Aula 27 - Aprendizado por Reforço

João B. Florindo

Instituto de Matemática, Estatística e Computação Científica
Universidade Estadual de Campinas - Brasil
jbflorindo@ime.unicamp.br

Outline

Aprendizado por reforço

- Usado em problemas recentes como o AlphaGo.

Aprendizado por reforço

- Usado em problemas recentes como o AlphaGo.
- Sem treinamento (explícito).

Aprendizado por reforço

- Usado em problemas recentes como o AlphaGo.
- Sem treinamento (explícito).
- Agente "aprende" por tentativa e erro, visando maximizar a recompensa (p.ex. a pontuação em um jogo).

Aprendizado por reforço

- Usado em problemas recentes como o AlphaGo.
- Sem treinamento (explícito).
- Agente "aprende" por tentativa e erro, visando maximizar a recompensa (p.ex. a pontuação em um jogo).
- Baseado em estados, ações e recompensas.

Aprendizado por reforço

- Usado em problemas recentes como o AlphaGo.
- Sem treinamento (explícito).
- Agente "aprende" por tentativa e erro, visando maximizar a recompensa (p.ex. a pontuação em um jogo).
- Baseado em estados, ações e recompensas.
- Meio termo entre aprendizado supervisionado e não supervisionado.

Aprendizado por reforço

- Imagine um jogo como PacMan ou SuperMario.

Aprendizado por reforço

- Imagine um jogo como PacMan ou SuperMario.
- O personagem é um **agente**.

Aprendizado por reforço

- Imagine um jogo como PacMan ou SuperMario.
- O personagem é um **agente**.
- O jogo em si é o **ambiente**.

Aprendizado por reforço

- Imagine um jogo como PacMan ou SuperMario.
- O personagem é um **agente**.
- O jogo em si é o **ambiente**.
- Cada *frame* do jogo no instante t é um **estado $\mathbf{s}_t$**.

Aprendizado por reforço

- Imagine um jogo como PacMan ou SuperMario.
- O personagem é um **agente**.
- O jogo em si é o **ambiente**.
- Cada *frame* do jogo no instante t é um **estado $\mathbf{s}_t$**.
- Os movimentos (direita, para frente, saltar, etc.) são **ações $\mathbf{a}_t$**.

Aprendizado por reforço

- Imagine um jogo como PacMan ou SuperMario.
- O personagem é um **agente**.
- O jogo em si é o **ambiente**.
- Cada *frame* do jogo no instante t é um **estado** $\mathbf{s}_t$.
- Os movimentos (direita, para frente, saltar, etc.) são **ações** $\mathbf{a}_t$.
- Cada doce/fruta/moeda coletados ou o simples fato de continuar vivo é uma **recompensa** $r(\mathbf{s}_t, \mathbf{a}_t)$.

Aprendizado por reforço

Loop geral:

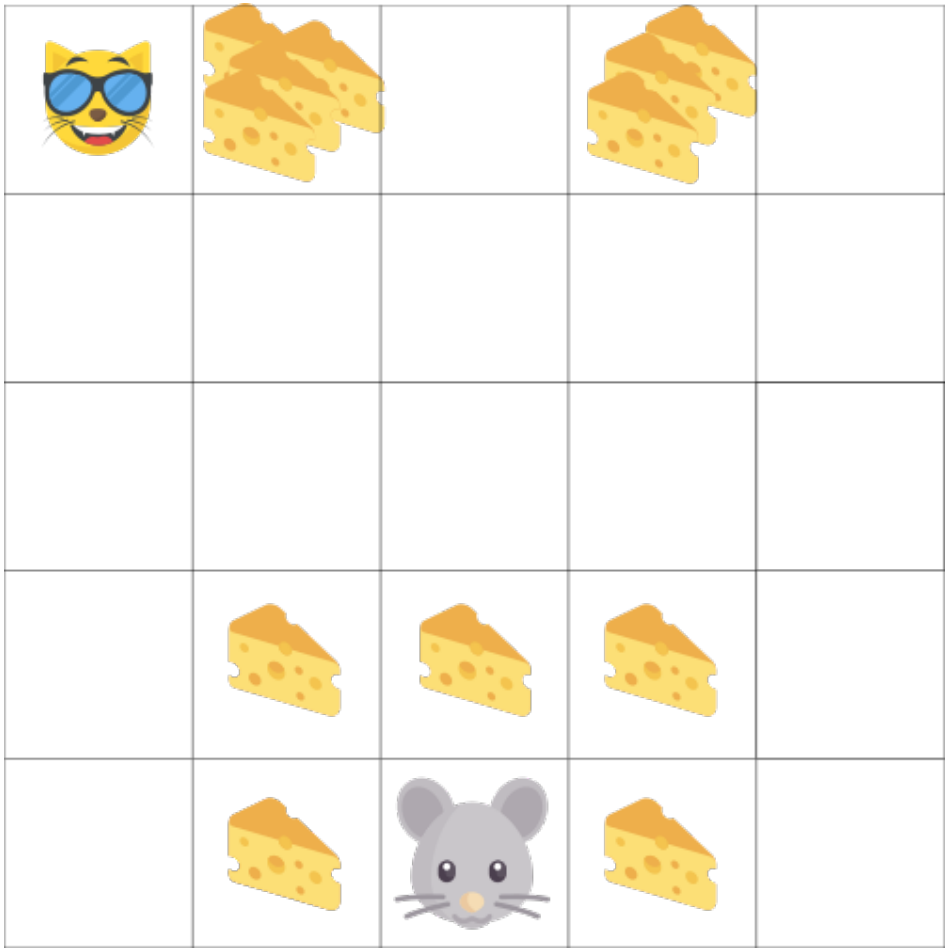
1. Parte do estado $\mathbf{s}_0$ recebido do ambiente
2. Agente faz ação $\mathbf{a}_0$ com base em $\mathbf{s}_0$
3. Ambiente muda para $\mathbf{s}_1$
4. Ambiente recebe recompensa $r(\mathbf{s}_1, \mathbf{a}_1)$

Baseado na "hipótese" da recompensa: maximizar a recompensa acumulada esperada:

$$G_t = r(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) + r(\mathbf{s}_{t+2}, \mathbf{a}_{t+2}) + \cdots = \sum_{k=0}^{T} r(\mathbf{s}_{t+k+1}, \mathbf{a}_{t+k+1}).$$

Porém recompensas no início do jogo são mais prováveis. Veja a seguir.

Aprendizado por reforço



Aprendizado por reforço

- O rato é o agente.

Aprendizado por reforço

- O rato é o agente.
- Objetivo é comer o máximo de queijo sem ser comido pelo gato!

Aprendizado por reforço

- O rato é o agente.
- Objetivo é comer o máximo de queijo sem ser comido pelo gato!
- Recompensa para estados próximos ao gato são descontados.

Aprendizado por reforço

- O rato é o agente.
- Objetivo é comer o máximo de queijo sem ser comido pelo gato!
- Recompensa para estados próximos ao gato são descontados.
- Taxa de desconto $\gamma \in [0, 1]$. Nova recompensa acumulada:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r(\mathbf{s}_{t+k+1}, \mathbf{a}_{t+k+1}).$$

Aprendizado por reforço

- O rato é o agente.
- Objetivo é comer o máximo de queijo sem ser comido pelo gato!
- Recompensa para estados próximos ao gato são descontados.
- Taxa de desconto $\gamma \in [0, 1]$. Nova recompensa acumulada:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r(\mathbf{s}_{t+k+1}, \mathbf{a}_{t+k+1}).$$

- Com o passar do tempo, o rato fica mais perto do gato e a recompensa se torna exponencialmente menos provável.

Aprendizado por reforço

- Conceito de **tarefa**: conjunto de recompensas, ações, estados e transições em uma janela de tempo.

Aprendizado por reforço

- Conceito de **tarefa**: conjunto de recompensas, ações, estados e transições em uma janela de tempo.
- **Tarefa episódica**: Possui começo e fim. Ex.: entre o início e o fim de um nível no jogo ou até que o personagem morra (episódio).

Aprendizado por reforço

- Conceito de **tarefa**: conjunto de recompensas, ações, estados e transições em uma janela de tempo.
- **Tarefa episódica**: Possui começo e fim. Ex.: entre o início e o fim de um nível no jogo ou até que o personagem morra (episódio).
- **Tarefa contínua**: agente não para de rodar até que alguém interrompa. Ex.: Sistema automático de compra e venda ações.

Aprendizado por reforço

- **Abordagem baseada na política**: encontrar uma função π chamada de **política** ótima. A política determina a ação **a** partindo do estado **s**:

$$\mathbf{a} = \pi(\mathbf{s}).$$

Aprendizado por reforço

- **Abordagem baseada na política**: encontrar uma função π chamada de **política** ótima. A política determina a ação **a** partindo do estado **s**:

$$\mathbf{a} = \pi(\mathbf{s}).$$

- Pode ser determinística (depende apenas do estado **s**) ou estocástica (ação segue distribuição de probabilidade condicionada ao estado atual).

Aprendizado por reforço

- **Abordagem baseada na política**: encontrar uma função π chamada de **política** ótima. A política determina a ação **a** partindo do estado **s**:

$$\mathbf{a} = \pi(\mathbf{s}).$$

- Pode ser determinística (depende apenas do estado **s**) ou estocástica (ação segue distribuição de probabilidade condicionada ao estado atual).

- **Abordagem baseada em valor**: Maximizar recompensa acumulada $V(S)$.

Aprendizado por reforço

- **Abordagem baseada na política**: encontrar uma função π chamada de **política** ótima. A política determina a ação **a** partindo do estado **s**:

$$\mathbf{a} = \pi(\mathbf{s}).$$

- Pode ser determinística (depende apenas do estado **s**) ou estocástica (ação segue distribuição de probabilidade condicionada ao estado atual).

- **Abordagem baseada em valor**: Maximizar recompensa acumulada $V(S)$.

- Depende da política π. Agente escolhe em cada passo o estado com maior valor.

Aprendizado por reforço

- **Abordagem baseada em modelo**: encontra um modelo geral para o ambiente.

Aprendizado por reforço

- **Abordagem baseada em modelo**: encontra um modelo geral para o ambiente.
- Mais complicada de todas pois cada ambiente pode ter um modelo próprio.

Aprendizado por reforço

- Objetivo é encontrar melhor ação para cada estado.

Aprendizado por reforço

- Objetivo é encontrar melhor ação para cada estado.
- Na prática, usa-se um processo de decisão de Markov.

Aprendizado por reforço

- Objetivo é encontrar melhor ação para cada estado.
- Na prática, usa-se um processo de decisão de Markov.
- Tabela Q mapeia cada par estado-ação para a recompensa mais provável dada pelo ambiente.

Aprendizado por reforço

- Objetivo é encontrar melhor ação para cada estado.
- Na prática, usa-se um processo de decisão de Markov.
- Tabela Q mapeia cada par estado-ação para a recompensa mais provável dada pelo ambiente.
- Inferência a partir de uma distribuição muito complicada (Monte Carlo).

Outline

- Agente deve aprender uma **política** $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$: distribuição sobre as ações dado o estado atual e os parâmetros θ.

- Agente deve aprender uma **política** $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$: distribuição sobre as ações dado o estado atual e os parâmetros θ.

- Ambiente representado como um **Processo Decisório de Markov** (MDP).

- Agente deve aprender uma **política** $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$: distribuição sobre as ações dado o estado atual e os parâmetros θ.
- Ambiente representado como um **Processo Decisório de Markov** (MDP).
- Condição de Markov: toda a informação relevante encapsulada no estado atual (independente dos estados passados).

- Agente deve aprender uma **política** $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$: distribuição sobre as ações dado o estado atual e os parâmetros θ.
- Ambiente representado como um **Processo Decisório de Markov** (MDP).
- Condição de Markov: toda a informação relevante encapsulada no estado atual (independente dos estados passados).
- Componentes de um MDP:

- Agente deve aprender uma **política** $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$: distribuição sobre as ações dado o estado atual e os parâmetros θ.
- Ambiente representado como um **Processo Decisório de Markov** (MDP).
- Condição de Markov: toda a informação relevante encapsulada no estado atual (independente dos estados passados).
- Componentes de um MDP:
  - Distribuição de estados iniciais $p(\mathbf{s}_0)$

- Agente deve aprender uma **política** $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$: distribuição sobre as ações dado o estado atual e os parâmetros θ.
- Ambiente representado como um **Processo Decisório de Markov** (MDP).
- Condição de Markov: toda a informação relevante encapsulada no estado atual (independente dos estados passados).
- Componentes de um MDP:
  - Distribuição de estados iniciais $p(\mathbf{s}_0)$
  - Política $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$

- Agente deve aprender uma **política** $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$: distribuição sobre as ações dado o estado atual e os parâmetros θ.
- Ambiente representado como um **Processo Decisório de Markov** (MDP).
- Condição de Markov: toda a informação relevante encapsulada no estado atual (independente dos estados passados).
- Componentes de um MDP:
  - Distribuição de estados iniciais $p(\mathbf{s}_0)$
  - Política $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$
  - Distribuição de transições $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$

- Agente deve aprender uma **política** $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$: distribuição sobre as ações dado o estado atual e os parâmetros θ.
- Ambiente representado como um **Processo Decisório de Markov** (MDP).
- Condição de Markov: toda a informação relevante encapsulada no estado atual (independente dos estados passados).
- Componentes de um MDP:
  - Distribuição de estados iniciais $p(\mathbf{s}_0)$
  - Política $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$
  - Distribuição de transições $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$
  - Função de recompensa $r(\mathbf{s}_t, \mathbf{a}_t)$

- Assume ambiente totalmente observável, i.e., $\mathbf{s}_t$ observável diretamente.

- Assume ambiente totalmente observável, i.e., $\mathbf{s}_t$ observável diretamente.
- Horizonte finito T.

- Assume ambiente totalmente observável, i.e., $\mathbf{s}_t$ observável diretamente.
- Horizonte finito T.
- Trajetória $\tau = (\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \cdots, \mathbf{s}_T, \mathbf{a}_T)$

- Assume ambiente totalmente observável, i.e., $\mathbf{s}_t$ observável diretamente.
- Horizonte finito T.
- Trajetória $\tau = (\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \cdots, \mathbf{s}_T, \mathbf{a}_T)$
- Probabilidade de uma trajetória;

$$p(\tau) = p(\mathbf{s}_0)\pi_\theta(\mathbf{a}_0|\mathbf{s}_0)p(\mathbf{s}_1|\mathbf{s}_0, \mathbf{a}_0) \cdots p(\mathbf{s}_T|\mathbf{s}_{T-1}, \mathbf{a}_{T-1})\pi_\theta(\mathbf{a}_T|\mathbf{s}_T)$$

- Recompensa para uma trajetória:

$$r(\tau) = \sum_{t=0}^{T} r(\mathbf{s}_t, \mathbf{a}_t).$$

- Recompensa para uma trajetória:

$$r(\tau) = \sum_{t=0}^{T} r(\mathbf{s}_t, \mathbf{a}_t).$$

- Objetivo: maximizar a recompensa esperada $R = \mathbb{E}_{p(\tau)}[r(\tau)]$.

- Recompensa para uma trajetória:

$$r(\tau) = \sum_{t=0}^{T} r(\mathbf{s}_t, \mathbf{a}_t).$$

- Objetivo: maximizar a recompensa esperada $R = \mathbb{E}_{p(\tau)}[r(\tau)]$.

- Tentativa e erro: amostrar uma trajetória aleatoriamente, se retorna recompensa alta torná-la mais provável, se não, torná-la menos provável.

- Recompensa para uma trajetória:

$$r(\tau) = \sum_{t=0}^{T} r(\mathbf{s}_t, \mathbf{a}_t).$$

- Objetivo: maximizar a recompensa esperada $R = \mathbb{E}_{p(\tau)}[r(\tau)]$.

- Tentativa e erro: amostrar uma trajetória aleatoriamente, se retorna recompensa alta torná-la mais provável, se não, torná-la menos provável.

- Similar a um gradiente ascendente em R.

Algoritmo:

Repita:

Amostre uma trajetória $\tau = (\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \cdots, \mathbf{s}_T, \mathbf{a}_T)$

$r(\tau) \leftarrow \sum_{k=0}^{T} r(\mathbf{s}_k, \mathbf{a}_k)$

For $t = 0, \cdots, T$:

$\theta \leftarrow \theta + \alpha r(\tau) \frac{\partial}{\partial \theta} \log \pi_\theta(\mathbf{a}_k | \mathbf{s}_k)$

Ou ainda, considerando que ações devem ser reforçadas com base apenas em recompensas futuras:

Repita:

Amostre uma trajetória $\tau = (\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \cdots, \mathbf{s}_T, \mathbf{a}_T)$

For $t = 0, \cdots, T$:

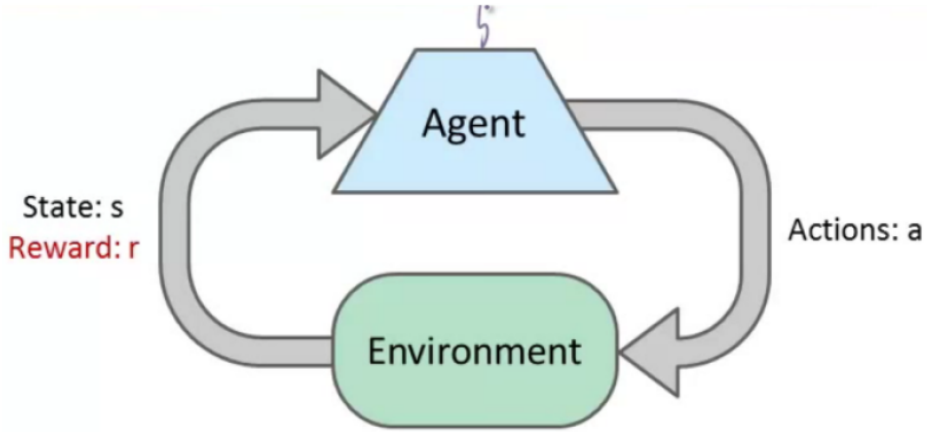
$r_t(\tau) \leftarrow \sum_{k=t}^{T} r(\mathbf{s}_k, \mathbf{a}_k)$

$\theta \leftarrow \theta + \alpha r_t(\tau) \frac{\partial}{\partial \theta} \log \pi_\theta(\mathbf{a}_k | \mathbf{s}_k)$

Outline

- Aprende uma função ação-valor que prevê recompensas futuras.

- Aprende uma função ação-valor que prevê recompensas futuras.
- Ambiente tratado como caixa preta: sem acesso às probabilidades de transição, distribuição de recompensas, etc.



- Ambiente representado como um **Processo Decisório de Markov** (MDP).

- Ambiente representado como um **Processo Decisório de Markov** (MDP).

- Condição de Markov: toda a informação relevante encapsulada no estado atual (independente dos estados passados).

- Ambiente representado como um **Processo Decisório de Markov** (MDP).
- Condição de Markov: toda a informação relevante encapsulada no estado atual (independente dos estados passados).
- Componentes de um MDP:

- Ambiente representado como um **Processo Decisório de Markov** (MDP).
- Condição de Markov: toda a informação relevante encapsulada no estado atual (independente dos estados passados).
- Componentes de um MDP:
  - Distribuição de estados iniciais $p(\mathbf{s}_0)$

- Ambiente representado como um **Processo Decisório de Markov** (MDP).

- Condição de Markov: toda a informação relevante encapsulada no estado atual (independente dos estados passados).

- Componentes de um MDP:
  - Distribuição de estados iniciais $p(\mathbf{s}_0)$
  - Política $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$

- Ambiente representado como um **Processo Decisório de Markov** (MDP).

- Condição de Markov: toda a informação relevante encapsulada no estado atual (independente dos estados passados).

- Componentes de um MDP:
  - Distribuição de estados iniciais $p(\mathbf{s}_0)$
  - Política $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$
  - Distribuição de transições $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$

- Ambiente representado como um **Processo Decisório de Markov** (MDP).

- Condição de Markov: toda a informação relevante encapsulada no estado atual (independente dos estados passados).

- Componentes de um MDP:
  - Distribuição de estados iniciais $p(\mathbf{s}_0)$
  - Política $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$
  - Distribuição de transições $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$
  - Função de recompensa $r(\mathbf{s}_t, \mathbf{a}_t)$

- Ambiente representado como um **Processo Decisório de Markov** (MDP).

- Condição de Markov: toda a informação relevante encapsulada no estado atual (independente dos estados passados).

- Componentes de um MDP:
  - Distribuição de estados iniciais $p(\mathbf{s}_0)$
  - Política $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$
  - Distribuição de transições $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$
  - Função de recompensa $r(\mathbf{s}_t, \mathbf{a}_t)$

- Assume ambiente totalmente observável, i.e., $\mathbf{s}_t$ observável diretamente.

- PORÉM, agora o horizonte de tempo é INFINITO.

- PORÉM, agora o horizonte de tempo é INFINITO.
- Mas, não podemos somar infinitas recompensas.

- PORÉM, agora o horizonte de tempo é INFINITO.
- Mas, não podemos somar infinitas recompensas.
- Precisamos de um fator de desconto: 100 reais hoje valem mais do que 100 reais daqui a um ano.

- PORÉM, agora o horizonte de tempo é INFINITO.
- Mas, não podemos somar infinitas recompensas.
- Precisamos de um fator de desconto: 100 reais hoje valem mais do que 100 reais daqui a um ano.
- Retorno descontado:

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots$$

em que $\gamma < 1$ é o **fator de desconto**.

- **Função de Valor** $V^{\pi}(\mathbf{s})$: retorno descontado esperado se começamos no estado $\mathbf{s}$ e seguimos a política π:

$$
\begin{aligned}
V^{\pi}(\mathbf{s}) &= \mathbb{E}[G_t | \mathbf{s}_t = \mathbf{s}] \\
&= \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i r_{t+i} | \mathbf{s}_t = \mathbf{s}\right]
\end{aligned}
$$

- **Função de Valor** $V^{\pi}(\mathbf{s})$: retorno descontado esperado se começamos no estado $\mathbf{s}$ e seguimos a política π:

$$V^{\pi}(\mathbf{s}) = \mathbb{E}[G_t|\mathbf{s}_t = \mathbf{s}]$$
$$= \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i r_{t+i}|\mathbf{s}_t = \mathbf{s}\right]$$

- Benefício: permite ver diretamente como uma ação afeta recompensas futuras sem amostrar trajetórias.

- **Função de Valor** $V^{\pi}(\mathbf{s})$: retorno descontado esperado se começamos no estado $\mathbf{s}$ e seguimos a política π:

$$V^{\pi}(\mathbf{s}) = \mathbb{E}[G_t|\mathbf{s}_t = \mathbf{s}]$$
$$= \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i r_{t+i}|\mathbf{s}_t = \mathbf{s}\right]$$

- Benefício: permite ver diretamente como uma ação afeta recompensas futuras sem amostrar trajetórias.

- Função de valor pode ser aprendida.

- Mais precisamente, aprendemos uma função de **ação-valor** (*Q*-function).

- Mais precisamente, aprendemos uma função de **ação-valor** (Q-function).

- Valor esperado do retorno se tomar a ação **a** e seguir sua política:

$$Q^{\pi}(\mathbf{s}, \mathbf{a}) = \mathbb{E}[G_t | \mathbf{s}_t = \mathbf{s}, \mathbf{a}_t = \mathbf{a}].$$

- Mais precisamente, aprendemos uma função de **ação-valor** (Q-function).

- Valor esperado do retorno se tomar a ação **a** e seguir sua política:

$$Q^{\pi}(\mathbf{s}, \mathbf{a}) = \mathbb{E}[G_t | \mathbf{s}_t = \mathbf{s}, \mathbf{a}_t = \mathbf{a}].$$

- Relação com a função de valor:

$$V^{\pi}(\mathbf{s}) = \sum_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) Q^{\pi}(\mathbf{s}, \mathbf{a})$$

- Mais precisamente, aprendemos uma função de **ação-valor** (Q-function).

- Valor esperado do retorno se tomar a ação $\mathbf{a}$ e seguir sua política:

$$Q^{\pi}(\mathbf{s}, \mathbf{a}) = \mathbb{E}[G_t | \mathbf{s}_t = \mathbf{s}, \mathbf{a}_t = \mathbf{a}].$$

- Relação com a função de valor:

$$V^{\pi}(\mathbf{s}) = \sum_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) Q^{\pi}(\mathbf{s}, \mathbf{a})$$

- Ação ótima:

$$\underset{\mathbf{a}}{\operatorname{argmax}}\, Q^{\pi}(\mathbf{s}, \mathbf{a}).$$

- Função de ação-valor é obtida pela fórmula recursiva da **Equação de Bellman**:

$$Q^{\pi}(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{p(s'|\mathbf{s},\mathbf{a})\pi(\mathbf{a}'|\mathbf{s}')}[Q^{\pi}(\mathbf{s}', \mathbf{a}')]$$

- Função de ação-valor é obtida pela fórmula recursiva da **Equação de Bellman**:

$$Q^{\pi}(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{p(s'|\mathbf{s},\mathbf{a})\pi(\mathbf{a}'|\mathbf{s}')}[Q^{\pi}(\mathbf{s}', \mathbf{a}')]$$

- A **política ótima** π^* é aquela que maximiza o retorno esperado.

- Função de ação-valor é obtida pela fórmula recursiva da **Equação de Bellman**:

$$Q^{\pi}(\mathbf{s}, \mathbf{a}) = r(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{p(s'|\mathbf{s},\mathbf{a})\pi(\mathbf{a}'|\mathbf{s}')}[Q^{\pi}(\mathbf{s}', \mathbf{a}')]$$

- A **política ótima** π^* é aquela que maximiza o retorno esperado.

- **Função de ação-valor ótima** Q^* é a função de ação-valor correspondente a π^*.

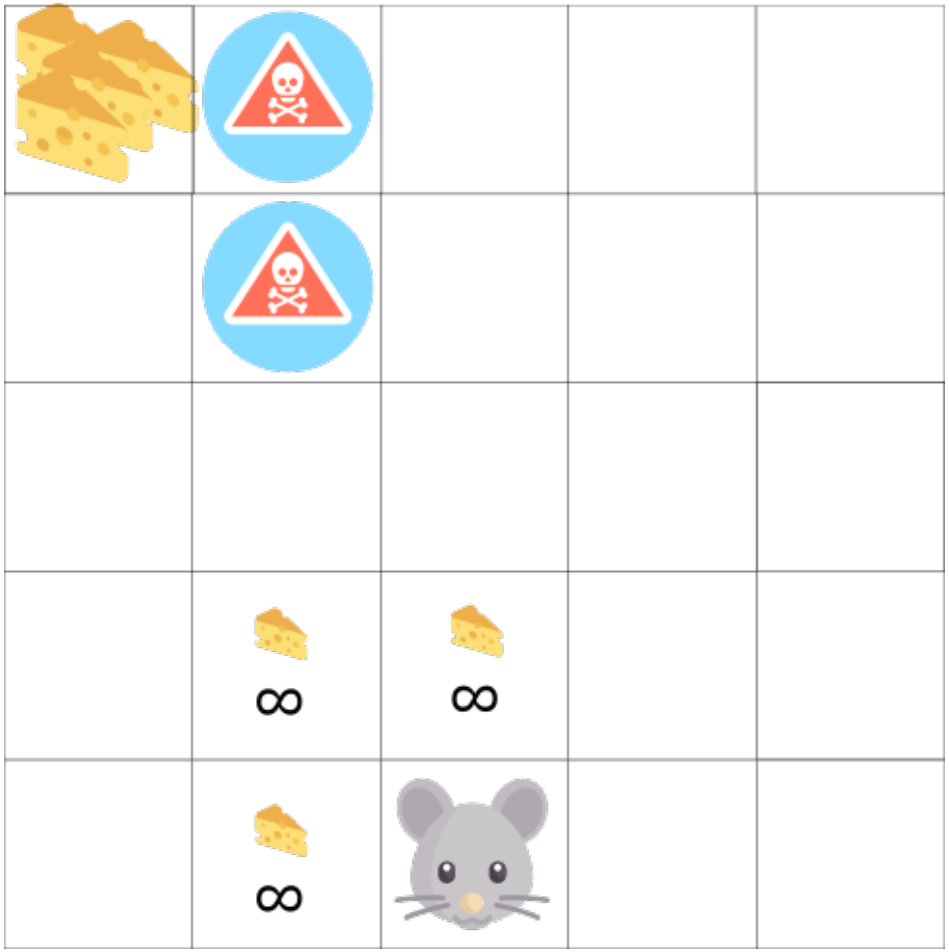
- *Q-learning* é o algoritmo que ajusta Q repetidamente visando se aproximar de Q^*.

- *Q-learning* é o algoritmo que ajusta Q repetidamente visando se aproximar de Q^*.
- Em cada tempo são amostrados estados e ações $(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$. Fazer então

$$Q(\mathbf{s}_t, \mathbf{a}_t) \leftarrow Q(\mathbf{s}_t, \mathbf{a}_t) + \alpha \left[r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \max_{\mathbf{a}} Q(\mathbf{s}_{t+1}, \mathbf{a}) - Q(\mathbf{s}_t, \mathbf{a}_t) \right],$$

em que a expressão entre colchetes é chamada de **erro de Bellman**.

- Um dilema que surge é o "*exploration vs. exploitation*"



- Suponha que o rato tenha infinitos pedaços pequenos de queijo próximo a ele.

- Suponha que o rato tenha infinitos pedaços pequenos de queijo próximo a ele.

- Baseado apenas na recompensa ele ficará preso ali (*exploitation*) e nunca buscará o queijo gigante no topo (*exploration*).

- Suponha que o rato tenha infinitos pedaços pequenos de queijo próximo a ele.
- Baseado apenas na recompensa ele ficará preso ali (*exploitation*) e nunca buscará o queijo gigante no topo (*exploration*).
- Agente pode sair da regra padrão em tempos aleatórios para minimizar este efeito.

- Suponha que o rato tenha infinitos pedaços pequenos de queijo próximo a ele.
- Baseado apenas na recompensa ele ficará preso ali (*exploitation*) e nunca buscará o queijo gigante no topo (*exploration*).
- Agente pode sair da regra padrão em tempos aleatórios para minimizar este efeito.
- Abordagem simples: política ϵ-*greedy*:

- Suponha que o rato tenha infinitos pedaços pequenos de queijo próximo a ele.
- Baseado apenas na recompensa ele ficará preso ali (*exploitation*) e nunca buscará o queijo gigante no topo (*exploration*).
- Agente pode sair da regra padrão em tempos aleatórios para minimizar este efeito.
- Abordagem simples: política ϵ-*greedy*:
  - Com probabilidade $1 - \epsilon$ escolha a ação ótima de acordo com Q

- Suponha que o rato tenha infinitos pedaços pequenos de queijo próximo a ele.
- Baseado apenas na recompensa ele ficará preso ali (*exploitation*) e nunca buscará o queijo gigante no topo (*exploration*).
- Agente pode sair da regra padrão em tempos aleatórios para minimizar este efeito.
- Abordagem simples: política ϵ-*greedy*:
  - Com probabilidade $1 - \epsilon$ escolha a ação ótima de acordo com Q
  - Com probabilidade ϵ escolha uma ação aleatória

- Aprendizado profundo usa redes profundas: *Deep Q-Learning*.

- Aprendizado profundo usa redes profundas: *Deep Q-Learning*.
- Abordagem "*deep*" usa rede para "aprender" estas relações e definir a recompensa e a ação mais adequada em cada estado.

Convolutional Agent