Aprendizagem e Extração de Conhecimento: Uma Visão Geral Sobre Case-Based Reasoning, Artificial Neural Networks e Support Vector Machines

Rui Miranda (a75488)[1], Thanira Rates (pg35406)[2], and Thiago Sposito (pg36099)[3]

Universidade do Minho - Departamento de Informática

Resumo *Machine Learning* nunca esteve tão em voga como nos dias de hoje, muito por causa dos recentes avanços em *Artificial Neural Networks*. Porém, não nos podemos esquecer que esse método não resolve todos os problemas e avanços ainda são alcançados em áreas mais tradicionais de *Machine Learning*, como *Case-Based Reasoning* e *Support Vector Machines*. Neste trabalho abordamos as principais características destes paradigmas supracitados e suas capacidades, explicitando as principais ferramentas de desenvolvimento e *frameworks*, abordando algumas soluções existentes no mercado atual, além de discutir quais métodos mais apropriados para diferentes tipos de problemas.

Keywords: case based reasoning, neural networks, support vector machines, deep learning, artificial intelligence, machine learning

1 *Case-based Reasoning - CBR*

1.1 Introdução

Case-Based Reasoning (Raciocínio Baseado em Casos) é uma abordagem padrão para resolução de novos problemas, em que se considera soluções já utilizadas em problemas anteriores similares, criando uma nova solução para adaptação da mesma em futuros casos. Essa característica de ligação à aprendizagem está a ser muito discutida nos últimos anos, desde a sua primeira abordagem nos Estados Unidos até ao recente crescimento do interesse por IA na Europa e no resto do Mundo.

O CBR é um paradigma de resolução de problemas que, em muitos aspetos, é fundamentalmente diferente de outras abordagens de IA. Em vez de depender apenas do conhecimento geral de um problema de domínio, ou fazer associações com relações generalizadas entre problemas descritores e conclusões, o *CBR* é capaz de utilizar o conhecimento específico de duas situações concretas e experientes de problemas (casos). Um novo problema é resolvido ao encontrar um caso anterior semelhante, e reutilizá-lo na situação de um novo problema. Uma segunda diferença importante é que o *CBR* também é uma abordagem para aprendizagem incremental e sustentada, uma vez que a nova experiência é adicionada à base de conhecimento sempre que o problema for resolvido, tornando-o imediatamente disponível para problemas futuros.[1]

No texto acima, retirado do *paper 'Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches'*, confirma que a definição de CBR não deixa de ser um raciocínio por recordação e afirma que problemas similares terão opções de soluções também similares por lembrar situações similares anteriores, reutilizar informações e conhecimentos e pode ser utilizado para classificação e regressão. Diferente das árvores de decisão e das redes neuronais, o problema de aprendizagem requer um trabalho feito na mesma altura em que a consulta é realizada.

Case-Based Reasoning favorece aprender com a experiência, já que geralmente é mais fácil de aprender mantendo uma experiência concreta de solução de problemas do que generalizar a partir dele. Ainda assim, a aprendizagem efetiva com *CBR* requer um conjunto de métodos bem elaborado para extrair conhecimento relevante da experiência, integrar um caso numa estrutura de conhecimento existente e indexar o caso para mais tarde combinar com casos semelhantes.[1]

Partindo de uma análise de um problema, *Case-Based Reasoning* encontra na memória de todos os casos um caso semelhante e usa o mesmo para a sugestão de uma solução para o problema. Aprendendo com essa mesma experiência, a solução é avaliada e assim, a memória de casos é atualizada.

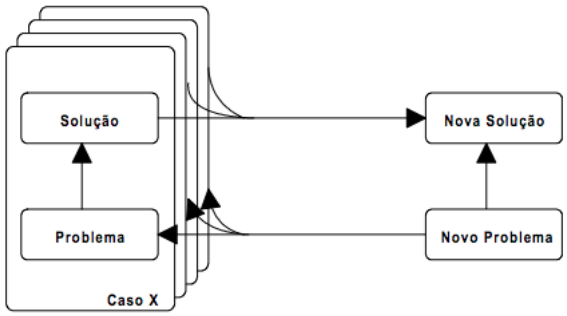

Figura 1. Modelo Ilustrativo de CBR[2]

A qualidade da solução no *CBR* depende de vários fatores, como as experiências que teve, a sua habilidade de entender novas situações em termos das experiências anteriores, a sua habilidade em adaptação e sua habilidade em avaliação. Porém os casos com menos experiências não quer dizer que seja ruim para esse método, pois o mesmo irá incluir tanto tentativas bem sucedidas quanto as que falham para atingir os objetivos: as sucedidas para propor soluções para novos problemas e as que falham serão usadas para o aviso de uma falha potencial.

1.2 Ciclo de Funcionamento de *Case-Based Reasoning*

De acordo com alguns autores como *Kolodner* (1993)[3], a etapa de aquisição do conhecimento é a mais importante, além de ser um componente crítico no desenvolvimento de um sistema. O raciocínio por reutilização de casos passados é uma ferramenta suscetível na resolução de problemas, e a sua característica mais significativa é a sua *aprendizagem*, que se ocorre naturalmente pelas resoluções dos casos.

Grande parte da sua aprendizagem vêm de duas maneiras, com acumulação de novos casos e a atribuição de índices: pode-se tornar um raciocínio mais eficiente lembrando soluções anteriores e adaptando a elas. Por exemplo, se um caso fosse adaptado de uma nova maneira e resolvendo o problema usando um método inovador, ou ainda combinando soluções de vários casos. No futuro novos casos não vão precisar de repetir todos os passos para resolver o mesmo problema, pois usará o mesmo raciocínio.

Um caso baseado em dados torna-se mais eficiente com o passar do tempo e experiência, obtendo melhores resultados, evitando assim erros cometidos no passado, pois captura problemas indexando-os por recursos que preveem seus antigos erros. Usa-se metodologias interpretativas para criticar e justificar uma solução para um problema, e para tal precisa-se de resolver os problemas. Um sistema de CBR pode ser dividido em quatro elementos e ser entendido através do seguinte ciclo de 4Rs:

1. **Retrieve** *(Recuperação)*: Recupera casos similares da base de casos;
2. **Reuse** *(Adaptação)*: Adapta os casos recuperados para ser utilizado no novo caso;
3. **Revise** *(Revisão)* : Avalia e revisa solução baseado no seu desempenho em resolução;
4. **Retain** *(Aprendizagem)* : Decisão se o novo caso é ou não integrado na base do caso.

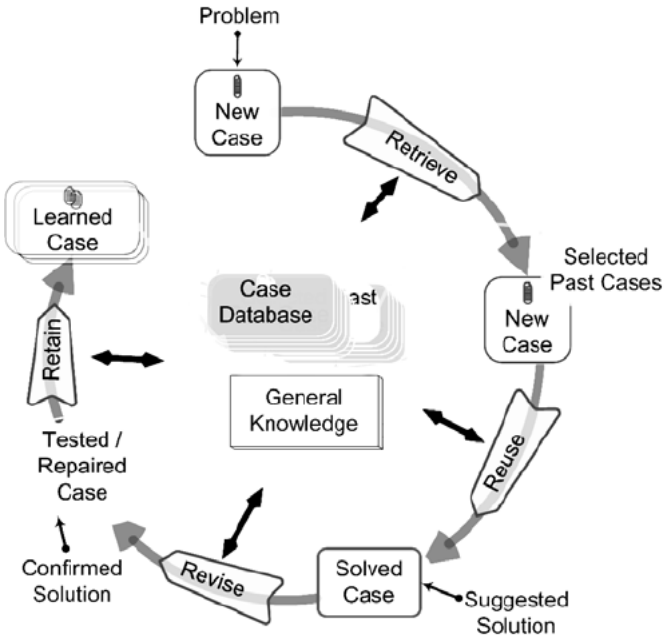

Figura 2. Ciclo de funcionamento um sistema CBR[4]

Na figura 2 pode-se observar que no início do ciclo de um novo caso, *Retrieve*, se recupera um ou mais casos similares para poder utilizar ou não suas experiências na resolução de um novo caso. Além disso, precisa-se interpretar a nova situação baseado nas experiências dos casos anteriores. Uma proposta resolução para o problema é extraindo a solução de alguns casos recuperados (às vezes essas soluções são incluídas de fora) seguido da adaptação, *Reuse*, o processo da formação de uma solução anterior para a nova situação, e a crítica, *Revise*, processo que avalia a nova solução ante de colocar em prática. Isso é seguido pela justificação, o processo de criação de um argumento para a proposta solução, feito por um outro processo de comparação da nova situação com casos anteriores, e novamente critica, o processo de justificação de argumento, que é feito gerando situações hipotéticas e tentando argumentar sobre essas. Na última

etapa do ciclo, *Retain*, o novo caso é integrado adequadamente na memória do caso para ser utilizado no futuro. O caso é composto pelo problema, pela sua solução, além de qualquer outro fator incluído, como o sistema sabendo utilizar o raciocínio e por final, seu resultado.

O novo caso pode ser recuperado durante um raciocínio posterior em momentos que pode ser mais útil. Não deve ter excesso de indexação, uma vez que não queremos que seja lembrado indiscriminadamente. Isto significa que a razão pode antecipar a importância do caso para posteriormente raciocinar. Estrutura de indexação da memória e sua organização também são ajustados nesta etapa. Este caso compartilha todos os seus problemas com o primeiro: devemos escolher índices apropriado para o novo caso usando o vocabulário certo, e devemos ao mesmo tempo certificar de que todos os outros itens permanecem acessíveis à medida que adicionamos na biblioteca de casos.[3]

1.3 Ferramentas

O *Case-Based Reasoning* é apropriado para ser usado quando existem grandes volumes de dados históricos; experiência vale tanto quanto o conhecimento; problemas não são completamente formalizáveis; existem conhecimentos para adaptação de casos; existem muitas exceções às regras e quando se é preciso aderir conhecimento online.

As ferramentas ajudam a acelerar o projeto e a avaliação de uma aplicação, pois o seu propósito geral é construir um sistema de *CBR* para ser usado em várias aplicações, com interface gráfica fácil para o utilizador com visualização útil. Alguns exemplos de *CBR Shells* mais utilizadas em aplicações comercializadas:

- **MyCBR**, é uma ferramenta de recuperação baseada em similaridade e um kit de desenvolvimento de software baseado em similaridade (SDK). Com o *Workbench* do *MyCBR*, o utilizador pode modelar e testar medidas de similaridade altamente sofisticadas e intensivas em conhecimento numa GUI poderosa e integrá-las facilmente nos aplicativos próprios usando o *myCBR SDK*. Os sistemas de recomendação de produtos baseados em casos são apenas um exemplo de aplicativos de recuperação baseados em similaridades.Pode ser visto como sucessor do *CBR Works* e contém muitos recursos úteis.[5]
- **CBRWorks** foi desenvolvido para aplicações de comércio eletrónico, mas pode ser usado para outros fins. Contém elementos de todos os recipientes de conhecimento e pode executar o CBR completo.[6]
- **jColibri** é uma estrutura geral que suporta muitos recursos, como interfaces gráficas, lógicas de descrição e ontologias, CBR textual, avaliação, E.T.C..[7]

1.4 Soluções existentes no mercado

O *Case-Based Reasoning* é uma efetiva ferramenta na resolução de casos. Como Kolodner(1993) citou[3], no mundo real o *CBR* é útil para pessoas e máquinas que sabem muito sobre uma tarefa e domínio porque lhes dá uma

maneira de reutilizar um raciocínio rígido que eles já fizeram no passado. É igualmente útil, no entanto, para aqueles que sabem pouco sobre uma tarefa ou domínio, quando o conhecimento está incompleto e/ou a evidência é escassa.Com isso, várias empresas comerciais oferecem *Shells* para a criação de sistemas CBR, como pode-ser ver a seguir:

Como exemplo de funcionalidade, o *shell ReMind* oferece um ambiente interativo para aquisição de casos, vocabulário de domínio, índices e protótipos. O usuário pode definir relações hierárquicas entre os atributos e uma medida de similaridade com base neles. A indexação é feita indutivamente ao construir uma árvore de decisão e permitir ao usuário editar graficamente a importância dos atributos. São suportados vários métodos de recuperação: (1) recuperação indutiva que combina o protótipo mais específico em um protótipo hierarquia, (2) recuperação de vizinho mais próximo e (3) recuperação do modelo semelhante ao SQL. A adaptação de casos é com base em fórmulas que ajustam valores com base em diferenças de casos recuperadas e novas. *ReMind* também tem a capacidade de representar relações causais usando um modelo qualitativo. Os primeiros produtos comerciais apareceram em 1991, incluindo sistemas de *Help-Desk*, *diagnóstico técnico*, *classificação* , *previsão*, *controle* , *monitoramento*, *planejamento* e *aplicações de design*. *ReMind* é uma marca registrada da *Cognitive Systems Inc.* e foi desenvolvido com o suporte da DARPA.[1]

ReCall é uma marca registrada do sistema CBR da *ISoft*, uma empresa de IA baseada em Paris, e as aplicações incluem (de acordo com a ISoft) sistemas de *help desk*, *diagnóstico de falhas*, *análise de empréstimos bancários*, *controle* e *monitoramento*. Os métodos de recuperação são uma combinação de métodos (1) e (2) no *ReMind*, mas oferece um padrão mecanismos de adaptação, como voto e analogia, e uma biblioteca de métodos de adaptação.[1]

Quando observamos as pessoas resolverem problemas no dia-a-dia, o método de CBR é comum ser usado, por exemplo, por *advogados*, quando usam casos precedentes para construção e justificação de novos argumentos em novos casos na corte, os *médicos*, que usam casos de pacientes que já tiverem os mesmos sintomas que um paciente novo para o diagnóstico; os *mecânicos*, que se encontra com um problema incomum e acessa problemas mecânicos anteriores para diagnóstico e conserto do novo problema; e também muito comum sistemas de *help-desk*, onde a principal preocupação é melhorar o atendimento de apoio ao cliente, de modo seguro, rápido e eficaz, onde normalmente utiliza-se o algoritmo para similaridade de casos.

Finalizando esta parte do documento, as tendências de desenvolvimento dos métodos de CBR podem ser agrupadas em torno de *quatro tópicos principais*, de acordo com Aamodt(1993)[1] : **integração com outros métodos de aprendizagem, integração com outros componentes de raciocínio, incorporação em processamento paralelo e avanços de métodos, enforcando novos aspetos cognitivos.**

2 Artificial Neural Networks

2.1 Introdução

Artifcial Neural Networks (ANNs) são sistemas computacionais com capacidade de aprendizagem inspirados pelos sistemas nervosos biológicos, sendo especialmente úteis em problemas de difícil resolução formal como os relacionados ao reconhecimento de padrões. [10]

Uma ANN pode ser entendida com uma rede de nós (análogos aos neurónios) dotados de um determinado estado interno. Estes nós são conectados por arestas com pesos variáveis (análogas aos axónios) uns aos outros. Cada neurónio é então capaz de *disparar* mensagens (sinais) para seus semelhantes conectados.

Diferentemente dos sistemas biológicos onde o sinal é binário (ou sinal é disparado ou não), as ANNs, por motivos de otimização e ganho de performance, no treinamento são capazes de passar sinais com diferentes intensidades.[8]

Redes neuronais geralmente são construídas numa arquitetura de camadas, onde temos uma camada de entrada, uma camada de saída, e um número arbitrário de camadas internas designadas por *hidden layers*. Estas camadas internas podem ser entendidas como representações de diferentes regiões do córtex, e são esperadas de processar progressivamente os sinais. Embora seja possível tentar inferir o que acontece dentro das camadas internas da RNA, o estados interno do sistema é geralmente considerado uma "*black box*".

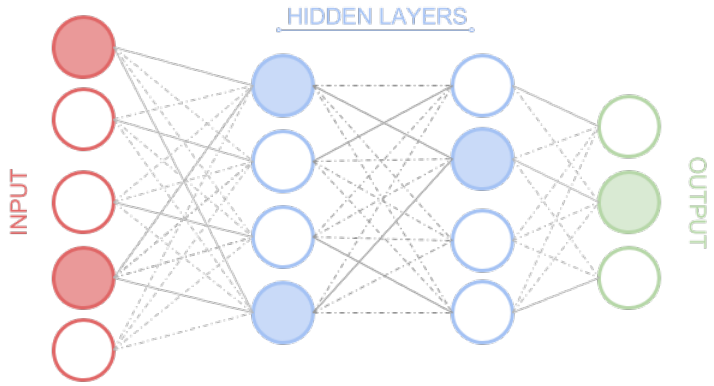

Figura 3. Diagrama simplificado exemplificando a arquitetura em camadas..

2.2 Aprendizagem

Quando pensamos em redes neuronais como nós interligados por *tensores* de peso variável, podemos entender a aprendizagem nesse sistema como o fino balanceamento destes pesos.

O processo que leva a este balanceamento é conhecido como "treinamento", e pode acontecer de várias formas distintas. Este processo se caracteriza por sucessivas iterações e idealmente a rede neuronal deve entender melhor o seu ambiente a cada passo. Estas iterações são compostas por ciclos onde: a rede neuronal é estimulada pelo ambiente, então ela muda o seu estado interno(valor dos peso das conexões entre o neurónios) em resposta a este estímulo. Finalmente ela passa a responder ao ambiente de forma diferente graças às mudanças no seu estado interno.[8]

O treinamento é chamado de **Aprendizagem Supervisionada** quando o *dataset* é supervisionado e classificado *a priori*, como se a ANN tivesse um professor a lhe ensinar como entender o ambiente. Neste caso o resultado da interpretação do ambiente depende da função derivada dos exemplos apresentados.

Para alguns tipos de situações nem sempre é possível contar com um professor. Em situações onde não temos informações sobre o ambiente, a rede neuronal deve ser capaz de deduzir padrões a partir do ambiente. Este paradigma é conhecido como **Aprendizagem não supervisionada**.

Para problemas de tomada de decisão ou problemas a serem resolvidos em tempo real é possível utilizar a **Aprendizagem por Reforço**. Neste paradigma a ANN, após observar uma série temporal de estados é capaz de deduzir uma melhor solução levando em consideração não um resultado imediato, mas um ao final de uma janela de eventos.

O treinamento em si baseia-se em escolher um conjunto de parâmetros, que dado o modelo escolhido minimize a função de custo. Temos então um problema de otimização, que é trivialmente resolvido aplicando o método do gradiente à derivada da função de custo, a fim de se encontrar os mínimos locais desta função.

2.3 Ferramentas

Torch é um *framework* de machine learning, com forte viés científico, com suporte às principais plataformas, entre elas iOS e Android. Tem o seu core escrito em C e C++ , fazendo uso de Tecnologia CUDA da *Nvidia*. O Torch usa LUA como linguagem de *script*. Para trabalhar com redes neuronais é preciso fazer uso de um pacote chamado nn dentro do Torch. Este pacote usa o conceito de módulos que possuem métodos para a passagem de sinais dentro da rede. É utilizado pela equipe de pesquisa do *Facebook*, pelo *Twitter* e pelo motor de busca *Yandex*. Com uma linguagem concisa e vasta documentação é um bom ponto de partida para estudar o assunto[12].

O **TensorFlow** é um *framework* desenvolvido pela *Google Brain Team*, sendo considerado por alguns uma evolução natural do que foi aprendido com o Torch. É um *framework* com uma curva de aprendizagem suave, principalmente por

usar linguagens mais conhecidas. Uma das principais vantagens deste *framework* é a sua ampla gama de plataformas disponíveis (inclusive plataformas móveis). O core do *framework* é escrito em C++, faz extenso uso de NumPy e tem suas funções expostas em *Python* por meio de *wrappers* [9]. Um diferencial interessante do TensorFlow é o TensorBoard que é uma interface gráfica que auxilia na arquitetura das redes neurais.

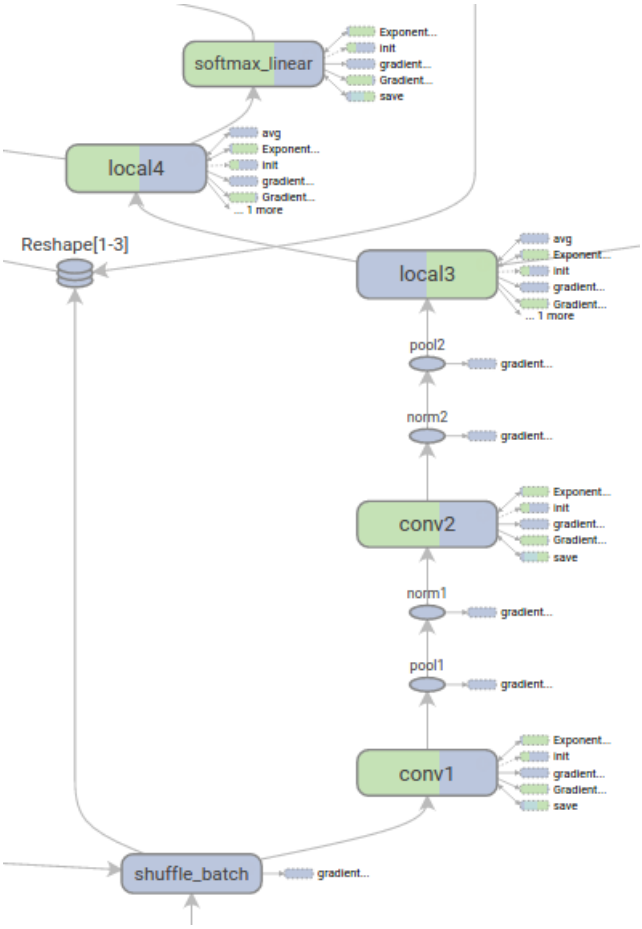

Figura 4. Interface do TensorBoard facilita muito a visualização da arquitetura das ANNs [17]

Theano é um *framework* desenvolvido pela Universidade de Montreal, reconhecida mundialmente pelo estudo e pesquisa em algoritmos aprendizado computacional. É um *framework* voltado para pesquisa, relativamente desprovido de abstrações pode ser bastante complicado de ser utilizado por iniciantes. Por

lidar diretamente com os aspetos estatísticos e matemáticos das rede neuronais é bastante passível de otimização é considerado um *framework* bastante rápido. Definitivamente mais voltado para pesquisa e para ser utilizado como base de outros *framework*s [11].

Caffe foi desenvolvido UC Berkeley, inicialmente como um projeto de PhD por Yangqing Jia. Escrito em C++ usa Python como linguagem de script. Tem como foco reconhecimento e classificação de imagens. A comunidade tem uma biblioteca de modelos conhecida como "Model Zoo"onde iniciantes podem usar modelos já treinados sem ter que escrever quase nenhum código.

2.4 Aplicações

Um dos campos que tradicionalmente têm sido problemáticos com algoritmos de aprendizado tradicionais é a tradução de textos. Bastante progresso foi feito com *Recurrent Neural Networks* o que fez com que serviços como o **Google Translate** tomassem a web de assalto. Mais recentemente mesmo empresas como **Facebook** vem avançando em pesquisas para serviços de tradução dentro de suas plataformas, inovando inclusive com o uso de *Convolutional Neural Networks* para tais tarefas[14].

Outro exemplo de empresas que integra *ANNs* em seus produtos é a **Adobe**, que já integra redes neuronais em muitos aspetos de sua suite como por reconhecimento de tags nos serviços cloud ou as suas funções de ajuste automático de cores dentro de programas como o **Photoshop** e o **After Effects**[16]. A equipa de pesquisadores da Adobe estão envolvidos com grupos de pesquisa nas principais universidades do mundo, e nós provavelmente veremos cada vez mais funcionalidades baseadas em redes neuronais nas futuras versões de seus produtos[15].

3 Support Vector Machines

3.1 Introdução

As *Support Vector Machines* são modelos de aprendizagem supervisionados, associados com algoritmos de aprendizagem que analisam dados para classificação de dados e análise de regressão. Dado um conjunto de exemplos com múltiplos atributos que pertencem em uma de duas categorias, as *SVMs* criam um modelo que classifica novos casos a uma das categorias. Um modelo pode ser modelado em pontos, com cada caso mapeado de forma que exista uma linha que seja mais distante entre os pontos mais próximos das duas categorias.[18]

As *Support Vector Machines* podem tanto fazer classificação linear como classificação não linear, com recurso ao chamado *kernel method*, mapeando os inputs para superfícies de dimensão superior[21].

3.2 Aprendizagem

As *Support Vector Machines* podem ser usadas na criação de fronteiras lineares para a separação dos dados pertencentes a duas categorias, podendo ser lineares ou não. As *SVMs* lineares são as mais simples, lidando com casos linearmente separáveis. A obtenção de fronteiras não lineares pode ser obtida com uma extensão das *SVMs* lineares conhecida como o *kernel method*[19].

Para as *SVMs* é dado um conjunto de treinamento constituído por vários pontos pontos da forma *(x1,y1),...,(xn,yn)*, onde *yi* pode ser 1 ou -1, dependendo do conjunto onde *xi* pertence, e cada *xi* um vetor com p dimensões.

Se os dados de treinamento forem linearmente separáveis, pode-se selecionar um hiperplano que separa as duas classes de dados. As *SVMs* maximizam a "margem", o intervalo que separa os pontos mais próximos do hiperplano. Isto pode ser representado num problema de otimização quadrático, sendo fácil de resolver usando técnicas de otimização existentes[20].

Devido à existência de ruídos, nem todos os casos são linearmente separáveis. Nessas situações temos de expandir as *SVMs*, de forma a lidar com conjuntos de treino mais gerais, introduzindo uma função de folga. Estas *SVMs* são chamadas de margens suaves, permitindo a existência de alguns erros de classificação[19].

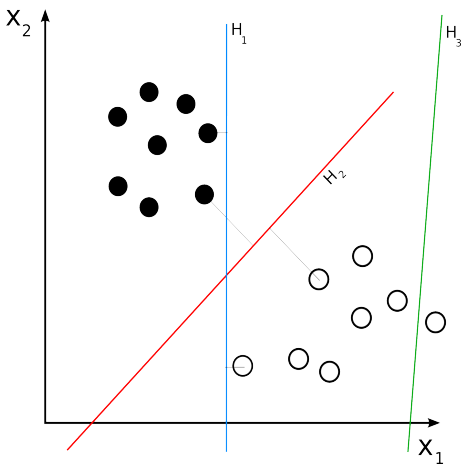

Figura 5. SVM a separar os hiperplanos [27]

Caso um conjunto não seja linearmente separável, onde não seja possível separar os dados de treino com um hiperplano, podem ser utilizadas as *SVMs* não lineares. Elas funcionam usando uma técnica denominada por *kernel method*, mapeando os dados de treino para uma dimensão superior, sendo depois possível separar os dados por uma SVM linear[21].

De forma a reduzir os custos computacionais, e não ter que calcular a transformação para cada vetor, o algoritmo das *SVMs* lineares é alterado, na qual o produto escalar é substituído por uma função (*Kernel Function*). Os *kernel functions* mais utilizados são o Polinomial, Gaussiano e Hiperbólico[19].

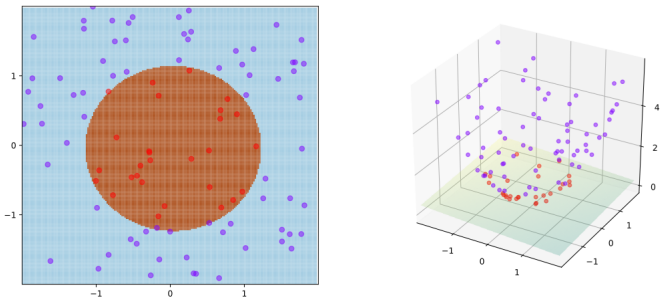

Figura 6. Treino de uma SVM não linear [28]

3.3 Ferramentas

Múltiplas ferramentas de desenvolvimento de *machine learning* disponibilizam *Support Vector Machines*, de notar o *MATLAB*, o *Weka*, *R* e *Python* com a biblioteca *scikit-learn*.

MATLAB é um ambiente de cálculo numérico, permitindo a manipulação de matrizes, implementação de algoritmos, e interação com outras linguagens de programação. O MATLAB implementa *Support Vector Machines* através das funções *svmclassify*[24] e *fitcsvm*.

O *Waikato Environment for Knowledge Analysis (Weka)* é uma suite de *software* de *machine learning* e análise de dados para *Java*

O *R* é uma linguagem de programação *open source* para análise de dados e computação estatística. A biblioteca *kernlab*[23] fornece métodos de aprendizagem para classificação e regressão, implementando funcionalidades de *SVM*.

A biblioteca *scikit-learn* permite a implementação e utilização de *Support Vector Machines* na linguagem de programação *Python*[22].

3.4 Aplicações

As *Support Vector Machines* são utilizadas para vários problemas, como por exemplo no conhecimento de caracteres escritos à mão, classificação e segmentação de imagens, e classificação de linguagem natural. Com *SVMs* não lineares, é possível capturar relações muito mais complexas no *dataset* sem realizar transformações complexas anteriormente. Contudo o tempo de treino é maior por ser mais computacionalmente intensivo.[25]

4 Conclusão

Nos tempos atuais as Redes Neuronais são colocadas como a panaceia do aprendizagem de máquina. Embora elas estejam a resolver vários problemas e demonstrem potencial quase ilimitado, é importante lembrar que no estado atual deste paradigma é muito difícil derivar generalizações e descrever de modo formal os seus estados internos.

Paradigmas mais tradicionais e com mais sólida fundamentação teórica como Case-Based Reasoning e Support Vector Machines ainda oferecem algumas vantagens sobre as ANNs. Support Vector Machines tem a grande vantagem de durante o treinamento garantir a localização do mínimo global, enquanto que as ANNs abrem margem para eventual retenção em um mínimo local [26]. Temos portanto mais confiança que SVMs não sofrerão em erros provocados por falhas no treinamento.

Case-Based Reasoning se mostra mais prático quando é necessário alimentar o sistema com novos dados [1]. Ao invés de ter que incorrer em novo treinamento da rede neural, o que pode ser por vezes bastante dispendioso, é possível simplesmente adicionar um novo caso ao CBS.

Com este trabalho ficou claro que ANNs têm potencial de mudar como lidamos com conceitos de aprendizagem computacional. Problemas antes confinados ao intelecto humano lentamente têm sendo resolvidos por sistemas computacionais, com grandes avanços feitos em campos da visão por computador, processamento de linguagem, robótica entre tantos outros. Mas ficou claro também que as redes neurais estão em sua infância, principalmente quando se trata de aplicações práticas. Ficou claro também que nos dias de hoje métodos de aprendizagem considerados mais tradicionais ainda tem muita relevância para aplicação e pesquisa.

Referências

1. Aamodt, E.P.: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications. IOS Press, Vol. 7: 1, pp. 39-59, (1994).
2. Novais, P., Never, J.: Texto Raciocínio Baseado em Casos, Universidade do Minho, (1998).
3. Kolodner, J.: Case-Based Reasoning. San Mateo,CA - Morgan Kaufmann. p. 8, (1993).
4. Modelo Ilustrativo de CBR `https://www.researchgate.net/figure/235645649_fig1_Figure-1-Case-based-Reasoning-CBR-cycle` 18/10/2017
5. Ciclo de funcionamento um sistema CBR `http://www.mycbr-project.net/` 18/10/2017
6. CBRWorks `http://www.empolis.de/cbr/` 22/10/2017
7. jColibri `http://gaia.fdi.ucm.es/research/colibri/jcolibri` 22/10/2017
8. Haykin, S.,: Neural networks: a comprehensive foundation."Prentice-Hall (1999)
9. Abadi, M., et al.: Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467 (2016)
10. Gurney, K.: An introduction to neural networks. CRC press, (1997)
11. Al-Rfou, R., et al.: "Theano: A Python *framework* for fast computation of mathematical expressions."arXiv preprint (2016)
12. Collobert, R., Bengio S. e Mariéthoz, J.: Torch: a modular machine learning software library. No. EPFL-REPORT-82802. Idiap, (2002)
13. Jia, Y., et al.: Caffe: Convolutional architecture for fast feature embedding. Proceedings of the 22nd ACM international conference on Multimedia. ACM, (2014)
14. Gehring, J., et al.: Convolutional Sequence to Sequence Learning."arXiv preprint arXiv:1705.03122 (2017)
15. Fujun, L., et al.: Deep Photo Style Transfer."arXiv preprint arXiv:1703.07511 (2017)
16. Zhicheng, Y. et al.: Automatic photo adjustment using deep neural networks."ACM Transactions on Graphics (TOG) 35.2 (2016)
17. TensorGraph `https://www.tensorflow.org/get_started/graph_viz`. 23/10/2017
18. Nello Cristianini, John Shawe-Taylor, "An Introduction to Support Vector Machines and other kernel-based learning methods", Cambridge University Press, (2000)
19. Support Vector Machine, `https://en.wikipedia.org/wiki/Support_vector_machine`. 22/10/2017
20. Introduction to Support Vector Machines `https://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html`. 22/10/2017
21. An introduction to Support Vector Machines (SVM) `https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/`. 22/10/2017
22. Machine Learning in Python `http://scikit-learn.org/stable/`. 22/10/2017
23. kernlab: Kernel-Based Machine Learning Lab `https://cran.r-project.org/web/packages/kernlab/index.html`. 22/10/2017
24. Classify using support vector machine (SVM) `https://www.mathworks.com/help/stats/svmclassify.html`. 22/10/2017
25. Why use SVM? `http://www.yaksis.com/posts/why-use-svm.html`. 22/10/2017
26. Burges,C.: A tutorial on support vector machines for pattern recognition."Data mining and knowledge discovery 2.2 (1998)

27. Svm separating hyperplanes https://en.wikipedia.org/wiki/File:Svm_separating_hyperplanes_(SVG).svg. 22/10/2017
28. Kernel trick idea https://en.wikipedia.org/wiki/File:Kernel_trick_idea.svg. 22/10/2017