Sistemas de Aprendizagem - CBR, PSO e SVM

Universidade do Minho, Departamento de Informática

Resumo Este trabalho foi desenvolvido no âmbito da unidade curricular de Aprendizagem e Extração de Conhecimento, pertencente ao perfil de Sistemas Inteligentes. Este mesmo tem como objetivo explorar alguns sistemas de aprendizagem.

1 Introdução

1.1 Contextualização

Este artigo corresponde ao resultado de investigação e consolidação de informação, efetuada para o primeiro trabalho prático da Unidade Curricular de Análise e Extração de Conhecimento, pertencente ao perfil de Sistemas Inteligentes do Mestrado em Engenharia Informática.

1.2 Caso de Estudo

Neste documento são abordados três sistemas de aprendizagem de informação distintos, denominados por Case Based Reasoning, Particle Swarm Optimization e Support Vector Machines.

1.3 Estrutura do Relatório

Para cada sistema de aprendizagem são apresentados aspetos como o seu funcionamento geral e respetivas características, a sua capacidade de aprendizagem, principais ferramentas existentes e soluções que estão presentes nos mercados atuais.

2 Case Based Reasoning

2.1 Descrição e Carateristicas

Case Based Reasoning ou Raciocínio Baseado em Casos pode ser visto como um processo que se serve de soluções já existentes para um dado conjunto de problemas, com a finalidade de produzir uma solução para um novo problema, semelhante aos anteriores. Este modelo pode ser utilizado em contextos distintos para resolver novos desafios, explicar novas situações e avaliar novas soluções para os problemas. É especialmente útil na resolução de problemas complexos, com uma quantidade elevada de soluções alternativas. A metodologia desta técnica de resolução de problemas consiste em encontrar e justificar a solução para um dado problema utilizando situações passadas similares.

2.2 Como exibe Capacidade de Aprendizagem

Assim que ocorre um problema é feita uma análise ao mesmo. Essa análise torna possível procurar em casos anteriores similaridades que irá permitir determinar uma solução. Para determinar essa solução é necessário rever e avaliar a soluções que foram aplicadas em casos similares reutilizando ou adaptando conhecimento de tais situações. Assim que é determinada a solução a base de dados de casos é atualizada e com isso aumenta a experiência do sistema. À medida que o sistema vai encontrando novas soluções para novos problema baseando-se em casos passados torna-se assim mais eficiente.

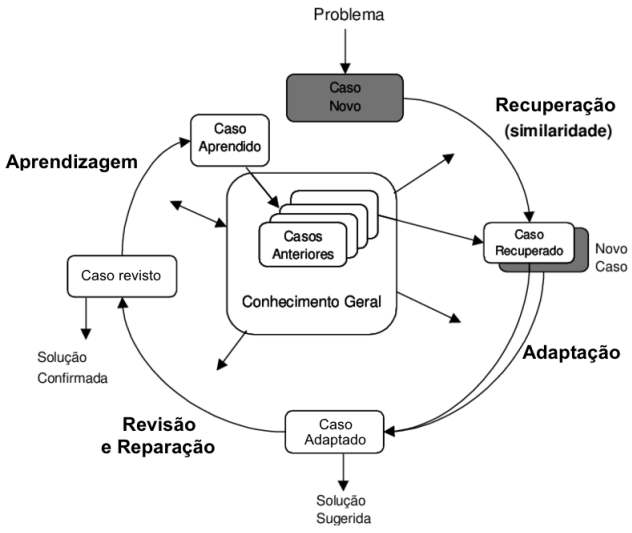

Figura 1. Recuperação (Retrieve), Adaptação (Reuse), Revisão (Revise), Aprendizagem (Retain).

Este processo pode então ser resumido em quatro etapas:

Recuperação A Recuperação seleciona o caso com maior similaridade com novo problema. Essa seleção é feita através da comparação de dois casos para verificar como um está ligado ao outro e como os mesmos podem partilhar soluções e consequências, sendo esta um dos pontos chaves de um CBR, a partir da qual será organizada a base de dados utilizada pelo sistema, e serão encontrados os casos com maior probabilidade de gerarem uma solução para o novo problema.

Adaptação Quando um caso similar é recuperado da base de dados para resolver o novo problema, ocorre uma reutilização de conhecimento obtido de um caso anterior que é similar ao novo problema. Nessa reutilização é necessário fazer uma adaptação para adequar a solução existente, ao novo problema, com o objetivo de facilitar a obtenção de uma solução do novo problema. Esta adaptação influenciará diretamente a flexibilidade do sistema de CBR, que irá determinar a capacidade do sistema em resoluções de novos casos.

Revisão Assim que se obtém a proposta de solução, é necessário verificar se realmente esta é a solução para o nosso problema, como tal é feita esta verificação para testar se o problema foi resolvido. Caso o problema não tenha sido resolvido com a solução encontrada, temos de recorrer as etapas anteriores para encontrar uma solução que se adeque ao novo problema.

Aprendizagem A aprendizagem consiste na representação do conhecimento, e na forma que se irá registar os problemas e as soluções vividas pelo sistema. Tem como objetivo adquirir o conhecimento resultante do novo caso, para que este seja adicionado ao repositório de dados, para que possa ser mais tarde utilizado para a resolução de novos que possam surgir. Nos casos em que o problema é solucionado, é armazenada a informação desse caso, que pode vir a contribuir para solucionar futuros problemas, mas caso o sistema não seja bem sucedido na resolução do problema, a informação que é armazenada vai no sentido de identificar as falhas, para que esses erros não voltem a ocorrer. Com este tipo de aprendizagem destes processos, consegue-se um sistema capaz de adaptar uma solução, ainda que esta tenha surgido de um caso com diferente contexto.

Na figura 1 é apresentado um diagrama de estados, que ilustra o funcionamento de um CBR.

2.3 Ferramentas de desenvolvimento

Existem várias ferramentas que permitem implementar um sistema CBR. De todas elas decidi-mos explorar duas delas, myCBR e jColibri.

myCBR - é composto por um conjunto de ferramentas de desenvolvimento de software, baseadas na similaridade de problemas/casos de estudo. Com o myCBR Workbench é possível modelar e testar problemas altamente sofisticados de semelhança de conhecimento, e facilmente integra-los noutras aplicações usando o myCBR SDK.

jCOLIBRI - é uma framework construída em Java para modelar sistemas de CBR. Este promove a reutilização de software, integrando na aplicação técnicas de engenharia de software, com uma descrição do nível de conhecimentos, que separa o método de resolução de problemas e o modelo de domínio.

Ainda que sem a mesma relevância existem no mercado outras ferramentas que permitem implementar um CBR:

- OpenCBR
- FreeCBR
- CBR Works
- Kate-Tools
- CBR-Express
- ReMind

2.4 Soluções existentes no mercado

No início dos anos 90 apareceram as primeiras ferramentas comerciais que aplicavam CBR, as empresas ao verificar que este método trazia benefícios, foram apostando no alargamento da gama de domínios, dos quais se destacam:

Diagnóstico - sistemas que selecionam casos passados, cuja lista de sintomas seja similar ao novo caso, e que sugerem um diagnóstico baseado nos casos mais similares. A maioria dos sistemas são deste tipo.

Centrais de Ajuda - estes sistemas são usados na área de Serviço ao cliente, e tem como função lidar com falhas nos produtos ou nos serviços.

Avaliação - usados para determinar valores de variáveis, comparando-as com os valores de variáveis similares. São comuns na área das finanças ou do marketing.

Suporte à Decisão - na tomada de decisão, quando uma pessoa se depara com um problema complexo, tem tendência em verificar soluções já existentes, na esperança de encontrar possíveis soluções. Os sistemas foram desenvolvidos para dar suporte neste tipo de problema.

Design - estes sistemas assistem o cliente apenas numa parte do processo de design, e precisam de ser combinados com outras formas de "reasoning" para suportar o processo de design completo.

Clavier (Suporte à decisão) Uma autoclave é uma câmara de pressão utilizada em processos industriais, que requerem temperaturas elevadas e pressão diferente da pressão ambiente. Esta ferramenta auxilia os profissionais da área, na determinação inteligente dos materiais a colocar na máquina, com a finalidade de balancear da melhor forma a relação custo/qualidade do processo de produção. O seu maior propósito é encontrar os grupos ou configurações mais apropriadas das várias peças, de modo a maximizar o rendimento da autoclave, e garantido que todas as partes são propriamente processadas. CBR é aplicado neste sistema para comparar as partes que precisam de ser processadas, com uma base de casos especificando as distribuições que obtiveram sucesso no passado, sugerindo assim a carga mais apropriada. Com isto são eliminadas produções de peças com baixa qualidade, que de outra forma seriam desperdiçadas, salvaguardando largos montantes no orçamento da empresa.

Cassiopee (Dianóstico de Máquinas) A inatividade de um avião, anda na casa dos 50% do tempo de vida, e o maior objetivo das companhias aéreas era reduzir para metade o tempo de diagnostico. Dessa necessidade surgiu este sistema que utiliza CBR, para selecionar casos semelhantes para que as soluções para os problemas possam ser encontradas mais rapidamente, e com maior facilidade, sem ser necessário seguir os passos de um manual. Atualmente esta ferramenta está a ser testada por várias companhias aéreas de todo o mundo, que podem compartilhar casos e adiciona-los à base de casos.

3 Particle Swarm Optimization

3.1 Descrição e Carateristicas

Particle Swarm Optimization (PSO), ou em português, Inteligência de Grupo, é um algoritmo de otimização inteligente, que é subconjunto de um ramo de computação conhecido como Computação Evolutiva. Tem por base um modelo matemático simples inspirado em fenómenos naturais, desenvolvido por Kennedy e Eberhart em 1995, com o objetivo de explorar as características do comportamento social de grupos de indivíduos, como pássaros e peixes.

É um método computacional que otimiza um problema tentando, iterativamente melhorar uma solução candidata em relação a uma dada medida de qualidade. Resolve um problema através de uma população de soluções candidatas, apelidadas de partículas, e movendo essas partículas no espaço de procura de acordo com uma fórmula matemática, que relaciona a posição e velocidade das partículas. O movimento de cada partícula é influenciado pela sua melhor posição local, mas também é guiada para as melhores posições conhecidas no espaço de procura, que são atualizadas à medida que melhores posições são encontradas por outras partículas. Com isto é esperado que o enxame se mova para melhores soluções.

Para além do algoritmo PSO original foram desenvolvidas outras variantes, como é o caso dos algoritmos PSO binários, cuja aplicabilidade centra-se maioritariamente na otimização de problemas num espaço discreto/binário. Outro

tipo são os algoritmos PSO híbridos, que têm por base a fusão da metodologia PSO a outras técnicas de computação evolutiva.

3.2 Como exibe Capacidade de Aprendizagem

O algoritmo PSO original é inicializado com uma população de soluções possíveis aleatórias ou predefinidas. denominadas partículas, cada uma possui os parâmetros velocidade e posição que lhe permite descolar-se no espaço de soluções. Cada partícula mantém o histórico das suas posições no espaço de procura, bem como o valor da função objetivo nessa posição, associado a isto é também armazenada a melhor solução que ela tenha alcançado até à data, chamado de pBest (personal best). Outra métrica analisada pelo PSO é o melhor valor de todos os valores obtidos por qualquer partícula da população, que é chamado gBest (global best). A meta heurística consiste em que a cada iteração do algoritmo, as partículas se desloquem em direção a pBest e gBest.

No PSO cada partícula ajusta o seu deslocamento com base na sua própria experiência e na dos seus companheiros, as partículas comunicam entre si, informando os valores da função objetivo nas respetivas posições, com o intuito de procurar um ótimo local.
Cada movimento de otimização da partícula é baseado em três parâmetros:

- Fator da sociabilidade – que determina a atração das partículas, para a melhor posição descoberta por qualquer elemento do enxame;
- Fator de individualidade – determina a atração da partícula com a sua melhor posição já descoberta;
- Velocidade máxima – delimita o movimento, uma vez que esse é direcional e determinado.

Na figura 2 é apresentado um diagrama de estados, que representa o algoritmo básico de PSO.

3.3 Ferramentas de desenvolvimento existentes

As ferramentas de desenvolvimento usadas para PSO são: o MatLab (Matrix Laboratory) que serve para encontrar o mínimo ou máximo de uma função de entrada única (single input) e saída única (single output) (SISO), ou uma função de múltipla entrada (multi-input) e saída única (MISO). Funciona com qualquer dimensão de tamanho do espaço de entrada, as diversas funções da caixa de ferramentas incluem funções de 2D até 7D. Também serve para treinar redes neuronais, para tal é necessário instalar Neural Net toolbox (caixa de ferramentas de Rede Neuronal), e construir uma função modelada na estrutura da mesma.

Outra ferramenta é PySwarms uma biblioteca de PSO em Python que é fácil de usar, sendo possível fazer uma otimização em menos de 8 linhas de código, esta caraterística é bastante importante especialmente para utilizadores que apenas querem um processo de otimização rápido e simples. Utilidades de suporte estão incluídas para avaliar o desempenho do optimizador, melhorias como utilidades de suporte são muito importantes na avaliação do comportamento do enxame, e do desempenho do optimizador.

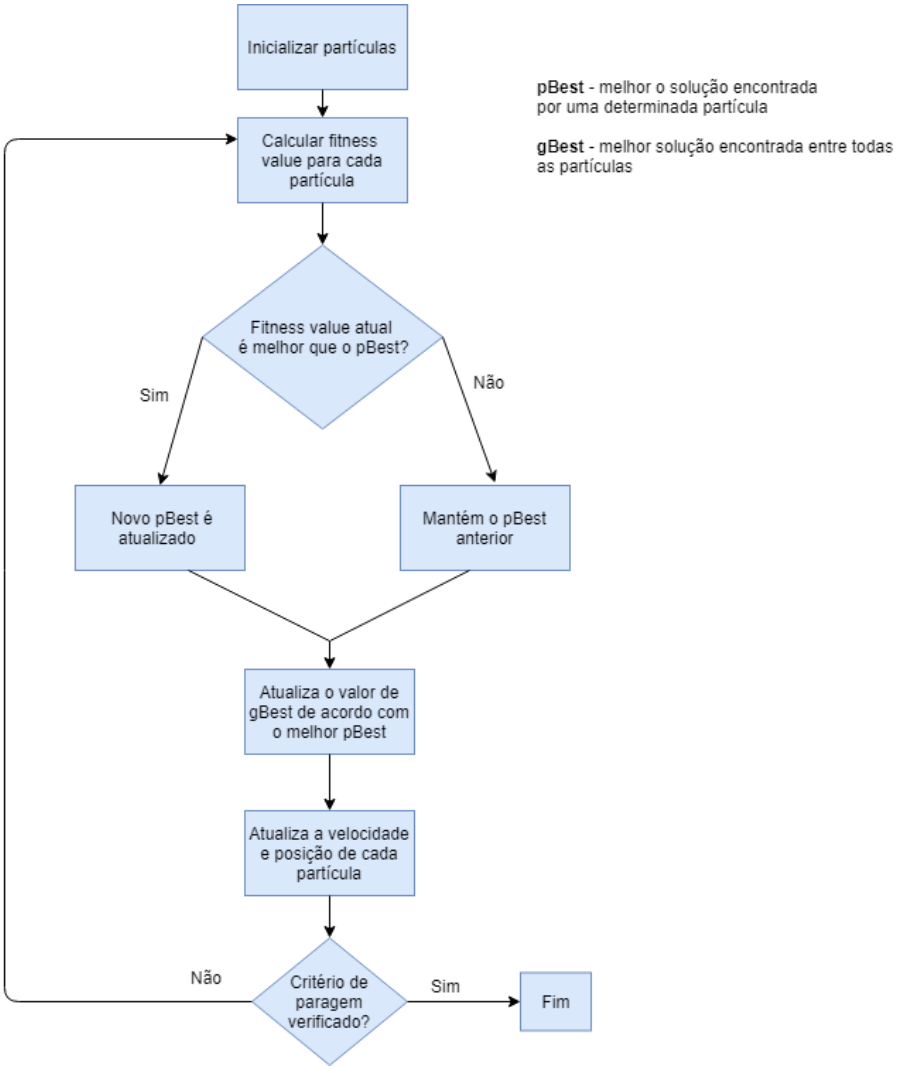

Figura 2. Diagrama de estados do algoritmo PSO básico

3.4 Soluções no mercado

Atualmente no mercado é possível verificar a utilização do *Particle Swarm Optimization* em sistemas desenvolvidos no âmbito de inúmeras áreas (indústria, científica, etc.). O facto de ser um sistema de aprendizagem que recorre a poucos parâmetros e sendo relativamente fácil de implementar, torna bastante apelativa a sua utilização, pois com pequenos ajustes é possível aplicar uma versão de um algoritmo em diferentes contextos.

- Otimização – PSO é utilizado ao nível da resolução de problemas de otimização com restrições, otimização multi-objetivo e problemas maximização e minimização.
- Evolução de redes neuronais – PSO têm sido usadas para treinar redes neuronais, não só ao nível dos pesos, mas também na estrutura das RNAs. Os resultados obtidos em várias áreas demonstram que o método é simples e eficiente, e que aliado a metodologias de treino tradicionais, como por exemplo backpropagation algorithm, produz resultados bastante positivos.

4 Support vector machine (SVM)

4.1 Descrição e características

Uma *Support Vector Machine* (SVM) é um algoritmo que nos indica o quão longe estamos da resposta correta), que analisa dados e reconhece padrões. É usado para classificação e análise de regressão (estudo da relação entre uma variável dependente com variáveis independentes específicas).

O algoritmo SVM, é um classificador linear binário não probabilístico, que possui um banco de dados, que têm objetos pré-categorizados (objetos cuja classe é conhecida), e prevê para cada nova entrada, a qual de duas classes existentes irá pertencer, com base nas suas características que irão ser comparadas.

Uma representação de modelos SVM é através de pontos no espaço, mapeados de maneira que os exemplos de cada categoria sejam divididos por um espaço claro que seja tão amplo quanto possível. Os novos exemplos são então mapeados para esse mesmo espaço e é-lhes atribuído uma categoria consoante o lado em que foram colocados.

Por outras palavras, o que as SVM fazem é encontrar uma linha de separação (denominada por hiper-plano) que procura maximizar a distância entre os pontos mais próximos de cada categoria (denominados por vetores de suporte), de modo a minimizar o erro de generalização.

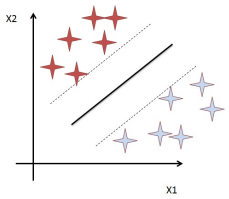

Figura 3. Exemplo da representação de um SVM

4.2 Como exibe Capacidade de Aprendizagem

Pensemos numa *Support Vector Machine*, que tem a capacidade de decidir se um animal é um cão ou um gato. Esta maquina toma a decisão com base nas características do comprimento do focinho e na geometria das orelhas. Se fizermos este exercício mental, podemos associar aos cães um focinho mais comprido, e aos gatos associar orelhas com uma geometria mais pontiaguda. Assim se pedirmos a classificação de um cão, e de um gato, ele facilmente nos vai dar a classe correta para cada um, cão e gato respetivamente.

E assim aconteceria para um número de casos infinitos, mas pensemos no caso dos cães de raça pug, que são conhecidos por ter um focinho achatado. Para que classe esta SVM iria mapear um pug? Até porque as orelhas dos pugs são relativamente pontiagudas, muito provavelmente iria classifica-lo como gato, o que não é verdade. Nestes casos estamos perante um outlier, e na figura 4, está a representação de um outlier no espaço.

Como reage uma SVM na presença de um outlier? Bem a resposta é depende! Se o outlier for de pouco impacto tal que a SVM consiga atribuir uma categorização correta, ele é considerado, se não, este outlier poderá no limite ser descartado, para continuarmos a ter uma separação linear, para que, tal como referido acima, continuemos a ter o mínimo de erro de generalização.

Mas o pugs não são os únicos cães com focinho achatado, os boxers também o são.

Figura 4. Exemplo de um outlier

Então se calhar com estas características, estas duas classes são sobrepostas, isto é, não existe uma separação linear entre elas, ou até pode não existir uma separação sequer. O que fazer nestes casos? Como manter a linearidade? É aqui

que entra o *Kernel Trick* que consiste na incrementação dimensional, para que se possa encontrar um hiperplano linear, tal como é mostrado na figura 5.

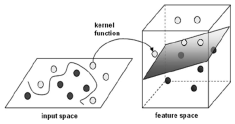

Figura 5. Passagem de 2D para 3D

Esta incrementação é momentânea, para não aumentar a complexidade computacional e a sua despesa, revertendo a dimensão anterior como se pode ver na figura 6.

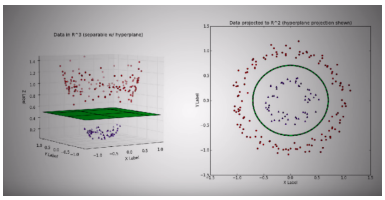

Figura 6. Passagem de 3D para 2D

O *Kernel Trick* é um bom exemplo da forma de aprendizagem das SVM's, que permite aumentar a sua capacidade, capacidade essa que pode ser entendida como a habilidade de uma máquina aprender qualquer conjunto de treino sem que cometa erros de generalização. Esquematicamente falando, também pode ser entendido como a maximização da separação entre as classes.

Vejamos o seguinte exemplo, um botânico com memória fotográfica, perante uma nova árvore diz que essa não é uma árvore, pois o número de folhas é diferente de todas as árvores que ele já viu, isto é um exemplo de demasiada capacidade.

Por outro lado, um algoritmo com pouca capacidade é como o irmão preguiçoso do botânico, que declara que qualquer coisa que seja verde é uma árvore (Burges, 1998). Em ambos os casos, não se pode fazer uma boa generalização, e por isso uma SVM com uma boa capacidade tem que encontrar um meio-termo razoável.

SVM's também tem a capacidade de categorizar com várias classes, recorrendo a sub-SVM's para mapear entre as várias classes, e encontrando a certa através de uma espécie de votação. Outra peculiaridade é a de ter essa mesma capacidade, para classes com pesos diferentes.

4.3 Ferramentas de desenvolvimento existentes

Existem várias ferramentas para desenvolvimento de SVM's, desde bibliotecas de várias linguagens de programação, até interfaces gráficas de drag and drop.

- LIBSVM - Biblioteca implementada em várias linguagens de programação, C, C++, Java, Python, R, S+, MATLAB, Perl, Ruby e LABVIEW. Estas bibliotecas dão suporte a classificação com várias classes, e classificação de classes com diferentes pesos.
- KNIME - Ferramenta de mineração de dados baseada numa interface gráfica de utilizador.
- Orange - Outra ferramenta de mineração de dados baseada numa interface gráfica de utilizador.
- mySVM - escrito em C++, é semelhante à LIBSVM, que inclui também funções de Kernel linear, polinomial e base radial.
- RapidMiner - Um interface gráfica de utilizador mais sofisticada, também ela uma ferramenta de mineração de dados, que inclui várias implementações de SVM, como por exemplo a LIBSVM.
- SVM Light – Outra biblioteca semelhante à LIBSVM e à mySVM, escrita em C++, que é conhecida pela sua performance.
- Weka – Uma popular ferramenta de mineração de dados que possui uma implementação SVM.

4.4 Soluções no mercado

O algoritmo de SVM tem sido aplicado a uma série de problemas de reconhecimento de padrões, regressão, estimação de densidade, deteção de novidades, entre outros. Aqui está uma lista de alguns exemplos.

- Deteção Facial - o algoritmo SVM classifica partes de uma imagem como face e outras como não-face, e cria um quadrado em volta da cara. Esta técnica pode ser vista nas câmaras de telemóveis.
- Reconhecimento de escrita manual - As SVMs podem ser usadas para reconhecer caracteres escritos manualmente.
- Classificação de imagens - O uso de SVM fornece uma boa precisão de pesquisa para classificação de imagens. Fornece melhor precisão em comparação com as técnicas tradicionais de pesquisa baseadas em consultas.
- Bioinformática - SVM é usado para identificar a classificação de genes, classificação de proteínas, pacientes com base no seu código genético, classificação de cancro, entre outros problemas biológicos.
- Deteção de proteínas e deteção de homologia remota - Aplicar algoritmos SVM para deteção de homologia remota de proteínas.
- Categorização de texto e hipertexto - as SVM's permitem a categorização de texto e hipertexto usando dados de treinamento para classificar documentos em diferentes categorias. Um exemplo é a classificação de artigos de notícias em "negócios" e "filmes".
- Controlo Previsível Generalizado - CPG baseado em SVM pode ser usado para controlar dinâmicas caóticas com parâmetros úteis.

5 Conclusão

Com a execução deste trabalho, conseguimos tirar algumas conclusões sobre estes assuntos, das quais se destacam algumas que iremos enunciar de seguida.

O *Case Based Reasoning* permite a construção de um protótipo antes de estar completa a estruturação do domínio permitindo assim diminuir a necessidade de aquisição de conhecimento, tendo também uma aprendizagem automática de novos casos e o reúso de conhecimento armazenado no repositório de dados.

Uma conclusão muita evidente no caso da *Particle Swarm Optimization*, é que quanto mais partículas são usadas, mais rápida é encontrada a solução, mas com um acréscimo de partículas, pode vir também um acréscimo de carga computacional. Por sua vez, um grupo inteligente com poucas partículas, corre o risco de não encontrar a solução ótima sequer.

No caso das *Support Vector Machines*, apesar de serem consideradas, por alguns autores, como um método de aprendizagem supervisionado, nós entendemos que estas são independentes. Visto que, com um conjunto de pontos, com a definição do hiperplano, e com a sua capacidade de generalização, são capazes de categorizar um dado caso corretamente, sem que seja preciso alguém lhe indicar o quão longe está da resposta correta.

Quando comparado com o CBR, podemos dizer que este olha para o passado para melhorar o presente, já um algoritmo PSO olha para o presente para melhorar o futuro. Um PSO é bom para usar, quando não há experiência. As SVM's seguem uma ideologia idêntica aos CBR.

6 Bibliografia e Referências.

- Aamodt A., Plaza E., "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches", in AI Communications, Vol. 7, No 1, pages 39-59, 1994.
- Paulo Novais, José Neves, "Raciocínio Baseado em Casos", Departamento de Informática, Escola de Engenharia, Universidade do Minho
- David Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning", Addison Wesley, 1989.
- Gerhard Vender, Jaroslaw Sobieszczanski-Sobieski, "Particle Swarm Optimization", AIAA Journal Vol.41 No.8 August 2003
- https://en.wikipedia.org/wiki/Particle_swarm_optimization Last accessed October 10, 2018
- Russel C.Eberhart, Yuhui Shi, "Particle Swarm Optimization: Developments, Applications and Resources", Vol 1. Evolutionary Computation 2001
- Kennedy, J. and Eberhart, R. C., "Particle swarm optimization", Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ. pp. 1942-1948, 1995.
- Nello Cristianini, John Shawe-Taylor, "An Introduction to Support Vector Machines and other kernel-based learning methods", Cambridge University Press, 2000.
- https://ljvmiranda921.github.io/projects/2017/08/11/pyswarms/