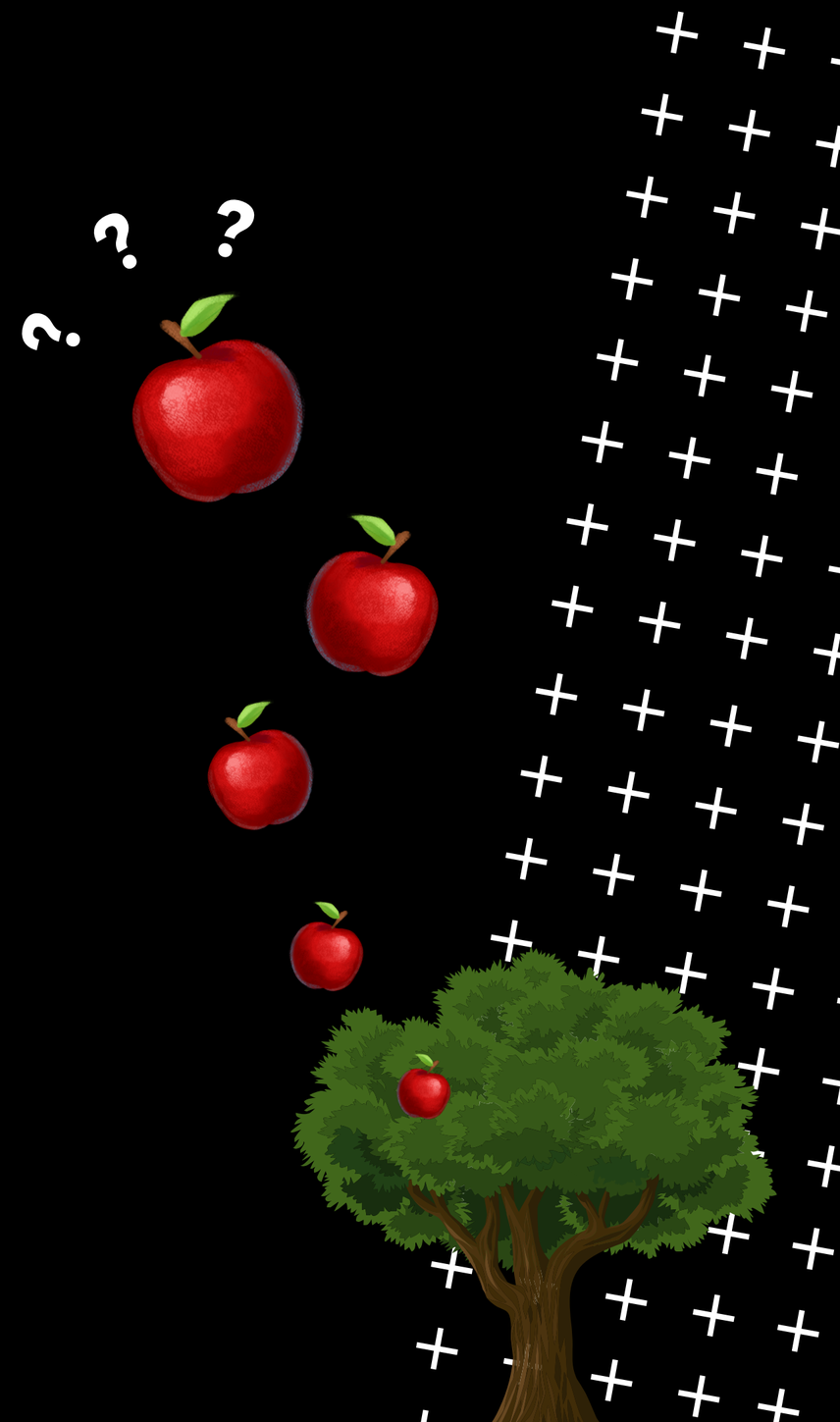
$$V^{(1)}(k) = \max_{k'} \{u(k^{\alpha} - k') + \beta V^{(0)}(k')\}$$

- Comparamos as funções valor;

  repetimos para $V^{(2)}$

- Repetimos até que:

$$\lim_{n \to \infty} ||V^{(n+1)} - V^{(n)}|| = 0$$



Referências e links úteis

- **Sargent, Thomas J.; Stachurski, John**

  Quantitative Economics with Python. Disponível em: <https://python.quantecon.org/intro.html>

- **Sims, Eric**

  Notes on Value Function Iteration. Disponível em: <https://www3.nd.edu/~esims1/val_fun_iter.pdf>

- **Sutton, Richard S.; Barto, Andrew G.**

  Reinforcement Learning: an introduction. MIT Press, 2015. Disponível em: <https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf>

Obrigado!