# Reinforcement Learning – Notas de Aula

Sumário

[Reinforcement Learning – Notas de Aula 1](#_Toc58163876)

[Unidade 01 - Conceitos de aprendizagem por reforço 3](#_Toc58163877)

[Reinforcement Learning 3](#_Toc58163878)

[Definições de Aprendizagem por Reforço 3](#_Toc58163879)

[Elementos para uma Aprendizagem por Reforço 5](#_Toc58163880)

[Aplicações de Aprendizagem por Reforço 6](#_Toc58163881)

[Exemplos no Mundo Real 7](#_Toc58163882)

[Aprendizagem por Reforço Seguro 7](#_Toc58163883)

[Processo de Exploração 8](#_Toc58163884)

[Aprendizagem por Reforço Inversa 9](#_Toc58163885)

[Aprendizagem por Reforço para Espaços Finitos 11](#_Toc58163886)

[Aprendizagem por Reforço Inverso de Trajetórias Amostradas 11](#_Toc58163887)

[Aprendizagem da Aprendizagem: Aprender com um Especialista 12](#_Toc58163888)

[Resumo da Unidade 01 13](#_Toc58163889)

[Unidade 02 - Processos de decisão de Markov 14](#_Toc58163890)

[O Ambiente - Processos de Decisão de Markov 14](#_Toc58163891)

[Exemplo Cadeia de Markov 15](#_Toc58163892)

[Definição de um MDP 16](#_Toc58163893)

[Política e Função Valor de um MDP 16](#_Toc58163894)

[Algoritmos para Solução de MDPs 17](#_Toc58163895)

[Melhoria de Política 18](#_Toc58163896)

[O Algoritmo de Iteração de Valor 19](#_Toc58163897)

[Contração em um Espaço Banach 20](#_Toc58163898)

[Exemplo do Super Mario 20](#_Toc58163899)

[Processos de Decisão de Markov Parcialmente Observáveis 23](#_Toc58163900)

[Dinâmica de Sistemas Modelados como POMDPs 24](#_Toc58163901)

[Estado de Informação 24](#_Toc58163902)

[Unidade 03 - Aprendizagem por reforço passiva e ativa 24](#_Toc58163903)

[Aprendizagem por Reforço Passiva 24](#_Toc58163904)

[Exemplo de Aprendizagem por Reforço Passiva 26](#_Toc58163905)

[Programação Dinâmica Adaptativa 27](#_Toc58163906)

[Aprendizagem por Diferença Temporal 27](#_Toc58163907)

[Comparativo entre as Abordagens PDA e DT 28](#_Toc58163908)

[Aprendizagem por Reforço Ativa 29](#_Toc58163909)

[Exploração 30](#_Toc58163910)

[Aprendizagem de uma Função de Ação-valor 32](#_Toc58163911)

[Unidade 04 - Q-Learning 33](#_Toc58163912)

[Introdução à Aprendizagem Q 33](#_Toc58163913)

[Aprendizagem Q 34](#_Toc58163914)

[Função Q 34](#_Toc58163915)

[Aprendizagem Q Ótima 34](#_Toc58163916)

[Aprendizagem Baseada em Política 34](#_Toc58163917)

[Algoritmo Q-Learning 34](#_Toc58163918)

[Detalhes do Algoritmo Q-Learning 35](#_Toc58163919)

[Q-Learning 36](#_Toc58163920)

[Aplicações Q-Learning 36](#_Toc58163921)

# Unidade 01 - Conceitos de aprendizagem por reforço

Essa unidade tem como objetivo:

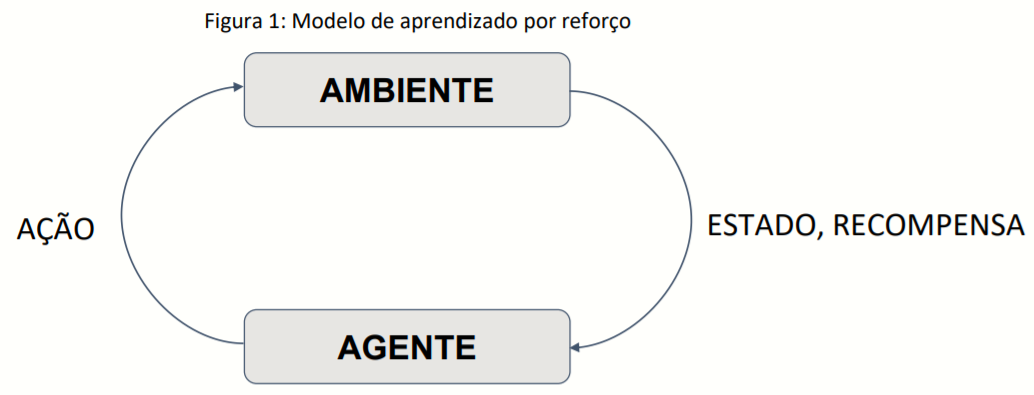
* Apresentar os principais conceitos de reinforcement learning;
* Exemplificar as diversas aplicações da aprendizagem por reforço;
* Conceituar aprendizagem por reforço seguro;
* Descrever aprendizagem por reforço inversa.

## Reinforcement Learning

* Aprendizado por reforço é um dos tipos de aprendizado em inteligência artificial, assim como aprendizado supervisionado, aprendizado não-supervisionado e aprendizado semi-supervisionado.
  + **Aprendizado supervisionado:** rótulos e classes são fornecidos a priori e o objetivo é classificar um novo padrão de entrada, o qual o rótulo não é conhecido, com base nos objetivos já rotulados. É uma metodologia que dada um conjunto de elementos já rotulados, ela será capaz de prever os rótulos de novas instâncias.
  + **Aprendizado não-supervisionado:** não há conhecimento sobre a classificação dos dados. O conhecimento que se tem é apenas sobre os atributos de entrada.
  + **Aprendizado semi-supervisionado:** É um meio termo entre o aprendizado supervisionado e o aprendizado não-supervisionado.
  + **Aprendizado por reforço:** Aprendizado é realizado com base nas observações do ambiente no qual o agente está inserido. O agente aprende a partir da interação do ambiente no qual ele está inserido. Não possui dados rotulados a priori. Agente aprende com base no ambiente e nas mudanças do ambiente. As ações que o agente executa geram recompensas e o ambiente vai mudando de estado.
    - **Exemplo:** Treinar a obediência de um cão. Se o cachorro obedecer, ele recebe uma bolinha ou ração (recompensas, reforço positivo). Se o cachorro não obedecer, ele recebe uma bronca e não ganha ração (sem recompensa, reforço negativo).
* Aprendizado por reforço é uma abordagem computacional de aprendizado a partir de uma ação e as recompensas recebidas.
  + **Exemplo:** Ser humano aprende sobre os perigos do fogo com base na interação. Se ficar distante do fogo pode iluminar um ambiente e se aquecer (reforço positivo). Se ficar muito perto pode ser queimar (reforço negativo).
* Aprendizado por reforço trabalho com ações de **causa e efeito**.
* O objetivo principal é **maximizar as recompensas** nas interações realizadas no ambiente.

## Definições de Aprendizagem por Reforço

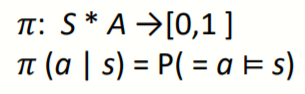
* Alguns conceitos de aprendizado por reforço
  + **Agente:** um agente executa ações.
    - Exemplo: o Super Mário percorrendo caminhos em um videogame.
  + **Meio ambiente:** é o mundo através do qual o agente se move.
  + **Ação:** é o conjunto de todas as atividades possíveis que o agente pode fazer.
    - **Exemplo:** No mercado financeiro as ações poderiam ser COMPRAR, VENDER, MANTER, a partir das oscilações nas cotações.
    - **Exemplo:** Para um drone as ações poderiam ser subir, descer, acelecer, dive (mergulho), seguir objeto.
  + **Estado:** um estado é uma situação concreta e imediata em que o agente se encontra, ou seja, um lugar e momento específico.
  + **Recompensa:** é o feedback pelo qual medimos o sucesso ou fracasso das ações de um agente. As recompensas podem ser imedatas ou atrasadas e efetivamente avaliam o desempenho do agente.
  + **Fator de desconto:** é multiplicado por recompensas futuras, conforme descobertas pelo agente, a fim de amortecer o efeito dessas recompensas na escolha da ação do agente.
    - As ações do agente devem prever o melhor resultado global possível, e não apenas o melhor resultado imediato. Logo, ações com bons resultados imediatos, mas resultados de longo prazo ruins podem receber recompensas menores, por exemplo.
  + **Política:** a política é a estratégia que o agente emprega para determinar a próxima ação com base no estado atual.
  + **Valor:** é o retorno esperado de longo prazo com desconto, em oposição à recompensa de curto prazo (definido como o retorno esperado a longo prazo do estado atual sob a política do agente).
    - Diferentemente da recompensa, o valor é a soma de todas as recompensas que podem anteceder um estado. É um valor a longo prazo, enquanto a recompensa é um prazer imediato.
    - **Exemplo:** Em um jantar comer uma salada, aos olhos do valor, seria contribuir para uma vida longa e saudável. Enquanto que a recompensa seria comer um hamburguer, que é mais saboroso e prazeroso no curto prazo, porém estaria comprometendo a saúde no futuro.



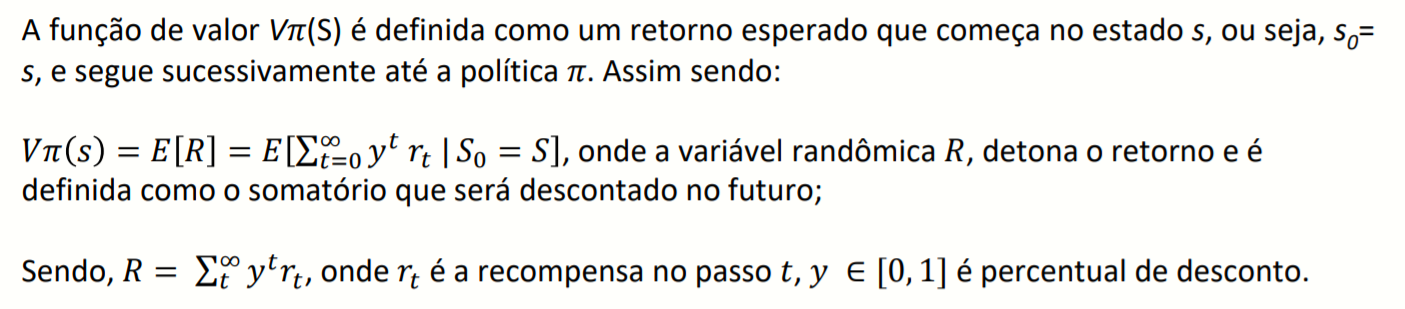
* O aprendizado por reforço só pode ser pensado sequencial em pares de ações, pois ocorrem uma após a outra.
* Modelo padrão de aprendizado por reforço
  + Um conjunto discreto de estados que o ambiente pode assumir;
  + Um conjunto discreto de ações que o agente pode tomar sobre o ambiente;
  + Um conjunto de valores escalares de reforço (geralmente {0,1}, ou os números reais).
* Aprendizado por reforço é não-determinístico, pois as ações podem gerar diferentes resultados.

## Elementos para uma Aprendizagem por Reforço

* Quatro elementos importantes para que ocorra a interação dinâmica do sistema:
  + Política de ação
  + Função de reforço
  + Função de valor
  + Modelo do Ambiente
* **Política de ação**
  + Mapeia o estado percebido do ambiente pelo agente para a ação a ser executada nesse estado, maximizando a satisfação dos seus objetivos.
  + As políticas determinam o comportamento do agente no ambiente.
  + As ações podem ser modeladas por um mapa de políticas:



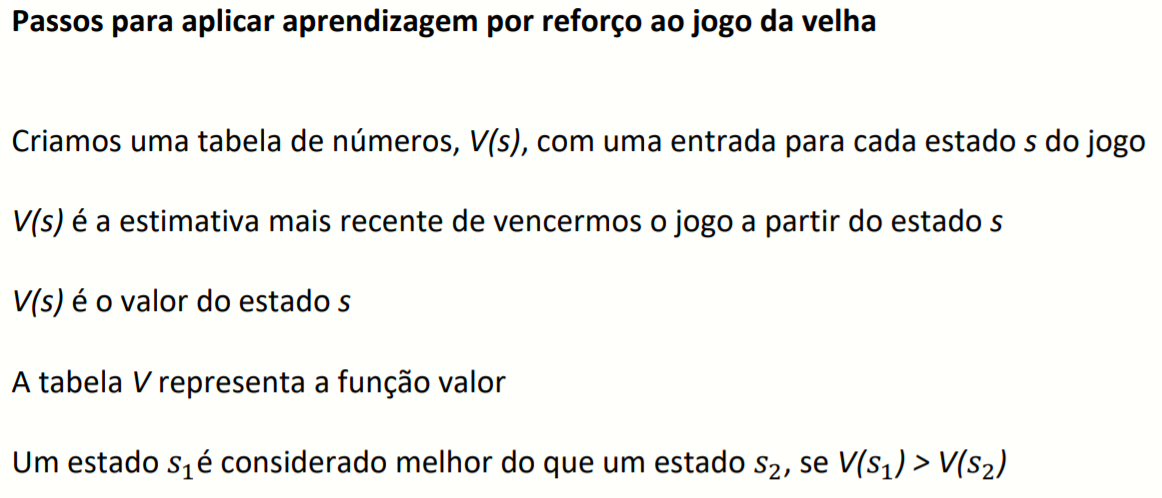
* + - S: Estados | A: Ações | s: Estado | a: Ação
  + O mapa de políticas fornece a **probabilidade** de escolher uma **ação a** quando estiver no **estado s**.
* **Função de reforço**
  + Mapeia estados do ambiente ou transição do ambiente de um estado para um outro para um número indicando a satisfação imediata dos objetivos do agente nesse estado ou no estado resultando da transição.
  + Em cada espaço de tempo o ambiente envia para o agente de aprendizagem por reforço um valor, um número correspondente a uma recompensa.
  + O sinal recebido de cada recompensa define quais ações são boas ou ruins para o agente.
  + A recompensa enviada depende da ação do agente e do estado do ambiente em que o agente se encontra.
* **Função de valor**
  + Mapeia o estado do ambiente para um número indicando a satisfação futura atingível dos objetivos do agente a partir desse estado.
  + As **funções de valores**, ao contrário das **funções de reforço**, indicam o que é bom para o sistema em longo prazo.
  + O que a função faz é garantir totalmente a recompensa que um agente espera acumular a partir daquele estado.
  + As funções de valor indicam o que é bom ou não para o sistema no longo prazo.

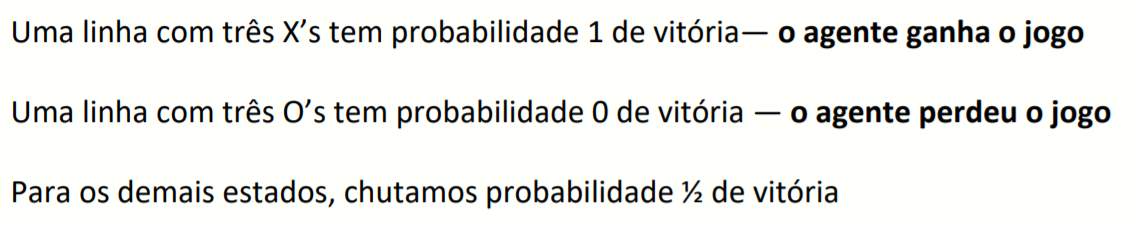


* **Modelo do ambiente**
  + Cada sistema de aprendizagem por reforço aprende um mapeamento de ações por meio de tentativa e erro com um ambiente dinâmico.
  + **Modelo perceptivo:** mapeia percepções para representação interna do estado do ambiente.
    - Exemplo: equipamentos com sensores que fazem a leitura do ambiente, descrição de símbolos ou processos mentais, como exemplo, a sensação de estar perdido em um lugar desconhecido e pensar em como sair daquele lugar, ou seja, mapeiam as percepções internas do estado do ambiente.
  + **Modelo efetivo:** mapeia ação a executar para representação interna do estado do ambiente. No modelo de sistema.

## Aplicações de Aprendizagem por Reforço

* **Jogo da Velha**
  + Agentes podem aprender sem supervisão especializada.
  + Os tipos de problemas mais adequados são aqueles complexos, nos quais parece não haver uma solução óbvia ou facilmente programável.
  + Jogo da velha: construir um agente que identifica as **imperfeições do oponente e aprende a maximizar as chances de sua vitória**.
  + Cenários: (1) um jogador experiente jamais perde esse jogo; (2) considere um adversário imperfeito, que às vezes toma decisões incorretas.
* O comportamento do oponente não pode ser conhecido a priori na aprendizagem por reforço, mas é viável a sua identificação por meio de interações.





## Exemplos no Mundo Real

* **Gerenciamento de Recursos em Clusters de Computadores**: O artigo “Gerenciamento de recursos com aprendizado por reforço profundo” explica como usar o aprendizado por reforço para aprender automaticamente a alocar e programar recursos de computador para tarefas em espera. O objetivo do referido trabalho é minimizar a desaceleração média da tarefa.
* **Controle de Semáforo**: No artigo “Sistema multiagente baseado em aprendizado por reforço para controle de sinais de tráfego de rede”, os pesquisadores tentaram projetar um controlador de semáforo para resolver o problema de congestionamento. Testados apenas em ambiente simulado, seus métodos mostraram resultados superiores aos métodos tradicionais e lançaram uma luz sobre os possíveis usos da aprendizagem por reforço de múltiplos agentes no projeto de sistemas de tráfego.
* **Robótica**: O artigo “Treinamento de ponta a ponta de políticas visomotoras profundas”, os pesquisadores treinaram um robô para aprender políticas para mapear imagens de vídeo brutas para as ações do robô. As imagens RGB foram alimentadas em uma CNN e as saídas foram os torques do motor. O componente RL era a pesquisa de política guiada para gerar dados de treinamento provenientes de sua própria distribuição de estado.
* **Configuração de Sistemas** **Web:** O artigo “Uma abordagem de aprendizado por reforço à auto-configuração do sistema Web online” mostrou a primeira tentativa no domínio sobre como fazer a reconfiguração autônoma de parâmetros em sistemas web multicamada em ambientes dinâmicos baseados em máquina virtual.
* **Química:** otimização de reações químicas.
* **Recomendações Personalizadas:** sistema de recomendação de notícias.
* **Problemas de controle:** como agendamento de elevador.
* **Preços dinâmicos:** ajustes de preços de acordo com a oferta e demanda.
* **Medicina:** no tratamento de cânceres, ajudando a processar dados clínicos para se chegar em uma estratégia de tratamento, usando-se indicadores recolhidos de pacientes.
* **Personalização de sugestões no e-Commerce:** reconhecer o perfil de cada cliente e sugerir preços e ofertas que terão mais chances de gerarem uma venda bem-sucedida.

## Aprendizagem por Reforço Seguro

* **Conceito:** Aprendizado por reforço seguro como o processo de **aprender políticas que maximizam a expectativa de retorno** em problemas nos quais é importante **garantir o desempenho razoável** do sistema e/ou respeitar as **restrições de segurança** durante os processos de aprendizado e/ou implantação.
* **Objetivo:** Criar um algoritmo de aprendizado que seja seguro durante o teste e o treinamento.
* **Exemplos de algumas restrições**
  + Exemplo 1: O caso de resfriamento do centro de dados, onde temperaturas e pressões devem ser mantidas abaixo dos respectivos limites em todos os momentos.
  + Exemplo 2: Um robô que não deve exceder os limites de velocidade, ângulos e torques.
  + Exemplo 3: Um veículo autônomo que deve respeitar suas restrições cinemáticas.
  + Esses problemas podem ser abordados de duas principais maneiras: **alterando os critérios de otimização** ou **alterando o processo de exploração**.
* Precisamos de formas para tratar as restrições da aprendizagem por reforço seguro. Usamos critérios de otimização.
* **Critérios de otimização**: Existem alguns métodos para incorporar o risco ao objetivo de otimização.
  + **Critérios de pior caso:** uma política é considerada ótima se tiver o retorno máximo do pior caso, ou seja, a pior recompensa obtida pela política é maximizada. Toda a tarefa simplifica a resolução do objetivo min-max abaixo:



* + - Onde Ω é um conjunto de trajetórias da forma (𝑠0, 𝑎0, 𝑠1, 𝑎1, … ) que ocorre sob a política π .
  + **Critérios sensíveis ao risco:** Inclui a notação de “risco” no objetivo de maximização da recompensa ao longo do prazo. Algumas literaturas definem como a variância do retorno. Considerando um exemplo de sensibilidade ao risco com base na função exponencial, uma função objetiva típica pode se parecer com:



* + - Uma expansão de Taylor do termo exp e log nos dá:



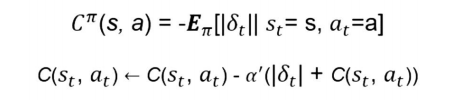
* + - Onde β denota o parâmetro sensível ao risco, com o efeito de que β é negativo tem como o objetivo reduzir a variação nas recompensas e, consequentemente o risco.
  + **Critérios restritos:** A expectativa de retorno está sujeita a uma ou mais restrições. A forma geral dessas restrições é mostrada abaixo:



* + - Onde 𝑐𝑖 são as restrições pertencentes ao conjunto 𝐶. Uma política é atualizada se for segura com certa confiança, dadas as restrições.

## Processo de Exploração

* **Comportamentos exploratórios clássicos na aprendizagem por reforço**
  + Assumem que o agente tem que explorar e aprender a pesar ações diferentes e a agir de forma otimizada.
  + O agente ignora o risco de ações, potencialmente terminando em estados perigosos. Explorações como a ε-greedy podem resultar em situações desastrosas.
  + Além disso, as políticas de exploração aleatória desperdiçam uma quantidade significativa de tempo explorando as regiões do estado e do espaço de ação onde a política ideal nunca será encontrada.
  + Inicialmente o agente deve tomar ações aleatórias, sem pensar em aumentar o retorno. O foco nesta fase é entender o ambiente **(exploration)**. Nas iterações seguintes o agente toma decisões para aumentar o retorno **(explotation)**.
    - **Ou seja, há uma mudança de comportamento de exploração (exploration) para melhora do retorno (explotation).**
  + É impossível evitar completamente situações indesejáveis em ambientes de risco sem conhecimento externo, porque o agente precisa visitar o estado perigoso pelo menos uma vez antes de rotulá-lo como “perigoso”. Pode haver duas maneiras de modificar o processo de exploração: **incorporar conhecimento externo ou exploração dirigida ao risco.**
* **Incorporar conhecimento externo**
  + Fornecer conhecimento inicial (pode ser considerado como um tipo de procedimento de inicialização) ou derivar uma política usando um conjunto finito de exemplos.
  + Exemplo: registrar um conjunto finito de demonstrações de um professor humano e fornecer a ele um algoritmo de regressão, para construir uma função Q parcial que pode ser usada para guiar ainda mais a exploração. Essas abordagens de inicialização não são suficientes para evitar situações perigosas que ocorrem na exploração.
  + Para derivar uma política de um conjunto de demonstrações, um professor, demonstra uma tarefa e as trajetórias de ações do estado são registradas.
  + Essas tarefas são usadas para derivar um modelo da dinâmica do sistema, e um algoritmo de aprendizagem por reforço encontra a política ótima nesse modelo.
* **Exploração dirigida ao risco**
  + Uma das abordagens define uma métrica de risco sobre a noção de “controlabilidade”.
  + Intuitivamente, se um estado particular (ou par de ação de estado) produzir muita variabilidade no sinal de erro de diferença temporal, ele será menos controlável. A controlabilidade do par de ação do estado é definida como:



* + - Onde δ é o sinal de erro de diferença temporal. O algoritmo de exploração procura usar controlabilidade como uma heurística de exploração em vez de uma exploração geral de Boltzmann. O agente é encorajado a escolher regiões controláveis do ambiente.

## Aprendizagem por Reforço Inversa

* **Conceito**
  + **Aprendizagem por reforço**, o objetivo é aprender um processo de decisão para produzir um comportamento que maximize alguma função de recompensa predefinida.
    - Foca nas melhores ações para atingir uma meta ou um objetivo.
  + **Aprendizado por Reforço Inverso (IRL)**, como descrito por Russell e Norvig (2010), aborda o problema e tenta extrair a função de recompensa do comportamento observado de um agente.
    - Decifra o objetivo subjacente quando recebe um conjunto de ações, estimando qual a função de recompensa que um humano deseja otimizar.
    - Torna-se cooperativa quando máquina e ser humano interagem para aperfeiçoar seu conhecimento sobre cenários humanos.
* **Exemplo - dirigir um carro autônomo**
  + Uma abordagem simples seria criar uma função de recompensa que capte o comportamento desejado de um motorista: parar nos semáforos, ficar fora da calçada, evitar pedestres e assim por diante.
  + Infelizmente, isso exigiria uma **lista exaustiva** de todos os comportamentos que gostaríamos de considerar, além de uma lista de pesos descrevendo a importância de cada comportamento (imagine ter que decidir exatamente o quanto os pedestres são mais importantes do que os sinais de trânsito!).
  + Na estrutura da IRL, a tarefa é pegar um conjunto de dados de condução gerados pelo homem e extrair uma aproximação da função de recompensa desse humano para a tarefa.
  + É claro que essa aproximação necessariamente lida com um modelo simplificado de direção. Grande parte da informação necessária para resolver um problema é capturada dentro da aproximação da função de recompensa verdadeira.
  + Segundo Russell e Norvig (2010), a **função de recompensa**, e não a política, é a **definição mais sucinta, robusta e transferível da tarefa**, já que quantifica quão boas ou ruins certas ações são.
  + Uma vez que temos a função de recompensa correta, o problema é encontrar a política correta, podendo ser resolvido com métodos de aprendizado por reforço padrão.
  + Neste caso, estaríamos usando dados de condução humana para aprender automaticamente os pesos de recursos certos para a recompensa.
  + A tarefa é descrita completamente pela função de recompensa, nem precisamos saber os detalhes da política humana, desde que tenhamos a função de recompensa correta para otimizar.
  + Os algoritmos que resolvem o problema da IRL podem ser vistos como **um método para alavancar o conhecimento especializado** para converter uma descrição de tarefa em função de recompensa compacta.
* **Alguns problemas**
  + O principal deles é converter uma tarefa complexa em uma função de recompensa simples, pois determinada política pode ser ideal para muitas funções de recompensas diferentes.
  + Ou seja, mesmo que tenhamos as ações de um especialista, existem muitas funções diferentes de recompensas que o especialista pode estar tentando minimizar.
  + Algumas dessas funções são bem simples: todas as políticas são ideais para a função de recompensa que é zero em todos os lugares, portanto, essa função de recompensa é sempre uma possível solução para o problema da IRL.
  + Porém, é necessário que a função de recompensa capture informações significativas sobre a tarefa e seja capaz de diferenciar claramente entre políticas desejadas e não desejadas.
* **Diante dos problemas**: Russell e Norvig (2010) formularam o aprendizado de reforço inverso como um problema de otimização, onde se quer escolher uma função de recompensa para a qual a política especializada especificada é a ideal. Mas, dada essa restrição, também se quer uma função de recompensa que maximize adicionalmente certas propriedades importantes.

## Aprendizagem por Reforço para Espaços Finitos

* Considere o conjunto de políticas ótimas para um Processo de Decisão de Markov (MDP) (que será visto na Unidade II) como um espaço de estado finito 𝑆, um conjunto de ações 𝐴 e matrizes de probabilidade de transição 𝑃𝑎 para cada ação.
* Russell e Norvig (2010) provaram que uma política 𝜋 dada por 𝜋 (𝑠) = 𝑎1 é ideal se, e somente se, para todas as outras ações, o vetor de recompensa 𝑅 (que lista as recompensas para cada estado possível) satisfaz a condição:



* A importância desse teorema é que ele mostra que podemos selecionar de forma eficiente a “melhor” função de recompensa para a qual 𝑎 é política ótima, usando algoritmos de programação linear.
* Apenas um estado é realmente recompensado.
* Isso permitiu que diversas literaturas tratem a IRL como um problema de otimização tratável, em que se tenta otimizar as heurísticas a seguir para que a função de recompensa se “encaixe” bem nos dados do especialista.
* **Passos**
  + **Maximize** a **diferença** entre a **qualidade da ação ótima** e a **qualidade da próxima melhor ação** (sujeita a um limite na magnitude da recompensa, para evitar diferenças arbitrariamente grandes). Logo, se quer encontrar uma política ótima que se distingue claramente de outras políticas possíveis.
  + **Minimize o tamanho das recompensas na função/vetor de recompensa.** A intuição é que o uso de pequenas recompensas incentiva a função de recompensa a ser mais simples, semelhante à regularização na aprendizagem supervisionada.
  + Russell e Norvig (2010) escolhem a norma 𝐿1 com um coeficiente de penalidade ajustável, que incentiva o vetor de recompensa a ser diferente de zero em alguns estados.
* **Referências**: RUSSELL, S.; NORVIG, P. Artificial Intelligence: A Modern Approach. 3.ed. New Jersey: Pearson Education, 2010.

## Aprendizagem por Reforço Inverso de Trajetórias Amostradas

* Russell e Norvig (2010) descreveram algoritmos de IRL para casos em que, em vez de uma política ótima total, podemos apenas amostrar trajetórias, a partir de uma política ótima.
* São conhecidos os estados, ações e recompensas geradas por uma política para um número finito de episódios, mas não a política em si.
* Esta situação é mais comum em casos aplicados, especialmente aqueles que lidam com dados de humanos especialistas.
* Nesta formulação do problema, substitui-se o vetor de recompensa que é usado para espaços de estados finitos com uma aproximação linear da função de recompensa, que usa um conjunto de funções para obter vetores de recursos com valor real (s).
* Recursos capturam informações importantes de um espaço de estados de alta dimensão (por exemplo, ao invés de armazenar a localização de um carro durante cada etapa de tempo, podemos armazenar sua velocidade média como um recurso).
* Para cada recurso 𝜙𝑖(𝑠) e peso 𝑎𝑖 temos:



* Onde o objetivo é encontrar os valores mais adequados para cada peso característico 𝑎𝑖 .
* A ideia por trás do IRL com trajetórias amostradas é melhorar de forma iterativa uma função de recompensa comparando o valor da política especializada aproximadamente ótima com um conjunto de políticas 𝑘 geradas.
* O algoritmo é inicializado gerando pesos aleatoriamente para a função de recompensa estimada e inicializando o conjunto de políticas candidatas com uma política gerada aleatoriamente.
* Principais passos para o algoritmo:
  + Estimar o valor da política ótima para o estado inicial 𝑣 𝜋 (𝑠0), bem como o valor de cada política gerada 𝑣 𝜋𝑖 (𝑠0) tomando a recompensa acumulada média de muitos ensaios aleatoriamente amostrados;
  + Gerar uma estimativa da função de recompensa 𝑅 resolvendo um problema de programação linear. Especificamente, define-se para maximizar a diferença entre a política ótima e cada um das outras 𝑘 políticas gerada;
  + Após um grande número de iterações, finalizar o algoritmo nessa etapa;
  + Caso contrário, usar um algoritmo de aprendizagem por reforço padrão para encontrar a política ideal para 𝑅. Essa política pode ser diferente da política ótima dada, já que nossa função de recompensa estimada não é necessariamente idêntica à função de recompensa estimada que estamos procurando;
  + Adicionar a política recém-gerada ao conjunto de políticas 𝑘 candidatas e repita o procedimento.

## Aprendizagem da Aprendizagem: Aprender com um Especialista

* Além de aprender uma função de recompensa de um especialista, podemos também aprender diretamente uma política para ter um desempenho comparável ao especialista.
* Útil se tivermos uma política especializada que seja apenas aproximadamente ideal.
* Abbeedl e Andrew (2004) propuseram um algoritmo que usa um MDP (Markov Decision Process) e uma política de um “especialista” aproximadamente ideal, e em seguida, aprende uma política com desempenho comparável ou melhor do que a política do especialista, usando exploração mínima.
* Essa propriedade mínima de exploração acaba sendo muito útil em tarefas frágeis como um voo autônomo de helicóptero.
  + **Exemplo:** Um algoritmo tradicional de aprendizagem por reforço poderia começar a explorar aleatoriamente, o que quase certamente levaria a um acidente de helicóptero no primeiro teste. Idealmente, poderíamos usar dados de especialistas para começar uma política de linha de base que pode ser melhorada com segurança ao longo do tempo.
    - Essa política de linha de base deve ser significativamente melhor do que uma política inicializada aleatoriamente, o que acelera a convergência.
* **Algoritmo de aprendizagem**
  + **Ideia principal:** usar ensaios da política de especialistas para obter informações sobre o MDP subjacente e, em seguida, executar iterativamente uma melhor estimativa da política ótima para o MDP real.
  + A execução de uma política também nos fornece dados sobre as transições do ambiente, que pode ser usada para melhorar a precisão do MDP estimado.
* **Funcionamento do algoritmo em um ambiente discreto**
  + Primeiro, usamos uma política de especialistas para aprender sobre o MDP:
  + 1. Execute uma quantidade fixa de testes usando a política de especialistas, registrando cada trajetória de ação do estado;
  + 2. Estime as probabilidades de transição para cada par de ação do estado usando os dados registrados por meio da estimativa de máxima verossimilhança;
  + 3. Estime o valor da política especializada, calculando a média da recompensa total em cada tentativa.
* **Aprende-se uma nova política**
  + 1. Aprenda uma política ótima para o MDP estimado usando qualquer algoritmo de aprendizagem por reforço padrão;
  + 2. Teste a política de aprendizado no ambiente real;
  + 3. Se o desempenho não for suficientemente próximo do valor da política especializada, adicione as trajetórias de ação do estado dessa avaliação ao conjunto de treinamento e repita o procedimento para aprender uma nova política.
  + **Vantagem:** em cada estágio, a política que está sendo testada é a melhor estimativa para a política ótima do sistema. Há uma diminuição na exploração, mas a ideia central do aprendizado é que podemos supor que a política especializada já está próxima do ideal.

## Resumo da Unidade 01

* O problema de aprendizagem por reforço envolve dois elementos: o **ambiente** e o **agente**.
* A cada instante de tempo, o ambiente se encontra em um estado, cuja evolução é condicionada pelas ações que o agente toma. O agente é capaz de ler, a cada momento, o estado do ambiente, e responder com uma ação.
* Após cada ação, o agente recebe um sinal de recompensa do ambiente. O objetivo do agente é maximizar a recompensa acumulada ao longo do tempo.
* A separação entre agente e ambiente é flexível, e depende da formulação do problema:
* Por exemplo, o agente pode ser:
  + Nosso corpo, recebendo sinais físicos através dos sentidos, ou então apenas o cérebro, recebendo sinais eletroquímicos do resto do corpo através dos neurônios.
  + Grupo de neurônios interagindo com o resto do cérebro.
* Determinada ação executada pelo agente pode levar a períodos intermediários de recompensa negativa, para só mais tarde trazer bons resultados.
* O agente deve ser capaz de realizar um planejamento visando não apenas uma boa recompensa imediata, mas uma boa recompensa acumulada a longo prazo. O agente não deve ser guloso em relação à recompensa imediata.

# Unidade 02 - Processos de decisão de Markov

Essa unidade tem como objetivo:

* Apresentar o ambiente: processo de decisão de Markov;
* Exemplificar os algoritmos para solução de MDPs;
* Descrever os processos de decisão de Markov parcialmente observáveis.

## O Ambiente - Processos de Decisão de Markov

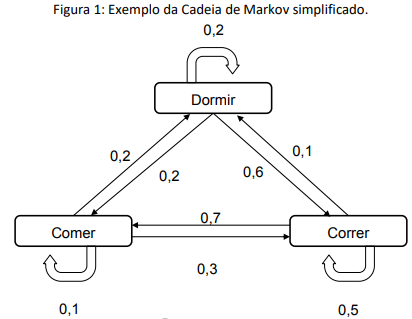
* **Cenários representados usando processos de decisão de Markov**
  + Representar situações onde é necessário executar ações em sequência em ambientes com incerteza.
* **Cenário:** pode ser modelado como um processo de **decisão de Markov** abrange um sistema com vários estados, assim como ações que provavelmente modifiquem o estado do sistema e a probabilidade de perceber o resultado de cada ação executada. Dada a descrição do problema, resolvê-lo significa encontrar uma política ótima que determine a cada momento que ação tomar para maximizar a recompensa esperada (minimizar o custo esperado).
* **Elementos:** o **ambiente** é visto como um processo de decisão de Markov, e o **agente** expressa-se em uma política que associa estados do ambiente com probabilidades de executar cada uma das ações pertencentes a um conjunto de ações possíveis.

**Markov Decision Process (MDP ou Processo de Decisão de Markov)**

* É uma maneira de construir processos onde as transições entre estados são probabilísticas, sendo possível observar em que estado o processo está sendo possível interferir no processo de tempos em tempos executando ações (HAUSKRECHT, 1997; PUTERMAN, 1994).
* Ações têm recompensas – ou custo – que decorrem do estado em que o processo se encontra. Recompensas podem ser definidas apenas por estado, sem que essas dependam da ação executada. Chamamos de “Markov” (ou “Markovianos”) os processos que foram construídos obedecendo à propriedade de Markov.
* O resultado de uma ação em um estado está sujeito a ação e ao estado atual do sistema e são chamados de “processos de decisão” porque modelam a possibilidade de uma agente (ou “tomador de decisões”) interferir periodicamente no sistema executando ações, diferentemente de **Cadeias de Markov**, onde não se trata de como interferir no processo.

## Exemplo Cadeia de Markov

* **Processo de decisão de Markov:** Uma maneira de descrever problemas de Aprendizado por Reforço que requerem tomadas de ações com incertezas, como diagnósticos médicos, tratamentos de enfermidades, controle da movimentação de um robô, e muito mais.
  + Consiste de um conjunto de estados (com estado inicial 𝑠0); um conjunto de ações aplicáveis em cada estado; um modelo de transição e uma função de recompensa R(s).
  + O resultado de uma ação em um estado está sujeito a ação e ao estado atual do sistema (e não como o processo chegou a este estado).
  + Chamados de “processos de decisão” porque modelam a possibilidade de uma agente interferir periodicamente no sistema executando ações, diferentemente de **Cadeias de Markov**, onde não se trata de como interferir no processo.
* “Dado o presente, o futuro independe do passado”.
* **Conceito de Cadeias de Markov:** um estado seguirá essa propriedade caso todos os estados anteriores a ele não influenciarem na decisão do próximo estado, o único que influencia é apenas o estado atual.
* **Exemplo:** Considere algumas atividades completamente independentes para realizar. Com elas identifique também os graus de probabilidade de ocorrência de uma, dada à outra.

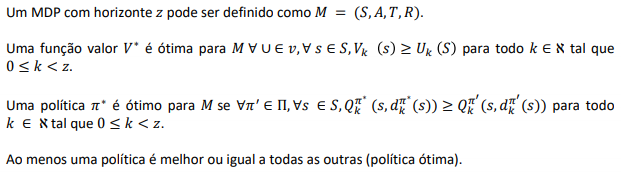


* + A ideia é simples, vamos analisar as probabilidades das ações continuarem ou serem revertidas em outras atividades. No caso, avaliar as três ações, particularmente independentes, mas que podem levar uma a outra.
  + O conjunto de ações é: S = {Comer, Dormir, Correr}
  + As probabilidades associadas, no exemplo, mostra que a correlação probabilística entre as ações é a seguinte:
    - P(Correr) = 0,5;
    - P(Comer) = 0,1;
    - P(Dormir) = 0,2;
    - P(Comer | Dormir) = 0,2;
    - P(Dormir | Comer) = 0,2;
    - P(Correr | Dormir) = 0,1;
    - P(Dormir | Correr) = 0,6;
    - P(Comer | Correr) = 0,3;
    - P(Correr | Comer) = 0,7;

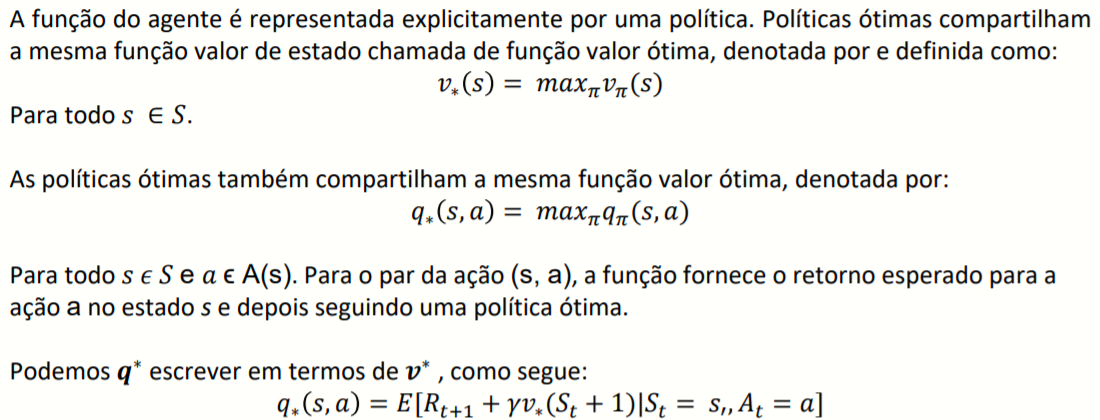
## Definição de um MDP

## Política e Função Valor de um MDP

* **Política e função valor**
  + A maioria dos algoritmos de aprendizagem por reforço envolvem a estimativa de valor de funções - funções de estados (ou de pares de ação do estado) que estimam quão bom é para o agente estar em um determinado estado (ou como é bom executar uma determinada ação em um determinado estado).
  + O termo “quão bom” pode ser definido em termos de recompensas futuras que podem ser esperadas ou para ser mais preciso, em termos de retorno esperado.
  + As recompensas que o agente pode esperar receber no futuro dependem das ações.
  + **Função valor** pode ser definida como as formas particulares de agir, chamadas **políticas.**
* **Definição: Política para um MDP**
  + A regra de decisão para um processo de decisão de Markov em uma época de decisão 𝑘 pode ser definida pela função 𝑑𝑘 ∶ 𝑆 ↦ 𝐴, que estabelece a ação a ser efetivada, dado o estado do sistema.
  + Uma política para um MDP é uma sequência de regras de decisão 𝜋 = {𝑑0, 𝑑1, … , 𝑑𝑧 −1}, uma para cada época de decisão.
  + É necessário também que seja definida a ideia de valor de uma política, determinada por uma **função valor**.
* **Definição: função valor de uma política para um MDP**
  + Uma função valor da política 𝜋 para um MDP 𝑀 = (𝑆, 𝐴, 𝑇, 𝑅) é definida como a função 𝑉 𝜋 ∶ 𝑆 ↦ 𝑅, tal que 𝑉 𝜋 (𝑠) atribuído ao valor que se espera da recompensa para esta política, consoante com o critério de otimalidade.
* **Política ótima e função valor ótima de um MDP**
  + Dada política processada a partir do estado inicial, a natureza estocástica do ambiente pode levar a um histórico de ambiente distinto.
  + A qualidade de uma política pode ser medida pela utilidade esperada dos históricos de possíveis ambientes concebidos por essa política. Uma **política ótima** é, portanto, uma que fornece a utilidade que se espera mais alta (RUSSELL e NORVIG, 2010).
  + Resolver uma tarefa de aprendizado de reforço significa encontrar uma política que alcance muita recompensa a longo prazo.
* **Definição: política ótima para um MDP**
  + Podemos definir precisamente uma política ótima da seguinte forma para MDPs finitos:



* + - Há uma única função que satisfaz o critério acima, chamada, portanto, de função valor ótimo.
    - Duas políticas podem ter a mesma função valor.



## Algoritmos para Solução de MDPs

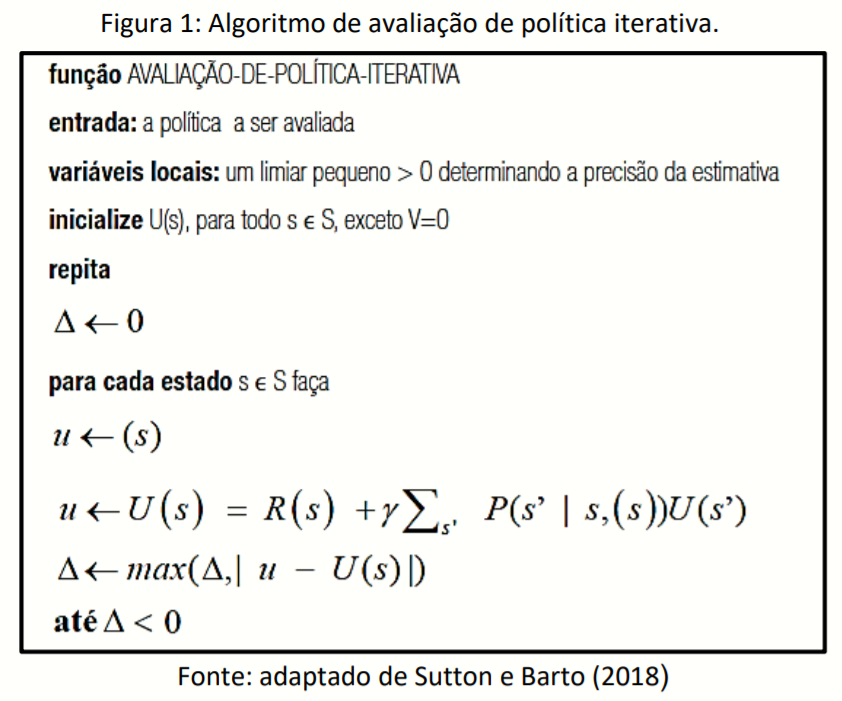
* **Algoritmos: iteração por política e iteração de valor**
  + **Iteração por política:** sugere um caminho alternativo para encontrar políticas ótimas, que mostra que, quando uma ação é claramente melhor que todas as outras, a magnitude exata das utilidades nos estados envolvidos não precisa ser exata.
    - O algoritmo de iteração política alterna a **avaliação de política** e a **melhoria de polític**a começando com uma política inicial 𝜋0.
  + **Iteração de valor:** para calcular uma política ótima. O propósito é calcular a utilidade de cada estado. Em seguida, utilizar as utilidades do estado para eleger uma ação ótima em cada estado.
* **Avaliação de política**
  + Dada uma política 𝜋𝑖 , calcular 𝑈𝑖 = 𝑈 𝜋𝑖 se a utilidade de cada estado fosse executada. A avaliação de política pode ser expressa como uma versão simplificada da equação de Bellman, relacionando a utilidade de 𝑠 (sob 𝜋𝑖 ), às utilidades de seus vizinhos:



* + Espaços de estados pequenos, a avaliação de política usando métodos de solução exata em geral é a abordagem mais eficiente. Espaços de estados grandes, o tempo do algoritmo sendo 𝑂(𝑛³) talvez não seja o recomendado.
  + Não é necessário fazer a avaliação de política exata. Podemos executar algum número de passos de iteração de valor simplificada para fornecer uma aproximação razoavelmente boa das utilidades. Nesse caso, a atualização de Bellman simplificada para esse processo é:



* + Repetida 𝑘 vezes para produzir a próxima estimativa de utilidade.
* **Avaliação de política iterativa – algoritmo que implementa a atualização de Bellman**



* + Observe como o algoritmo lida com a terminação.
  + Formalmente a avaliação de política iterativa converge apenas no limite, mas na prática deve ser interrompida.
  + Dessa forma, o algoritmo testa a quantidade max 𝑠 ∈𝑆 | 𝑈𝑖+1 𝑠 − 𝑈𝑖 𝑠 | depois de cada varredura e para quando é suficientemente pequeno.

## Melhoria de Política

* **Melhoria de política**
  + Consiste no processo de fazer uma nova política que melhora uma política original, tornando-a gulosa no que diz respeito à função valor da política original.
  + Suponha que a nova política gulosa, 𝜋′ é tão boa quanto, mas não melhor que, a política original 𝜋. Em outras palavras, vamos considerar a nova política gulosa 𝜋′ , dada por:



* + - Onde, 𝑎𝑟𝑔𝑚𝑎𝑥𝑎 denota o valor de 𝑎 em que a expressão que se segue é maximizada.
  + A política gulosa toma a ação que parece melhor a curto prazo, de acordo com 𝑣𝜋 .
* **Teorema da melhoria de política**
  + Seja 𝜋 e 𝜋′ qualquer par de políticas determinísticas tais que, para todo 𝑠 𝜖 𝑆,



* + A política deve ser tão boa ou melhor.
  + Podemos verificar que a política gulosa satisfaz as condições do teorema da melhoria política, e é tão boa quanto, ou melhor, que a política original.
  + Melhoria de política, deve nos dar uma política estritamente melhor, exceto quando a política original já é ótima.
* **Algoritmo da iteração de política**
  + A maneira para encontrar uma política ótima é chamada de iteração de política.
  + Cada política é garantidamente uma melhoria estrita em relação à anterior. Como um MDP finito tem apenas um número finito de políticas, esse processo deve convergir para uma política ótima e uma função valor ótima em um número finito de iterações.

## O Algoritmo de Iteração de Valor

* **A equação de Bellman**
  + A utilidade de uma sequência de estados é a soma das recompensas descontadas obtidas durante a sequência. Possibilita comparar políticas, comparando as utilidades esperadas obtidas quando as executamos.
  + Há uma relação direta entre a utilidade de um estado e a utilidade de seus vizinhos: “a utilidade de um estado é a recompensa imediata correspondente a esse estado mais a utilidade descontada esperada do próximo estado, assumindo que o agente escolha a ação ótima” (RUSSELL e NORVIG, 2010).
  + A utilidade de um estado 𝑠 é conhecida como Equação de Bellman e pode ser definida da seguinte forma:



* + Onde:
    - 𝛾 é o fator de desconto;
    - 𝑈(𝑆) é a recompensa total a “longo prazo” de 𝑠;
    - 𝑅(𝑠) é a recompensa a “curto prazo” por estar em 𝑠.
  + Toda função valor satisfaz essa equação, dada uma política
* **O algoritmo de interação de valor**
  + A equação de Bellman é a base do algoritmo de iteração de valor para a solução de MDPs. Caso haja 𝑛 estados possíveis, teremos 𝑛 equações de Bellman, sendo uma para cada estado. Assim, 𝑛 equações contêm 𝑛 incógnitas - as utilidades dos estados (RUSSELL e NORVIG, 2010).
  + Há um problema ao tentar **resolver** essas **equações simultaneamente** para encontrar as utilidades: as **equações não são** lineares porque o **operador** 𝑚𝑎𝑥 **não** é um operador **linear**.
  + Em sistemas de equações lineares normalmente são solucionados rapidamente usando técnicas de álgebra linear (sistemas de equações não lineares são mais difíceis de ser solucionados).
  + Sanar esse problema: **abordagem iterativa**.

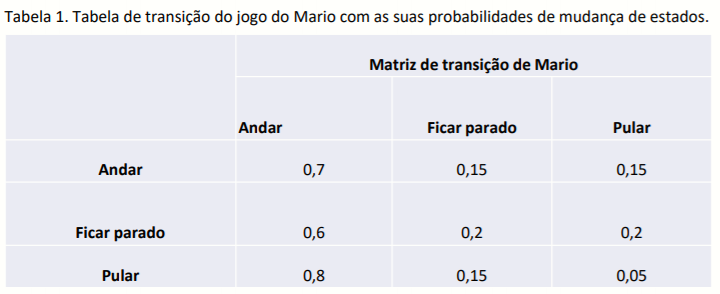
## Contração em um Espaço Banach

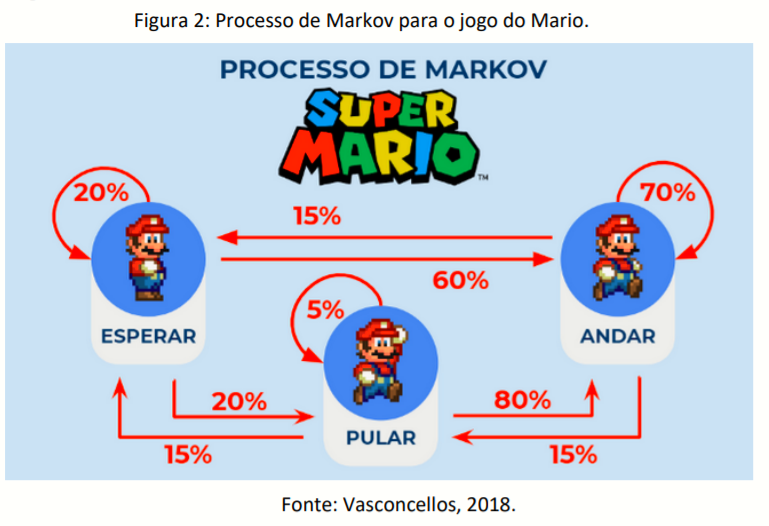
## Exemplo do Super Mario

* Aprendizagem por reforço leva em consideração recompensas ou punições de acordo com as ações que o agente executar.
* O agente vai aprendendo quais ações ele deve seguir para obter o maior número de recompensas, e assim poder atingir o seu objetivo.
* **Jogo do Super Mario**
  + Leva em consideração o Mario em uma fase do jogo, com o objetivo de salvar a Princesa das garras do Bowser.
  + O objetivo somente será atingido se ele seguir algumas regras e caminhos: ele precisa passar por várias fases, derrotando os inimigos que aparecem pelo caminho e pegando os brindes que aparecerem.
  + Para passar de uma fase no jogo, há a necessidade de um algoritmo treinado, imagine que é sabido quando, quantas vezes e quais botões, em quais sequências é preciso apertar para cada tipo de desafio na fase.
* **O algoritmo de interação de valor**

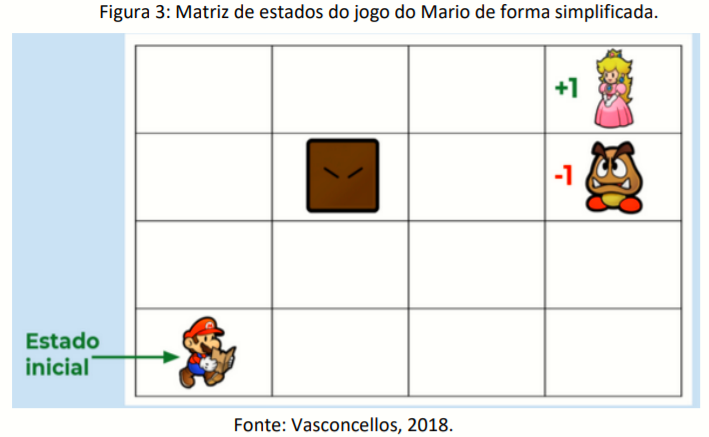


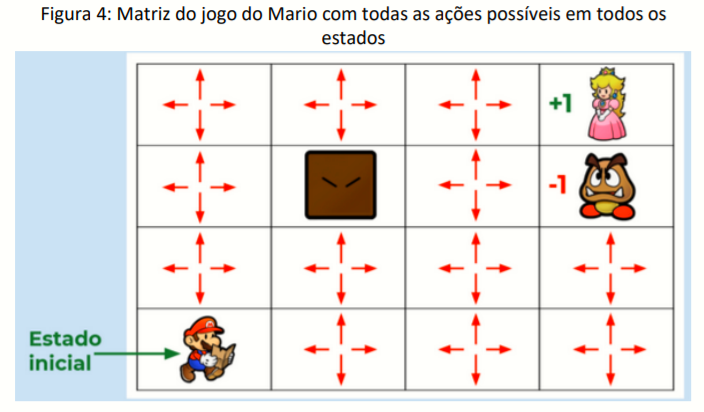
* Como já é sabido, na Cadeia de Markov, o estado que o Mario vai estar no futuro só depende do estado em que ele está no momento.
* Considere que ele está parado, o fato dele está parado vai indicar no seu próximo movimento e estado, mas como ele estava antes de estar parado não importa.
* Isto é, não interessa se Mario ficou pulando loucamente para passar um espaço vago, se ele está parado nos últimos minutos, é o que precisa ser analisado.

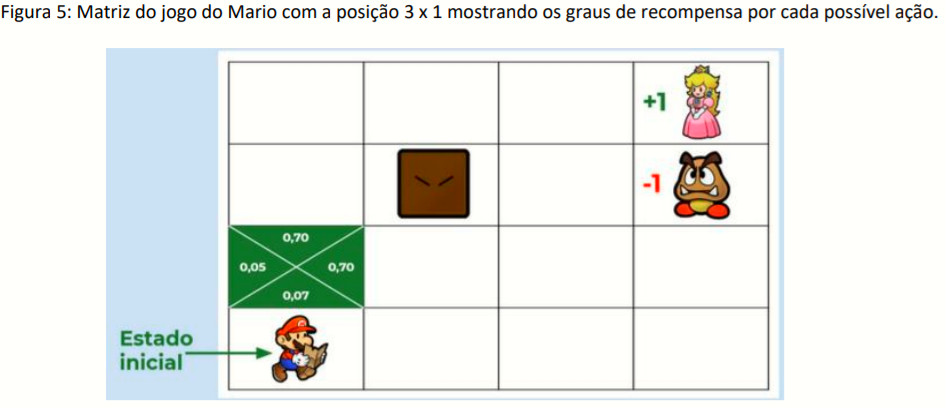




* O processo de decisão de markov é representado por algumas variáveis, que são:
  + **S (conjunto finito de estados)** - o estado que o seu personagem ou o seu objeto pode se encontrar. No jogo do Mario, os exemplos de estados foram está parado, andando ou pulando. Porém, lembre-se que isto é avaliado para cada caso, imagine que eu estou jogando xadrez, o estado “pulando” não iria se encaixar.
  + **A (conjunto finito de ações)** - igualmente intuitivo, a ação é o ato de fazer algo acontecer. Se quero que o Mario pule, eu preciso tomar alguma iniciativa, fazer alguma ação para que ele vá para o estado “pulando”. Como o assunto é o jogo do Mario, as ações são os movimentos que o jogador faz para que seu avatar (Mario, Luigi) vá para um estado, por exemplo, eu cliquei no botão “↑↑↑”, essa ação faz com que meu Mario pule 3 vezes.
  + **P (modelo de probabilidade)** - a probabilidade de uma ação que será escolhida pelo jogador para um estado futuro, ser baseada no estado atual do Mario. Se a fase é aquela em que tenho que “ir para o céu”, então a probabilidade de eu escolher a ação “↑” e ir para o estado pulando é muito maior do que a ação “←” e ir para o estado andando.
  + **R (recompensa)** - é uma variável que representa um número que o Mario vai receber depois de executar a ação passada para ele e ele entrar no estado atual. Neste caso tem-se duas possibilidades como vimos nos conceitos, uma de recompensa positiva e uma de recompensa negativa. Imagine que você tentou pular para ir para o céu mas errou e caiu no abismo, sua ação levou a uma recompensa negativa, você foi punido, você perdeu uma vida! Mas se você pulou certo e chegou num bloco que te deu um cogumelo, sua ação te levou a uma recompensa positiva, e você foi beneficiado com esta ação.
  + **𝛾(fator de desconto)** - é um número, que normalmente varia entre 0 e 1, que molda o total de recompensa que o agente vai ganhar no futuro. Vamos supor que no jogo, se Mario pegar o caminho A qualquer, que tem um fator de desconto de 0,7 e pode pegar 100 moedas. Mas se ele escolher o caminho B, que tem um fator de desconto 0,9, ele pode pegar 300 moedas. Isto significa dizer que suas ações atuais (que não tem influência nas ações futuras), ainda sim, podem interferir nelas - maximizar a recompensa futura, não só a atual.



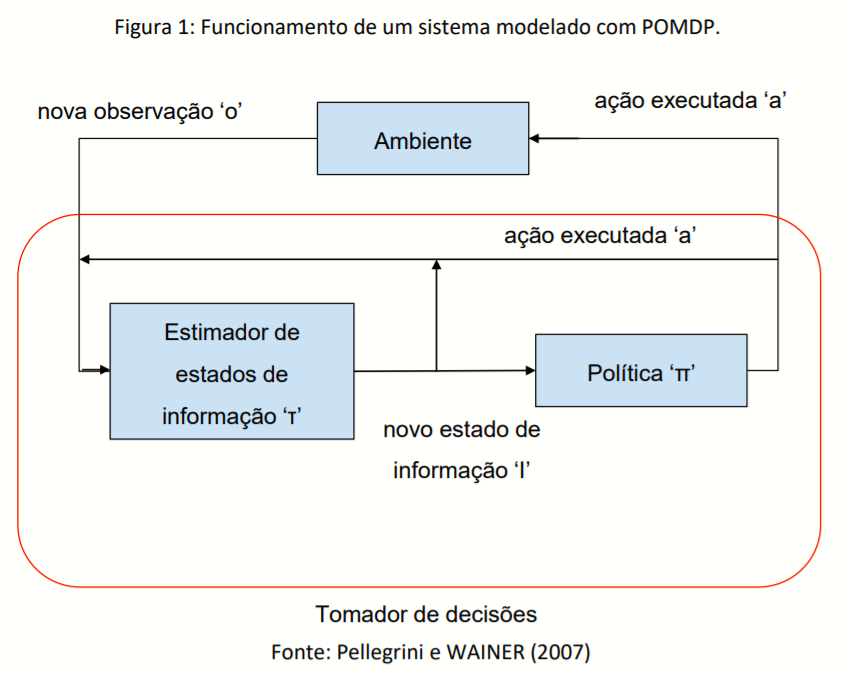




## Processos de Decisão de Markov Parcialmente Observáveis

* **Conceito**
  + Um processo de decisão de Markov parcialmente observável (POMDP) é uma generalização de MDPs na qual o estado atual do sistema não é obrigatoriamente conhecido.
  + O agente recorda das ações que ele efetuou e das observações que reparou ao longo do tempo, tentando utilizar ainda essas informações para perceber sua próxima decisão.
  + Por exemplo: É provável que ao contrário de um “estado atual do sistema”, uma distribuição de possibilidades seja mantida durante o tempo em que as decisões são tomadas.
  + POMDPs são mais difíceis de solucionar que os MDPs, no entanto, são mais significativos.
* **POMDPs usado para solucionar problemas em diversas áreas**: Manutenção de máquinas, navegação de robôs, controle de elevadores, visão computacional, modelagem de comportamento em ecossistemas, aplicações militares, diagnóstico médico, educação e diversas outras áreas.
* Em um POMDP não temos a oportunidade de verificar de forma direta o estado em que o sistema se encontra em um certo momento, ou seja, o agente não conhece o estado atual 𝑠 (de forma análoga ao que acontece num problema construído como MDP).
* Cada ação resulta em alguma percepção que é probabilisticamente relacionada ao estado do sistema.
* Como o estado atual do sistema não é acessível ao agente, é possível usar o histórico anterior de ações e percepções para escolher a melhor ação.

## Dinâmica de Sistemas Modelados como POMDPs



## Estado de Informação

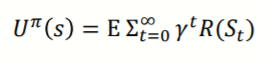
# Unidade 03 - Aprendizagem por reforço passiva e ativa

Essa unidade tem como objetivo:

* Apresentar a aprendizagem por reforço passiva;
* Descrever a aprendizagem por reforço ativa.

## Aprendizagem por Reforço Passiva

* **Conceito**
  + **Aprendizagem por reforço passiva** é quando o ambiente passa de um estado para outro **sem intervenção do agente**, onde a cada transição o agente apenas percebe o estado e recebe reforço, atualizando sua representação do valor desse estado.
  + Faz uso de um conceito baseado em estados, num ambiente completamente observável. A política do agente é fixa, no estado s ele executa a ação (𝑠).
  + A tarefa aprendizagem passiva é semelhante à de avaliação de política. A **diferença principal** é que o agente de aprendizagem passiva **não conhece o modelo de transição** 𝑃(𝑠’ | 𝑠, 𝑎) descrito no algoritmo de iteração de política (visto na seção 2.1.4 da Unidade II), que especifica a probabilidade de alcançar o estado 𝑠’ a partir do estado 𝑠 e depois realizar a ação 𝑎; ele também **não conhece a função de recompensa** 𝑅(𝑠), que especifica a recompensa para cada estado (RUSSELL e NORVIG, 2010).
* O agente executa um conjunto de experiências no ambiente usando sua política. Em cada experiência, o agente começa num estado inicial e experimenta uma sequência de transições de estados até alcançar um dos estados terminais. Suas percepções fornecem tanto o estado atual quanto a recompensa recebida nesse estado.
* A seguinte equação descreve a aprendizagem por reforço passiva:



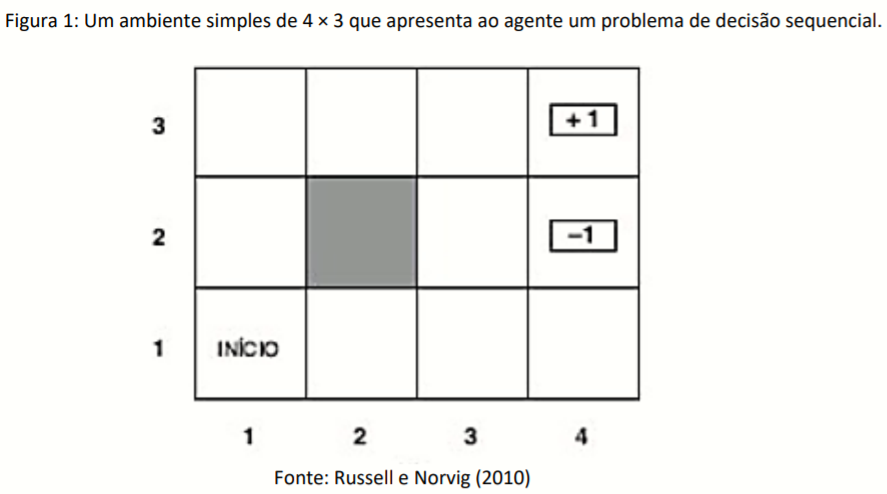
* **Objetivo**
  + Utilizar as informações sobre recompensas para aprender a utilidade esperada associada a cada estado não terminal s. A utilidade é definida como a soma esperada de recompensas (descontadas) obtidas se a política π for seguida.
  + A recompensa 𝑅(𝑠) para o estado 𝑆𝑡 (uma variável aleatória) é o estado alcançado no tempo 𝑡 quando é executada a política 𝜋 e 𝑆0= 𝑠, e a variável 𝛾 representa o fator de desconto.
  + Métodos: **Estimativa de Utilidade Direta, Programação Dinâmica Adaptativa** e **Diferença Temporal**.
* **Estimativa de utilidade direta**
  + Ainda estamos operando sob um ambiente estocástico, portanto, uma ação específica executada em um estado específico nem sempre leva ao mesmo estado seguinte.
  + Se quisermos aprender as utilidades desses estados sob uma política fixa, poderemos imaginar uma maneira bastante direta de fazê-lo:
    - execute a política várias vezes;
    - no final de cada execução, calcule a utilidade para cada estado na sequência (lembre-se, a utilidade de um estado é a soma de recompensas para esse estado e todos os estados subsequentes;
    - atualize a utilidade média para cada um dos estados que observamos com nossos novos pontos de dados.
  + **Ideia:** utilidade de cada estado é a recompensa total que se espera a contar desse estado em diante (conhecido como recompensa a obter), e a cada teste propicia uma amostra dessa quantidade para cada estado visitado (RUSSELL e NORVIG, 2010).
  + **Resultados menos prováveis** acontecerão com **menos frequência**. Afetarão menos nossas estimativas, o que significa que não precisamos conhecer um modelo de transição para que isso funcione. Acabará convergindo para as verdadeiras utilidades, mas é lento, porque não tira proveito do processo de decisão de Markov.
  + **Lembrete:** uma vez que calculamos a utilidade de um estado como recompensa, a utilidade de cada estado pode ser escrita estritamente em termos das utilidades de seus vizinhos imediatos.
  + Os valores de utilidade obedecem às equações de Bellman para uma política fixa:

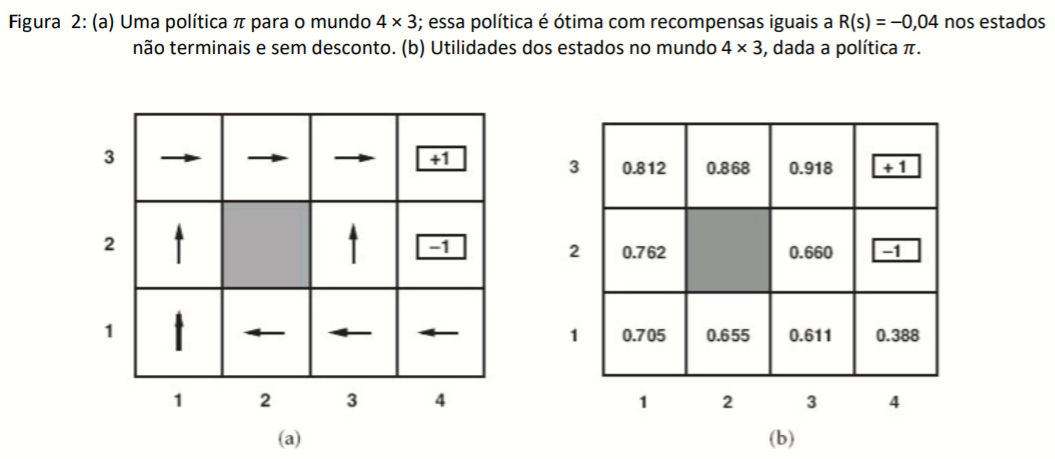


* + Pode-se visualizar a estimativa de utilidade direta como a busca em um espaço de hipóteses 𝑈 muito maior do que precisa ser no sentido de incluir muitas funções que violam as equações de Bellman (RUSSELL e NORVIG, 2010).

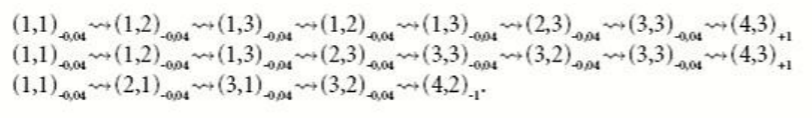
## Exemplo de Aprendizagem por Reforço Passiva

* **Exemplo – Ambiente 4 × 3**
  + Para manter a simplicidade, começaremos com o caso de um agente de aprendizagem passiva que utiliza uma representação baseada em estados em um ambiente completamente observável.
  + Na aprendizagem passiva, a política 𝜋 do agente é fixa: no estado 𝑠, ele sempre executa a ação 𝜋(𝑠).
  + **Meta:** simplesmente aprender o quanto a política é boa, ou seja, aprender a **função utilidade** 𝑈 𝜋 (𝑠).





* O agente executa um conjunto de **experiências** no ambiente usando sua política π. Em cada experiência, o agente começa no estado (1,1) e experimenta uma sequência de transições de estados até alcançar um dos estados terminais, (4,2) ou (4,3). Suas **percepções** fornecem tanto o **estado atual** quanto a **recompensa recebida** nesse estado. Experimentos típicos seriam:



* Observe que cada percepção de estado tem como subscrito a recompensa recebida. O objetivo é utilizar as informações sobre recompensas para aprender a utilidade esperada 𝑈 𝜋 (𝑠) associada a cada estado não terminal s.
* A utilidade é definida como a soma esperada de recompensas(descontadas) obtidas se a política π é seguida.



* Onde R(s) é a recompensa para o estado, 𝑆𝑡 (uma variável aleatória) é o estado alcançado no tempo t quando é executada a política π e 𝑆0 = s. Incluiremos um fator de desconto 𝛾 em todas as nossas equações, mas, para o mundo 4 × 3, definiremos 𝛾 = 1.

## Programação Dinâmica Adaptativa

## Aprendizagem por Diferença Temporal

* **Algoritmos de aprendizado por diferença temporal – DT (SUTTON, 1984)**
  + Aprendem **novas estimativas** do valor com **base** em **outras estimativas**.
  + Não exige um modelo exato do sistema. Procura estimar valores de utilidade para cada estado do ambiente por recompensas oriundas das transições e de valores de estados sucessivos.
  + A **aprendizagem** ocorre **diretamente** a partir da **experiência**. Não é necessário um modelo completo do ambiente. Atualizar as estimativas da função valor a partir de outras estimativas já aprendidas em estados sucessivos. Não é preciso alcançar o estado final de um episódio antes da sua atualização.
  + Avaliação de uma política é encarada como um problema de predição. Estima-se a função valorestado sob a política.
* **Vantagens:** Em relação aos métodos apresentados anteriormente:
  + não exige o modelo MDP do ambiente (não exige conhecimento prévio do modelo de transição do ambiente);
  + pode ser implementado de forma totalmente incremental para aplicações on-line (a sua atualização considera apenas o estado seguinte);
  + tem garantida a convergência assintótica para a resposta correta (embora as atualizações da função valor não sejam obtidas a partir dos dados reais, mas de valores aproximados);
  + os métodos DT são mais rápidos na sua convergência para tarefas estocásticas (TSITSIKLIS, 1994).
* **Transições**
  + Usa as transições obtidas para ajustar os valores dos estados observados, de forma a esses concordarem com as equações de restrições (que definem o ambiente).
  + A taxa de aprendizagem 𝛼 (assume valores no intervalo [0, 1]) determina a velocidade com que o agente assimila a informação, apresentando-se menor à medida que 𝛼 → 0.
* **Estratégias para aprendizagem por reforço utilizando DT (KAELBLING; LITTMAN; MOORE, 1996)**
  + **Crítico Heurístico Adaptativo (CHA)** (BARTO, SUTTON e ANDERSON, 1983): consiste de dois componentes, um **crítico adaptativo** (CHA) e um de **aprendizagem por reforço** (CAR).
  + O componente de aprendizado por reforço busca agir de forma a maximizar o valor heurístico v, que é calculado pelo crítico. O crítico usa o sinal real externo de reforço r para aprender a avaliar os estados s. Na maioria das implementações, os componentes CHA e CAR operam simultaneamente e apenas para essas implementações há garantia, sob condições adequadas, de convergência para uma política ótima.
* **Estratégias para aprendizagem por reforço utilizando DT (KAELBLING; LITTMAN; MOORE, 1996)**
  + **Q-Learning:** o algoritmo de aprendizagem por reforço mais estudado para os propósitos de controle, possuindo provas de convergência.
  + Q-Learning estima a avaliação do par estado-ação diretamente a partir de sinais externos de reforço utilizando uma política gulosa.

## Comparativo entre as Abordagens PDA e DT

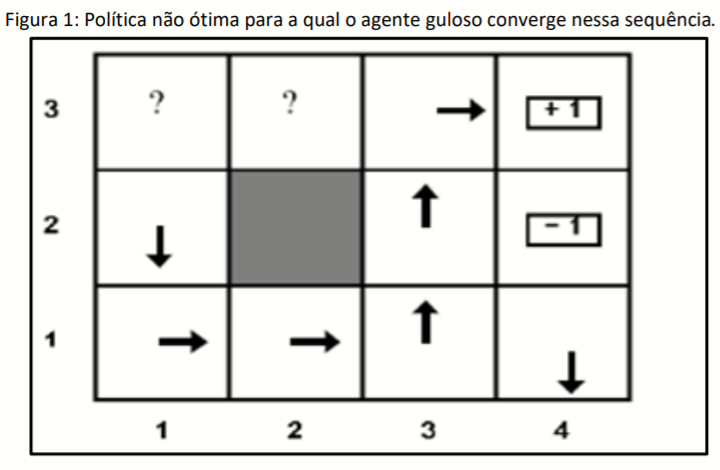
* Ambas estão na realidade estreitamente relacionadas. Tentam fazer ajustes locais para as estimativas de utilidade, a fim de fazer cada estado “concordar” com seus sucessores.
* Uma diferença é que DT ajusta um estado para concordar com seu sucessor.
* Enquanto PDA ajusta o estado para concordar com todos os sucessores que poderiam ocorrer, ponderados por suas probabilidades.
* Uma **diferença mais importante** é que, enquanto DT faz um único ajuste por transição observada, PDA faz tantos quantos necessita para restaurar a consistência entre as estimativas de utilidade U e o modelo de ambiente P.
* Embora a transição observada faça apenas uma mudança local em P, seus efeitos talvez tenham de ser propagados ao longo de U. Desse modo, DT pode ser visualizada como uma primeira aproximação, crua mas eficiente, para PDA.
* É possível estender a abordagem de DT para usar um modelo de ambiente que gere várias pseudoexperiências — transições que o agente de DT talvez imagine que poderiam acontecer, dado seu modelo atual.
* Para cada transição observada, o agente de DT pode gerar grande número de transições imaginárias. Desse modo, as estimativas de utilidade resultantes se aproximarão cada vez mais das estimativas de PDA — é claro, a um preço de um tempo maior de computação.
* De modo semelhante, podemos gerar versões mais eficientes de PDA pela aproximação direta dos algoritmos de iteração de valor ou iteração de política. Mesmo que o algoritmo de iteração de valor seja eficiente, ele é intratável se tivermos, digamos, 10100 estados.
* No entanto, muitos dos ajustes necessários para os valores de estado em cada iteração serão extremamente pequenos.
* Uma abordagem possível para gerar respostas de qualidade razoável com rapidez é **limitar o número de ajustes feitos depois de cada transição observada**. Também poderíamos utilizar uma heurística para ordenar os ajustes possíveis de modo a executar apenas os mais significativos.
* A heurística de **varredura priorizada** prefere fazer ajustes em estados cujos **prováveis** sucessores acabaram de sofrer um **grande** ajuste em suas próprias estimativas de utilidade.
* Usando heurísticas como essa, os algoritmos de PDA aproximada em geral podem aprender quase tão rápido quanto a PDA completa, em termos do número de sequências de treinamento, mas podem ser várias ordens de magnitude mais eficientes em termos de computação.
* Isso lhes permite manipular espaços de estados que são muito maiores para a PDA completa. Os algoritmos de PDA aproximada têm uma vantagem adicional: nas fases iniciais de aprendizagem de um novo ambiente, o modelo de ambiente P com frequência estará longe de ser correto e, assim, haverá pouca razão para calcular uma função utilidade exata que corresponda a esse modelo

## Aprendizagem por Reforço Ativa

* **Conceito**
  + O **agente** tenta localizar uma **boa política de ação**. A cada transição, o agente compreende o estado e obtém o reforço, atualizando sua interpretação do valor nesse estado e escolhendo uma ação a ser executada que mudará o estado do ambiente.
  + Um agente deve escolher quais ações deve executar. **Diferentemente da aprendizagem passiva** onde é a política fixa que estabelece sua conduta. Torna um agente de reforço ativo mais poderoso, mas adiciona complicações quando tentamos formalizar seu processo em um algoritmo.
  + Não só é preciso escolher a ação em cada iteração, mas também é necessário garantir que nossas escolhas garantam que possamos encontrar uma política ótima.
  + O agente precisa aprender um modelo completo com probabilidades de resultados para todas as ações, em vez de aprender apenas o modelo para a política fixa.
  + O mecanismo de aprendizagem do algoritmo AGENTE-PDA-PASSIVO funciona bem para isso, desde que se leve em conta o fato de que o agente tem uma escolha de ações.
  + As utilidades que o agente precisa aprender são definidas pela política ótima, obedecendo-as à equação de Bellman (RUSSELL e NORVIG, 2010).
* **Cenário**: Considerando esta equação no contexto de tentar aprender uma política ótima, pode fazer sentido escolher a melhor ação que conhecemos cada vez, e usar o resultado dessa ação para atualizar a utilidade do estado s.
* **Exemplo:** se sua refeição favorita desde criança é frango e macarrão com queijo, então comer esta refeição quando você está com fome é a escolha que maximiza sua recompensa (ela é sua comida favorita!). Porém, se você sempre escolhe esta ação (que maximiza a sua recompensa), não tem a chance de descobrir melhores opções de comida, caso elas existam. Então, para encontrar uma política ótima, temos que arriscar alguns resultados ruins pela possibilidade de que algumas ações possam levar a melhores resultados do que já conhecemos. Isso nos dá o problema de como equilibrar a **exploitação** (fazendo escolhas que sabemos que vai nos ajudar) contra a **exploração** (descobrir quão valiosas são as escolhas que ainda não fizemos).
* **Aprendizagem ativa x aprendizagem passiva:** A grande diferença dessas aprendizagens é a **exploração**, em que um agente deve experimentar tanto quanto possível do seu ambiente, com o objetivo de aprender a se comportar nele.

## Exploração

* **Um agente de PDA e o modo como ele deve ser modificado para decidir quais ações executar**
  + Um agente PDA que segue a recomendação da política ótima para o modelo aprendido em cada etapa.
  + Podemos verificar que o agente não aprende as utilidades verdadeiras ou a política ótima verdadeira. Ao contrário, ele encontra uma política que alcança a recompensa +1 ao longo da rota inferior.



* **Exploração na aprendizagem ativa**
  + Após experiências com pequenas variações, o agente se fixa nessa política. Dessa forma, ele nunca aprende as utilidades dos outros estados e assim, nunca encontra a rota ótima via (1, 2), (1, 3) e (2, 3).
  + A esse agente é dado o nome de agente guloso. Raramente converge para a política ótima correspondente a esse ambiente.
  + As vezes converge para políticas realmente ruins. As ações fornecem recompensas de acordo com o modelo atual aprendido. Fazem bem mais que isso, elas também contribuem para a aprendizagem do modelo verdadeiro afetando as percepções que são recebidas.
  + Faz com que haja um melhoramento do modelo, fazendo com que o agente receba recompensas melhores no futuro.
  + Para isso ocorrer: um agente deve assumir um compromisso entre **exploitação** (aproveitamento) para maximizar a sua recompensa (que reflete em suas estimativas de utilidades atuais) e **exploração**, a fim de maximizar seu bem-estar a longo prazo.
  + Usar apenas a **exploitação**, corre o risco de ficar paralisado num único lugar. A exploração sozinha para melhorar o conhecimento é inútil, se não colocar em prática o conhecimento.
  + Os agentes devem então, explorar no começo e aproveitar no final.
* **Problema de decisão de um agente**
  + Pode ser visto como o problema de selecionar uma determinada ação quando ele está em um determinado estado.
  + Exemplo conhecido de um problema de decisão envolvendo um único agente é uma variação do problema do “**bandido de k braços**” ou k-armed bandit (SUTTON e BARTO, 1998), nomeado em analogia a uma hipotética máquina de caça níqueis com k alavancas.
  + Esse problema consiste em um agente que pode escolher dentre k opções de ação, cada uma com uma probabilidade diferente e imutável de ser a escolha correta.
  + Cada vez que o agente escolhe a opção correta, esta recebe um sinal de recompensa positivo, nos demais casos recebe um sinal de recompensa negativo.
  + Conforme o agente explora estas opções ao longo de múltiplas iterações, tende a adequar sua política de tomada de decisões para favorecer a opção com maior probabilidade de resultar em recompensa positiva.
  + Problemas como esse são extremamente difíceis de resolver com exatidão para se obter um método de exploração ótimo.
  + No entanto, eles apresentam um esquema razoável que eventualmente leva a um comportamento ótimo do agente.
* **Ações gulosas**
  + Se mantivermos estimativas dos valores das ações, a qualquer momento haverá pelo menos uma ação cujo valor estimado seja maior.
  + Chamamos essas ações de **ações gulosas**. Quando selecionamos uma dessas ações, dizemos que estamos explorando seu conhecimento atual dos valores das ações. Ao contrário, selecionarmos uma ação não gulosa dizemos que estamos explorando, porque isso permite o aumento da estimativa de valor da ação não gulosa.
  + **Exploitação** é o certo a se fazer para maximizar a recompensa esperada em um único passo.
  + A exploração pode produzir a maior recompensa total a longo prazo.
* **Exemplo**
  + Suponha que o valor de uma ação gulosa seja conhecido com certeza, enquanto várias outras ações são estimadas como quase tão boas, mas com uma incerteza substancial.
  + A incerteza é tal que pelo menos uma dessas ações provavelmente é melhor do que a ação gulosa, mas não se sabe qual.
  + Se tivermos muitos passos à frente para fazer seleções de ações, talvez seja melhor explorar as ações não gulosas e descobrir quais delas são melhores do que a ação gulosa.
  + A recompensa é menor a curto prazo, durante a exploração. Maior a longo prazo, porque depois de ter descoberto as melhores ações, podemos explorá-las mais vezes. Como não é possível realizar exploração e exploitação com uma única seleção de ação, geralmente nos referimos ao conflito entre exploração e exploitação (SUTTON e BARTO, 2018).
* **Função de exploração**
  + Escolher se é melhor explorar ou exploitar depende, de maneira complexa, dos valores precisos das estimativas, das incertezas e do número de etapas restantes.
  + Existem muitos métodos sofisticados para equilibrar exploração e exploitação.
  + A estratégia mais fácil para atingir esse equilíbrio é simplesmente tomar uma ação aleatória algumas vezes.
  + Uma maneira um pouco sofisticada é definir uma **função de exploração** 𝑓(𝑢, 𝑛), dessa forma, influenciará no valor percebido das ações que determinam sua entrada.
  + Podemos apenas escolher a melhor ação; mas agora, os estados menos visitados parecerão melhores do que realmente são.
  + A função de exploração 𝑓(𝑢, 𝑛) toma como entradas a utilidade atual conhecida de um estado (𝑢) e o número de vezes que esse estado foi visitado (𝑛). A função de exploração precisa estar aumentando em 𝑢 e diminuindo em n.

## Aprendizagem de uma Função de Ação-valor

* **Função valor x função de ação-valor**
  + **Função valor:** associa o valor diretamente a estados do ambiente (visto na Unidade II). O valor de uma ação indiretamente definido como valor do estado aceitável apenas para agente com modelo efetivo do ambiente (prevendo o estado resultante da execução de cada ação). Aplicável apenas para agente com modelo efetivo do ambiente (para prever o estado resultado da execução de cada ação.
  + **Função de ação-valor:** associa o valor diretamente a pares (estado, ação). Aplicável ao agente sem modelo efetivo do ambiente, quando não há nenhum modelo de ambiente acessível e com o modelo apenas perceptivo de ambiente inacessível.
* **Aprendizagem de uma função de ação-valor**
  + Anteriormente, construímos um agente de PDA ativo, agora vamos considerar como construir um agente de aprendizagem ativo de diferença temporal (DT).
  + A principal diferença em relação ao caso passivo é que o agente não está mais equipado com uma política fixa.
  + Se for aprender uma função utilidade 𝑈, necessitará compreender um modelo apto a escolher uma ação que se baseia em 𝑈 por meio da observação prévia de um passo (RUSSELL e NORVIG, 2010).
  + O problema de aquisição de modelo para o agente DT é idêntico ao do agente de PDA, sendo a mesma equação de atualização do DT utilizada para este novo agente. Possível pela seguinte razão:
    - Considere que o agente realize um passo que comumente leve a um bom destino, mas devido ao não determinismo do ambiente, o agente acaba em um estado catastrófico.
    - A regra de atualização DT levará a circunstância tão a sério de forma que seu resultado se torna o resultado normal da ação.
    - O resultado improvável ocorrerá com pouca frequência em um grande conjunto de sequências de treinamento.
  + A **longo prazo** seus **efeitos** serão **ponderados** de forma proporcional a sua probabilidade.
  + Mostra que o algoritmo de DT convergirá para os mesmos valores que a PDA, à medida que o número de sequências de treinamento tender a infinito (RUSSELL e NORVIG, 2010).
  + A **aplicação** do **método da diferença temporal** na **função-valor da ação permite** que o **ambiente** seja **desconhecido**, e leva a um dos métodos mais difundidos da aprendizagem por reforço, o Qlearning (será visto em detalhes na Unidade IV).

# Unidade 04 - Q-Learning

Essa unidade tem como objetivo:

* Descrever aprendizagem Q;
* Apresentar o algoritmo Q-Learning;
* Exemplificar aplicações Q-Learning.

## Introdução à Aprendizagem Q

* A **aprendizagem por reforço** ajuda a ajustar nossas ações físicas e habilidades motoras.
* **Ações** que um organismo executa resultam em um feedback, traduzido em **recompensa** positiva ou negativa por essa ação.
* **Exemplo**: um bebê aprendendo a andar
  + Cair e sentir um pouco de dor. Feedback **negativo** ajudará o bebê a aprender o que não fazer;
  + Capaz de ficar em pé por uma questão de tempo, fazendo algo certo. Feedback **positivo**;
  + Continua tentando andar, desenvolverá coordenação motora de forma que a **recompensa** será **maximizada**. Dor, fome, sede e prazer são alguns exemplos de reforços naturais.
* **Ações** podem resultar em **recompensas imediatas** ou uma cadeia mais longa de **ações** que levam à **recompensa.**

**Q-Learning**

* É um tipo específico de aprendizagem por reforço que atribui **valores a pares de estado-ação**.
* O estado do organismo é uma soma de todos os seus dados sensoriais, incluindo sua posição corporal, sua localização no ambiente, a atividade neural em sua cabeça etc.
* Em Q-Learning: para cada estado há um **número de ações possíveis** que poderiam ser tomadas, cada **ação dentro de cada estado** tem um **valor** de acordo com quantas ou quão poucas recompensas o organismo obterá por completar aquela ação.
* Duas formas básicas de aprendizagem Q: **aprendizagem Q ótima** e **aprendizagem Q baseada em política**.

## Aprendizagem Q

* Na aprendizagem Q (Q-Learning), o **objetivo é atingir o estado com a maior recompensa**, de modo que se o agente chega ao objetivo, ele permanecerá lá para sempre.
  + Esse tipo de meta é chamado de “absorbing goal” (objetivo absorvente).
* Matriz similar Q: memória do que o agente aprendeu pela experiência
* As linhas representam o estado atual do agente e as colunas representam as possíveis ações que levam ao próximo estado.
* A regra de transição da aprendizagem Q é uma fórmula muito simples:
  + **Q(estado, ação) = R(estado, ação) + Gama \* Max [Q(próximo estado, todas as ações)]**

## Função Q

* A função 𝑄, ou função ação-valor

## Aprendizagem Q Ótima

* A aprendizagem 𝑸 é um método de política-off. A função valor ótima é estimada, independentemente da política atual (exploração) que está sendo utilizada para gerar as trajetórias amostradas.
* Aprendizagem 𝑄 converge com probabilidade 1 para o valor ótimo 𝑄 ∗ quando 𝑡 → ∞

## Aprendizagem Baseada em Política

* A aprendizagem 𝑸 baseada em política tenta aprender, a função 𝑄 para alguma política projetada.
* A política pode ser ou não a política que de fato está sendo seguida durante o treinamento.

## Algoritmo Q-Learning

* O algoritmo é empregado pelo agente para aprender com a experiência
* Cada episódio é equivalente a uma sessão de treinamento.
* Cada sessão de treinamento, o agente explora o ambiente recebe a recompensa até atingir o estado objetivo.
* O objetivo do treinamento é melhorar o “cérebro” do nosso agente, representado pela matriz Q.
* Mais treinamento resulta em uma matriz mais otimizada Q. Matriz Q aprimorada, o agente encontrará a rota mais rápida para o estado objetivo.
* O gama ([0,1]) mais próximo de zero, o agente tenderá a considerar recompensas imediatas. Gama mais próximo de um, o agente considerará recompensas futuras com maior peso, disposto a atrasar a recompensa.

**Passos do algoritmo para utilizar a matriz Q**

* O agente rastreia a sequência de estados, do estado inicial ao estado objetivo. O algoritmo localiza as ações com os maiores valores de recompensa registrados na matriz Q para o estado atual.
  + 1. defina o estado atual = estado inicial;
  + 2. a partir do estado atual, localize a ação com maior valor Q;
  + 3. defina o estado atual = próximo estado;
  + 4. repita as etapas 2 e 3 até o próximo estado atual = estado objetivo.
  + 5. O algoritmo acima retornará a sequência de estados do estado inicial para o estado objetivo.
  + O algoritmo acima retornará a sequência de estados do estado inicial para o estado objetivo.

## Detalhes do Algoritmo Q-Learning

**Políticas de seleção de ações para o algoritmo Q-Learning**

* Uma política de seleção de ações tem como objetivo estabelecer o comportamento do agente aprendiz para que ele alterne adequadamente entre o uso do conhecimento já adquirido e a aquisição de novo conhecimento.
* Otimizar o processo de exploração/exploitação do espaço de busca.
* A ideia de experimentar mais de uma política de seleção de ações para o Q-Learning, tem como meta verificar qual dessas políticas é mais adequada para ser utilizada na implementação dos métodos híbridos propostos.

**Política 𝜺 – gulosa**

* Escolhe a ação que tem o maior valor esperado, com probabilidade definida por (1 − 𝜀), e de ação aleatória, com probabilidade 𝜀.

**Política ε - gulosa adaptativa**

* Semelhante à política gulosa descrita anteriormente. Permite escolher a ação que tem o maior valor esperado, com probabilidade definida por (1 - ε ), e ação aleatória, com probabilidade.
* Diferença e justifica o termo “adaptativa” é que o valor sofre um decaimento exponencial.
* A ideia é permitir que sejam feitas escolhas mais aleatórias e, à medida que o número de episódios aumente, o espectro guloso seja mais explorado.

**Política baseada na contagem de visitas**

* A escolha de ações é feita baseada em uma técnica denominada Comparação de Reforço (Reinforcement Comparison).
* Princípio de que ações seguidas de grandes recompensas devem ser preferidas em detrimento de ações seguidas de pequenas recompensas.
* “Grande recompensa” compara com um nível de recompensa padrão denominado recompensa referencial.
* A ideia semelhante à técnica de comparação de reforço. A escolha das ações preferidas é feita com base na contagem de visitas aos estados atingidos por tais ações.

## Q-Learning

**Q-LEARNING 𝜆 (ELIGIBILITY TRACES)**

* Eligibility traces (ou traços de elegibilidade) apresentado por Klopf (1972) no âmbito de sistemas adaptativos para a USAF (United States Air Force).
* Em aprendizagem por reforço somente foi realizada em 1989 por Watkins (1989) ao incorporar este no mecanismo de aprendizagem Q-Learning.
* **Elegibility traces** consiste no registo temporário da ocorrência de um determinado evento, quer seja a passagem por um estado ou a seleção de uma determinada ação. Registro permite a sinalização de quaisquer parâmetros associados ao evento como elegíveis para posteriormente poderem sofrer alterações na sua definição. Na ocorrência de um erro, este poderá ter a sua origem relacionada com os estados e/ou ações sinalizadas.

## Aplicações Q-Learning