# Sistemas de Aprendizagem - Reinforcement Learning, Artificial Neural Networks and Support Vector Machines

Diogo Braga, Pedro Ferreira, Ricardo Caçador

University of Minho, Department of Informatics, 4710-057 Braga, Portugal
e-mail: {a82547,a81135,a81064}@alunos.uminho.pt
Grupo 10

**Resumo:** Neste documento é apresentada uma introdução a três tópicos inseridos na área de *machine learning*: *Reinforcement Learning*, *Artificial Neural Networks* e, por fim, *Support Vector Machines*, sendo abordados quatro aspetos por cada um deles. Em cada tema, primeiramente, é feita uma descrição característica após a qual se procede à explicação da forma como este exibe a capacidade de aprendizagem. De seguida, são enunciadas várias ferramentas que suportam e permitem o desenvolvimento de sistemas desse género. Por fim, são apresentados exemplos de aplicações práticas do mundo real envolvendo o sistema em causa.

**Keywords:** Sistemas de Aprendizagem · Reinforcement Learning · Artificial Neural Networks · Support Vector Machines

## 1 Introdução

As técnicas de *machine learning* consistem em algoritmos e modelos estatísticos que podem ser aplicados pelos computadores sem que estes precisem de instruções explícitas. Estes algoritmos constroem um modelo matemático a partir de um conjunto de dados de treino, de forma a que, posteriormente, o sistema seja capaz de prever o resultado de uma nova observação.

Para atingir esse efeito, é utilizado intensivamente o conceito de indução, que consiste em generalizar o *input* e aplicar o conhecimento que daí advém a situações futuras. Estas operações são fortemente fundamentadas por conceitos estatísticos como a média, moda, mediana, desvio padrão e variância uma vez que em muitas situações procura-se extrair caraterísticas do conjunto de dados que sustentem a generalização. Medidas como a covariância, correlação e técnicas como a regressão também são aplicadas frequentemente em *machine learning*.

Em relação a formas de aprendizagem indutivas, podemos destacar e diferir entre modelos supervisionados e modelos não supervisionados. Enquanto no primeiro modelo este é provisionado com pares de *input* e o valor que se lhe encontra associado (*output* esperado), no segundo a nenhum dado é associado um valor esperado, tendo de ser o sistema a aprender a agrupar os dados. Em ambos os casos, é seguido um fio de execução semelhante ao apresentado na figura 1.

Numa primeira fase, os dados devem ser pré-processados, procedendo-se à sua normalização, transformação e validação. Esta operação é de importância capital ao bom funcionamento do sistema de aprendizagem, tendo repercussões não só na sua qualidade como no seu desempenho. De seguida, o modelo deve ser treinado o número de vezes necessária até que este produza o resultado desejado. Geralmente, divide-se o conjunto de dados de *input* em conjunto de treino e de teste, para que seja possível fazer um *benchmark* ao sistema. Uma vez treinado o modelo, este encontra-se apto a avaliar novas observações.

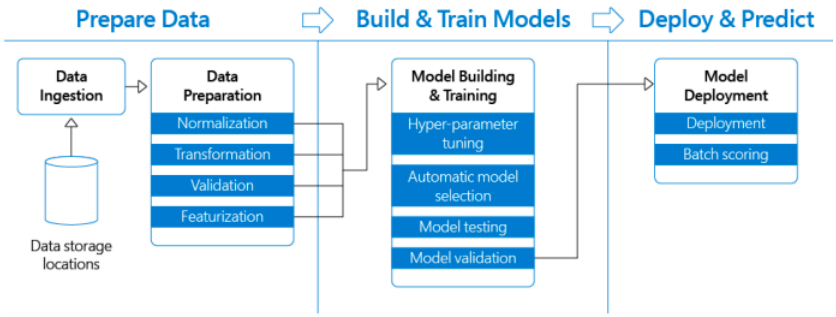


**Fig. 1.** *Machine Learning Pipeline* [1]

Podemos ainda identificar um terceiro tipo de modelo de aprendizagem: o *Reinforcement Learning*. Neste caso, um agente aprende a realizar uma determinada tarefa através de sucessivas interações com o ambiente em que se encontra inserido.

O presente documento foi produzido no âmbito da unidade curricular de Aprendizagem e Extração de Conhecimento, inserida no plano de estudos do perfil de especilização em Sistemas Inteligentes, sendo nele abordada a temática do *Reinforcement Learning* na secção 2. De seguida, na secção 3 são abordadas as Redes Neuronais Artificiais, que podem ser interpretadas como modelos supervisionados ou não supervisionadas conforme a rede em questão. Por fim, é apresentado um modelo de aprendizagem supervisionado: as *Support Vector Machines* (SVMs).

Para cada um dos sistemas referidos, será feita uma abordagem às suas principais características, aos modos de aprendizagem, às ferramentas de desenvolvimento existentes e às soluções de mercado atuais, pretendendo-se demonstrar diferentes métodos para construir sistemas inteligentes, partindo do pressuposto que quase todos os problemas necessitam de uma solução diferenciada, sendo desta forma interessante analisar vários métodos de resolução de problemas.

## 2 Reinforcement Learning

O *Reinforcement Learning* é um dos três paradigmas básicos de *Machine Learning*. Os algoritmos de aprendizagem que se enquadram neste paradigma, evoluem e tornam-se mais inteligente através de sucessivas interações de um agente com o ambiente em que este se encontra. Nesta área, a principal preocupação passa por definir a forma como os agentes devem realizar as ações. Geralmente, cada ação realizada despoleta a entrega de uma recompensa ao agente, procurando-se maximizar a recompensa obtida, e simultaneamente, minimizar o esforço despendido.

No geral, e de forma muito simplista, podemos caraterizar o *Reinforcement Learning* pela seguinte imagem:

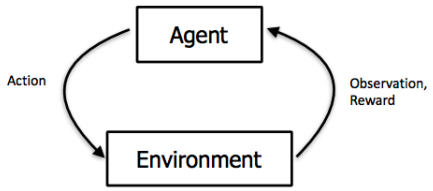


**Fig. 2.** Reinforcement Learning

Neste paradigma de *Machine Learning*, além do ambiente e dos agentes participativos, é possível identificar mais quatro sub elementos que pertencem a este paradigma: a política, o sinal de recompensa, a função valor e, possivelmente, o modelo do ambiente [3]. De seguida, apresentam-se as suas definições de forma a auxiliar a compreensão dos modos de capacidade, apresentados na secção seguinte.

- **Política:** define o modo de comportamento dos vários agentes para determinado momento. É feito um mapeamento entre os vários estímulos recebidos pelos vários agentes envolvidos no ambiente e as ações que estes agentes devem executar face aos estímulos recebidos. A política é definida, normalmente, através de um função simples, podendo também envolver uma computação mais complexa.
- **Sinal de recompensa:** é o valor da recompensa resultante de uma determinada ação num determinado estado do ambiente. O objetivo de cada agente é maximizar este valor. Este sinal serve como medida para perceber se a política aplicada é a mais correta. Caso este valor seja baixo, é provável que a política deva ser modificada.
- **Função valor:** este valor tem em conta a recompensa a longo prazo, ao contrário do sinal de recompensa. Basicamente, este valor é o somatório de todos os sinais de recompensa que um agente espera atingir no futuro. Assim, este valor é mais impactante do que o sinal de recompensa, pois é calculado a longo prazo, e não apenas para um determinado momento. Porém, não é um valor fácil de calcular, uma vez que assenta em previsões a longo prazo.
- **Modelo do ambiente:** este modelo tenta imitar o comportamento do ambiente onde se encontram os agentes, de modo a poder inferir que ações o caracterizam. Assim, poderemos prever quais os estados após determinada ação.

### 2.1 Modo de Capacidade de Aprendizagem

Existindo diversos algoritmos de *Reinforcement Learning*, estes podem ser divididos em duas grandes categorias consoante o seu modelo base. De seguida apresentam-se essas categorias, destacando as suas diferenças.

## 2.2 Model-Free vs Model-Based

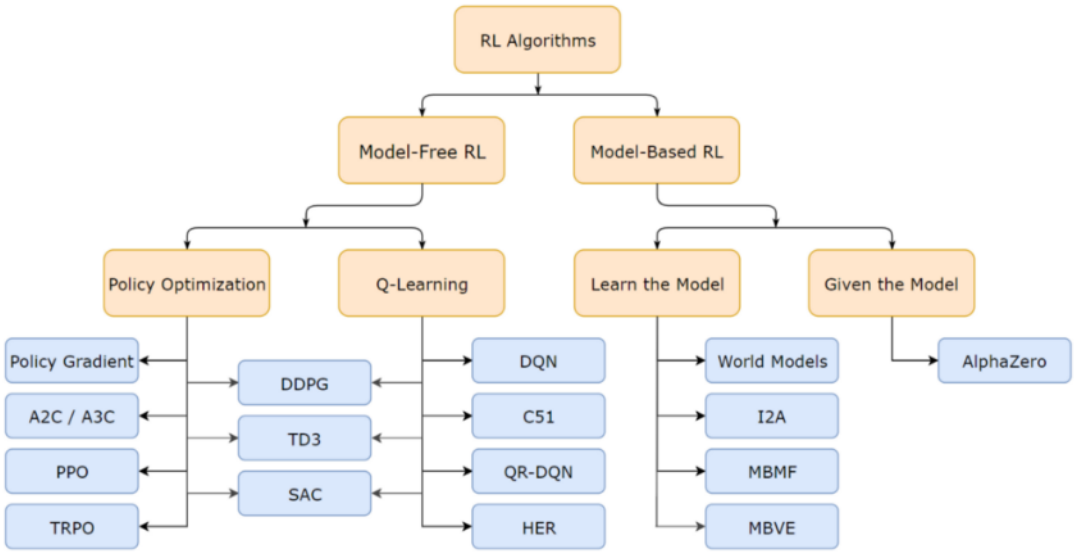


**Fig. 3.** Model Based vs Model Free

No caso de algoritmos *Model-Free*, é utiliza a experiência para construir um modelo interno de transições, e dos respetivos resultados, no ambiente onde o agente está inserido. Depois, as ações são escolhidas por procura no modelo real.

Por outro lado, em algoritmos *Model-Based*, utiliza-se a experiência resultante de uma ou duas transições que consigam transmitir um comportamento óptimo, sem fazer a estimativa para o modelo do mundo real.

### 2.2.1 Algoritmos Model-Free

– **Policy Optimization:** nestes algoritmos, existem estados melhores que outros - aqueles que garantem melhores recompensas. Existem, por outro lado, estados que não são apropriados para o progresso do agente no ambiente, ou seja, que possuem um valor baixo de recompensa. Sendo assim, como o objetivo do agente é maximizar as recompensas adquiridas e evitar estado menos proveitosos, é preciso encontrar uma política de ações com o objetivo de escolher a ação com a recompensa maior.

  Num conjunto de todas as políticas possíveis, existirá uma política ótima, cujo agente terá que descobrir. Para essa descoberta, o agente vai comparar diferentes políticas através da sua interação com o ambiente, formulando assim uma estratégia de ações.

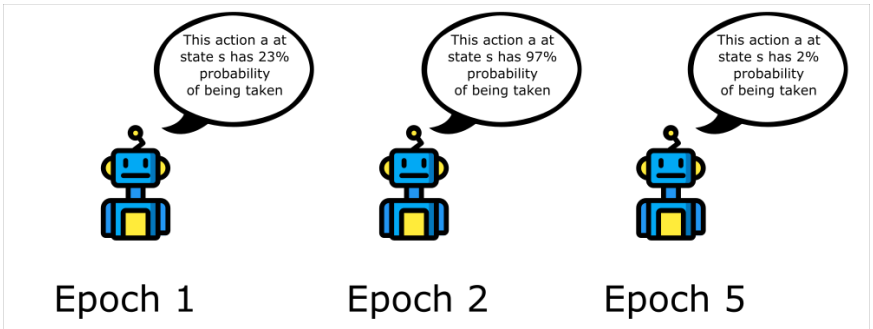


**Fig. 4.** Policy Optimization

– **Q-Learning:** este algoritmo torna possível determinar a política de ações ótima através do conhecimento único do valor ótimo da função valor de ações.

$$Q^{\pi}\ (s_t, a_t\ ) = r(s_t, a_t\ ) + \gamma max_{a'}[Q(s_{t+1}, a')].$$

**Fig. 5.** Função Q-Learning

O algoritmo desta função funciona da seguinte forma (com , a taxa de aprendizagem e $\lambda$ o fator de desconto):

1. Para cada estado **s** e ação **a**, inicializar a função $Q$.
2. Para cada episódio, observar o estado atual e escolher uma ação para ser executada. Para cada passo desse episódio efetuar o mesmo e assim sucessivamente até o estado **s** ser o estado terminal.

### 2.2.2 Algoritmos Model-Based

– **Learn the Model:** neste algoritmo é corrida uma política base, enquanto os resultados e a trajetória do modelo são observadas. O modelo é feito segundo dados exemplo que nele são inseridos. Assim, este é treinado para minimizar o erro quadrático para a função com os dados exemplo.

– **Given the Model:** como o próprio nome indica, neste tipo de algoritmo é passado um modelo previamente feito, sendo este corrido, recolhendo-se *feedback* da sua aplicação.

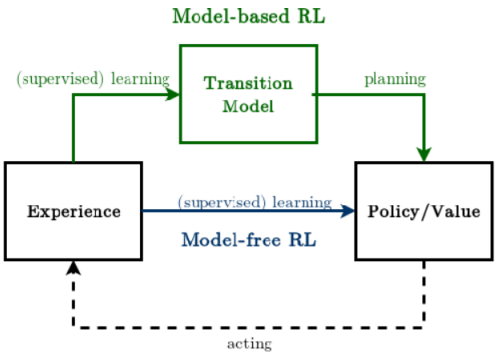


**Fig. 6.** Funcionamento Model Free vs Model Based. [2]

### 2.3 Ferramentas de Desenvolvimento

Existem, no mercado, várias ferramentas que permitem trabalhar com *Reinforcement Learning*, destacando-se as seguintes:

- **Project Malmo:** plataforma, desenvolvida pela Microsoft, para fins de experimentação e investigação de algoritmos de IA (como o caso do Reinforcement Learning). Esta ficou conhecida por ter como base o famoso jogo Minecraft.
- **TensorFlow:** permite o treino de agentes para a realização de certas tarefas.
- **PyBrain:** biblioteca de Python para implementação de vários algoritmos de Reinforcement Learning.
- **RL4J (Reinforcement Learning For Java):** biblioteca para implementação de algoritmos.
- **OpenAIGym:** conjunto de ferramentas que para desenvolvimento e comparação de vários algoritmos de *Reinforcement Learning*.

### 2.4 Soluções no Mercado

Atualmente, o *Reinforcement Learning* é utilizado por várias empresas em diferentes indústrias como forma de resolução para vários problemas. É notória a existência de vários casos de empresas que utilizam robots para auxiliar tarefas que se mostrem repetitivas, em contexto fabril. Nestes casos, através da tentativa e erro, o robot treina-se, autonomamente, até ser capaz de executar todas as tarefas de forma exímia.

Estes algoritmos são, também, utilizados para apresentação e recomendação de conteúdo personalizado aos utilizadores em diversos websites. Devido ao sucessivo feedback que recebem conseguem adaptar-se consoante os gostos e seleções dos utilizadores.

- **Contexto Fabril:** na empresa Fanuc é utilizado *Reinforcement Learning* para a utilização de robots que participam em tarefas repetitivas.

**Fig. 7.** Robots realizam tarefas na fábrica da Fanuc

– **Finanças:** Pit.AI é a empresa responsável e líder no desenvolvimento de algoritmos de *Reinforcement Learning* para este sector. Esta empresa estuda estratégias de *trading* no mercado, sendo uma ferramenta muito robusta devido ao grande número de dados que envolve o mercado financeiro.

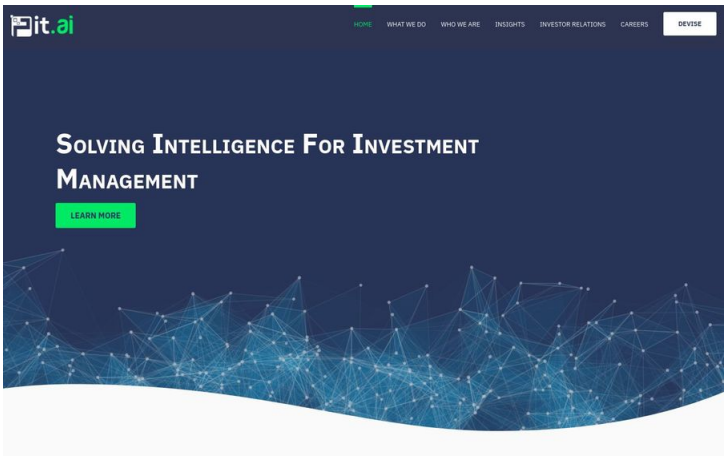


**Fig. 8.** Pit.AI

## 3 Artificial Neural Networks

*Artificial Neural Networks* (ANN) ou Redes Neuronais Artificiais (RNA), são uma área de muito interesse devido à possibilidade de representação do cérebro humano, altamente complexo e não-linear.

Exibindo capacidade de organização dos seus neurónios para realização de reconhecimentos ou perceções, o cérebro humano é muito mais rápido do que os computadores a processar todos esses acontecimentos. Um exemplo muito representativo de tais capacidades é o uso dos olhos para a construção da representação do ambiente em volta do ser humano, de forma a que, de seguida, seja possibilitada a interação com o ambiente. Deste modo, a representação computacional do cérebro humano tem sido uma área de muito estudo [4].

De forma a possibilitar a adaptação ao meio envolvente, as Redes Neuronais são constituídas por neurónios artificias, que possuem capacidade de desenvolvimento e de aprendizagem. Uma RNA é, portanto, um processador paralelo, composto por várias unidades de processamento simples (os nodos), que possui uma propensão natural para armazenar conhecimento empírico e torná-lo acessível ao utilizador. É, de uma forma mais prática,

uma máquina capaz de modelar uma resposta de modo semelhante ao que o cérebro faria para qualquer pergunta ou requerimento. O conhecimento é adquirido a partir de um ambiente, através de um processo de aprendizagem, sendo armazenado nas conexões, também designadas por ligações ou sinapses, entre nodos [5].

A figura 9 ilustra os conceitos abordados até ao momento.

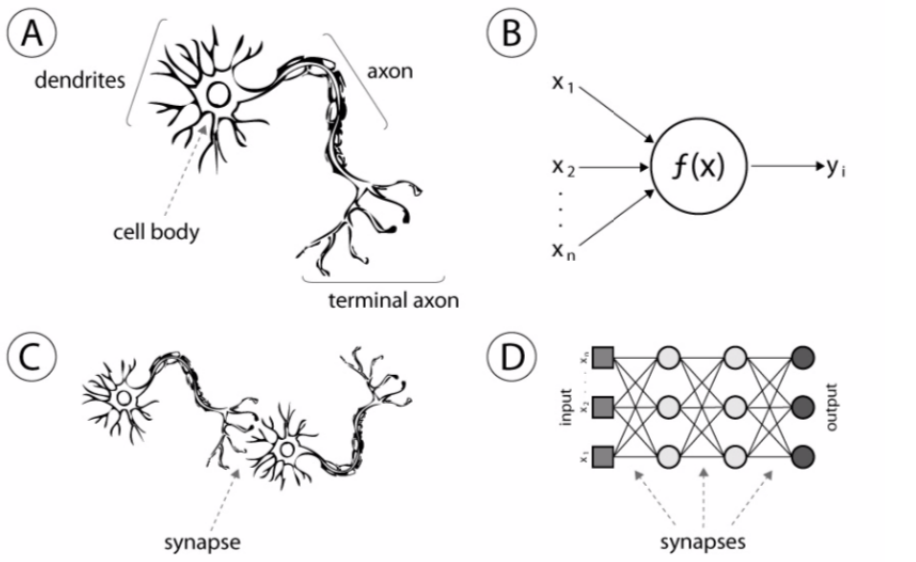


**Fig. 9.** (A) Neurónio Humano; (B) Neurónio Artificial; (C) Sinapse Biológica; (D) Sinapses das RNA.

### 3.1 Benefícios das RNAs

O poder computacional de uma RNA deriva de dois aspetos: ser uma estrutura paralelamente distribuída; possuir apetência para aprender e consequentemente generalizar, isto é, ser possível produzir respostas com base em experiências passadas. São estas duas características que tornam possível a resolução de problemas, que de outra forma seriam intratáveis. No entanto, as RNAs podem, por vezes, não conseguir dar resposta a um determinado problema, pelo que, nessas situações, são integradas com outros sistemas.

Segundo [5], as RNAs apresentam características como:

– **aprendizagem e generalização**: conseguindo descrever o todo a partir de algumas partes, constituindo-se como formas eficientes de aprendizagem e armazenamento de conhecimento;

– **processamento paralelo**: permitindo que tarefas complexas sejam realizadas num curto espaço de tempo;

– **transparência**, podendo ser vistas como uma caixa negra que transforma *inputs* em *outputs*, através duma função desconhecida;

– **não-linearidade**: atendendo a que muitos dos problemas reais a resolver são de natureza não-linear;

– **adaptabilidade**: podendo modificar a sua topologia de acordo com as mudanças do meio envolvente, adaptando-se a este;

– **resposta evidencial**: onde uma saída da rede traduz não só um processo de decisão mas também o grau de certeza a conferir a esta;

– **robustez e degradação suave**: permitindo processar o ruído ou informação incompleta de forma eficiente, assim como sendo capazes de manter o seu desempenho quando há desativação de algumas das suas conexões;

– **flexibilidade**: apresentando um grande domínio de aplicação.

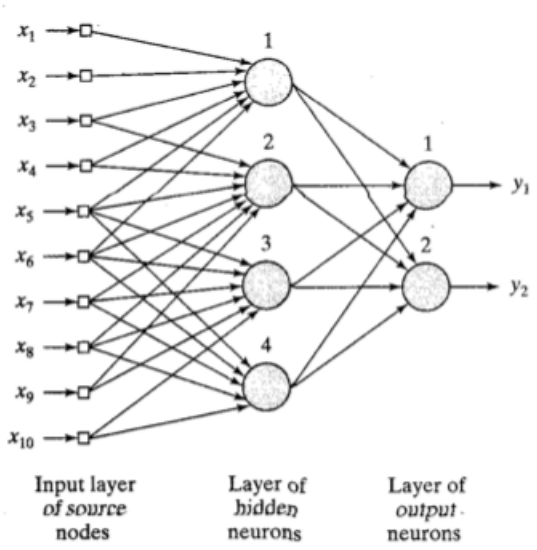


**Fig. 10.** Arquitetura duma RNA [4]

### 3.2 Modo de Capacidade de Aprendizagem

Uma propriedade muito importante numa RNA é a sua capacidade de aprendizagem, de forma a conseguir melhorar a performance do seu desempenho.

No contexto das redes neuronais, segundo [6], aprendizagem é definida por:

*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.*

A aprendizagem de uma RNA envolve a seguinte sequência de eventos [5]:

1. a RNA é estimulada por um dado ambiente;
2. certos parâmetros livres, normalmente os pesos das conexões, são alterados em resultado deste estímulo;
3. a RNA responde de uma forma nova ao ambiente em virtude das alterações na sua estrutura interna.

Todo este processo de aprendizagem é executado a partir de um conjunto de regras bem definidas, às quais se chama de algoritmo de aprendizagem ou treino. Uma vez que as RNAs são utilizadas em vários contextos, existem algoritmos de aprendizagem distintos, que diferem entre si na forma como os pesos são formulados e adaptados tendo em conta o objetivo final do problema em questão. Além disso, os algoritmos distinguem-se, também, na forma como se relacionam com o ambiente, tornando-se importante associar, aqui, o conceito de paradigma de aprendizagem.

### 3.2.1 Paradigmas de Aprendizagem [4]

– **Supervisionada**: Este paradigma envolve um "professor" que fornece respostas corretas à rede. A rede aprende a partir de um conjunto de padrões, onde cada um corresponde a um par formado por um *input* e o *output* correspondente. Em cada iteração é realizada uma comparação entre o valor desejado e o valor de saída da rede, e a partir dessa diferença são os valores das sinapses são atualizados.

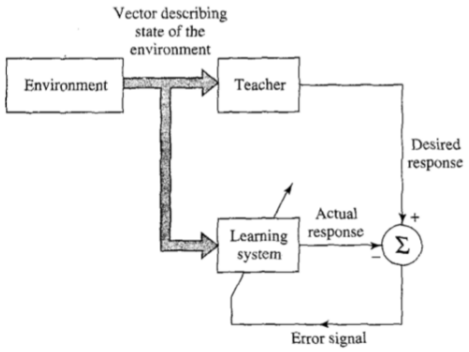


**Fig. 11.** Aprendizagem Supervisionada [4]

– **De Reforço**: Neste tipo de aprendizagem não existe um "professor", mas sim um género de avaliação. Apenas é fornecida uma indicação sobre se a resposta é correta ou errada, tendo a rede de usar esta informação para melhorar a sua eficácia.

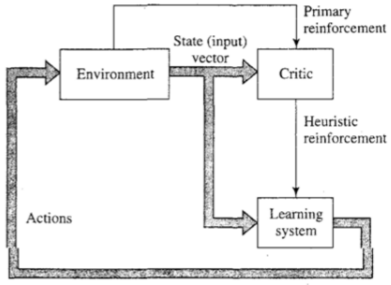


**Fig. 12.** Aprendizagem De Reforço [4]

– **Não Supervisionada**: Não é fornecida qualquer informação externa ao sistema acerca da resposta correta. A aprendizagem é realizada pela descoberta de regularidades e padrões nos dados de entrada.

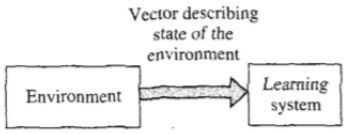


**Fig. 13.** Aprendizagem Não Supervisionada [4]

### 3.2.2 Regras de Aprendizagem [5]

– ***Hebbian***: Esta regra propõe que os pesos das sinapses sejam ajustados se houver sincronização entre os níveis de atividade das entradas e saídas, da seguinte forma:
  - Se dois nodos em cada lado de uma conexão são ativados simultaneamente, então a força dessa conexão é aumentada;
  - Se dois nodos em cada lado de uma conexão são ativados de forma assíncrona, então a conexão é enfraquecida ou até mesmo eliminada.

  Esta regra é muito utilizada em aprendizagem sem supervisão.

– **Competitiva**: As saídas dos nodos da mesma camada competem entre si para se tornarem ativas, com apenas um nodo a ser ativado num dado instante. No início do processo, os nodos de uma camada possuem conexões com pesos diferentes. Quando um padrão é fornecido à rede, um dos elementos da camada responderá melhor do que os outros, sendo por isso "premiado" com um reforço dos pesos das suas conexões.

– **Estocástica**: Neste tipo de aprendizagem, os pesos são ajustados de um modo probabilístico.

– **Baseada na Memória**: Esta regra tem por base as experiências passadas, que são guardadas em memória e posteriormente aplicadas em casos semelhantes.

– **Gradiente Descendente**: Para cada padrão apresentado à rede, esta produz um resultado. Após calcular o erro entre o valor pretendido e o valor que sai da rede, as conexões são ajustadas de modo a reduzir esta distância. Esta técnica está diretamente ligada à aprendizagem com supervisão.

## 3.3 Ferramentas de Desenvolvimento

Em seguida, são apresentadas algumas ferramentas que permitem o desenvolvimento de RNAs:

– **JustNN**: Um software intuitivo, que permite construir e testar redes neuronais artificiais a partir de um conjunto de dados.

– **R-Studio**: Utilizando o *neuralnet* é possível construir e gerir redes neuronais. São providenciados diversos *packages* para qualquer técnica de machine learning.

– **MATLAB**: Com o pacote Neural Network Toolbox fornece algoritmos e modelos capazes de criar, treinar, visualizar e simular redes neuronais.

– **Neural Designer**: Plataforma de machine learning que ajuda a descobrir relações, reconhecer padrões, prever tendências e encontrar associações entre dados.

– **Neuroph**: Framework do Java para desenvolvimento de arquiteturas comuns de redes neuronais. Contém classes com os conceitos necessários de redes neuronais.

### 3.4 Soluções no Mercado

Atualmente, verifica-se a utilização de redes neuronais artificias em diversas áreas de atividade. De seguida são enumerados alguns exemplos [7]:

- **Previsão no mercado financeiro**: Previsão é algo necessário nas decisões comerciais do dia-a-dia (por exemplo, vendas e investimento financeiro), na política económica, nas finanças e no mercado de ações. Os problemas de previsão são muito complexos. Prever a variação de ações é um exemplo disso, uma vez que existem muitos fatores subjacentes. Os modelos tradicionais de previsão apresentam limitações na consideração dessas relações não lineares. As RNAs, por outro lado, podem fornecer uma alternativa robusta, dada a sua capacidade de modelar e extrair recursos.

- **Reconhecimento de padrões**: Dada a capacidade das RNAs de receber muitos *inputs* e os processar para inferir relações complexas e não lineares, estas têm um papel fundamental nesta área. O reconhecimento de padrões, como a caligrafia, tem muita aplicação na detecção de fraudes (por exemplo, fraudes bancárias). O reconhecimento de imagens é uma área em crescimento, com aplicações abrangentes, como por exemplo o reconhecimento facial nas redes sociais.

Para além destes exemplos, as RNAs estão ligadas a aplicações em áreas como medicina, segurança, finanças, além de governo, agricultura e defesa.

## 4 Support Vector Machines

As *Support Vector Machines* (SVM), em português Máquinas de Vetores de Suporte, são sistemas de aprendizagem supervisionada que podem ser usados para classificação de dados e análise de regressão. São treinadas com um algoritmo de otimização,fundamentado por conceitos provenientes da Teoria de Aprendizagem Estatística [8], cujo objectivo passa por definir uma função que mapeie o *input* no *output*, podendo ser uma função classificadora ou de regressão, conforme a aplicação em específico.

Dado um conjunto de dados de treino, as SVMs constroem um modelo que consiste na representação desses dados como pontos no espaço. Dizemos que os dados são linearmente separáveis caso seja possível desenhar um hiperplano que separe os elementos de cada uma das classes. Às SVMs que têm como objetivo a identificação desse hiperplano, chamamos de *Linear Support Vector Machines* e o algoritmo que se encontra na sua génese foi introduzido por Vapnik, em 1963 [9]. Dada a complexidade do mundo real, este tipo de máquinas tem uma aplicabilidade baixa, uma vez que existem muitos dados que não podem ser linearmente separadados.

É neste contexto que surgem as *Non-Linear Suport Vector Machines*, apresentadas por Cortes e Vapnik, em 1995 [10]. Estes sistemas utilizam o *kernel trick*, que consiste em mapear implicitamente o *dataset* para uma dimensão superior à dos dados, onde estes possam ser separados linearmente, conforme exemplificado na figura 14. Por esta razão, as SVMs são referidas como sistemas de aprendizagem que usam um espaço de hipóteses linear num espaço de características de dimensão superior [8]. Os pontos que se encontram mais próximos do hiperplano supramencionado chamam-se *support vectors*, sendo os únicos elementos do conjunto de dados realmente importantes para o sucesso do algoritmo.

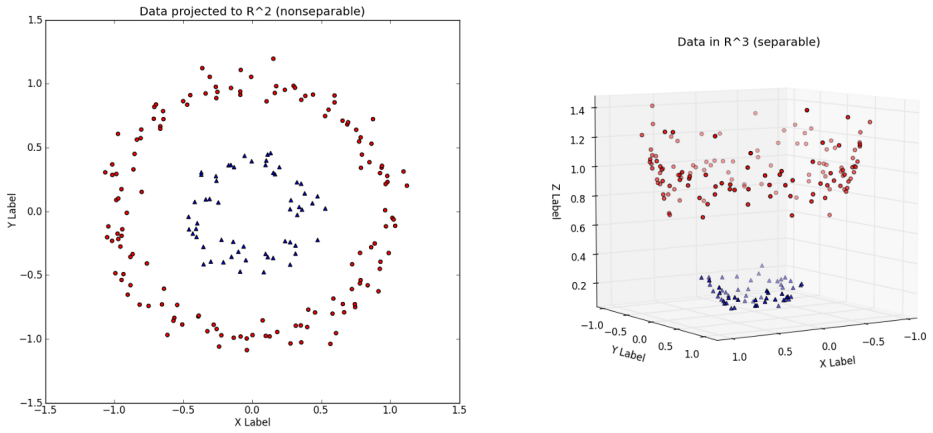


**Fig. 14.** Aplicação do *kernel trick* a conjunto de dados não separável linearmente [11]

### 4.1 Modo de Capacidade de Aprendizagem

Como descrito anteriormente, o algoritmo aplicado por uma SVM varia consoante esta seja linear ou não linear. Neste documento será abordado o caso em que o conjunto de treino é linearmente separável, seguindo-se do caso de um *dataset* maioritariamente linearmente separável, isto é, que apresenta presença de ruído e de *outliers*, e, por fim, será apresentada o procedimento para classificação de dados não separáveis linearmente. Em todos os casos, apenas será considerado o cenário de classificação binária, ou seja, quando o conjunto de dados apenas apresenta duas classes distintas. Assim, temos um conjunto de treino com $L$ elementos de dimensão $D$, pertencentes a uma de duas classes, $y_i = 1$ ou $y_i = -1$, da seguinte forma:

$$(x_1, y_1), ..., (x_L, y_L) \in \mathbb{R}^D \times \{-1, +1\} \tag{1}$$

#### 4.1.1 SVMs Lineares com Margens Rígidas

A função que uma SVM aprende para fazer a distinção entre duas classes é baseada na seguinte classe de hiperplanos:

$$f(x) = \mathbf{w} \cdot \mathbf{x} + b = 0 \tag{2}$$

Onde:

- **w** é a normal ao hiperplano;
- **x** é um vetor do *input*;
- $\frac{b}{\|\mathbf{w}\|}$ corresponde À distância entre o hiperplano e a origem.

Uma vez que a equação apresentada divide o espaço do conjunto de treino em duas regiões,

$$\mathbf{w} \cdot \mathbf{x} + b > 0 \tag{3}$$

$$\mathbf{w} \cdot \mathbf{x} + b < 0 \tag{4}$$

podemos utiliza-la para a atribuição de uma classe a observações futuras. Para isso, basta aplicar a função sinal ao resultado da aplicação de $f(x)$ à nova observação:

$$sgn(f(x)) = \begin{cases} 1 & \text{se } \mathbf{w} \cdot \mathbf{x} + b > 0 \\ -1 & \text{se } \mathbf{w} \cdot \mathbf{x} + b < 0 \end{cases} \tag{5}$$

De forma a minimizar o número de erros em observações futuras, acrescenta-se uma margem à superfície de separação das classes. Esta margem diz-se funcional pois é incorporada na função classificadora, tendo valor 1 [8]. Um hiperplano com margem funcional de valor unitário diz-se hiperplano canónico. Em essência, implementar uma SVM resume-se a escolher as variáveis **w** e $b$ de forma a que o conjunto de treino possa ser descrito pelas seguintes equações:

$$\mathbf{w} \cdot \mathbf{x}_i + b \geqslant +1 \text{ para } \; y_i = +1 \tag{6}$$

$$\mathbf{w} \cdot \mathbf{x}_i + b \leqslant -1 \text{ para } \; y_i = -1 \tag{7}$$

Estas duas equações podem ser combinadas em:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geqslant 1, \;\; \forall (x_i, y_i) \tag{8}$$

Como a partir de $f(x)$ é possível obter um número infinito de hiperplanos equivalentes, será necessário escolher o melhor. A abordagem mais simples, que também foi a primeira a ser aplicada, passa por tentar maximizar as margens entre o hiperplano e o conjunto de vetores de suporte, $T$. Estes devem verificar a seguinte equação:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) = 1, \;\; \forall (x_i, y_i) \in T \tag{9}$$

Na figura 15 encontra-se exemplificada uma situação em que existe um hiperplano canónico a separar as duas classes de dados, e, a tracejado, as respetivas margens. Os pontos cuja representação se encontra sobre as margens $H_1$ ou $H_2$ são os vetores de suporte.

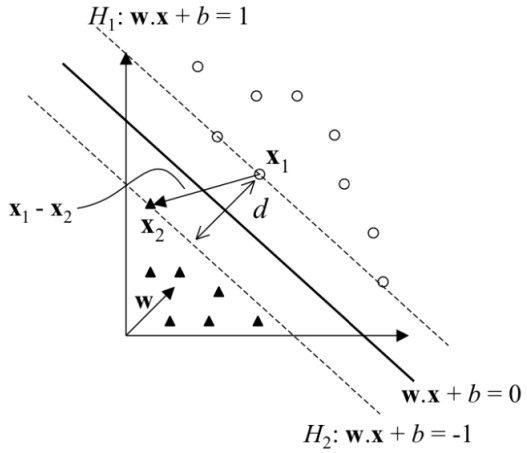


**Fig. 15.** Hiperplano e margens

Por geometria de vetores tem-se que a distância entre a margem e o hiperplano corresponde $\frac{1}{\|\mathbf{w}\|}$, pelo que maximizar a margem corresponde a minimizar a norma de **w** [12]. A resolução deste problema de otimização passa por torná-lo num problema de Programação Quadrática:

$$\begin{aligned} \min_{w} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{sujeito a} \quad & y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geqslant 1, \;\; \forall (x_i, y_i) \end{aligned} \tag{10}$$

e, de seguida, introduzir multiplicadores de Lagrange, podendo ser consultada, na íntegra, em [8]. A sua solução corresponde a uma função classificadora única, cuja margem até aos

vetores de suporte é a maior possível:

$$g(x) = sgn(\sum_{x_i \in T} y_i \alpha_i x_i \cdot x + b) \tag{11}$$

Em que:

$$b = \frac{1}{n_T} \sum_{x_j \in T} (\frac{1}{y_j} - \sum_{x_i \in T} \alpha_i y_i x_i \cdot x_j) \tag{12}$$

Onde $n_T$ corresponde ao número de vetores de suporte existentes e $\alpha_i \in \mathbb{R}$ pode ser visto como uma medida de quanto valor informacional $x_i$ tem [13]. Para os os valores do *dataset* que não façam parte do conjunto de vetores de suporte, este valor será 0.

### 4.1.2 SVMs Lineares com Margens Suaves

As SVMs apresentadas anteriormente apresentam margens rígidas, isto é, não admitem que nenhum ponto seja classificado incorretamente, nem que esteja no interior da margem, conforme a equação (8). No entanto, em situações reais, é dificil encontrar dados que sejam linearmente separáveis. Isto acontece devido a diversos fatores, nomeadamente a presença de ruídos e de *outliers*. De forma a extender as SVMs a casos em que os dados não sejam linearmente separáveis mas apresentem uma distribuição maioritariamente linear, introduz-se o conceito de margens suaves, permitindo que alguns pontos do conjunto de treino não verifiquem a equação (8). Deste relaxamento das restrições surgem, então, duas situações:

- Um ponto pode encontrar-se dentro da região de separação, mas no lado correto da superfície de decisão, conforme ilustrado na figura 16 a).
- Um ponto pode encontrar-se do lado incorreto da superfície de decisão, conforme ilustrado na figura 16 b).

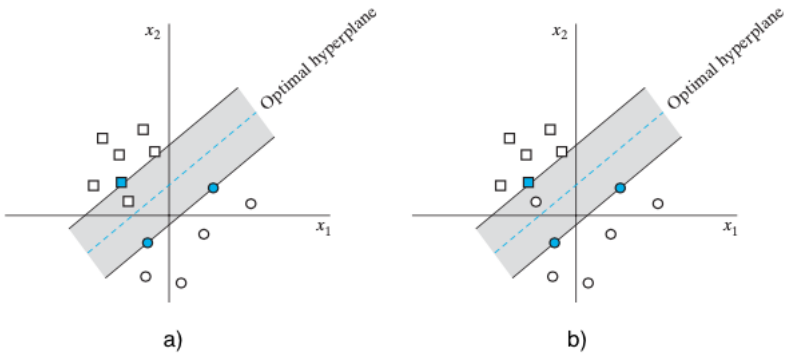


**Fig. 16.** Hiperplano de margens suaves. a) Ponto dentro da região de separação. b) Ponto do lado errado da superfície de decisão. [4]

Assim, a equação (8) deve ser reescrita, introduzindo uma variável de folga positiva, $\xi$ , o que possibilita o aparecimento das situações enumeradas:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geqslant 1 - \xi_i, \quad \text{para } i = 1, ..., L \tag{13}$$

A função objetivo (10), que procura identificar o melhor hiperplano para classificação binária de de dados linearmente separáveis, também deve ser modificada, incorporando $\xi$:

$$\begin{aligned} \min_{w} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{L}\xi_i \\ \text{sujeito a} \quad & y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geqslant 1 - \xi_i, \text{ para } i = 1, ..., L \\ & \xi_i \geq 0, \quad \text{para } i = 1, ..., L \end{aligned} \tag{14}$$

A constante C controla o equilíbrio entre a complexidade da SVM e o número de pontos não separáveis linearmente. Deve ser atribuído um valor elevado a C quando se tem elevada confiança na qualidade do conjunto de treino. Contrariamente, quando o conjunto de treino apresentar muito ruído, o valor de C deve ser pequeno, permitindo que $\xi_i$ tome valores maiores e consequentemente mais flexibilidade na classificação dos pontos.

Para $0 < \xi_i \leqslant 1$, o ponto $(x_i, y_i)$ encontra-se dentro da região de separação, mas no lado correto da superfície de decisão. Para $\xi_i > 1$ encontra-se no lado errado do hiperplano [4].

### 4.1.3 SVMs Não Lineares

Na maioria das aplicações do mundo real, as SVMs lineares não têm poder expressivo suficiente para fazer uma análise correta dos dados. Existem muitos conjuntos de dados que não podem ser separados por um hiperplano, sendo necessário definir uma fronteira de separação curva.

As SVMs não lineares fundamentam-se no teorema de Cover [14], que afirma que dado um conjunto de dados não separável linearmente este pode, com alta probabilidade, ser transformado num conjunto de dados linearmente separável se for projetado para um espaço de elevada dimensão, através de uma função não linear. Posto isto, seria possível mapear todos os elementos do conjunto de dados para um espaço de dimensão superior (chamado de espaço de caraterísticas), e aí, utilizar uma SVM linear para calcular um hiperplano capaz de separar as classes. No entanto, este procedimento é bastante ineficiente, principalmente se o mapeamento for feito para um espaço com um número considerável de dimensões ou se o *dataset* for muito volumoso.

Na prática, utilizam-se funções *kernel* que fazem esse mapeamento implicitamente, tirando partido do facto de numa SVM linear, no calculo da superfície de separação, serem usados produtos internos entre vetores do conjunto de treino, de acordo com a equação (11). Segundo [8], o *kernel* $k(x_i, x_j)$ é uma função que calcula o produto interno das imagens produzidas em espaço de caraterísticas através da aplicação da função de transformação a dois pontos de dados do espaço de *input*. As funções *kernel* mais famosas são:

– **Polinomial de grau p:**

$$k(x_i, x_j) = (1 + x_i \cdot x_j)^p \tag{15}$$

Ilustrado na figura 17.

– **Radial Basis:**

$$k(x_i, x_j) = \mathrm{e}^{\frac{\|x_i - x_j\|^2}{2\sigma^2}} \tag{16}$$

Ilustrado na figura 18.

– **Sigmoid:**

$$k(x_i, x_j) = tahn(\beta_0 x_i \cdot x_j + \beta_1) \tag{17}$$

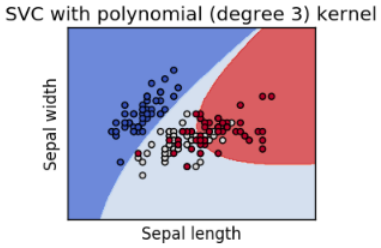


**Fig. 17.** Decisões de fronteira com kernel polinimial de grau 3.

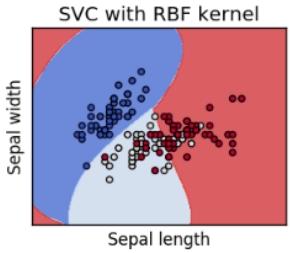


**Fig. 18.** Decisões de fronteira com kernel RBF

### 4.2 Ferramentas de Desenvolvimento

– **LIBSVM**: Tal como o nome sugere, é uma livraria para SVMs. Esta procura disponibilizar as SVMs, enquanto ferramenta de trabalho, a qualquer pesssoa, independentemente da sua àrea cientifica, pelo que apresenta, ainda, uma inteface gráfica. Permite trabalhar com diferentes tipos de SVMs e disponibiliza todas as funçõs kernel supramencionadas. Na figura 19 encontra-se representado um exemplo de interação esta ferramenta.

– **MATLAB**: Através da *Statistics and Machine Learning Toolbox* podemos ter acesso a funções como `fitcsvm` que permite treinar uma função classificadora para SVMs, podendo ser utilizadas diversas funções *kernel* inclusive funções desenhadas pelo utilizador.

– **SVMlight**: Corresponde a uma implementação de SVMs em C, baseada no modelo apresentado por Vapnik e Cortes em [10], sendo capaz de resolver tanto problemas de classificação como de regressão.

– **kernlab: Kernel-Based Machine Learning Lab**: Biblioteca para R que implementa um conjunto variado de algoritmos *machine learning* baseados em funções *kernel*.

Exitem ainda muitas outras ferramentas de desenvolvimento disponíveis tais como scikit-learn (biblioteca de *machine learning* para Python) e Shark (biblioteca de *machine learning* para C++).

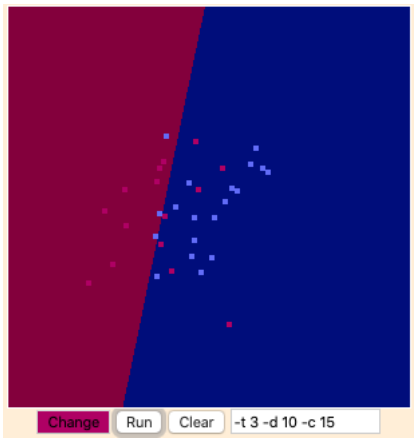


**Fig. 19.** Exemplo de utilização de LIBSVM: classificação binária utilizando SVM de margens suaves com kernel polinimial de grau 3 e parâmetro C=15

### 4.3 Soluções no Mercado

As SVMs foram aplicadas a vários problemas do mundo-real, sendo que em certas áreas produziram resultados *state-of-the-art*. O *paper* publicado por Cortes e Vapnik, em 1995 [10], que introduziu as SVMs, continha, já, um estudo prático sobre a aplicabilidade do modelo apresentado à identificação de números escritos à mão, motivado pelo facto de os US Postal Service precisarem de automatizar a ordenação de correio usando os códigos-postais escritos à mão.

Na figura 20 encontra-se representado o procedimento de classificação utilizado:

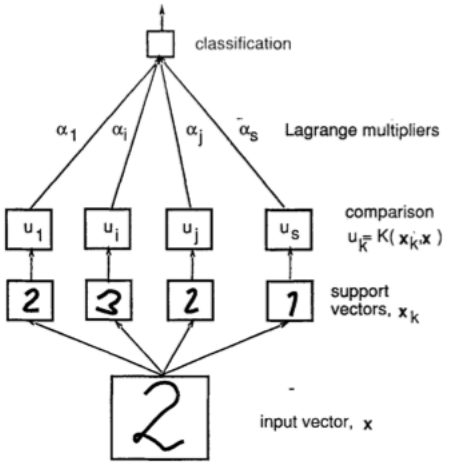


**Fig. 20.** Classificação de um padrão desconhecido por uma SVM. [10]

De entre os principais domínios de aplicação das SVMs encontram-se:

– **Categorização de texto**: esta atividade consiste em classificar documentos dentro de um certo número de categorias, de acordo com o seu conteúdo. Este problema enquadra-se em algumas situações, tais como filtrar emails, procuras na web e automatização em escritórios. Para cada documento é calculado uma pontuação, sendo que quando esta é superior a um determinado valor, o documento é enquadrado num categoria, caso contrário é classificado como documento geral. Em [15] e [16] encontram-se resultados da aplicação de SVMs a este domínio.

– **Reconhecimento de imagens**: a caraterização automática de imagens manifesta extrema utilidade em filtros de dados da internet, aplicações médicas e na deteção de objetos em cenas visuais. Devido à alta dimensionalidade deste tipo de dados, as SVMs conseguem apresentar melhores resultados que outros métodos de aprendizagem. A título exemplificativo, o documento [17] apresenta a aplicação de SVMs à categorização de imagens e o respetivo *benchmark*, sendo que o conjunto de treino apresenta 7 categorias diferentes (aviões, pássaros, barcos, edíficios, peixes, pessoas e veículos).

– **Bioinformática**: na área de biologia computacional as SMVs têm sido aplicadas na deteção de homologias de proteínas, de forma a prever caraterísticas estruturais e funcionais de uma proteína com base na sua sequência de aminoácidos.

– **Reconhecimento facial**: através de classificação de partes de imagem como sendo "cara" ou "não-cara", seguido de extração de caraterísticas dos pixeis de forma a desenhar uma superfície de divisão quadrada à volta da cara.

## 5 Conclusão

Neste trabalho, foi apresentada uma visão sobre três tipos de aprendizagem diferentes, cujas metodologias e abordagens diferem. Cada um deles possui vantagens e desvantagens próprias, características únicas e pressupostos distintos. No entanto, estas partilham um objetivo comum: o de tornar um sistema inteligente.

No caso do *Reinforcement Learning*, podemos dizer que esta metodologia permite resolver problemas mais complexos, problemas esses que outros algoritmos não conseguiriam resolver, sendo de notar que essa resolução será a longo prazo, algo difícil de se alcançar com outros algoritmos. Por outro lado, será de notar que outras das características que torna o RL especial é o facto de este possuir a capacidade de correção dos próprios erros durante o treino do modelo.

Apesar das vantagens supramencionadas, existem, no entanto, algumas desvantagens. Assim, podemos dizer que o RL não parece ser o algoritmo indicado para resolver problemas simples, na medida em que se trata de um tipo de algoritmo "data hungry", ou seja, que necessita de grandes quantidades de dados e computação para atingir os resultados pretendidos e maximizar a capacidade do algoritmo. Outro dos problemas a apontar ao RL prende-se à dificuldade que é testá-lo no mundo real, a título de exemplo, note-se o caso dos robots, toda a tecnologia envolvida num robot é demasiado cara, o que acaba por ser um entrave à realização de testes destes algoritmos.

Vistas algumas vantagens e apesar das desvantagens presentes nesta metodologia, a verdade é que no mundo real a utilização de algoritmos RL está presente em praticamente todos os robots que trabalham em fábricas, sendo a indústria fabril um dos pontos principais de todas as economias, podemos, então, dizer que será uma tendência futura o aumento da utilização deste tipo de robots que com a utilização destes algoritmos conseguem, autonomamente, aprender a executar as determinadas tarefas de uma empresa.

Concluindo e tendo por base o exposto, parece-nos complicado encontrar desvantagens tão significativas que afastem a ideia de que o RL é um dos paradigmas mais interessantes e úteis no presente e no futuro do Machine Learning.

As Redes Neuronais Artificiais foram o segundo tema de aprendizagem abordado. A Inteligência Artificial é, por vezes, percecionada como uma máquina que, aprendendo com experiências do seu meio envolvente, se ajusta e evolui no sentido de realizar ações o mais parecidas possível com as dos seres humanos. Ora, as RNAs formam o sistema que melhor consegue representar o funcionamento do cérebro humano, e é com base no interesse crescente em modelar o cérebro do ser humano que estes sistemas têm evoluído.

As RNAs diferenciam-se das restantes abordagens devido à sua grande capacidade de aprendizagem, devido à sua capacidade para resolver problemas de natureza não-linear, e devido à sua flexibilidade aplicacional, podendo estar presente em inúmeros e variados problemas do seu meio envolvente. Além dessas qualidades, as RNAs têm a versatilidade de poder ser aplicadas em paradigmas supervisionados e não supervisionados, deixando essa decisão para as preferências e necessidades do utilizador. Perante as capacidades apresentadas em todo o documento, constata-se que as RNAs são, de facto, uma escolha muito viável entre os sistemas de aprendizagem existentes, e tal é comprovado pelas inúmeras aplicações que este sistema possui no mercado, tanto em previsões como em reconhecimentos.

Neste texto foram também abordadas as SVMs. Estas, devido à utilização do *kernel trick*, são adequadas para o tratamento de dados com muitas características, conseguindo alcançar melhores resultados que outras técnicas de *machine learning*. Uma vez que podemos aplicar várias funções *kernel*, esta técnica consegue adaptar-se facilmente a diferentes *datasets*, independentemente de quão diversa seja a representação no espaço de *input*. Para tal, basta escolher a função que melhor se adeque, sendo uma abordagem por tentativa e erro uma forma plausível de a escolher. Outro aspeto que propicia os bons resultados alcançados pelas SVMs passa pelo problema de optimização não linear quadrática que se encontra no cerne da escolha da superfície de decisão. Este é um problema convexo o que

significa que não apresenta mínimos locais. Esta constitui uma das vantagens das SVMs em relação a outros tipos de sistemas de aprendizagem como as RNAs Perceptron multicamadas. Em contrapartida, as SVMs apresentam, na fase de treino, complexidade $O(n^2)$, em que $n$ representa o número de elementos do *dataset* de treino, pelo que não apropriadas para conjuntos de dados muito grandes.

Todos os algoritmos para a determinação de superfície de separação apresentados, apenas consideram o caso em que estamos perante um conjunto de treino binário, isto é, a *label* apenas pode ser -1 ou +1. As SVMs podem também ser aplicadas a problemas de classificação multiclasses, sendo que neste caso se recorre a uma abordagem decomposicional, isto é, o problema de classificação é subdividido em vários problemas de classificação binária, até que todas as classes sejam identificadas. Este artigo também se limitou a abordar situações em que se estava à procura de uma função classificadora. No entanto, as SVMs também podem ser usadas para análise de regressão, procurando prever um valor real.

## References


1. Melo, Eduardo. “How to Accelerate DevOps with Machine Learning Lifecycle Management | Blog | Microsoft Azure.” Microsoft.Com, Microsoft Azure, 2019, azure.microsoft.com/en-in/blog/how-to-accelerate-devops-with-machine-learning-lifecycle-management/. Accessed 19 Oct. 2019.
2. Learning Multimodal Transition Dynamics for Model-Based Reinforcement Learning, semanticscholar.org/paper/Learning-Multimodal-Transition-Dynamics-for-Moerland-Broekens/0e76e8d623882945ef8891a02fc01b3675fd1222/figure/0
3. Richard S. Sutton and Andrew G.Barto, "Reinforcement Learning: An Introduction", 2nd Edition, November 5 2017.
4. Haykin, S., “Neural Networks – A Comprehensive Foundation”, Prentice-Hall, New Jersey, 2nd Edition, 1999.
5. Cortez, P., Neves, J., "Redes Neuronais Artificiais", Braga, Portugal, 2000.
6. Mendel e McClaren, 1970.
7. Jahnavi Mahanta. "Introduction to Neural Networks, Advantages and Applications" Medium, Towards Data Science, 10 Jul. 2017, towardsdatascience.com/introduction-to-neural-networks-advantages-and-applications-96851bd1a207. Accessed 19 Oct. 2019.
8. Nello Cristianini, John Shawe-Taylor, “An Introduction to Support Vector Machines and other kernel-based learning methods”, Cambridge University Press, 2000.
9. Vapnik, V., and A. Lerner. Pattern recognition using generalized portrait method. Automation and Remote Control, 24, 774–780, 1963
10. C. Cortes, V. Vapnik, in Machine Learning, pp. 273–297 (1995)
11. Harish Kandan. “Understanding the Kernel Trick.” Medium, Towards Data Science, 30 Aug. 2017, towardsdatascience.com/understanding-the-kernel-trick-e0bc6112ef78. Accessed 19 Oct. 2019.
12. Fletcher, Tristan. (2009). Support Vector Machines Explained.
13. Busuttil, Steven. "Support vector machines." Proceedings of CSAW’03. 2003.
14. Cover, Thomas M. “Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition.” IEEE Transactions on Electronic Computers, vol. EC-14, no. 3, June 1965, pp. 326–334
15. Joachims, T. Text categorisation with support vector machines, In Proceedings of European Conference on Machine Learning (ECML), 1998.
16. Dumais S., Platt, J., Heckerman, D. and Sahami, M. Inductive learning algorithms and representations for text categorisation. In 7th International Conference on Information and Knowledge Management, 1998
17. Chapelle, O., et al. “Support Vector Machines for Histogram-Based Image Classification.” IEEE Transactions on Neural Networks, vol. 10, no. 5, 1999, pp. 1055–1064