# Aprendizagem por reforço com redes neurais

Ricardo Yamamoto Abe
ricardoy@ime.usp.br

11 de junho de 2018

# Objetivo

Treinar um agente capaz de vencer algum jogo de Atari 2600

# Q-Learning

Algoritmo de aprendizagem por reforço. Dados um agente, um conjunto de estados $S$, um conjunto de ações $A$, temos:

- A partir de um estado $s \in S$, agente executa uma ação $a \in A$.
- Uma recompensa (valor numérico) é dada ao agente após cada ação.

# Q-Learning

Definimos uma função $Q : S \times A \rightarrow R$. O algoritmo baseia-se em iteração de valores:

$$Q(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha(r_t + \gamma \max_{a'} Q(s_{t+1}, a'))$$

Onde $\alpha$ é a taxa de aprendizagem e $\gamma$ é o fator de desconto (determina o quão importante são as recompensas futuras).

# Q-Learning

Após convergência, a função ótima $Q^*(s, a)$ é encontrada.

# Problema

Função $Q^*(s, a)$ é estimada para cada par (estado, ação). Se o total de ações e estados for suficientemente grande, ou se o domínio do conjunto de estados contiver valores reais, tal abordagem torna-se impraticável.

# Solução

Utilizar uma função de aproximação:

$$Q(s, a, \theta) \approx Q^*(s, a)$$

# Solução

Usar redes neurais como função de aproximação é uma boa idéia?

# Dificuldades

Segundo [Baird, 1995], *Q-Learning* pode não convergir de maneira apropriada quando implementado sobre sistemas generalizados de aproximações de funções, como é o caso de redes neurais com ativação não-linear.

- Sucessão de sequências não são independentes.
- Pequenas alterações nos Q-valores podem fazer a política oscilar muito.

# Proposta de melhoria

- Repetição de experiências
- Fixar rede neural alvo

# Repetição de experiência

Em [Mnih et al., 2103], foi proposto um modelo de memória de repetição de experiências que, em conjunto com aprendizagem em lote numa rede neural convolucional, alcançou o estado da arte em 7 jogos de Atari 2600.

# Repetição de experiência

- Experiência do agente: tupla $(s_t, a_t, r_t, s_{t+1})$.
- Armazenar as tuplas em uma base $D$, chamada memória para repetições.
- Durante o treinamento, amostras de $D$ são utilizadas para construção do lote utilizado na atualização dos pesos.

# Vantagens

- Uso mais eficiente dos dados.
- Previne aprendizagem com dados fortemente correlacionados.
- Suaviza o aprendizado ao agregar dados de várias tuplas anteriormente observadas, tornando a distribuição dos comportamentos de entrada menos enviesados.

# Fixar rede neural alvo

Para evitar oscilações, [Mnih et al., 2015] propôs manter 2 parametrizações de rede neural em memória: $\theta$ e $\theta^-$.

- Os valores alvo do Q-Learning são calculados utilizando a parametrização fixa $\theta^-$:

$$r + \gamma \max_{a'} Q(s', a', \theta^-)$$

- Em cada iteração do treinamento, a parametrização $\theta$ é atualizada.
- Periodicamente, atualizar a parametrização fixa: $\theta^- \leftarrow \theta$.

# Descrição do agente

Foi utilizada a biblioteca OpenAI-gym. Ela contém ferramentas para implementação e comparação de vários problemas de aprendizagem por reforço.

# Descrição do agente

Especificamente para Atari 2600:

- Encapsula o emulador Stella.
- Para cada jogo, é responsável pela entrada e saída de dados, além da geração das recompensas.
- Controle de *Frame-skip*: cada ação é executada por 3, 4 ou 5 *frames*.

# Estrutura da solução

- Dados de entrada: RAM do Atari 2600 (trivia: qual o tamanho?).
- Rede neural com ativação não-linear como aproximação para $Q^*(s, a)$.
- Atualizações em lote com repetição de experiência.
- Fixar rede neural alvo.

# Estrutura da rede neural

1. *inputLayer = RAM(128)*
2. *hiddenLayer1 = DenseLayer(RAM, 1024, ReLU)*
3. *batchNorm1 = BatchNormalization(hiddenLayer1)*
4. *hiddenLayer2 = DenseLayer(batchNorm1, 1024, ReLU)*
5. *batchNorm2 = BatchNormalization(hiddenLayer2)*
6. *outputLayer = DenseLayer(batchNorm2, numberOfActions, no activation)*

# Normalização do lote

Em aprendizagem de máquina, é usual aplicar mudança de escala dos dados de entrada: intervalo limitado ou normalização.
A mesma idéia pode ser utilizada para cada camada da rede neural.

# Normalização do lote

No treinamento, dados valores $x$ de um lote $B = \{x_1, \ldots, m\}$, efetuamos:

1. $\mu_B \leftarrow \frac{1}{m} \sum_{i=1}^{m} x_i$
2. $\sigma_B^2 \leftarrow \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_B)^2$
3. $\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$
4. $y_i = \gamma \hat{x}_i + \beta$

Durante o teste, podem ser utilizados os valores da média e variância amostral de todas as entradas observadas até o momento pela rede neural e seguir a partir do passo 3.

# Resultado

Mostrar vídeo do modelo treinado.

# References


Jakub Sygnowsk and Henryk Michaelewski (2016)
Learning from the Memory of Atari 2600


Volodymyr Mnih and Koray Kavukcuoglu and David Silver and Alex Graves and Ioannis Antonoglou and Daan Wierstra and Martin A. Riedmiller (2013)
Learning from the Memory of Atari 2600


Sergey Ioffe and Christian Szegedy (2015)
Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift


Leemon Baird (1995)
Residual algorithms: Reinforcement learning with function approximation

# References

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, Demis Hassabis

Human-level control through deep reinforcement learning

# Repositório

https://github.com/ricardoy/atari_rl

# Fim

Obrigado!