Arcade Learning Environment Review

Daniel Margarido

Politécnico de Coimbra
Instituto Superior de Engenharia de Coimbra
a21220124@isec.com

Setembro de 2017

Resumo

Neste artigo é realizada uma review do estado da arte para técnicas de aprendizagem por reforço para jogos da Atari2600. Isto foi conseguido através da comparação de abordagens e algoritmos de aprendizagem por reforço para jogar jogos da Atari2600 associados ao Arcade Learning Environment. Foram percebidas as capacidades de cada abordagem e foi obtido um termo comparativo entre as várias técnicas, percebendo para cada jogo qual das técnicas responde melhor. Importância e Alcance dos Resultados.

Index Terms - games, machine learning, AI, game bot, ALE, arcade learning environment, reinforcement learning, atari2600, review

I. Introdução

Aprendizagem por reforço é um dos caminhos para a inteligência artificial geral, contudo este ainda é um caminho longo. Um passo neste sentido foi a criação do ambiente ALE(Arcade Learning Environment)[1] que fornece uma excelente base para testar e desenvolver estes algoritmos em diferentes mundos(jogos da Atari2600). Este ambiente proporcionou um primeiro passo para se conseguir comparar de forma mais evidente as capacidades de cada técnica.
Mesmo assim, quando se quer entender quais as várias abordagens já existentes e as suas vantagens/desvantagens, é necessário ir analisando artigo a artigo sobre o tema, tornando-se uma tarefa que consome bastante tempo. Deste modo, este artigo tem como objetivo principal aglomerar informações dos vários artigos publicados sobre o ALE de forma a simplificar a compreensão do estado da arte das técnicas utilizadas.
As informações que serão apresentadas para cada artigo serão as principais abordagens utilizadas, o seu foco e os resultados obtidos com a sua aplicação.
O restante deste artigo está estruturado da seguinte forma. A secção 2 apresenta o Arcade Learning Environment, a secção 3 apresenta o Estado da Arte, a secção 4 apresenta a Metodologia, a secção 5 apresenta a Descrição de Algoritmos e Abordagens, a secção 6 apresenta a Comparação dos Algoritmos e Abordagens, a secção 7 apresenta a Discussão e a secção 8 as Conclusões e Trabalho Futuro.

II. Estado da Arte

O trabalho apresentado neste artigo foi motivado por abordagens como apresentado em [2], [3], [4] que aprendem através da própria experiência de jogo, sem serem programadas com regras específicas para o domínio que estão a resolver.
Estas abordagens sempre tiveram uma proximidade com os jogos, por exemplo, desde sempre foi muito empolgante a sua aplicação ao xadrez, desde a disputa entre o Kasparov e o DeepBlue em 1997[5], até aos algoritmos criados hoje em dia, como o Giraffe descrito em [3]. Este possui regras, função objetivo e técnicas

de pesquisa que evoluem ao longo do tempo com a experiência. Esta abordagem conseguiu em 72h de treino um agente capaz de jogar ao nível FIDE International Master.
Foram analisados ambientes que permitam a comparação destas técnicas e que ao mesmo tempo se apresentem como um desafio complexo para a evolução do estado da arte. O Super Mario Evolution[6] mostra que os jogos são o ambiente perfeito para testar algoritmos. Apresenta uma base para benchmarks, utilizando o jogo Super Mário e um gerador de níveis infinito. Devido à geração aleatória, o ambiente já força os algoritmos a generalizar e a se adaptar a diferentes situações. Uma plataforma mais recente é o Arcade Learning Environment[1], esta permite jogar os jogos da Atari2600 tendo um estado inicial aleatório em todos eles e permitindo também trocar entre os vários jogos, forçando a uma maior capacidade de generalização dos algoritmos, pois, de jogo para jogo, as estratégias que precisam de ser criadas variam bastante.
Foram selecionadas as seguintes abordagens que usam o ALE como base. O artigo [7] apresenta um modelo de deep learning que aprende políticas de controlo diretamente de dados sensoriais de alta dimensão aplicando aprendizagem por reforço. O [8] demonstra uma framework para deep reinforcement learning que utiliza gradiente descente assíncrono para otimização de controladores de deep neural networks. O [4] introduz a utilização de abordagens neuro-evolutivas para jogar jogos da Atari 2600 com pouco conhecimento do domínio. O [9] exibe a criação de um agente capaz de jogar jogos da Atari2600 em tempo real melhor que uma DQN. O [10] propõem a adição de recorrência a uma DQN de forma a resolver os problemas de memória limitada dos agentes e da necessidade de compreender um estado completo a cada ponto de decisão. Por fim, o [11] demonstra que o bootstrapped DQN pode combinar deep exploration com deep neural networks de modo a obter aprendizagem exponencialmente mais rápida que qualquer estratégia de decisão.

III. Arcade Learning Environment

O ALE apresentado em [1], estabeleceu-se como um desafio e, ao mesmo tempo, como uma plataforma de avaliação da criação de inteligência artificial geral, independente de domínio.

Figura 1: *Jogo breakout da atari2600.*

Este apresenta uma interface para correr os jogos da Atari2600 utilizando o emulador Stella[12]. Este permite a recolha de dados sobre o estado do jogo e a possibilidade de interagir com o mesmo. Para os agentes treinarem e decidirem que ações escolher, estes podem utilizar os seguintes dados:

- Ações possíveis no jogo;
- Píxeis no monitor com resolução de 160x210 e 7 bits de cor;
- 128 bytes da RAM;
- Recompensa da última ação;
- Quantidade de vidas restantes;
- Conhecimento de que o jogo terminou.

As ações tomadas variam consoante o jogo que o agente está a jogar, pois cada jogo tem diferentes mecânicas e problemas a ultrapassar, tendo até um máximo de 18 ações possíveis.
Este consegue avaliar a capacidade de inteligência artificial geral possuída pelos agentes, pois estes podem ser treinados com um jogo e testados nos outros, garantindo que os dados de treino sejam diferentes dos dados de teste.

IV. Descrição de Algoritmos e Abordagens

Após o lançamento do ALE em 2013 já foram publicados vários artigos com diferentes e interessantes abordagens para tentar conseguir a melhor pontuação possível nos jogos existentes.

Playing Atari with Deep Reinforcement Learning[7], utiliza uma CNN(Convolution Neural Network) treinada com uma variante de Q-learning, com stochastic gradient descent para actualizar os pesos. Corre em tempo real, tendo como input os pixeis em bruto do monitor do jogo e como saída uma estimativa da recompensa futura.
Visto que o agente apenas observa as imagens do monitor atual a tarefa é apenas parcialmente observada, pois ficam vários estados possíveis por considerar. Deste modo foram utilizadas também as sequências de ações e observações do agente. Sendo que todas as sequências no emulador terminam numa sequência finita de tempo, podemos definir cada sequência como um estado. Foi ainda aplicada uma técnica chamada replay memory que tem como base a ideia de que aprender diretamente de amostras seguidas temporalmente, é ineficiente. Isto pois irão ser criadas relações fortes entre as amostras. Esta abordagem consiste em armazenar as experiências do agente num dataset a cada etapa. Deste dataset serão recolhidas aleatoriamente amostras que ao aplicar o Q-learning irão ficar com um tamanho fixo de experiência para ser passada à CNN que terá como output a ação a tomar pelo agente.
Os autores chamam a abordagem de Deep Q-learning e a rede resultante como DQN(Deep Q-Network).

Asynchronous Methods for Deep Reinforcement Learning[8], focasse na criação de técnicas leves e simples que utilizem deep reinforcement learning aplicando grandient descent assíncrono para otimização dos controladores. As técnicas apresentadas são variações de algoritmos de aprendizagem por reforço comuns, com foco na redução dos custos de execução para o hardware e no paralelismo. Os algoritmos criados foram:

- Asynchronous one-step Q-learning, cada thread interage com uma cópia própria do ambiente e a cada passo calcula o gradiente do Q-learning. É utilizada uma rede partilhada e que muda lentamente. São também acumulados os gradientes durante vários passos antes de serem aplicados, isto reduz as possibilidades de múltiplos atores subscreverem os updates do outro;
- Asynchronous one-step Sarsa, é semelhante ao one-step Q-learning contudo tem um target diferente para o Sarsa;
- Asynchronous n-step Q-learning, para calcular um update o algoritmo escolhe ações até um numero de passos definidos ou até um estado terminal do jogo. Este processo resulta numa quantidade de recompensas igual ao numero de passos. O algoritmo calcula então gradientes para updates de cada par estado-ação encontrados desde o ultimo update. Os updates acumulados são aplicados num passo único de gradiente à rede partilhada;
- Asynchronous advantage actor-critic, possui políticas e a função objetivo variáveis. Estes são atualizados como o Asynchronous n-step Q-learning atualiza os seus dados.

Mesmo simplificando bastante a aprendizagem como os algoritmos simplificados e a correrem em paralelo, estes introduzem alguma variação extra no modelo de aprendizagem. Isto faz com que o modelo consiga aprender melhor com menos dados e não fique facilmente viciado no resultado mais favorável ao algoritmo atual.

A Neuroevolution Approach to General Atari Game Playing [4], demonstra uma abordagem neuro-evolutiva para jogar com pouco conhecimento de domínio jogos da Atari 2600. Foram apresentadas os seguintes algoritmos neuro-evolutivos:

- CNE(Conventional Neuro-evolution), utiliza uma ANN(Artificial Neural Network) de topologia fixa e apenas os seus pesos vão alterando. Crossover e mutação são utilizados para gerar a descendência. Desta maneira este algoritmo consegue ajustar cada peso individualmente, dado tempo suficiente pode encontrar politicas que consigam excelente jogabilidade;
- CMA-ES(Covariance Matrix Adaptation Evolution Strategy), a topologia é semelhante à da CNE, o que difere é a utilização do CMA-ES em vez de Crossover e mutação. Este algoritmo é um algoritmo de pesquisa de políticas que gera e avalia conjuntos de candidatos sorteados utilizando uma distribuição gaussiana multivariada. Depois de avaliados os candidatos, a média da distribuição gaussiana multi-variada é recalculada utilizando os candidatos com a maior fitness;
- NEAT(Neuro-Evolution of Augmented Topologies), é uma técnica que evolui ao longo do tempo tanto a topologia como os pesos de uma ANN. Alterar a topologia da rede permite obter capacidade para escalar a complexidade da rede de modo a coincidir com a complexidade do problema. O NEAT é baseado em três princípios chave: seguir genes com um marcador histórico para permitir o crossover entre topologias, aplicar especiação para preservar a inovação e desenvolver topologias incrementalmente começando de uma estrutura simples;
- HyperNEAT, evolui uma CPPN(Compositional Pattern Producing Network) que é utilizada para definir os pesos da ANN que produz a solução para o problema. Considera-se que a CPPN tem uma noção geométrica, pois cada peso gerado por esta é uma função das coordenadas de nós presentes em cada camada da ANN. Desta maneira esta ANN terá implicitamente conhecimento sobre relações geométricas no domínio. A própria CPPN irá ser evoluída através do NEAT. Esta abordagem permite expressar redes neuronais bastante grande enquanto se evolui apenas uma pequena e compacta CPPN.

Foram ainda feitos testes com diferentes tipos de inputs para os algoritmos apresentados, com objetos previamente identificados pelos autores para cada um dos jogos, com ruído e com pixeis em bruto.

Deep Learning for Real-Time Atari Game Play Using Offline Monte-Carlo Tree Search Planning[9], tem como objetivo a construção de agentes online que consigam ser utilizados sem um modelo inicial definido e sejam mais rápidos que o normal, dado que esta abordagem costuma ser lenta. Para tal foi inicialmente utilizado um agente UCT(Upper Confidence Bound 1 applied to trees) de modo a prover dados para os outros agentes. De seguida foram então implementados 3 métodos:

- UCTtoRegression, corre um jogo 800 vezes jogado pelo agente UCT e constrói um dataset a partir destas jogadas. Mapeia os últimos 4 frames de cada estado com os pares de ação-valores dos valores calculados pela UCT. Usa os dados obtidos para treinar a CNN por regressão;
- UCTtoClassification, é semelhante à UCTtoRegression, mas utiliza cada coluna dos dados obtidos para treinar a CNN por classificação e de forma gananciosa;
- UCTtoClassification-Interleaved, corre um jogo 200 vezes jogado pelo agente UCT e constrói um dataset a partir destas jogadas. Os dados obtidos são utilizados para treino da CNN como na UCTtoClassification. 3. A CNN é utilizada para decidir as ações nas próximas 200 execuções. São repetidos os procedimentos anteriores até atingir as 800 execuções.

A arquitetura da CNN coincide com arquitetura apresentada em [7].

Deep Recurrent Q-Learning for Partially Observable MDPs[10], procura encontrar uma solução para os controladores criados com deep reinforcement learning pois estes tem memória limitada e precisam de com-

preender todo o estado do monitor para cada decisão. Para resolver este problema foi trocada no algoritmo DQN a primeira camada post-convolutional fully-connected por uma LSTM(Long Short Term Memory) recorrente, resultando numa DRQN(Deep Recurent Q-learning Network) que mesmo que passa a ser capaz de visualizar um frame de cada vez, contudo consegue ter uma noção temporal e resistir melhor a ruído.

Deep Exploration Via Bootstrapped DQN[11],

V. Metodologia

Para cada uma das abordagens serão apresentados aspetos interessantes dos seus resultados e de serão comparados os pontos obtidos em cada um dos jogos por cada algoritmo. Além da pontuação de cada algoritmo ainda são apresentadas pontuações para os melhores humanos, de modo a servir de baseline.

VI. Resultados das Várias Abordagens

Cada abordagem tem resultados específicos consoante os seus diferentes focos, criando uma complicação para comparar diretamente a pontuação de todos os algoritmos, além de que mesmo quando feitos os testes estes podem ter sido realizados em jogos diferentes. Será realizada então uma comparação para as que possuem pontuações em jogos semelhantes. Esta comparação pode ser observada na tabela 1.

Agora já com conhecimento sobre os algoritmos e o funcionamento de cada uma das abordagens vamos apresentar os resultados obtidos em cada uma delas.

Playing Atari with Deep Reinforcement Learning, conseguiu a criação de um método capaz de dominar diferentes políticas de controlo para a Atari2600.

Asynchronous Methods for Deep Reinforcement Learning, mesmo utilizando para aprendizagem os algoritmos simplificados e a correrem em paralelo, estes introduzem variação extra no modelo de aprendizagem. Isto faz com que o modelo consiga aprender melhor com menos dados e não fique facilmente viciado no resultado mais favorável ao algoritmo atual. Conseguem também obter várias políticas de decisão enquanto aplicam os algoritmos em simultâneo. Com isto obteve-se até 24x a velocidade normal de treino apenas aumentando o numero de cores de uma máquina normal, e mesmo assim manteve-se uma boa capacidade de aprendizagem.

A Neuroevolution Approach to General Atari Game Playing, estes não atingiram pontuações tão boas como a pontuação humana. No geral as políticas evoluídas podem ser superiores às planeadas se forem capazes de explorar uma sequência de jogo que seja possível repetir.

Deep Learning for Real-Time Atari Game Play Using Offline Monte-Carlo Tree Search Planning, normalmente os agentes UCT não conseguem jogar os jogos da Atari2600 pois são muito lentos e precisam de acesso ao estado do jogo que não está disponível para os humanos, no entanto com as técnicas implementadas conseguiram agentes extremamente capazes e que conseguem jogar em tempo real.

Deep Recurrent Q-Learning for Partially Observable MDPs, comprovou que que a rede DRQN proposta, tem uma maior capacidade para aguentar ruído que a DQN.

Deep Exploration Via Bootstrapped DQN.

VII. Discussão

O método de comparação avalia a parte mais importante de cada algoritmo que é a capacidade de jogar os jogos, contudo cada método pode ter outras capacidade que não são tão bem apresentadas apenas com esse numero. Outra parte desse problema é que nem todos os artigos tem a sua pontuação detalhada, ou não tem a pontuação nos mesmos jogos. Para mitigar esta situações foi feita uma descrição de resultados interessantes de cada abordagem

Algoritmo	**B.Rider**	**Breakout**	**Enduro**	**Pong**	**Q*bert**	**Seaquest**	**S. Invaders**	**Bowling**
Deep Q-learning Network	5184	225	661	21	4500	1740	1075	—
Asynchronous one-step Q-learning	8500	290	—	18	3700	—	725	—
Asynchronous one-step Sarsa	—	—	—	—	—	—	—	—
Asynchronous n-step Q-learning	10800	300	—	19	4300	—	750	—
Asynchronous advantage actor-critic	**14600**	**500**	—	20	11200	—	1425	—
Conventional Neuro-evolution	—	—	—	—	—	—	—	**252**
Covariance Matrix Adaptation Evolution Strategy	—	—	—	—	—	—	—	—
Neuro-Evolution of Augmented Topologies	—	—	—	—	—	—	—	—
HyperNEAT	—	—	—	—	—	—	—	—
UCTtoRegression	2405	143	566	19	12755	1024	441	—
UCTtoClassification	10514	351	942	21	29725	5100	1200	—
UCTtoClassification-Interleaved	10732	413	**1026**	21	**29900**	6100	910	—
Deep Recurent Q-learning Network	3269	—	—	—	—	—	—	62
Human	7456	31	368	-3	18900	**28010**	**3690**	237

Tabela 1: *Resultados de Algoritmos*

além de apresentar as pontuações dos jogos.
Com estes resultados conseguimos perceber que não há um algoritmo que resolva todos os problemas. Podemos confirmar a capacidade dos agentes com base em UCT de que os resultados destes continuam a ser o estado da arte em alguns jogos. Os algoritmos simplificados ainda se destacam em vários jogos também. Os algoritmos neuro-evolutivos obtiveram resultados médios mais baixos, contudo em alguns jogos destacaram-se bastante. Quanto à pontuação do Pong, 21 é a pontuação máxima, então quando os algoritmos com 21 jogaram o jogo de forma perfeita.

VIII. Conclusões e Trabalho Futuro

Com este artigo, as explicações de cada abordagem para jogar os jogos da Atari2600, a apresentação de cada algoritmo e os resultados obtidos conseguimos de forma simples obter uma boa noção do estado da arte para abordagens para jogar jogos da Atari2600 utilizando o ALE. Este feito vai permitir a qualquer pessoa ter um bom mapa de onde pode começar a investigar artigos da Atari2600 em vez de ter de analisar um de cada vez.
O ALE permite-nos ter uma boa noção dos algoritmos de aprendizagem por reforço que estão a ser testados e evoluídos, fornecendo uma benchmark para poder comparar os resultados de cada investigador. Contudo limitando-nos apenas a este ambiente pode-se não analisar técnicas pioneiras bastante interessantes que sejam aplicadas noutro ambiente.
Os resultados obtidos nestes artigo permitem rapidamente comparar alguns dos algoritmos utilizados no ALE que estão publicados, e os resultados permitem também chamar a atenção para o problema de que muitos dos artigos que utilizam o ALE mesmo realizando a avaliação das pontuações não as publicam, algo que faria todo o sentido.
Para trabalho futuro há várias situações a abordar. Visto que agora já se entendeu o estado da arte atual, o próximo passo será tentar criar uma nova abordagem, integrando as já existentes para o ambiente ALE. Dado que o estado da arte está em constante evolução, dentro de algum tempo será também necessário realizar novamente uma review deste tipo.

Bibliografia

[1] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *Journal of Artificial Intelligence Research*, p. 253–279, 2013.

[2] S. Thrun, "Learning to play the game of chess," *Tesauro, G., Touretzky, D. S., & Leen, T. K. (Eds.), Advances in Neural Information Processing Systems 7 Cambridge, MA. The MIT Press*, 1995.

[3] M. Lai, "Giraffe: Using deep reinforcement learning to play chess," *https://arxiv.org/abs/1509.01549*, 2015.

[4] M. Hausknecht, J. Lehman, R. Miikkulainen, and P. Stone, "A neuroevolution approach to general atari game playing," *IEEE Transactions on Computational Intelligence and AI in Games*, p. 1–13, 2013.

[5] "Deep blue versus garry kasparov." `https://en.wikipedia.org/wiki/Deep_Blue_versus_Garry_Kasparov`. Acedido a 07 de Setembro de 2017.

[6] J. Togelius, S. Karakovskiy, J. Koutnik, and J. Schmidhuber, "Super mario evolution," *Proceedings of IEEE Symposium on Computational Intelligence and Games (CIG)*, 2009.

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, , and M. Riedmiller, "Playing atari with deep reinforcement learning," *Neural Information Processing Systems (NIPS) Deep Learning Workshop*, 2013.

[8] V. Mnih, A. Badia, M. Mirza, T. Harley, T. Lillicrap, D. Silver, and K. Kavukcuoglu, "Asynchronouos methods for deep reinforcement learning," *Asynchronous Methods for Deep Reinforcement Learning*, pp. 1928–1937, 2016.

[9] X. Guo, S. Singh, H. Lee, R. Lewis, and X. Wang, "Deep learning for real-time atari

game play using offline monte-carlo tree search planning," *NIPS 2014*, 2014.

[10] M. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," *arXiv preprint arXiv:1507.06527*, 2015.

[11] I. Osband, C. Blundell, and A. P. andBenjamin Van Roy, "Deep exploration via bootstrapped dqn," *arXiv preprint arXiv:1602.04621*, 2016.

[12] "Stella a multi-platform atari 2600 vcs emulator." `https://stella-emu.github.io/`. Acedido a 07 de Setembro de 2017.