

SCC0270/SCC5809 - Redes Neurais

Aprendizado por Reforço

Profa. Dra. Roseli Aparecida Francelin Romero

SCC - ICMC - USP

30 de outubro de 2018

- 1 Introdução
- 2 Policy Gradient
- 3 Exploração vs Exploração

Características do aprendizado por reforço

- Não existe supervisão, apenas sinal de *recompensa*
- O feedback das ações do agente não é instantâneo
- O tempo é importante (dados são sequenciais, não são i.i.d.)
- Ações do agente afetam dados subsequentes

Exemplos de aplicações

- Manobras com helicóptero
- Jogar GO (AlphaGo)
- Fazer um robô humanoide andar
- Jogar Atari

- Uma **recompensa** R_t é um sinal escalar de feedback
- Indica o quão bem o agente está indo no passo t
- O objetivo é maximizar a recompensa acumulativa

Definição

Todos os objetivos do agente podem ser descritos pela maximização da recompensa acumulativa esperada

Um dos desafios do aprendizado por reforço é modelar o problema desta forma

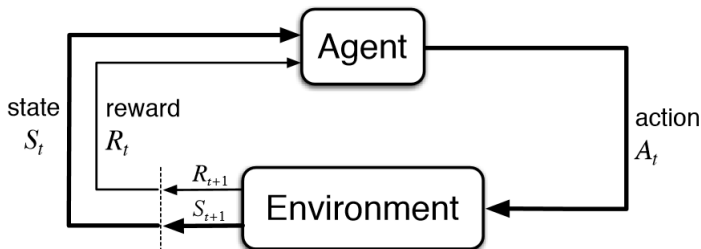
Exemplos de Recompensa

- Manobras com helicoptero
 - + Seguir o caminho desejado
 - - Bater
- Jogar GO (AlphaGo)
 - +/- Ganhar/Perder o jogo
- Fazer um robô humanoide andar
 - + Ir para frente
 - - Quebrar thresholds de segurança
- Jogar Atari
 - +/- Aumentar/Diminuir score

Tomada de decisão sequencial

- Objetivo: Maximizar a recompensa total futura
- Ações podem ter consequências de longo prazo
- Recompensa pode demorar
- Sacrificar recompensas no curto prazo pode ser uma estratégia melhor para avanços no futuro
 - Investimento financeiro (pode levar meses para dar retorno)
 - Abastecer o helicóptero (pode evitar que caia)
 - Bloquear movimentos do inimigo (pode criar novas chances de vitória muitos passos a frente)

Agente e Ambiente



- A cada passo t o agente:
 - Recebe observação O_t
 - Executa ação A_t
 - Recebe recompensa R_t

- O Ambiente:
 - Recebe ação A_t
 - Emite observação O_{t+1}
 - Emite recompensa R_{t+1}

- O **histórico** é a sequência de observações, ações e recompensas
$$H_t = O_1, R_1, A_1, \dots, A_t - 1, O_t, R_t$$
- Os acontecimentos seguintes dependem do histórico:
 - O agente escolhe ações
 - O ambiente emite observações/recompensas
- O **estado** é a informação usada para determinar o que acontece em seguida
- Formalmente, o estado é uma função do histórico:

$$S_t = f(H_t)$$

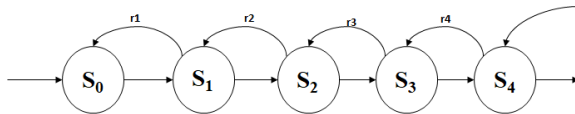
- Soma dos reforços descontados

$$E\left(\sum_{t=0}^{\infty} \Gamma^t r_t\right)$$

- Reforço Médio

$$\lim_{h \rightarrow \infty} E\left(\frac{1}{h} \sum_{t=0}^h r_t\right)$$

Aprendizado por Reforço



Markov Decision Process (MDP)

- Conjunto de estados \mathcal{S}
- Conjunto de ações \mathcal{A}
- Transições estocásticas / modelo dinâmico $T(s, a, s') =$
Probabilidade de estar no estado s' após tomar ação a no estado s
- Recompensa $R(s,a)$
- Política $\pi : s \rightarrow a$
- A política ótima é aquela que produz a maior recompensa acumulada

Um estado de Markov contém toda informação relevante do histórico

Definição

Um estado S_t é markoviano se e somente se

$$P[S_{t+1}|S_t] = P[S_{t+1}|S_1, \dots, S_t]$$

- O futuro não depende do "passado", apenas do presente"
- Uma vez que o estado é conhecido, podemos jogar o histórico fora
- O estado atual é suficiente para determinarmos o futuro

Componentes de um agente

Um agente pode incluir um ou mais dos seguintes componentes:

- Modelo: Representação do ambiente do agente
- Função de valor: Quão bom é cada estado e/ou ação
- Política: Função de comportamento do agente

- Um modelo prediz o que o ambiente vai fazer em seguida
- \mathcal{P} prediz o próximo estado
- \mathcal{R} prediz a próxima recompensa

$$\mathcal{P}_{ss'}^a = P[S_{t+1} = s' | S_t = s, A_t = a]$$

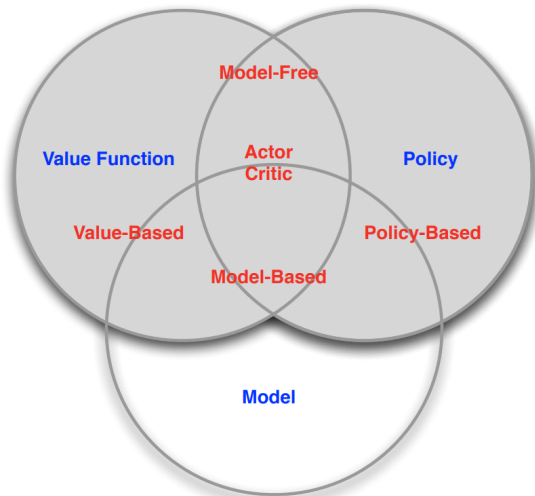
$$\mathcal{R}_s^a = E[R_{t+1} | S_t = s, A_t = a]$$

- É uma estimativa de recompensa futura
- Usadas para avaliar estados e assim escolher ações
- Desconta-se as estimativas de recompensa de acordo com o tempo até o timestep

$$v_{\pi}(s) = E_{\pi}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s]$$

- É o comportamento do agente
- É um mapa de estados para ações
- Política determinística: $a = \pi(s)$
- Política estocástica $\pi(a|s) = P[A_t = a|S_t = s]$

Taxonomia do Agente



Resolvendo Problemas Grandes com RL

- Gamão: 10^{20} estados
- Go: 10^{270} estados
- Helicóptero: estados contínuos

Mapear todos os estados se torna inviável!

Likelihood Ratio Policy Gradient

Calculando gradiente para sua política

$$\begin{aligned}U(\theta) &= \sum_{\mathcal{T}} P(\mathcal{T}; \theta) R(\mathcal{T}) \\ \nabla_{\theta} U(\theta) &= \nabla_{\theta} \sum_{\mathcal{T}} P(\mathcal{T}; \theta) R(\mathcal{T}) \\ &= \sum_{\mathcal{T}} \frac{P(\mathcal{T}; \theta)}{P(\mathcal{T}; \theta)} \nabla_{\theta} P(\mathcal{T}; \theta) R(\mathcal{T}) \\ &= \sum_{\mathcal{T}} P(\mathcal{T}; \theta) \frac{\nabla_{\theta} P(\mathcal{T}; \theta)}{P(\mathcal{T}; \theta)} R(\mathcal{T}) \\ &= \sum_{\mathcal{T}} \nabla_{\theta} \log P(\mathcal{T}; \theta) R(\mathcal{T})\end{aligned}$$

$$\sum_{\mathcal{T}} \nabla_{\theta} \log P(\mathcal{T}; \theta) R(\mathcal{T})$$

- Ajuste na direção que torne a trajetória atual mais provável
 $\rightarrow \nabla_{\theta} \log P(\mathcal{T}; \theta)$
- O tamanho do ajuste é dado pelo tamanho da recompensa $\rightarrow R(\mathcal{T})$
- Trajetórias melhores terão um ajuste maior e assim, serem mais prováveis
- $\nabla_{\theta} \log P(\mathcal{T}; \theta) = \sum_{t=1}^H \nabla_{\theta} \log \pi_{\theta}(a_t | s_t)$

Estimativa da Função de valor

O valor do estado s seguindo uma política π é dado pela recompensa imediata mais a esperança de valor nos estados seguintes

$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s, a) [R(s, a, s') + \gamma V^\pi(s')]$$

Iteração de V ajustada:

- Init $V_{\phi_0}^\pi$
- Coletar dados $\{s, a, s', r\}$
- $\phi_{i+1} \leftarrow \min_{\phi} \sum_{(s, a, s', r)} \| r + V_{\phi_0}^\pi(s') - V_{\phi_0}(s) \|_2^2 + \lambda \| \phi - \phi_i \|_2^2$

Combinando as duas técnicas

- Init $\pi_{\theta_0} V_{\phi_0}^{\pi}$
- Loop:
 - Coletar dados $\{s, a, r, s'\}$
 - $\phi_{i+1} \leftarrow \min_{\phi} \sum_{(s, a, s', r)} \| r + V_{\phi_0}^{\pi}(s') - V_{\phi_0}(s) \|_2^2 + \lambda \| \phi - \phi_i \|_2^2$
 - $\theta_{i+1} \leftarrow \theta_i + \alpha \frac{1}{m} \sum_{k=1}^M \sum_{t=1}^{H-1} \nabla_{\theta} \log \pi_{\theta}(a_t^{(k)} | s_t^{(k)}) (\sum_{t'=1}^{H-1} r_{t'}^{(k)} - V_{\phi_i}^{\pi}(s_{t'}^{(k)}))$

Avalia pares de estado e ação (s,a) ao invés de apenas o estado s

$$Q^*(s, a) = \sum_{s'} P(s'|s, a) [R(s, a, s') + \gamma \max_{a'} Q^*(s', a')]$$

- Init Q_{ϕ_0}
- Loop:
 - Coletar dados $\{s,a,r,s'\}$
 - $\phi_{i+1} \leftarrow \min_{\phi} \sum_{(s,a,r,s')} \| r + \max_{a'} Q_{\phi_i}(s', a') - Q_{\phi}(s, a) \|^2 + \lambda \| \phi - \phi_i \|^2$

- Duas questões:
- como interagimos com o ambiente efetivamente (por exemplo, exploração versus exploração, eficiência amostral) e
- como aprendemos efetivamente com a experiência (por exemplo, atribuição de crédito de longo prazo, sinais de recompensa esparsos).

Exploração vs Exploração

- Aprendizado por reforço é como aprendizado por tentativa e erro
- O agente deve descobrir uma boa política a partir de suas experiências
- Sem abrir mão de muita recompensa pelo caminho

Exploração vs Exploração

- Explorar para descobrir mais sobre o ambiente
- Abusar do conhecimento já obtido para maximizar a recompensa
- O agente precisa balancear os dois

- Escolhendo um restaurante
 - Ir ao seu restaurante favorito
 - Conhecer um lugar novo
- Propagandas Online
 - Mostrar a propaganda mais engajadora
 - Testar uma diferente
- Jogos
 - Fazer o movimento que parece melhor
 - Fazer um movimento experimental

- Forma simples e efetiva de adicionar exploração ao agente
 - Com probabilidade $1 - \epsilon$ selecione $a = \underset{a \in \mathcal{A}}{\operatorname{argmax}} Q(a)$
 - Com probabilidade ϵ selecione uma ação aleatória
- ϵ pode ser reduzido constantemente para diminuir a exploração com o aumento da experiência

Deep Reinforcement Learning

- O DQN (deep Q-network) da DeepMind foi um dos primeiros sucessos inovadores na aplicação da aprendizagem profunda em RL. Ele usa uma rede neural para aprender funções Q para jogos clássicos do Atari, como Pong e Breakout, permitindo que o modelo fosse direto da entrada bruta de pixels para uma ação.
- Algoritmicamente, o DQN baseia-se diretamente nas técnicas clássicas de Q-learning.
- No Q-learning, o valor Q, ou “qualidade”, de um par de estado/ação é estimado através de atualizações iterativas baseadas na experiência.

- Em essência, com cada ação que tomamos em um estado, podemos usar a recompensa imediata que recebemos e uma estimativa de valor de nosso novo estado para atualizar a estimativa de valor de nosso par de estado/ação original.

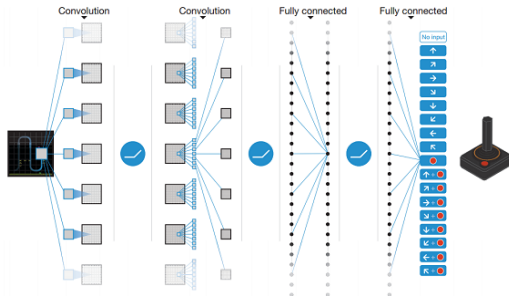
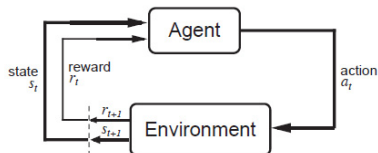
- O DQN de treinamento consiste em minimizar o MSE (erro quadrático médio) do erro de Diferença Temporal, ou erro de TD:

$$Q(s_t, a_t; \theta) \leftarrow Q(s_t, a_t; \theta) + \alpha \underbrace{[r_t + \underbrace{\max_a \hat{Q}(s_{t+1}, a; \theta')}_{\text{target}} - Q(s_t, a_t; \theta)]}_{\text{TD-error}}$$

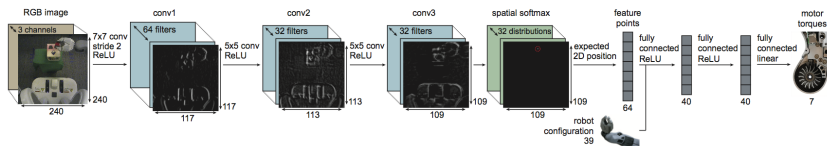
Adaptação de QL para Redes Profundas

- **Replay de experiência:** cada tupla de transição estado / ação (s, a, r, s') é armazenada em um buffer de “repetição” de memória e amostrada aleatoriamente para treinar a rede, permitindo a reutilização de dados de treinamento e de correlação de amostras consecutivas de trajetória;
- **Uso de uma rede alvo separada:** a parte \hat{Q} da equação acima - para estabilizar o treinamento, para que o erro TD não seja calculado a partir de um alvo em constante mudança da rede de treinamento, mas sim de um alvo estável gerado por uma rede fixa.

Exemplo 1: Aprende a jogar Atari



Exemplo 2: Aprende a manipular objetos no mundo real



Asynchronous Advantage Actor Critic - A3C

- Posteriormente, o A3C da DeepMind e a variante síncrona do OpenAI, A2C, popularizaram uma abordagem baseada no aprendizado profundo, muito bem-sucedida para os métodos de ator-crítico.
- Os métodos ator-crítico combinam métodos de gradiente de política com uma função de valor aprendida. Com o DQN, nós só tínhamos a função de valor aprendido - a função Q - e a “política” que seguimos era simplesmente tomar a ação que maximizava o valor Q em cada etapa.
- Com o A3C, assim como o resto dos métodos crítico-ator, aprendemos duas funções diferentes: a política (ou “ator”) e o valor (o “crítico”).

- A política ajusta as probabilidades de ação com base na vantagem atual estimada de executar essa ação, e a função de valor atualiza essa vantagem com base na experiência e nas recompensas coletadas seguindo a política:

$$d\theta \leftarrow d\theta + \underbrace{\nabla_{\theta} \log \pi(a_i | s_i; \theta)(R - V(s_i; \theta_v))}_{\text{REINFORCE policy update}}$$

$$d\theta_v \leftarrow d\theta_v + \underbrace{\partial(R - V(s_i; \theta_v))^2 / \partial \theta_v}_{\text{advantage}}$$

Asynchronous Advantage Actor Critic - A3C

- A **rede Valor** aprende um valor de estado $V(s_i; \theta_v)$, com o qual pode-se comparar nossa estimativa de recompensa atual, R , para obter a “vantagem” e a **rede de políticas** ajusta as probabilidades de registro de ações baseadas nessa vantagem através do algoritmo clássico REINFORCE.
- A contribuição real do A3C vem de sua arquitetura paralela e assíncrona: vários atores-aprendizes são despachados para separar instâncias do ambiente; todos eles interagem com o ambiente e coletam experiência e, de forma assíncrona, empurram suas atualizações de gradiente para uma “rede de destino” central (uma ideia emprestada da DQN).

- RL hierárquica é uma classe de métodos de aprendizado por reforço que aprende a partir de múltiplas camadas de políticas, cada uma das quais é responsável pelo controle em um nível diferente de abstração temporal e comportamental.
- O nível mais baixo da política é responsável pela produção de ações ambientais, deixando níveis mais altos de políticas livres para operar sobre objetivos mais abstratos e prazos mais longos.

- Tudo o que fazemos no nosso dia-a-dia é dividir tarefas complexas e um conj. de subtarefas.
- Isso sugere uma hierarquia inerente e a composicionalidade em tarefas do mundo real, nas quais ações atômicas simples podem ser agrupadas, repetidas e compostas para concluir tarefas complicadas.
- Nos últimos anos, a pesquisa descobriu até mesmo paralelos diretos entre os componentes do RLH e estruturas neurais específicas dentro do córtex pré-frontal.

- Existem várias maneiras diferentes de implementar o HRL.
- Um artigo recente do Google Brain adota uma abordagem particularmente clara e simples e introduz algumas correções fora da política para treinamento com eficiência de dados. Seu modelo é chamado HIRO.

RL Hierarquico - RHL

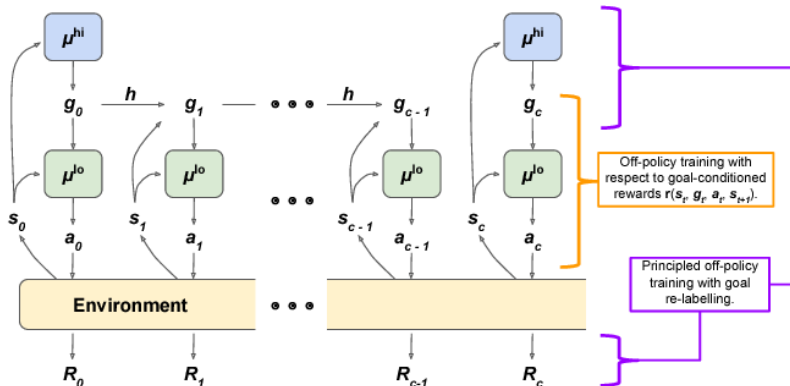






Figura: RHL

-  David Silver - Reinforcement Learning - UCL/DeepMind (2015)
-  CS294-112 - Reinforcement Learning - UC Berkeley (2017)
-  Pieter Abbeel - Deep Reinforcement Learning - OpenAI (2017)
-  Deep Reinforcement Learning Bootcamp (2017)