Sistemas de Aprendizagem

Ana Ribeiro Jéssica Lemos Tiago Sousa

Universidade do Minho, Departamento de Informática

23 de Outubro de 2019

Resumo

Quando pretendemos elaborar um sistema inteligente, revela-se essencial escolher um sistema de aprendizagem que se adeque às metas a atingir. O principal objetivo deste trabalho consiste em explorar os diferentes sistemas de aprendizagem existentes no mercado. No nosso caso iremos abordar a aprendizagem por reforço, os algoritmos genéticos e as árvores de decisão.

1 Introdução

Este artigo é o resultado de investigação e consolidação do funcionamento dos sistemas de aprendizagem por reforço, algoritmos genéticos e árvores de decisão no âmbito da unidade curricular de Aprendizagem e Extração de Conhecimento do perfil de especialização de Sistemas Inteligentes do Mestrado Integrado em Engenharia Informática. Para cada um dos sistemas, será realizada uma descrição geral do seu funcionamento e da sua capacidade de aprendizagem. Além disso, são também apresentadas algumas ferramentas de desenvolvimento para cada um dos modelos, e ainda alguns casos reais da sua utilização.

2 Reinforcement Learning

2.1 Descrição e características

Aprendizagem por Reforço ou Reinforcement Learning (RL) é um método de programação em que o agente deverá aprender como se comportar num ambiente dinâmico, executando as ações que lhe permitam maximizar um sinal numérico de recompensa. Como o aprendiz não é informado sobre quais as ações que deve efetuar, este tem de descobrir as que possuem uma maior recompensa experimentando-as. Existem casos em que as ações não afetam apenas a recompensa imediata, mas também as subsequentes. Desta forma, podemos considerar a pesquisa por tentativa-erro e a recompensa atrasada como as duas

características essenciais deste modelo. Neste tipo de aprendizagem não programamos algoritmos para aprender mas sim as características do problema sobre o qual pretendemos aprender.

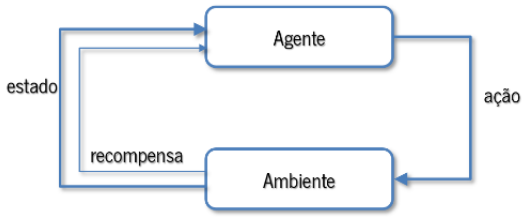

Figura 1: Interação entre o ambiente e o agente

Um dos grandes desafios deste tipo de aprendizagem é a escolha entre *exploitation* e *exploration*. De forma a determinar quais os melhores comportamentos, o agente necessita de ir experimentando novas ações de modo a verificar o efeito que provocam no ambiente em que se encontram (*exploration*). Contudo, por vezes é necessário optar por ações que já tenham sido testadas no passado e que permitiram atingir determinados objetivos (*exploitation*). O grande dilema é que nenhuma das duas pode ser realizada exclusivamente sem falhar na tarefa.

A aprendizagem por reforço deve ser considerada como um novo paradigma de "machine learning", uma vez que difere da aprendizagem supervisionada e da não supervisionada. A aprendizagem supervisionada baseia-se num conjunto de exemplos de treino fornecidos por um supervisionador externo. Cada exemplo possui uma descrição do comportamento a ter para uma determinada situação e o objetivo consiste em adaptá-los para casos que não se encontrem especificados. Este tipo de aprendizagem é fundamental, no entanto não se revela adequado para problemas que se baseiam na interação. A aprendizagem por reforço apesar não conter um supervisionador também difere da não supervisionada, uma vez que esta se concentra em detetar padrões em dados não categorizados.

Por fim, é possível verificar que a aprendizagem por reforço é a que mais se assemelha à do ser humano, na medida em que ao longo da nossa vida vamos decidindo quais as melhores ações a executar através das sensações que cada experiência nos causa (dor ou prazer).

2.2 Modo como exibe a capacidade de aprendizagem

Num sistema de aprendizagem por reforço, é possível identificar para além do agente e do ambiente quatro subelementos que caracterizam a capacidade de aprendizagem, nomeadamente a política, o sinal de recompensa, a função de valor e, opcionalmente, o modelo do ambiente. O principal objetivo consiste em analisar estes elementos para que seja possível alterar o comportamento dos agentes de modo aumentar a recompensa final.

A política determina o modo de comportamento do agente num determinado momento. Basicamente é o mapeamento entre os estados percecionados

do ambiente em que o agente se encontra e as ações que este deve realizar face ao estado atual. O grau de complexidade da política definida varia consoante a situação. Tanto pode corresponder a uma função simples ou a uma tabela de pesquisa como pode envolver uma computação extensiva, tal como um processo de procura. Atendendo ao facto de a política ser capaz de definir o comportamento do agente sozinha, podemos considerá-la uma das bases fundamentais da aprendizagem por reforço.

O sinal de recompensa representa o valor que o ambiente envia ao agente sempre que este executa uma ação. O objetivo do agente consiste em maximizar este valor ao longo do tempo. Este valor recebido, pode alterar uma dada política, uma vez que classifica as ações executadas como benéficas ou prejudiciais. Assim, quando uma ação realizada numa determinada situação recebe uma recompensa baixa, podemos alterar a política para que quando o agente se volte a encontrar no mesmo caso não a execute. De forma análoga, os seres vivos têm tendência a não repetir ações que lhes tenham causado dor no passado.

O valor da função ao contrário do valor da recompensa especifica o que é benéfico ao longo do tempo e não no momento em que a ação é executada. Este representa aproximadamente a soma total de todos os valores de recompensa que o agente espera alcançar no futuro a partir de um determinado estado. Desta forma, podemos considerar o valor da função mais relevante do que o sinal de recompensa, na medida em que é fundamental obter um bom proveito a longo prazo e não apenas num momento especifico. Contudo é evidente que este é mais difícil de determinar do que o valor da recompensa, dado que se baseia em previsões.

Por fim, o modelo do ambiente imita o comportamento do ambiente, para que seja possível prever quais os estados que se seguem a uma determinada ação. Esta característica é essencialmente utilizada para a elaboração de um plano de tomada de decisão. É importante referir que os métodos que utilizam modelos e planeamento são denominados "model-based", enquanto que os "model-free" basicamente aprendem por tentativa-erro.

2.3 Ferramentas de desenvolvimento existentes

Existem várias ferramentas que permitem implementar um sistema baseado na aprendizagem por reforço, pelo que optamos por apresentar quatro das mais promissoras.

- **OpenAI Gym** – Esta ferramenta foi criada em Python e oferece diversos ambientes de simulação de inteligência artificial e suporte para executar treino a agentes em áreas como os jogos, a ciência e a física. Atualmente, é o ambiente mais popuplar no desenvolvimento e comparação de modelos de aprendizagem de reforço. É importante referir que esta é compatível com outras ferramentas computacionais, como é o caso do TensorFlow.

- **TensorFlow** – Este ambiente de desenvolvimento elaborado pela Google fornece diversos módulos RL que podem ser personalizados. A sua im-

plementação pode ser realizada em linguagens de programação bastante conhecidas, como o Java, Python, C e JavaScript.

- **Keras** – Com esta ferramenta é possível elaborar redes neuronais de uma forma simples. Esta é compatível com outras ferramentas da área e caracteriza-se pela sua rápida execução.

- **Pytorch** - Esta ferramenta caracteriza-se pelas suas redes neuronais dinâmicas e pela sua forte aceleração de GPU, pelo que é bastante utilizada nesta área.

2.4 Soluções existentes no mercado

Atualmente a aprendizagem por reforço é utilizada em diversas áreas, assim de seguida serão identificadas as que se destacam:

- **Automatização de Indústria** - A *Google*, por exemplo, aplicou RL no controlo de energia dos seus centros de dados o que permitiu uma redução significativa do uso da energia elétrica.

- **Educação** - A RL está a ser utilizada nesta área, de forma a que seja possível fornecer instruções personalizadas e materiais adaptados às necessidades de cada aluno.

- **Saúde** - As aplicações recorrem a RL, para encontrar políticas de tratamento ótimas e controlar as doses de medicamentos e o uso de material médico.

- **Marketing e Publicidade** - A RL é utilizada nesta área com o intuito de personalizar a publicidade que chega ao utilizador, isto é, apresentar apenas os anúncios que lhe poderão interessar.

3 Genetic Algorithms

3.1 Descrição e características

Algoritmos genéticos ou *Genetic algorithms* (AGs) são algoritmos de optimização e procura, desenvolvido por John Henry Holland, que tem por base os mecanismos de seleção natural e da reprodução genética. Estes princípios em que se baseiam os algoritmos são simples. De acordo com a teoria de Charles Darwin, a seleção natural privilegia os indivíduos melhor adaptados com mais hipóteses de sobreviver, ou seja, maior longevidade e, como tal, maior probabilidade de se reproduzirem. Indivíduos com maior número de descendentes têm maior probabilidade de perpetuarem a sua genética para as gerações posteriores.

Nestes algoritmos tal como nos princípios referidos procuram-se as melhores soluções para um dado problema e para tal a técnica de resolução aplicada consiste numa procura paralela e estruturada, contudo aleatória, para encontrar-se

os pontos mais aptos. Embora aleatório, é explorada, com eficiência, a informação histórica de modo a obter-se os pontos de pesquisa onde são esperados os melhores resultados.

Assim, os AGs diferem dos métodos tradicionais de procura e optimização uma vez que:

- Trabalham com uma codificação do conjunto de parâmetros e não com os próprios parâmetros
- Procura uma população de pontos e não um único ponto
- Utilizam informações de custo e não derivadas ou outro conhecimento auxiliar
- Utilizam regras de transição probabilísticas e não determinísticas

Estas quatro diferenças tornam estes algoritmos mais robustos, e como tal mais vantajoso comparativamente às tecnicas mais usadas.

3.2 Modo como exibe a capacidade de aprendizagem

Os algoritmos genéticos combinam a sobrevivência dos indivíduos mais aptos com estruturas de strings. Para a resolução de um problema inicialmente, é gerada uma população formada por um conjunto aleatório de indivíduos que podem ser vistos como eventuais soluções. Durante o processo evolutivo, é avaliado o nível de adaptação dos indivíduos às restrições impostas através de uma função denominada de *fitness*, que associa a cada indivíduo da população um determinado valor numérico representativo da sua adaptação à resolução do problema. Há diversas maneiras de avaliar cada indivíduo, mas isso dependerá sempre do problema a resolver e dos seus objetivos.

Após a avaliação, a população é submetida a um processo de seleção onde um grupo de indivíduos é escolhido para criar a próxima geração. Um dos métodos mais usados para a seleção da população é o Método da Roleta onde os indivíduos de uma geração são escolhidos para fazer parte da próxima, através de um sorteio de roleta. Nesta cada indivíduo é representado tendo em conta a sua aptidão, ou seja, indivíduos com alta aptidão serão representados na maior parte da roleta enquanto que os que apresentam mais baixa são representados numa menor parte da roleta. Assim, posteriormente gira-se a roleta um determinado número de vezes sendo escolhidos os indivíduos que farão parte da próxima geração.

A transformação da população através das sucessivas gerações, deve-se aos operadores genéticos, que certificam-se que esta se diversifica para além de manter as características de adaptação adquiridas por gerações anteriores. Os três operadores são:

- **Reprodução** é um processo em que as strings de cada indivíduo são copiadas de acordo com os valores da funcão objetivo, f. Podemos pensar, de uma forma intuitiva, na função f como uma medida de lucro ou utilidade

que pretendemos maximizar. Um valor mais alto tem maior probabilidade de contribuir com um ou mais descendentes na próxima geração. Desta forma, este operador é uma versão artificial da seleção natural.

- **Crossover** tem duas etapas, primeiro são escolhidos dois indivíduos de modo a ser selecionado um ponto de cruzamento. Por fim, é escolhida uma posição na string aleatoriamente. No exemplo seguinte a posição escolhida foi a 4.

$$\begin{array}{ll} A_1 & 0\ 1\ 1\ 0 \mid 1 \\ A_2 & 1\ 1\ 0\ 0 \mid 0 \end{array}$$

  Assim, são criadas duas strings novas, ou seja indivíduos, trocando os caractéres tal como podemos verificar de seguida.

$$\begin{array}{ll} A_1\text{'} & 0\ 1\ 1\ 0\ 0 \\ A_2\text{'} & 1\ 1\ 0\ 0\ 1 \end{array}$$

- **Mutação** é a troca aleatória das posições dos valores na string como é demonstrado posteriormente. Este é um operador importante porque apesar da reprodução e do crossover procurarem e recombinarem material genético existe a possibilidade de algum deste, que seria útil, ser perdido. Assim, nos AGs a mutação é um processo completamente aleatório e tem como objetivo manter um nível de diversidade adequado na população.

$$\begin{array}{ll} A_1 & 0\ 1\ 1\ 0\ 0 \\ A_1\text{'} & 0\ 1\ 0\ 0\ 1 \end{array}$$

A reprodução, o crossover e a mutação já provaram ser computacionalmente simples e eficazes na resolução de vários problemas importantes de otimização.

O resultado final deste passo é uma nova geração de novos indivíduos que resultou principalmente dos melhores indivíduos das gerações anteriores. As etapas anteriores repetir-se-ão em ciclo, em que cada geração seguinte será submetida aos vários métodos já referidos, de modo a criar indivíduos mais aptos até que se encontre uma solução que seja suficientemente boa ou até que um determinado número de gerações sejam criadas.

Desta forma, normamente é obtido o indivíduo mais apto da geração final, e se o algoritmo foi bem sucedido, essa será a resposta ao problema que o utilizador está a resolver.

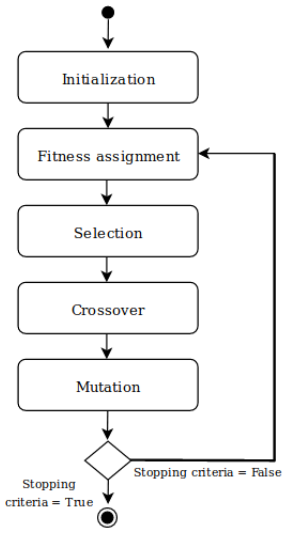

Figura 2: Diagrama de estados dos Algoritmos genéticos

3.3 Ferramentas de desenvolvimento existentes

Os AGs vêm sendo utilizados em várias áreas de pesquisa e em situações do mundo real com bons resultados pelo que são várias as ferramentas de desenvolvimento existentes para a aplicação destes algoritmos. De seguida passamos a enumerar algumas:

- **MATLAB** - É considerado pelos utilizadores uma das melhores ferramentas para algoritmos genéticos. O GAT - Genetic Algorithm ToolBox - é uma coleção de rotinas escrita principalmente em m-files nos quais estão implementadas as funções mais importantes destes algoritmos. Este usa funções da matriz MATLAB para construir um conjunto de ferramentas versáteis para implementar numa ampla gama de métodos de algoritmos genéticos.

- **JCLEC** - Java Class Library for Evolutionary Computation é um sistema para a implementação e execução de algoritmos de computação evolutiva que oferece suporte para algoritmos genéticos, programação genética e programação evolutiva.

- **JGAP** - Java Genetic Algorithms Package é um componente de algoritmos genéticos e programação genética fornecido pelo Java e que foi projetado para ser de fácil utilização e intuitivo. Este fornece mecanismos genéticos básicos que podem ser facilmente usados para aplicar princípios evolutivos às soluções problemáticas.

- **DEAP** - Distributed Evolutionary Algorithm in Python é uma estrutura de computação evolutiva para prototipagem e teste de ideias que suporta uma grande variedade de algoritmos evolutivos, entre os quais algoritmos genéticos.

3.4 Soluções existentes no mercado

Tendo em conta que os AGs têm caraterísticas que permitem resolver problemas difíceis rapidamente e de uma forma confiável a sua aplicação torna-se bastante apelativa, tal como podemos constatar de seguida.

- **Otimização** - Os algoritmos genéticos são normalmente usados em problemas de otimização cujo objetivo é maximizar ou minimizar um determinado valor de função objetivo tendo em conta um determinado conjunto de restrições.
- **Problema do Caixeiro Viajante** - Na distribuição as empresas usam algoritmos genéticos para o clássico problema dos vendedores ambulantes. O novo modelo, baseado em algoritmos genéticos, pesquisa com eficiência de modo a encontrar a melhor estratégia que faça o melhor uso dos recursos da empresa e também reduza o tempo de entrega.
- **Análise de DNA** - Os GAs são usados para determinar a estrutura do DNA recorrendo a dados espectrométricos sobre a amostra.
- **Redes Neuronais** - Estas também são treinadas por estes algoritmos, principalmente as redes neuronais recorrentes.
- **Processamento de imagem** - Estes algoritmos são usados também para tarefas de processamento de imagem digital (DIP), bem como a correspondência densa de pixels.
- **Aplicações de negócio** - Os algoritmos genéticos são também utilizados para caracterizar vários modelos económicos, negociações financeiras, avaliações de crédito, distribuição de orçamento e até deteção de fraude.
- **Geração de trajetória de um robô** - Os GAs foram usados para definir o caminho que um braço de robô percorre, movendo-se de um ponto para outro.
- **Projeto paramétrico de aeronaves** - Estes algoritmos têm sido usados para projetar aeronaves variando os parâmetros de modo a se desenvolver melhores soluções.

4 Decision Trees

4.1 Descrição e características

Árvores de decisão são métodos de aprendizagem indutiva que colocam um conjunto de atributos e decisões em forma de árvore.

Estas árvores contêm nos nós intermédios testes sobre o valor dos atributos, nos ramos as várias hipóteses de valor dos mesmos, e nas folhas o valor da classe que classifica o objeto. O valor das classes vai depender da ponderação de todos os atributos que conduziram àquela folha.

É um método de aprendizagem supervisionada usado para classificação de objetos através de um conjunto de atributos que dividem a população inicial em sub-populações cada vez menores.

Toda a informação sobre os objetos de estudo é representada através de um conjunto fixo de atributos.

As árvores podem ser feitas de forma bottom up, quando temos especialistas que fazem a classificação prévia antes da inserção dos casos de estudo, ou top down quando a partir dos casos de estudo são induzidas as classificações.

As classes podem ser:

- Contínuas quando resolvem problemas de regressão, apresentado um resultado numérico.
- Discretas quando resolvem problemas de categorização apresentado a categoria nominal do objeto.

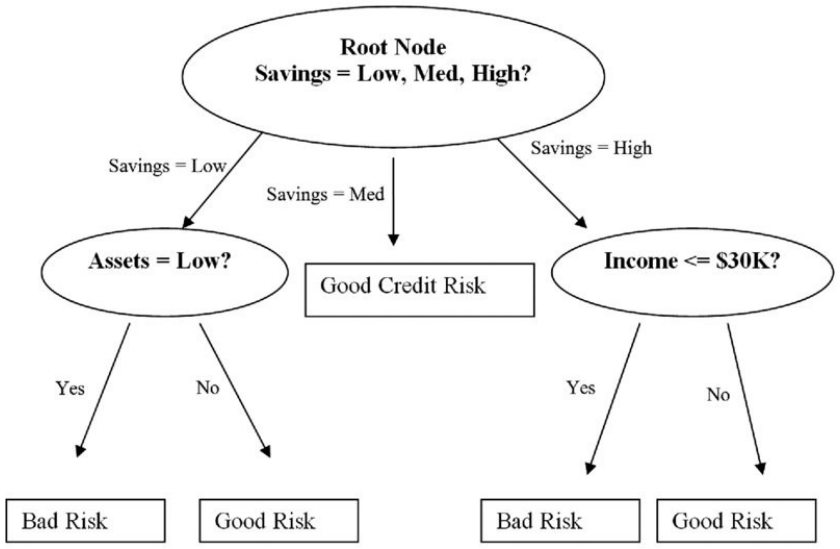

Figura 3: Árvore de decisão para prever o risco de empréstimo

4.2 Modo como exibe a capacidade de aprendizagem

A aprendizagem das árvores de decisão é feita através da aplicação de um algoritmo a um dataset criando um classificador que servirá para classificar não

só os objetos que já recebeu no treino, como novos objetos adicionados como input.

4.2.1 Algoritmos TDIDT

Top down induction of decision trees é uma família de algoritmos muito usados na construção de decision trees, fazendo-o desde a raiz até às folhas. O método recursivo usado por estes algoritmos é geralmente do tipo seguinte:

1. Usar todo o dataset como input.
2. Encontrar uma divisão em subconjuntos baseada num atributo escolhido de acordo com o algoritmo utilizado.
3. Criar nós intermédios na árvore para cada subconjunto do passo 2.
4. Reaplicar os passos 2 e 3 a cada sub-conjunto até ao caso de paragem.
5. Colocar nas folhas a classificação respetiva do sub-conjunto.
6. Fazer pruning para reduzir o overfitting.

Alguns dos algoritmos TDIDT mais importantes são:

- **ID3** O algoritmo ID3, criado por Ross Quinlan, foi o primeiro a ser desenvolvido para gerar uma decision tree a partir de um dataset.

  É um algoritmo greedy, ou seja, pensa apenas no benefício imediato em cada passo, utilizando como critério de escolha de atributos o ganho de informação.

  Este algoritmo faz a escolha do atributo seguinte a ser utilizado na árvore através da entropia do nó pai e dos nós filhos gerados por cada atributo restante. O atributo que gerar maior diferença de entropia entre o pai e os filhos é o escolhido pois traz maior ganho de informação.

  Dois problemas deste algoritmo são não aceitar variáveis contínuas e valores nulos, nem permitir post-pruning.

- **C4.5** O algoritmo c4.5 foi inventado pelo mesmo criador do ID3 de forma a resolver algumas das limitações apresentadas por ele, nomeadamente:
  - Capacidade de fazer post-pruning.
  - Aceita valores contínuos.
  - Utiliza a razão do ganho em vez do ganho de informação.
  - Permite utilização de valores nulos.
  - Lida com problemas em que atributos possuem custos diferenciados.

- **CART** O algoritmo CART, Classification and Regression Trees, é uma técnica de indução de árvores de regressão e categorização , dependendo se os atributos são nominais ou contínuos. Estas árvores têm as seguintes categorias:
  - São árvores binárias podendo ser percorridas da raiz às folhas respondendo apenas a perguntas de sim ou não.
  - Utiliza post-pruning baseado na redução do fator custo/complexidade o que produz árvores mais simples, precisas e com boa capacidade de generalização quando comparado com outros métodos de pruning.
  - A divisão em subconjuntos para quando o algoritmo deteta que não há mais ganho possível.
  - São árvores mais legíveis porque são baseadas apenas em condições if-else.
  - Relações não lineares entre atributos não afetam o resultado final das árvores.
  - Uma pequena mudança no dataset pode tornar a árvore instável, o que é uma desvantagem.

4.2.2 Escolha do melhor atributo

Dependendo do algoritmo, a escolha do melhor atributo vai ser feita de forma diferente, este passo é muito importante na construção das árvores de decisão para garantir que associado a cada nó se encontra o atributo que permite maior ganho de informação. Os principais critérios de escolha de atributos são:

- **Ganho de informação** - Este critério usado no ID3 tenta maximizar a informação contida na árvore, dando preferência a atributos com muitos valores possíveis ou seja, que têm um grande número de arestas associadas. Calcula-se pela diferença de entropia entre o nó pai e os nós filhos gerados pelo atributo em questão.
- **Razão do ganho de informação** - É o resultado da divisão do ganho de informação pela entropia, este método foi usado por Ross Quinlan no algoritmo C4.5 para melhorar os resultados obtidos no ID3.

  Este critério de escolha favorece atributos com maior ganho de informação e menor entropia sendo superior ao ganho de informação tanto em termos de acurácia quanto em termos de complexidade das árvores de decisão geradas.
- **Gini** - É o critério usado no algoritmo CART, é uma equação criada por Corrado Gini para calcular o índice de dispersão estatística.

4.2.3 Pruning

Pruning é uma técnica usada para aumentar a eficácia das decision trees cortando secções das árvores de modo a torná-las menos específicas e mais generalizadas.

Neste método temos que lidar com o problema do tamanho das árvores, em que uma árvore muito pequena pode não conter os níveis de detalhe necessários para caracterização de um objeto, e uma árvore demasiado grande vai estar tão especificada para os casos de treino que ao inserir novos casos vai ter problemas para os classificar.

Este método é subdividido em post-pruning e pre-pruning.

Post-Pruning é o pruning feito após o fim processo de construção da árvore removendo tudo o que está abaixo de um certo nó transformando esse nó em folha.

Pre-Pruning é o pruning feito durante o processo de construção da árvore e funciona como critério de paragem dando por terminada a construção da árvore.

4.3 Ferramentas de desenvolvimento existentes

- **Scikit** - Este package da linguagem python tem vários mecanismos de árvores de decisão.
- **Weka** - Este pacote de software Weka (Waikato Environment for Knowledge Analysis) da linguagem Java apresenta um conjunto de algoritmos de machine learning, entre eles árvores de decisão.
- **rPart** - Este package de linguagem R permite a criação de decision trees segundo o algoritmo de CART.

4.4 Soluções existentes no mercado

As decision trees são aplicadas nas mais diversas áreas, como por exemplo:

- **Saúde** - Na área da saúde os diagnósticos médicos podem ser feitos através de decision trees que através de várias condições e sintomas do paciente classificam o seu estado de saúde.
- **Compras** - Na área das compras existem shopper decision trees, árvores que acompanham todo o p2p (path to purchase) do cliente, que através de registos dos hábitos de consumo e das tomadas de decisão do consumidor durante o processo de compra, se adequam às suas necessidades e preferências. Este processo torna o processo de compras mais fácil e rápido para o utilizador.
- **Atribuição de créditos** - No setor da atribuição de créditos bancários são usadas árvores de decisão para decidir se o crédito é ou não atribuído ao cliente baseado em atributos que vão aumentar ou diminuir o risco do crédito, como o salário, o cadastro ou o agregado familiar.

5 Conclusão

Com a execução deste artigo concluimos que cada um destes sistemas de aprendizagem, apesar de distintos, se unem no propósito final de solucionar problemas de forma inteligente e precisa. Todos estes sistemas que apresentamos são extremamente úteis, porém cada um tem pontos fortes e pontos fracos quando comparado com os restantes em diferentes situações.

A aprendizagem por reforço revela-se bastante adequada para situações em que o agente necessita de interagir com o ambiente para aprender as ações que são mais benéficas. Este tipo de comportamento é muito similar ao que os seres vivos adotam. No entanto, por vezes a quantidade de estados que o ambiente pode assumir e o número de ações que o agente pode realizar revela-se computacionalmente incomportável tornando-se muito difícil alcançar uma política ótima.

Os algoritmos genéticos são reconhecidos pela sua eficiência e rapidez na obtenção de um conjunto de soluções. Estes tornam-se bastante úteis quando o espaço de pesquisa é grande e a quantidade de variáveis associadas é considerável, contudo não é apropriado a todos os problemas dado que podem se tornar computacionalmente dispendiosos. Para além disso, como é um processo estocástico a obtenção de uma solução e a optimização não é garantida.

No caso das decision trees, percebemos que são muito úteis para indução, permitindo uma tomada de decisões estruturada baseada em condições bastante simples e legíveis. Por outro lado este sistema tem o problema de não serem muito flexíveis à mudança, havendo muitos problemas principalmente quando tenta classificar dados diferentes dos aprendidos no treino.

Referências

[1] Richard S. Sutton and Andrew G. Barto, "Reinforcement Learning: An Introduction", The MIT Press, 2nd edition, 2012.

[2] Top 5 tools for Reinforcement Learning, https://hub.packtpub.com/tools-for-reinforcement-learning/

[3] David Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning", Addison Wesley, 1989

[4] Tutorials Point - https://www.tutorialspoint.com/genetic_algorithms/

[5] Medium - https://medium.com/greyatom/decision-trees-a-simple-way-to-visualize-a-decision-dc506a403aeb

[6] Introduction to decision tree learning - https://heartbeat.fritz.ai/introduction-to-decision-tree-learning-cd604f85e236

[7] Quinlan, J. R., "Induction of Decision Trees", Machine Learning 1: 81-106, Kluwer Academic Publishers, 1986