Sistemas de Aprendizagem: Aprendizagem por Reforço, Algoritmos Genéticos e Support Vector Machines

Ana Esmeralda Fernandes and Miguel Dias Miranda

Universidade do Minho, Departamento de Informática, 4710-057 Braga, Portugal,
WWW home page: https://www.uminho.pt/PT e-mail:
{a74321,a74726}@alunos.uminho.pt

Resumo Partindo dos fundamentos de *Machine Learning*, o presente artigo procura abordar alguns dos processos de aprendizagem, onde a partir de métodos computacionais e análise de dados ou estímulos, se procura dotar um sistema com a capacidade de decisão, previsão ou reação autónoma a um determinado contexto. Dentro dos vários processos aplicados para aprendizagem, será citada, com detalhe, a teoria dos processos de Aprendizagem por Reforço, Algoritmos Genéticos e *Support Vector Machines*. Como metodologia, é apresentado para cada tema uma descrição geral e contextualização, assim como referências aos algoritmos por eles aplicados, ferramentas de desenvolvimento e soluções atualmente existentes baseadas neste processo de aprendizagem.

Keywords: Sistemas Inteligentes, Inteligência Artificial, Processos de Aprendizagem não Supervisionada, Aprendizagem por Reforço, Algoritmo *Q-Learning*, Algoritmos Genéticos, Algoritmos Evolutivos, *Support Vector Machines*, Classificadores

1 Introdução

Vista aos olhos da sociedade como a ciência da ficção, a inteligência artificial (IA) surge como o paradigma de computação do futuro, apesar do tema já ser discutido à mais de cinco décadas e de ser, atualmente, parte integrante do dia-a-dia de cada indivíduo e de diversos setores económicos da sociedade.

Existindo um interesse em sistemas baseados em IA, é inevitável a discussão de processos de aprendizagem, como forma de dotar estes sistemas de "inteligência"e capacidade de decisão. Sistemas de aprendizagem (*Machine Learning*), representam na sua interpretação mais simples a aplicação de algoritmos no processamento de dados, como forma de aprender e gerar modelos de previsão relacionados com um contexto específico do ambiente onde o sistema se insere. Desta forma, em vez de codificar à força e por extensão todas as instruções necessárias para se atingir um determinado objetivo, é em oposição dada a um sistema a habilidade de indução, de forma a que o mesmo consiga aprender a realizar uma determinada tarefa.

Este relatório apresenta assim três secções principais, abordando diferentes algoritmos de aprendizagem. Em primeiro lugar, será abordado o processo de Aprendizagem por reforço, onde com recurso a um critério de recompensa, se indica a um agente se a sua ação teve consequências positivas ou negativas no contexto de atuação. De seguida, são abordados Algoritmos genéticos, onde utilizando conceitos da teoria da evolução de Darwin, se procuram fundir as melhores soluções para um problema, na esperança de se gerar descendência mais apta ao contexto do problema. Por fim, será introduzido o tema de *Support Vector Machines*, como um processo supervisionado de classificação de dados, e apresentada uma conclusão relacionando as principais características e vantagens de cada processo.

2 Aprendizagem por Reforço

2.1 Descrição geral

Processos de aprendizagem por reforço utilizam o estudo e a aplicação de algoritmos computacionais a um sistema, de forma a dotar um agente da habilidade de aprendizagem autónoma e capacidade de melhorar as suas ações através da sua experiência. Fundindo conceitos de inteligência artificial e processos de *Machine Learning*, um sistema será assim capaz de aceder a conjuntos de dados e usa-los para aprender por si mesmo, sem que as suas decisões tenham sido explicitamente programadas.

Visualizando um agente como uma entidade capaz de compreender um determinado ambiente e atuar sobre ele através de ações específicas, os algoritmos de aprendizagem por reforço introduzem um critério de avaliação de ações, levando o agente a moldar a sua aprendizagem de forma a maximizar o número de avaliações positivas que recebe pelo desempenho das suas ações.

Esta avaliação é conhecida como sinal de reforço (*reinforcement signal*) e devolve assim ao agente um *feedback* que lhe permite determinar se a sua ação foi apropriada para atingir os objetivos que o mesmo procura resolver. O conceito recai na ideia que quanto melhor forem as consequências da ação tomada, no contexto de atuação do agente, maior será a avaliação recebida pela mesma.

Utilizando esta noção de avaliação de ações, os algoritmos de aprendizagem por reforço ganham bastante interesse. Não sendo necessário um vasto conhecimento sobre o problema e como atingir a melhor solução de forma direta, nem sendo preciso criar regras complexas que indiquem ao agente de que forma atingir os seus objetivos, o desafio da aprendizagem por reforço passa apenas por saber avaliar se a prática de uma ação é ou não benéfica para resolver um determinado problema.

2.2 Processo de Reforço

A utilização de um reforço como técnica de aprendizagem dedica-se assim a situações onde um sistema evolui exclusivamente por um processo de tentativa e erro, através das suas decisões e ações sobre o ambiente dinâmico em que se insere. Não havendo a necessidade de existir uma entidade externa que forneça exemplos de treino ou que ensine a um sistema qual o melhor caminho a seguir para alcançar a solução de um dado problema, este processo de aprendizagem não exige a presença de um "professor". Numa certa analogia com outros processos de aprendizagem, os dados de treinos de um agente que siga um processo de aprendizagem por reforço podem ser vistos como os resultados obtidos por si próprio ao atuar no meio através das suas decisões.

Como referido anteriormente, o único modo que o agente usa para se adaptar ao meio são as avaliações que recebe após desempenhar as suas ações no ambiente. Estes *feedbacks* podem assim ser negativos, positivos ou neutros e procuram simular recompensas ou punições no agente pelas suas ações. Como um dos objetivos dos agentes é maximizar o desempenho das suas recompensas, estes *feedbacks* permitem corrigir os erros das ações iniciais do agente e desenvolver novos planos orientados para a solução correta para o problema.

A maioria dos problemas de aprendizagem por reforço podem ser formulados com recurso a um processo de decisão Markoviano, isto porque as decisões do agente devem ser tomadas em sequência e o resultado de cada decisão e respetiva atuação no meio não são totalmente conhecidas para o agente.

Este processo de decisão de Markov consiste assim num tuplo ($\boldsymbol{S}$, $\boldsymbol{A}$, $\boldsymbol{T}$, $\boldsymbol{R}$) onde:

- $\boldsymbol{S}$ representa o conjunto de estados onde o ambiente pode estar;
- $\boldsymbol{A}$ representa o conjunto de ações que o agente pode tomar nos diferentes períodos de decisão;
- $\boldsymbol{T}$ representa o modelo de transição.

  A função dá a probabilidade do sistema passar para o estado $s' \in \boldsymbol{S}$, dado que o sistema estava no estado prévio $s \in \boldsymbol{S}$ e o agente tomou a decisão de executar a ação $a \in \boldsymbol{A}$.

  T : S x A x S $\rightarrow$ [0,1]
  T(s,a,s') = p(s' s, a)
- $\boldsymbol{R}$ representa a função de avaliação. O seu resultado indica a recompensa que o agente recebe por tomar a decisão $a \in \boldsymbol{A}$ quando o sistema estava no estado $s \in \boldsymbol{S}$.

  R : S x A $\rightarrow \mathbb{R}^2$

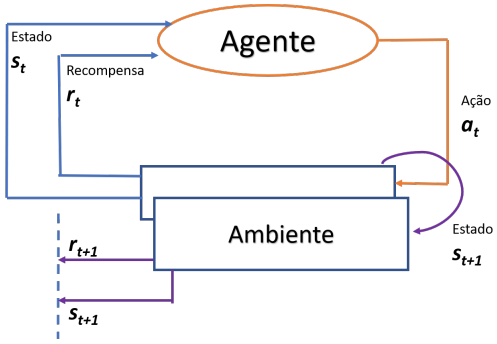

Figura 1. Modo de atuação entre o agente e o ambiente em que se insere

O comportamento do agente é visto como uma sequência de ações ao longo do tempo onde em cada período de tempo t o sistema encontra-se no estado s_t. Quando o agente aplicar uma ação a_t ao ambiente, o ambiente é afetado e muda para o estado s_{t+1} conforme a probabilidade de transição dada por $\mathbf{T}$. Com esta alteração, o agente recebe do ambiente informações sobre o estado resultante s_{t+1}, assim como da recompensa r_{t+1}, pela ação desempenhada.

Por se modelar este sistema segundo um processo de decisão Markoviano, tem que ser definida uma política π, que atribui uma probabilidade de sucesso a todas as possíveis ações em cada estado.

Para determinar esta política é necessário que:

- Ou se conheça bastante bem o ambiente e as consequências que cada ação vai desempenhar no meio, após a sua execução;
- Ou se estime a recompensa de cada uma das ações, por exemplo, através do processo *Q-Learning* (Abordado na secção seguinte)

Se o agente agir segundo uma política, irá escolher a sua ação conforme as probabilidades atribuídas a cada ação, em cada estado. Procurar as melhores avaliações corresponde assim a encontrar uma política ótima π que maximiza o somatório da recompensas recebidas, ou seja, num determinado estado pode até não ser escolhida a decisão que obtenha o reforço máximo, mas numa perspetiva geral e olhando para todos os estados, a sequência de decisões tomadas maximiza o somatório das recompensas obtidas.

2.3 Algoritmo *Q-Learning* e SARSA

A modelação de problemas de aprendizagem por reforço, segundo a perspetiva de um processo de decisões de Markov, requer que exista um modelo perfeito que descreva o ambiente onde se insere o agente, pois de outra forma deixa de ser possível determinar o estado ***s'*** para que o ambiente transita após a realização de uma ação ***a***. Cortando esta possibilidade de determinar qual o próximo estado, após uma determinada ação, é inviabilizada a possibilidade de

estimar as recompensas e determinar uma política ótima que permita maximizar o total das recompensas obtidas pelas decisões do agente.

Como em contextos reais, um agente pode de facto não conhecer nem conseguir prever as mudanças que as suas ações causam no sistema, existem alguns algoritmos que permitem contornar esta situação, utilizando políticas pré definidas.

O algoritmo *Q-Learning*, surge assim como uma das alternativas muito utilizadas para estimar o reforço que uma ação a obtém, quando aplicada a um estado s, através do uso função Q(s,a). Esta função é consultada sempre que é necessário tomar uma nova decisão e atualizada sempre que efetivamente se realiza uma nova ação e se recebe a respetiva recompensa. A função Q referida, no contexto do algoritmo *Q-Learning*, é definida da seguinte forma:

$$Q(s_t,a_t) = Q(s_t,a_t) + \alpha_t[\ r(s_t,\ a_t) + \gamma * \max(\ Q(s_{t+1},\ a_{t+1})\ ) - Q(s_t,a_t)\ ]$$

- O valor de α representa o ritmo de aprendizagem do processo *Q-Learning*. É um parâmetro que varia [0, 1] sendo que, caso o seu valor seja próximo de 0, as estimativas perante novos estados do ambiente nunca se alteram, e portanto o conhecimento ganho é nulo. Em oposição, valores altos deste parâmetro permitem que o processo de aprendizagem ocorra de forma rápida.
- O parâmetro γ representa o fator de valorização das recompensas futuras face à recompensa atual e varia também entre [0, 1]. Caso o valor de γ seja próximo de zero, o agente dará importância apenas ao valor das recompensas mais imediatas, enquanto que se o seu valor for elevado, o agente vai considerando o valor de recompensas futuras. Isto significa que o agente, num determinado estado s, pode escolher a ação a que não representa a maior recompensa no presente, mas que perante a função Q(s,a) leva a uma sequência futura de decisões que maximiza a soma de recompensas. Na prática o valor deste parâmetro deve ser próximo de zero para permitir a convergência do algoritmo em tempo satisfatório.
- O valor de *max* representa a recompensa máxima que pode ser obtida no estado seguinte, como se de certa forma fosse seguida uma política *greedy*.

O algoritmo *Q-Learning* é ainda caracterizado como sendo *off-policy*, pelo facto de não precisar de seguir sempre uma política especifica. Na atualização do valor de Q(s,a), no parâmetro ***max***, o algoritmo assume que no próximo estado s_{t+1} vai escolher a decisão a_{t+1} que maximiza a esperança de um retorno elevado naquele estado, não considerando a exploração de outras hipóteses.

Em oposição, o algoritmo SARSA apresenta uma aprendizagem *on-policy*, por estimar o retorno da próxima ação determinando-a através da política atual, e não assumindo que a ação seguinte será a que devolve o maior recompensa. A atualização da função Q, na abordagem SARSA, será assim do tipo:

$$Q(s_t,a_t) = Q(s_t,a_t) + \alpha_t[\ r(s_t,\ a_t) + \gamma * Q(s_{t+1},\ a_{t+1}) - Q(s_t,a_t)\ ]$$

A designação SARSA representa a sigla das palavras *State-Action-Reward-State-Action* ou, em português, Estado-Ação-Recompensa-Estado-Ação. Neste

algoritmo, e como o nome sugere, o agente começa por se encontrar num estado s_t e desempenhar a ação a_t, recebendo assim a recompensa associada a esta atuação no ambiente.

Como ao executar uma ação, o ambiente transita para o estado s_{t+1}, o algoritmo vai observar este estado e determinar qual a melhor ação a_{t+1} para este novo estado, usando para isso a mesma política com que selecionou a ação a_t. Só depois de analisar o novo estado e escolher a próxima ação é que o algoritmo vai atualizar o valor da função Q(s,a), em função do estimulo associado à recompensa recebida.

Como referido, a principal diferença face ao algoritmo *Q-Learning* está no facto deste ultimo assumir que a próxima ação será a que trás uma maior recompensa ao passo que o processo SARSA analisa qual é a próxima ação através da política em uso e define essa ação como sendo a próxima a executar.

Os procedimentos gerais do algoritmo *Q-Learning* e do algoritmo SARSA podem ser descritos em pseudo-código da seguinte forma:

```
iniciar_Q();           //Iniciar o valor da estimativa Q(s,a) para todo os estados s e ações a

Estado s = observa_estado()                    //Inicializa s com o valor do estado atual

cond_paragem = avaliar_estado( s );

while( !cond_paragem ){
      for( cada episodio de decisão ){
            // Escolhe a ação com maior recompensa, considerando todas as ações possíveis de
            // executar naquele instante, para o estado s, e segundo a politica π que utiliza
            // as estimativas da função Q para maximizar a soma total de recompensas
            Ação escolha = seleciona_Acao_Q(s);

            executa( escolha );                // Executa a ação no ambiente;

            Recompensa r = recebe_Recompensa() // Recebe a recompensa por executar a sua ação;

            atualiza_Q(s, escolha, r);         // Atualiza o valor que Q(s,a) com o valor da
                                               // recompensa efetivamente recebida;

            s = observa_estado();              // Observa o novo estado para o qual o ambiente
                                               // transitou após executar a ação;

            cond_paragem = avaliar_estado(s);  // Verificar se s é um estado terminal ou se
                                               // foi atingido o número de iterações máximo;
      }
}
```

Figura 2. Esboço da estrutura de código de um algoritmo *Q-Learning*

```
iniciar_Q();           //Iniciar o valor da estimativa Q(s,a) para todo os estados s e ações a

Estado s = observa_estado()                    //Inicializa s com o valor do estado atual

Cond_paragem = avaliar_estado( s );

while( !cond_paragem ){
      // Escolhe a ação com maior recompensa, considerando todas as ações possíveis de
      // executar naquele instante, para o estado s, e segundo a politica π que utiliza
      // as estimativas da função Q para maximizar a soma total de recompensas
      Ação escolha = seleciona_Acao_Q(s);

      for( cada episodio de decisão ){
            executa( escolha );                // Executa a ação no ambiente;
            Recompensa r = recebe_Recompensa() // Recebe a recompensa por executar a sua ação;
            s' = observa_estado();             // Observa o novo estado para o qual o ambiente
                                               // transitou após executar a ação;

            Ação next = seleciona_Acao_Q(s');  // Para o novo estado, determina qual a melhor
                                               // ação seguinte segundo a politica atual;

            // Atualiza o valor que Q(s,a) com o valor da recompensa efetivamente recebida
            // assim como o próximo estado e decisão next a tomar.
            atualiza_Q(s, escolha, r, s', next);

            s = s'; escolha = next;            // Define que a próxima escolha a tomar será
                                               // a que foi determinada para o novo estado;
            cond_paragem = avaliar_estado(s);  // Verificar se s é um estado terminal ou se
                                               // foi atingido o número de iterações máximo;
      }
}
```

Figura 3. Esboço da estrutura de código de um algoritmo SARSA

2.4 *Exploration* vs *Exploitation*

Com a utilização de mecanismos de aprendizagem por reforço, surge como desafio o problema conhecido como "*Exploration* x *Exploitaion*", que pode ser traduzido como "Exploração vs Investigação". Para obter os melhores resultados de aprendizagem, o sistema deve ser capaz de decidir quando deve aprender e quando não deve aprender, usando em alternativas informação que o sistema já possua.

- **Exploração** recai assim sobre a procura de novo conhecimento, através do desempenho de novas ações, ainda não executadas no ambiente.
  Ou seja, o agente explora outras possibilidades de ação além daquelas que já escolheu no passado, podendo vir a beneficiar de uma recompensa maior pela nova ação que executa.
  A exploração permite construir uma melhor estimativa em termos da função Q ideal, ao escolher uma ação diferente do que a que o agente pensa ser melhor naquele momento.
- **Investigação**, em oposição à exploração, consiste numa posição mais conservadora, onde o agente não arrisca executar novas ações. Sempre que o agente realizar uma ação, vai atualizando e guardando na função Q a recompensa que essa ação gerou. Ao utilizar as informações que possui pelas suas

experiências de atuação passadas, o agente escolhe a partir delas repetir desempenhar determinadas ações, por saber que receberá novamente a mesma recompensa pela sua prática.

Para que um sistema seja verdadeiramente autónomo e obtenha a maximização de recompensas no seu percurso até alcançar o seu objetivo, é necessário um balanço entre estes dois tipos de decisão por parte do agente.

2.5 Capacidade de Aprendizagem

O processo de aprendizagem presente neste tipo de algoritmos está exatamente no conceito de reforço após o desempenho de uma ação. Através da valorização ou punição de uma ação, não se indica ao agente como fazer corretamente a ação para a qual ele foi desenvolvido, mas são-lhe dadas indicações que permitem que o mesmo identifique quais as características das melhores decisões a tomar, de forma a adaptar e aprimorar as suas atuações no meio em prol do objetivo do problema.

Este processo de atribuição de avaliações às ações do agente, representa assim uma espécie de entidade imparcial e objetiva que obriga o agente a aprender, por tentativa e erro, quais são as melhores decisões que permitem maximizar o total de recompensas recebidas, até atingir um ponto em que desempenhe sempre a melhor ação no ambiente.

2.6 Ferramentas de Desenvolvimento

Por ser um dos métodos mais conhecidos dentro do tema de *Machine Learning*, existem inúmeras bibliotecas e esqueletos de código que permitem a sua utilização e introdução nas mais diversas linguagens de programação.

Em específico, destaca-se a biblioteca *PyBrain*, que apresenta métodos de aprendizagem por reforço, inteligência artificial e redes neuronais desenvolvidos em *python*. A biblioteca oferece assim uma forma simples de utilizar estes algoritmos e inclui ainda um conjunto de ambientes pré definidos para testar e comparar estes algoritmos.

Também dentro da linguagem *python* , existe ainda a ferramenta *TensorFlow*, originalmente desenvolvida por investigadores e engenheiros do projeto *Google Brain Team*. Esta ferramenta permite novamente a implementação de praticamente qualquer método relacionado com a aprendizagem máquina e adaptação de problemas para o contexto de aprendizagem por reforço.

2.7 Soluções existentes no mercado

Uma vez que os processos de aprendizagem por reforço não precisam de nenhuma adaptação trabalhosa para serem usados em áreas especificas, os mesmos podem ser aplicados a praticamente qualquer contexto ou tipo de problema que envolva tomada de decisões.

Para poder aplicar o algoritmo, basta apenas saber avaliar o desempenho de ações no ambiente que representa o universo do problema em estudo. Esta característica é uma grande vantagem em prol do uso deste algoritmo e permite assim que um grande número de problemas reais e relacionados com inteligência artificial possam ser mapeados para problemas de tomada de decisão cuja aprendizagem é feita por reforço.

Dos vários cenários de aplicação existentes, podem ser referidos alguns dos principais:

Produção

Existe no presente um grande número de fábricas que utilizam autónomos nas suas linhas de produção. Estes usam na sua implementação os conceitos de aprendizagem por reforço para desempenhar as tarefas necessárias ao fabrico das peças da empresa.

Como exemplo específico, a empresa *Fanuc* apresenta-se como uma das empresas lider no setor da automação industrial, abrangendo uma diversa gama de setores da indústria, desde áreas como a eng. aeroespacial, medicina, agricultura, embalamento e até mesmo relojoaria.

Um dos seus robôs, permite apanhar objetos de uma caixa e coloca-los num outro recipiente, por exemplo para embalamento ou passagem para uma nova etapa na linha de produção. Este robô, conforme falhe ou acerte no lugar onde coloca o objeto, está a aprender a realizar a tarefa através das atribuições de reforços às suas ações, sendo que após algum tempo é capaz de a realizar com rapidez e precisão a sua tarefa.

Gestão de Entregas

O algoritmo de *Q-Learning* pode ser utilizado para problemas relacionados com entregas de encomendas em que apenas um veículo tem que visitar todos os lugares e consequentemente tentar fazer o caminho mais vantajoso, neste caso, o caminho mais curto.

Gestão de Inventários

Através do uso de equipamentos especializados, tal como no setor da produção, os algoritmos de aprendizagem por reforço podem ser usados para reduzir o tempo necessário para armazenar e localizar produtos, melhorando assim estas operações dentro de um armazém e otimizando os espaços usados para a colocação de *stock*.

Setor Financeiro

Alguns contextos do setor financeiro, como a bolsa de valores, podem ser vistos como ambientes de atuação bastante instáveis, onde não é possível prever se o desempenho de uma ação vai trazer as consequências desejadas nesse ambiente.

Nestas situações, processos de aprendizagem por reforço com recurso ao *Q-Learning* apresentam-se assim como excelentes soluções para encontrar estratégias de negociação comercial e prever as melhores decisões a longo prazo.

Jogos

A aprendizagem por reforço, dentro do contexto de jogos, está a ganhar ímpeto, especialmente devido ao *DeepMind* da *Google*. Esta empresa, teve sucesso de implementação deste algoritmo em quase metade dos 50 jogos *Atari* em que ele foi aplicado. Tendo tido assim mais sucesso que qualquer outro método empregue previamente.

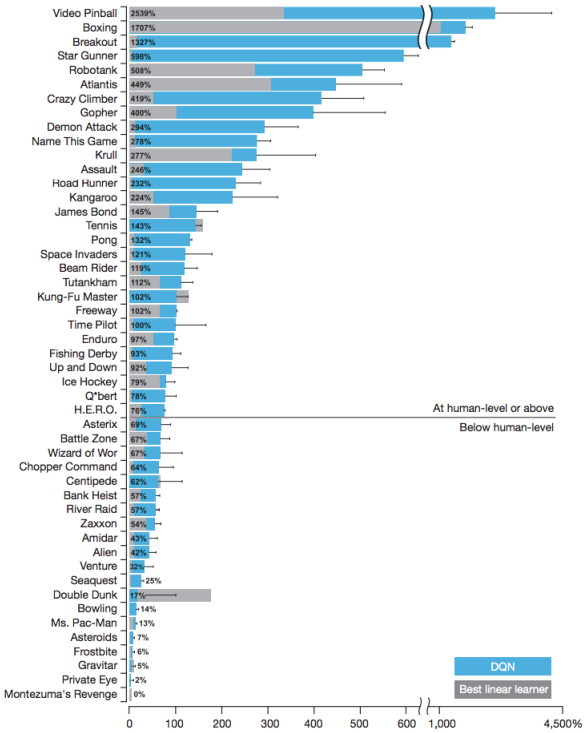

Figura 4. Escala representativa dos jogos que após o treino com aprendizagem por reforço ficaram abaixo do desempenho humano, ao mesmo nível e com melhor desempenho.

3 Algoritmos Genéticos

3.1 Descrição geral

Os algoritmos genéticos representam métodos de procura de soluções, geralmente aplicados em problemas de otimização ou aprendizagem, que se baseiam nos conceitos da teoria da evolução, apresentada por Darwin.

De uma forma geral, a partir de uma população inicial, que representa um conjunto de possíveis soluções para o problema, os elementos com maior probabilidade de maximizar a solução são selecionados e, a partir deles, são geradas novas gerações de soluções, cada vez melhores e mais adaptadas ao contexto do problema. À medida que novas gerações são criadas, as principais características de uma geração são combinadas e passadas à seguinte, criando assim novos indivíduos, com melhor adaptação ao meio.

Como a cada iteração se gera uma nova geração de soluções, quando o algoritmo convergir será possível obter não só uma possível solução para o problema, mas sim um conjunto de soluções, representadas por cada elemento da geração final após a convergência do algoritmo.

Como o algoritmo é regido por probabilidades, nunca duas iterações do algoritmo vão gerar as mesmas soluções. No geral, as soluções obtidas podem não representar uma solução ótima para o problema, mas serão soluções próximas da solução ótima.

A seguinte figura procura esboçar em pseudo-código as fases de um algoritmo genético.

```
timer t = 0;                                  //Iniciar contador tempo a 0
P(t) = inicia_populacao(t);                   //Gerar aleatoriamente a população inicial
cond_paragem = avaliar_populacao( P(t) );     //Avaliar a população inicial

while( !cond_paragem ){                       //Enquando nao se verifica o critério paragem...
    P(t+1) = seleciona_pais( P(t) );          //Selecionar os elementos com maior pontuação
    P(t+1) = reproduzir_e_mutar( P(t+1) );    //Aplicar operadores genéticos para gerar a nova geração
    t = t++;                                  //Incrementar o contador de tempo
    cond_paragem = avaliar_populacao( P(t+1) );
}
```

Figura 5. Esboço da estrutura do código de um algoritmo genético

3.2 Codificação de soluções

Considerando que um algoritmo genético é independente da área e contexto dos problemas onde é aplicado, a forma das soluções por si encontradas é uma representação dos parâmetros que caracterizam uma solução real para um problema. Esta codificação das variáveis de uma solução num elemento capaz de ser interpretado e processado pelo algoritmo, dá origem a um cromossoma.

Cada cromossoma contem assim informação relativa à solução candidata que representa. Esta codificação num cromossoma é geralmente uma cadeia binária,

de 0s ou 1s, podendo em algumas adaptações mais recentes do algoritmo ser também números inteiros, sequências de strings ou até mesmo árvores de decisões. Tal como na biologia, o conteúdo de um cromossoma é designado por genótipo e a interpretação semântica do seu conteúdo designa-se por fenótipo. Esta interpretação semântica será uma combinação do conteúdo de um cromossoma com o contexto do problema.

No caso de uma representação binária, um bit ou subconjuntos de bits podem representar alternativas sobre um determinado parâmetro da solução. Considerando como exemplo um problema de otimização no embalamento de produtos, podemos ter quatro variáveis na decisão:

Material a embalar	Espessura (cm)	Material lacrar caixa	Pegas laterais
00 – Ferro 01 – Alumínio 10 – Madeira 11 – Papel	$L \in [1,7], L\ inteiro$ 001 – 1cm 010 – 2cm ... 110 – 6cm 111 – 7cm	00 – Cola 01 – Fita cola 10 – Plástico	1 – Sim 0 – Não

Figura 6. Exemplo de parâmetros a decidir no embalamento de produtos.

Com o exemplo apresentado, qualquer cromossoma que represente uma solução candidata será uma sequencia binária de 8 bits. Por exemplo, o embalamento de uma placa de alumínio, com 5cm de espessura, cuja caixa é fechada com fita cola e apresenta pegas laterais dará origem ao cromossoma cuja sequencia binária é **01 011 01 1**.

3.3 População Inicial

O primeiro passo na implementação do algoritmo é a criação de uma população inicial. Os elementos que constituam esta população inicial devem ser gerados de forma aleatória, criando assim alguma diversidade de soluções. Tal como no suporte da teoria da evolução biológica, este fator de diversidade entre as soluções é essencial para descrever diferentes hipóteses de solução e criar futuras gerações que gradualmente se vão adaptando e especializando como solução do problema. Se a dimensão da população inicial for reduzida, é cortada a hipótese de criar sucessivas gerações e o algoritmo converge prematuramente, gerando soluções limitadas e pouco ótimas. Em oposição, populações iniciais vastas e com muitos casos aumentam consideravelmente o tempo de execução do algoritmo.

Tendo um conjunto de soluções iniciais, o algoritmo pode assim aplicar os métodos de evolução: seleção e reprodução dos elementos mais favoráveis e aplicação de mutações para manter a diversidade entre gerações.

3.4 Seleção de Soluções

A cada iteração, cada um dos elementos da população é analisado e avaliado a nível de qualidade enquanto possível solução para o problema a tratar. Esta medida é obtida através de uma função de avaliação, que deve ter em consideração o contexto específico do problema e privilegia os indivíduos melhor adaptados, ou seja, quanto mais próximo for um individuo de uma solução ótima, melhor é a avaliação atribuída pela função de avaliação a esse indivíduo.

Esta função objetivo, também designada por função de *fitness*, pode envolver o uso de simuladores com a solução em análise e é geralmente complexa de calcular.

Os indivíduos com maiores medidas, têm assim maior probabilidade de serem escolhidos e se "reproduzirem", originando uma descendência que combina as suas características.

Dos diversos métodos de seleção, destacam-se:

- **Método da roleta** : Depois de calculado o valor da função *fitness* para cada um dos cromossomas da população, a cada um deles é atribuída uma porção da roleta, conforme a proporção da sua pontuação com a soma de todos as pontuações da geração.
  Para que cada indivíduo receba uma porção da roleta, é necessário que a função de *fitness* retorne sempre um valor positivo superior a zero.
  Neste método, a probabilidade de reprodução será assim tanto maior quanto o valor dado pela função de *fitness*. No caso de um cromossoma ter uma classificação muito elevada, pode levar a que o mesmo ocupe a maior parte de roleta e, portanto, seja sempre o escolhido para reprodução. Esta situação levaria à perda de diversidade genética.

- **Método Elitista** : Devido aos operadores genéticos que se vão aplicar para gerar novos cromossomas, pode acontecer que as melhores características dos parentes da geração a ser criada sejam perdidos na sua descendência.
  Para contornar este fator e evitar a perda de diversidade genética, existem métodos de seleção que transportam para a nova geração alguns dos melhores cromossomas da geração anterior, garantindo assim a presença das melhores soluções nas gerações futuras.

- **Método Torneio** : Perante a população de soluções, o método de seleção por torneio escolhe um conjunto de **N** cromossomas e é escolhido para reprodução o indivíduo desse sub grupo com melhor pontuação segundo a função de avaliação.
  Depois de selecionado o melhor indivíduo, os elementos são "repostos"na geração atual e outros subconjuntos de **N** elementos são retirados aleatoriamente, até se ter o número suficiente de cromossomas para a fase seguinte de reprodução.
  Se este parâmetro **N** for demasiado grande, a seleção de um grande número de elementos da população em simultâneo, pode levar à perda de variabili-

dade genética, porque o indivíduo com a maior pontuação do subconjunto vai-se sobrepor a outros elementos vantajosos desse sub conjunto.

3.5 Reprodução

Organizando os cromossomas selecionados por pares para reprodução, é aplicada a técnica de *crossover* através da definição de pelo menos um ponto de corte. A posição de corte representa assim a zona onde os dois cromossomas irão trocar as suas características, gerando assim uma descendência que os combine.

No caso de cromossomas com representação binária, a posição de corte pode tirar significado ao cromossoma, sendo assim um processo de verificação e correção dos cromossomas gerados.

A figura seguinte procura esquematizar a criação de duas gerações através da técnica de reprodução por *crossover*. No primeiro esquema a descendência é criada por um plano de corte e, no segundo exemplo, por dois planos de corte.

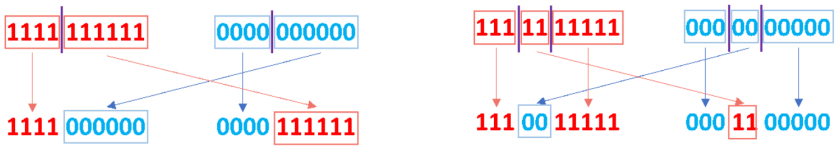

Figura 7. – Exemplos da técnica de reprodução por crossover.

De forma semelhante, o mesmo conceito de cruzamento de indivíduos pode ser aplicado a outras formas de representação de soluções, como árvores de decisão.

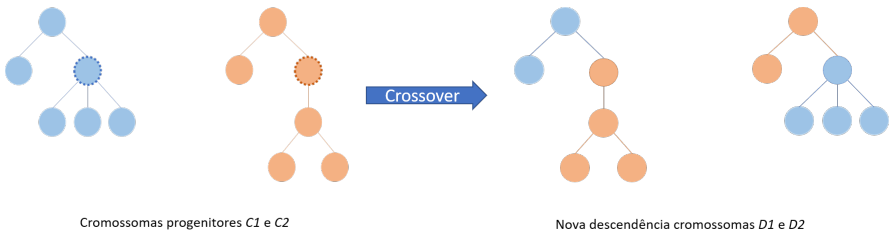

Figura 8. Exemplo da técnica de reprodução por crossover em soluções representadas por árvores.

3.6 Mutação

Além da diversidade gerada pelo cruzamento dos melhores cromossomas na fase de reprodução, é ainda aplicada uma fase de mutação a alguns elementos da

nova população. Tal como no contexto biológico, as mutações ocorrem de forma extraordinária e rara e procuram dotar os indivíduos de novas características, possivelmente vantajosas para a sua adaptação ao meio.

No contexto dos algoritmos genéticos, a aplicação de mutações a cada cromossoma tem uma probabilidade baixa de acontecer, porque elevadas taxas de mutação poderiam levar à perda de boas características herdadas previamente pelos cromossomas progenitores.

No caso de uma codificação binária, uma mutação corresponde à alteração aleatória do valor de um dos seus bits. Por exemplo um cromossoma com a sequencia binária **11010110** pode mutar-se para **11010010**. Cada bit de um determinado cromossoma tem a mesma probabilidade de ser mutado.

3.7 Capacidade de Aprendizagem

A capacidade de aprendizagem nos algoritmos genéticos pode ser analisada na fase de reprodução do mesmo. Considerando que só os melhores indivíduos são selecionados para esta fase, um par de cromossomas selecionado para o cruzamento vai dar origem a duas novas soluções que irão pertencer à geração futura. Estas novas soluções geradas, vão herdar uma fusão das características dos seus progenitores, adquirindo dos melhores indivíduos das gerações passadas aquilo que pode ser visto como o seu conhecimento.

Com este processo, as soluções que se encontram na última iteração do algoritmo, vão então conter as principais e melhores características de todos os cromossomas que fizeram parte da sua antecedência, adquirindo e aprimorando o seu conhecimento ao longo das gerações.

3.8 Ferramentas de Desenvolvimento

Considerando a ferramenta de análise de dados *WEKA*, é possível aplicar métodos genéticos utilizando, por exemplo, o classifier BayesNet. Dentro das opções deste classifier, na opção *searchAlgorithm* surge um processo designado por *GeneticSearch* que segue os conceitos de algoritmos genéticos apresentados nesta secção. Nas suas opções é possível definir parâmetros como:

- Ativar ou desativar os operadores genéticos de crossover e mutação;

- Definir o tamanho da população

- Definir qual o método de seleção dos melhores indivíduos a cada ciclo do algoritmo.

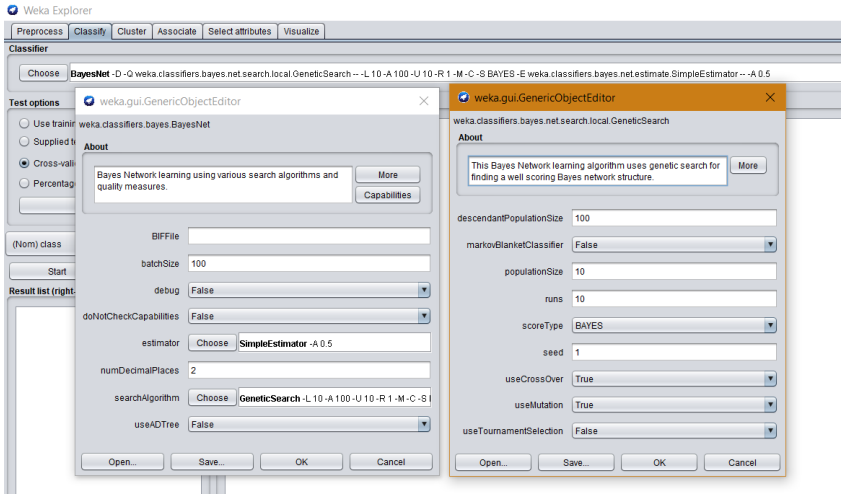

Figura 9. Parâmetros fornecidos pela ferramenta *WEKA* para configurar o funcionamento de uma pesquisa segundo um algoritmo genético.

Outras ferramentas de processamento de dados, como o *MathLab* permitem utilizar um processo de algoritmos genéticos para, por exemplo, encontrar os extremos de uma dada função, através do solver designado por *ga* (*genetic algorithm*). Um exemplo da aplicação deste método, para encontrar o minimo de uma função é explicado com detalhe na documentação do matLab [11].

Além destes exemplos mais específicos, existem ainda bibliotecas que permitem incluir algoritmos genéticos num programa, como o caso do *JCLEC - Java Class Library for Evolutionary Computation* e inúmeros exemplos da estrutura do algoritmo, para poder ser implementado em linguagens mais recentes, como *python*.

3.9 Soluções existentes no mercado

Aplicação na área das comunicações rádio

Considerando setores que operam em áreas críticas ou situações limite, baseando a sua comunicação em redes móveis, é necessário desenvolver antenas capazes de reproduzir um determinado comportamento de emissão, para que as comunicações sejam garantidas de forma correta.

Este processo, quando feito manualmente, exige porções de tempo e trabalho bastante altos para os resultados limitados que obtém. Com a evolução dos algoritmos genéticos e dos simuladores de eletromagnetismo, a combinação destes dois métodos passou a ser utilizada no processo de desenvolvimento da forma de uma antena.

O *design* da solução final é assim totalmente modelado computacionalmente e apresenta os melhores padrões de emissão e receção. Um exemplo desta reali-

dade é o projeto *NASA's Space Techonogy 5*[10] cuja imagem seguinte procura mostrar.

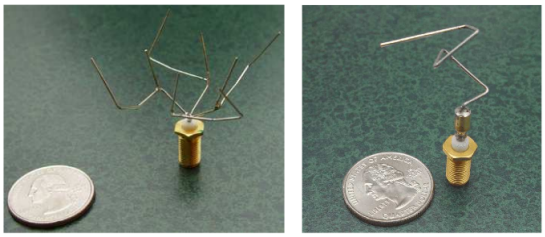

Figura 10. Exemplo de dois protótipos de antenas desenhadas através de algoritmos genéticos

Design aplicado à Engenharia

Algoritmos genéticos podem ser usados para tirar partido da grande variedade de materiais existentes, podendo otimizar assim o *design* estrutural e operacional de casas, prédios, fábricas e de certas máquinas. Combinar estes dois elementos, ou seja, algoritmos genéticos e problemas de engenharia, pode ser uma tarefa difícil, mas que pode não só resolver problemas de *design* mas também pode passar a detetar erros e pontos fracos dos modelos para que futuramente possam ser corrigidos e evitados.

Geração de anedotas e trocadilhos

Dentro das várias aplicações de algoritmos genéticos existem aqueles que geram anedotas e que criam jogos de palavras. Este tipo de algoritmo pode ser uma ajuda válida em profissões como comediantes e palhaços, que vêm o seu sucesso a ser determinado pelo seu público alvo. Este processo permite inserir uma palavra ou uma expressão com a qual se queira formular um trocadilho ou uma piada e gera-os de forma autónoma.

Jogos de Computador

Neste contexto *gaming* temos o exemplo de aplicação de algoritmos genéticos para a criação de civilizações que interagem com a nossa personagem quando jogamos por exemplo em modo *offline*.

4 Máquinas de Vetores Suporte

4.1 Descrição geral

Máquinas de vetores de suporte, ou *Support Vector Machines* (SVM), considerando a convencional designação em inglês, são algoritmos de otimização matemática baseados numa aprendizagem supervisionada. Tendo ganho atualmente uma atenção crescente, são normalmente aplicados a contextos de tratamento, classificação de dados e processamento de imagens.

Num algoritmo SVM, cada ponto do conjunto de dados é representado num espaço de $\boldsymbol{N}$ dimensões, onde $\boldsymbol{N}$ representa o número de atributos das classes em estudo. Um ponto é assim reproduzido pelo valor numérico dos seus atributos nos respetivos eixos que definem o espaço de representação. Cada um destes pontos individuais é visto como um vetor de suporte e a fronteira que delimita as classes de dados é chamada por máquina de vetores suporte.

Com a aplicação do algoritmo SVM a um conjunto de dados, pretende-se criar um conjunto de regras que permitam analisar os atributos de cada registo e classifica-lo como pertencente a uma das classes do contexto do problema.

Por exemplo, num conjunto de dados para avaliar o sexo de cada indivíduo, usando como parâmetros a altura do individuo e o comprimento do seu cabelo, teremos duas classes (sexo feminino e sexo masculino) e dois atributos (altura e comprimento do cabelo do indivíduo), podendo os dados do problema ser representados num espaço de 2 dimensões.

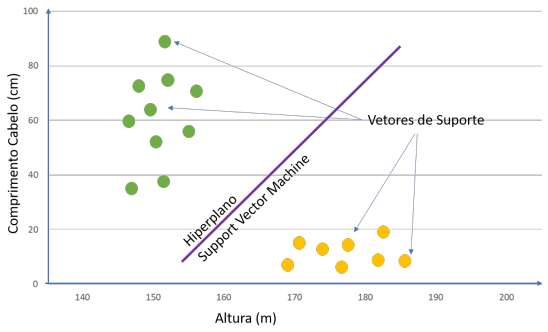

Figura 11. Seleção do melhor hiperplano de corte maximizando as margens entre as classes

4.2 SVM Linear

A tarefa do algoritmo SVM, enquanto classificador, é assim a criação de uma técnica de classificação para reconhecer os padrões associados a cada uma das classes, através do valor dos atributos dos casos que recaem nessas classes.

Mais especificamente, o SVM irá definir a equação de um hiperplano capaz de delimitar e categorizar as classes da melhor forma. De referir que em alguns casos podem ser usados mais do que um hiperplano para delimitar as classes.

No caso de existirem diversos hiperplanos aparentemente semelhantes para dividir as classes, o critério de seleção recai sobre o conceito de maximização das margens. Para cada hiperplano, é calculada a distância até ao vetor de suporte mais próximo de cada classe. O hiperplano que maximizar esta distância entre as diferentes classes é o selecionado.

Este procedimento de maximização das margens entre as classes e o hiperplano, é utilizado para aumentar a robustez da solução ótima calculada. A definição de um hiperplano mais próximo de uma classe do que de outra, pode levar à classificação incorreta de casos no futuro. Com isto, o modelo de classificação gerado será capaz de categorizar corretamente os casos com que foi treinado, mas estará pouco preparado para catalogar casos futuros, com possíveis variações nos padrões dos atributos.

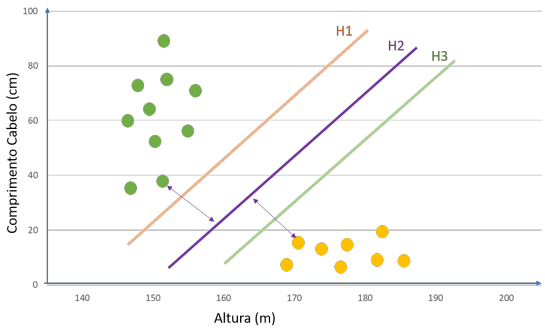

Figura 12. Representação de dados no contexto apresentado como exemplo.

Considerando os hiperplanos apresentados na figura:

- O plano de corte **H1**, por ter uma margem muito próxima aos casos da classe *Sexo Feminino*, classificaria como pertencentes à classe *Sexo Masculino* casos que são da classe *Sexo Feminino*;
- De forma análoga, o plano de corte **H3**, leva a um classificador que considera erradamente da classe *Sexo Feminino* casos que pertencem à classe *Sexo Masculino*;
- O hiperplano **H2**, representa assim a solução ótima, por procurar maximizar as margens entre as duas classes. Desta forma permite que em avaliações com novos dados exista uma flexibilidade na variação dos valores dos atributos, sem que os mesmos sejam classificados de forma errada.

Acima deste critério de maximização das margens entre classes, o algoritmo SVM procura selecionar o hiperplano que maximiza a precisão da previsão para

cada classe. Desta forma, o hiperplano vai tentar dividir explicitamente todos os casos analisados no treino do classificador, mesmo que este processo implique que a solução final se apresente com uma margem inferior perante uma determinada classe.

No caso do conjunto de dados conter entradas que se identifiquem como *outliers*, o algoritmo tem ainda a capacidade de os detetar e os ignorar, mantendo os processos acima referidos.

A figura seguinte procura esquematizar o cenário em que o critério de precisão na divisão de classes se sobrepõe ao critério de maximização das margens entre classes e um segundo cenário onde a presença de *outliers*, que impede que um plano linear corte as duas classes, é descartada da análise permitindo assim definir o plano ***H1*** de forma correta.

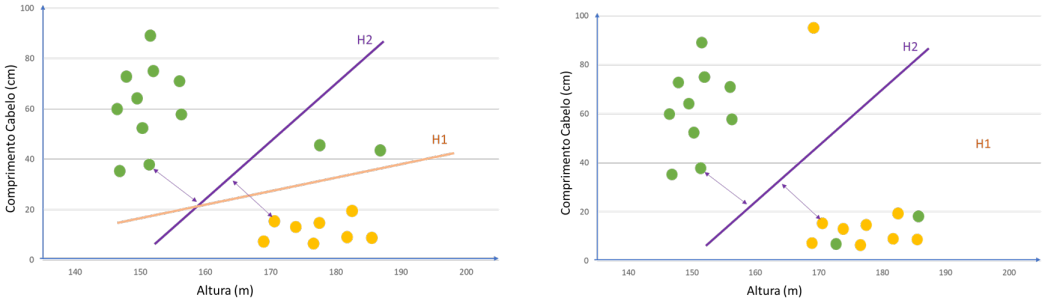

Figura 13. Representação de cenários excecionais

4.3 SVM não linear e *kernel trick*

No caso das classes não permitirem uma separação linear, o algoritmo SVM tem a capacidade de adaptar o contexto do problema, considerando um domínio de representação superior ao inicial, procurando nessa nova representação dos dados obter a divisão entre as diferentes classes.

Esta conversão é realizada automaticamente pelo algoritmo, através de um processo chamado de *kernel trick*. O processo é associado a um "truque" pelo facto de facilitar o aumento do domínio quando o espaço de representação passa a ser superior a três dimensões.

Apesar do aumento da dimensão do espaço de soluções levar a uma maior complexidade na visualização dos dados, a aprendizagem do algoritmo para definir o híperplano de corte apresenta uma maior simplicidade, daí esta técnica ser realizada.

De referir que o plano de corte obtido neste método deixa de ser necessariamente uma separação linear entre classes. O exemplo da figura abaixo procura esquematizar a passagem de um domínio em $\mathbb{R}^2$ para $\mathbb{R}^3$, considerando $z = x^2 + y^2$

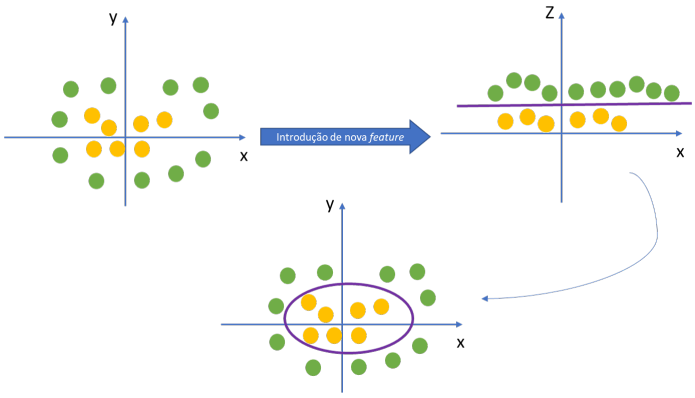

Figura 14. Exemplo da aplicação do *Kernel Trick*

4.4 Capacidade de Aprendizagem

Uma vez que a aplicação de SVM se enquadra dentro de contextos de aprendizagem supervisionada, é nesta característica que se denota a sua capacidade de aprendizagem. Através de existência de um conjunto de dados robusto, aplicados no processo de treino do algoritmo, o mesmo será capaz de extrair o conhecimento representado por esses dados e criar um modelo de previsão que permita classificar corretamente novas entradas no futuro, que possivelmente serão exemplos que não estavam sequer representados no conjunto usado para treino.

Este processo de indução e perceção de padrões no conjunto de dados inicial é onde se reflete a capacidade de aprendizagem do algoritmo, que ao longo das suas iterações procura encontrar um modelo capaz de prever corretamente e com precisão os novos dados sobre os quais o modelo seja aplicado.

Além do conceito de aprendizagem, é ainda relevante realçar a capacidade de adaptação do algoritmo, através da minimização da influência de possíveis *outliers* no processo de indução e na adaptação do espaço de representação dos dados, para procurar uma divisão não linear que separe todas as classes de um dado problema.

4.5 Ferramentas de Desenvolvimento

Para o desenvolvimento de soluções baseadas em SVM, existe o *software* *LIBSVM* que disponibiliza um conjunto de métodos para resolver problemas recorrendo a classificadores SVM, regressões e estimativa de distribuições.

Esta ferramenta apresenta extensões e interfaces para praticamente todas as principais linguagens de programação desde java, python, perl, C, .NET, CUDA, MatLab, Weka e R.

No caso da ferramenta *Weka*, existe já por omissão o classificador functions.SMO, que permite utilizar o conceito de SVM aplicados a conjuntos de

dados. Dentro dos vários parâmetros possíveis de regular destaca-se a disponibilização de vários tipos de funções para utilizar no mecanismo de *kernel*.

4.6 Soluções existentes no mercado

Deteção de *spam* através de SVM

A ideia inicial de utilizar algoritmos de Máquinas de Vetores de Suporte para classificar emails como sendo ou não de *spam* surgiu inicialmente abordado no artigo *Support Vector Machines for spam Categorization*[14], onde os autores procuravam avaliar o desempenho de quatro algoritmos no contexto indicado.

Na modelação do problema, cada palavra de uma mensagem de email é vista como uma característica (*feature*) e a análise do conteúdo de todos os dados de treino permitiram criar um dicionário de palavras.

Tendo este dicionário de palavras, de dimensão N, foi criado para cada email analisado um vetor de características $\boldsymbol{x}$ com a mesma dimensão. Neste vetor, a posição $\boldsymbol{i}$ corresponde à característica/palavra que ocupa a posição $\boldsymbol{i}$ do dicionário de dados.

Neste processo foram tidos em conta dois casos:

- Frequência da palavra: o valor da coordenada na posição $\boldsymbol{i}$ do vetor $\boldsymbol{x}$ indica o número de vezes que a palavra aparece na mensagem;
- Representação binária: O valor contido no índice $\boldsymbol{i}$ do vetor $\boldsymbol{x}$ indica se a palavra aparece ou não na mensagem.

Após a análise dos resultados, é apresentado na conclusão do estudo que convertendo todas as palavras para caracteres em *lower case* e utilizando a representação binária, o algoritmo SVM tem um rendimento bastante aceitável a nível de velocidade e precisão na classificação do conteúdo dos emails.

Reconhecimento do género do através de imagens

Um projeto recente, datado de 2016, envolvendo a aplicação de SVM está relacionado com o reconhecimento do género de um indivíduo através do uso de imagens térmicas e imagens retiradas por câmaras normais.

Este processo é extremamente relevante e de interesse em contextos relacionados com sistemas de segurança e vigilância ou a criação de análises estatísticas relativamente ao número de elementos de cada sexo que utilizam, por exemplo, um determinado espaço comercial ou estações de serviços públicos.

Considerando sistemas de interação homem-máquina, como caixas de multibanco, esta possível deteção em tempo real permite, por exemplo, que o sistema seja capaz de responder com "Sr."ou "Sra."aos seus utilizadores, baseado no seu género, criando assim uma maior imersividade e proximidade na interação com os sistemas.

Como variáveis externas como sombras, iluminação e roupas afetam a capacidade de processamento das imagens, este projeto procura diferenciar-se pela

combinação de imagens térmicas em conjunto com imagens tiradas no espectro de luz visível.

No desenvolvimento deste sistema a aplicação de classificadores SVM surge depois do pré-processamento e extração das características dos dois tipo de imagens referidas. Esta análise das imagens representa assim o conjunto de dados que é utilizado para criar um modelo de previsão.

De uma forma geral, os testes realizados permitiram concluir que o uso dos dois tipos de imagens e dois tipos de classificadores SVM em fases diferentes do algoritmo, permitiu obter resultados mais concisos e precisos face a outros sistemas de reconhecimento existentes. A experiência encontra-se apresentada com mais detalhe no artigo *Body-Based Gender Recognition Using Images from Visible and Thermal Cameras* [15]

Outras aplicações

- Deteção Facial: São usadas Máquinas de Vetor Suporte para classificar zonas de uma imagem, fazendo a divisão entre o rosto e o resto da imagem onde não existe um rosto;
- Categorização de texto: Categorização de texto é o ato de, dado um conjunto de assuntos e uma coleção de documentos em texto, mapear o tópico correto para um texto específico.
  Neste contexto, são usados SVM para classificar os documentos nas diferentes categorias. Este processo de categorização, baseia-se na geração de valores e compara-los com o valor de *threshold*;
- Bioinformática: Casos resolvidos com utilização de *Support Vector Machines* incluem problemas de classificação de proteínas ou deteção de células cancerígenas.

5 Conclusão

Iniciando o artigo com a abordagem ao processo de aprendizagem por mecanismos de Reforço, é apresentada nesta primeira secção um conjunto de tópicos que permitissem realizar uma descrição geral e explicação do modo de funcionamento do algoritmo de **Aprendizagem por Reforço**, relacionando a sua aplicação com a noção de agentes inteligentes e o ambiente onde os mesmos atuam. Dentro deste contexto de aprendizagem, foram referidos os principais algoritmos que aplicam os seus conceitos e é ainda uma análise do problema *Exploration vs Exploitation*.

Após o estudo realizado neste algoritmo podemos avaliar que, como qualquer outro processo, existem vantagens e desvantagens. As principais vantagens consistem no facto do processo não ser computacionalmente complexo, nem ser necessário um conhecimento vasto sobre o contexto ou o tipo de ambiente onde se insere o sistema. Para que se obtenham bons resultados, basta apenas saber

avaliar o desempenho de uma ação, mesmo não sabendo se ação praticada é ou não ótima.

A nível de desvantagens, pode ser referido que este processo de exploração de ações se baseia num processo de tentativa-erro para aprender. Em situações limite, o desempenho de uma ação que traga consequências negativas no ambiente, que indiretamente afetem o agente, pode prejudicar a sua capacidade de alcançar os seus objetivos. Por exemplo, um sistema complexo regido por este mecanismo de aprendizagem pode tomar uma ação que o danifique, tornando o seu processo de aprendizagem autónoma possivelmente dispendioso.

Na secção seguinte, são apresentados os conteúdos que abordam o conceito de algoritmos evolutivos, focando com detalhe **Algoritmos Genéticos**. Como referido anteriormente, este processo baseia-se nas bases da Teoria da Evolução Natural de Darwin, usando o conceito de "sobrevivência dos elementos mais aptos"para encontrar as melhores soluções para um problema. Adaptando conceitos dos processos biológicos como seleção natural, reprodução e mutação, o algoritmo procura simular a forma como estes operadores genéticos ocorrem com os cromossomas e DNA, mas aplicando-os a problemas reais.

Como ponto de interesse nesta abordagem, refere-se o facto do algoritmo ser intrinsecamente paralelo. Por utilizar uma população de possíveis soluções em cada iteração, o algoritmo trabalha simultaneamente sobre várias soluções para o problema, diferindo assim de outros algoritmos tradicionais que operam soluções singulares de forma sequencial.

Além deste aspeto, o algoritmo em si não precisa de conhecimentos específicos dos problemas sobre os quais é aplicado. A sua forma de iteração é geral, sendo apenas nas fases de codificação das soluções e definição da função objetivo (*fitness*) que se tem em consideração o contexto especifico do problema a resolver.

Como aspetos negativos, denota-se a sensibilidade do algoritmos a parâmetros como o tamanho l da população inicial e ao número de gerações máximo, definido como critério de paragem. Se não existir uma correta conjugação destes parâmetros com a dimensão do problema a resolver, o algoritmo pode demorar demasiado tempo a convergir, ou até nem o fazer. Como problema oposto, o processo evolutivo pode também convergir prematuramente se logo de inicio surgir por acaso um individuo com boas características. A existência desta aparente solução ótima logo no inicio da iteração, pode ocultar outras soluções viáveis para o problema.

Por fim, explora-se o tema recente das **Support Vector Machines**. Além de uma descrição geral, são abordados detalhes mais específicos como a utilização deste método em conjuntos de dados possíveis de classificar com hiperplanos lineares e situações mais complexas, onde a dimensão dos dados exige que se usem métodos não lineares para representar e encontrar uma separação concisa entre as classes.

Como aspetos que favorecem a utilização de SVM, destaca-se o facto do algoritmo ter um desempenho e resultados muito satisfatórios, mesmo em conjuntos de dados com poucos casos de treino ou com um grande número de atributos. Na prática, não existe nenhum limite ao número de atributos, a não ser possíveis

limitações a nível do hardware usado para processar o algoritmo. Apesar de não abordadas neste artigo, processos de aprendizagem através de Redes Neuronais tradicionais, não apresentam um bom desempenho sobre um grande volume de atributos nos dados em comparação com SVM.

Tal como acontece com outros tipos de classificadores, a aplicação de SVM exige que o conjunto de dados de treino esteja corretamente normalizado, tendo os atributos que estar dentro do domínio numérico. No caso de atributos categóricos (não numéricos) é normalmente possível criar uma escala de conversão entre os valores iniciais e um mapeamento numérico, ou criar algum tipo de truque de codificação alternativa, contornando assim este problema.

De uma forma geral, existindo um conjunto de dados sobre os quais possa ser possível analisar padrões e, não existindo a possibilidade de descrever e resolver o problema matematicamente, podemos então concluir que estamos numa situação onde a aplicação de processos de aprendizagem é viável para a extração de conhecimento útil na análise e resolução do problema.

Referências

1. Hochländer aus Wiesbaden, Aaron. *Deep Learning for Reinforcement Learning in Pacman* (2014, Juli)
   Universidade técnica de Darmstast
2. Rangel Guimarães Serra, Mauricio. Aplicações de Aprendizagem por Reforço em Controlo de tráfego Veicular Urbano. (2004)
   Universidade Federal de Santa Catarina
3. Pinõl, Michael. *Exploration versus Exploitation Dilemma in Reinforcement Learning.*
   Universidade de Stanislaus, California.
4. T. Monteiro, Sildomar, H. C. Ribeiro, Carlos. Desempenho de Algoritmos de aprendizagem por reforço sob condições de ambiguidade sensorial em robótica móvel.
   Departamento Ciência da Computação, Instituto Tecnológico de Aeronáutica, São Paulo, Brasil.
5. Pellegrini, Jerônimo, Wainer, Jacques. Processos de Decição de Markov: um tutorial [online] Available at:
   http://www.seer.ufrgs.br/rita/article/viewFile/rita_v14_n2_p133-179/3544
   [Accessed 16 Oct. 2017].
6. Marcus V. C. Guelpeli, Carlos H. C. Ribeiro e Nizam Omar. Utilização de Aprendizagem por Reforço para Modelagem Autónoma do Aprendiz em um Tutor Inteligente. Departamento de Eng. Elétrica, Instituto tecnológico de Aeronáutica, São Paulo, Brasil.
7. Arranz de la Peña, Jorge and Parra Truyol, António. (n.d.). Algoritmos Genéticos. Departamento de Engenharia Telemática, Universidade Carlos III de Madrid.
   [online] Available at: http://www.it.uc3m.es/jvillena/irc/practicas/06-07/05.pdf
   [Accessed 16 Oct. 2017].
8. Goldberg, D. (2012). Genetic algorithms in search, optimization, and machine learning. Boston [u.a.]: Addison-Wesley.
9. Souza Santos, J. (2008). Mineração de Dados Utilizando Algoritmos Genéticos.
   Universidade Federal da Bahia.

10. Hornby, G. S., Globus, A., Linden, D. S., Lohn, J. D. (2006, September). Automated antenna design with evolutionary algorithms.
11. MatLab Documentation: find minimum of function using genetic algorithm. [online] Available at: https://www.mathworks.com/help/gads/ga.html
12. Carolina Lorena, Ana, Carvalho, André F., Uma introdução às *Support Vector Machines*
13. Silva, Luis Miguel Domingues Ferreira (2014). Máquinas de Vetores Suporte para Classificação do Onset em dados temporais de Eletromiografia. Universidade Aberta
14. H. Druker, Donghui. Wu, Vladimir N. Vapnik. *Support Vector Machines for Spam Categorization.* IEEE Transactions on Neural Networks, 1999.
15. Nguyen, Dat Tien, Park, Kang Ryoung. Body-Based Gender Recognition Using Images from Visible and Thermal Cameras [online] Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4801534/ [Accessed 20 Oct. 2017].