Sistemas de Aprendizagem

André Almeida Gonçalves A75625, Rogério Gomes Lopes Moreira A74634,
Tiago Filipe Oliveira Sá A71835,

Aprendizagem e Extração de Conhecimento, Perfil de Sistemas Inteligentes,
Universidade do Minho

Resumo O presente trabalho tem como objetivo aumentar os conhecimentos nos vários sistemas de aprendizagem atualmente existentes no mercado. As áreas exploradas são: Aprendizagem por Reforço, Redes Neuronais Artificiais e Algoritmos Genéticos. Para cada um deles são descritos os processos de aprendizagem, as ferramentas disponíveis no mercado, entre outros.

1 Introdução

Quase todos os dias ouvimos falar em sistemas de aprendizagem. Tudo é inteligente, seja o frigorífico em nossa casa ou a análise dos nossos exames médicos num hospital. Contudo, nem sempre percebemos os sistemas que estão por detrás dessa inteligência. Torna-se portanto essencial perceber os vários tipos de aprendizagem atualmente existentes. Neste trabalho exploramos três áreas: Aprendizagem por Reforço, Redes Neuronais Artificiais e Algoritmos Genéticos. Para cada um deles são explorados os seguintes tópicos:

1. Descrição característica;
2. De que modo exibe a capacidade de aprendizagem;
3. Que ferramentas de desenvolvimento existem;
4. Que soluções de mercado existem no mercado baseadas em cada tema.

Numa fase inicial do documento são descritos os dois primeiros pontos para cada área de aprendizagem e numa fase posterior são descritos as ferramentas e as soluções existentes no mercado para cada uma das áreas.

2 Aprendizagem por Reforço

"Aprendizagem é o processo pelo qual as **competências, habilidades, conhecimentos, comportamento ou valores** são adquiridos ou modificados, como resultado de estudo,**experiência**, formação, raciocínio e observação. Este processo pode ser analisado a partir de diferentes perspetivas, resultando em diferentes teorias de aprendizagem. Aprendizagem é uma das funções mentais mais importantes em humanos e animais e também pode ser aplicada a **sistemas artificiais** (13))

Desta definição, é importante reter a ideia de adquirir certos comportamentos, através da experiência, uma vez que a aprendizagem por reforço incide fortemente sobre este princípio. De uma forma geral é inspirada em psicologia comportamental e aborda o como um **agente** deve interagir com o **ambiente**, de forma a maximizar um valor numérico - a **recompensa**.

2.1 Problemática da aprendizagem por reforço:

Os problemas de aprendizagem por reforço envolve aprender a mapear situações em ações, tal que o "ganho"seja máximo. Essencialmente, são problemas de "ciclo fechado", uma vez que as ações do sistema de aprendizagem influenciam os seus "inputs"posteriores. Para além disso, o agente não é instruído acerca de quais ações tomar, como em muitas das outras formas de "machine learning", mas tem de descobrir quais proporcionam uma melhor recompensa, testando-as todas, de uma maneira muito similar à lógica de "tentativa e erro". Nos casos mais desafiantes e interessantes, as ações podem afetar não só a recompensa imediata, como a próxima situação e a partir disso, todas as recompensas sucessivas. Podemos então inferir três características determinantes da aprendizagem por reforço: o facto de ser essencialmente um "ciclo fechado", não ter instruções diretas sobre que ações tomar e o facto das consequências das ações se propagarem por intervalos de tempo alargados.

Uma vez referidos estes aspetos da aprendizagem por reforço, é necessário também abordar a questão do **ambiente**. Este é, tipicamente, formulado como um processo de decisão de Markov (MDP).Os MDP's fornecem uma estrutura matemática capaz de modelar tomadas de decisões em situações, onde os resultados são parcialmente aleatórios e parcialmente sobre o controlo de quem toma as decisões. Quando as probabilidades ou as recompensas são desconhecidas, trata-se de um problema de aprendizagem por reforço. Para tal, é útil definir a seguinte função, que corresponde a tomar a ação a e optar sempre pela política ótima.

$$Q(s,a) = \sum_{s'} P_a(s,s')(R_a(s,s') + \gamma V(s')).$$

Figura 1. Função que descreve o Q-Learning

Enquanto que esta função é desconhecida, a experiência durante a aprendizagem é baseada nos pares (s,a) (juntamente com os resultados de s'; isto é, "estava no estado s, executei a e s' aconteceu"). Assim, cria-se um array **Q** e usa-se as experiências para o atualizar diretamente. Isto é conhecido como **Q-Learning**.

O problema da otimização dos MDP consiste em capturar os aspetos mais importantes do problema a resolver no ambiente em que está inserido para atingir um objetivo. O agente necessita de sentir o estado do ambiente, até um certo grau, e deve tomar ações que afetam esse estado. O agente deve ter também um ou vários objetivos, relativamente ao estado do ambiente. A formulação dos MDP pretende incluir apenas estes três aspectos: estado, ação, recompensa. Qualquer método que seja capaz de resolver este tipo de problemas, é considerado um método de aprendizagem por reforço.

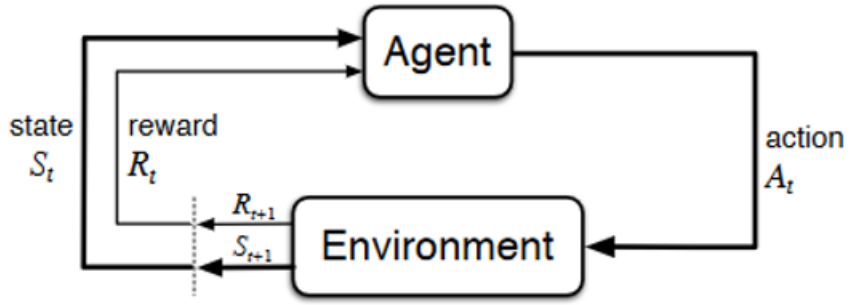

Figura 2. A interação sobre ambiente de agentes na aprendizagem por reforço

A aprendizagem por reforço é diferente da aprendizagem supervisionada, uma vez que esta assenta, sobretudo na aprendizagem através de um conjunto de exemplos de treino, especificados, fornecidos por um supervisor externo. Cada exemplo contém uma descrição, e uma especificação do comportamento correto a ter para uma determinada situação. O objetivo é generalizar as respostas, de forma a agir corretamente a situações que não estão no conjunto de treino. Este é um tipo de aprendizagem importante, mas não o mais adequado para aprendizagem a partir da interação. Em território desconhecido, onde a aprendizagem

se provaria mais benéfica, um agente tem de ser capaz de aprender das suas próprias experiências.

A aprendizagem por reforço é também diferente da chamada aprendizagem não supervisionada. Uma vez que, esta foca-se mais em encontrar padrões em dados não categorizados. Embora haja a tentação de considerar a aprendizagem por reforço, como aprendizagem não supervisionada, não o podemos fazer, porque não se rege por exemplos de comportamento correto. Está apenas a tentar maximizar a recompensa, e não a tentar encontrar padrões escondidos. Por isso, é possível considerar a aprendizagem por reforço como um paradigma de "machine learning", a par da aprendizagem supervisionada e da aprendizagem não supervisionada, entre outros.

Um dos desafios que surge na aprendizagem por reforço, e não em outros tipos de aprendizagem, é o "trade-off"entre "exploration"e "exploitation". Para maximizar a recompensa, um agente deve preferir ações que já experienciou no passado e que provaram ser eficazes em produzir uma recompensa positiva. Mas para descobrir tais ações, ele tem que tentar ações que não selecionou antes. O agente tem que explorar o que já sabe para obter a recompensa ("exploitation"), mas também tem que explorar, de forma a escolher melhor ações no futuro. O dilema é que nem a "exploration"nem a "exploitation"podem ser realizadas exclusivamente sem falhar na tarefa. O agente deve tentar uma variedade de ações e favorecer progressivamente aquelas que parecem ser melhores. Numa tarefa estocástica, cada ação deve ser testada muitas vezes para obter uma estimativa confiável da sua recompensa esperada.

2.2 Elementos da aprendizagem por reforço:

Para além do agente e do ambiente, é possível identificar 4 sub-elementos de um sistema de aprendizagem por reforço: a política, a recompensa, a função de valores e, opcionalmente, o modelo do ambiente.

A política define o comportamento do agente, num determinado momento. De grosso modo, é o mapeamento entre os estados percecionados do ambiente e as ações a tomar nesses estados. Em alguns casos pode ser uma simples função ou uma tabela de consulta, enquanto que noutros casos pode envolver computação extensiva, tal como um processo de procura. A política é o núcleo do agente, no sentido que, sozinha é suficiente para determinar o seu comportamento.

A recompensa define o objetivo de um problema de aprendizagem por reforço. A cada passo, o ambiente envia ao agente um valor - a recompensa. O objetivo do agente é maximizar este valor a longo prazo. Dependendo do valor da recompensa, as ações do agente são consideradas como boas ou más, e tem um papel fulcral na alteração da política.

A função de valores, ao contrário da recompensa que é imediata, específica o que é bom a longo prazo. Aproximadamente, o valor do um estado é a quantidade total de recompensa que um agente pode esperar acumular ao longo do futuro, a partir desse estado. As recompensas são, de certo modo, primárias, enquanto que os valores, como previsões de recompensas, são secundários. Sem recompensas, não poderia haver valores, e o único propósito de estimar os valore,

é para obter maiores recompensas. No entanto, é com os valores com os quais estamos mais preocupados ao fazer e avaliar decisões. As escolhas das ações são feitas com base em juízos de valor. Procuramos ações que geram estados de maior valor, e não a recompensa mais alta, porque essas ações obtêm a maior recompensa a longo prazo. Infelizmente, é muito mais difícil determinar valores do que é determinar recompensas. De facto, o componente mais importante de quase todos os algoritmos de aprendizagem por reforço que consideramos é um método para estimar eficientemente os valores.

O quarto e último elemento de alguns sistemas de aprendizagem por reforço é o modelo do ambiente. Isto é algo que imita o comportamento do ambiente, e tenta prever, por exemplo, dado um estado e uma ação, o próximo estado e a recompensa. Os métodos que usam modelos e **planeamento** são chamados métodos "model-based", em oposição aos métodos "model-free"(mais simples) que basicamente aprendem por **"tentativa e erro"**.

2.3 Aprendizagem por reforço, um contexto real:

"A aprendizagem por reforço copia um princípio muito simples da natureza, a associação de certos comportamentos com o objetivo desejado – ação e consequência.

Alguns dos primeiros investigadores na área da inteligência artificial acreditavam que este princípio poderia ser reproduzido em máquinas. Em 1951, Marvin Minsky, um estudante em Harvard, que acabaria por se tornar um dos pais da AI como professor do MIT, construiu uma máquina que usava uma forma simples de aprendizagem por reforço para imitar um rato a aprender a percorrer um labirinto. "Minsky's Stochastic Neural Analogy Reinforcement Computer", ou SNARC, consistia em dezenas de tubos, motores e "garras" que simulavam o comportamento de 40 neurónios e sinapses. Assim que um rato simulado escapasse do labirinto virtual, a força de algumas conexões sinápticas aumentaria, reforçando o comportamento inerente.

Houve apenas alguns sucessos nas décadas seguintes. Em 1992, Geral Tesauro, um investigador da IBM, demonstrou um programa que usava esta técnica para jogar "Gamão". O programa ficou bom o suficiente, que podia competir com os melhores jogadores (humanos), um marco histórico para a AI. Mas a aprendizagem por reforço provou-se difícil de escalar para problemas mais complexos. "As pessoas consideravam-na uma ideia porreira, mas que não resultava mesmo", diz David Silver, um investigador da DeepMind no Reino Unido e um dos grandes empreendedores da aprendizagem por reforço atualmente.

Mas esse ponto de vista mudou drasticamente em Março de 2016. Foi quando o AlphaGo, um programa treinado usando aprendizagem por reforço, "destruiu" um dos melhores jogadores de Go de todos os tempos, o sul coreano Lee Sedol. O acontecimento foi tão extraordinário, porque é virtualmente impossível de construir um bom programa capaz de jogar Go, usando métodos de programação convencionais. O jogo não é apenas extremamente complexo, como até grandes jogadores Go podem ter dificuldades em caracterizar certas jogadas como boas ou más, então os princípios do jogo são muito difíceis de traduzir em código.

Muitos dos investigadores na área da AI estimavam que demoraria uma década para um computador jogar o jogo tão bem como um humano muito experiente."
Will Knight

3 Redes Neuronais Artificiais

As redes neuronais artificiais foram inspiradas no sistema biológico de redes neuronais constituintes do cérebro. Estes sistemas foram implementados para executar tarefas através de exemplos anteriores (casos de teste), não sendo assim necessário programação específica para o efeito. Por exemplo, estes sistemas são capazes de identificar um certo objeto numa imagem, este processo passa por treinar o sistema com imagens, as quais foram previamente identificadas como tendo ou não o objeto. Usando estes casos de teste o sistema é capaz de encontrar as mesmas características noutras imagens e assim sendo identificar se o objeto está ou não presente na imagem. Como funciona

Uma rede neuronal artificial, à semelhança das redes neuronais, é representada por um conjunto de neurónios, unidos uns aos outros através de sinapses para transmitir informação entre eles. Os neurónios podem apresentar um estado (normalmente representado por um número real entre 0 e 1) e podem também, à semelhança das sinapses, ter um "peso", que consiste num valor que varia ao longo do processo de aprendizagem de forma a obter resultados mais coerentes. Além disto, podem também apresentar um patamar, tal que, apenas os sinais acima (ou abaixo) do mesmo são reenviados para o próximo neurónio. Por norma os neurónios estão organizados por camadas. Diferentes camadas tem objetivos diferentes,podendo-se separar os neurónios em 3 tipos, neurónios de entrada (input do programa), neurónios intermediários e neurónios de saída (output).

Neste momento as redes neuronais já se encontram presentes no nosso dia a dia e conseguem desempenhar tarefas tais como, reconhecimento de imagens e fala, tradução de conteúdos ou aplicações ao nível de redes sociais e plataformas de comércio eletrónico.

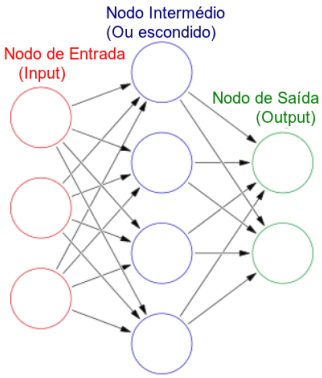

Figura 3. Exemplo de uma rede neuronal

3.1 Componentes das redes

Uma rede neuronal artificial é comparável a um grafo, onde os nodos correspondem a neurónios e as sinapses correspondem às ligações ou arestas. Para que estes grafos funcionem são necessários os seguintes componentes:

1. Neurônios
2. Um neurônio de uma rede artificial é constituído por:
3. Um parâmetro de ativação segundo um tempo discreto;
4. Um patamar, que pode apenas ser alterado pela função de aprendizagem;
5. Uma função de ativação que permite uma nova ativação num novo tempo;
6. Uma função de output, que forma o output da ativação. ...

Um neurónio de input não tem predecessor, serve como interface da rede e é usado para alimentar a rede com nova informação, analogamente, os neurónios de saída não possuem sucessor, e servem como output da rede. Função de Aprendizagem A função de aprendizagem é um algoritmo que modifica a rede neuronal ao longo do seu treino de forma a obter melhores resultados, modificando o peso e patamar dos neurónios e sinapses dentro da rede. Sinapses Nas redes neuronais, os neurónios são unificados por ligações as quais chamamos sinapses, estas têm um peso que vai sendo alterado à medida que a aprendizagem ocorre, sempre com o objetivo de melhorar a capacidade de previsão da rede. Função de custo Esta função é usada para avaliar a função de aprendizagem, sendo que, quanto melhor a função de aprendizagem, menor será o custo desta.

3.2 Aprendizagem

As redes neuronais atraíram interesse devido a sua capacidade de aprendizagem. No meio de redes neuronais, aprendizagem significa, dada uma tarefa e um conjunto de funções, usar um conjunto de observações (casos de teste) para encontrar a função que melhor resolve o problema. Um dos conceitos mais importantes na aprendizagem é a função de custo, que mede quão longe está a solução atual dos resultados pretendidos. Através do output desta função, o algoritmo de aprendizagem procura no conjunto de funções a função que melhor resolve o problema (função com menor custo). A função de custo é escolhida consoante o problema apresentado.

4 Algoritmos Genéticos

Algoritmos são, segundo o dicionário, processos de cálculo. Os Algoritmos Genéticos (Holland, 1975) não fogem à regra. São procedimentos de procura baseados na mecânica da seleção natural e da genética e bons a navegar em grandes quantidades de dados, procurando por soluções que de outra forma não seria possível descobrir (graças à geração de variantes), encontrando uma solução boa e robusta, segundo uma certa medida de desempenho (fitness criteria).

Este tipo de algoritmos, um tipo de algoritmos evolutivos/ algoritmos de aprendizagem, foram inicialmente desenvolvidos com o objetivo de explicar os processos adaptativos dos sistemas naturais e, com isto, desenhar sistemas artificiais de software que conservassem os mecanismos mais importantes dos sistemas naturais. São, por isso, ideais para problemas de otimização, onde se conhece o objetivo e onde os sistemas de aprendizagem necessitam de medidas de desempenho, sendo normalmente utilizados em problemas de natureza combinatória onde a procura de soluções exige grandes esforços computacionais (6).

Apresentam uma estratégia de pesquisa paralela e estruturada, contudo aleatória, cumprindo o principio básico da seleção natural, descrito por Darwin: "Quanto melhor um indivíduo se adaptar ao seu meio ambiente, maior será sua chance de sobreviver e gerar descendentes.", ou seja, à medida que o algoritmo itera os pontos a reforçar são aqueles que tem os melhores valores. Apesar de aleatórios, os pontos de pesquisa são direcionados já que exploram informações históricas para encontrar novos pontos de pesquisa partindo daqueles que tiveram melhores desempenhos na iteração anterior, também chamada de geração anterior.

Um Algoritmo Genético (GA) é assim constituído por quatro componentes principais:

1. A **Função Objetivo**, o objeto de otimização podendo se aplicar a problemas de otimização, a um conjunto de testes para identificar os indivíduos mais aptos ou até mesmo num problema onde apenas conhecemos o formato das entradas e o valor a otimizar não sendo assim necessário para o algoritmo saber como funciona a função objetivo, sendo apenas necessária ter a função disponível para aplicar aos indivíduos e comparar os resultados.
2. Os **Indivíduos**, à semelhança da natureza, o portador do código genético que nada mais é do que uma representação do espaço a pesquisar. São também chamados de cromossomas.
3. A **Seleção** é uma das partes cruciais do algoritmo. Na maioria dos casos os indivíduos são ordenados de acordo com a função objetivo e é lhes atribuído uma probabilidade que resulta da adequação do individuo em comparação com a população onde se encontra. A escolha é depois feita, aleatoriamente, tendo em conta essas probabilidades de forma a que se escolha os melhor adaptados, não deixando de parte a diversidade da população. Contudo, em outros casos, podem-se aplicar outros métodos de seleção como a seleção por torneio, por classificação ou por ranking.
4. A **Reprodução** é, geralmente, dividida em três fases. O acasalamento onde são escolhidos dois indivíduos para se reproduzirem gerando dois novos indivíduos. A recombinação, onde os descendentes recebem a parte do código

genético do "pai" e da "mãe". Esta recombinação garante que os indivíduos recebem as informações os permitem ser ainda mais aptos ao meio onde estão inseridos. Por último, a mutação que tem como objetivo maior variabilidade genética e impede que a pesquisa fique estagnada num mínimo local.

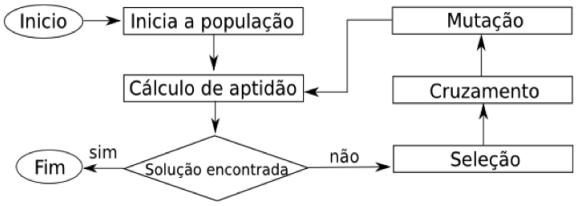

Figura 4. Exemplo do processo de um algoritmo genético

Em cada geração, são aplicados os princípios de seleção e reprodução a uma população de candidatos. O número de indivíduos desta população varia consoante dois fatores: a complexidade do problema a resolver e os recursos de hardware disponíveis. É através da seleção que se determina quais serão os indivíduos da população que mais facilmente se conseguirão reproduzir, gerando assim os descendentes, com uma probabilidade igual ao seu índice de aptidão (quanto maior o seu índice de aptidão, maiores as chances de se reproduzir).

Estes algoritmos diferem dos algoritmos genéricos em quatro aspetos (3):

1. Os Algoritmos Genéticos funcionam com uma representação do conjunto de parâmetros e não com os próprios parâmetros.
2. Os Algoritmos Genéticos funcionam com uma população e não com apenas um ponto.
3. Os Algoritmos Genéticos utilizam informações de recompensa (função objetivo) e não derivadas ou outro conhecimento auxiliar.
4. Os Algoritmos Genéticos utilizam regras de transição probabilísticas e não determinísticas.

Nos sistemas de aprendizagem tenta-se criar soluções para um dado problema partindo de dados ou exemplos (9) de duas maneiras: ou construindo a solução a partir dos dados disponíveis ou procurando uma solução no conjunto de dados. Este segundo método de aprendizagem pode ser feito recorrendo aos algoritmos mais convencionais, contudo o tempo e os recursos necessários para tal tornam-no em algo praticamente impossível, levando a que os Algoritmos Genéticos sejam uma forma mais efetiva de o fazer já que:

1. usam conjuntos de dados discretos;
2. são algoritmos de reforço, ou seja, a certeza dos resultados pode ser medida (9) e evolve em gerações;
3. em alguns casos a melhor solução de um problema são várias soluções.

Por exemplo, num jogo de xadrez os Algoritmos Genéticos nunca seriam usados para ensinar a máquina a jogar mas sim para procurar o melhor movimento a cada jogada

Para quase todos os problemas de computação, é provável que facilmente encontraremos um algoritmo sequencial que o resolve melhor que qualquer Algoritmo Genético. No entanto, esse não é esse o seu intuito. Os AGs são usados quando o nosso problema é uma série de problemas ou quando os seus parâmetros a analisar são muito diferentes (8).

Por exemplo, a programação da função "andar" na robótica é algo extremamente complicado e que pelos algoritmos sequenciais certamente iria falhar. Mesmo, em caso de sucesso, a solução nunca se poderia aplicar a outro robot porque as suas características ou as do meio poderiam mudar. É então mais vantajoso usar Algoritmos Genéticos para "aprender o robot a aprender a andar" em vez de "aprender o robot a andar" e aí encontramos a grande diferença deste tipo de algoritmos (8).

5 Ferramentas de desenvolvimento

5.1 Redes Neuronais Artificiais

Já existem várias frameworks para trabalhar com este tipo de redes no mercado, entre as quais se destaca a ferramenta **JustNN**(`http://www.justnn.com/`),que é uma ferramenta gratuita que funciona com vários tipos de ficheiros, tais como txt, csv, xls, bmp or ficheiros binários. É uma ferramenta bastante acessível para o utilizador e portanto usada em larga escala. Além desta ferramenta, existem outras como é o caso da **OpenNN** (`http://www.opennn.net/`), uma ferramenta com um aspeto mais robusto que a anterior, opensource, desenvolvida em C++ e com ambiente gráfico ou a NeuralDesigner(https://www.neuraldesigner.com/) que permite ao utilizador uma utilização mais prática da biblioteca.

5.2 Aprendizagem por Reforço

Piqle: a Generic Java Platform for Reinforcement Learning

- `https://sourceforge.net/projects/piqle/`

"The Reinforcement Learning Toolbox"

- `https://web.archive.org/web/20120722205525/http://www.igi.tugraz.at/ril-toolbox/general/overview.html`

Java

- `http://www.cse.unsw.edu.au/~cs9417ml/RL1/sourcecode.html`
- `https://github.com/deeplearning4j/rl4j`
- `http://burlap.cs.brown.edu/`

Python

- `https://github.com/openai/gym`

Reinforcement Learning Glue

- `http://glue.rl-community.org/wiki/Main_Page`

5.3 Algoritmos genéticos

Algumas das ferramentas disponíveis no mercado que usam AGs são:

- EvolveDotNet (Framework open-source para Algoritmos Genéticos - C#)
- GAlib (Framework open-source para Algoritmos Genéticos - C
- GAUL (Biblioteca open-source para Algoritmos Genéticos e meta-heurísticas - C)

- GeneticSharp(Biblioteca open-source e multiplataforma para Algoritmos Genéticos - C#)
- JAGA (Pacote open-source para Algoritmos Genéticos e Programação Genética - Java)
- OpenBeagle (Biblioteca de Algoritmos Evolutivos e Genéticos - C
- JGAP (Pacote open-source para Algoritmos Genéticos - Java)
- jMetal(Framework open-source para otimização multi-objetivo que contém Algoritmos Genéticos - Java)
- Pyevolve (Framework open-source para Algoritmos Genéticos e Programação Genética - Python)
- Paradiseo(Framework para meta-heurísticas e algoritmos genéticos em C
- KEEL (Ferramenta de software para extração de conhecimento)
- ECJ (Sistema de pesquisa de computação evolutiva em Java)
- Evolutionary Optimizer (Sistema de visualização de Algoritmos Genéticos)
- GEATbx(Toolbox para Matlab com Algoritmos Evolutivos e Genéticos)
- GPTIPS(Programação Genética e Data Mining para Matlab)
- Pgapack(Biblioteca para Fortran e C para o uso de Algoritmos Genéticos)
- PIKAIA (Software para o uso de Algoritmos Genéticos)
- EVA2 (Workbench de Algoritmos Evolutivos)
- AIGenetic(Biblioteca de Algoritmos Genéticos - Perl)

6 Produtos no Mercado

6.1 Redes Neuronais Artificiais

No mercado começam a aparecer soluções na área de:

- Reconhecimento facial e tratamento de imagens (`http://ieeexplore.ieee.org/abstract/document/554195/?reload=true`)
- Pesquisas ao nivel do DNA (`http://www.pnas.org/content/88/24/11261.short`)
- Analises metereologicas (`http://www.tandfonline.com/doi/abs/10.1080/02626669809492102`)

6.2 Aprendizagem por Reforço

- DeepMind
  - Rebenta os jogos da Atari... `https://deepmind.com/blog/deep-reinforcement-learning/`
- Mobileye
  - The highway merging software was demoed in Barcelona by Mobileye, an Israeli automotive company that makes vehicle safety systems used by dozens of carmakers, including Tesla Motors. `https://www.technologyreview.com/s/603501/10-breakthrough-technologies-2017-reinforcement-learning/`
- OpenAI
  - `https://openai.com/research/`
- Google
  - Self driving car; `https://www.technologyreview.com/s/603501/10-breakthrough-technologies-`
- Uber
  - Self driving car; `https://www.technologyreview.com/s/603501/10-breakthrough-technologies-`

6.3 Algoritmos genéticos

Os Algoritmos Genéticos podem ser aplicados nas mais diversas áreas de desenvolvimento, como por exemplo:

- Área da **Música** - foi apresentado em 1999 na CEC99 (IEEE) um ambiente interativo, utilizando Algoritmos Genéticos para a avaliação de músicas (sequências de acordes) (10).
- **Telecomunicações** - a US West, uma empresa de telecomunicações americana usou um sistema baseado em Algoritmos Genéticos que permitiam projetar, em cerca de duas horas, redes óticas especializadas, trabalho que levaria mais de seis meses caso fosse realizado por humanos (10).
- **Medicina** - foram utilizados este tipo de algoritmos em 1999 para auxiliar na elaboração das escalas dos médicos de uma maternidade. O objetivo pretendido era ao mesmo tempo que se diminuía o esforço e desgate dos médicos garantir que havia disponibilidade 24h (10).
- **Petróleo e Gás** - os Algoritmos Genéticos foram utilizados para o problema da inversão sísmica, bastante importante no campo da geologia e que consiste na determinação da estrutura dos dados de subsolo a partir da projeção geológica (10).

7 Conclusão

Os três sistemas de aprendizagem aqui explorados são diferentes entre eles. Enquanto que as Redes Neuronais Artificiais e a Aprendizagem por Reforço são utilizadas numa fase inicial de aprendizagem, os Algoritmos Genéticos são utilizados numa fase posterior na pesquisa por uma solução válida tendo em conta um contexto do momento. Vemos muitas vezes sistemas em que se usa um conjunto de Redes Neuronais Artificiais com Algoritmos Genéticos para criar uma solução capaz de aprender e aplicar esse conhecimento noutros contextos. As Redes Neuronais Artificiais assimilam-se ainda mais aos Algoritmos Genéticos na medida em que são ambos inspirados na biologia e nos sistemas biológicos. Já a Aprendizagem por Reforço é aplicado em situações em que o agente tem que interagir com o ambiente, aprendendo quais as ações que têm mais recompensas. Este tipo de comportamento é também similar ao que acontece na natureza. Vemos assim que todos os sistemas tem origem na biologia e que apesar das suas diferenças relacionam-se de certo modo.

Agradecimentos

Gostariamos de agradecer aos Professores responsáveis pela unidade curricular de Aprendizagem e Extração do Conhecimento: Professor César Analide Freitas Silva Costa Rodrigues e Professor José Carlos Ferreira Maia Neves, não só pela proposta de trabalho e pelo interesse no seu desenvolvimento como pelos seus valiosos ensinamentos durante todas as aulas.

Referências

1. CORTEZ, Paulo, "Algoritmos Genéticos e Redes Neuronais na Previsão de Séries Temporais", Universidade do Minho
2. CORTEZ, Paulo; ROCHA, Miguel; NEVES, José, "Genetic and Evolutionary Algorithms for Times Series Forecasting", Universidade do Minho
3. CORTEZ, Paulo; ROCHA, Miguel; NEVES, José, "Genetic and Evolutionary Algorithms for Times Series Forecasting", Universidade do Minho
4. GOLDBERG, David, "Genetic Algorithms in Search, Optimization, and Machine Learning", Addison Wesley, 1989.
5. LINDEN, R., 2006, "Algoritmos Genéticos, 1a Edição, Ed. Brasport, Brasil
6. MONTESINOS, F. G; ARNOSO, J.; VIEIRA, R., 2005, "Using a genetic algorithm for 3-D inversion of gravity data in Fuerteventura (Canary Islands)", Int J Earth Sci (Geol Rundsch) n. 94: pp. 301–316
7. HAUPT, RANDI L.; HAUPT, SUN ELLEN, 2004, "Pratical Genetic Algorithms", WILLEY
8. MITCHELL MELANIE, 1998, "An Introduction to Genetic Algorithms", The MIT Press
9. D. Michie, D.J. Spiegelhalter, C.C. Taylor, "Machine Learning, Neural and Statistical Classification", 1994
10. SAPHIRO Jonathan, "Genetic Algorithms in Machine Learning", University of Manchester
11. REDUSINO, Augusto, "Aplicações de algoritmos genéticos", Faculdade Salesiana Maria Auxiliadora
12. `https://en.wikipedia.org/wiki/Reinforcement_learning`
13. `https://pt.wikipedia.org/wiki/Aprendizagem`
14. `https://en.wikipedia.org/wiki/Markov_decision_process`
15. `https://www.technologyreview.com/s/603501/10-breakthrough-technologies-2017-reinforcement-learnin`
16. Richard S. Sutton and Andrew G. Barto, "Reinforcement Learning: An Introduction", The MIT Press, 2nd edition, 2017.
17. `https://en.wikipedia.org/wiki/Artificial_neural_network`
18. Haykin, S., "Neural Networks – A Comprehensive Foundation", Prentice-Hall, New Jersey, 2nd Edition, 1999.