Sistemas de Aprendizagem

Bruno Martins, Catarina Machado, and Filipe Monteiro

(Dated: 18 de Janeiro de 2020)

Neste artigo são explorados três sistemas de aprendizagem, *Case Based Reasoning*, *Genetic Algorithms* e *Decision Trees*, onde para cada um deles são realçadas as suas características diferenciadoras, o modo como exibem a capacidade de aprendizagem, ferramentas de desenvolvimento disponíveis e ainda soluções atualmente existentes no mercado.

Keywords: *Machine Learning, Case Based Reasoning, Genetic Algorithms, Decision Trees*

I. INTRODUÇÃO

No universo dos sistemas de aprendizagem existe um enorme leque de métodos, cada um com as suas características e especificidades. Estes métodos são normalmente utilizados em problemas de *machine learning* mas a maioria pode também ser aplicado a problemas de dados. Entre todos os métodos existentes, neste artigo abordaremos três: *Case Based Reasoning*, *Genetic Algorithms* e *Decision Trees*.

Case Based Reasoning ou Raciocínio Baseado em Casos (*CBR*) pode-se descrever como a capacidade de um sistema armazenar informação relacionada com os casos que ocorreram numa certa área e, utilizando esse conhecimento prévio, auxiliar nas tomadas de decisões subsequentes. Esta forma de resolver problemas é fundamentalmente diferente de qualquer outra abordagem no ramo da Inteligência artificial. Isto deve-se ao facto de não utilizar apenas conhecimento generalizado do domínio que está a ser tratado, mas também de utilizar o conhecimento concreto adquirido em experiências anteriores.

Os *Genetic Algorithms* (Algoritmos Genéticos) são um tipo peculiar de algoritmo que, baseando-se na evolução dos animais na natureza, ataca um problema de procura e otimização com uma grande eficiência, conseguindo resultados mais rapidamente com um grande precisão. Este pertence a uma classe maior de *ML* (Machine Learning) denominada de *Reinforcement Learning*.

As *Decision Trees* (Árvores de Decisão), através de factos já observados, descrevem graficamente as decisões a serem tomadas, os eventos que podem ocorrer e os resultados associados às combinações de decisões e eventos. A grande vantagem deste algoritmo é fornecer o contexto da solução do problema.

II. CASE BASED REASONING

CBR gira à volta do conceito de tendo uma base mais ou menos extensa, na chegada de um novo problema utilizar toda essa base de casos prévios e, analisando caso a caso o problema a que se refere, o método envolvido na resolução e o resultado obtido, determinar qual destes problemas prévios mais se assemelha ao problema atual levando à melhor solução possível. Após a obtenção de resultados, avalia-os e atualiza o seu conhecimento de forma a que caso tenha existido, tanto sucesso como insucesso, numa próxima ocasião os resultados sejam o mais próximo de ótimos.

Esta forma de decisão aparentemente é simples, concreta e não muito variável no que toca à forma de aplicação, mas o *CBR* possui uma variedade de métodos substancial em todos os pontos da tomada de decisão. Tanto na organização, recolha, utilização e indexação de conhecimento adquirido nos casos anteriores.

No âmbito do *CBR*, existem diferentes tipos de métodos, todos com um objetivo comum, mas com diferentes características.

Exemplar based reasoning:

No ponto de vista deste método, um conceito é definido extensivamente como um conjunto dos seus exemplares. Neste, resolver um problema é uma tarefa de classificação; de forma resumida é encontrar a classe correta para enquadrar o exemplar que estamos a avaliar. A classe da maioria dos casos anteriores semelhantes que já avaliamos é a solução para a classificação do nosso problema. Cada conjunto de classes diferentes representa uma solução diferente para os problemas apresentados.

Instance based reasoning:

Este método é uma especialização do descrito anteriormente, sendo utilizado nos casos em que é necessário compensar a falta de informação no âmbito geral do conteúdo que estamos a analisar. Sendo assim, são necessárias um grande número de instâncias de forma a se conseguir aproximar da definição conceito em estudo. A representação destas instâncias é simples, por exemplo num vetor, visto que o objetivo é o estudo na aprendizagem automatizada sem *input* do utilizador. Resumindo, este método tem uma abordagem não generalista ao conceito do problema da aprendizagem que as perspetivas clássicas e indutivas não têm em relação ao *machine learning*.

Memory based reasoning:

Esta abordagem baseia-se numa grande coletânea de casos. A organização destes casos e o seu acesso são o foco da maioria dos modelos *CBR*. Este tipo de modelos também implementam técnica de processamento paralelo, e é esta característica que distingue este modelo de todos os outros. O armazenamento destes dados pode ser feito de forma sintática, ou pode se tentar utilizar conhecimento do domínio geral.

Case based reasoning:

CBR é um termo usado genericamente nesta descrição, mas o *Case based reasoning* mais concreto apresenta algumas características que o distingue das outras abordagens até agora já referidas. Em primeiro lugar, é assumido que a informação sobre o caso é até um certo ponto relevante, tem conteúdo e complexidade no que toca à sua composição interna.

Outra propriedade base deste método é a sua capacidade de se modificar ou adaptar uma solução quando esta vai ser aplicada num contexto diferente do original. Os métodos mais paradigmáticos do *CBR* utilizam também conhecimento geral acerca do contexto em que estão inseridos. As metodologias fundamentais do *CBR* também são baseadas nas teorias da psicologia cognitiva.

Analogy based reasoning:

Este termo é normalmente sinónimo do *CBR* com uma única diferença, o *CBR* tradicional é utilizado em casos sempre do mesmo domínio. No caso do *analogy based reasoning* é utilizado um caso anterior de um domínio diferente para tentar resolver o problema atual. O maior foco de estudo deste campo do *CBR* é a reutilização dos casos anteriores ao qual se dá o nome de mapeamento do problema, encontrar uma forma de transferir a analogia encontrada para o problema atual.

A. Modo como exibe a capacidade de aprendizagem

Uma das características mais importantes do *Case Based Reasoning* é o facto de demonstrar capacidade de aprendizagem. Um dos grandes motores deste tipo de métodos é a comunidade por trás do *M*L (*Machine Learning*) visto que o *CBR* é considerado por muitos um subdomínio de ML. Assim sendo, o *CBR* não denota apenas um tipo particular de raciocínio, mas também demonstra o paradigma de *machine learning* que permite uma aprendizagem sustentada por atualizar a sua base de casos passados após um novo problema ser resolvido. Como é possível inferir através da descrição anterior, a aprendizagem do *CBR* é um produto natural da resolução de problemas. Quando um problema é resolvido de forma bem-sucedida, essa experiência fica guardada de forma a resolver problemas semelhantes no futuro. Quando uma tentativa de resolução aplicada a um problema falha, a razão pela qual não teve sucesso é identificada e registada de forma a evitar o mesmo erro no futuro.

O *CBR* favorece a aprendizagem através da experiência porque, normalmente, é mais fácil aprender olhando para os resultados de experiências passadas do que tentar generalizar a partir delas. Ainda assim, o CBR necessita de um conjunto de métodos bem trabalhados de forma a extrair informação relevante da experiência e inserir na sua base de conhecimento. Outro fator muito importante é a indexação para mais tarde encontrar os casos necessários.

1. Ciclo CBR

O ciclo do *CBR* descrito anteriormente pode ser caracterizado da seguinte maneira:
Retrieve: Procurar o/os caso/casos mais semelhantes;
Reuse: Reutilizar essa informação e conhecimento;
Revise: Rever a solução obtida;
Retain: Reter as partes da solução relevantes para ajudar na solução de problemas que possam surgir no futuro.

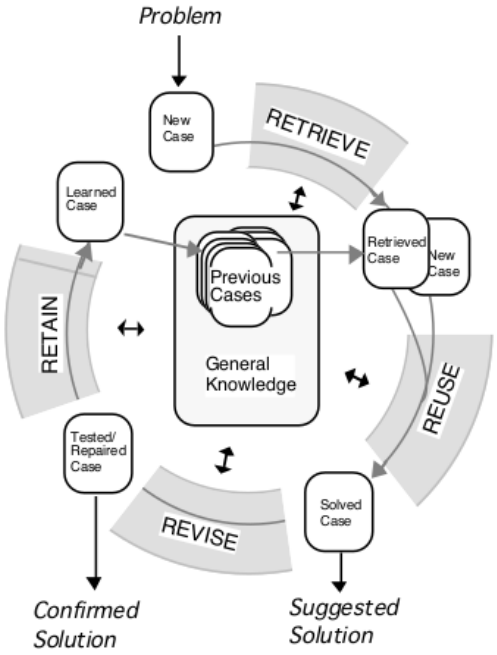

Figura 1. Ciclo CBR.

B. Ferramentas de desenvolvimento existentes

Neste momento as opções de desenvolvimento de modelos *CBR* no mercado já são diversificadas. O intuito destas aplicações é permitir um rápido desenvolvimento a nível aplicacional.

Um exemplo destas características é o *ReMind* da *Cognitive Systems Inc.* que oferece um ambiente interativo para inserir casos, vocabulário relacionado com domínio em que estamos a trabalhar, indexes e protótipos. Este software permite criar relacionamentos entre os atributos tendo em conta as semelhanças existentes entre eles. Para além disso, programa exibe ainda a capacidade de adaptação aos casos utilizando fórmulas e adaptando os seus valores baseados nos valores atuais e das diferenças do novo caso.

Outro software utilizado neste campo de estudo é o *ReCall*, um sistema registado da empresa *ISoft*, uma empresa de AI de Paris que utiliza esta aplicação para di-

agnóstico de falhas, análise de empréstimos bancários, controlo e monitorização.

No contexto académico muitas vezes é utilizado o *myCBR*, um software *open-source* que, utilizando um ambiente gráfico e intuitivo, consegue avaliar as semelhanças entre os casos e decidir qual a melhor decisão a tomar tendo em conta todos ou apenas alguns parâmetros do domínio em que estamos a trabalhar. Estas decisões tomadas têm associadas a si uma probabilidade de certeza.

Estes são apenas alguns exemplos do que o mercado oferece. Tendo em conta o rápido desenvolvimento desta área é esperado que o mercado de oferta mude rapidamente.

C. Soluções no mercado baseadas em Case Based Reasoning

No mercado existem diversas áreas que aplicam modelos de *CBR*. Entre elas estão as áreas dos sistemas de recomendação e da medicina.

No caso dos sistemas de recomendação os métodos de *CBR* são utilizados nos casos em que as ofertas que as empresas ou os motores detêm são muito extensivos e é necessário diminuir a escolha para apresentar ao consumidor. É neste contexto que surgem as aplicações dos modelos *CBR*. Os resultados obtidos para apresentar são analisados e providenciados ao consumidor na ordem que o programa ache mais relevante. Os softwares utilizam funções de similaridade para conseguir comparar dados que vão ser usados para procurar criar novos dados ou informações de teste. Caso existam muitas categorias para avaliar é usada normalização de *min-max*.

Quanto à área da medicina o *CBR* está cada vez mais presente, isto deve-se ao facto de os médicos terem de processar uma grande quantidade de informação, muitas vezes contraditória, até realizarem o diagnóstico final. Outro caso que complica ainda mais as decisões é existirem várias formas de resolução de um problema. Apesar de grande parte dos recursos do *CBR* na medicina estarem focados no diagnóstico, as áreas de epidemiologia e da psiquiatria também começam a ter influência nos modelos aplicados.

Além do diagnóstico, o *CBR* também é utilizado para sistemas de classificação, por exemplo, na análise de imagens hospitalares ou classificação filogenética. Na área do planeamento, este método também é utilizado, por exemplo, para fazer planos alimentares ou tratamentos de radioterapia. Por fim, outra área de aplicação na medicina do *CBR* é em sistemas de acompanhamento, por exemplo, em terapia de antibióticos nos cuidados intensivos e também no auxílio de doentes cardiológicos.

Um dos desafios na área da medicina é o acesso aos dados. Apesar de já ser normal os processos dos pacientes estarem informatizados nos países desenvolvidos existe um problema do ponto de vista da organização e acesso a esses mesmos dados.

III. GENETIC ALGORITHMS

Algoritmos genéticos são algoritmos que, baseados na manifestação da seleção natural encontrada na natureza, conseguem ser excelentes na otimização/procura de valores em problemas de grande dimensão, onde a procura "linear" é extremamente ineficiente. Este tipo de algoritmo, para problemas de domínio infinito (ou sem aparente solução concreta), consegue aproximar a solução da melhor possível.

Um exemplo simples da sua eficácia, embora fora de lógica, foi dado por *Daniel Shiffman* num livro chamado **The Nature of Code**. Consiste na ideia de que, se tentássemos descobrir como escrever a frase *To be or not to be that is the question* através de força bruta, sabendo que a probabilidade de carregar em cada uma das 27 teclas disponíveis é igual, a probabilidade de acertar na frase é ridiculamente baixa e, mesmo que escrevesse 1 milhão de frases por segundo, demoraria milhões e milhões de anos até chegar à solução certa. Usando agora um algoritmo genético, a partir de frases geradas aleatoriamente, podemos ir evoluindo a frase na direção correta (dizendo-lhe se se está a aproximar da solução final), podendo chegar à frase objetivo de forma muito mais rápida. Embora a solução para este problema seja óbvia, serve para demonstrar que o algoritmo possui uma maior eficácia na sua maneira de proceder perante outros algoritmos tradicionais.

A. Modo como exibe a capacidade de aprendizagem

Como dito anteriormente, as GA's são baseadas na seleção natural proposta por Charles Darwin e, como tal, tem componentes fundamentais tais como:

- População de N "entidades";
- Função de Fitness;
- Função de Seleção Natural;
- Função de Cruzamento;
- Função de Mutação.

Para um melhor entendimento do algoritmo faremos uma analogia entre a área da Biologia e tecnicidade do algoritmo:

Evolução Natural	Algritmo Genético
Ser-vivo	Entidade/objeto
Fitness (habilidade de sobreviver)	Função objetivo
Genotype (genes)	Representação dos dados
Phenotype	Solução

Tabela I. Tabela de conversão de termos biológicos para matemáticos/algorítmico.

De seguida, iremos explicar o que é e como são utilizadas dentro do algoritmos as componentes anteriormente descritas.

1. População

A população terá um tamanho fixo e será inicializada aleatoriamente. A população será o conjunto de valores que o algoritmo tentará otimizar com o auxílio de uma série de funções auxiliares. Importante reparar que, ao contrário de outros algoritmos de procura, este utiliza um conjunto de pontos para a procura, ao contrário de apenas um.

Esta população irá variar ao longo das iterações, ficando cada vez mais ótima (**evolução**).

2. Função de Fitness

À semelhança da natureza e de acordo com Darwin, nem todos os seres vivos de uma espécie sobrevivem, sendo que os que conseguem possuem algum tipo de diferença nos **genes** (os seus dados) em relação aos outros. Para isso, criamos uma função de *fitness* que é nada mais que uma função capaz de expressar numericamente a qualidade da solução daquela entidade. Esta função é importantíssima pois é quem decidirá o quão próximo os valores estão da solução ótima - uma função objetivo.

3. Seleção

Depois de filtrar todas as soluções pela função de *fitness*, escolhem-se as melhores soluções para fazerem descendência na nova população. Atualmente, esta seleção não é feita de forma linear (**N** maiores ficam, resto desaparece) mas sim com uma componente aleatória que diz que todos podem permanecer na população mas com diferentes probabilidades de acontecer (conhecida como seleção proporcional) - algo como um jogo da roleta:

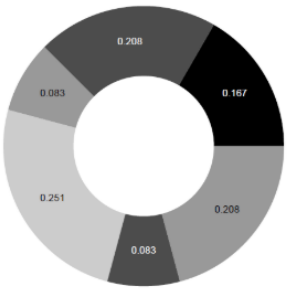

Figura 2. Exemplo das probabilidades da seleção de vários indivíduos.

4. Cruzamento

O processo de cruzamento é o equivalente à **reprodução**, onde pegando aleatoriamente em pares de soluções gera-se uma nova solução, possuindo parte das duas (**cruzamento genético**). Isto assegura uma melhor eficácia na evolução da população, prova da existência de diversidade de soluções na população. Esta troca de "material genético" pode ser feita de várias formas, entre elas:

- *One-point crossove*r: divisão do material genético das duas entidades em duas partes de tamanho aleatório e trocando apenas a última parte;
- *N-point crossover*: equivalente à anterior, mas sendo divididas em N e trocando cada segunda parte;
- *Segmented crossover*: semelhante à anterior, mas o número de divisões pode variar;
- *Uniform crossover*: simples algoritmo que para cada posição do material genético, decide aleatoriamente se deve ser trocada ou não.

5. Mutação

Por último, é necessário deformar material genético das novas entidades para que haja uma maior ainda diversidade, alcançando novas soluções, usando uma componente aleatória.

Claro que esta componente aleatória tem de ser de grau bastante baixo para evitar uma mudança extrema da população, provocando um efeito de procura aleatório - enfraquecendo a eficácia do algoritmo.

Por fim, uma explicação abstrata do funcionamento do algoritmo:

Algorithm 1 Implementação do Algoritmo Genético

```
1: Inicializar população com valores aleatórios;
2: while função objetivo não cumprida do
3:     Selecionar indivíduos para reprodução, utilizando a função de fitness;
4:     Gerar as descendências através do cruzamento dos indivíduos seleccionados;
5:     Possibilidade de aplicar mutações aos descendentes;
6:     Gerar nova população com os descendentes;
```

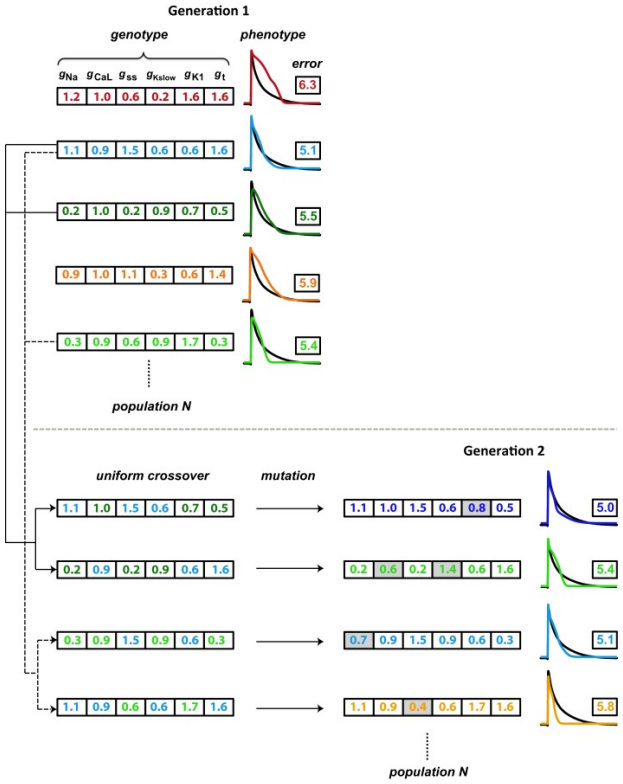

Figura 3. Exemplo de uma seleção, cruzamento e mutação de uma pequena população.

B. Ferramentas de desenvolvimento existentes

Talvez devido à facilidade em usar algoritmos genéticos, não encontrámos muitas ferramentas que proporcionam um meio para desenvolver os agentes. Contudo, encontramos as seguintes:

- *Solver* do Excel (tem até uma opção do solver chamado Evolutionary Solver);
- *ECJ* para JAVA;
- MATLab.

Concluímos que faz bastante sentido que sejam mais encontradas em contexto matemático visto serem uma técnica bastante eficaz na otimização/procura de valores e, sendo de tão fácil implementação, talvez não seja justificada a criação de um ambiente especializado.

C. Soluções no mercado baseadas em Genetic Algorithms

Atualmente, existe uma grande variedade de utilizações dos algoritmos genéticos, mesmo que existindo de forma subentendida, ou seja, existe de suporte a outros tipos de algoritmos.

Por exemplo, as **RNN** (*Recurrent Neural Networks*) utilizam algoritmos genéticos para otimizar os parâmetros da rede neuronal, conseguindo assim o melhor caso possível da sua configuração.

Estudos foram também realizados em várias áreas, como por exemplo no problema de encaminhamento de veículos, de maneira a otimizar custos, reduzir emissão de CO2, entre outros, com resultados bastante positivos.

Por fim, *Genetic Algorithms* são também usados em grandes mercados financeiros, no filtro e processamento de sinal, e na criptografia, no auxílio da decriptação de códigos.

IV. DECISION TREES

Os algoritmos baseados em Árvores de Decisão são considerados um dos melhores e mais utilizados métodos de aprendizagem supervisionada (têm uma variável objetivo predefinida) em *Machine Learning* e *Data Mining*.

De uma forma geral, a árvore de decisão culmina num diagrama do tipo fluxograma no qual é possível analisar os vários percursos e respetivos resultados tendo em consideração as ações que são tomadas. Estas árvores têm uma construção condicional ("*if... then...else...*"), o que as torna numa estrutura facilmente programável e muito simples de ser interpretada.

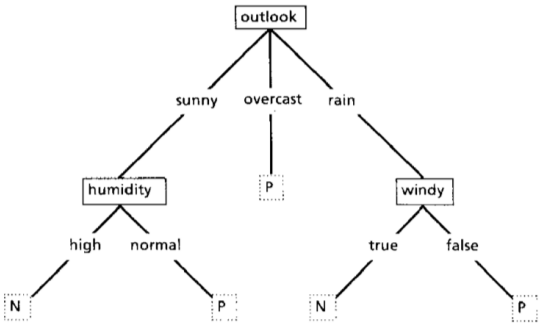

Figura 4. Exemplo de uma árvore de decisão.

É comum dividir-se uma árvore de decisão nas seguintes partes: nó raiz/nós internos, ramos e nós das folhas. O nó raiz e os nós internos são **atributos** do problema (por exemplo o vento), os ramos são os **valores** possíveis desses atributos (por exemplo se está vento ou se não está) e, por fim, os nós das folhas correspondem à **classe**, ou seja, à decisão tomada após o cálculo de todos os atributos. Os caminhos da raiz para as folhas representam **regras de classificação**.

Estes algoritmos são ideais para resolver qualquer tipo de problema (classificação ou regressão) e são também de alta precisão e estabilidade. Ao contrário dos modelos lineares, eles também mapeiam muito bem as relações não lineares.

A. Modo como exibe a capacidade de aprendizagem

Os algoritmos baseados em árvores de decisão são algoritmos recursivos, que vão dividindo os dados até obterem somente subconjuntos puros. A indução de árvores de decisão seguem uma abordagem *top-down* (TDIDT).

Estes algoritmos seguem o princípio da generalização uma vez que os novos objetos são identificados a partir das características que têm em comum com objetos já analisados.

Para se poder iniciar o treino dos objetos é necessário conhecer os atributos, respetivos valores e respetiva classe (decisão) de um conjunto de dados. Para uma melhor explicação do funcionamento geral dos algoritmos baseados em árvores de decisão, vamos recorrer a um pequeno exemplo.

O problema em curso é descobrir se o João irá praticar atletismo num determinado dia. Para tal, teremos como atributos o Tempo (valores: Sol/Normal/Chuva), a Humidade (valores: Alta/Baixa) e o Vento (valores: Sim/Não). Cada objeto irá pertencer à classe "Sim" ou à classe "Não", que indica se o João praticou atletismo ou não, respetivamente. É de notar que cada objeto deve sempre pertencer a uma classe entre um conjunto de classes mutuamente exclusivas.

A partir dos seguintes dados, vamos tentar construir, passo a passo, a árvore de decisão resultante.

Tempo?	**Humidade?**	**Vento?**	**Foi?**
Sol	Alta	Não	Sim
Sol	Baixa	Sim	Sim
Sol	Alta	Sim	Sim
Normal	Alta	Não	Não
Chuva	Alta	Sim	Não
Chuva	Baixa	Sim	Não
Chuva	Alta	Não	Sim
Chuva	Baixa	Não	Sim

Tabela II. Conjunto de dados do problema.

O primeiro passo é analisar cada um dos atributos e perceber se algum deles pode ser dividido num subconjunto puro (todos os objetos com um determinado valor pertencerem à mesma classe, por exemplo, sempre que está vento a decisão é sim).

Atributo Tempo:

Tempo?	**Foi?**
Sol	Sim
Sol	Sim
Sol	Sim

Tabela III. Objetos com valor Sol no atributo Tempo.

Tempo?	**Foi?**
Normal	Não

Tabela IV. Objetos com valor Normal no atributo Tempo.

Tempo?	**Foi?**
Chuva	Não
Chuva	Não
Chuva	Sim
Chuva	Sim

Tabela V. Objetos com valor Chuva no atributo Tempo.

Ao analisarmos o primeiro atributo do conjunto de dados, Tempo, percebemos que já encontramos pelo menos um subconjunto puro, isto é, sempre que está Sol o João vai praticar atletismo, e sempre que está Normal o João não vai. Desta forma, encontrámos o primeiro nó da árvore.

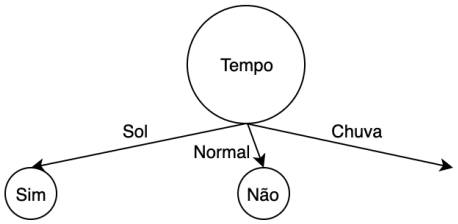

Figura 5. Início da árvore de decisão.

Em seguida, voltamos a efetuar o mesmo processo, mas desta vez já teremos que fixar a informação anterior, pois já só faz sentido analisar os dados onde o tempo é Chuva.

Atributo Humidade:

Tempo?	**Humidade?**	**Foi?**
Chuva	Alta	Não
Chuva	Alta	Sim

Tabela VI. Objetos com valor Alta no atributo Humidade, e Chuva no atributo Tempo.

Tempo?	**Humidade?**	**Foi?**
Chuva	Baixa	Não
Chuva	Baixa	Sim

Tabela VII. Objetos com valor Baixa no atributo Humidade, e Chuva no atributo Tempo.

Tal como se pode ver, não conseguimos encontrar nenhum subconjunto puro e, por isso, vamos analisar o atributo Vento.

Atributo Vento:

Tempo?	Vento?	Foi?
Chuva	Não	Sim
Chuva	Não	Sim

Tabela VIII. Objetos com valor Não no atributo Vento, e Chuva no atributo Tempo.

Tempo?	Vento?	Foi?
Chuva	Sim	Não
Chuva	Sim	Não

Tabela IX. Objetos com valor Sim no atributo Vento, e Chuva no atributo Tempo.

Obtivemos dois subconjuntos puros. Desta forma, cobrimos todos os objetos do conjunto de dados fornecido e podemos construir a árvore de decisão completa do problema.

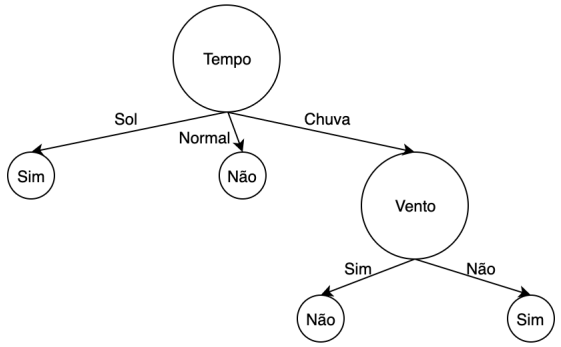

Figura 6. Árvore de decisão final.

Para sabermos se, por exemplo, o João vai praticar atletismo num dia em que está Sol mas não está Vento e a Humidade é baixa, conseguimos, através da árvore de decisão, perceber que a resposta é Sim.

Este algoritmo é uma espécie de "*Divide & Conquer*". Porém, o exemplo apresentado é bastante simples, e, em casos reais onde existem muitos dados e vários atributos com diferentes valores, a indução da árvore de decisão terá que seguir um algoritmo mais complexo e específico. A grande questão passa por descobrir em que atributo dividir o conjunto. Como é que a árvore decide onde dividir?

Esta decisão tem que ter em conta vários fatores, como por exemplo se se trata de uma classificação ou regressão, qual o tipo de variáveis de destino, entre outros. Consequentemente, utilizar o algoritmo mais adequado terá impacto direto na precisão da árvore. Atualmente, os algoritmos específicos das árvores de decisão mais utilizados são os seguintes:

- ID3 (*Iterative Dichotomiser 3*);
- C4.5 (sucessor do ID3);
- CART (Árvore de Classificação e Regressão);
- CHAID (*Chi-square automatic interaction detection*): executa divisões de vários níveis ao calcular árvores de classificação;
- MARS: estende as árvores de decisão para lidar melhor com dados numéricos;
- *Conditional Inference Trees*: abordagem baseada em estatística que usa testes não paramétricos como critérios de divisão, corrigidos para vários testes para evitar *overfitting* (excesso de ajustes).

Analisaremos com mais pormenor o algoritmo ID3. Neste algoritmo, é utilizada a Entropia para medir o grau de pureza de um subconjunto, de modo a, posteriormente, podermos decidir onde dividir a árvore. Assumindo que existem dois tipos de classes: "p" e "n", a fórmula da entropia é a seguinte:

$$I(p,n) = -P(p) * \log_2 P(p) - P(n) * \log_2 P(n)$$

Onde P(p) é a probabilidade de respostas "p" ($P(p) = \frac{p}{p+n}$) e P(n) é a probabilidade de respostas "n"($P(n) = \frac{n}{p+n}$).

No entanto, teremos que agregar a informação dos diferentes subconjuntos possíveis para conseguirmos, por exemplo, escolher apenas um subconjunto quando existe mais do que um subconjunto puro. É aqui que se utiliza a fórmula de *Information Gain*. Com esta fórmula, conseguimos obter o "número" de informações obtidas se optarmos por ramificar em A. O objetivo é maximizar este valor.

$$gain(A) = I(p,n) - \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} * I(p_i, n_i)$$

I(p,n) é a informação necessária de todo o conjunto de dados, **A** é o subconjunto do atributo onde dividimos a árvore e **v** é os valores possíveis para o atributo de A.

De forma a ilustrar a ideia, vamos regressar ao exemplo anteriormente apresentado.

Na Tabela II temos 8 diferentes objetos, onde 6 pertencem à classe "Sim" (p) e 2 pertencem à classe "Não" (n). Assim, a informação necessária para a classificação é a seguinte:

$$I(6,2) = -\frac{6}{8} * \log_2 \frac{6}{8} - \frac{2}{8} * \log_2 \frac{2}{8} = 0.811$$

Agora, vamos considerar somente o atributo Tempo, que possui os valores Sol (3 objetos), Chuva (4 objetos) e Normal (1 objeto). No caso do valor Sol, 3 pertencem à classe p e 0 pertencem à classe n. Desta forma, temos:

$$I(3,0) = -\frac{3}{3} * \log_2 \frac{3}{3} - \frac{0}{3} * \log_2 \frac{0}{3} = 0$$

(significa que o subconjunto é puro)

De forma análoga, no que diz respeito ao valor Chuva, temos:

$$I(2,2) = -\frac{2}{4} * \log_2 \frac{2}{4} - \frac{2}{4} * \log_2 \frac{2}{4} = 1$$

(significa que o subconjunto é inpuro)

Por fim, no caso de Tempo Normal, temos:

$$I(1,0) = -\frac{1}{1} * \log_2 \frac{1}{1} - \frac{0}{1} * \log_2 \frac{0}{1} = 0$$

Juntando estas informações, podemos calcular o *gain* do atributo Tempo:

$$gain(Tempo) = 0.811 - (\frac{3}{8} * 0 + \frac{4}{8} * 1 + \frac{1}{8} * 0) = 0.311$$

Calculando para os restantes atributos, obtemos:

$$gain(Humidade) = 0.811 - (\frac{5}{8} * 0.722 + \frac{3}{8} * 0.918) = 0.0155$$

$$gain(Vento) = 0.811 - (\frac{4}{8} * 1 + \frac{4}{8} * 0.811) = -0.095$$

Desta forma, o algoritmo ID3 iria escolher o Tempo para nó raiz da árvore de decisão. Para calcular o resto da árvore, seria necessário repetir o processo para os restantes dados.

No entanto, em todos os algoritmos existem adversidades que poderão surgir no processo da construção da árvore de decisão. Por exemplo, existir dois objetos que possuam valores iguais para todos os atributos, mas pertencerem a classes diferentes. Neste caso, os atributos serão denominados inadequados para o processo de treino e, consequentemente, para a tarefa de indução.

Uma forma de avaliarmos a precisão preditiva de uma árvore de decisão é usar apenas uma parte do conjunto de objetos fornecido como conjunto de treino e verificar se a árvore de decisão satisfaz os restantes objetos.

Como seria de esperar, o recurso a árvores de decisão para a resolução de problemas tem também as suas vantagens e desvantagens. Os principais pontos positivos são os seguintes: são fáceis de entender, uma vez que a visualização da árvore resultante é bastante intuitiva para qualquer pessoa, mesmo que não tenha conhecimentos de estatística; são úteis na exploração dos dados, visto que são uma das formas mais rápidas de identificar as variáveis mais significativas e a relação entre duas ou mais variáveis; têm uma menor necessidade de limpar dados em comparação com outras técnicas de modelação e, até um certo nível, não são influenciadas por *outliers* nem por *missing values*. Outra das vantagens das árvores de decisão é não serem restritas a um tipo de dados, pois podem manipular tanto variáveis numéricas como categóricas. Para além disso, também não requerem muito esforço na preparação dos dados e a sua performance não é afetada por relações não lineares entre parâmetros.

Por outro lado, estas árvores têm também vários pontos fracos. Um dos quais é o facto de não serem adequadas para variáveis numéricas contínuas, pois ao trabalhar com estas variáveis a árvore de decisão perde informações (ao categorizar variáveis em diferentes categorias). As árvores de decisão podem também ser instáveis porque pequenas variações nos dados podem resultar na geração de uma árvore completamente diferente (variância). Uma árvore pode ainda sofrer de *overfiting* quando começa a crescer muito (muitos nós), uma vez que se irá tornar muito específica (*singletons*) e não conseguirá generalizar muito bem os casos que ainda não tiver visto antes. A técnica *pruning* (remover nós das árvores, processo oposto de *splitting*) ajuda a suavizar este problema. No lado oposto, pode também acontecer *underfiting*, quando não tem atributos suficientes para tomar uma decisão. Nas árvores de decisão, por se tratar de um algoritmo *greedy*, não é garantido que obtemos a árvore ótima, mas isso pode ser atenuado pelo treino de várias árvores.

B. Ferramentas de desenvolvimento existentes

Existem várias linguagens de programação que possuem bibliotecas que permitem implementar árvores de decisão de uma forma simples. Entre as quais "ctree", "rpart", "randomForest" e "tree" em R. Em Python, "Scikit-learn" é uma biblioteca responsável por diversos algoritmos de *Machine Learning*, incluindo as *Decision Trees*.

Para além destas opções, existem também vários softwares que permitem criar árvores de decisão, como o *SmartDraw*, *IBM SPSS Decision Trees*, *LucidChart Decision Tree Software*, *RapidMiner*, *Weka* e *Zingtree Interactive Decision Tree Template*. O Microsoft Excel é ainda outra opção.

Todos estes recursos têm um objetivo comum: permitir que qualquer *data scientist* consiga facilmente implementar a sua própria árvore de decisão.

C. Soluções no mercado baseadas em Decision Trees

As *Decision Trees*, devido à sua simplicidade e ao facto de darem contexto à solução do problema, são usadas numa ampla gama de indústrias e disciplinas, incluindo planeamento civil, setor energético, financeiro, engenharia, saúde, farmacêutico, educação, direito e negócios.

No que diz respeito à área da medicina, existem neste momento soluções no mercado que ajudam na priorização do tratamento de pacientes de emergência, usando um modelo preditivo baseado em fatores como idade, pressão arterial, sexo, localização, grau da dor, entre outras medidas. Para além disso, ajudam no diagnóstico de uma condição médica a partir de sintomas, onde as classes das *Decision Trees* podem ser os vários estados de doença ou as possíveis terapias. Atualmente no mercado, estas árvores permitem ainda a análise da Síndrome da

Morte Súbita Infantil (SMSI). Mais direcionado para a vertente da farmacologia, as árvores de decisão desenvolvem análises da eficácia de medicamentos.

No ramo dos negócios, as árvores de decisão são atualmente utilizadas para avaliar oportunidades de expansão de uma marca para uma empresa recorrendo a dados de históricos de vendas; determinar prováveis compradores de um produto utilizando dados demográficos que permitam a segmentação do público-alvo de um anúncio (no caso de haver um orçamento limitado) e ainda prever a probabilidade da falta de cumprimento de pagamento de empréstimos, usando modelos preditivos gerados a partir do histórico do cliente.

Em astronomia, o uso das árvores de decisão permite a filtragem de ruído das imagens do Telescópio Espacial Hubble.

Relativamente à biologia e à biomedicina, a análise de aminoácidos no Projeto Genoma Humano e a identificação dos recursos necessários nos dispositivos médicos implantáveis, são algumas das soluções que existem atualmente no mercado.

Noutros ramos, podemos encontrar também projetos de otimização de processos de produção de maquinaria, avaliação de materiais químicos e decidir a partir de observações atmosféricas se uma grande tempestade é possível, pouco provável ou provável.

Estes são apenas alguns dos vários exemplos que existem atualmente no mercado, mas poderíamos apresentar muitos outros serviços. Isto prova que as árvores de decisão são um algoritmo realmente muito utilizado no mercado atual e que proporcionam uma melhoria num vasto leque de indústrias.

V. CONCLUSÃO

Nos sistemas de aprendizagem a escolha do algoritmo é muito importante, visto que influencia todo o processo futuro. Desta forma, após analisarmos com detalhe os algoritmos *Case Based Reasoning*, *Genetic Algorithms* e *Decision Trees*, conseguimos perceber que cada um deles tem características próprias que os fazem mais ou menos indicados para diferentes áreas de aplicação.

Concluímos desta forma que o ramo do CBR tem muitas vertentes por onde pode ser explorado. A favor da aplicação deste tipo de método temos o facto de ser bastante rápido e eficiente pois não perde tempo a calcular respostas do zero, permite obter respostas em qualquer domínio do conhecimento mesmo que o ator que está a manusear o software não esteja por dentro dos conceitos, a lembrança do passado é muito útil na identificação de erros futuros, os casos muitas vezes ajudam a perceber o domínio de conhecimento em que estamos e, por fim, o CBR dá mais importância aos pontos mais fulcrais do problema. No entanto nem tudo é bom, duas grandes falhas apontadas a este sistema é a aceitação dos resultados sem existir espírito crítico e a probabilidade de existirem soluções tendenciosas.

Um dos desafios grandes nesta área é a escolha do conjunto de dados mais apropriados, quer seja pela variedade ou pela escassez dos mesmos. Outro fator muito importante no futuro desta abordagem é a utilização da mesma, nos primórdios do desenvolvimento apenas de forma académica mas, neste momento, está a ser adotada cada vez mais na indústria.

Perante outros tipos de algoritmos de procura, *Genetic Algorithms* (Algoritmos Genéticos) são bastante mais eficientes e precisos, devido à sua componente aleatória na procura de soluções juntamente com os seus métodos de desenvolvimento e evolução das soluções encontradas. Mas, apesar de ser um algoritmo que apresenta uma grande capacidade de aprendizagem e de evolução do seu conhecimento, esta apenas consegue se focar na otimização de valores, áreas onde outro tipo de algoritmos de aprendizagem são menos eficientes. Uma grande vantagem deste é a autonomia que possui (*unsupervised learning*), pois estando a função objetivo bem definida, o crescimento da população é feito sem qualquer intervenção externa, chegando eventualmente à solução.

As *Decision Trees*, apesar de poderem sofrer de *overfitting* e perderem um pouco de precisão, são muito fáceis de entender e de interpretar. Principalmente por podermos ver a árvore completa, conseguimos ter acesso não só à decisão final, mas também a todo o contexto da solução. Em áreas como a saúde, é essencial o profissional de saúde poder ter acesso a todo esse enquadramento da solução. Isto torna as árvores de decisão num dos mais poderosos métodos de *machine learning* e uma excelente opção para todos os que procuram um algoritmo estável, simples e eficiente.

REFERÊNCIAS

[1] Aamodt, A., and Plaza, E. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications, Vol. 7 Nr. 1* (Mar 1994), 39–59.

[2] Behbahan, N. S., and Behbahan, M. S. An introduction to the decision trees and its types. *International journal of advanced research in IT and engineering*.

[3] Bodenhofer, U. *Genetics Algorithms: Theory and Applications*, 3ª ed. 2003.

[4] Brid, R. S. Decision trees - a simple way to visualize a decision, Oct 2018.

[5] Costa, P. R., Carroll, P., Mauceri, S., and Pallonetto, F. A genetic algorithm for the green vehicle routing problem.

[6] Dahiya, S., and Gupta, M. Study of case-based reasoning system. *IJSRD - International Journal for Scientific Research Development 2*, 07 (Jul 2014).

[7] Fatmawatie, B. D., and Baizal, Z. K. A. Tourism recommender system using case based reasoning approach (case study: Bandung raya area). *Journal of Physics: Conference Series 1192* (2019), 012050.

[8] Forsey, C. Decision trees: The simple tool that'll make you a radically better decision maker.

[9] Goldberg, D. E. *Genetics Algorithms in Search, Optimization Machine Learning*, vol. 1st. Addison-Wesley Longman Publishing Co., Inc., 1989.

[10] Holt, A., Perner, P., and Bichindaritz, I. Medical applications in case-based reasoning. *The Knowledge Engineering Review 00:0* (Sep 2005), 1–4.

[11] Masum, A. K. M., Shahjalal, M., Faruque, M. F., and Sarker, I. H. Solving the vehicle routing problem using genetic algorithm. *International Journal of Advanced Computer Science and Applications* (Aug 2011).

[12] Quinlan, J. R. *Induction of decision trees*. New South Wales Institute of Technology, School of Computing Sciences, 1985.