Sistemas de Aprendizagem

André Freitas[1] e Dalila Reis[1]

[1] Universidade do Minho, Braga, Portugal

Abstract. Existe, atualmente, uma grande variedade de sistemas de aprendizagem e neste trabalho abordamos alguns deles, fazendo a sua descrição característica, referindo algumas das vantagens e desvantagens destes sistemas, assim como as ferramentas de desenvolvimento que existem e as soluções disponíveis no mercado baseadas em cada um, i.e., algumas das suas aplicações reais. Mais concretamente, e após uma introdução geral aos sistemas de aprendizagem, começamos por abordar as *Support Vector Machines*, que constituem um modelo de aprendizagem supervisionada que permite a previsão de dados futuros, estando apoiado numa base teórica matemática muito forte Depois é tratado o tema das *Decision Trees*, um sistema graficamente apelativo em árvore, como o próprio nome indica, que permite fazer previsões consoante os valores de determinadas variáveis de *input*. Por último, analisamos *Case-Based Reasoning*, um tipo de sistema de aprendizagem que usa as experiências passadas para resolver problemas novos. No final, é feita uma reflexão sobre o trabalho realizado e os conhecimentos adquiridos sobre estes sistemas.

Keywords: Sistemas de Aprendizagem, *Support Vector Machines, Case-Based Reasoning, Tree Decisions.*

1 Introdução

Os sistemas de aprendizagem permitem a agentes executar tarefas e resolver vários tipos de problema com sucesso, baseando-se na experiência, aquisição de dados e erros cometidos. Essa aprendizagem pode ser supervisionada, não supervisionada ou baseada em experiências e, ao longo deste trabalho, ficarão esclarecidos estes termos e serão, também, abordados alguns destes sistemas. Mais especificamente, abordaremos *Support Vector Machines* (pp. 1 a 10), *Case-Based Reasoning* (pp. 10 a 15) e *Decision Trees* (pp. 15 a 21), fazendo referência a vários aspetos relevantes em cada um destes temas.

2 *Support Vector Machine* (SVM)

Uma Máquina de Vetores de Suporte (SVM - *Support Vector Machine*) é um modelo de aprendizagem supervisionada, o que significa que é necessário treinar a SVM com um conjunto de dados (*training data*). É um sistema com uma extensa fundamentação

teórica apoiado, principalmente, na Teoria da Aprendizagem Estatística e na Otimização Matemática [12]. É apresentado em 1963 por Vapnik e Chervonenkis e, até 1996, vai sendo desenvolvido e aperfeiçoado por esses e outros autores [15].

Este tipo de sistema de aprendizagem pode ser utilizado em problemas de classificação (mais frequente) ou de regressão (SVR – *Support Vector Regression*), para aprender a previsão de dados futuros [6]. As SVMs podem dividir-se em duas categorias gerais, conforme os dados analisados:

1. **Dados linearmente separáveis:** quando existem infinitas linhas ou hiperplanos que separam os dados em duas classes distintas - SVM linear com Margem Rígida (Classificador de Margem Máxima);
2. **Dados não linearmente separáveis:** quando as duas classes se sobrepõem – SVM linear de Margem Suave e/ou SVM não-linear com Kernels (Truque de Kernel).

SVM linear com Margem Rígida (*Hard-margin linear SVM)*

Os dados de entrada são separados em duas classes distintas com etiquetas y +1 ou -1. O conjunto de treino, com n dados xi ∈ X, é representado como um conjunto de pontos num espaço de n-dimensões, em que n é o número de características ou variáveis da amostra e o valor de uma coordenada particular é o valor de uma das variáveis.

Com dados linearmente separáveis, existem infinitos hiperplanos que separam os exemplos positivos dos negativos. O objetivo da SVM é encontrar o hiperplano de separação ótimo que maximiza as margens [15], i.e., aquele que está mais distante dos pontos mais próximos de qualquer uma das classes. Estes pontos são denominados vetores de suporte (ver Fig. 1).

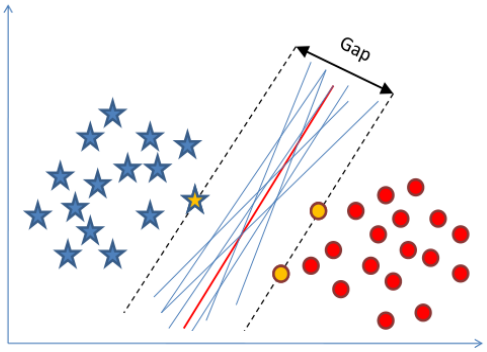

Fig. 1. Representação dos dados de treino num gráfico. Os vetores de suporte são os pontos a amarelo [14].

Este hiperplano não será tendencioso para qualquer uma das classes e, assim, é expectável que providencie os resultados mais precisos. A equação de um hiperplano é apresentada na Equação 1, em que $w.x$ é o produto escalar entre os vetores w e x, w é designado como vetor peso (largura da margem corresponde a $\frac{\|w\|}{2}$) e b como a tendência [1].

$$f\,(x) = w\,.x + b = 0 \tag{1}$$

Como esta equação divide o espaço dos dados X em duas regiões (evidente na Fig. 2), uma função sinal g(x) = sgn(f(x)) pode ser utilizada na obtenção das classificações [7].

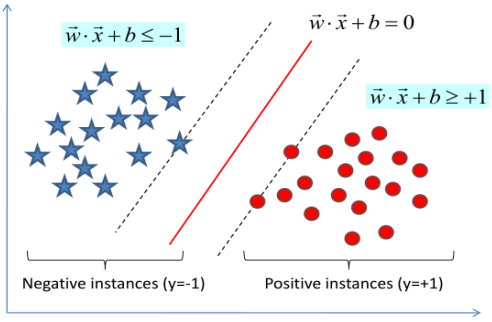

Fig. 2. Hiperplano que separa os dados de treino em duas classes distintas [14].

Calcular o hiperplano de margem máxima resulta num problema de otimização quadrática convexa, em que a função objetivo é minimizar $\|w\|$, com restrições que asseguram que não existem dados entre as margens de separação, daí a designação de SVM com Margens Rígidas (Equação 2) [12].

$$\underset{\mathbf{w},b}{\text{Minimizar}} \ \frac{1}{2}\|\mathbf{w}\|^2 \qquad (2)$$
$$y_i\left(\mathbf{w}\cdot\mathbf{x}_i+b\right)-1\geqslant 0, \ \ \forall i=1,\ldots,n$$

SVM linear com Margem Suave (*Soft-margin linear SVM*)

Geralmente, em situações reais, os dados não são linearmente separáveis e, mesmo nesse caso, é possível obter uma margem maior se for permitido classificar erradamente alguns pontos [1]. Esta estratégia tolera ruído e *outliers* nos dados, considera mais pontos de treino além dos que estão na fronteira (vetores de suporte) e permite a ocorrência de erros de classificação [7].

Para esse efeito, introduzem-se variáveis de folga ξi (i = 1,..n) ≥ 0 no problema, que permitem que um exemplo se situe na margem (erro de margem, quando $0 \leq \xi i \leq 1$) ou que seja mal classificado (erro de classificação, quando $\xi i > 1$). O somatório dos ξi representa o limite nos erros de classificação [1]. O problema de otimização é agora:

$$\underset{\mathbf{w},b}{\text{minimize}} \qquad \tfrac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i \qquad (3)$$
$$\text{subject to:} \quad y_i(\mathbf{w}^{\mathsf{T}}\mathbf{x}_i+b)\geq 1-\xi_i, \quad \xi_i\geq 0.$$

A constante C > 0 (parâmetro de custo ou constante da margem suave) penaliza os erros de classificação e de margem. Quando C é um valor muito alto, o modelo não permite erros de classificação no conjunto de treino, tornando-se um modelo SVM de Margem

Rígida (suscetível a *overfitting*). Na prática, o parâmetro C é determinado experimentalmente (tentativa e erro) usando um conjunto de validação distinto ou através da técnica de validação cruzada (*cross-validation*) que verifica o desempenho ótimo de C usando apenas o conjunto de treino [3] [10].

SVM não-linear com Kernels (Kernel trick)

Um dos pontos fortes da SVM é que consegue classificar dados que não são linearmente separáveis através da projeção destes através de uma função não linear $\varphi : X \rightarrow F$ para um espaço com uma dimensão mais alta (*mapping*). Isto significa que os pontos dos dados são "levantados" do seu espaço original (X: *input space*) para um novo espaço (F: *feature space*) onde existe um hiperplano que separa as duas classes [5] (Fig. 3).

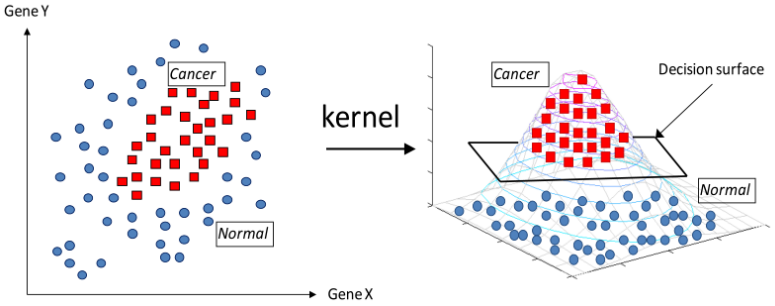

Fig. 3. *Mapping* dos dados de treino para um novo espaço de características (*feature space*) [14].

O problema de computar explicitamente as características não-lineares de um conjunto de dados com n características reside no aumento quadrático de recursos, tanto na memória usada para armazenar as características, como no tempo necessário para calcular a função discriminante do classificador (a dimensionalidade do espaço F é quadrática em relação ao espaço original X). Esta complexidade quadrática não é praticável para dados com muitas dimensões (por exemplo, dados de expressão genética) [1].

Este problema foi resolvido graças a um truque computacional conhecido como *Kernel Trick*. Uma função kernel K recebe dois pontos x_i e x_j do espaço de *input* e computa o produto escalar desses dados no espaço *feature* (Equação 4).

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{\Phi}(\mathbf{x}_i) \cdot \mathbf{\Phi}(\mathbf{x}_j) \tag{4}$$

Como o mapeamento resulta do cálculo de produtos escalares entre os dados no espaço *feature*, o uso de uma função kernel evita o passo de mapear explicitamente os dados para um espaço de maior dimensão, reduzindo os custos de computação [5]. De forma a garantir que o problema de programação quadrático subjacente é solucionável, a função kernel utilizada tem de satisfazer as condições de Mercer: dar origem a matrizes positivas semi-definidas K, em que cada elemento $K_{i,j}$ é definido por $K_{i,j} = K(x_i, x_j)$, para $i,j = 1,...,n$ [7].

De seguida, apresentam-se algumas das funções kernel mais utilizadas com SVMs, categorizadas por aplicação mais habitual [7] [8]:

- Processamento de imagem: Kernel Polinomial;
- Categorização de texto: *Linear splines kernel in one-dimension;*
- Propósitos gerais, quando não existe qualquer informação prévia sobre os dados: Kernel Gaussiano, RBF (*Radial Basis Function*);
- *Proxy* para redes neuronais: Kernel Sigmoidal;
- Problemas de regressão: *ANOVA radial basis kernel*.

Aquelas mais comuns encontram-se resumidas na Tabela 1, que indica os parâmetros que devem ser determinados pelo utilizador.

Tabela 1 - Funções Kernel mais comuns[1] [12]

Tipo de Kernel	**Função** $K(\mathbf{x}_i, \mathbf{x}_j)$	**Parâmetros**
Polinomial	$(\delta(\mathbf{x}_i \cdot \mathbf{x}_j) + \kappa)^d$	δ, κ e d
Gaussiano	$\exp\left(-\sigma \|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$	σ
Sigmoidal	$\tanh(\delta(\mathbf{x}_i \cdot \mathbf{x}_j) + \kappa)$	δ e κ

Atualmente, a SVM para classificação mais utilizada é a versão margem suave com kernels, que combina todas as anteriores.

As SVMs são classificadores de duas classes, mas é possível resolver problemas multiclasses reformulando o algoritmo de treino das SVMs. Estas técnicas são computacionalmente dispendiosas, logo a alternativa mais usada é converter um classificador de duas classes para uma multiclasse. O método mais comum é denominado *one-vs-the-rest* onde, para cada classe, é treinado um classificador dessa classe contra o resto das classes [1]

Support Vector Regression (SVR)

O método do Vetor de Suporte pode ser aplicado para problemas de regressão (quando o resultado desejado é um valor contínuo, i.e., uma função que descreve um comportamento) mantendo as mesmas características do algoritmo de margem máxima: uma função não-linear é aprendida por uma máquina de aprendizagem linear num espaço de características (induzido por uma função kernel), enquanto a capacidade do sistema é controlada por um parâmetro que não depende da dimensionalidade do espaço. Como no caso da classificação, o algoritmo de aprendizagem minimiza uma função convexa e a sua solução é escassa [3] [11].

Em SVR, uma função de perda é definida e usada para penalizar erros que estão para além de uma banda ε, denominada função de perda ε-insensível, ignorando os erros que estão dentro dessa margem, i.e., a uma certa distância do valor verdadeiro.

[1] O Kernel Sigmoidal satisfaz as condições de Mercer apenas para alguns valores de δ e κ. Os Kernels Polinomiais com d = 1 são denominados lineares.

As variáveis ξ medem o custo dos erros no conjunto de treino (são iguais a zero para todos os pontos dentro da banda ε) [3] (ver Fig. 4).

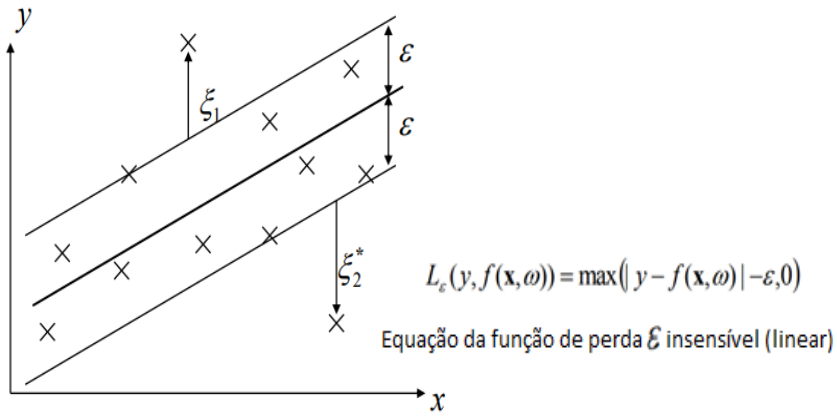

Fig. 4. Representação gráfica da banda ε e dos respetivos erros ξ_i [11].

Existem várias funções de perda que podem ser utilizadas de acordo com a população de dados a ser estudada, sendo a mais comum a função já referida. A única imposição na sua escolha é que a função resultante seja convexa, para assegurar um mínimo global no problema de otimização [11].

A função de perda ε-insensível assegura a existência de um mínimo global e, ao mesmo tempo, a otimização de um limite de generalização confiável [3].

Os problemas de otimização das SVRs encontram-se resumidos na Tabela 2, conforme o caso de aplicação [13].

Tabela 2 – Problemas de otimização das SVRs

Caso	Linear (*soft-margin*)	Não-linear (Kernel)
Minimizar	$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right)$	$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}(\xi_i + \xi_i^*)$
Restrições	$y_i - wx_i - b \le \varepsilon + \xi_i$ $wx_i + b - y_i \le \varepsilon + \xi_i^*$ $\xi_i, \xi_i^* \ge 0$	$y_i - (\mathbf{w}\cdot\varphi(\mathbf{x}_i)) - b \le \varepsilon + \xi_i$ $(\mathbf{w}\cdot\varphi(\mathbf{x}_i)) + b - y_i \le \varepsilon + \xi_i^*$ $\xi_i, \xi_i^* \ge 0, i = 1,\ldots,m$

O parâmetro C determina o *trade off* entre a complexidade do modelo e o grau em que os desvios maiores que ε são tolerados no problema de otimização, e.g. se C está próximo de infinito, o objetivo é minimizar o risco empírico apenas. O parâmetro ε controla a largura da zona ε-insensível usado para ajustar os dados de treino. Quanto maior o valor de ε, menos vetores de suporte são selecionados para construir a função de

regressão. Assim, ambos os parâmetros influenciam a complexidade do modelo, mas de formas diferentes [3].

2.1 Resolução do Problema

O problema de otimização obtido durante o treino das SVMs é quadrático, e pode ser solucionado usando técnicas comuns de programação quadrática (QP). Como a função objetivo que é minimizada é convexa, o problema possui apenas um mínimo global. Os problemas deste tipo podem ser resolvidos com a introdução de uma função Lagrangiana e respetivos parâmetros (multiplicadores de Lagrange α_i). Esta formulação do problema é conhecida como dual (problema com *N* variáveis, em que *N* é o número de amostras), enquanto a original é referida como formulação primal (problema com *n* variáveis, em que *n* é o número de características no conjunto de dados) [7].

A formulação dual apresenta algumas vantagens: não é necessário aceder aos dados originais, porque permite a representação do problema em termos de produtos escalares (útil, principalmente, nos casos de SVMs não-lineares), e o número de parâmetros livres está limitado pelo número de vetores de suporte e não pelo número de variáveis (favorável para problemas de alta dimensão). As formulações duais dos problemas, conforme o caso de SVM, encontram-se na Tabela 3 [7] [14].

Tabela 3 – Formulação dual dos problemas de otimização segundo o tipo de SVM.

Caso	Linear (*soft-margin*)	Não-linear (Kernel)
Maximizar	$\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \, (\mathbf{x}_i \cdot \mathbf{x}_j)$	$\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$
Restrições	$0 \leqslant \alpha_i \leqslant C, \quad \forall i = 1, \ldots, n$ $\sum_{i=1}^{n} \alpha_i y_i = 0$	$0 \leq \alpha_i \leq C, i = 1, \ldots, l$ $\sum_{i=1}^{l} \alpha_i y_i = 0$

No entanto, pode ser computacionalmente dispendioso usar algoritmos de programação quadrática. Existem métodos que podem acelerar a computação e poupar memória. Os três métodos mais conhecidos incluem: ***Chunking***, Decomposição e **Otimização Sequencial Mínima** (*Sequential Minimal Optimization* – SMO). Simplificando, estes métodos dividem o conjunto de dados em subconjuntos mais pequenos e a rotina de programação quadrática corre nesses subconjuntos. O método mais popular é o SMO que soluciona subconjuntos de apenas dois pontos em cada iteração (o que permite uma solução analítica, eliminando a necessidade da programação quadrática). O SMO original foi desenvolvido para problemas de classificação, porém já foi estendido para regressão e estimativa de densidades [12].

2.2 Implementação de SVMs – Vantagens e Desvantagens

Para treinar uma SVM é necessário tomar uma série de decisões: como fazer o pré-processamento dos dados, que kernel escolher e, então, definir os parâmetros da SVM e do kernel. Como qualquer sistema de aprendizagem é dependente dos dados, é aconselhável experimentar várias funções kernel. No entanto, o procedimento mais típico é tentar um kernel linear primeiro e então ver se é possível melhorar o desempenho usando um kernel não-linear. Em muitas aplicações bioinformáticas (número pequeno de amostras, com uma dimensão alta), o kernel linear proporciona os melhores resultados. Além disso, uma SVM com um kernel linear é mais fácil de ajustar, uma vez que o único parâmetro que afeta o desempenho é a constante C (*soft-margin*) [1].

A capacidade de generalização da SVM (a correta previsão da classe de novos dados) depende de uma correta definição dos parâmetros de kernel, C e ε (no caso da SVR). O *software* atual de implementação de treino de SVMs requer a definição destes parâmetros que deve refletir a distribuição dos valores de *input* dos dados de treino, assim como o conhecimento sobre o domínio da aplicação [15].

Vantagens das SVMs [12] [14]:

- Robustas para um número muito grande de variáveis e amostras pequenas;
- Podem aprender modelos de classificação simples e extremamente complexos;
- Utilizam princípios matemáticos sofisticados para evitar o *overfitting*;
- Apresentam resultados empíricos superiores;
- Implementam tanto a minimização do risco empírico, como a minimização do risco estrutural, equivalente à minimização do erro de generalização;
- Produzem representações duais "escassas" da hipótese, resultado em algoritmos extremamente eficientes;
- Têm um número reduzido de parâmetros diferentes de zero e não possuem mínimos locais (funções convexas).

Desvantagens:

- Os modelos são opacos (*black box*), i.e., difíceis de explicar e interpretar. No entanto, existem métodos que mitigam este problema, e.g. a análise de sensibilidade, que mede as alterações da resposta quando uma dada variável de entrada é modificada, permitindo compreender como cada variável afeta as propriedades estudadas [16];
- A dificuldade na escolha dos parâmetros de kernel pode levar ao teste de muitos valores possíveis, o que pode resultar num tempo de computação mais longo [8].

2.3 Ferramentas de desenvolvimento de SVM

Um dos solucionadores de SVM mais utilizados é o LIBSVM. Estima-se que a complexidade de treinar SMVs não-lineares com solucionadores como o LIBSVM seja quadrática em relação ao número de exemplos de treino, o que pode ser proibitivo para conjuntos de dados muito grandes [1].

Para SVMs lineares estão disponíveis solucionadores muito eficientes (tempo linear ao número de exemplos). Existem dois tipos de *software* que proporcionam algoritmos de treino de SVM. O primeiro tipo é *software* especializado cujo principal objetivo é facultar um solucionador de SVM, e.g. LIBSVM, SVM *light* [1], BSVM (para grandes problemas de regressão e classificação de multiclasses) [2], TinySVM (classificação e regressão, usa CVS e suporta Ruby, Pearl, Java e Python) [17], mySVM (também para classificação e regressão) [9] e DTREG, um aplicação que lê ficheiros do tipo .csv e pode criar vários tipos de modelos de análise, incluído SVM (pago) [41].

A outra classe inclui bibliotecas de aprendizagem de máquina que oferecem uma variedade de métodos de classificação e outras facilidades, como métodos para a seleção de características, pré-processamento, etc. Existe um grande número de escolhas e, entre elas, incluem-se os seguintes ambientes que oferecem um classificador SVM: Orange, LibLinear, The Spider, Weka, Lush, Shogun, RapidMiner e PyML [1].

As SVMs são, geralmente, implementadas em C, R (com *caret*, pacote kernlab), Matlab e Python [4]. Um repositório de *software open source* de aprendizagem de máquina está disponível no *website* http://mloss.org [1].

2.4 Aplicações reais de SVM

Nos últimos anos, a popularidade das SVMs em tarefas de aprendizagem tem vindo a crescer devido, principalmente, ao desempenho e qualidade de resultados que apresentam em aplicações na vida real. As SVMs têm sido utilizadas, especialmente, para classificação de padrões, *data mining* e aplicações baseadas em regressão [12].

Algumas das aplicações mais comuns são [8]:

- **Deteção facial -** as SVMs classificam partes da imagem como face e não-face, criando um limite à volta da face;
- **Categorização de texto e hipertexto -** usam os dados de treino para classificar os documentos em categorias diferentes;
- **Classificação de imagens –** oferecem melhor precisão em comparação com as técnicas de busca tradicionais baseadas em *queries*;
- **Bioinformática –** inclui classificação de proteínas e cancro. Usadas para identificar a classificação de genes, pacientes (com base nos genes) e outros problemas biológicos. Usadas, também, num dos problemas centrais em bioinformática: a deteção de homologia remota (classificação de proteínas em classes funcionais e estruturais, de acordo com as suas sequências de aminoácidos);
- **Reconhecimento de escrita manual –** reconhecimento de caracteres escritos à mão;

- **Controlo preditivo generalizado (GPC) –** este modelo de predição baseado em SVM tem sido usado para controlar dinâmicas caóticas com parâmetros úteis – desempenho excelente no controlo de sistemas;
- **Reconhecimento ótico de caracteres (OCR) –** utilização de SVR [11];
- **Aplicações financeiras – (***financial time series forecasting***)** previsões no mercado de ações, análise de correlação em séries de dados, etc. usando SVR [11];
- **Deteção de novidades –** SVM de uma classe que separa todos os pontos de dados da origem (no espaço *feature*) e maximiza a distância entre este hiperplano e a origem: há uma pequena região correspondente aos dados de treino (positivos) e tudo o resto irá situar-se no espaço negativo [5];

É possível consultar (e contribuir) uma lista detalhada de projetos realizados com o uso de SVMs no *website* http://www.clopinet.com/SVM.applications.html.

3 Case-Based Reasoning (CBR)

Raciocínio baseado em casos é uma metodologia de resolução de problemas que, em muitos aspectos, é fundamentalmente diferente de outras abordagens de Inteligência Artificial. O CBR é capaz de utilizar o conhecimento específico de problemas concretos, previamente experienciados. Além disso, o CBR é, também, uma abordagem à aprendizagem sustentável e incremental, uma vez que uma nova experiência é retida (aprendida) cada vez que um problema é solucionado, tornando-a imediatamente disponível para a resolução de problemas futuros [18].

Assim, e resumidamente, CBR é raciocínio por lembrança: os problemas solucionados anteriormente (casos passados) são usados para sugerir soluções para problemas novos, mas similares. Existem quatro suposições sobre o mundo que nos rodeia que sustentam a abordagem do CBR [34]:

1. **Regularidade:** as mesmas ações executadas sob as mesmas condições tendem a ter os mesmos ou semelhantes resultados;
2. **Tipicidade:** experiências tendem a repetir-se;
3. **Consistência:** pequenas alterações na situação requerem apenas ligeiras modificações na interpretação e solução;
4. **Adaptabilidade:** quando há repetição, as diferenças tendem a ser pequenas e facilmente compensáveis.

O modelo geral de CBR pode ser descrito como um ciclo com os seguintes quatro processos (ver Fig. 5) [18]:

1. **Recuperação** (*retrieve*) do(s) caso(s) mais semelhante(s);
2. **Adaptação** (*reuse*) da informação e conhecimento desse caso para resolver o problema;
3. **Revisão** (*revise*) da solução proposta;
4. **Aprendizagem** (*retain*) das partes da experiência que têm maior probabilidade de ser úteis para resolução de problemas futuros.

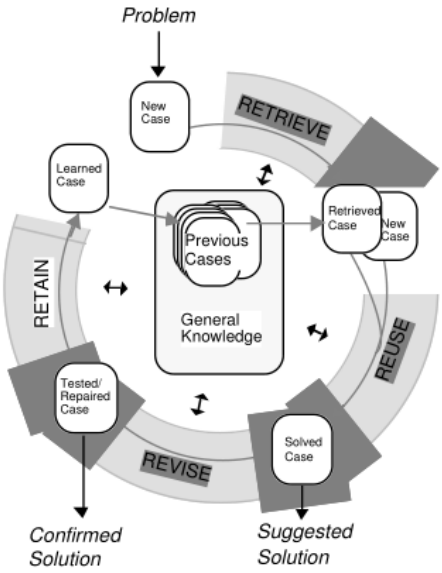

Fig. 5. Representação do ciclo de CBR [18].

A aprendizagem em CBR ocorre como um produto natural da resolução do problema. Quando o problema é solucionado, a experiência é retida para resolver problemas similares no futuro, quando uma solução não é encontrada, a razão para essa falha é identifica e recordada para evitar a ocorrência do mesmo erro [18].

Há um outro modelo que expande este ciclo para seis passos, sendo que os novos passos fazem parte de uma fase de manutenção [20]:

5. **Avaliação** (*review*) do estado atual dos repositórios de conhecimento e da sua qualidade;
6. **Restauro** (*restore*) do nível de qualidade se o passo anterior identificou uma qualidade do caso não desejável (são sugeridas medidas para aumentar o nível acima de um valor crítico e, se aprovadas, são implementadas).

Resumindo, e na fase de aplicação, um problema novo é resolvido através da recuperação de um ou mais casos prévios, a adaptação desse caso de uma forma ou outra, revendo a solução com base nessa adaptação e, finalmente, a nova experiência é incorporada na base de conhecimentos existente (aprendizagem) [18].

3.1 Tipos de conhecimento em CBR

Os sistemas CBR usam vários tipos de conhecimento sobre o domínio do problema para o qual foram desenhados. É possível identificar quatro repositórios de conhecimento, sendo que os três primeiros representam, habitualmente, o conhecimento geral sobre o domínio do problema. A seguir, apresenta-se um resumo dos mesmos [34]:

1. **Vocabulário:** conhecimento necessário para escolher as características utilizadas para descrever os casos; devem ser úteis na recuperação de outros casos e suficientemente discriminativas para prevenir falsas soluções e/ou desempenho reduzido;
2. **Medidas de similaridade:** inclui o conhecimento sobre a própria medida de similaridade e o conhecimento utilizado para escolher a organização mais eficiente da base de casos utilizada, assim como o método de recuperação mais apropriado;
3. **Conhecimento de adaptação:** conhecimento necessário para implementar as fases de adaptação e avaliação do ciclo de CBR. Geralmente, a fase da adaptação requer conhecimento sobre como as diferenças nos problemas afetam as soluções. Como para muitos domínios do problema este é um dos conhecimentos mais difíceis de adquirir, frequentemente, a adaptação é da responsabilidade do utilizador do sistema;
4. **Casos:** incluem o conhecimento sobre as instâncias de problemas resolvidos e, em muitos sistemas CBR, representam a aprendizagem do sistema durante a sua utilização. O que os casos irão conter é, principalmente, determinado pelo vocabulário escolhido.

Representação de casos. Um caso é uma peça de conhecimento contextualizada que representa uma experiência. Contém a lição passada (o conteúdo do caso) e o contexto no qual a lição pode ser usada. Em geral, um caso inclui [34]:

- A **descrição** do problema, que retrata o estado do mundo quando o caso ocorreu;
- A **solução** obtida para esse problema e/ou;
- O **resultado**, que descreve o estado do mundo após a ocorrência do caso.

Casos que incluem a descrição e solução do problema podem ser usados para obter soluções para novos problemas, enquanto que casos com a descrição e resultado podem ser aplicados para avaliar situações novas. Quando os casos contêm os três tipos de informação, podem ser usados para avaliar o resultado das soluções propostas e prevenir problemas potenciais. No entanto, adicionar toda a informação disponível ao sistema torna-o mais complexo e, logo, mais difícil de usar [34].

3.2 Vantagens e limitações de CBR

CBR é um método de resolução de problemas preguiçoso e partilha muitas características com outros métodos do género. Entre as vantagens dos métodos de CBR incluem-se [34]:

- Facilidade na extração do conhecimento;
- Resolução de problemas não tendenciosa;
- Aprendizagem incremental;
- Aptidão para espaços de solução complexos e não completamente formalizados;
- Aptidão para resolução de problemas sequencial;
- Facilidade de explicação;
- Facilidade de manutenção.

As principais desvantagens de solucionadores de problemas preguiçosos são os requisitos de memória e o tempo de execução devido ao processamento necessário para responder a *queries*. As limitações de CBR podem ser resumidas da seguinte forma [34]:

- Requisitos de memória/armazenamento e tempo de execução altos quando as bases de casos são grandes;
- Domínios do problema dinâmicos (os sistemas podem ser incapazes de seguir uma alteração na forma como os problemas são resolvidos, porque aplicam o que resultou anteriormente);
- Ruído nos casos pode levar ao armazenamento do mesmo problema por causa das diferenças introduzidas pelo ruído, o que leva ao armazenamento e recuperação ineficientes dos casos;
- Operação completamente automática implica que os sistemas de CBR esperem pelo *input* do utilizador quando não encontram uma solução.

3.3 Ferramentas de desenvolvimento de CBR

Os sistemas CBR apareceram em ferramentas comerciais no início dos anos 90 e, desde então, têm vindo a distribuir-se por vários domínios [22].

Uma ferramenta de CBR deve suportar os quatro processos principais de CBR: recuperação, adaptação, revisão e aprendizagem. No entanto, é comum encontrar ferramentas de CBR que conseguem extrair informação relevante, mas que deixam a responsabilidade de fornecer uma interpretação e produzir uma decisão final ao utilizador: os passos de adaptação e revisão não são implementados. De facto, a recuperação apenas pode suportar significativamente o processo de decisão humana [32].

Uma boa ferramenta deve ser capaz de lidar com grandes bibliotecas de casos, com o tempo de recuperação aumentando linearmente (no pior dos casos) com o número de casos [22]. Seguem-se alguns exemplos de ferramentas que existem no mercado atual [35]:

- **Induce-it** (*Inductive Solutions Inc*.): cria sistemas CBR a partir de bases de dados de folhas de cálculo do Microsoft Excel [35];
- **jCaBaRe** (gratuita): API de CBR para Java. O utilizador precisa de passar o objeto Java que representa o novo caso e a coleção de casos passados e a API irá encontrar os objetos mais similares (através de *reflection*) [30];
- **FreeCBR:** implementação em Java de um motor CBR (*open source*). O pacote inclui aplicações GUI, linha de comandos e web, um componente MS ActiveX nativo, um Java *bean* e uma API de desenvolvimento [27];
- **COLIBRI** (gratuita): plataforma para desenvolvimento de *software* CBR académico. Oferece uma arquitetura bem definida para o *design* de sistemas CBR, uma implementação de referência dessa arquitetura (jCOLIBRI *framework*) e várias ferramentas de desenvolvimento que assistem o utilizador na implementação e partilha de novos sistemas e componentes de CBR [28];

- **MyCBR** (gratuita): kit de desenvolvimento de *software* (SDK) de recuperação baseada em similaridade (*open source*). Possível testar e modelar medidas de similaridade extremamente sofisticadas e intensivas em relação ao conhecimento numa GUI poderosa (myCBR *Workbench*) e integrá-las facilmente em aplicações próprias usando o SDK myCBR [33].

3.4 Aplicações de CBR

As aplicações de CBR abrangem muitas áreas diferentes: medicina, indústria, mercado financeiro, educação, direito, inteligência artificial, entre outras. Segue-se uma descrição mais detalhada e concreta dos vários usos de CBR:

- **Suporte de decisão:** abrange várias áreas, sendo mais comum na área financeira, industrial e médica. Em processos de decisão, e face a problemas complexos, procuram-se problemas análogos para encontrar soluções possíveis. Os sistemas CBR foram desenvolvidos para suportar este processo de recuperação para encontrar problemas semelhantes relevantes. Os CBRs são particularmente bons na busca de documentos estruturados, modulares e não homogéneos [22]. Apresenta-se um exemplo comercial de seguida:
  - **Spotlight** (*ATP – CaseBank*): suporte de decisão, diagnóstico e *troubleshooting* de equipamento complexo, particularmente na área da aviação [23].
- **Aplicações médicas:** muito populares em sistemas de diagnóstico médico – recuperam casos passados cujos sintomas são semelhantes ao novo caso e sugerem diagnósticos baseados nos casos recuperados com uma maior correspondência. Existem, também, sistemas CBR que definem dietas para pacientes, conforme as suas necessidades específicas [22] [26]. Já foram desenvolvidos sistemas CBR que podem melhorar aplicações futuras de prescrição eletrónica de medicamentos [25];
- **Apoio ao cliente:** gestão de atendimento (*help desks*) - lidam com problemas apresentados pelo cliente em relação a um produto ou serviço [22]. Alguns exemplos concretos incluem:
  - **WinSTIPS** (*ServiceSoftware*): apoio ao cliente (*help desks*), diagnóstico de problemas e sugestão de soluções (*troubleshooting)* na área de eletrónicos de consumo [36];
  - **Deskbot-3** (*TreeTools*): apoio ao cliente (*service* e *help desk*) e atendimento virtual (*chatbot*) [24];
- **Direito:** resolução de disputas *online* utiliza sistemas CBR que, através de experiências passadas, conseguem encontrar uma solução mais rapidamente para o problema. Alguns exemplos incluem:

    - **HYPO**: programa desenvolvido para raciocínio jurídico que utiliza casos atuais e hipotéticos no domínio da jurisprudência. Existe, também, o TAX-HYPO que atua no domínio da lei de impostos [22] [26];

- **Mercado financeiro e de *marketing*:** usados para avaliação – determinam valores para variáveis através da comparação com valores conhecidos de variáveis semelhantes [22];
- ***Design*:** assistem uma parte do processo de *design* industrial e na arquitetura, ao recuperarem casos passados, mas precisam de ser combinados com outras formas de raciocínio para suportarem o processo de *design* completo [22];
- **Educação:** particularmente, na área de Aprendizagem Assistida. Sistemas de CBR usados na monitorização do progresso e desempenho do estudante, registando as suas necessidades de aprendizagem e mesmo respondendo a questões [26]. Um exemplo, na área de Direito:

    - **CATO:** ambiente de aprendizagem inteligente desenhado para ajudar os estudantes de direito a aprender aptidões básica de argumentação. CATO usa um modelo abstrato de argumentos legais baseados em casos para gerar exemplos de argumentação no contexto do trabalho corrente do aluno [19].

- **Problemas de classificação:** plataforma **COBRA** usada, atualmente, no diagnóstico de falhas em sensores de gás industriais [32];
- **Avaliação de risco ambiental:** utilizados modelos de *case-based ranking* [32];
- **Culinária:** aplicação *web* **CookIIS** recomenda e adapta receitas ou cria um menu completo tendo em consideração as preferências do utilizador, como ingredientes explicitamente excluídos ou dietas previamente definidas [32];
- **Música:** sistema **SaxEx** capax de sintetizar baladas de jazz (com solos de saxofone tenor de alta qualidade) baseadas em casos que representam atuações de solos humanos [31];
- **Segurança de informação:** sistema CBR desenvolvido para organizações militares e que permitem o planeamento e análise de cenários para processos de treino e auditoria [21];
- **Inteligência artificial:** CBR proposto para "traduzir" os termos abstratos dos regulamentos humanos para que os agentes de *software* usados em negócios consigam operar sob regulamentos precisos e inequívocos [20];

4 Árvores de Decisão

As Árvores de Decisão (*Decision Trees*) consistem em ramificações em árvore com nós intermédios que representam testes sobre valores de atributos, ramos com os possíveis valores para os atributos e folhas com as decisões sobre a previsão, ou seja, o valor da variável dependente, que é aquilo que se quer descobrir. Pode-se dizer que, em geral, representam disjunções de conjunções.

Por exemplo, na árvore utilizada para se perceber se se deve ou não jogar ténis consoante alguns parâmetros meteorológicos como *Wind e Humidity*, há um primeiro teste ao valor do atributo *Outlook*, estando os possíveis valores desse atributo (*suny*, *overcast e rain*) nos ramos que saem desse nó, que neste caso é também a raiz da árvore (ver Fig. 6).

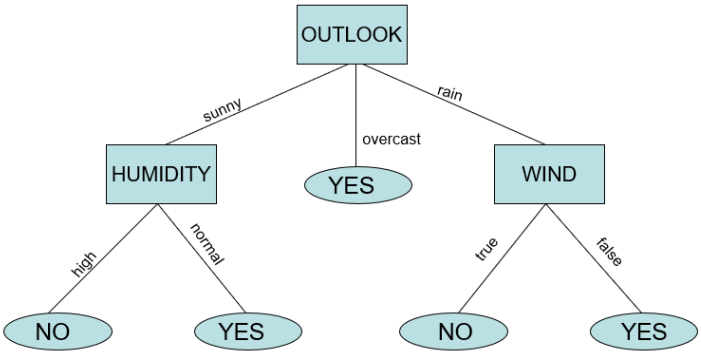

Fig. 6. Árvore de decisão para jogar ou não ténis.

Consoante os ramos, poderemos ir para outro nó com um teste sobre outro atributo ou ir para uma folha que indica que se vai jogar caso o valor de Outlook seja *overcast*. Supondo uma situação em que estaria a chover (*Outlook* seria *rain)*, iríamos verificar se estaria vento e, caso estivesse, não iríamos jogar (*Wind* seria *true*) e, caso não estivesse, iríamos jogar (*Wind* seria *false*).

Para melhor perceber o tipo de problemas em que são usadas, é importante fazer uma abordagem aos problemas de classificação, pois correspondem à principal utilização das Árvores de Decisão.

Problemas de classificação

Um problema de classificação consiste num problema em que queremos tomar decisões sobre uma ou mais classes, tendo em conta os valores das várias variáveis de *input* designadas por atributos. As classes funcionam, por isso, como o output destes problemas, visto que o objetivo é fazer a previsão dos seus valores para várias situações.

Alguns exemplos deste tipo de problemas são:

1. Decidir a partir de observações atmosféricas (temperatura, vento, etc.) se uma tempestade é pouco provável, provável ou muito provável.
2. Diagnosticar se um paciente tem gripe tendo em conta os sintomas apresentados (dor de cabeça, febre, etc.).

No exemplo 1, diríamos que os atributos são as várias observações atmosféricas e a classe é a probabilidade de ocorrência de uma tempestade, enquanto que no exemplo 2, os atributos seriam os sintomas apresentados e a classe seria o diagnóstico da doença (se tem ou não tem gripe) [44].

Tipos de Árvores de Decisão. Existem dois tipos de árvores de decisão:

- **Árvores de Classificação –** utilizadas em problemas de classificação;
- **Árvores de Regressão –** utlizadas em problemas de regressão, ou seja, quando a variável dependente corresponde a valores contínuos.

Nas árvores de classificação, o valor obtido nas folhas corresponde à classe em que se insere um determinado caso, enquanto que nas árvores de regressão corresponde a um valor médio ou a um modelo linear de regressão.

Em ambos os tipos de árvores, aplicam-se as mesmas estratégias para a sua construção a partir de um conjunto de dados de treino, sendo isso abordado mais à frente na secção "Capacidade de Aprendizagem" [35] [37].

4.1 Vantagens e Desvantagens das Árvores de Decisão

As árvores de decisão têm várias vantagens e desvantagens em relação a outros sistemas de aprendizagem. Algumas dessas vantagens são [37] [39] [42]:

- Facilidade de compreensão e interpretação, principalmente quando representadas graficamente;
- Capacidade de lidar com variáveis numéricas e categóricas;
- Pouca necessidade de preparação dos dados, enquanto que outras técnicas precisam muitas vezes de normalização de dados;
- Boa performance para grandes conjuntos de dados, não necessitando de recursos computacionais muito elevados;
- Bom para simulação do modo de tomada de decisão humano.

Exemplos de desvantagens são [37] [39] [42]:

- Criação de árvores demasiado complexas, em que as folhas têm poucos exemplos ou pouco representativos, o que reduz a capacidade preditiva (*overfitting*);
- Pouca robustez em certas árvores, o que significa que uma pequena alteração nos dados de treino pode representar uma grande alteração na árvore e na sua capacidade preditiva;
- Menor precisão que outros sistemas de aprendizagem.

4.2 Capacidade de Aprendizagem

As árvores de decisão constituem um sistema de aprendizagem supervisionado, ou seja, em que se sabe qual a variável dependente para a qual queremos fazer previsões. Para que se possam obter essas árvores, é necessário que haja um conjunto de dados de treino que possa ser utilizado para ser feita a sua construção (ver Fig. 7).

Num	Outlook	Temp	Humidity	Wind	PLAY
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rain	mild	high	FALSE	yes
5	rain	cool	normal	FALSE	yes
6	rain	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rain	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rain	mild	high	TRUE	no

Fig. 7. Conjunto de dados de treino usado para a árvore da Fig. 6.

Começamos com uma região em que se inserem todos os casos do conjunto de dados no topo da árvore, sendo essa região dividida em sub-regiões que não se sobrepõem em relação ao valor do atributo a partir do qual foi feita essa divisão. Cada sub-região vai-se dividindo de igual forma em relação aos restantes atributos, fazendo a árvore crescer até que um determinado critério de paragem previamente definido seja atingido, como por exemplo, o número de casos por nó ser inferior a 20.

Trata-se, por isso, de um processo de divisão recursiva com uma abordagem *top-down*, visto que começamos no topo da árvore com o conjunto de dados completo e vamos dividindo conforme vamos descendo e tornando mais específicas as previsões que serão feitas. A escolha do atributo pelo qual se vai fazendo a divisão dos casos ao longo da árvore é feita normalmente por um algoritmo do tipo *greedy*, que apenas se foca em escolher a melhor variável na sub-região em que se encontra, não se preocupando com as divisões futuras, o que poderia, por vezes, levar à obtenção de uma melhor árvore.

Um dos problemas na construção de uma árvore de decisão é escolher o melhor atributo para a raiz, o que é conseguido através de medidas de ganho informativo como a entropia, em que o melhor atributo é aquele que tem a menor entropia. Outro problema que pode ocorrer na construção das árvores é o *overfitting*, em que há excesso de ramos com folhas pouco representativas e, por isso, quando surgem novos casos nunca antes vistos, há maior dificuldade em conseguir obter uma previsão correta. Para evitar isso, podem ser impostas algumas restrições (ver Fig. 8) para evitar que a árvore cresça muito tais como número máximo de níveis ou número mínimo de casos numa folha ou pode-se proceder à utilização de técnicas de *Prunning*.

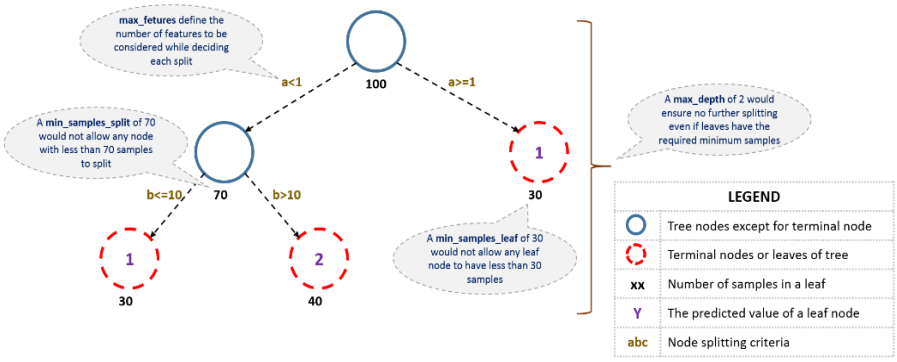

Fig. 8. Exemplos de restrições.

Prunning. Consiste na simplificação do modelo de previsão para combater o *overfitting* e existem duas possíveis estratégias:

- *Pré-pruning*: parar de expandir um ramo quando a informação se torna pouco fiável. Tem o risco de poder parar o processo demasiado cedo;
- *Pós-pruning*: deixar crescer a árvore até ao fim e depois podar sub-árvores pouco fiáveis. É a estratégia mais fácil de implementar e é a que origina melhores resultados.

Na prática, isto consiste em obter árvores mais pequenas e mais generalizadas que sejam melhores para novos casos.

4.3 Ferramentas de Desenvolvimento de Árvores de Decisão

Existem vários algoritmos para a construção de uma árvore de decisão como: ID3, CART ou C4.5 [37]. O pacote mais popular é o C4.5 de Quinlan (em C), que é capaz de lidar com atributes nominais e numéricos, além de vir equipado com várias características como *prunning*, capacidade de lidar com valores em falta e de agrupar valores de atributos. Inclui, também, rotinas para correr experiências de validação cruzada e para converter árvores de decisão aprendidas para regras de produção, que são mais adequadas para o desenvolvimento de sistemas profissionais [38]. Resumidamente, os passos do algoritmo C4.5 consistem (sendo S o conjunto de treino) [40]:

1. Iniciar com uma árvore vazia;
2. Escolher o melhor atributo A para a raiz;
3. Dividir S em subconjuntos segundo os valores de A;
4. Para cada subconjunto de S construir uma sub-árvore de decisão;
5. Parar quando o subconjunto tiver elementos de uma só classe.

Outro pacote possível é o MLC++ que implementa em C++ uma variedade de algoritmos de aprendizagem, incluindo para árvores de decisão. O seu principal objetivo é

assistir no desenvolvimento e comparação de algoritmos de aprendizagem [38]. Existem, também, vários *softwares* e linguagens de programação que permitem a implementação de algoritmos de criação de árvores de decisão, tais como:

- **Weka**, um *software* gratuito que permite a criação e teste de árvores de decisão através de algoritmos como J48 (corresponde a uma implementação do C4.5) *e SimpleCart* (corresponde a uma implementação do CART), possuindo também opções para vários métodos de *prunning*;
- **R**, uma linguagem de programação e, também, ambiente de desenvolvimento que permite fazer cálculos estatísticos e gráficos, mas também utilizar alguns sistemas de aprendizagem, sendo um deles as árvores de decisão para as quais possui em vários *packages* algumas implementações do algoritmo CART;
- **Python**, uma linguagem de programação que também permite a utilização deste sistema através de uma livraria chamada *scikit-learn*;
- **MATLAB**, um *software* de alta performance essencialmente voltado para o cálculo numérico, mas que também permite o uso destas árvores;
- **SPSS Modeler**, da IBM, é um *software* que disponibiliza quatro algoritmos para efetuar classificação e análise de segmentação. Os resultados e os dados de entrada usados na construção da árvore podem ser contínuos (gera uma árvore de regressão) ou categóricos (árvore de classificação). Os algoritmos são [40]:
    - Nó de **Classificação e Regressão (C&R)** gera uma árvore de decisão que permite prever ou classificar observações futuras (apenas dois subgrupos);
    - Nó **CHAID** gera árvores usando estatísticas chi ao quadrado para identificar divisões ótimas (pode gerar árvores não binárias, com mais de 2 ramos);
    - Nó **QUEST** oferece um método de classificação binária desenhado para reduzir o tempo de processamento necessário na análise de árvores C&R grandes. Os dados de entrada podem ser contínuos, mas o resultado é categórico;
    - Nó **C5.0** constrói uma árvore de decisão ou um conjunto de regras. Os resultados têm de ser categóricos;
- **RapidMiner,** uma plataforma para equipas das ciências dos dados que unifica preparação de dados, aprendizagem de máquina e lançamento de modelos preditivos, também permite a construção de árvores de decisão [45];
- **DTREG,** aplicação que lê ficheiros do tipo .csv para criar árvores de decisão, entre outros tipos de modelos. Possível criar árvores de classificação (resultados categóricos) e árvores de regressão (resultados contínuos) [41].

4.4 Aplicações reais de árvores de decisão

Atualmente, as árvores de decisão são usadas em múltiplas e variadas áreas. Uma das principais é a área das Finanças e de Marketing (consumismo). Seguem-se alguns exemplos de aplicações de árvores de decisão [46]:

- **Agricultura:** usada uma árvore de decisão para a classificação da ocorrência de ferrugem asiática em culturas comerciais de soja, com base em variáveis meteorológicas [42];
- **Astronomia:** usadas para filtrar o ruído das imagens capturadas pelo telescópio Hubble; árvores de decisão também ajudaram na classificação de galáxias, na sua contagem e na descoberta de quasars;
- **Engenharia Biomédica:** identificação de características que deviam ser usadas nos dispositivos de implante;
- **Indústria e Produção:** árvores de decisão usadas para testar a qualidade de soldaduras (de forma não destrutiva), para aumentar a produtividade, em processos de otimização de máquinas eletroquímicas, para planear linhas de montagem de placas de circuitos, para descobrir falhas no processo de produção de um Boeing e para controlo de qualidade;
- **Medicina:** Indução automática de árvores de decisão em diagnóstico, cardiologia, psiquiatria, deteção de microcalcificações em mamografias, análise do síndrome de Morte Súbita de Crianças, etc.;
- **Reconhecimento de objetos:** classificação baseada em árvores usada para reconhecer objetos tridimensionais;
- **Processamento de texto:** ID3 usado para a classificação de textos médicos;
- **Finanças e Negócios:** nos negócios, as árvores de decisão oferecem um modelo geral de determinação de soluções e de apoio à decisão; usadas, também, na atribuição de crédito (*credit scoring*), determinação e previsão do preço de opções, etc.;
- **Marketing:** determinação das árvores de decisão que influenciam as compras e escolhas do consumidor numa loja ou *online;* previsão do impacto de campanhas de *marketing*.

5 Conclusão

Os desenvolvimentos na área da inteligência artificial têm levado ao surgimento de vários sistemas de aprendizagem ao longo dos anos que oferecem grande utilidade em várias áreas, quer seja pela sua capacidade de previsão, quer pela capacidade de aprender a executar determinadas tarefas.

Em suma, este trabalho ilustra a utilidade dos sistemas de aprendizagem e aprofunda os nossos conhecimentos na área da Aprendizagem e Extração de Conhecimento e, mais concretamente, em relação aos vários sistemas de aprendizagem, servindo como um bom complemento aos assuntos abordados nas aulas deste módulo.

Referências

Support Vector Machine

1. Ben-hur, A. & Weston, J.: A User's Guide to Support Vector Machines. In: Department of Computer Science, Colorado State University (2007).
2. BSVM, http://www.csie.ntu.edu.tw/~cjlin/bsvm/, último acesso 23/10/2017.
3. Cristianini, N. & Shawe-Taylor, J.: An Introduction to Support Vector Machines and other kernel-based learning methods. Cambridge University Press (2000).
4. Fernández-Delgado M., Cernadas E., Barro S. & Amorim D.: Do we need hundreds of classifiers to solve real world classification problems? J. Mach. Learn. Res. 15, 3133–3181 (2014).
5. Introduction to One-class Support Vector Machines, http://rvlasveld.github.io/blog/2013/07/12/introduction-to-one-class-support-vector-machines/, último acesso 20/10/2017.
6. Jakkula, V.: Tutorial on Support Vector Machine (SVM). School of EECS, Washington State University, Pullman 99164. Disponível *online:* http://www.ccs.neu.edu/course/cs5100f11/resources/jakkula.pdf (Acesso em 20 de Outubro de 2017).
7. Lorena, A. C. & Carvalho, A. C. P. L. F.: Uma introdução às support vector machines. Revista de Informática teórica e aplicada, v. XIV, n. 2, p. 43-67 (2007).
8. Machine Learning, http://data-flair.training/blogs/applications-of-svm/, último acesso 20/10/2017.
9. MySVM - a support vector machine, http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/index.html, último acesso 20/10/2017.
10. New features in Support Vector Machine (SVM) tool, https://community.alteryx.com/t5/Engine-Works-Blog/New-features-in-Support-Vector-Machine-SVM-tool/bap/6768, último acesso 20/10/2017.
11. Paisitkriangkrai, P.: Linear Regression and Support Vector Regression. University Lecture (2012). Disponível online: http://cs.adelaide.edu.au/~chhshen/teaching/ML_SVR.pdf (Acesso em 20 de Outubro de 2017).
12. Santos, E. M.: Teoria e aplicação de Support Vector Machine à aprendizagem e reconhecimento de objetos baseados na aparência. Dissertação (Programa de Pós-Graduação em Informática) – Universidade Federal da Paraíba, PA (2002).
13. Smola, A. & Scholkopf, B.: A tutorial on support vector regression. NeuroCOLT Technical Report NC-TR-98-030, Royal Holloway College, University of London, UK (1998). Disponível online: http://www.svms.org/regression/SmSc98.pdf (Acesso em 20 de Outubro de 2017).
14. Statnikov, A., Aliferis, C. F., Hardin, D. P. & Guyon, I.: A gentle introduction to support vector machines in biomedicine. World Scientific (2011). Disponível online: https://www.eecis.udel.edu/~shatkay/Course/papers/UOSVMAllifeRisWithoutTears.pdf (Acesso em 20 de Outubro de 2017).
15. SVM Tutorial, https://www.svm-tutorial.com/svm-tutorial/, último acesso 20/10/2017.
16. Tinoco, J., Gomes Correia, A. & Cortez, P.: Support Vector Machines in Mechanical Properties Prediction of Jet Grouting Columns. Semana da Escola de Engenharia Outubro 24-27 (2011).

17. TinySVM: Support Vector Machines, http://chasen.org/~taku/software/TinySVM/#download, ultimo acesso 23/10/2017.

Case-Based Reasoning

18. Aamodt, A. & Plaza, E.: Case-based reasoning: foundational issues, methodological variations, and system approaches. AI Communications 7 (1) 39–59 (1994).
19. Aleven, V. & Ashley, K.: Teaching Case-Based Argumentation through a Model and Examples Empirical Evaluation of an Intelligent Learning Environment. University of Pittsburgh, Pittsburgh, PA (1999). Disponível *online*: https://www.researchgate.net/publication/2575859_Teaching_Case-Based_Argumentation_through_a_Model_and_Examples_Empirical_Evaluation_of_an_Intelligent_Learning_Environment (Acesso em 23 de Outubro de 2017).
20. Balke, T., Novais, P., Andrade, F. & Eymann, T.: From Real-World Regulations to Concrete Norms for Software Agents – A Case-Based Reasoning Approach, ICAIL (2009).
21. Borges, J., Martins, J., Andrade, J. & Santos, H.: Design of a case-based reasoner for information security in military organizations. In: Academic Conferences and Publishing International (2015)
22. Case Based Reasoning, http://www.aiai.ed.ac.uk/links/cbr.html, último acesso 20/10/2017.
23. CaseBank Homepage, http://www.casebank.com/, último acesso 23/10/2017.
24. Deskbot-3, http://www.servicebot.com.br/index.php/robo-para-servicos, último acesso 23/10/2017.
25. Duarte, J., Neves, J., Cabral, A., Gomes, M., Marques, V., Santos, M. F., Abelha, A. & Machado, J.: Towards intelligent drug electronic prescription. In: European Simulation and Modelling Conference, Guimarães, Portugal (2011).
26. Fernandes, B., Freitas, M., Analide, C., Vicente, H. & Neves, J., Handling Default Data Under a Case-Based Reasoning Approach. In S. Loiseau, J. Filipe, B. Duval & J. van den Herik Eds., Proceedings of the 7th International Conference on Agents and Artificial Intelligence (ICAART 2015), Vol. II, pp. 294–304, Scitepress – Science and Technology Publications, Lisbon (2015).
27. Free CBR Homepage, http://freecbr.sourceforge.net/, último acesso 23/10/2017.
28. GAIA – Group of Artificial Intelligence Applications, http://gaia.fdi.ucm.es/research/colibri, último acesso 23/10/2017.
29. Induce-It, http://inductive.com/softcase.htm, último acesso 23/10/2017.
30. jCaBaRe Homepage, http://jcabare.sourceforge.net/, último acesso 23/10/2017.
31. Mantaras, R. L., Perner, P. & Cunningham, P.: Emergent Case-Based Reasoning Applications. Cambridge Univ Press (2005)
32. Montani, S. & Jain, L.C. (Eds.): Successful Case-based Reasoning Applications—1. Springer, New York, NY (2010).
33. MyCBR Homepage, http://www.mycbr-project.net/, último acesso 23/10/2017.
34. Pantic, M.: Introduction to Machine Learning & Case-Based Reasoning. Machine Learning (course 395). Computing Department, Imperial College London (2015). Disponível online: https://ibug.doc.ic.ac.uk/media/uploads/documents/courses/syllabus-CBR.pdf (Acesso em 23 de Outubro de 2017).
35. Software showcase, https://ai-cbr.cs.auckland.ac.nz/tools.html, último acesso 20/10/2017.

36. WinSTIPS Service Tips, https://www.servicesoftware.com/winstips.aspx, último acesso 23/10/2017.

Árvores de Decisão

37. A Complete Tutorial on Tree Based Modeling from Scratch (in R & Python), https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python/#fourteen, último acesso 23/10/2017.
38. Almuallim, H., Kaneda, S. & Akiba, Y.: Development and Applications of Decision Trees. Expert Systems, Vol. 1 (2003)
39. Decision tree learning, https://en.wikipedia.org/wiki/Decision_tree_learning, último acesso 23/10/2017.
40. Decision Tree Models, https://www.ibm.com/support/knowledgecenter/en/SS3RA7_15.0.0/com.ibm.spss.modeler.help/nodes_treebuilding.htm, último acesso 23/10/2017.
41. DTREG – Predictive Modelling Software, https://www.dtreg.com/, último acesso 23/10/2017.
42. Megeto, G. A. S., Oliveira, S. R. De M., Ponte, E. M. De & Meira, C. A. A.: Árvore de decisão para classificação de ocorrências de ferrugem asiática em lavouras comerciais com base em variáveis meteorológicas. Engenharia Agrícola , Jaboticabl, v.34, n.3, p. 590-599, mai./jun. (2014).
43. O que é um diagrama de árvore de decisão?, https://www.lucidchart.com/pages/pt/tudo-sobre-%C3%A1rvores-de-decis%C3%A3o, último acesso 23/10/2017.
44. Quinlan, J. R.: Induction of Decision Trees. Machine Learning, 1(1):81-106 (1986).
45. RapidMiner, https://rapidminer.com/products/, último acesso 23/10/2017.
46. Selected real-world applications, http://www.cbcb.umd.edu/~salzberg/docs/murthy_thesis/survey/node32.html, último acesso 23/10/2017.
47. Trigueiros, D.: As Árvores de Decisão. Dissertação "Sistemas de Apoio à Decisão" (Mestrado em Ciências Empresariais) – Instituto Universitário de Lisboa, Lisboa (1991). Disponível online: http://home.iscte-iul.pt/~dmt/publ/tx/Arvores_de_Decisao_INDEG_ISCTE.pdf (Acesso em 23 de Outubro de 2017).
48. Von Zuben, F. J. & Attux, R. R. F.: Árvores de Decisão. Departamento de Engenharia de Computação e Automação Industrial, Faculdade de Engenharia Elétrica e de Computação, Universidade de Campinas, Brasil. Disponível online: ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ia004_1s10/notas_de_aula/topico7_IA004_1s10.pdf (Acesso em 23 de Outubro de 2017).