



# APRENDIZADO POR REFORÇO

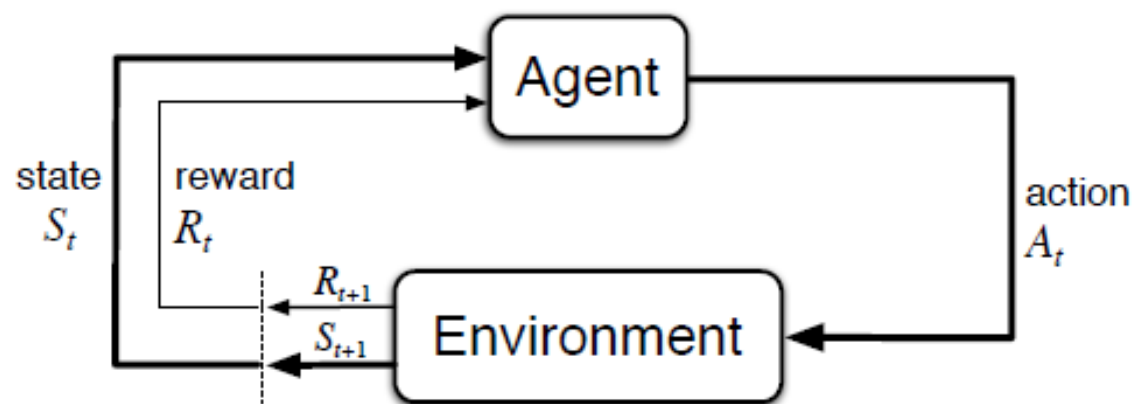
## Aula 8: Estudo de Caso

Lucas Pereira Cotrim  
Marcos Menon José

lucas.cotrim@maua.br  
marcos.jose@maua.br

# Relembrando últimas aulas

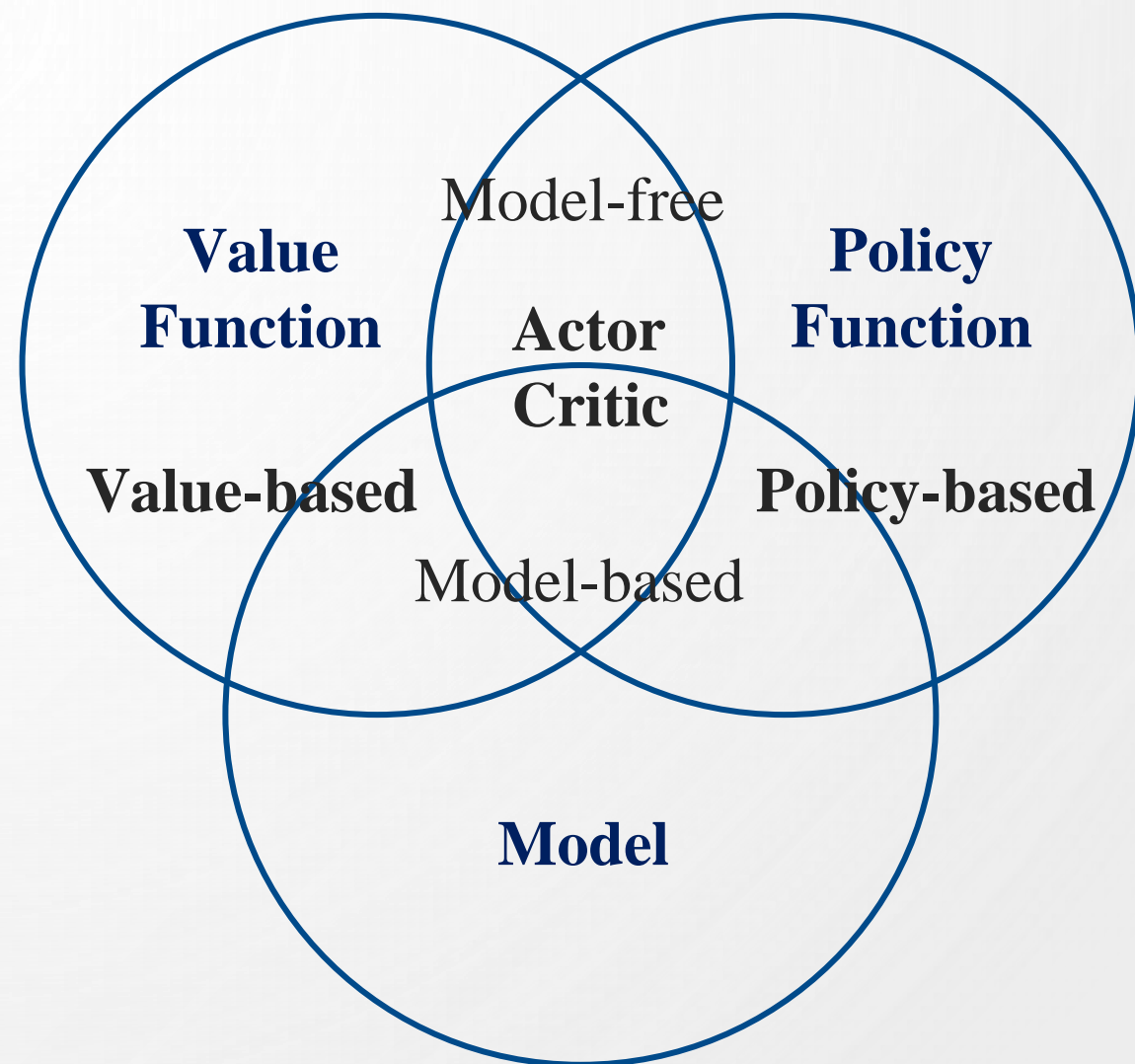
- Área de Aprendizado de Máquina associada a como agentes devem escolher ações em determinado ambiente com o objetivo de maximizar recompensas.
- Características do problema:
  - Agente possui sensores que observam o **estado** do **ambiente**.
  - O agente possui um conjunto de possíveis **ações** que pode tomar para alterar esse estado.
  - A cada ação tomada ele recebem uma **recompensa** que indica a qualidade da ação dado o estado.



# Relembrando últimas aulas

Algoritmos de aprendizado por Reforço podem ser classificados de acordo com os componentes treinados e presentes no agente:

- Value Function:  $V(s)$ ,  $Q(s, a)$
- Policy Function:  $\pi(a|s)$
- Model:  $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$



# Tópicos da Aula

- Apresentar algumas das vantagens e desvantagens de aplicações reais
- Apresentar exemplos de aplicações nas seguintes áreas:
  - Sistemas de Recomendação
  - *Bidding Optimization*
  - *Goal Oriented Chatbots*
  - Sistemas de controle de Tráfego
- Fazer um estudo de caso mais aprofundado de uma aplicação em robótica

# Quando utilizar aprendizado por reforço e suas vantagens

- Aplicações com tomada de decisões!
- Um agente que maximiza recompensas
- Aprendizado por reforço lida bem com evolução sequencial do tempo (como um jogo de xadrez)
- O agente é capaz de olhar no longo prazo (recompensas futuras)
- Não necessita de um dataset, consegue aprender através da interação com o ambiente
- É possível controlar a exploração
- Aprendizado similar ao aprendizado humano e animal

# Problemas do aprendizado por reforço e seus cuidados

- Ambiente precisa ser estruturado
- Em alguns casos pode ser difícil obter convergência. Há muitos hiperparâmetros
- A escolha da função recompensa deve ser precisa. Deve ser representado exatamente o que se deseja que o agente execute
- O agente pode escolher caminhos indesejados (por exemplos resolver problemas encontrando bugs)
- A amostragem é ineficiente (o agente precisa explorar bastante para aprender)
- Pode ser computacionalmente pesado dependendo do tamanho do espaço de estados e espaço de ações
- Aprendizado *offline* nem sempre traz bons resultados online e treinamento online pode ser custoso
- Problemas de segurança: precisa se programar muito bem as restrições para evitar problemas durante a exploração.
- Não utilizar quando há *target data* (dataset rotulado), pois aprendizado supervisionado é mais eficiente em termos de amostra (*sample efficient* )



# Como escolher o algoritmo? Model Based X Model Free

- Métodos Model-Based são indicados quando:
  - O modelo do MDP  $M = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$  é conhecido (jogos/regras) - Métodos com modelos conhecidos de Monte Carlo Search (AlphaZero, AlphaGo, ...)
  - Interações com ambiente real são custosas e é mais eficiente treinar um modelo aproximado e aprender por simulação (Deep Dyna-Q, I2A, SOLAR, ...)
- Métodos Model-Free são indicados quando:
  - A aplicação é uma simulação
  - Há uma base de dados que pode ser utilizada para treino e depois realizar *fine-tuning* usando o ambiente real
  - Interações com ambiente real não são custosas
  - Sua construção é mais simples e não requer tanto processamento

# Qual algoritmo Model Free escolher?

- Quando o espaço de ações é discreto (lembrando que é sempre possível discretizar um espaço de ações contínuo):
  - Algoritmos como DQN com suas adições (Target network, Double DQN, Dueling DQN, DQN with Prioritized experienced replay)
  - Usando multiprocessamento: PPO costuma ter a convergência mais rápida mas há outros como A2C
- Quando o espaço de ações é contínuo:
  - Uma primeira versão e mais simples pode ser construída com REINFORCE, mas sua convergência pode ser complicada
  - Usando multiprocessamento: PPO é o algoritmo mais estável enquanto A3C converge mais rápido
- Enquanto o DQN é o algoritmo mais famoso e mais utilizado, o PPO é hoje o algoritmo moderno mais estável (lembrando que novos algoritmos estão sendo desenvolvidos e isso pode mudar).

Fonte: <https://medium.datadriveninvestor.com/which-reinforcement-learning-rl-algorithm-to-use-where-when-and-in-what-scenario-e3e7617fb0b1>



Tarefa

# Tarefa: Estudo de Caso

- A última tarefa da disciplina é a criação de um estudo de caso sobre um tema livre
- O tema, se possível, deve ser algo próximo do trabalho/pesquisa de vocês. Caso não tenha uma aplicação muito direta, pode escolher outro tema
- Pode-se escolher um dos temas apresentados em sala de aula, mas é esperado que haja uma expansão dos conceitos

# Tarefa: Estudo de Caso

- A proposta é criar um texto de aproximadamente 3 páginas com a seguinte sugestão de formato (não é necessário seguir, apenas uma sugestão):
  - Tema: explicar brevemente o tema escolhido
  - Trabalhos correlatos: escolher e apresentar algo como 2 ou mais trabalhos correlacionados
  - Objetivo da Aplicação: o porquê de utilizar aprendizado por reforço
  - Espaço de Estados
  - Espaço de Ações
  - Função de Recompensa
  - Algoritmos estudados que poderiam ser escolhidos

# Tarefa: Estudo de Caso

- Lembrando que não é necessário entregar nenhum código, apenas o texto
- A ideia é a criação de um estudo de casos prévio a sua implementação.
- Pode ser algo completamente novo que ninguém implementou anteriormente
- O objetivo do trabalho é ser simples. Não se espera que seja uma ideia brilhante que resolva todos os problemas da área
- Qualquer dúvida, entre em contato conosco que iremos ajudar no que pudermos!

# Aplicações

- Vamos passar alguns exemplos de aplicações neste formato para ajudar na tarefa:
  - Sistemas de Recomendação
  - *Bidding Optimization*
  - *Goal Oriented Chatbots*
  - Sistemas de controle de Tráfego
- E um estudo de caso mais profundo apresentando resultados de uma aplicação que nós construímos em robótica

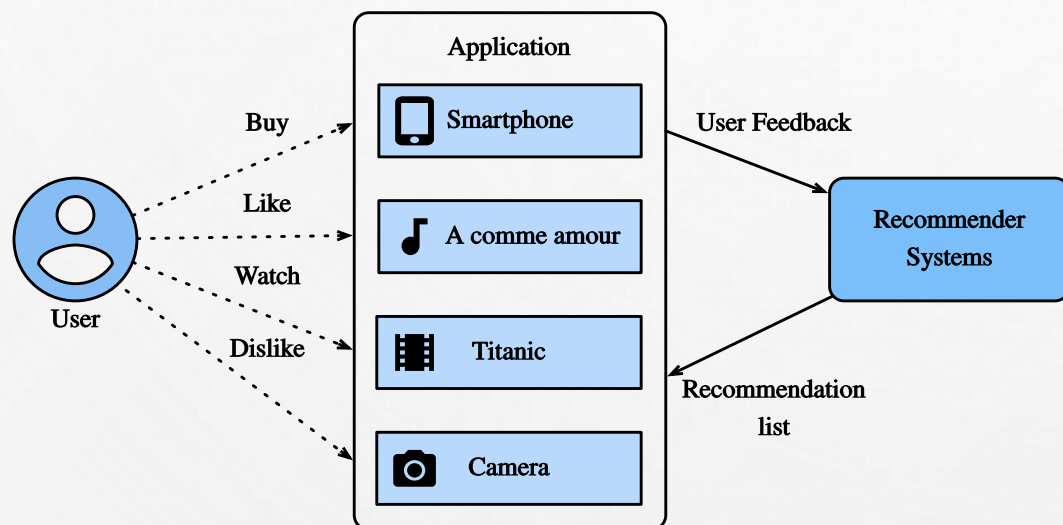


# Sistemas de Recomendação

Sistemas de Recomendação

# Sistemas de Recomendação

- Sistemas de Recomendação são softwares que tem como objetivo sugerir itens personalizados que maximizam o consumo de usuários
- Eles são aplicados em diversas áreas como redes sociais, e-commerce, canais de notícia e empresas como Facebook, Amazon, Youtube, Google e Spotify.



Fonte:

[https://d2l.ai/chapter\\_recommender-systems/recsys-intro.html](https://d2l.ai/chapter_recommender-systems/recsys-intro.html)

# Exemplos de sistemas de recomendação que usam RL

- Um estudo de caso feito pela equipe do Youtube:  
[https://www.youtube.com/watch?v=HEqQ2\\_1XRTs](https://www.youtube.com/watch?v=HEqQ2_1XRTs)
- E o respectivo paper: <https://arxiv.org/pdf/1812.02353.pdf>
- Como o Netflix usava aprendizado por reforço na escolha da arte do filmes que eram mostrados para os usuários: <https://netflixtechblog.com/artwork-personalization-c589f074ad76>
- Recomendação de páginas de um e-commerce usando DPPG (recomendo muito a leitura caso alguém tenha interessa de implementar): <https://dl.acm.org/doi/pdf/10.1145/3240323.3240374>
- Sistema de Recomendação utilizando Pseudo Dyna-Q (Model Based):  
<https://tbbaby.github.io/pub/wsdm20.pdf>

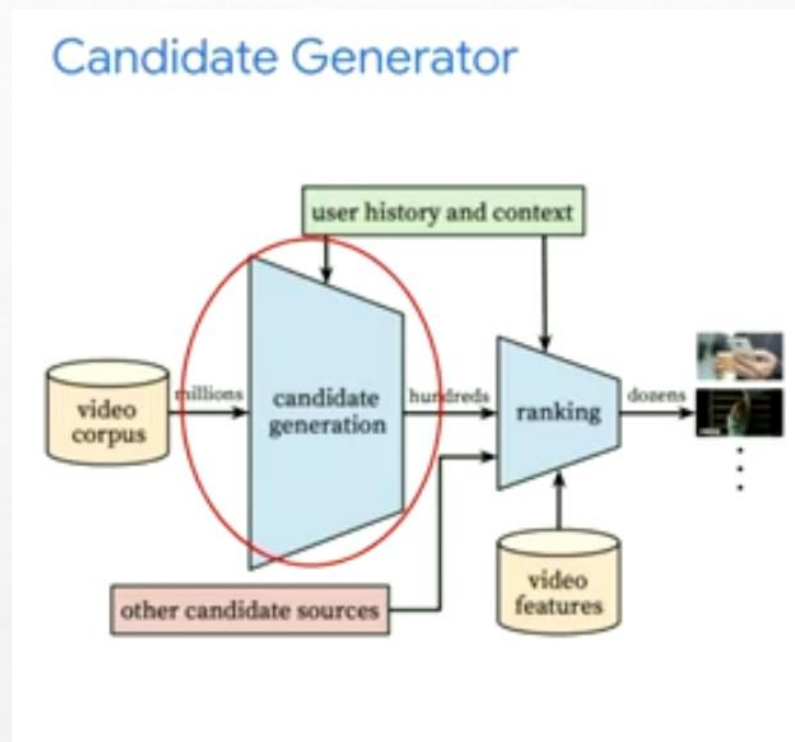
# Objetivo

- Existem diversos jeitos e técnicas de criar um sistema de recomendação, mas hoje vamos conferir técnicas de aprendizado por reforço.
- O exemplo que será seguido será o estudo de caso do Youtube
- O objetivo é recomendar vídeos que maximize o tempo do usuário consumindo na plataforma



# Arquitetura Proposta

- O aprendizado por reforço é utilizado dentro do *Candidate Generation*, que tenta reduzir quais vídeos podem ser recomendados
- Depois há um *ranking* que escolhe quais e a ordem que aparecem os vídeos (usando outro algoritmo)





# Espaço de Estados

- O espaço de estados neste caso pode ser dividido em algumas partes:
  - Detalhes do usuário: informações da conta google como idade, sexo
  - Vídeos consumidos previamente
  - Vídeos consumidos que foram indicados (interações passadas)
- Como o número de vídeos consumidos varia para cada usuário, uma boa opção é utilizar Redes Neurais Recorrentes (como LSTM) que conseguem através do *encoder* tratar essa diferença.

# Espaço de Ações

- Existem bilhões de vídeos disponíveis no Youtube o que torna impossível colocar todos como espaço de ações!
- A solução que utilizaram foi amostrar um conjunto de vídeos usando Fast Nearest Neighbor.
- Encontraram grupos de vídeos com características similares (seria possível utilizar técnicas de clusterização como K-means)

Fonte:

<https://twitter.com/pastelbio/status/999313016398237696?lang=ar>

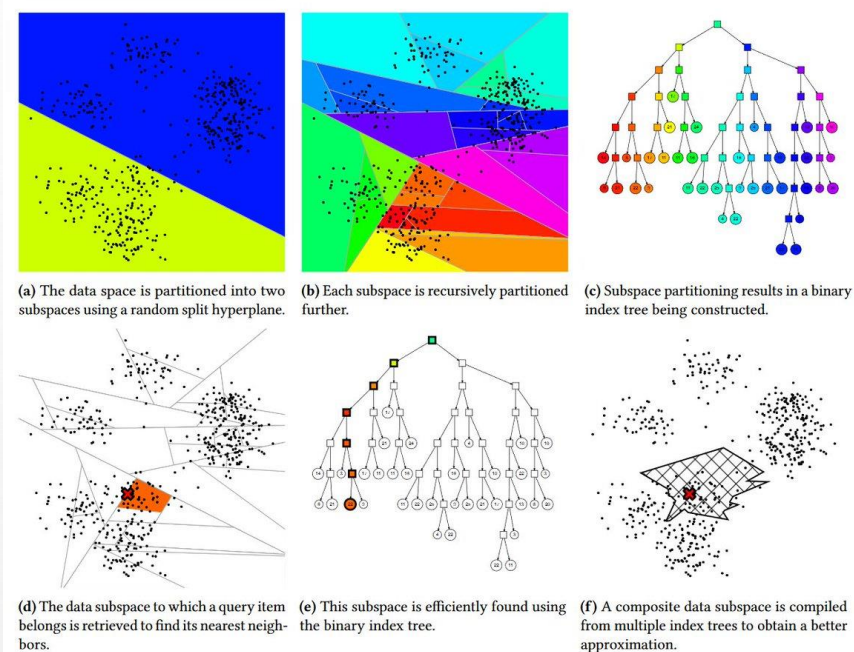


Figure 2: Approximate nearest neighbor indexing and searching using an ensemble of random split hyperplane index trees.

# Função Recompensa

- A recompensa neste caso é simples: a soma descontada do tempo gasto pelo usuário em cada vídeo, além de medir a interação e engajamento (curtidas, comentários).
- O objetivo é simples: quanto mais tempo o usuário passa e com mais atenção se encontra, mais propagandas ele consome!

Exponentially discounted future reward,

$$\mathcal{R}_{s_t, a_t} = r(s_t, a_t) + \gamma r(s_{t+1}, a_{t+1}) + \gamma^2 r(s_{t+2}, a_{t+2}) + \dots$$



Queen - Love of My Life

4 mins 



The Story of Queen's History Making (1971-2018)

30 mins



Astronauts - I Want World to Know a Thing About You Girl

4 mins

# Algoritmos Possíveis

- A equipe do Youtube utilizou uma modificação do algoritmo REINFORCE chamado Top-K REINFORCE, pois a ação do agente é escolher um conjunto de tamanho  $k$  de vídeos (não apenas 1).
- Uma outra opção possível seria utilizar uma saída contínua equivalente a cada vídeo amostrado (ou grupo de vídeos) e escolher os Top- $k$  com maior valor
- Como o treino online é custoso, duas soluções são recomendadas:
  - Utilizar um sistema model-based que simula o usuário e diminui as interações necessárias
  - Realizar um treinamento offline com a base de dados antiga ou simulada e realizar fine tuning online

# *Bidding Optimization*

*Bidding Optimization*



# Bidding Optimization

- Um das formas mais comuns de Marketing Digital é o modelo de leilão, no qual as empresas que querem colocar um anúncio competem entre si para colocar sua propaganda para uma audiência específica.
- O algoritmo deve conseguir maximizar o número de cliques em seu anúncio escolhendo a audiência e site certos e minimizar o dinheiro gasto.

# Exemplos de *Bidding Optimization* que usam RL

- O maior exemplo dessa aplicação na vida real é o grupo Alibaba (empresa de e-commerce chinesa) que apresenta uma série de trabalhos e artigos sobre o assunto:
  - Primeiro artigo da série: <https://arxiv.org/pdf/1803.00259.pdf>
  - Foco na restrição de orçamento: <https://arxiv.org/pdf/1802.08365.pdf>
  - Implementação multiagente: <https://arxiv.org/pdf/1802.09756.pdf>
  - Um artigo de jornal interessante: <https://analyticsindiamag.com/how-this-research-by-alibaba-group-has-used-reinforcement-learning-to-change-real-time-bidding/>
- Trabalho muito similar com restrição de orçamento: <https://sci-hub.se/https://ieeexplore.ieee.org/document/8916257>

# Objetivo

- Vamos focar no primeiro trabalho do Alibaba
- Neste caso, o agente tem uma quantidade de dinheiro  $B$  para gastar em um dia inteiro de anúncios (1 dia representa um episódio)
- Cada timestep é definido como um anúncio que aparece e que o agente tem que fazer uma oferta
- O objetivo é maximizar a quantidade de cliques recebidos nos anúncio, considerando a quantidade  $B$  de dinheiro no dia

# Espaço de Estados

- O espaço de estados é dividido em três informações diferentes:
  - Dinheiro restante para investir
  - Timestep  $t$  (quantos anúncios já ofertou)
  - Características do site e audiência que está leiloando o espaço para anúncio

**State  $s$ :** We design a general representation for states as  $s = \langle b, t, \overrightarrow{auct} \rangle$ , where  $b$  denotes the budget left for the ad,  $t$  denotes the step number of the decision sequence, and  $\overrightarrow{auct}$  is the auction (impression) related feature vector that we can get from the advertising environment. It is worth to note that, for generalization purpose, the  $b$  here is not the budget preset by the advertiser. Instead, it refers to the cost that the ad expects to expend in the left steps of auctions.

# Espaço de Ações

- Neste caso, o grupo decidiu discretizar a ação entre faixas de preço que o agente pode gastar
- Desta forma, há valores pré-estabelecidas e o agente escolhe uma delas

**Action  $a$ :** The decision of generating the real-time bidding price for each auction.



# Função Recompensa

- Nessa parte eles, o trabalho simplifica e considera apenas a quantidade total de anúncios ganhos como recompensa (o algoritmo apenas tenta comprar o maior número de anúncios sem se importar com a quantidade de cliques que isso vai gerar)

**Reward**  $r(s, a)$ : The income (in terms of *PUR\_AMT*) gained according to a specific action  $a$  under state  $s$ .

# Algoritmos Possíveis

- Como as ações foram discretizadas previamente, foi possível utilizar DQN como agente
- Caso fosse desejado manter as ações contínuas, outros algoritmos como PPO e DDPG poderiam ser utilizados no lugar
- O trabalho comenta que o DQN não foi muito eficiente no aprendizado (precisou amostrar 150 milhões de batchs) no treinamento online.
- Talvez técnicas model-based conseguissem atingir resultados com menos interações online ao simular um ambiente.

# *Goal oriented Chatbots*

*Goal Oriented Chatbots*

# Goal oriented Chatbots

- *Chatbots* orientados a objetivo (ou *Goal Oriented Chatbots*) ajudam o usuário a chegar a um objetivo predefinido dentro de um ambiente fechado.
- O *Chatbot* interage com o usuário através de texto com troca de diálogo ao longo de algumas interações
- Aplicações como *Chatbot* que ajudam a reservar e comprar tickets, resolver problemas técnicos, ajudar em vendas e muitos outros

# Exemplos de trabalho em *Goal oriented Chatbots*

- Um grupo de pesquisa desenvolveu uma aplicação para cinema com alguns materiais disponíveis incluindo seus códigos:
  - Primeiro artigo explicativo no Towards Data Science:  
<https://towardsdatascience.com/training-a-goal-oriented-chatbot-with-deep-reinforcement-learning-part-i-introduction-and-dce3af21d383>
  - Artigo publicado: <https://arxiv.org/pdf/1703.01008.pdf>
  - Github com todos os códigos: <https://github.com/maxbren/GO-Bot-DRL>
- Aplicação com *transfer learning*: <https://www.ijcai.org/proceedings/2018/0572.pdf>
- Aplicação com algoritmos hierárquicos: <https://arxiv.org/pdf/2005.11729.pdf>

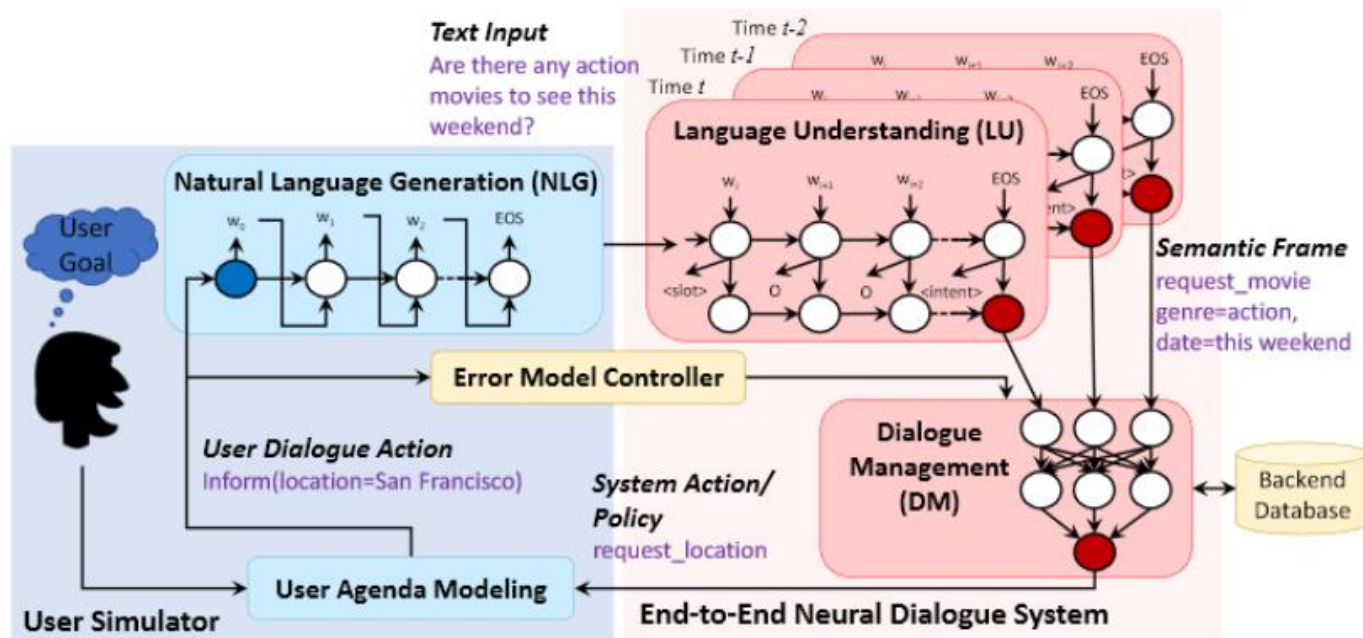
# Objetivo

- A técnica mais utilizada para resolver problemas de *Goal Oriented Chatbots* hoje é aprendizado por reforço, pois há sucessivas interações com o usuário e um objetivo claro para ser utilizado como recompensa
- Comparando com *chatbots* baseados em regras definidas por um programador, RL torna mais simples a programação pois o agente descobre sozinho como chegar no objetivo, porém é menos confiável (pode falar coisas sem sentido)
- Vamos focar em um *chatbot* (primeiro artigo) que ajuda o usuário a comprar e reservar um ingresso de cinema



# Arquitetura do projeto

- Nesse projeto, há mais módulos que apenas o com aprendizado por reforço.
- Vamos focar no *Dialogue Management* que recebe as intenções do usuário na última interação, além das interações anteriores



Dialogue flow for TC-Bot

# Espaço de Estados

- O espaço de estados é construído pelo módulo de *Language understanding* que captura as intenções do usuário através da leitura do texto
- Basicamente são informações como: nome do filme, gênero (uma pessoa pode perguntar quais os filmes de ação que estão em cartaz), data.
- E também a informação se o usuário está reservando ingresso ou se está buscando mais informações antes de fazer a reserva

# Espaço de Ações

- As ações são pré-definidas e são basicamente perguntas que devem ser feitas para o usuário como:
  - Qual o local deseja assistir?
  - Qual gênero de filme quer assistir?
  - Qual filme quer assistir?
  - Qual data e horário?
- E também pode informar o cliente caso tenha sido requisitada tal informação
  - Existem tais filmes de ação em cartaz
  - Os horários disponíveis são

# Recompensa

- A única recompensa do sistema é a resposta do usuário se ele conseguiu o que precisava (tirar suas dúvidas ou reservar o ticket)
- Há uma recompensa negativa no caso a troca de mensagens tenha sido uma falha e uma recompensa positiva no caso de sucesso
- Lembrando que aprendizado por reforço consegue aprender mesmo com recompensas futuras

# Algoritmos possíveis

- Como as ações são discretas, o autor escolheu o algoritmo DQN como agente
- Alguns algoritmos mais modernos com multiprocessamento poderiam ser empregados para agilizar o treino como o algoritmo PPO
- Como já há uma simulação de usuário construída, não há a necessidade de fazer model based (o autores mostram que a simulação é muito próxima de um usuário real)
- O objetivo é treinar com o usuário simulado para depois fazer *fine-tuning* no ambiente online numa proposta de aplicação real

Controle de Tráfego



# Controle de Tráfego

- O controle de Tráfego urbano é hoje extremamente importante para as cidades do mundo pois trânsito é hoje uma das maiores perdas de tempo, dinheiro e vidas além gerar poluição
- Existem diversas formas de ajudar no trânsito como:
  - Controle de semáforos
  - Construção de novas vias
  - Alterações nas regras de trânsito
  - Incentivos negativos a utilização de automóveis (impostos, rodízio, pedágio)
- Vamos focar aqui no controle de semáforos (tempo e coordenação)

# Exemplos de sistemas de recomendação que usam RL

Existem diversos papers, artigos e githubs abertos sobre aprendizado por reforço no controle de tráfego:

- Um paper que faz estudos com dados reais em Manhattan (parte mais rica da cidade de Nova Iorque): <https://chacha-chen.github.io/papers/chacha-AAAI2020.pdf>
- Uma conferência com diversos artigos: <https://traffic-signal-control.github.io/>
- Uma survey com diversos artigos listados:

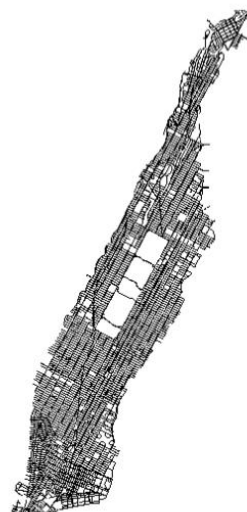
[https://www.kdd.org/exploration\\_files/5. CR. 30. Recent Advances in Reinforcement Learning for Traffic Signal Control A Survey of Models and Evaluation-2.pdf](https://www.kdd.org/exploration_files/5_CR_30_Recent_Advances_in_Reinforcement_Learning_for_Traffic_Signal_Control_A_Survey_of_Models_and_Evaluation-2.pdf)

# Objetivo

- O objetivo desta aplicação é controlar os semáforos de uma cidade (ou parte dela) para gerar o menor trânsito possível, sendo assim descentralizado (cálculo para cada cruzamento individual)
- A Aplicação que vamos seguir utiliza dados reais da ilha de Manhattan para treino



(a) Manhattan



(b) Manhattan in simulator

Figure 6: Road network of Manhattan in our experiments.

# Arquitetura do Projeto

- O grupo definiu que o agente é chamado diversas vezes para cada um dos semáforos
- O que coordena cada uma das intersecções é um algoritmo chamado FRAP específico para controle de sinais

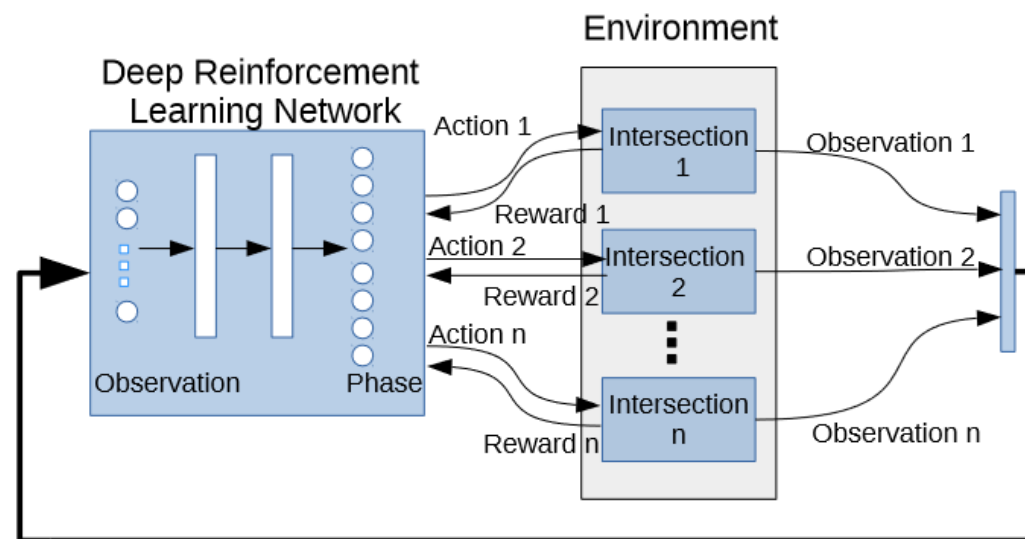


Figure 4: The framework of MPLight for multi-intersection signal control.

# Espaço de Estados

- O espaço de estados é definido como *pressure*, diferença na quantidade de carros que saiu do cruzamento contra o que entrou, para cada um dos sentidos

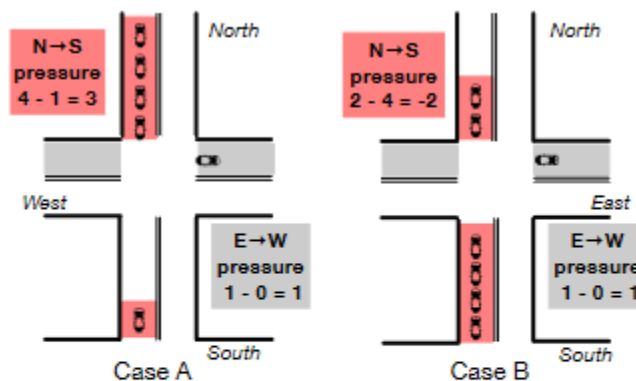
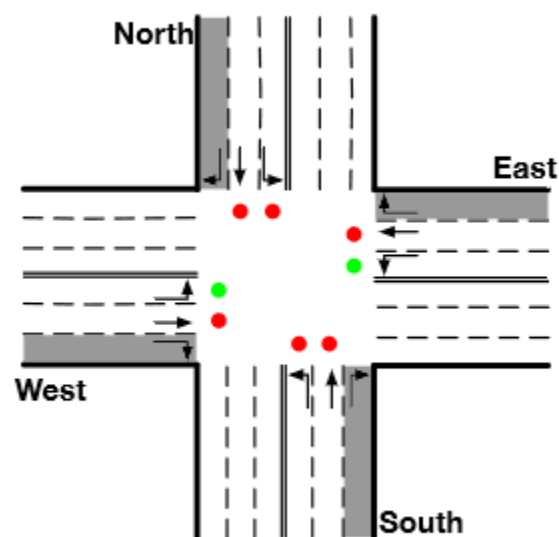


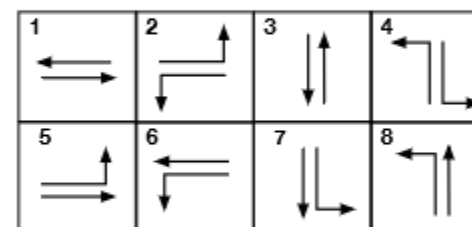
Figure 1: Illustration of max pressure control in two cases (Wei et al. 2019a). In Case A, green signal is set in the North→South direction; in Case B, green signal is set in the East→West direction.

# Espaço de Ações

- O paper divide em 8 possíveis ações: escolher um dos 8 modelos de tráfego para o semáforo



(a) Intersection

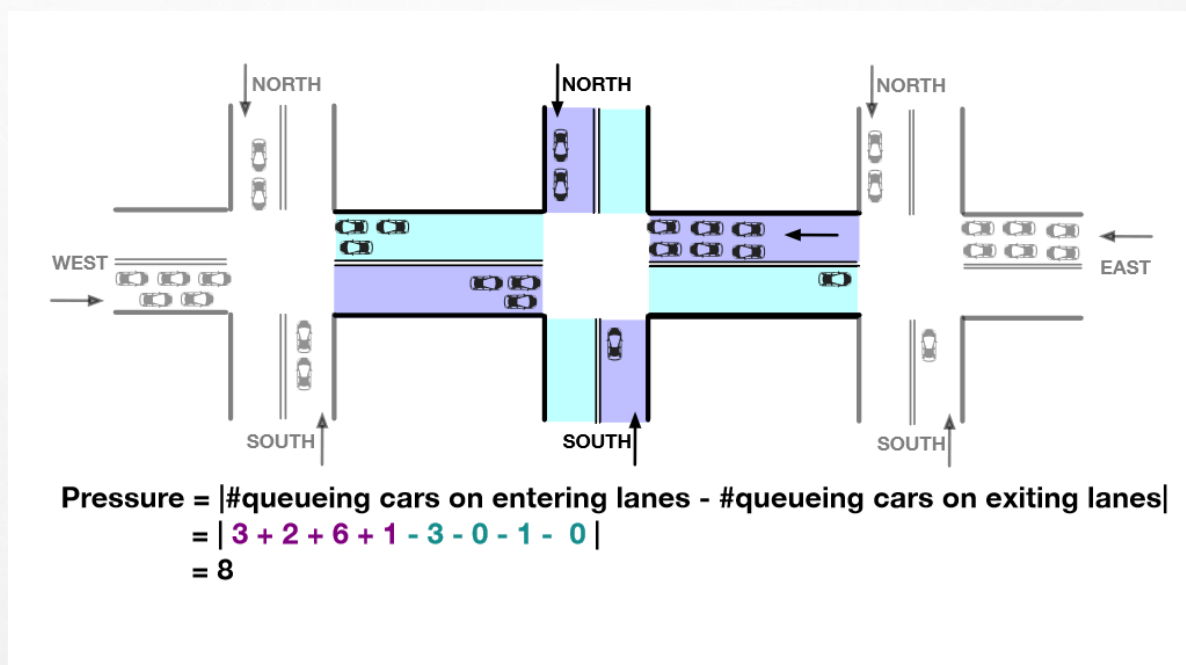


(b) Eight phases



# Função Recompensa

- A função recompensa é a diferença entre a quantidade de carros que saiu de filas de trânsito e de carros que entraram em novas filas
- Isso significa uma recompensa positiva quando o trânsito diminuiu e negativa quando ele aumentou



# Algoritmos Possíveis

- Nesse caso a rede é simples e como a saída é discreta, DQN parece uma boa solução
- O treino ocorre com dados antigos da ilha de Manhattan antes de ser implementado
- Conseguiu resultados a par com o estado da arte

Model	Travel Time			
	Config 1	Config 2	Config 3	Config 4
FixedTime	573.13	564.02	536.04	563.06
MaxPressure	361.17	402.72	360.05	406.45
GRL	735.38	758.58	771.05	721.37
GCN	516.65	523.79	646.24	585.91
NeighborRL	690.87	687.27	781.24	791.44
PressLight	354.94	353.46	348.21	398.85
FRAP	340.44	298.55	361.36	598.52
<b>MPLight</b>	<b>309.33</b>	<b>262.50</b>	<b>281.34</b>	<b>353.13</b>

# Trading – Trabalho T2

[+ Código](#)[+ Texto](#)

## Trabalho T2: DDPG Portfolio Management

Neste trabalho vamos treinar um robô trader utilizando uma adaptação do trabalho do AI4Finance a partir do seguinte github

<https://github.com/AI4Finance-LLC/FinRL>

**Objetivo do trabalho:** Criação de agente DDPG com biblioteca keras-rl2 e treinamento em ambiente FinRL para tarefa de *portfolio management* de 15 ações entre as mais negociadas na bolsa brasileira no período entre 2008 e 2020.

**O que o agente faz:** O agente escolhe como alocar os recursos entre as 15 ações escolhidas, com uma porcentagem de alocação para cada ação. Assim, trata-se de um problema de ações contínuas, no qual métodos *Value-Based* não poderiam ser aplicados.

**Separação entre teste e treino:** O agente deve ser treinado com os dados entre 2008 e 2018 e testado com os dados entre 2019 e 2020.

**Lembrando:** está é uma simplificação de um modelo real. Lembre que há taxas (de corretagem), impostos, cálculos de risco (como sharpe ratio que mostra que nem sempre mais rentabilidade significa o melhor resultado) e muitas outras complexidades envolvidas numa aplicação real. Recomendamos a quem tiver interesse estudar diversas aplicações e inclusive os 4 papers disponibilizados pela AI4Finance para aplicação online reais. Mas, apesar de tudo isso, é possível notar o poder dos algoritmos de aprendizado por reforço no auxílio da prática de trading.

# Trabalho interessantes de Trading usando RL

- Hoje, aprendizado por reforço é considerada uma técnica estado da arte para Stock Trading em análise Quant
- Os primeiros links são relativos ao grupo AI4Finance que deu origem a biblioteca adaptada do Trabalho T2
  - Github da biblioteca: <https://github.com/AI4Finance-LLC/FinRL>
  - Paper sobre Stock Trading: <https://arxiv.org/abs/2011.09607>
  - Paper sobre estratégias de liquidação: <https://arxiv.org/abs/1906.11046>
- E outros trabalhos interessantes:
  - [https://www.sciencedirect.com/science/article/pii/S0020025520304692?casa\\_token=0yzUegpoMGEA AAAA:xtk1hl7KnZ3VWfLcIwBOHTna\\_6M-noZtCsbNQ1u0yYVR-dyh2gnKzfIvEyKNLJkKIdxE\\_4\\_9Xu4](https://www.sciencedirect.com/science/article/pii/S0020025520304692?casa_token=0yzUegpoMGEA AAAA:xtk1hl7KnZ3VWfLcIwBOHTna_6M-noZtCsbNQ1u0yYVR-dyh2gnKzfIvEyKNLJkKIdxE_4_9Xu4)
  - <https://link.springer.com/content/pdf/10.1007/s00607-019-00773-w.pdf>

# Existem muitas outras aplicações!

Abaixo estão alguns links que fazem compilados de trabalhos de aprendizado por reforço:

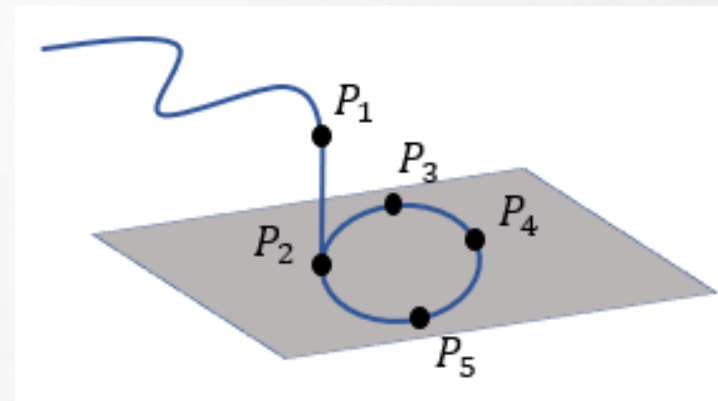
- <https://medium.com/@yuxili/rl-applications-73ef685c07eb>
- [https://neptune.ai/blog/reinforcement-learning-applications?utm\\_source=datacamp&utm\\_medium=crosspost&utm\\_campaign=blog-reinforcement-learning-applications&utm\\_campaign=News&utm\\_medium=Community&utm\\_source=DataCamp.com](https://neptune.ai/blog/reinforcement-learning-applications?utm_source=datacamp&utm_medium=crosspost&utm_campaign=blog-reinforcement-learning-applications&utm_campaign=News&utm_medium=Community&utm_source=DataCamp.com)
- <https://www.deeplearningbook.com.br/aplicacoes-da-aprendizagem-por-reforco-no-mundo-real/>
- <https://towardsdatascience.com/applications-of-reinforcement-learning-in-real-world-1a94955bcd12>

Robótica



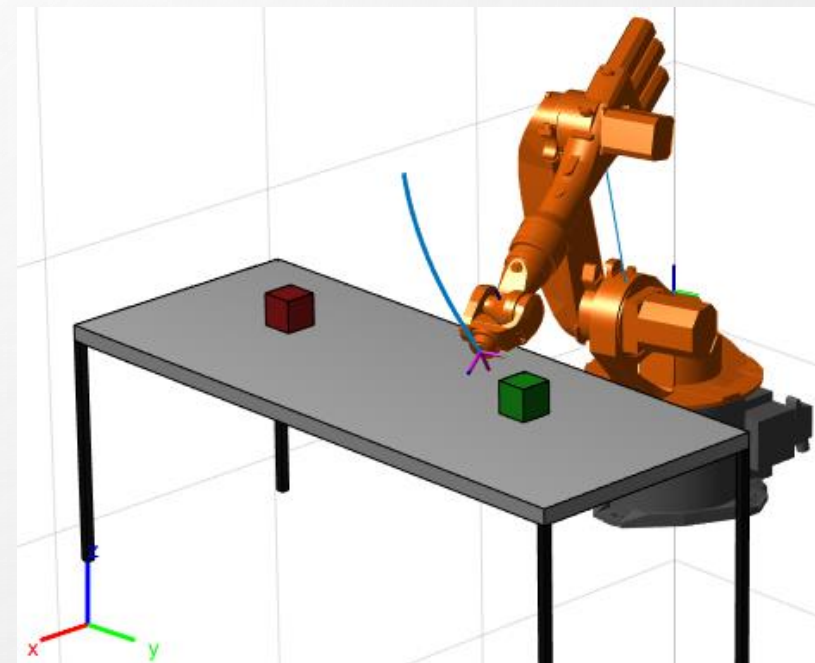
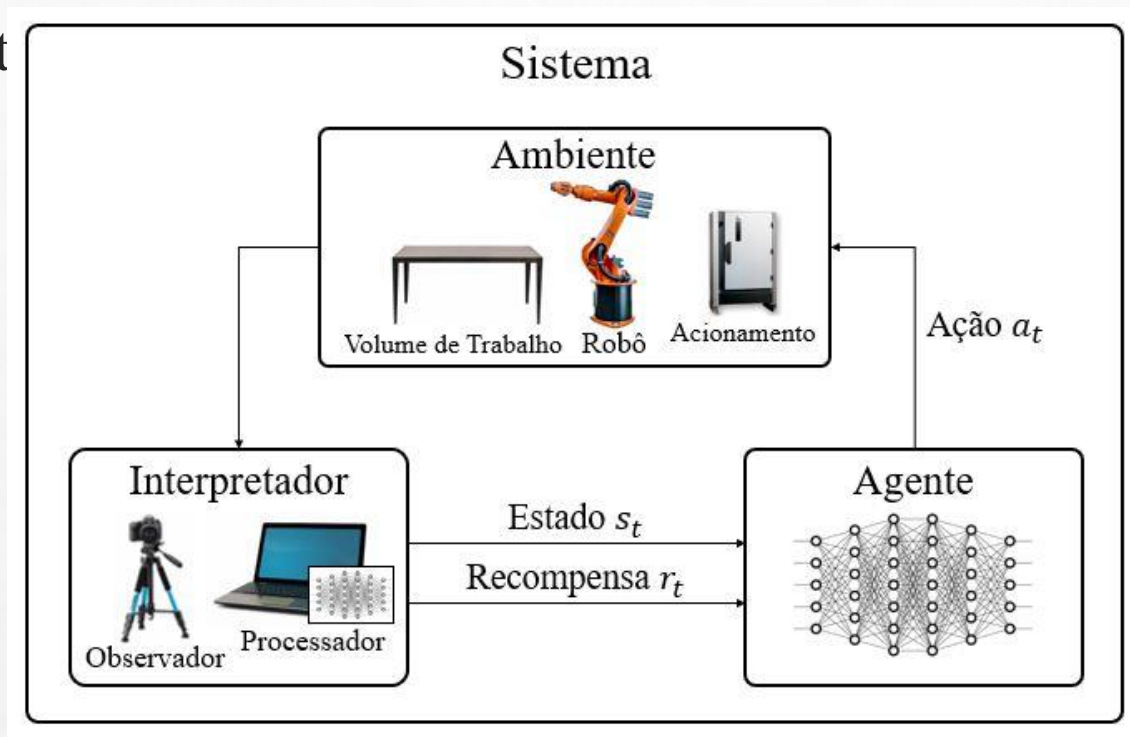
# Projeto Kuka rl: Motivação

- Programação de robôs industriais é lenta.
  - Definição manual de pontos no espaço de trabalho.
  - Interpolação de trajetórias do robô entre pontos definidos.
- Problemas:
  - Baixa flexibilidade para modificação de tarefa a ser executada pelo robô.
  - Procedimento manual e demorado.
- Ideia:
  - Treinamento de agente de Aprendizado por Reforço para controle automático de robô manipulador.



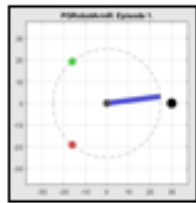
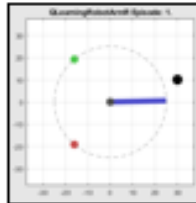
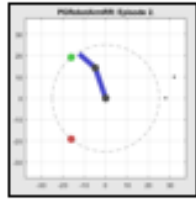
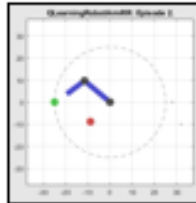
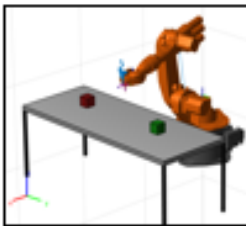
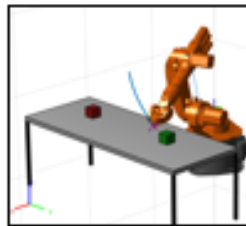
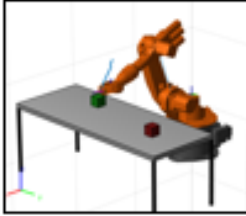
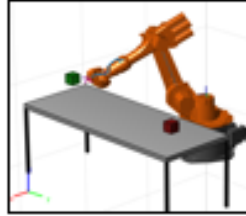
# Projeto Kuka rl: Detalhamento do projeto

- **Objetivo:** Controle de atuadores de robô manipulador Kuka-KR16 por meio de Aprendizado por Reforço para aprendizado automático de trajetórias de posicionamento e capacidade de desvio de obst



# Projeto Kuka rl: Detalhamento do projeto

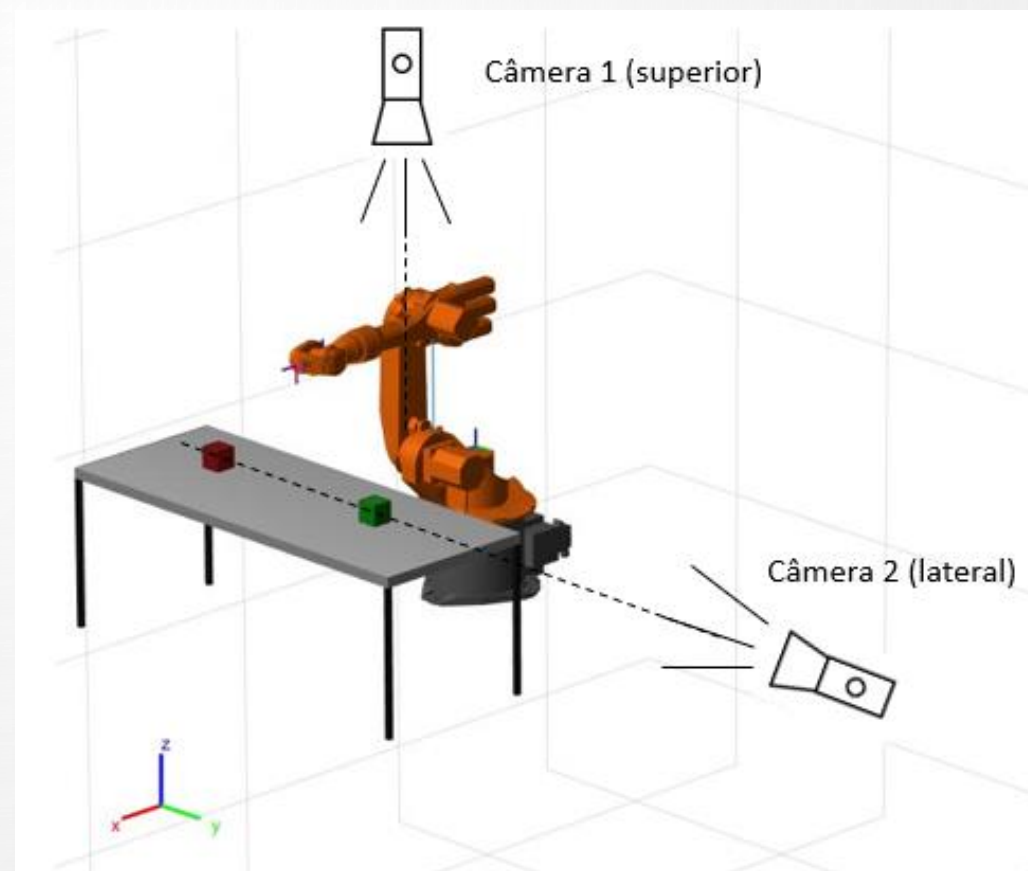
- **Projetos de Teste:**
  - Objetivo: Análise de convergência e comparação de funções de recompensa em versões simplificadas do problema original
  - Espaços de estados e ações de menor dimensão.
  - Estados: Posições Angulares de articulações do robô.

Projetos de Teste	REINFORCE	Q-Learning
1 Grau de Liberdade (R)		
2 Graus de Liberdade (RR)		
6 Graus de Liberdade (6R), configuração fixa		
6 Graus de Liberdade (6R), configurações aleatórias		

# Projeto Kuka rl: Detalhamento do projeto

## Espaço de Estados do Projeto Final

- Cada estado  $s$  é um conjunto de duas imagens RGB de tamanho  $24 \times 24 \times 3$  (dimensão 3456)
- Configuração de câmeras para visualização lateral e superior



# Projeto Kuka rl: Detalhamento do projeto

## Espaço de Ações do Projeto Final

- Cada ação  $a$  corresponde a uma combinação de variações angulares de cada articulação.
- $\Delta\theta_i = 1^\circ, i = 1, \dots, 6$

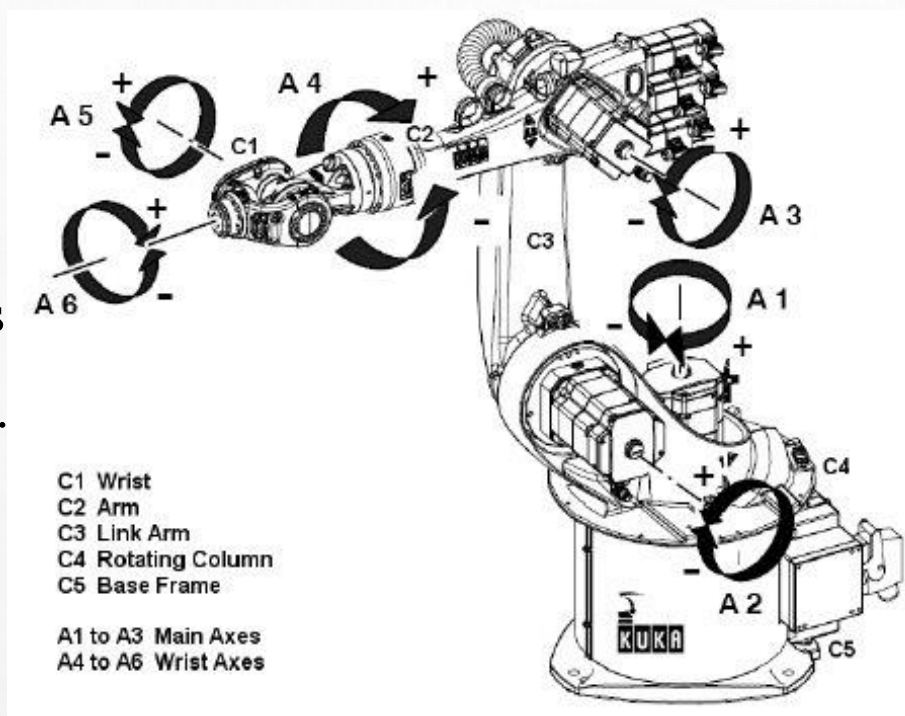


Figura 3: Diagrama de Articulações de robô KUKA-KR16 (fonte: KUKA ROBOTICS: Kuka KR6, KR16, KR16 L6, KR 16S specification, 2003)

$$A = \begin{matrix} & \begin{matrix} a_{A_1} & a_{A_2} & a_{A_3} & a_{A_4} & a_{A_5} \end{matrix} \\ \begin{matrix} \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \end{matrix} & \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & -1 \\ 1 & 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 & 0 \\ 1 & 1 & 1 & -1 & -1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -1 & -1 & -1 & -1 & 1 \\ -1 & -1 & -1 & -1 & 0 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix} \end{matrix}$$

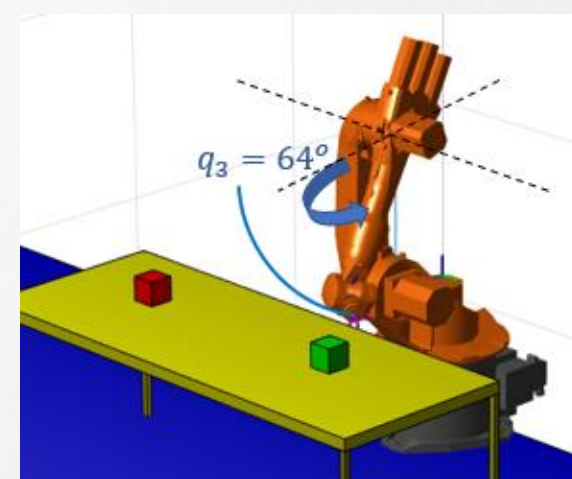
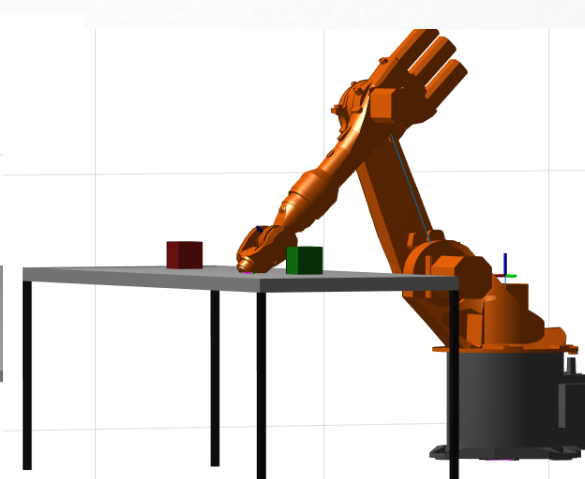
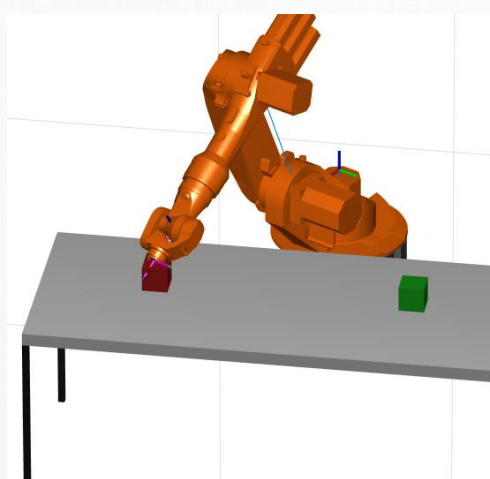
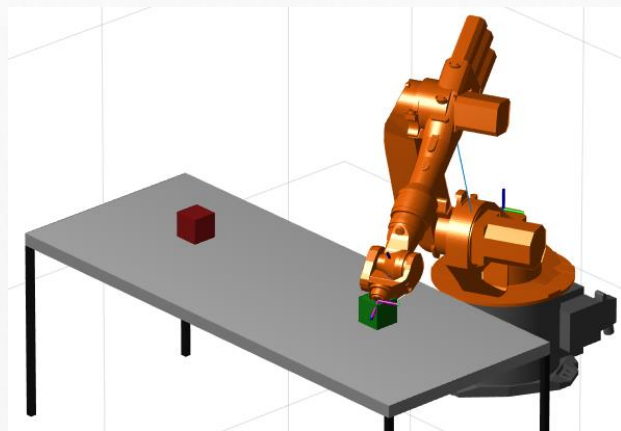
243 ações discretas



# Projeto Kuka rl: Detalhamento do projeto

## Estados Terminais

- Posição Desejada Alcançada
- Colisão com Obstáculo ou Mesa
- Limite de Curso de Articulação



- Simulação de trajetória é encerrada se
  - Um dos três critérios de parada é satisfeito
  - Número máximo  $T$  de transições é alcançado
- O agente recebe um bônus ou uma penalidade em função do critério de parada



Funções de Recompensa

# Projeto Kuka rl: Funções de Recompensa

## Funções de Recompensa

- Comparação entre diferentes funções de recompensa em projetos de teste
- Determinação de função mais adequada para projeto final

### $r_1(s, a)$ : Distâncias Absolutas

$$\begin{aligned}
 r_1(s, a) = & \\
 & k(r_{setpoint}(s, a) + r_{obstacle}(s, a) + c) \\
 & + B_{goal}(s, a) \\
 & + P_{joint\ boundary}(s, a) \\
 & + P_{collision}(s, a)
 \end{aligned}$$

Onde:

$$\begin{aligned}
 r_{setpoint}(s, a) &= -k_1 \left\| \mathbf{p}_{setpoint} - \mathbf{p}'_{ef} \right\|_2 \\
 r_{obstacle}(s, a) &= k_2 \left\| \mathbf{p}_{obstacle} - \mathbf{p}'_{ef} \right\|_2
 \end{aligned}$$

# Projeto Kuka rl: Funções de Recompensa

## Funções de Recompensa

- Comparação entre diferentes funções de recompensa em projetos de teste
- Determinação de função mais adequada para projeto final

$r_1(s, a)$ : Distâncias Absolutas

$r_2(s, a)$ : Aproximação ou Distanciamento

$$\begin{aligned}
 r_2(s, a) = & \\
 & k_s r_{setpoint}(s, a) + k_o r_{obstacle}(s, a) \\
 & + B_{goal}(s, a) \\
 & + P_{joint\ boundary}(s, a) \\
 & + P_{collision}(s, a)
 \end{aligned}$$

Onde:

$$r_{setpoint}(s, a) = \begin{cases} -1, & \text{se } \|p_{setpoint} - p'_{ef}\|_2 > \|p_{setpoint} - p_{ef}\|_2 \\ 0, & \text{se } \|p_{setpoint} - p'_{ef}\|_2 = \|p_{setpoint} - p_{ef}\|_2 \\ 1, & \text{se } \|p_{setpoint} - p'_{ef}\|_2 < \|p_{setpoint} - p_{ef}\|_2 \end{cases}$$

distanciamento

aproximação

# Projeto Kuka rl: Funções de Recompensa

## Funções de Recompensa

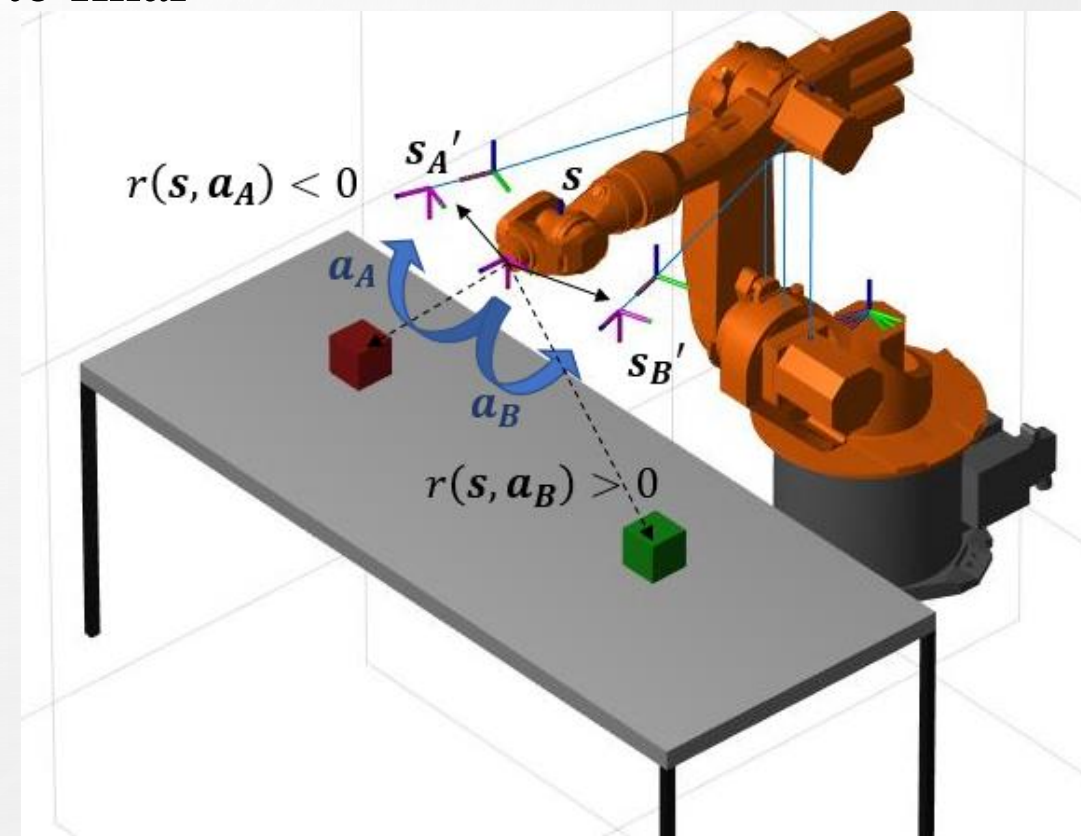
- Comparação entre diferentes funções de recompensa em projetos de teste
- Determinação de função mais adequada para projeto final

$r_1(s, a)$ : Distâncias Absolutas

$r_2(s, a)$ : Aproximação ou Distanciamento

$r_3(s, a)$ : Projeção de Vetor Deslocamento

$$r_3(s, a) = [k_s r_{setpoint} + k_o r_{obstacle}] + B_{goal} + P_{joint\ boundary} + P_{collision}$$



# Projeto Kuka rl: Funções de Recompensa

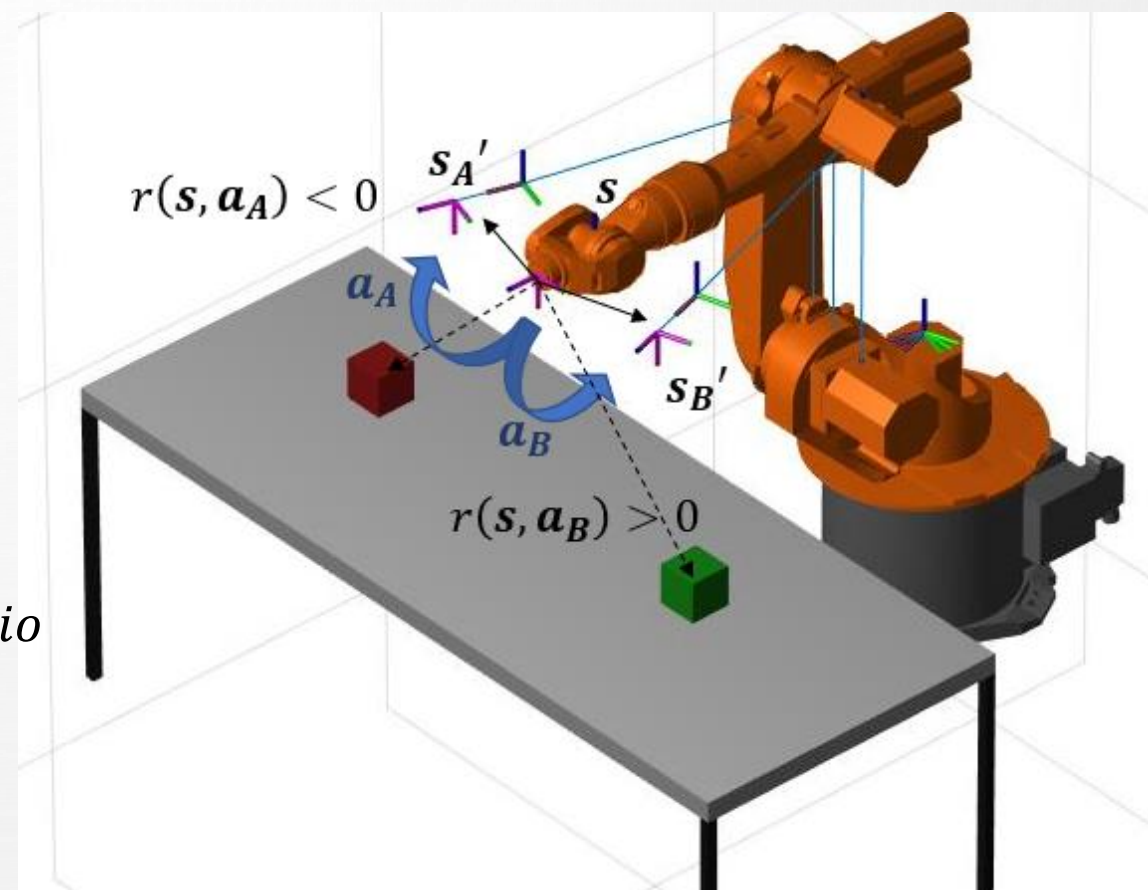
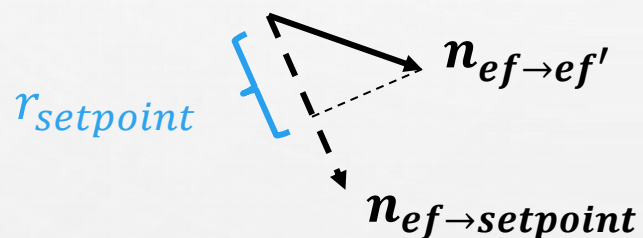
## Função de Recompensa Escolhida: $r_3(s, a)$

Posição de destino e de obstáculo

$$r_3(s, a) = [k_s r_{\text{setpoint}} + k_o r_{\text{obstacle}}] + \underbrace{B_{\text{goal}}}_{\text{Bônus}} + \underbrace{P_{\text{joint boundary}} + P_{\text{collision}}}_{\text{Penalidades}}$$

$$r_{\text{setpoint}}(s, a) = \mathbf{n}_{ef \rightarrow ef'} \cdot \mathbf{n}_{ef \rightarrow \text{setpoint}}$$

$$r_{\text{obstacle}}(s, a) = \begin{cases} 0, & \text{se } \|\mathbf{p}_{ef'} - \mathbf{p}_{\text{obstacle}}\| > r_{\text{infl}} \\ -(\mathbf{n}_{ef \rightarrow ef'} \cdot \mathbf{n}_{ef \rightarrow \text{obstacle}}), & \text{caso contrário} \end{cases}$$





# Projeto Kuka rl: Algoritmos

## Algoritmos Implementados

### REINFORCE

#### Algoritmo 1: REINFORCE Episódico

- Inicializa Robô, *setpoint*, *obstáculo*, estado inicial  $\mathbf{s}_0$  e espaço de ações  $\mathcal{A}$ ;
- Inicializa Hiperparâmetros (bônus e penalidades, tamanho da rede, número de *timesteps*, trajetórias e épocas, fator de desconto  $\gamma$  e taxa de aprendizado  $\alpha$ );
- Inicializa estruturas de armazenamento de épocas e políticas de ações;
- Inicializa política de ações parametrizada  $\pi_{\theta_0}$  aleatória e armazena em PolicyBuffer(1);

Gera N trajetórias  $\{\tau_n\}_{n=1}^N$  a partir de política de ações  $\pi_{\theta_0}$ , onde

$$\tau_n = \mathbf{S}_0, \mathbf{A}_0, R_0, \dots, \mathbf{S}_{T-1}, \mathbf{A}_{T-1}, R_T;$$

Calcula retornos  $\{G_t\}_{t=0}^{T-1}$  e armazena em cada trajetória;

Armazena  $\{\tau\}_{n=1}^N$  em EpochBuffer(1);

for  $ep \leftarrow 2$  to  $MaxEpoch$  do

    Aplica Gradiente Ascendente sobre  $\pi_{ep-1}$  para obter  $\pi_{ep}$ :

$$\theta_{ep} \leftarrow \theta_{ep-1} + \alpha \frac{1}{N} \sum_{n=1}^N \sum_{t=0}^{T-1} G_t \frac{\nabla \pi_{\theta}(\mathbf{s}_t, \mathbf{a}_t)}{\pi_{\theta}(\mathbf{s}_t, \mathbf{a}_t)};$$

    Armazena  $\pi_{ep}$  em PolicyBuffer(ep);

    Gera N trajetórias  $\{\tau_n\}_{n=1}^N$  a partir de política de ações  $\pi_{\theta_{ep}}$ , onde

$$\tau_n = \mathbf{S}_0, \mathbf{A}_0, R_0, \dots, \mathbf{S}_{T-1}, \mathbf{A}_{T-1}, R_T;$$

    Calcula retornos  $\{G_t\}_{t=0}^{T-1}$  e armazena em cada trajetória;

    Armazena  $\{\tau\}_{n=1}^N$  em EpochBuffer(ep);

    Mostra performance média da época atual;

    Mostra melhor trajetória da época atual;

end

## DQN

#### Algoritmo 2: DQN

- Inicializa Robô, *setpoint*, *obstáculo*, estado inicial  $\mathbf{s}_0$  e espaço de ações  $\mathcal{A}$ ;
- Inicializa Hiperparâmetros (bônus e penalidades, tamanho da rede e das imagens, número de *timesteps*, épocas e transições no *Buffer*, fator de desconto  $\gamma$ , taxa de aprendizado  $\alpha$ ) e  $\epsilon$ ;
- Inicializa estruturas de armazenamento de épocas e redes DQN;
- Inicializa rede DQN parametrizada  $Q_{\theta_0}$  aleatória e armazena em DQNBuffer(1);

for  $ep \leftarrow 1$  to  $MaxEpoch$  do

    inicializa estado:  $\mathbf{s} \leftarrow \mathbf{s}_0$ ;

    Preenche *Buffer* de Experiência com N transições segundo política  $\epsilon$ -Greedy e rede atual  $Q_{\theta_{ep}}$ ;

    Amostra *mini-batch* aleatório de tamanho  $N_{batch}$ ;

    for  $i \leftarrow 1$  to  $N_{batch}$  do

        Ler  $i$ -ésima transição:  $(\mathbf{s}, \mathbf{a}, r, \mathbf{s}', bool_{term})$ ;

        if  $bool_{term} == true$  then

$y = r$ ;

        else

$y = r + \gamma \max_{\mathbf{a}' \in \mathcal{A}} Q_{\theta_{ep}}(\mathbf{s}', \mathbf{a}')$ ;

        end

        Armazenar saída prevista  $q = Q(\mathbf{s}, \mathbf{a})$  e alvo  $y$ ;

    end

    Aplica Gradiente Descendente para minimizar função custo dada por

$\mathcal{L}(\theta) = \frac{1}{2} (Q_{\theta}(\mathbf{s}, \mathbf{a}) - y)^2$ , ou seja:

$$\theta_{ep+1} \leftarrow \theta_{ep} - \alpha \frac{1}{N_{batch}} \sum_{i=1}^{N_{batch}} \nabla_{\theta} \frac{1}{2} (Q_{\theta}(\mathbf{s}, \mathbf{a}) - y)^2;$$

    Armazena rede  $Q_{\theta_{ep+1}}$  em DQNBuffer(ep+1);

    Limpa *Buffer* de transições;

    Mostra trajetória *greedy* atual;

    Mostra valor médio de recompensas imediatas;

end











# Projeto Kuka rl: Resultados parciais

Resultados Parciais:  
Projetos de Teste

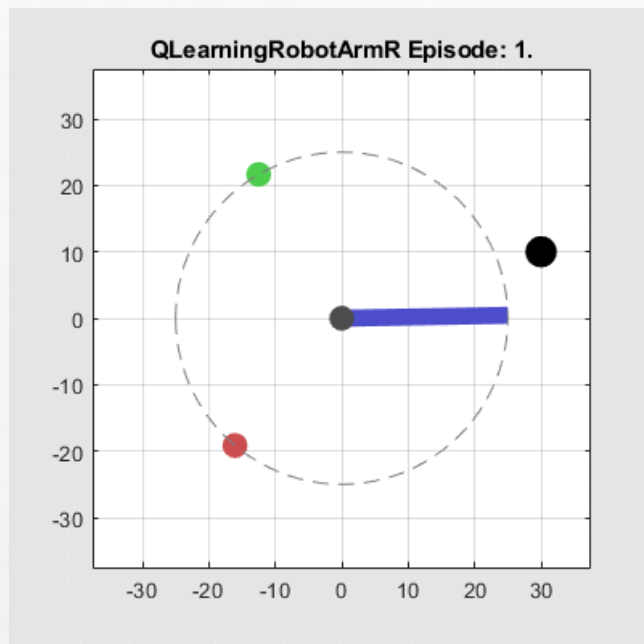
# Projeto Kuka rl: Resultados Parciais

## 1 Grau de Liberdade (R)

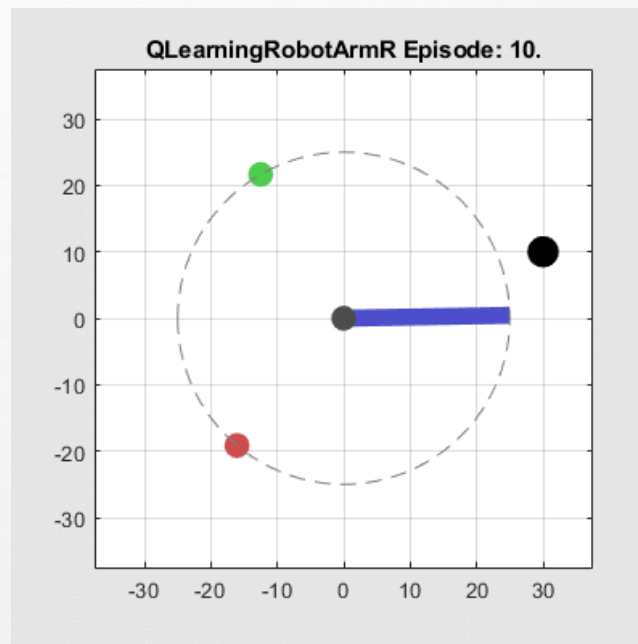
Projetos de Teste	REINFORCE	Q-Learning
1 Grau de Liberdade (R)		
2 Graus de Liberdade (RR)		
	DRL	DQN
6 Graus de Liberdade (6R), configuração fixa		
6 Graus de Liberdade (6R), configurações aleatórias		

Q-Learning: Trajetórias ao longo do treino com  $r_2$

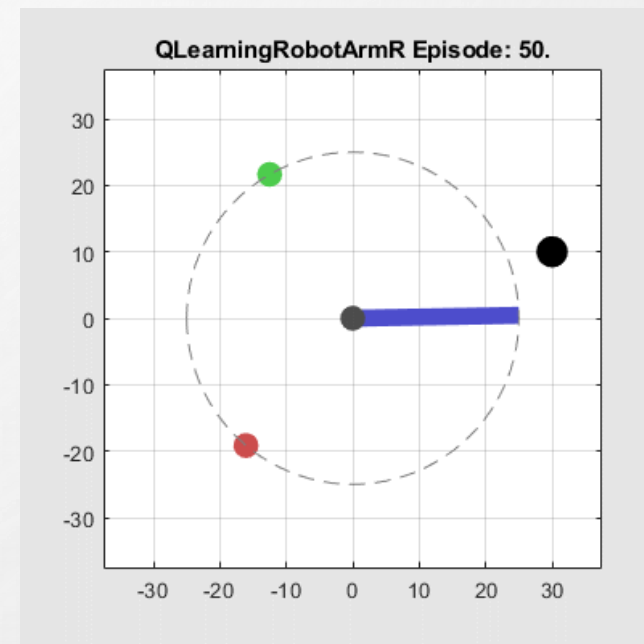
Época 1



Época 10











Época 50



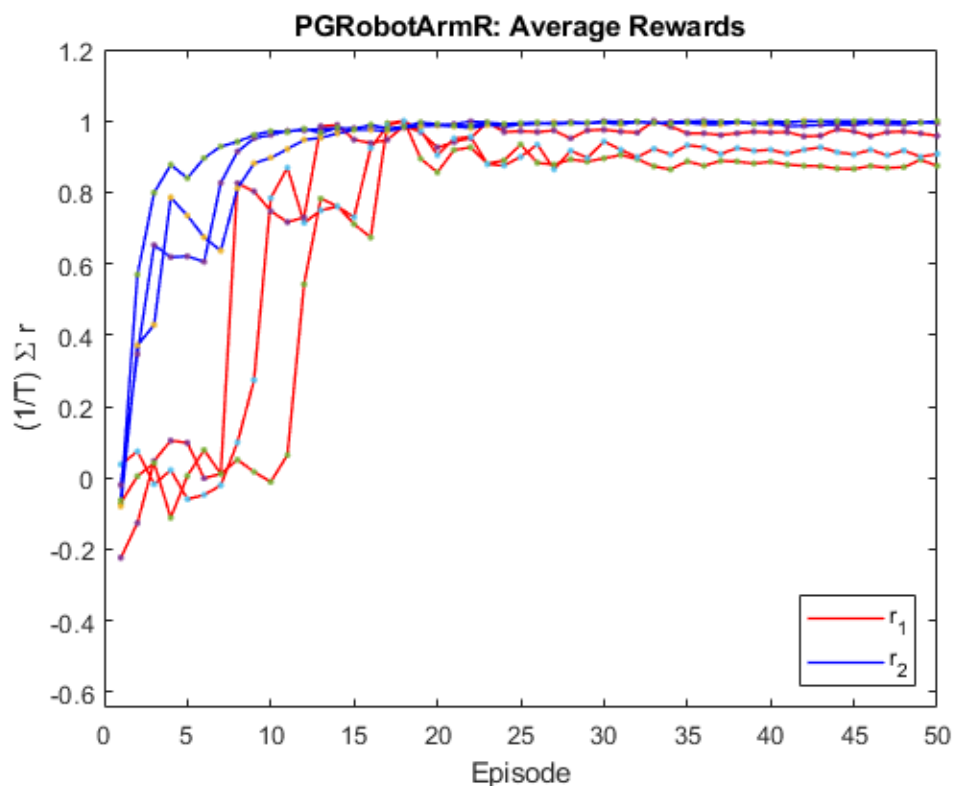
# Projeto Kuka rl: Resultados Parciais

## 1 Grau de Liberdade (R)

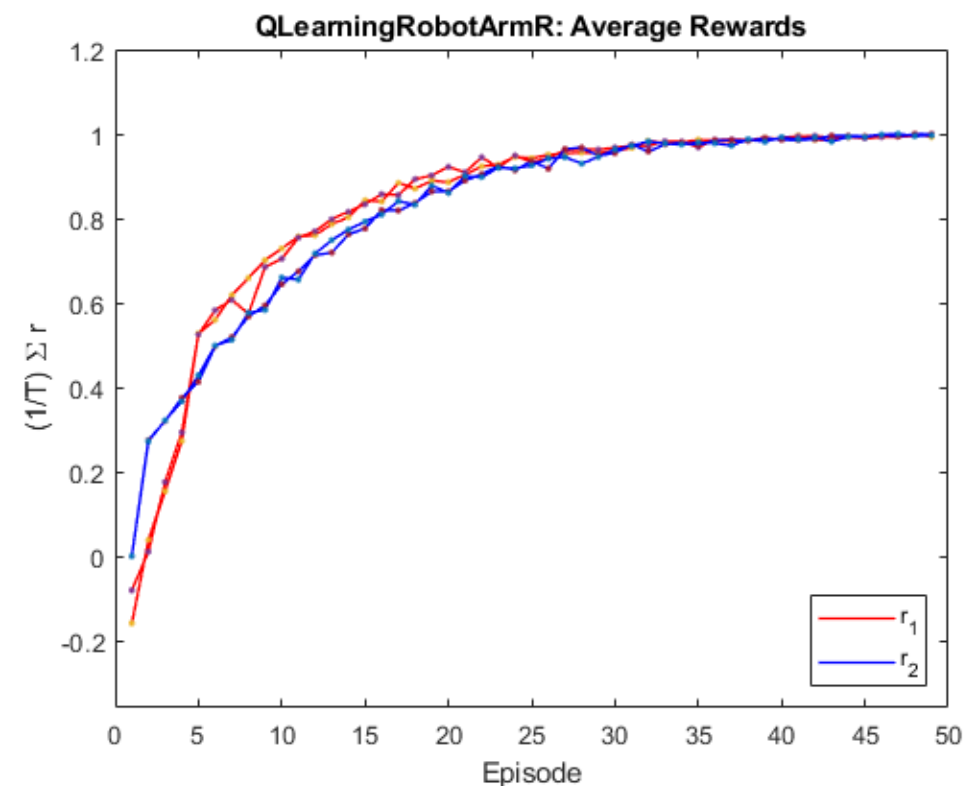
Projetos de Teste	REINFORCE	Q-Learning
1 Grau de Liberdade (R)		
2 Graus de Liberdade (RR)		
6 Graus de Liberdade (6R), configuração fixa		
6 Graus de Liberdade (6R), configurações aleatórias		

Comparação entre funções de recompensa  $r_1$  e  $r_2$

REINFORCE











Q-Learning



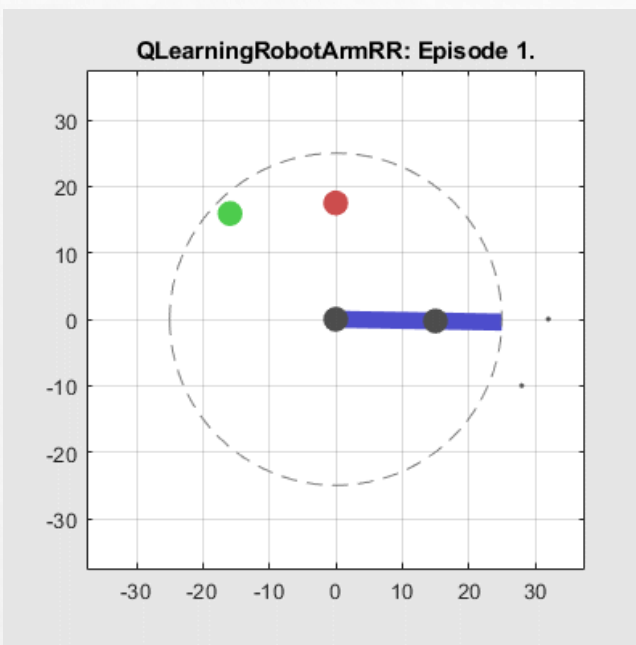
# Projeto Kuka rl: Resultados Parciais

## 2 Graus de Liberdade (RR)

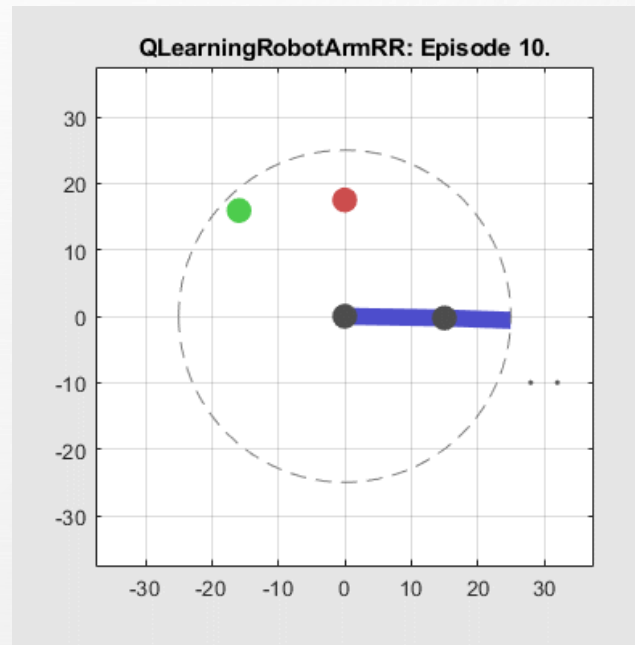
Projetos de Teste	REINFORCE	Q-Learning
1 Grau de Liberdade (R)		
2 Graus de Liberdade (RR)		
	DRL	DCQN
6 Graus de Liberdade (6R), configuração fixa		
6 Graus de Liberdade (6R), configurações aleatórias		

Q-Learning: Trajetórias ao longo do treino com  $r_3$

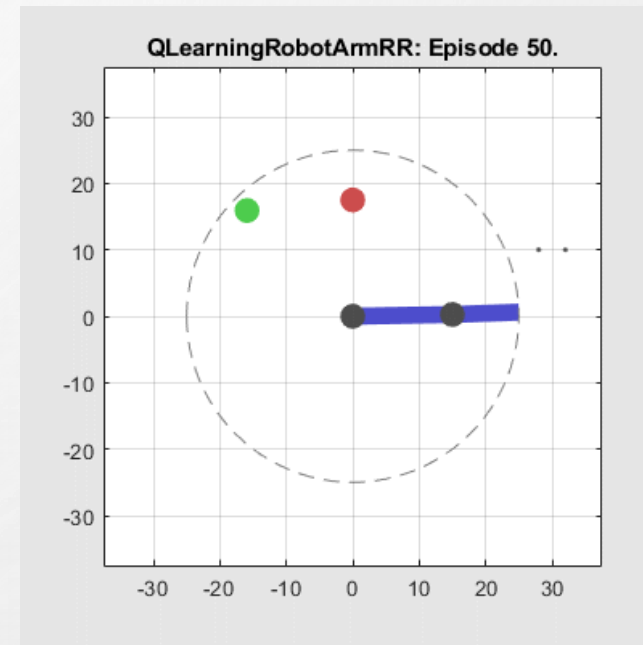
Época 1



Época 10











Época 50



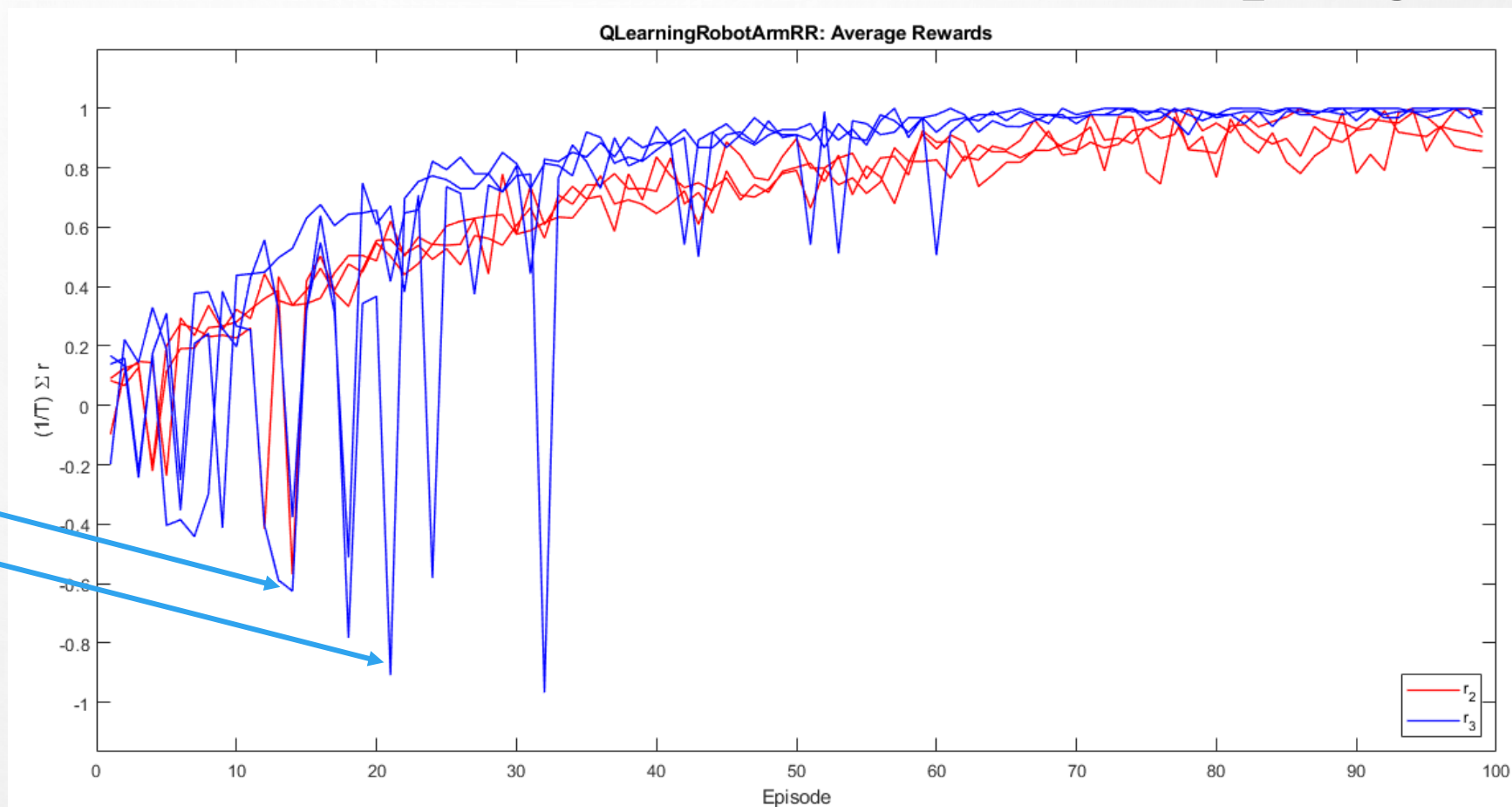
# Projeto Kuka rl: Resultados Parciais

## 2 Graus de Liberdade (RR)

Projetos de Teste	REINFORCE	Q-Learning
1 Grau de Liberdade (R)		
2 Graus de Liberdade (RR)		
	DRL	DCQN
6 Graus de Liberdade (6R), configuração fixa		
6 Graus de Liberdade (6R), configurações aleatórias		









Colisões  
com  
obstáculo

## Q-Learning: Comparação entre funções de recompensa $r_2$ e $r_3$

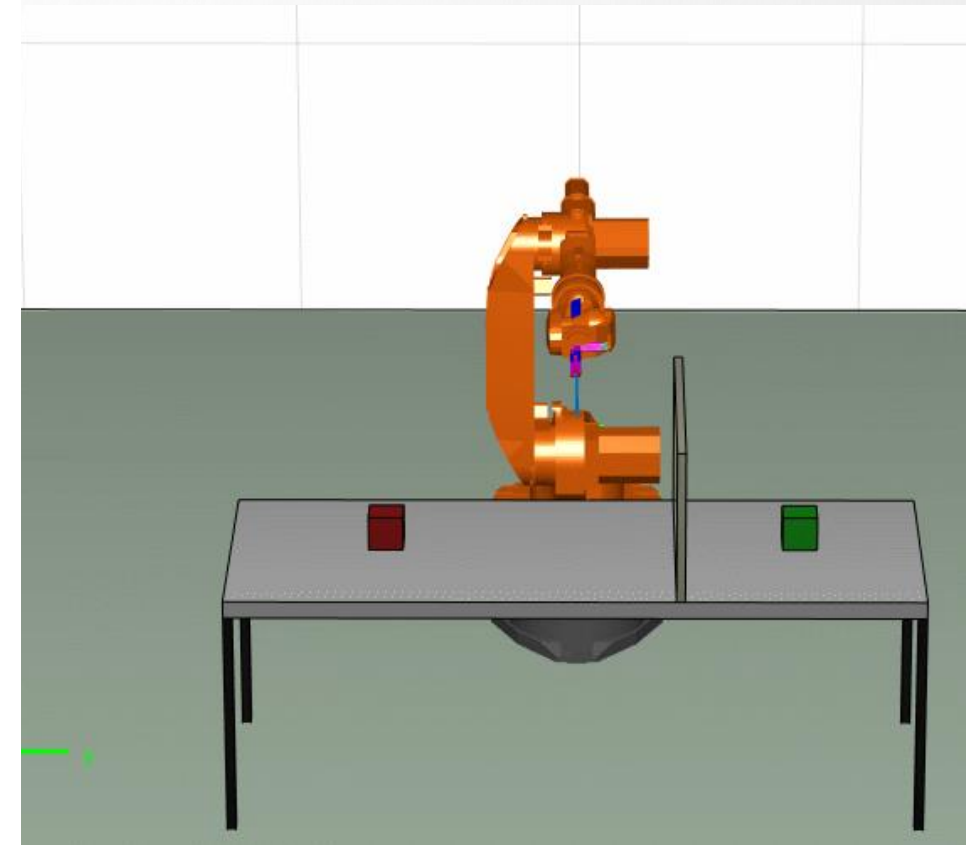
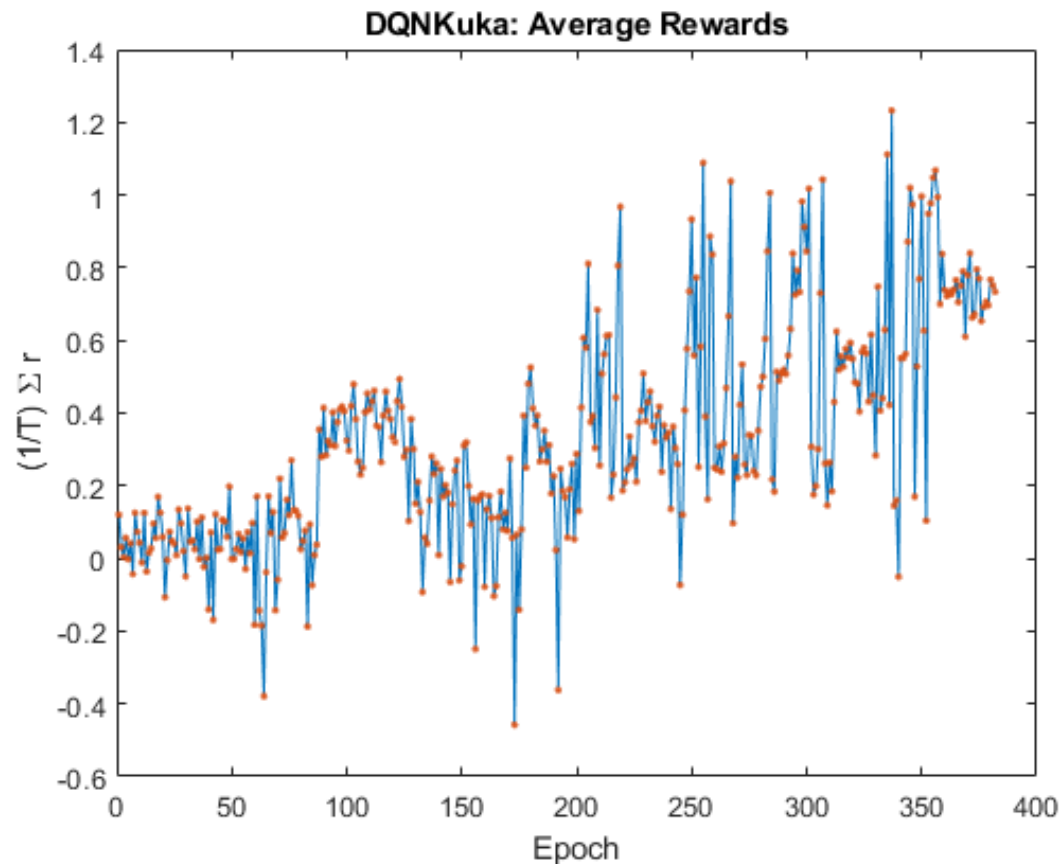


# Projeto Kuka rl: Resultados Parciais

## 6 Graus de Liberdade (6R)

Projetos de Teste	REINFORCE	Q-Learning
1 Grau de Liberdade (R)		
2 Graus de Liberdade (RR)		
	DRL	DQN
6 Graus de Liberdade (6R), configuração fixa		
6 Graus de Liberdade (6R), configurações aleatórias		









## DQN: Obstáculo Desconhecido





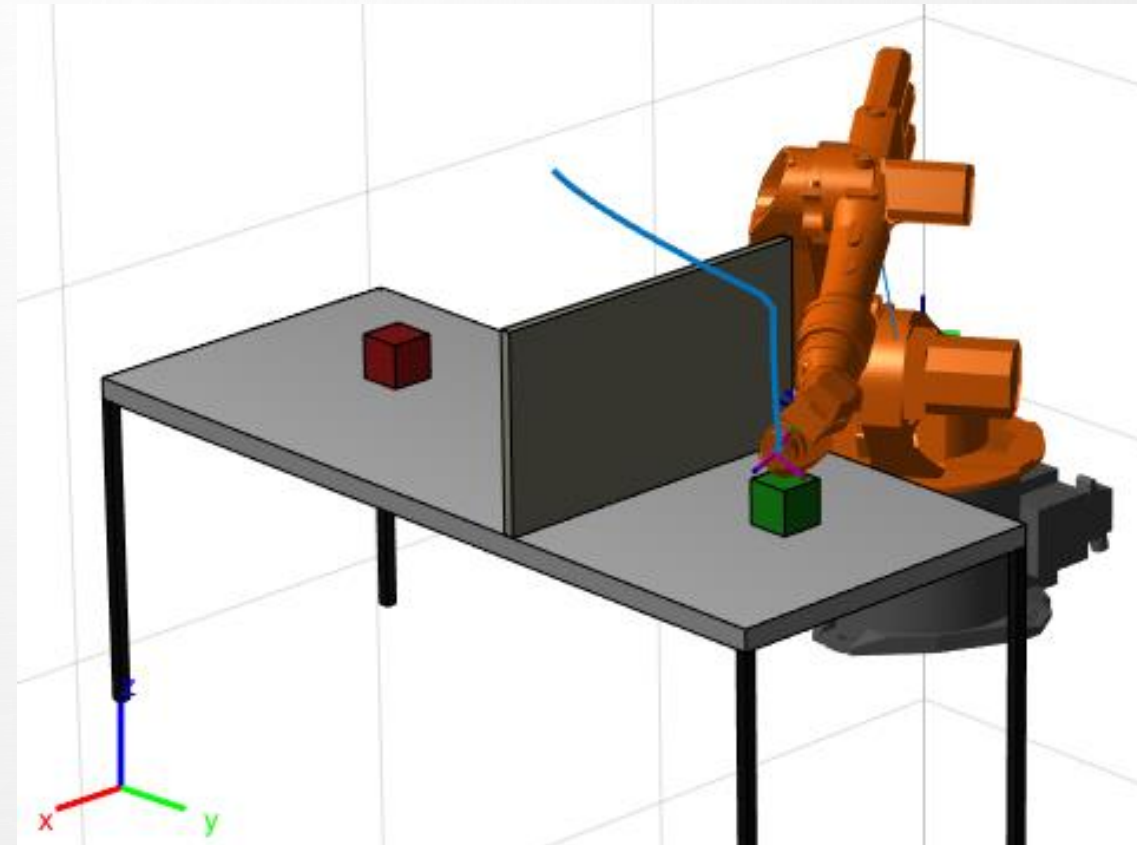
# Projeto Kuka rl: Detalhamento do projeto

## 6 Graus de Liberdade (6R)

Projetos de Teste	REINFORCE	Q-Learning
1 Grau de Liberdade (R)		
2 Graus de Liberdade (RR)		
	DRL	DQN
6 Graus de Liberdade (6R), configuração fixa		
6 Graus de Liberdade (6R), configurações aleatórias		









DQN: Obstáculo Desconhecido

Trajetória obtida por agente treinado



# Projeto Kuka rl: Detalhamento do projeto

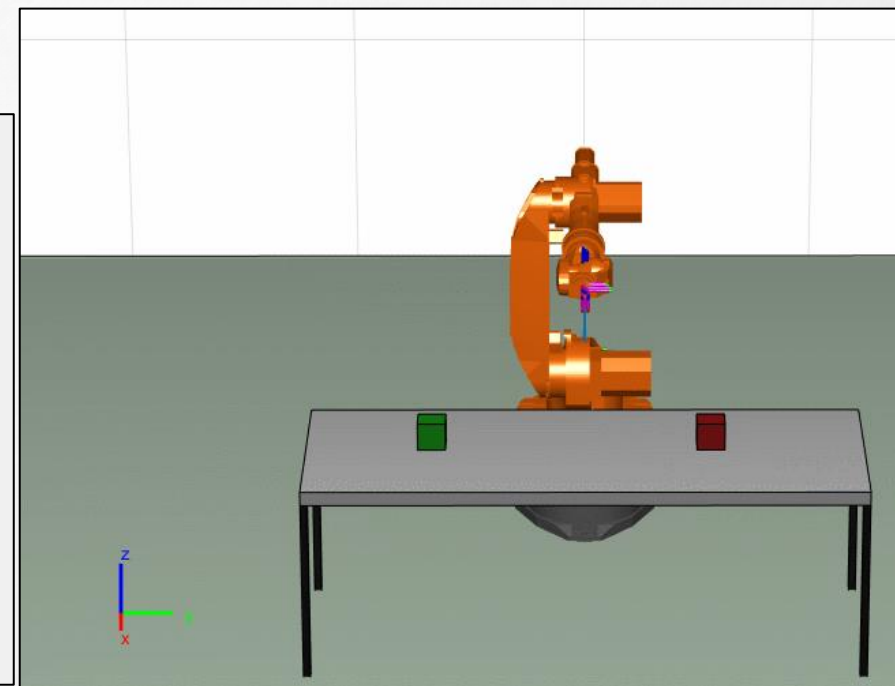
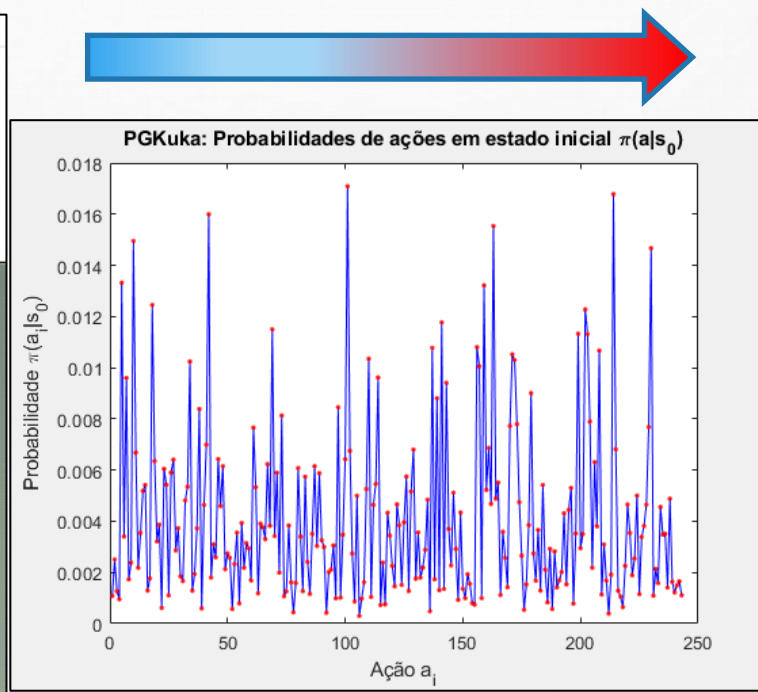
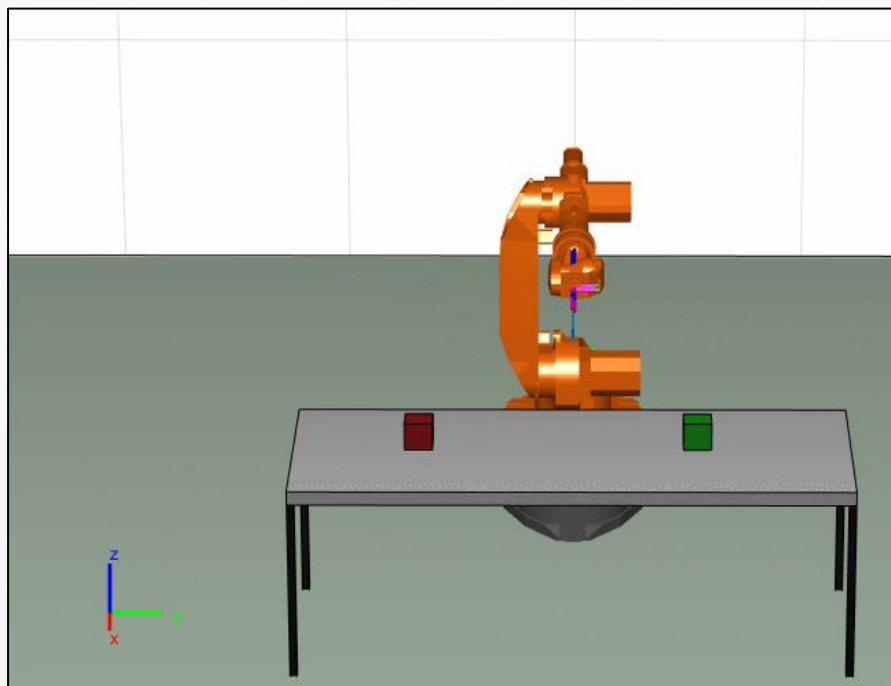
## 6 Graus de Liberdade (6R)

Projetos de Teste	REINFORCE	Q-Learning
1 Grau de Liberdade (R)		
2 Graus de Liberdade (RR)		
	<b>DRL</b>	<b>DQN</b>
6 Graus de Liberdade (6R), configuração fixa		
6 Graus de Liberdade (6R), configurações aleatórias		

## REINFORCE: Retreinamento de Agente

1º treino

2º treino



# Projeto Kuka rl: Detalhamento do projeto

## 6 Graus de Liberdade (6R)

Projetos de Teste	REINFORCE	Q-Learning
1 Grau de Liberdade (R)		
2 Graus de Liberdade (RR)		
6 Graus de Liberdade (6R), configuração fixa		
6 Graus de Liberdade (6R), configurações aleatórias		

## REINFORCE: Retreinamento de Agente

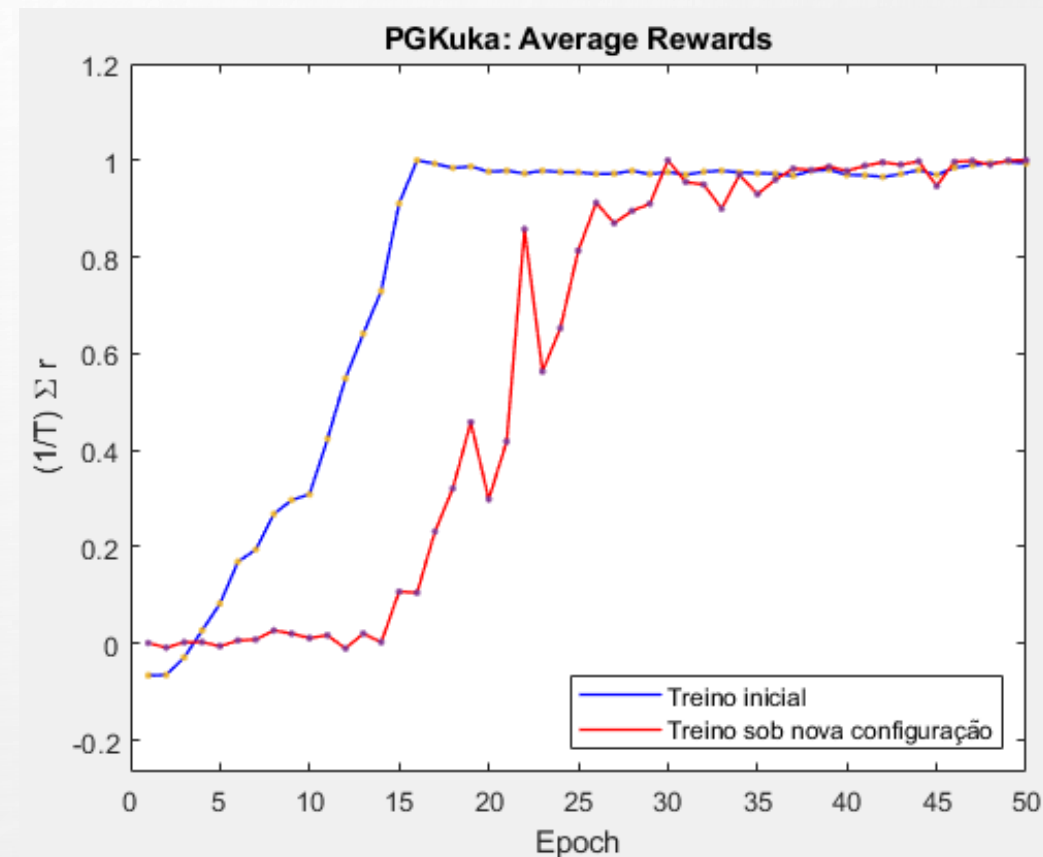
**Treino Inicial**

- Política inicial  $\pi_{\theta_0}$  aleatória
- Convergência mais rápida

**Retreinamento**









- Política inicial  $\pi_{\theta_0}$  sub-ótima
- Convergência mais lenta
- Perda de aprendizado

**Solução:** Treinar simultaneamente para diversas configurações

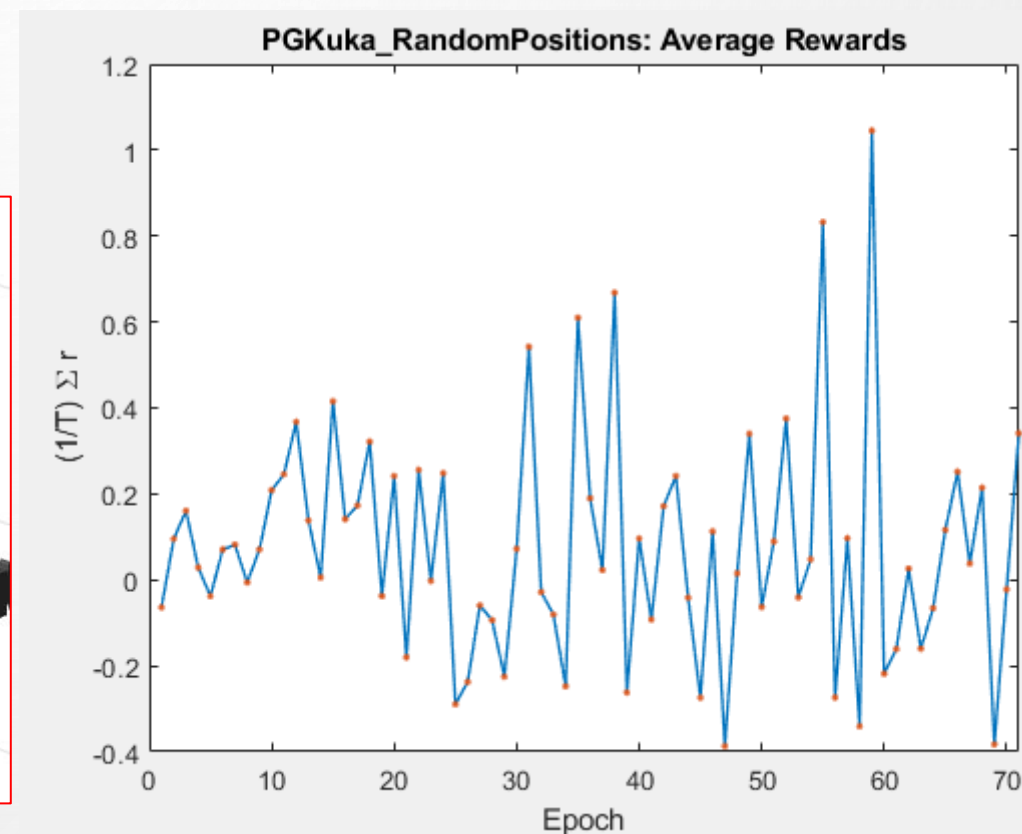
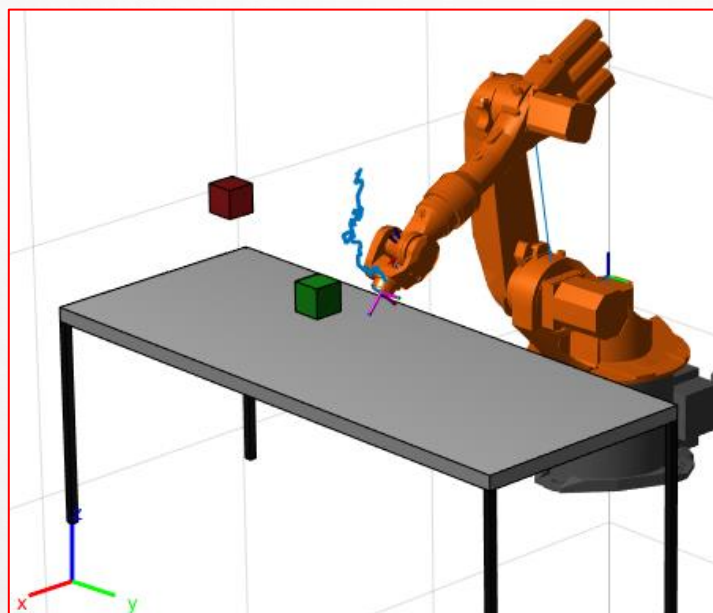
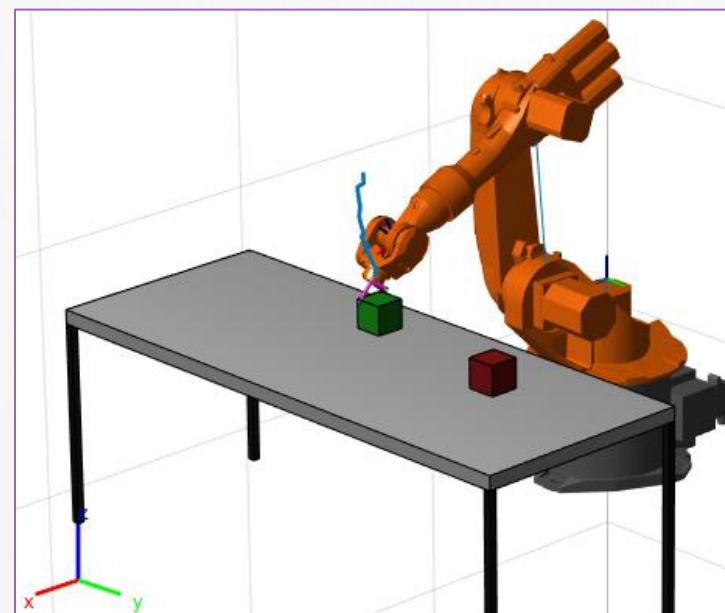


# Projeto Kuka rl: Detalhamento do projeto

## 6 Graus de Liberdade (6R) e Configurações Genéricas

Projetos de Teste	REINFORCE	Q-Learning
1 Grau de Liberdade (R)		
2 Graus de Liberdade (RR)		
	DRL	DQN
6 Graus de Liberdade (6R), configuração fixa		
6 Graus de Liberdade (6R), configurações aleatórias		









**REINFORCE: Não foi observada convergência**

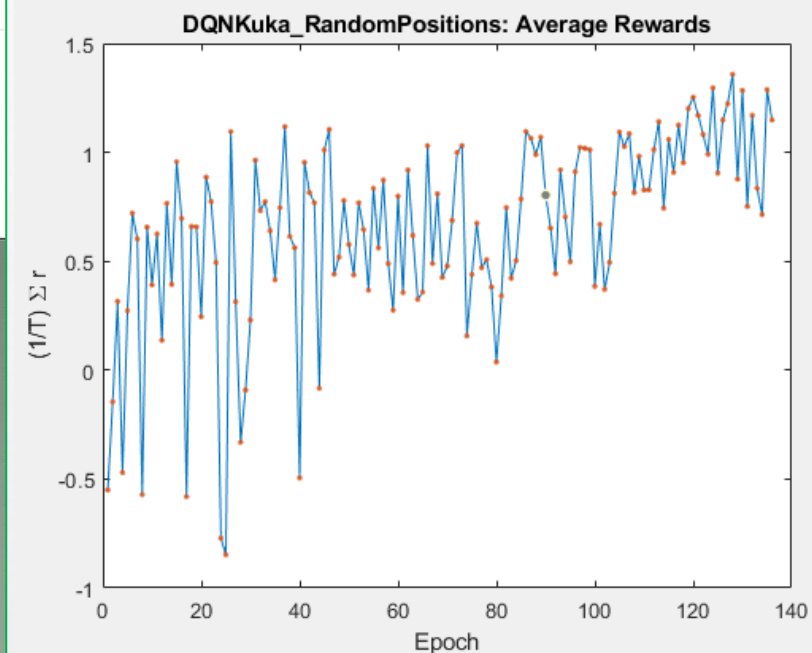
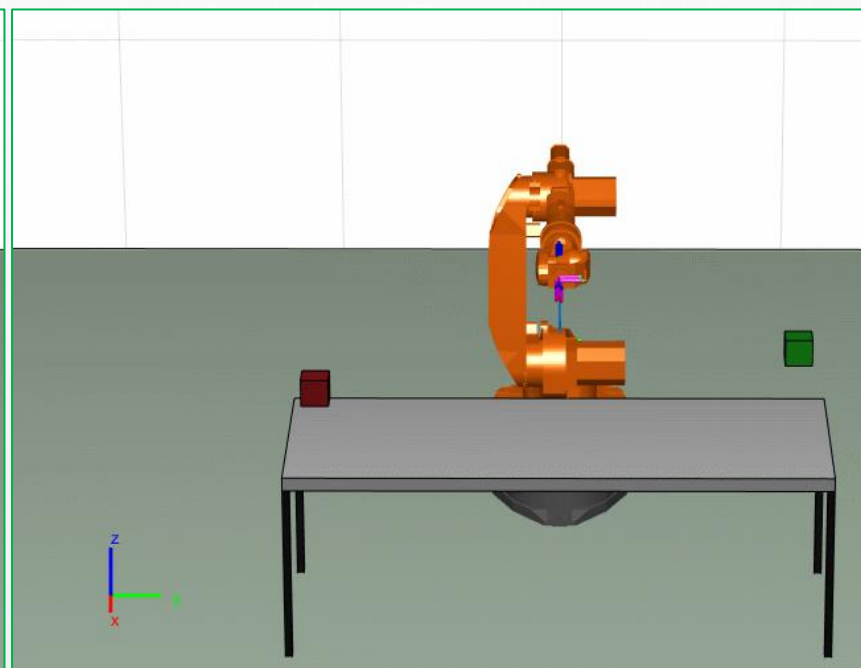
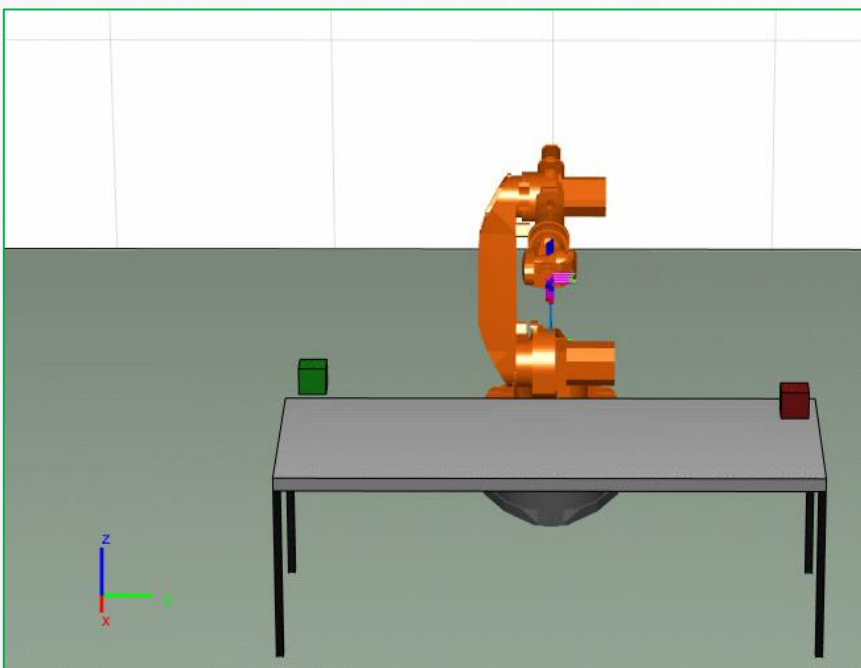


# Projeto Kuka rl: Detalhamento do projeto

## 6 Graus de Liberdade (6R) e Configurações Genéricas

### DQN: Convergência para configurações específicas

Projetos de Teste	REINFORCE	Q-Learning
1 Grau de Liberdade (R)		
2 Graus de Liberdade (RR)		
	DRL	DQN
6 Graus de Liberdade (6R), configuração fixa		
6 Graus de Liberdade (6R), configurações aleatórias		



# Projeto Kuka rl: Resultados parciais

Resultados Finais

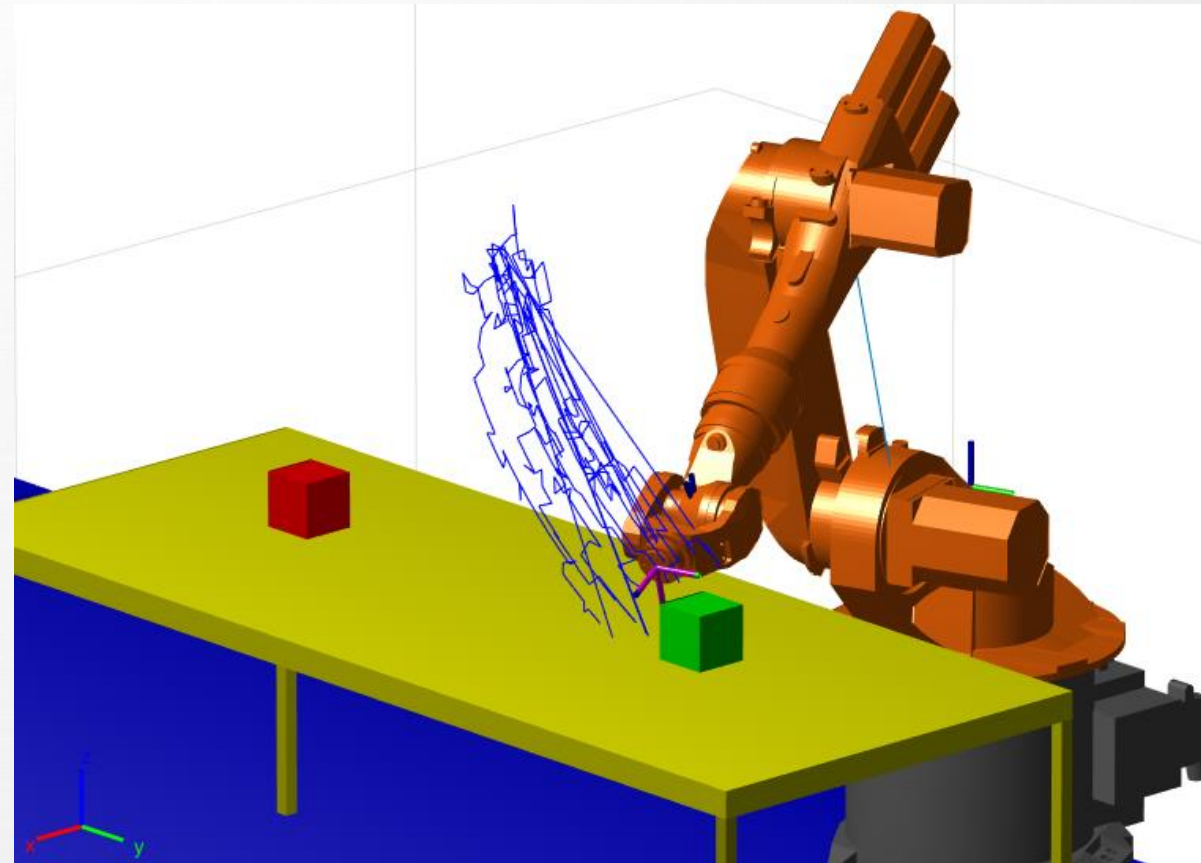
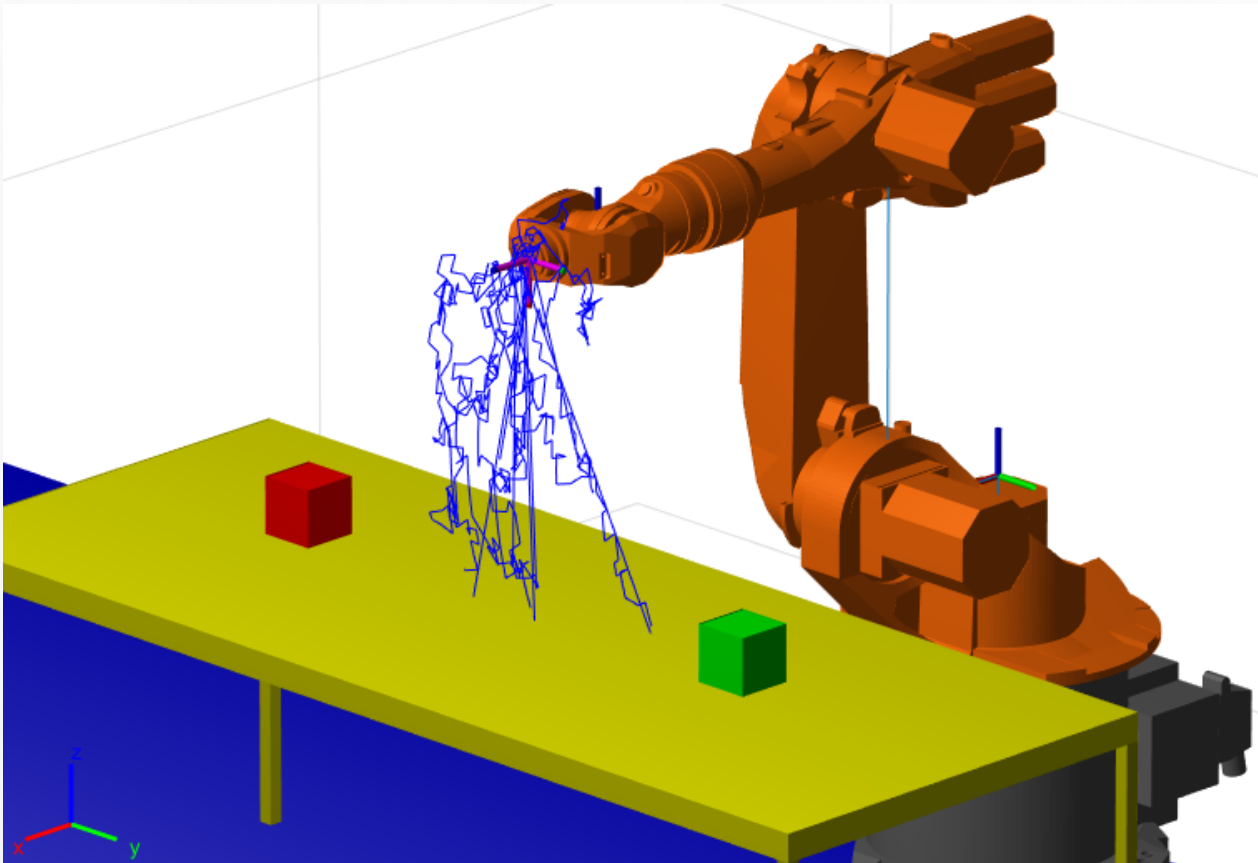


# Projeto Kuka rl: Detalhamento do projeto

## Exploração ao longo do treino

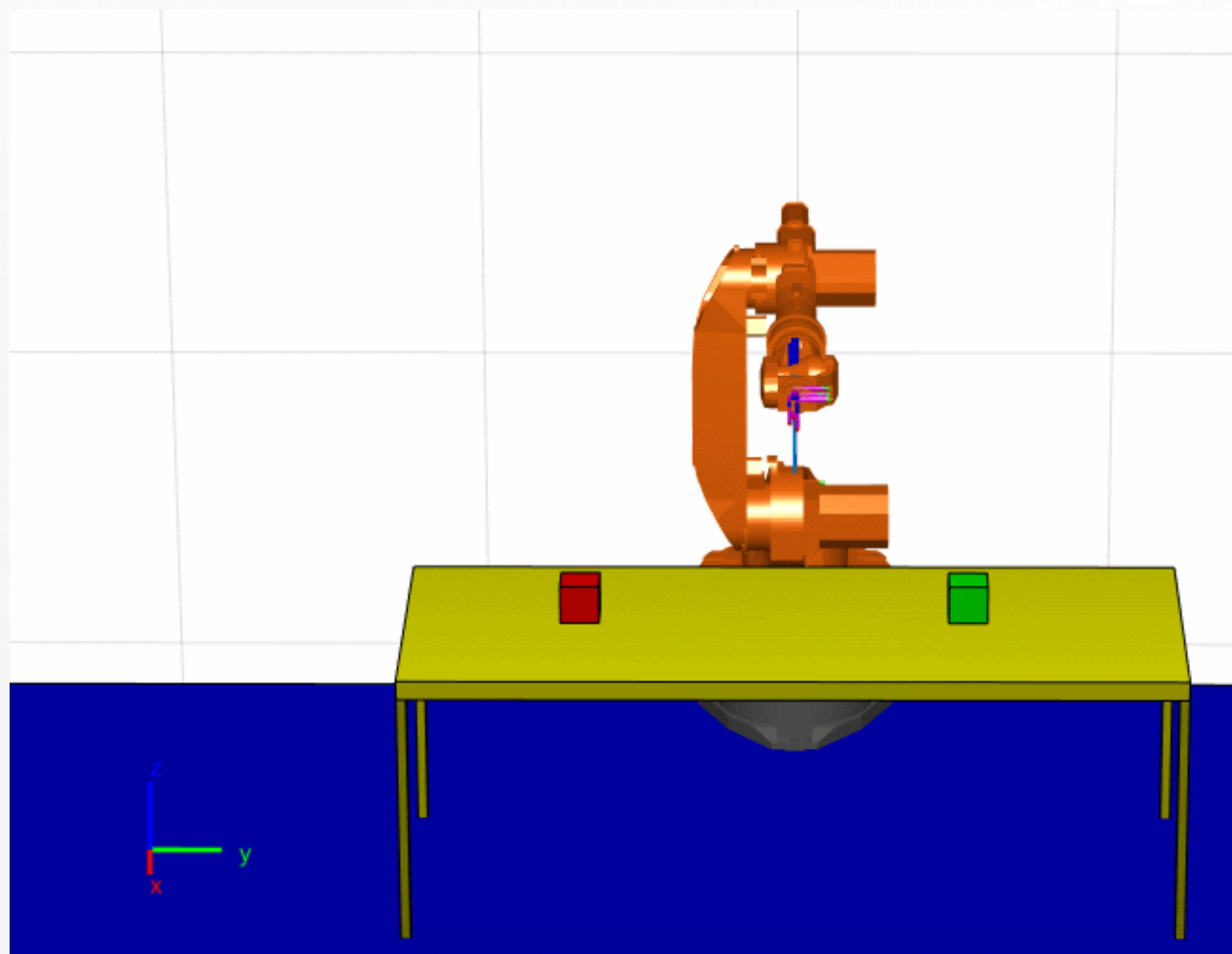
Época 1

Época 30

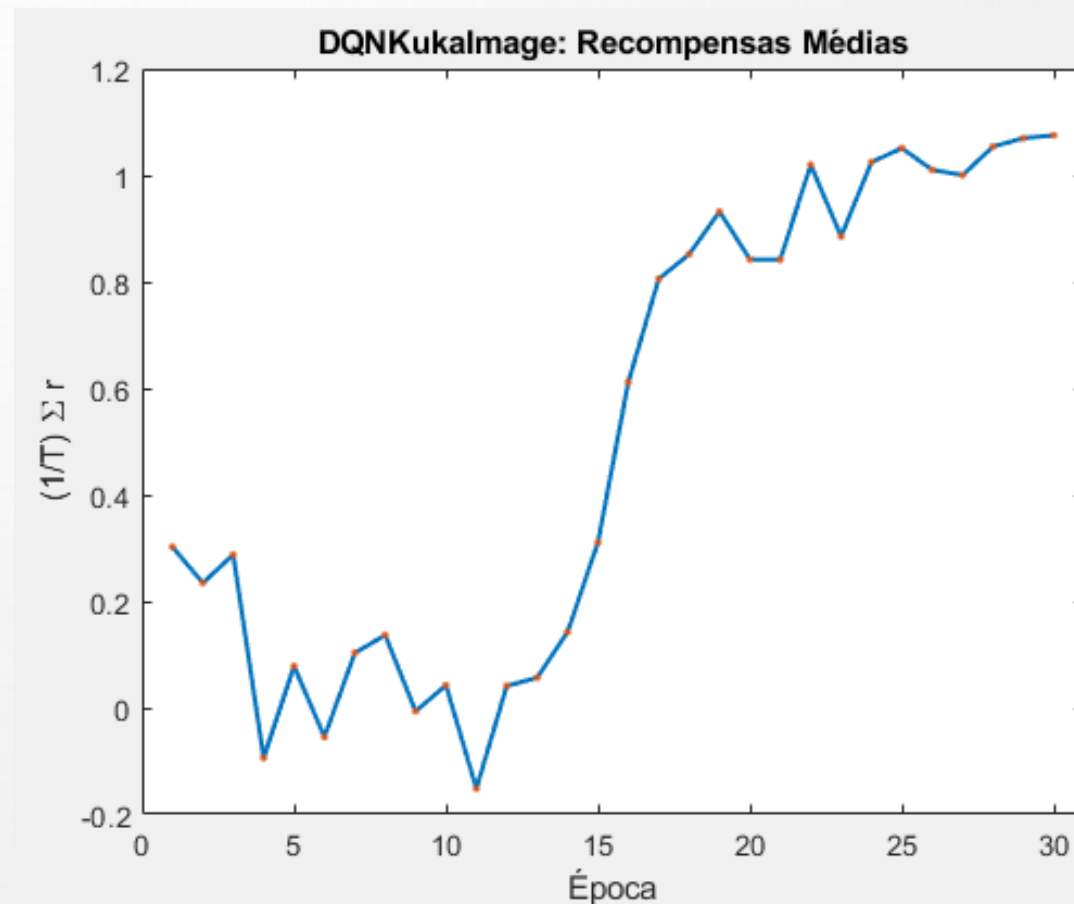


# Projeto Kuka rl: Detalhamento do projeto

## Trajetória Obtida



## Curva de Aprendizado



# Entregas

- A entrega do último trabalho e qualquer outro que ficou faltando é até domingo dia 16/05. Isso porque o fechamento é até o dia 24/05 e precisamos corrigir
- Se quiserem podem enviar novamente trabalhos já avaliados, mas é necessário nos avisar no Microsoft Teams (o Open LMS não nos avisa)
- Na próxima segunda feira vamos postar todos gabaritos dos exercícios
- Caso haja algum problema na entrega do trabalho final (até porque o tempo é menor), nos avise

Muito obrigado a todos pelo curso!

Esperamos que tenham gostado

Dúvidas e Sugestões?

Lucas Pereira Cotrim  
Marcos Menon José

lucas.cotrim@maua.br  
marcos.jose@maua.br