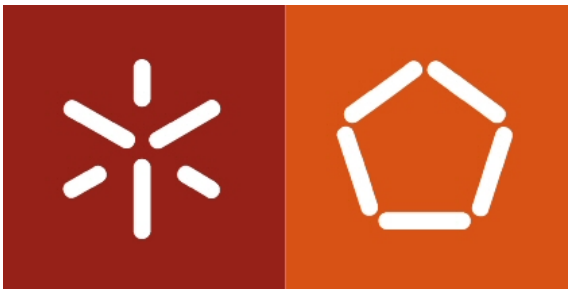

Universidade do Minho

Mestrado em Engenharia Informática

Sistemas Inteligentes

Aprendizagem e Extração de Conhecimento

Sistemas de Aprendizagem

Inês Alves (A81368)
José Pinto (A80741)
Ricardo Milhazes (A81919)

21 de Outubro de 2019

Conteúdo

1 Introdução

O presente trabalho foi desenvolvido no âmbito da Unidade Curricular de Aprendizagem e Extreação de Conhecimento, integrada no perfil de especialização de Sistemas Inteligentes e cujo objetivo passa por abordar e compreender o funcionamento da **Aprendizagem por Reforço**, das **Redes Neuronais Artificais** e **Árvores de Decisão**.

Daqui em diante, o presente relatório irá elaborar cada tópico tendo em conta as suas características, algoritmos de aprendizagem, ferramentas de desenvolvimento e aplicações possíveis no mercado.

2 Reinforcement Learning

2.1 Descrição Característica

Reinforcement Learning é um paradigma de *Machine Learning* centrado na tomada de decisões por parte de um agente num ambiente, com o objetivo de maximizar uma determinada recompensa.

Este ambiente pode ser representado através de um Processo de Decisão de Markov (conceito explorado no ponto 1.1.1). A principal característica deste paradigma é a aprendizagem através do estudo do efeito que uma ação a tem no estado s do ambiente.

Em vez de, como no Supervised Learning, a aprendizagem ser feita através da correspondência entre dados de input e o output correto, aqui o modelo evolui no sentido de maximizar os valores obtidos para uma dada função de recompensa, calculada para cada transição de um estado s para s' devido à tomada da ação a pelo agente.

$$R_a(s, s')$$

Há vários critérios segundo os quais pode ser escolhida a ação a tomar por parte do agente, como por exemplo a definição de uma função de política que retorne a probabilidade de uma determinada opção, a escolha da ação correpondente à maior recompensa esperada, ou a força bruta (calcular o estado seguinte para todas as ações possíveis e a respetiva recompensa, o que por vezes pode não ser possível).

Outro aspeto interessante do Reinforcement Learning é o equilíbrio entre o aproveitamento do conhecimento adquirido e a exploração de novas estratégias ou de partes do ambiente ainda descohecidas para o agente (o que muitas vezes é assegurado introduzindo uma componente estocástica no proceso de tomada de decisão).

2.1.1 Processos de Decisão de Markov

Uma vez que o ambiente num problema de Reinforcement Learning pode em grande parte dos casos ser descrito utilizando o conceito de Processo de Decisão de Markov (PDM), é oportuno explorar esse conceito quando procuramos aprender sobre esta àrea de Machine Learning. O PDM consiste num processo estocástico de controlo de decisão, modelando matematicamente contextos de tomada de decisão em que os reultados são parcialmente aleatórios. Assim, podemos definir um PDM à custa destes 4 elementos:

- S - Um conjunto finito de estados.
- A_s - Um conjunto finito de ações disponíveis no estado s.
- $P_a(s, s') = Pr(s_{t+1} = s' \quad | \quad s_t = s, \quad a_t = a)$ - Probabilidade de a ação a no estado s resultar no estado s'.
- $R_a(s, s')$ - a recompensa imediata obtida pela transição de s para s' através da ação a.

É importante mencionar que apesar do poder deste conceito, no sentido em que nos permite descrever e trabalhar com processos de tomada de decisão, a maioria dos algoritmos de Reinforcement Learning mais utilizados são *model free*, ou seja, não fazem uso do modelo de transição (nome geralmente utilizado para designar a distribuição probabilística $P_a(s, s')$ e a função de recompensa a si associada).

2.2 Exemplos de Algoritmos de Reinforcement Learning

A descrição feita até agora do objeto de estudo sugere a existência de diferentes abordagens ou estratégias de implementação de Reinforcement Learning. Estes algoritmos podem ser divididos em duas partes:

- Atualização do critério de tomada de decisão (como por exemplo em certos casos a evolução de ruma rede neuronal)
- Atualização da função valor (que relaciona a recompensa imediata da última transição com o valor anteriormente calculado para a evolução desde o estado inicial seguindo a política de tomada de decisão em questão)

Entre os algoritmos mais populares escolhemos dois dos mais populares e versáteis de que passamos a falar.

2.2.1 Q-Learning

O Q-Learning é um algoritmo *model free* com o objetivo de chegar a uma política de tomada de decisão. O seu nome deve-se ao facto de um dos seus elementos centrais ser a função

$$Q(s_t, a_t) = (1 - \alpha) * Q(s_{t-1}, a_{t-1}) \quad + \quad \alpha * (r_t + \gamma * \max_a(Q(s_{t+1}, a)))$$

em que α é a *learning rate* (entre 0 e 1, regula o ritmo de aprendizagem para que a política de decisão não seja demasiado vulnerável a um número reduzido de transições de estado), γ é o *discount factor* (que, também entre 0 e 1, regula o peso das recompensas futuras na decisão atual). Esta função representa-se pela letra Q porque de serta forma representa a Qualidade de uma ação para um determinado estado, e tem de ser inicializada aquando da partida de um estado inicial (porque não houve iterações anteriores). Aqui podemos visualizar em pseudocódigo o esboço de uma implementação deste algoritmo.

```
Initialize Q(s, a) arbitrarily
Repeat (for each episode):
   Initialize s
   Repeat (for each step of episode):
      Choose a from s using policy derived from Q
         (e.g., ε-greedy)
      Take action a, observe r, s'
      Q(s, a) <-- Q(s, a) + α [r + γ max_α,Q(s', a') - Q(s, a)]
      s <-- s';
   until s is terminal
```

2.2.2 SARSA - State, Action, Reward, State, Action

O algoritmo SARSA é semelhante ao anteriormente mencionado, no entanto, trata-se daquilo a que é chamado um algoritmo *on-policy*, uma vez que tem em consideração (no fator das recompensas futuras) as mudanças na política de decisão que ocorrem na próxima transição. Assim, apesar de o algoritmo não convergir para uma política de decisão tão rapidamente, este acaba por explorar mais o ambiente do problema e é mais conservativo em casos em que as ações que retornam a maior recompensa imediata (e que seriam escolhidas no Q-Learning) se podem suceder de recompensas bastante negativas (um fenómeno demonstrado num problema conhecido como *cliff walking*). Assim, em determinados problemas este algoritmo pode ser uma melhor opção do que o anterior (sobretudo se houver uma componente estocástica nas transições de estado).

```
Initialize Q(s, a) arbitrarily
Repeat (for each episode):
   Initialize s
   Choose a from s using policy derived from Q
         (e.g., ε-greedy)
   Repeat (for each step of episode):
      Take action a, observe r, s'
      Choose a' from s' using policy derived from Q
         (e.g., ε-greedy)
      Q(s, a) <-- Q(s, a) + α[r + γQ(s', a')- Q(s, a)]
      s <-- s'; a <-- a';
   until s is terminal
```

2.3 Ferramentas de Desenvolvimento

Nos dias que correm, já muitas existem muitas bibliotecas de machine learning (com funcionalidades de reinforcement learning, que cada vez é mais popular devido à visibilidade que tem ganho devido à sua aplicação a problemas como por exemplo a criação de inteligências artificiais capazes de jogar videojogos), e estas encontram-se disponíveis em muitas linguagens. Exemplos dessas ferramentas são:

- Tensorflow - biblioteca open source da Google com inúmeras ferramentas tanto de alto como de baixo nível, direcionadas a bastantes vertentes do machine learning.
- Keras - biblioteca de redes neuronais artificiais, de alto nível, que pode ser integrada e utilizada em conjunto com o Tensorflow ou outras bibliotecas de machine learning
- OpenAI Gym - biblioteca com a finalidade de facilitar a investigação e benchmarking de soluções para problemas utilizando diferentes técnicas de machine learning e inteligência artificial.

2.4 Soluções Existentes

Bastantes áreas de problemas reais já têm vindo a benificiar com o uso de reinforcement learning duas das quais são:

- Jogos - são os casos mais conhecidos, devido à complexidade de alguns problemas e na facilidade em compreender as recompensas subjacentes aos objetivos dos mesmos. Casos como o do AlphaGo (no jogo de tabuleiro Go) ou o OpenAI Five (no videojogo Dota 2) contribuiram fortemente para a popularidade do Reinforcement Learning.
- Recomendações personalizadas - apesar de ser um conjunto de problemas que costuma ser abordado através de uma grande variedade de técnicas de inteligência artificial, o reinforcement learning começa a ganhar destaque nesta área. Um possível fator que influencie a recompensa pode ser, por exmeplo, o número de *clicks* por parte de utilizadores.

3 Artificial Neural Networks

3.1 Descrição Característica

Artificial Neural Networks (ANN) ou Redes Neuronais Artificiais (RNA), também conhecidas como sistemas computacionais representativos de conhecimento não simbólico (ou conexionista), apresentam um modelo inspirado no Sistema Nervoso Central do ser humano (organismo inteligente), sendo estes capazes de realizar uma apredizagem automática, bem como o reconhecimento de padrões, pois adquirem conhecimento através da experiência.

A influência do cérebro humano na elaboração deste tipo de sistema de aprendizagem é facilmente justificada se considerarmos que o cérebro é um sistema de processamento de informação altamente complexo, não-linear e paralelo.

Estas redes têm sido usadas para resolver uma grande variedade de tarefas que são difíceis de resolver utilizando programação baseada em regras comuns, incluindo visão por computador e reconhecimento de voz.

As Redes Neuronais Artificiais são um método para solucionar problemas através da simulação do cérebro humano, ou seja, aprendem com os erros e fazem novas descobertas.

Sendo assim, podemos definir uma RNA como uma estrutura interligada de unidades computacionais, designadas **neurónios**, com capacidade de aprendizagem.

Um neurónio biológico consiste numa única célula capaz de realizar um forma simples de processamento. Cada neurónio é estimulado por uma ou mais ligações vindas de outros neurónios, chamadas **sinapses**, dependendo o sinal produzido tanto da força das ligações como da sua natureza. Este sinal é propagado ao longo do **axónio** indo, por sua vez, estimular outros neurónios. O funcionamento dos neurónios artificiais baseia-se, na generalidade dos casos, neste modelo simplificado dos neurónios biológicos.

Tendo por base esta relação entre uma RNA e o cérebro humano, é fundamental identificar dois aspetos idênticos entre ambos:

- o conhecimento é adquirido a partir de um ambiente, através de um processo de aprendizagem;
- o conhecimento é armazenado nas **conexões**, também designadas ligações ou sinapses, entre nodos (ou nós).

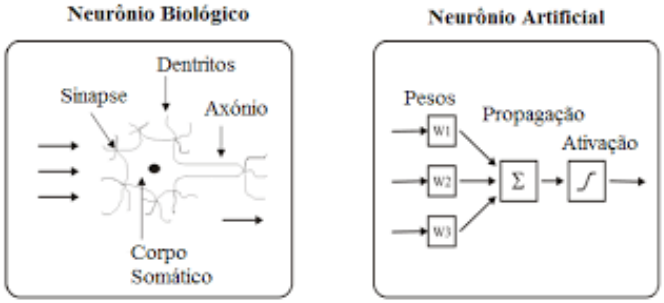

Figura 1: Neurónio Biológico e Neurónio Artificial

A fim de percebermos melhor o que são as RNAs, é importante conhecermos como funcionam os neurónios.

3.1.1 O Comportamento dos Neurónios

Os neurónios são as células que formam o nosso cérebro. Elas são compostas basicamente por três partes: os dentritos, que captam informações ou do ambiente ou de outras células; o corpo

celular, responsável pelo processamento das informações; e um axónio, para distribuir a informação processada para outros neurónios ou células do corpo. No entanto, uma célula dificilmente trabalha sozinha. Quanto mais células trabalharem em conjunto, mais elas podem processar e mais eficaz se torna o trabalho realizado pelas mesmas. Logo, para um melhor rendimento do sistema são necessários muitos neurónios.

3.1.2 Dos neurónios às Redes Neuronais

Foi a pensar em como os neurónios biológicos trabalham que os cientistas desenvolveram neurónios artificiais. Cada um tem dois ou mais recetores de entrada, responsáveis por perceberem um determinado tipo de sinal. Estes também possuem um corpo de processadores, responsável por um sistema de *feedback* que modifica a sua própria programação dependendo dos resultados de entrada e saída. Finalmente, eles possuem uma saída binária para representar a resposta "Sim" ou "Não", dependendo, por sua vez, do resultado do processamento.

Um neurónio artificial é capaz de um único processamento. Cada entrada recebe apenas um tipo de sinal ou informação (excitativo, inibidor ou nulo). Como um neurónio pode possuir várias entradas, então ele pode receber diferentes sinais. Porém, ligar vários neurónios similares em rede, faz com que o sistema consiga processar mais informações e oferecer melhores resultados.

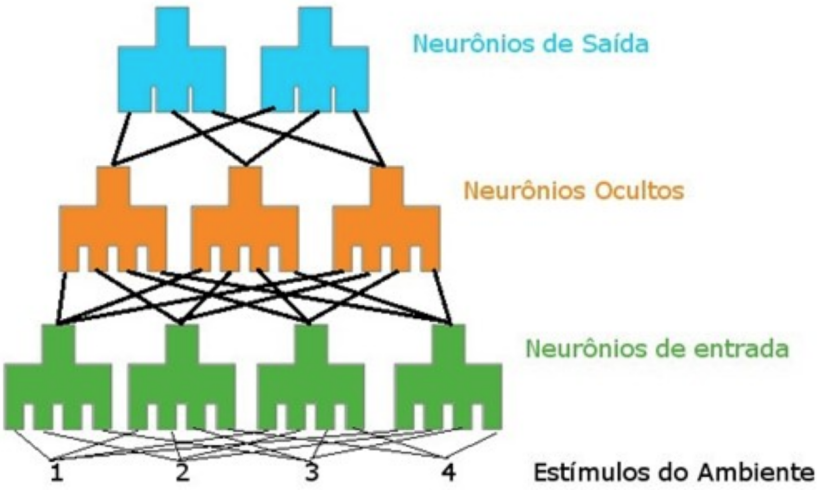

Figura 2: Exemplo de sistema

3.2 Arquiteturas de RNAs

- **Arquitetura Feedforward de uma só camada (RFSC):**
  - a informação flui sempre para a frente;
  - tem 1 camada de entrada, cujos valores de saída são fixados externamente;
  - tem 1 camada de saída;
  - a camada de entrada não é contabilizada como camada numa RNA devido ao facto de nesta não se efetuarem quaisquer cálculos.
- **Arquitetura Feedforward de várias camadas (RFMC):**
  - esta arquitetura é semelhante à anterior, no entanto, tem 1 ou mais camadas intermédias;

  - ao se acrescentarem mais camadas intermédias está-se a aumentar a capacidade da rede em modelar funções de maior complexidade.

- **Arquitetura Recorrente:**
  - uma saída de um elemento influencia, de algum modo, a entrada para esse mesmo elemento;
  - esta arquitetura implementa ciclos, pelo que não produz os melhores resultados.

3.3 Capacidade de Aprendizagem

Um dos objetivos deste tipo de sistema centra-se em criar um modelo com capacidade de resolver problemas com base em conhecimento passado ou dados sobre a resolução de outros problemas. Deste modo, a sua capacidade de aprendizagem torna-se fundamental para um melhor desempenho da rede.

Segundo Mendel e McClaren o conceito de aprendizagem é definido da seguinte maneira:

"Aprendizagem é um processo pelo qual os parâmetros livres de uma rede neuronal são adaptados através de um processo de estimulação pelo ambiente na qual a rede está inserida. O tipo de aprendizagem é determinado pela maneira pela qual a modificação dos parâmetros ocorre."

Todo este processo de aprendizagem é executado a partir de um conjunto de regras bem definidas, a que chamamos de **algoritmo de aprendizagem** ou de treino.

A cada resultado correto, os neurónios envolvidos no processamento ganham um ponto e aquela rede é reforçada. Assim, o sistema cria a rotina de seguir o caminho com mais pontos. Quanto mais tentativas, mais aprimorado fica o sistema, chegando ao final da sua aprendizagem a executar tarefas quase sem erros.

A grande vantagem disto é que, para executar tarefas, uma rede neuronal não precisa de guardar instruções e executá-las de forma lógica, como num computador tradicional. Ao invés disso, a rede aprende o que é preciso ser feito e executa a função. Desta forma, uma mesma rede, se ela for capacitada com os neurónios necessários para tal, é capaz de executar várias funções diferentes, independentemente do espaço em memória.

Uma vez que as RNAs são utilizadas em vários contextos, existe uma variedade substancial de algoritmos de aprendizagem. Cada um deste algoritmos tem as suas próprias vantagens, que diferem entre si principalmente na forma como os pesos são modificados e que são adaptados com base no objetivo final do problema em questão. Para além disso, os algoritmos são distinguidos também na forma como operam com o ambiente, tornando-se assim fundamental associar o conceito de **paradigma de aprendizagem**.

3.3.1 Paradigmas de Aprendizagem

- **Aprendizagem Supervisionada:** neste paradigma, é efetuada uma comparação entre o valor desejado e o valor de saída da rede, originando um erro. O erro resultante é utilizado para, de alguma forma, ajustar os pesos da rede, para que este erro seja reduzido. Podemos dizer que este faz uso de um "professor"que indica à rede a resposta desejada para um determinado padrão de entrada;

- **Aprendizagem por Reforço:** fornece-se uma indicação sobre se a resposta da rede é correta ou não, tendo a rede de usar esta informação para melhorar a sua eficácia;

- **Aprendizagem Não-Supervisionada:** neste paradigma, não há nenhuma informação externa acerca da resposta correta. A aprendizagem é feita pela descoberta de características nos dados de entrada.

3.3.2 Regras de Aprendizagem

- **Hebbian:**
  - muito utilizada em aprendizagem sem supervisão;
  - depende do tempo e da localização;
  - se 2 neurónios adjacentes são ativados simultaneamente (sofrem variações no mesmo sentido), o peso da ligação aumenta exponencialmente;
  - se 2 neurónios adjacentes são ativados de forma assíncrona (variações ocorrem em sentidos opostos), o peso da ligação diminui progressivamente;
  - modificações nos pesos tendem a cooperar.
- **Competitiva:**
  - as saídas dos nodos da mesma camada compete entre si para se tornarem ativas;
  - o neurónio com maior output para um dado input é declarado vencedor, sendo o único a alterar os pesos.
- **Estocástica:**
  - os pesos são ajustados de um modo probabilístico.
- **Baseada na memória:**
  - atem em consideração as experiências passadas, que são guardadas em memória e, posteriormente, aplicadas em casos semelhantes.
- **Gradiente Descendente:**
  - esta técnica está diretamente ligada à aprendizagem com supervisão e é muito utilizada;
  - os pesos iniciais são atrbuídos aleatoriamente;
  - os inputs são processados pela rede e comparados com o output desejado;
  - o erro atua como um mecanismo de controlo, onde os sucessivos ajustamentos deste vão originar melhores respostas.

3.4 Ferramentas de Desenvolvimento

Em seguida apresentamos um conjunto de algumas ferramentas que permitem o desenvolvimento de RNAs:

- **JustNN:** ferramenta simples e intuita que permite construir e testar redes neuronais artificiais a partir de um conjunto de dados;
- **MatLAB:** com o pacote *Neural Network Toolbox* fornece algoritmos e modelos capazes de criar, treinar, visualizar e simular redes neuronais. É possível realizar a classificação, regressão, agrupamento, redução de dimensão e modelagem do sistema;
- **R-Studio:** através do pacote *neuralnet* permite construir e gerir redes neuronais, utilizando variações da técnica de *backpropagation* que consiste em utilizar a descida do gradiente para ajustar os parâmetros das redes e assim melhorar o desempenho nos pares de entrada-saída.

3.5 Soluções no Mercado

Atualmente, no mercado, é possível encontrar um conjutno alargado de aplicações reais onde as RNAs são utilizadas. Como tal, apresentamos os seguintes exemplos:

- **Reconhecimento Facial:** o facto das RNA estarem organizadas em camadas e possuírem uma grande capacidade de aprendizagem, tornam-se essenciais quando o objetivo é a obtenção de padrões. São também utilizadas no reconhecimento de voz e tratamento de imagens;
- **Previsões no mercado financeiro:** as RNAs podem servir de apoio a decisão de compra e venda de ações;
- **Identificação de Fraudes com cartão de crédito:** um banco americano de seu nome *Mellon Bank* instalou um sistema de deteção de fraudes com cartão bancário baseado nas técnicas das RNAs e os prejuízos evitados foram bastante elevados. Desde aí, outros bancos começaram a usar estes sistemas como forma de controlo de fraudes.

São ainda exemplos das aplicações destas redes as aplicações meteorológicas, análise e processamento de sinais, entre outras.

4 Decision Trees

4.1 Descrição Característica

Uma *Decision Tree* representa um modelo de apoio à decisão em forma de árvore. Estes modelos apresentam atributos que são escolhidos tendo em conta o objeto em estudo e a tarefa para o classificar. Cada um destes atributos têm um conjunto de valores que servirão para determinar a classe correspondente ao objeto em estudo através das *classification rules*.

Ainda que seja uma forma relativamente simples de representar conhecimento adquirido sendo que contém pouca expressividade no seu conteúdo, serve perfeitamente para resolver problemas difíceis de uma forma prática.

Podemos observar um exemplo bastante simples de uma *Decision Tree*:

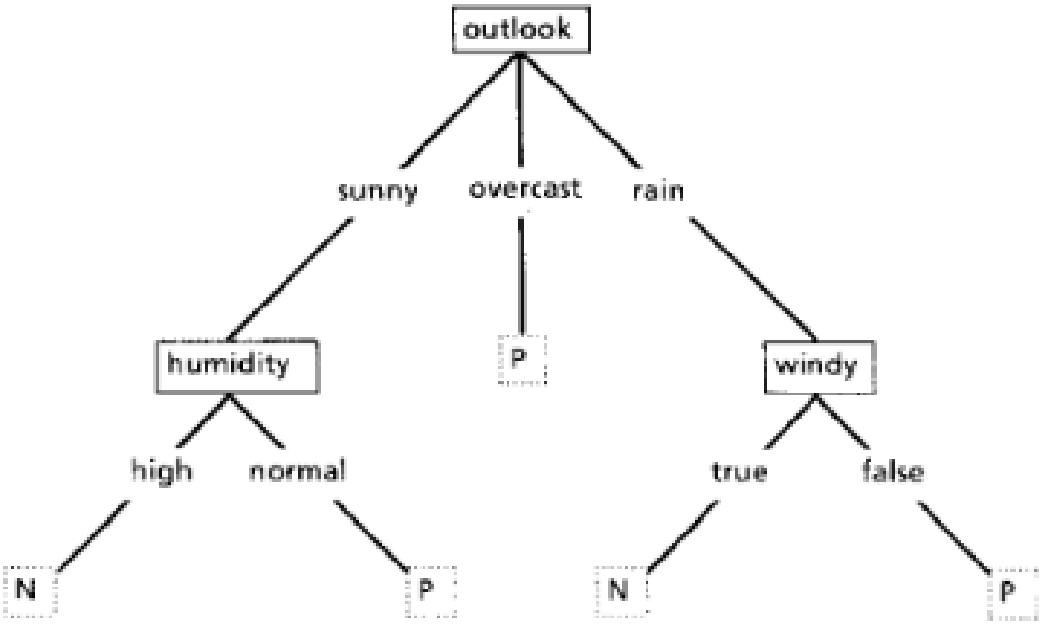

Figura 3: *Simple Decision Tree*

Neste caso os atributos são: *outlook*, *humidity* e *windy*.

Os valores respetivos aos atributos desta *Decision Tree* são: *sunny*, *overcast* e *rain*; *high* e *normal*; *true* e *false*.

As classes são **P** e **N** em que **P** poderia ser algo positivo e **N** algo negativo.

4.2 De que modo exibe a capacidade de aprendizagem

A palavra chave para descrever o modo que uma *Decision Tree* tem de exibir a sua capacidade de aprendizagem é: **Indução**.

Através de um *training set* e da aplicação de técnicas como *Recursive Partitioning* (ver secção 3.2.1), uma *Deciding Tree* tem a capacidade para apresentar um conjunto de *classification rules* que irão determinar a classe de um objecto tendo em conta o valor dos seus atributos.

É necessário ter uma atenção redobrada no que conta aos atributos utilizados para caracterizar os objectos no *training set*, sendo que se os atributos forem inadequados, a informação poderá não ser a suficiente para, por exemplo, diferenciar dois objetos que têm valores de atributos idênticos e uma classe diferente. Se não for possível fazer esta diferenciação, não se poderá determinar a classe de um objeto com valores de atributos idênticos aos dois que estão presentes no training set, sendo que existirá um conflito de classes.

No.	Attributes				Class
	Outlook	Temperature	Humidity	Windy	
1	sunny	hot	high	false	N
2	sunny	hot	high	true	N
3	overcast	hot	high	false	P
4	rain	mild	high	false	P
5	rain	cool	normal	false	P
6	rain	cool	normal	true	N
7	overcast	cool	normal	true	P
8	sunny	mild	high	false	N
9	sunny	cool	normal	false	P
10	rain	mild	normal	false	P
11	sunny	mild	normal	true	P
12	overcast	mild	high	true	P
13	overcast	hot	normal	false	P
14	rain	mild	high	true	N

Figura 4: Exemplo de um *training set*

4.2.1 Recursive Partitioning

Recursive Partitioning consiste em dividir uma população em sub-populações tendo em conta um determinado critério. Este processo acaba quando se atinge um determinado critério de paragem (daí ser um processo recursivo).

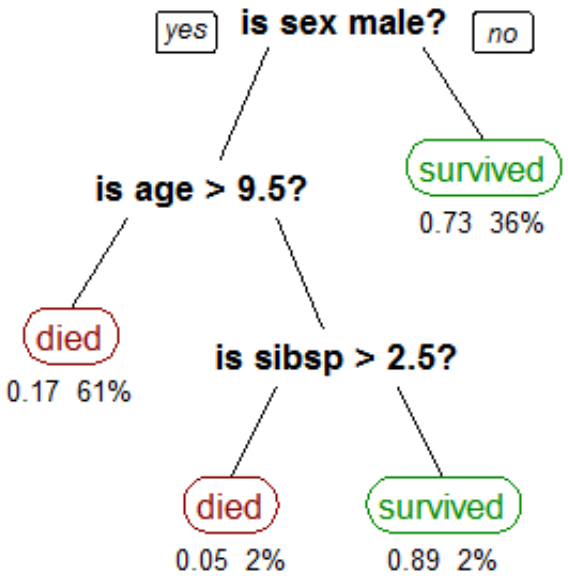

Figura 5: *Exemplo de Recursive Partitioning*

Os métodos mais conhecidos utilizam algoritmos do estilo **ID3**. O algoritmo **ID3** consiste em, inicialmente, pesquisar pelo atributo que apresenta menos entropia considerando este a raiz da árvore e, de seguida, definir os próximos nodos tendo em conta o mesmo critério.
É importante definir critérios de paragem para as sub-populações, como por exemplo: todos os atributos já foram seleccionados ou todos os elementos da sub-população pertencem à mesma classe (isto leva a transformar o nodo num *leaf-node*).

Este tipo de métodos têm bastantes vantagens sendo que são métodos bastante intuitivos e permitem, na maior parte das vezes, mais precisão, mas também trazem desvantagens sendo que a principal é a falta de adaptação a casos novos tendo em conta um conjunto de dados anteriormente utilizados. Uma forma prática de resolver isto é através da redução do tamanho da árvore, tornando-a mais geral e mais flexível a casos novos (*prunning*).

4.3 Ferramentas de desenvolvimento

Para desenvolver *Decision Trees* é necessário que o *software* forneça bibliotecas que classifiquem as árvores bem como algoritmos de execução destas.

Um exemplo de biblioteca que proporciona este tipo de funcionalidades é o "Scikit-Learn". Basta utilizar a classe "DecisionTreeClassifier()"ou a classe "DecisionTreeRegressor()"para treinar a árvore com os dados anteriormente importados ou criados.

```
>>> from sklearn import tree
>>> X = [[0, 0], [1, 1]]
>>> Y = [0, 1]
>>> clf = tree.DecisionTreeClassifier()
>>> clf = clf.fit(X, Y)
```

Figura 6: Código para *Decision Tree* dum modelo de classificação

De seguida o modelo permite prever a classe de uma amostra colocado como input.

```
>>> clf.predict([[2., 2.]])
array([1])
```

Figura 7: Código para prever classe de amostra

Existem outras formas de programar *Decision Trees* como por exemplo em **R** sendo que a própria linguagem tem uma *package tree*. Também existe o software **Weka** que foi escrito em **Java** e que tem um painel denominado *Classify Panel* que permite a aplicação de modelos de classificação e regressão a dados originando *Decision Trees*.

4.4 Soluções existentes no mercado

Decision Trees são aplicadas em diversas situações devido à sua precisão e facilidade de utilização.
Apresentamos então alguns exemplos:

1. Diagnósticos Médicos - Através da observação de casos anteriores é possível criar *classification rules* através dos sintomas apresentados nesses casos, sendo que o resultado poderá ser, por exemplo, o estado da doença do paciente ou até mesmo uma possível terapia para os sintomas.

2. Jogos de estratégia - Através da leitura do jogo podemos observar diferentes indicadores como, por exemplo, se a posição onde nos encontramos de momento no tabuleiro é uma posição favorável ou não.

3. Metereologia - Tendo em conta diferentes observações atmosféricas podemos determinar, por exemplo, se vai chover ou não.

Para facilitar a compreensão do uso das *Decision Trees*, criamos um exemplo de *Decision Tree* que serviria perfeitamente para utilizar no dia-a-dia.

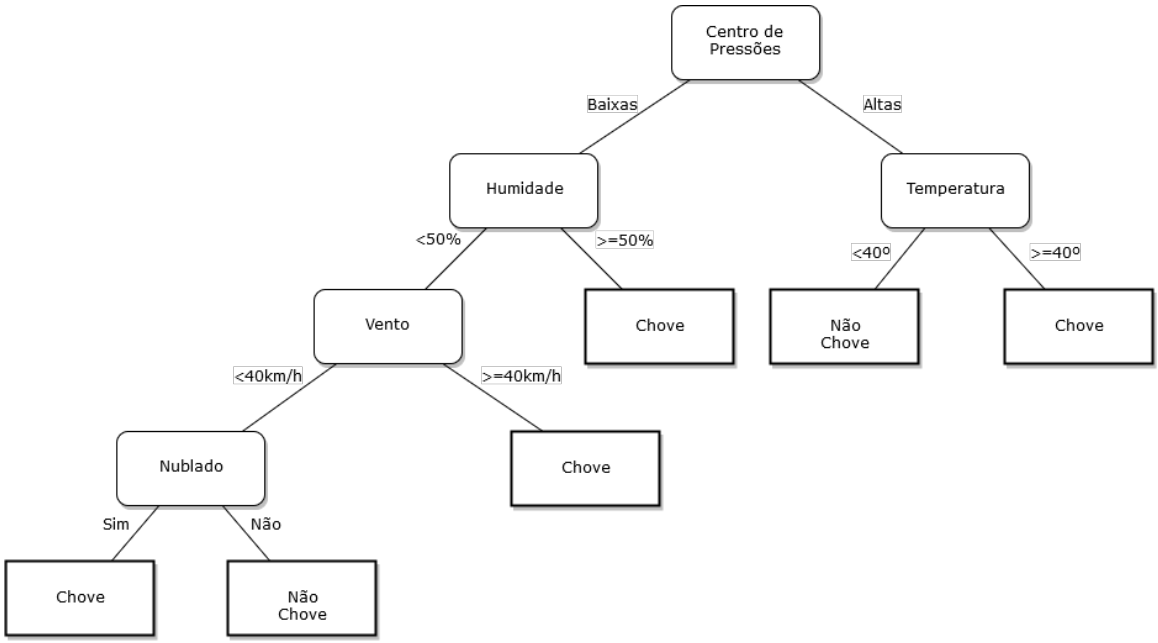

Figura 8: *Exemplo de Decision Tree*

5 Conclusão e Análise Crítica

Os tópicos abordados ao longo de todo este documento permitem-nos compreender a importância dos Sistemas de Aprendizagem na elaboração de um Sistema Inteligente. Como referido anteriormente, cada um dos modelos descritos tem as suas vantagens e desvantagens e devem ser utilizados consoante o objetivo final a que o utilizador pretende chegar.

Em relação às RNAs, sabemos que estas devem ser adotadas quando existe uma grande quantidade de dados para treinar a rede e assim poder obter um melhor desemepenho da mesma. As pricincipais vantagens identificadas são a sua capacidade de generalização, rapidez e autoaprendizagem. Contudo, existem algumas limitações, como é o caso do treino demorado, isto é, a solução pode demorar algum tempo a ser apresentada. Para além disso, torna-se impossível perceber como a rede chega a uma determinada conclusão, já que os critérios de decisão e a função objetivo do modelo são encriptados.

Já no caso das *Decision Trees*, podemos afirmar que é um método que apresenta bastantes vantagens quando o objetivo é ser mais preciso relativamente à classificação de um objeto, mas pode causar alguns problemas quando o objeto a classificar é relativamente diferente de todos os outros casos presentes no *training set*. Para isso podemos utilizar a estratégia de *prunning* para generalizar mais a árvore, ainda que isso possa causar menos precisão na caracterização do objeto. Resumindo, as *Decision Trees* são um método muito intuitivo e prático de aprendizagem.

O *Reenforcement Learning*, podendo ser implementado com uma grande variedade de mecanismos a servir de política de decisão, como por exemplo os outros dois tópicos abordados neste trabalho. É importante referir também que, tal como grande parte das técnicas de machine learning e inteligência artificial existentes ainda estão em pleno desenvolvimento, pelo que é importante explorar não só novas estratégiascomo também combinar as existentes desta forma.

Outro aspeto a mencionar é que, apesar de a ideia de não termos de definir completamente o ambiente do processo de decisão para chegar a uma política de decisão ótima (ou próxima disso), há que manter em mente que é necessário ter rigor na formulação dos componentes do modelo de aprendizagem em questão, caso contrário caímos numa ilusão de que a inteligência artificial é algo mágico que nos dá soluções facilmente, e estamos simplesmente a criar soluções pouco eficientes e subótimas.

Consideramos que o trabalho foi realizado sem grandes dificuldades sendo que existem imensos artigos que mencionam este tipo de sistemas, até porque os temas relativos a Inteligência Artificial são temas extremamente concorridos, de momento.

Concluimos portanto que, com a realização deste trabalho, cumprimos os objetivos do mesmo, tendo este proporcionado a todos os elementos do grupo oportunidade de conhecer e perceber alguns métodos de aprendizagem.