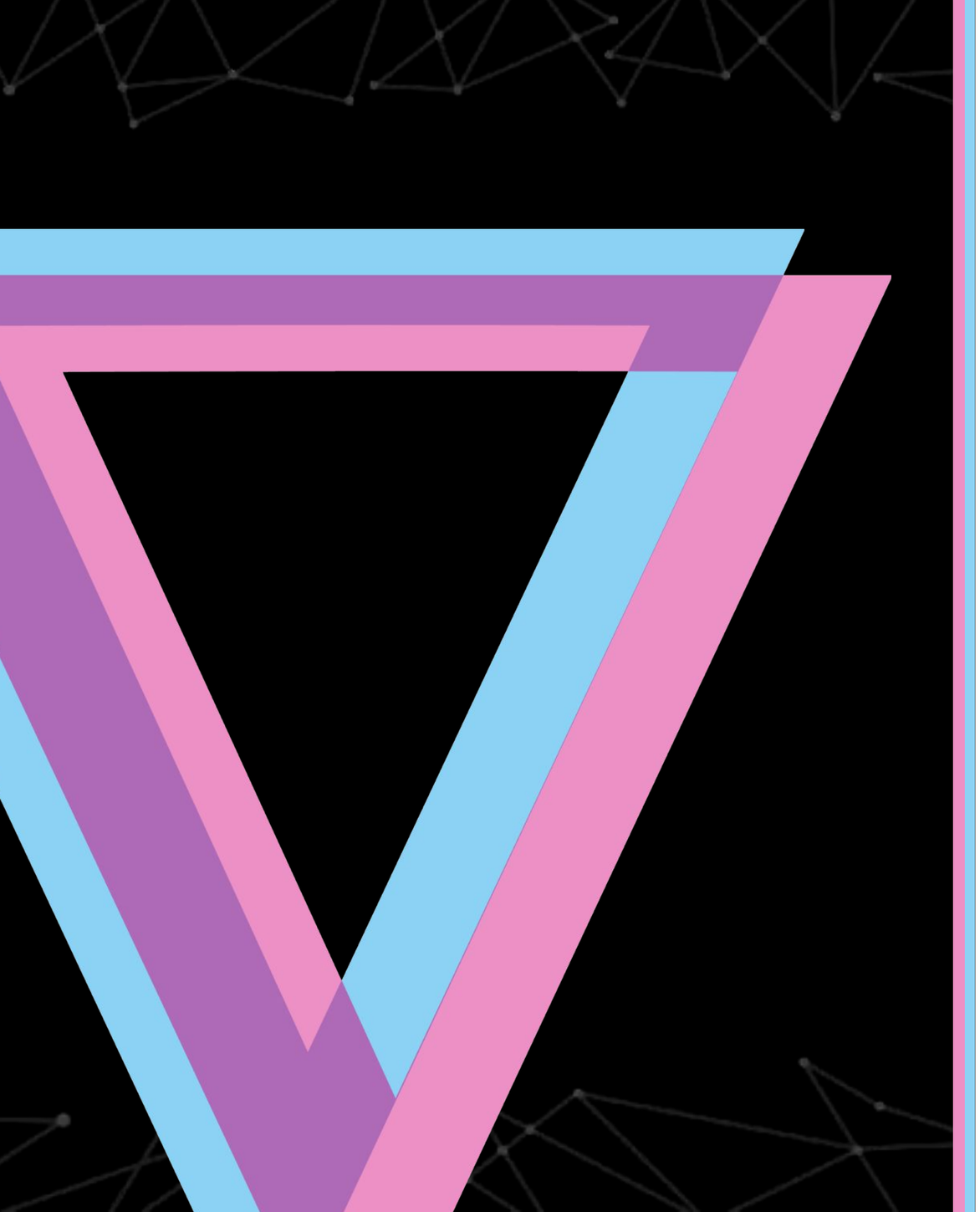


Reinforcement Learning

Glauco Fleury
@s/linkedin

Presença

- Linktree: Presente na bio do nosso instagram
- Presença ficará disponível até 1 hora antes da próxima aula
- É necessário 70% de presença para obter o certificado

Material

- Lembrete:
  - leiam a apostila das aulas para um entendimento mais amplo e completo

Presença

Aprendizado por Diferença Temporal (TD)

Definição de TD

- Considerada a ideia **mais central e inovadora** do Aprendizado por Reforço
- É uma combinação de ideias de **Monte Carlo (MC)** e **Programação Dinâmica (DP)**
- Todos os métodos (TD, MC, DP) usam a estrutura da Iteração de **Política Generalizada (GPI)**
- A principal diferença entre eles está em como resolvem o **problema de predição**

O Melhor de Dois Mundos

Dos métodos de Monte Carlo (MC), o TD herda:

- A capacidade de ser **livre de modelo** (*model-free*)
- Aprende diretamente da **experiência bruta**, sem conhecer a dinâmica do ambiente

Da Programação Dinâmica (DP), o TD herda:

- A capacidade de fazer ***bootstrapping*** (aprender um palpite a partir de outro)
- Atualiza estimativas com base em outras estimativas, **sem esperar o fim do episódio**

TD Prediction

No Monte Carlo, atualizamos o valor de um estado baseado nos **retornos** observados para ele:

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[G_t - V(S_t)\Big]$$

Ou seja, precisamos chegar **no fim do episódio** para poder atualizar o valor dos estados, já que o retorno é a soma de **todas** as recompensas futuras

α: parâmetro stepsize

TD Prediction

No TD, nós não precisamos fazer essa espera. Apenas olhamos **alguns passos na frente**. No caso mais simples de TD, olhando somente um passo à frente, temos:

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\Big]$$

Assim, podemos atualizar a função valor imediatamente seguindo uma transição de estado e **durante o episódio**

TD Prediction

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\Big]$$

Como atualizamos a função valor de cada estado a partir da função valor de outros estados, temos **bootstrapping.**

A diferença entre MC e TD é então o alvo das estimativas:

Em MC: $$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$$

Em TD: $$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s]$$

TD Prediction

Input: the policy π to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(\mathit{terminal}) = 0$

Loop for each episode:
    Initialize S
    Loop for each step of episode:
        $A \leftarrow$ action given by π for S
        Take action A, observe R, S'
        $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
        $S \leftarrow S'$
    until S is terminal

Vantagens dos Métodos TD

vs. Programação Dinâmica (DP):

- Não exigem um **modelo do ambiente** (são *model-free*), aprendendo diretamente da experiência

vs. Monte Carlo (MC):

- São **online e incrementais**: atualizam o valor a cada passo de tempo, sem esperar o fim do episódio
- Muito mais eficientes em tarefas com **episódios longos** ou em **tarefas contínuas** (que não têm fim)

TD: Confiável e Rápido

A convergência é garantida?

- **Sim.** Foi provado matematicamente que o TD(0) - forma mais simples de TD - converge para a função de valor verdadeira (v_{π}).

Qual método é mais eficiente (TD vs. MC)?

- **Teoricamente:** É uma questão em aberto, sem uma prova matemática definitiva.
- **Na prática:** Em tarefas com aleatoriedade (estocásticas), o **TD geralmente converge mais rápido**.

DP vs MC vs TD

Critério	Programação Dinâmica (DP)	Monte Carlo (MC)	Diferença Temporal (TD)
Exige Modelo do Ambiente?	**Sim**	Não	Não
Faz Bootstrapping?	**Sim**	Não	**Sim**
Tipo de Aprendizado	Offline	Offline	**Online e Incremental**
Atualização do Valor	Baseado em Sucessores Diretos	Ao final do episódio (retorno completo)	**A cada passo** (recompensa + próximo estado)

Otimalidade do TD(0)

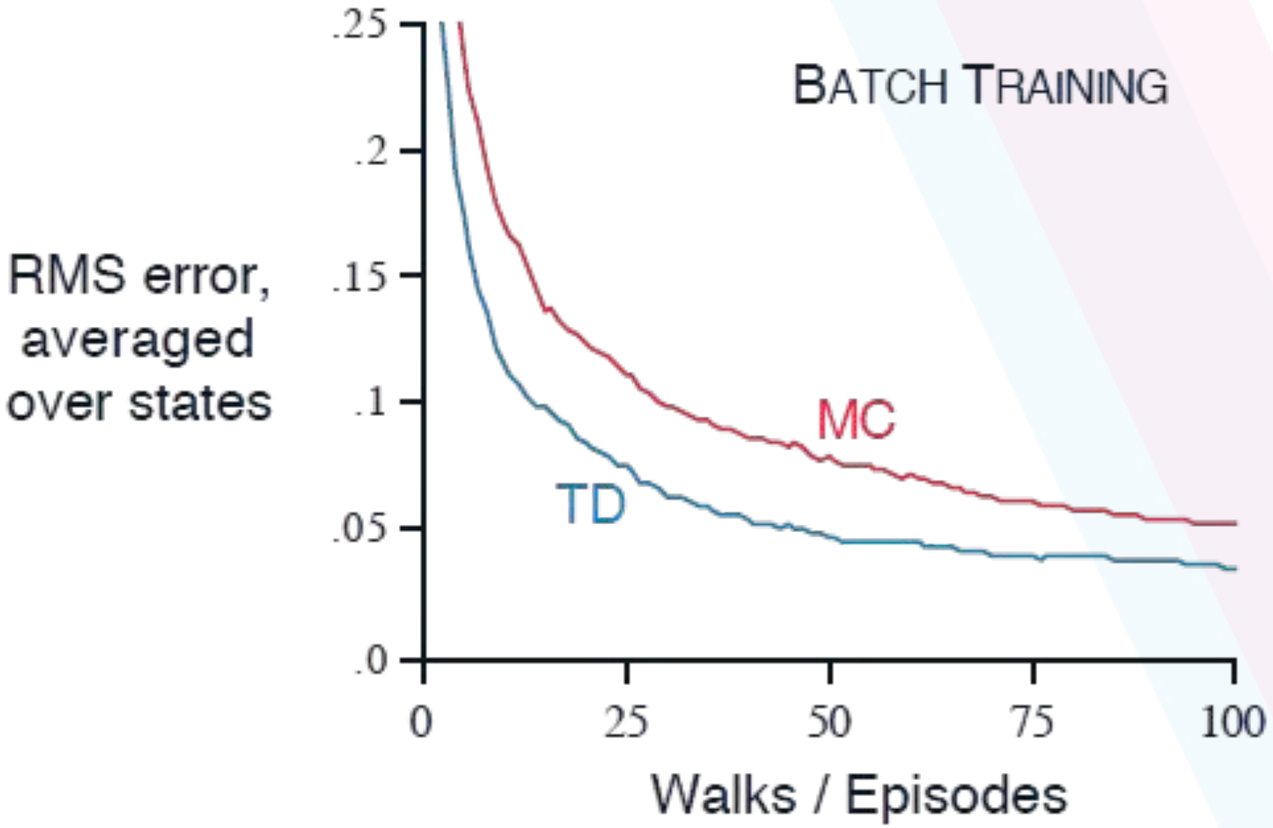
Na prática:

- Métodos TD geralmente convergem mais rápido que os de Monte Carlo (MC) em tarefas estocásticas.

Por que isso ocorre?

- **Para entender o motivo** dessa eficiência, analisamos os métodos em **Treinamento em Lote (Batch Updating)**
- **Treinamento em Lote:** o algoritmo aprende repetidamente com um conjunto fixo de experiências até as estimativas convergirem

Desempenho no treinamento



Evidências empíricas:

- Sob treinamento em lote, TD(0) obtém menores erros do que o método Monte Carlo (MC).

Como?

- **Como** o TD(0) pode ser melhor que o método que já encontra a solução “ótima” dos dados?

Solução MC: Foco nos dados

Caminho:

- Não ter viés, mas obter aproximações mais lentas de V(s)

Convergência:

- **Método MC** converge para V(s), média amostral dos retornos experienciados após visitar cada estado.

Resultado:

- **Maior fidelidade aos dados.** Ajustando ao que foi observado nas experiências.

Solução TD(0): Foco no ambiente

Caminho:

- Encontrar parâmetros que melhor adequem o meio ao MDP (MLE), com estimativas enviesadas

Convergência:

- **Aprender o modelo:** constrói o modelo de máxima semelhança (maximum-likelihood model) a partir dos dados.
- **Resolver o modelo:** calcular o valor que seria correto para o modelo.

Resultado:

- **Maior fidelidade ao ambiente.** Já que o modelo tenta espelhar o ambiente, tem melhor resultado com dados diferentes do treino.

Exemplo

Dados do Lote:

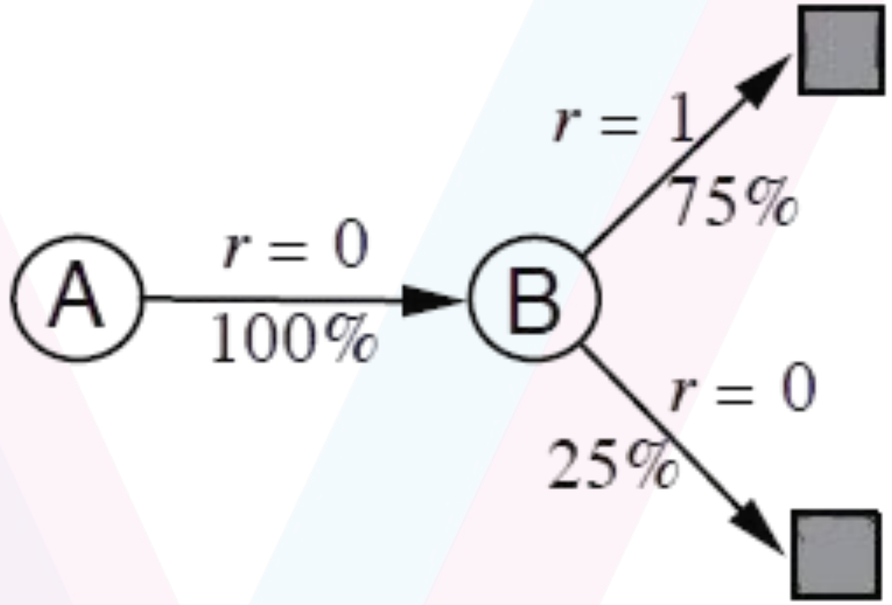
- **8 episódios:**
  - 1: A, 0, B, 0
  - 2 ~ 7: B, 1
  - 8: B, 0

Valor de B:

- Média dos valores. **V(B)** = 6/8 = **3/4**

Valor de A:

- Qual o valor **ótimo** de **V(A)?**



Modelo máxima semelhança

Exemplo

Monte Carlo (MC):

V(A) = 0

Justificativa:

- O único episódio que começou em A terminou em 0, portanto, **V(A) = 0**.
- Assim, acerta perfeitamente o caso 1.

Solução TD(0):

V(A) = 3/4

Justificativa:

- A sempre transita para B (com base no modelo) e, como V(B) = 3/4, **V(A) = 3/4.**
- Resposta correta para o modelo mais provável do ambiente.

TD(0): Maior eficiência

Por que o TD é, geralmente, mais eficiente?

Generalização:

- O algoritmo busca a estrutura do problema ao invés de acertar aos dados, resultando em uma melhor generalização para dados futuros.

Eficiência:

- Pela análise em lote, TD apresenta melhor eficiência do que MC. A cada atualização, a estimativa se aproxima do valor mais correto.

Viabilidade:

- **Menos custosa computacionalmente.** Mais viável de se aproximar da solução de certeza-equivalência.

Algoritmos de TD

Sarsa

- Sarsa é um algoritmo on-policy
- Target: $r_{t+1} + \gamma Q(s_{t+1}, a_{t+1})$
- 2 parâmetros:
  - step-size (se muito alto, vira MC)
  - alpha
- mais “seguro” no contexto geral da GPI

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \Big[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \Big]$$

Sarsa

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize S
    Choose A from S using policy derived from Q (e.g., ϵ-greedy)
    Repeat (for each step of episode):
        Take action A, observe R, S'
        Choose A' from S' using policy derived from Q (e.g., ϵ-greedy)
        $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$
        $S \leftarrow S'$; $A \leftarrow A'$;
    until S is terminal

Q-Learning

- Q-Learning é um algoritmo off-policy
  - utilizamos a política exploratória para descobrir as funções ação-valor da gananciosa
- Target: $r_{t+1} + \gamma \max_a Q(s_{t+1}, a)$
- os mesmos 2 parâmetros do Sarsa
- menos “seguro” no contexto geral da GPI
- busca o caminho mais óptimo possível de forma gulosa

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

Q-Learning

Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
    Initialize S
    Repeat (for each step of episode):
        Choose A from S using policy derived from Q (e.g., ϵ-greedy)
        Take action A, observe R, S'
        $Q(S, A) \leftarrow Q(S, A) + \alpha\left[R + \gamma \max_a Q(S', a) - Q(S, A)\right]$
        $S \leftarrow S'$;
    until S is terminal

Expected-Sarsa

- Expected-Sarsa é um algoritmo on-policy
- Ideia: reduzir variância do Sarsa com valor esperado
- Target: $r_{t+1} + \gamma \sum_a \pi(a \mid s_{t+1})\, Q(s_{t+1}, a)$
- os mesmos 2 parâmetros do Sarsa
- Intermediário entre Sarsa e Q-Learning
  - incorpora aleatoriedade, mas nem tanto

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \sum_a \pi(a \mid s_{t+1})\, Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

Double Learning

Viés de Maximização

- É a superestimação dos valores reais.
- Q-Learning: utiliza a mesma estimativa para escolher e avaliar.

Corrigindo o problema

- O Double Learning utiliza dois estimadores diferentes.

$$Q_1(s,a) \leftarrow Q_1(s,a) + \alpha \left[r + \gamma \cdot Q_2 \left(s', \arg\max_{a'} Q_1(s',a') \right) - Q_1(s,a) \right]$$

Double Learning

- a cada episódio, há 50% de chance de se atualizar a função ação-valor A, e 50% de fazê-lo para B
- ação escolhida p/greedy policy: soma de Qa e de Qb

Prática

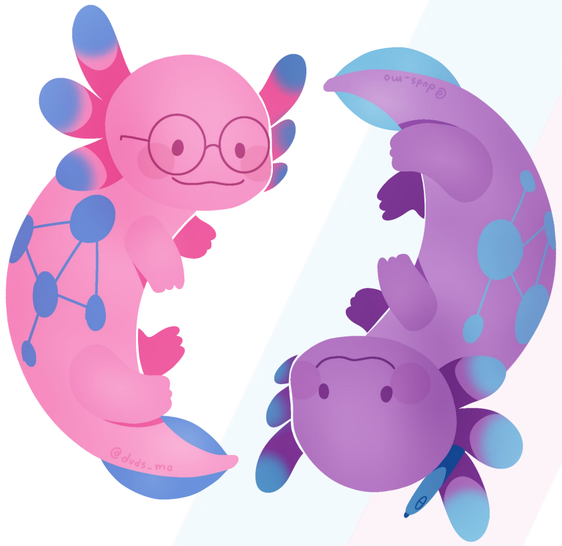
data@icmc.usp.br

@data.icmc

/c/DataICMC

/icmc-data

data.icmc.usp.br



obrigado!