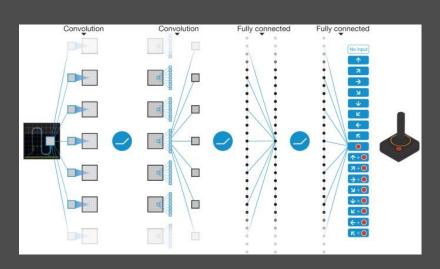
## Aprendizado por Reforço

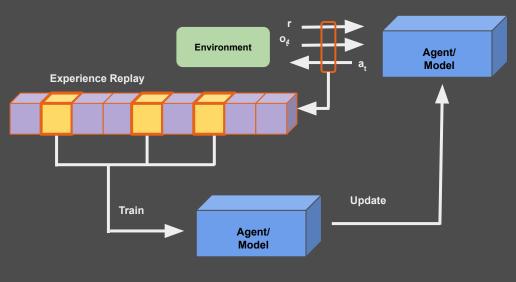
AULA - 7

**Tópicos Avançados** 

#### De onde viemos...

#### Deep Q-Network





$$L(\theta) = \left(r + \gamma \max_{a'} Q(s', a'; \phi) - Q(s, a; \theta)\right)^{2}$$

#### De onde viemos...

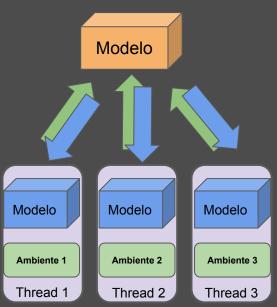
#### • A3C + PPO

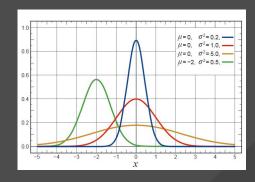
Method	Training Time	Mean	Median
DQN	8 days on GPU	121.9%	47.5%
Gorila	4 days, 100 machines	215.2%	71.3%
D-DQN	8 days on GPU	332.9%	110.9%
Dueling D-DQN	8 days on GPU	343.8%	117.1%
Prioritized DQN	8 days on GPU	463.6%	127.6%
A3C, FF	1 day on CPU	344.1%	68.2%
A3C, FF	4 days on CPU	496.8%	116.6%
A3C, LSTM	4 days on CPU	623.0%	112.6%

*Table 1.* Mean and median human-normalized scores on 57 Atari games using the human starts evaluation metric. Supplementary Table SS3 shows the raw scores for all games.





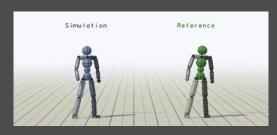




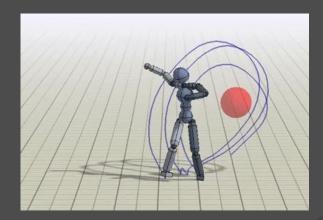
## Hoje

- Como aprender comportamentos extremamente complexos?
- Posso aprender várias tarefas ao mesmo tempo?
- O que acontece se eu tiver vários agentes aprendendo?
- Posso imitar comportamento com generalização?
- E se eu quiser aplicar meu algoritmo no mundo real?









# Inverse Reinforcement Learning

#### Como IRL funciona

- Aprender uma função de recompensa tendo uma política
- Funciona com trajetórias de experts
  - Inicializar Política(theta) e estimativa de Recompensa(phi)
  - Treinar Política(theta) usando Recompensas(phi)
  - Calcular erro entre Política(theta) treinada e Expert
  - Atualizar estimador de Recompensa(phi) com o erro
- Existem variações para o cálculo do erro

## Meta-Learning

#### Meta

- Aprendendo a Aprender
- Objetivo é conseguir aprender em tarefas novas de forma rápida e eficiente
- Algumas tarefas são usadas para treino, enquanto outras são usadas em teste
- É preciso que haja certa similaridade entre as tarefas\*

## O que se aprende?

#### Gradientes

 Dada uma tarefa, qual a melhor forma de mudar os pesos para que, com menos gradientes, eu chegue na solução?

#### Função de Perda

 Dada uma tarefa, qual função de perda melhor indica o rumo que meus pesos devem seguir?

#### Hiperparâmetros

Dada uma tarefa, quais hiperparâmetros melhor aceleram o aprendizado?

#### Estratégias de Exploração

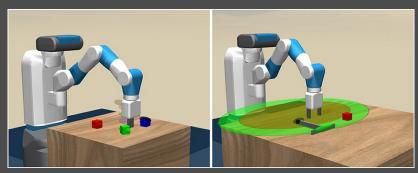
o Dada uma tarefa, como explorar o espaço de busca de forma mais eficiente?

# Transfer Learning

#### Transferência de Conhecimento

- Transferir conhecimento de uma tarefa para outra
- Tarefas similares = Soluções similares
- Pegando um modelo treinado em uma tarefa, ele pode ter facilidade de aprender tarefas similares
- Buscando generalização entre tarefas diferentes

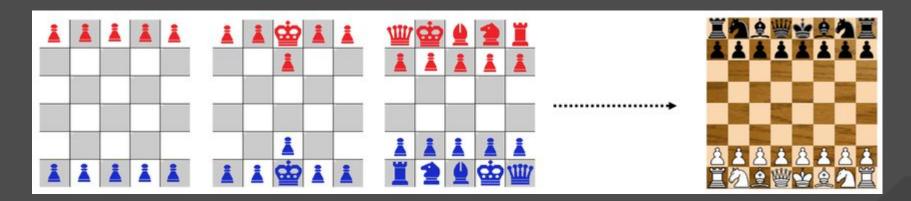




# Curriculum Learning

## Ideia Principal

- Quebrar tarefas complexas em subtarefas de dificuldade ascendente
- Transferência de conhecimento é comprovada



#### Como desenhar um currículo?

- Tarefas começam simples, e ganham dificuldade
- Tarefas subsequentes devem aumentar a tarefa anterior, mas ainda manter suas características.
- Caso as tarefas se diferenciem muito nas competências necessárias para resolvê-las, pode haver esquecimento.
- É preciso saber quando aumentar a dificuldade, não é bom ter overfit em uma subtarefa
- Tarefas devem convergir, no fim, para a tarefa completa/final





Ficar de Pé



Andar



Correr/Pular



Aumentar inimigos, adicionar unidades diferentes

#### Vale a pena?

- Sim, se seu objetivo for complexo e a situação permitir
- Diminui bastante o tempo de treino

- Provavelmente será específico da situação
- Mais tarefas a serem projetadas

#### Existe currículo automático?

- Alguns trabalhos buscam o chamado Auto-Curriculum
- Formas generalistas de quebrar problemas e treinar agentes mais rápido

## Sim-to-Real

## Aplicando Soluções no Mundo Real

- Algoritmos treinados no mundo real são ok
- Algoritmos treinados em simulação precisam de adaptação
- Simulações não descrevem corretamente o mundo real

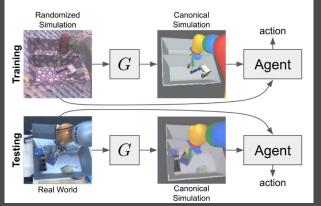
- Principal solução: Generalização
- Presumindo: um algoritmo que consegue agir em ambientes diferentes, conseguirá agir no mundo real

#### Adicionando Ruído

Domain Randomization (imagens)

- Modelo Generativo (imagem)
  - o Faz imagens reais parecer vindas do simulador





Ruído em medidas (dados do simulador)

## Model Based

#### Prevendo o próximo Estado e Recompensa

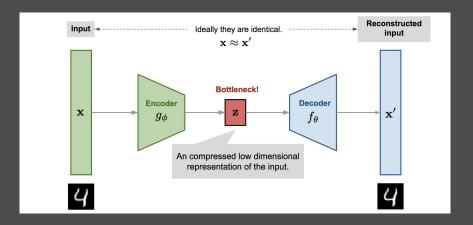
- Aprendendo as dinâmicas do ambiente
- Permite planejamento de ações

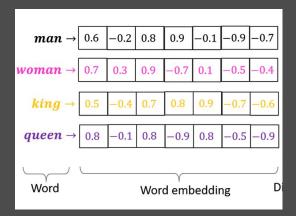
- Auxílio na Exploração
  - Curiosidade: Se eu sei com certeza qual o próximo estado, melhor ir para outro lugar
- Aprender com Experts presentes no ambiente
- Planejamento de ações (olhando para o futuro)

# Representation Learning

## Aprendendo Representações

- Aprender uma representação diferente, simplificada, menor, ou comprimida, de algo
- Ex: Autoencoders





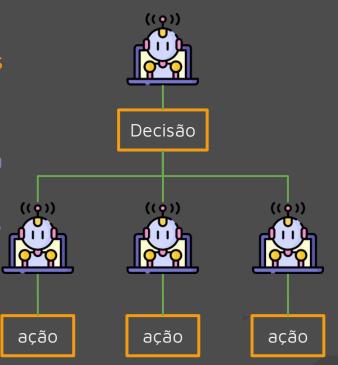
## Representações em Reforço

- Representações de Estados
  - Melhores representações de estados podem levar a um aprendizado mais rápido
  - o Como aprender representações de estados através de tarefas diferentes?
- É possível ter uma representação de uma tarefa? De um Objetivo?

# Hierarchical Reinforcement Learning

## Hierárquico

- Decisões são feitas em vários níveis
- Níveis baixos dependem de decisões em níveis mais altos
- Um agente que escolhe qual (outro) agente irá agir dependendo da situação
- Um agente que observa a situação completa e envia estratégias para outros agentes que observam de forma parcial

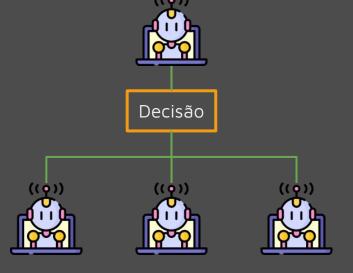


#### Treinando de forma Hierárquica

É difícil propagar o gradiente através de muitos níveis

A recompensa geralmente depende das ações dos níveis

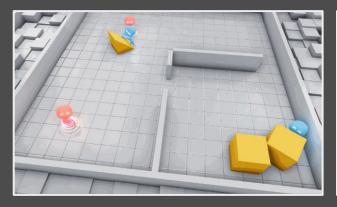
mais baixos



# Multi-Agent

## E se eu tiver vários agentes?

- Agentes interpretam uns aos outros como parte do ambiente
- Há vários tipos de ambientes multiagente
- Tarefas cooperativas ou competitivas
- Agentes simétricos ou assimétricos
- Atingem comportamentos coordenados







#### Controlando agentes

- Centralizado (não é bem multi-agente)
  - Uma rede controla todos os agentes ao mesmo tempo
  - Maior coordenação (conhecimento centralizado)
- Agentes que compartilham redes
  - Agentes são controlados separadamente por uma uma única rede (as redes de cada agente tem os mesmos pesos)
  - Compartilham experiências (mais dados)
- Agentes que aprendem separadamente
  - o Cada agente tem sua própria rede e aprende de forma separada
  - Soluções mais específicas (útil para tipos de agentes diferentes)

# Self-Play

## Jogando contra si mesmo

- Um modelo pode jogar contra si mesmo em ambientes com certas características
  - Ambientes Simétricos (capacidades iguais\* dos dois lados)
  - Ambientes Competitivos
- Por Que?
- O ambiente não possui um adversário padrão
  - Ex: Xadrez





SPOAL FOR INCIDENCE

FINANCE OF THE PROPERTY O

AlphaStar

OpenAl Five

#### Estratégias

 Jogar contra outro modelo aprendendo (não é bem self-play)

Jogar contra a última versão do modelo

Jogar contra versões passadas do modelo

# Jogar contra outro modelo aprendendo (não é bem self-play)

- Interessante observar as dinâmicas de aprendizado
- Quando um aprende a vencer, ele domina completamente o adversário

## Jogar contra a última versão do modelo

- Pegar experiências dos dois lados (2x mais dados por ambiente)
- Teoricamente, o adversário nunca é bom demais, ou ruim demais (Auto-Curriculum)
- Catastrophic Forgetting: O agente esquece estratégias que não usa mais
- Moving Target: O adversário/objetivo muda constantemente quando o modelo é atualizado

## Jogar contra versões passadas do modelo

- Mais estabilidade no treino
- Mantém Auto-Curriculum
- Eventualmente o modelo ficará melhor que o adversário e ele precisará ser atualizado

- Pode-se usar uma distribuição de versões anteriores (onde adversários mais recentes são mais prováveis)
- Dependendo critério de atualização do adversário, não haverá overfitting

## Sua vez...