2
3
4
5
6
7
8
9
10
11
12
13

Universidade do Minho

Trabalho Prático de
Aprendizagem e Extração de Conhecimento

Sistemas de Aprendizagem

Carlos Pereira (PG32826) e Rúben Cruz (PG30869)

24 de Outubro de 2017

Conteúdo

Capítulo 1

Resumo

Neste trabalho iremos aprofundar 3 sistemas de aprendizagem: Raciocínio Baseado em Casos, Árvores de Decisão e *Support Vector Machine*. Em todos eles, vai ser feita uma descrição, as capacidades de aprendizagem, ferramentas de desenvolvimento que atualmente estão disponíveis, e quais as soluções que são utilizadas no mercado.

Capítulo 2

Introdução

Machine Learning é definido como a área que dá aos computadores a capacidade de aprender sem serem especificamente programados para tal. Um computador aprende com a experiência em função de uma tarefa, se a sua performance nessa tarefa aumentar com a experiência. No perfil de sistemas inteligentes do mestrado em engenharia informática, procuramos encontrar os melhores métodos para dar aos computadores esta habilidade. Especificamente na unidade curricular de aprendizagem e extração do conhecimento, percorremos várias metodologias utilizadas para desenvolver algoritmos inteligentes, tais como raciocínio baseado em casos, redes neuronais, árvores de decisão, etc.

Dentro deste contexto, o nosso trabalho foca-se em explorar os conceitos de raciocínio baseado em casos, árvores de decisão e *support vector machines*. Ao longo deste documento iremos dar a nossa análise sobre estes conceitos explicando o seu funcionamento, que tipo de problemas pretendem resolver e aplicações.

Capítulo 3

Raciocínio Baseado em Casos

De uma forma muito resumida, raciocínio baseado em casos procura resolver um problema, analisando problemas anteriores semelhantes e providenciando uma solução baseada nesses problemas. Este sistema proposto no final da década de 1970, utiliza uma base de dados com casos semelhantes, que são utilizados para tentar resolver um novo problema. Após a resolução o novo caso é adicionado à base de dados, para no futuro ser utilizado como possível referência.

3.1 A família de raciocínio baseado em casos

O termo raciocínio baseado em casos é um termo genérico que descreve uma família de métodos que procuram resolver problemas a partir de uma base de dados, cada um com as suas vantagens e limitações. Vamos aqui tentar indicar alguns, apesar de nas futuras secções, mencionarmos raciocínio baseado em casos de forma generalizada.

3.1.1 Raciocínio baseado em exemplos

Este termo é utilizado para os métodos que definam um conceito como o conjunto de todos os seus exemplos. Neste método, um problema é uma tarefa de classificação, ou seja, encontrar o grupo certo para o nosso novo caso, e a solução é a classe de casos similares. Sendo assim, este método não procura adaptar soluções a problemas.

3.1.2 Raciocínio baseado em instâncias

Este método é uma especialização do raciocínio baseado em exemplos. Este método baseia-se numa sintaxe elevada. De modo a ser preciso, este método

requer um conjunto vasto de exemplos, geralmente representados de forma simples (por exemplo, vetores).

O algoritmo de *k-nearest neighbors algorithm*, *kernel methods* e *radial basis function network* são exemplos de algoritmos que se enquadram neste método.

3.1.3 Raciocínio baseado em memória

O foco destes métodos é a memória que o sistema utiliza e os seus métodos de organização e acesso. É comum utilizar-se métodos de técnicas de processos paralelos nesta metodologia.

3.1.4 Raciocínio baseado em casos

Apesar de ser utilizado para descrever o conjunto de famílias, os métodos de raciocínio baseado em casos são distinguíveis dos outros métodos aqui mencionados. A informação tem tendência a ser rica e completa, e costuma possuir métodos complexos na organização interna. Estes métodos também se distinguem pela capacidade de adaptarem as soluções e os seus casos a novas circunstâncias.

3.1.5 Raciocínio baseado em analogias

Este método distingue-se dos restantes pela capacidade de conseguir resolver problemas utilizando casos de domínios diferentes, enquanto que os restantes métodos tendem a utilizar apenas um domínio.

3.2 O ciclo de raciocínio baseado em casos

Nesta secção iremos falar sobre o ciclo que define os métodos de raciocínio baseado em casos. Este ciclo, representado na figura 2.1, é iniciado com a entrada de um novo caso, e termina quando este possuir uma solução, sendo ambos posteriormente adicionados à base de casos. Durante esta secção iremos falar em detalhe sobre ciclo.

3.2.1 Recuperação

O primeiro passo deste ciclo foca-se em encontrar na base de casos aquele que o nosso algoritmo considerar como o mais próximo ao problema. Para o algoritmo ser capaz de fazer a recuperação, é necessário criar uma métrica

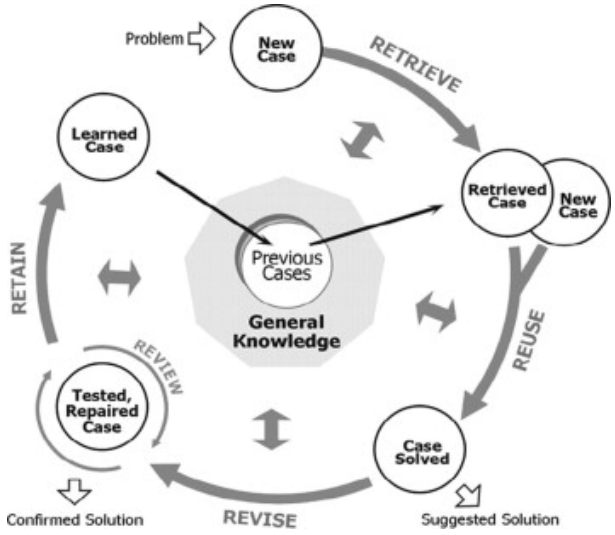

Figura 3.1: Constituição do ciclo de raciocínio baseado em casos

das características dos casos. Então, podemos simplesmente utilizar o calculo da distância euclidiana entre dois pontos, ou seja, sendo e_i^1 a caracterização numérica da característica X_i do problema em questão, e e_i^2 a caracterização numérica da mesma característica num dado caso na base de casos, então podemos calcular a proximidade entre dois casos com:

$$d(e^1, e^2) = \sqrt{\sum_{i=1}^{n}(e_i^1 - e_i^2)^2}$$

Podemos também atribuir um peso de importância a cada característica W_i, criando assim a função de calculo de distância:

$$d(e^1, e^2) = \sqrt{\sum_{i=1}^{n} W_i(e_i^1 - e_i^2)^2}$$

Estes são apenas exemplos simples de calculo de distância, sendo que existem funções mais complexas, que permitem tornar esta metodologia mais eficiente.

No final, é escolhido o caso mais semelhante ao problema como referência.

3.2.2 Reutilização

A partir do caso recuperado, a solução desse caso é proposta como a nova solução do nosso problema. Em situações em que seja apenas necessário classificar o problema, a solução pode ser simplesmente copiada para o novo caso. Caso o problema necessite que a solução seja adaptada existem duas

principais maneiras de adaptar a solução. A primeira é reutilizar a solução do passado, denominada de reutilização por transformação. Neste caso a solução não é exatamente copiada, mas sim adaptada através de transformadores operacionais. A segunda é reutilizar o método que construiu a solução, denominada de reutilização derivacional. Neste caso, o caso recuperado possui alguma informação sobre o método de construção da solução, tal como uma justificação dos operadores usados, alternativas geradas, etc.

3.2.3 Revisão

Após ser apresentada uma possível solução para o problema, é necessário valida-la, e, caso não seja válida, podemos tomar isto como um método para descobrir o motivo do insucesso. Esta fase é composta por duas partes.

Avaliar a solução

A avaliação da solução é feita aplicando a solução num ambiente real. Esta parte não costuma ser enquadrada no contexto de raciocínio baseado em casos, visto que a aplicação e respetiva análise do sucesso são feitos nesse ambiente. Como este processo pode demorar algum tempo, a solução e o caso podem ser guardados (aprendidos) temporariamente até à solução ser validada.

Reparação

Caso a solução não tenha sido validada, é necessário analisar porque é que falhou e reparar a solução. Primeiro é necessário analisar a causa da falha da solução. Tendo em conta a sua explicação, o sistema pode tentar adaptar de novo a solução para esta funcionar.

3.2.4 Retenção

Após termos corrigido o nosso novo problema, é necessário guarda-lo na base de casos para podermos reutiliza-lo mais tarde, e assim, aprender com ele.

Começamos por extrair a informação que obtivemos a partir deste caso, podendo assim regista-la. Nesta informação podemos registar os dados do problema, a solução apresentada e mesmo justificações e alguns registos sobre as falhas que tenham ocorrido com outras possíveis soluções apresentadas.

De seguida é necessário indexar o problema dentro da base de casos por forma a tornar o seu acesso rápido e eficiente. Este é um problema central

dentro da área do raciocínio baseado em casos, pelo que diversas abordagens existem, cada uma com vantagens e desvantagens.

Finalmente, devemos integrar o novo caso dentro da base de casos. Isto melhora a indexação feita e permite que o nosso sistema calcule melhor as similaridades com um novo caso. No final de tudo, podemos testar esta nova indexação, reintroduzindo o problema e verificando se o sistema funciona devidamente.

3.2.5 Representação dos Casos

Como os acessos à base de casos que o método de raciocínio baseado em casos utiliza são feitos de cada vez que queremos introduzir um novo caso, torna-se essencial sabermos representar bem os casos e guarda-los de forma eficiente para mais tarde podermos recuperar rapidamente a informação de que necessitamos. Para além da necessidade de tornar eficientes as 4 fases do ciclo do raciocínio baseado em casos, é também necessário estudar a melhor forma de estruturar os nossos casos na base de dados. Vamos nesta subsecção falar sobre dois modelos de memória importantes na área do raciocínio baseado em casos.

Modelo de memória dinâmica

O objetivo deste modelo de memória é organizar casos específicos que possuem propriedades semelhantes dentro de uma estrutura geral, denominada de episódio generalizado. Um episódio generalizado contém 3 diferentes objetos: normas, casos e índices. As normas são as características comuns a todos os casos dentro do episódio generalizado e os índices são as características que os distinguem.

Os processos de procura e armazenamento do novo caso são semelhantes. Quando um novo caso é inserido, são analisadas as suas propriedades, e, quando é encontrado um episódio generalizado com uma ou mais características em comum com o novo caso, a procura continua dentro deste episódio generalizado, até eventualmente, entrarmos dentro de um caso que tem as características mais semelhantes. Para armazenar, quando uma característica do novo caso é semelhante a uma característica de outro caso, um novo episódio generalizado é encontrado. Assim, cada caso pode pertencer a mais do que um episódio generalizado. Então, um caso é recuperado, encontrando o episódio generalizado com mais características em comum com o novo caso, e, dentro deste, encontrando o caso com características ainda mais similares.

Modelo de categoria e exemplar

Neste modelo, a memória é composta por uma estrutura de categorias, casos e apontadores para índices. Cada caso está associado a uma categoria. Os índices apontam para categorias ou casos e podem ser de três tipos: apontadores das características das descrições do problema para casos ou categorias, apontadores de categorias para os respetivos casos, e ligações com diferenças entre casos da mesma categoria, que diferenciam-se apenas e uma ou poucas mais características.

Para recuperar um caso, combinamos as características de forma a gerar um apontador para a categoria, ou caso, que mais características em comum tem com o novo caso. Se o apontador for para uma categoria fazemos uma procura transversal dentro dos seus casos. Guardamos um novo caso encontrando o caso recuperado e estabelecendo os indíces das características próprios.

3.3 Aplicações de raciocínio baseado em casos

Nesta secção iremos falar sobre algumas áreas onde esta metodologia é aplicada atualmente.

3.3.1 Diagnósticos médicos

Muitos sistemas de ajuda médica utilizam um sistema que utiliza casos de doentes com sintomas semelhantes, de modo a ajudar a diagnosticar e tratar novos pacientes. Os sistemas de raciocínio baseado em caso enquadram-se perfeitamente neste tipo de problemas e por isso são muito utilizados atualmente.

3.3.2 Ajuda no Apoio ao Cliente

Diagnosticar os problemas técnicos que um cliente tem nos seus aparelhos através de linhas de apoio não é fácil. Nesse sentido, sistemas de raciocínio baseado em casos são utilizados para facilitar o trabalho dos operadores.

3.3.3 Desenvolvimento artístico

Alguns artigos no final dos anos 90 e início do século XX demonstram a capacidade de sistemas que utilizam peças musicais em saxofone, desenvolverem

novas peças. Estes sistemas utilizam raciocínio baseado em casos que regista variáveis associadas às composições analisadas, e a partir destas, desenvolve novas peças.

Também dentro desta área começaram a surgir artigos sobre métodos que demonstram a capacidade de gerar poemas utilizando uma base de dados de poemas.

3.3.4 Biologia Molecular

Também na área da biologia molecular têm surgido artigos que demonstram o potencial deste tipo de sistemas. Alguns cientistas procuram utilizar sistemas de raciocínio baseado em casos para tentarem acelerar o processo de crescimento de cristal em proteínas. Este não é o único caso em que este tipo de sistemas é utilizado em biologia molecular.

Capítulo 4

Árvores de Decisão

Uma árvore de decisão é uma representação da classificação e possíveis consequências acerca de um input. A aprendizagem é não incremental através de exemplos, onde é feita a etiquetagem da classe sobre tuplos/subconjuntos do input, onde cada nodo denota um atributo, cada ramo da árvore representa o resultado de um teste/condição, e cada folha representa a classe. O processo é recursivamente efetuado em cada tuplo/subconjunto gerado, e só pára quando já não é possível efetuar mais partições. Este é um exemplo de um algoritmo *greedy* sobre as TDIDT (Top-Down Induction of Decition Trees).

Os métodos de aprendizagem usados nas TDIDT são consideravelmente menos complexos que os implementados em sistemas que expressão os seus resultados numa linguagem mais poderosa.

Exemplos de classificação:

- O diagnóstico de uma condição médica através dos sintomas do paciente, no qual as classes podem ser vários estados de doenças, ou possíveis terapias;
- Determinar a estratégia de um jogo de xadrez, com as classes de ganhou brancas, perderam as brancas e empate;
- Através da observação atmosféricas, concluir se uma tempestade é provável, improvável ou possível.

4.1 A indução

A base de indução é um conjunto/universo de objetos que são descritos consoante uma coleção de atributos, onde cada atributo mede a importância de

alguma característica de um objeto. Por exemplo, considerando os dias da semana, se os objetos forem as manhãs de Sábado e as tarefas de classificação envolvem a meteorologia, então os atributos podem ser:

- céu, com atributos ensolarado, nublado, chuvoso
- temperatura, com atributos frio, amena, quente
- humidade, com atributos alta, normal
- vento, com atributos sim, não

Estes atributos formam uma linguagem de caracterização dos objetos sem uma ordem definida. Em particular, uma manhã de sábado pode ser descrita da seguinte forma:

- céu: nublado
- temperatura: frio
- humidade: normal
- vento: não

No universo dos objetos, cada objeto corresponde a um conjunto mutuamente exclusivo de classes, ou seja, cada objeto pertence a um conjunto de classes que não podem ocorrer ao mesmo tempo. Para simplificar este tipo de definição, vamos assumir, sem perda de generalidade para um número arbitrário de classes, que existem duas classes P e N. Em tarefas de indução sobre duas classes, *two-class induction*, os objetos da classe P e N são referidos como instâncias positivas ou negativas, respetivamente, do conceito a ser aprendido.

A outra base neste sistema é a aprendizagem através de um conjunto de objetos aos quais a classe é conhecida. A tarefa de indução desenvolve uma regra de classificação que consegue determinar a classe de qualquer objeto através dos seus valores e atributos. Um problema que surge é o facto de que se dois objetos, de classes diferentes, contenham valores idênticos para cada atributo, e assim torna-se impossível distinguir os dois objetos baseando-se nos atributos e valores. Nesta situação, o conjunto de casos de treino torna-se inadequado.

Nesta tabela está um exemplo de um conjunto de objetos que podem ser usados na aprendizagem da máquina. Usa os atributos de "Manhã de sábado", onde cada valor de cada objeto é apresentado juntamente com a classe pertencente. E agora representada a árvore de decisão do respetivo exemplo acima representado.

Nº	Atributos				Classe
	Céu	Temperatura	Humidade	Vento	
1	ensolarado	quente	alta	não	N
2	ensolarado	quente	alta	sim	N
3	nublado	quente	alta	não	P
4	chuvoso	amena	alta	não	P
5	chuvoso	frio	normal	não	P
6	chuvoso	frio	normal	sim	N
7	nublado	frio	normal	sim	P
8	ensolarado	amena	alta	não	N
9	ensolarado	frio	normal	não	P
10	chuvoso	amena	normal	não	P
11	ensolarado	amena	normal	sim	P
12	nublado	amena	alta	sim	P
13	nublado	quente	normal	não	P
14	chuvoso	amena	alta	sim	N

Tabela 4.1: Exemplo de um conjunto de objetos de aprendizagem

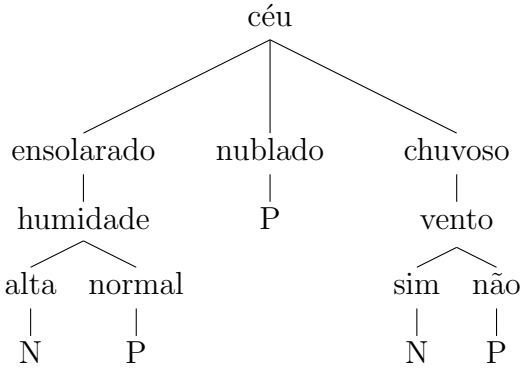

Figura 4.1: Árvore de decisão

Cada folha representa uma classe, e cada nodo representa os atributos, onde cada ramo dá um possível resultado.

4.2 ID3

Uma forma de indução é a geração de todas as árvores de decisão possíveis, que classifiquem corretamente o conjunto utilizado para aprendizagem, e utilizar a mais simples. O número de árvores é finito, mas muito grande, fazendo com que esta ideia só possa ser utilizada em tarefas pequenas, pois levaria muito tempo a concluir em tarefas maiores. O algoritmo ID3 (Iterative Dichotomiser 3) foi desenhado no sentido oposto, ou seja, existem muitos atributos e conjuntos mas consegue-se obter uma árvore de decisão razoavelmente boa sem muito poder de computação.

Um subconjunto do conjunto inicial é escolhido aleatoriamente e forma-se através dele uma árvore de decisão. Este subconjunto chamar-se-á *window*. Esta árvore classifica corretamente todos os objetos na *window*, e todos os outros objetos são classificados através desta árvore. Caso a árvore devolva uma resposta correta para todos esses objetos, então ela é correta para todo o conjunto inicial e termina assim o algoritmo. Senão, adicionam-se todos os objetos aos quais a árvore não respondeu corretamente à *window* e recomeça o algoritmo.

Assim, ao fim de algumas iterações, obtém-se a árvore de decisão para o conjunto inicial.

4.3 C4.5

Este algoritmo é uma extensão do acima vista, o *ID3*. A árvore de decisão criada tem o mesmo fim, mas neste algoritmo suporta tanto atributos discretos como contínuos. A construção de árvores de decisão é feita da mesma forma que o *ID3*, mas utilizando o conceito de entropia. A cada nodo da árvore, o algoritmo escolhe um atributo que divida o conjunto dos objetos em subconjuntos. Este conceito de divisão é normalização do ganho de informação. O atributo que tenha o valor mais alto é o escolhido.

4.4 Gestão de Projetos/Negócios de Risco

Dependendo das situações, a tomada de decisões em negócios ou projetos de risco varia conforme as oportunidade e ameaças. Mas suponhamos que conseguimos calcular o valor monetário expectável em cada decisão. Assim, utilizando Árvores de Decisão, consegue-se calcular tal valor para a análise de riscos.

Suponhamos que uma organização usa um software desatualizado e sem suporte. Alguns acionistas dessa empresa pensarão que ao atualizar para um

software mais recente pode poupar grandes quantidades de dinheiro, mas no entanto, a outra parte dos acionistas pensa que ao manterem-se no mesmo software é a melhor opção, pois é a mais segura. Suponhamos agora que o conjunto de acionistas que suporta a atualização se divide em dois subconjuntos: aqueles que preferiam comprar o software e aqueles que o preferem desenvolver dentro da própria empresa.

Ao utilizar uma árvore de decisão, explora-se todas as possibilidade e consequências das decisões tomadas, pode-se quantificar o ganhos e perdas.

Capítulo 5

Support Vector Machines

Support vector machines são algoritmos na área de machine learning capazes de fazer classificação, regressão linear e deteção de *outliers*. Estes algoritmos encaixam-se na área dos algoritmos de *machine learning* supervisionada, ou seja, necessitam de uma base de dados com casos previamente avaliados para funcionar.

5.1 Funcionamento das SMV's

Genericamente, as SMV's recebem um conjunto de exemplos, definidos como pontos num espaço cartesiano (com n dimensões, sendo cada dimensão um parâmetro). Estes pontos estão catalogados como pertencentes a uma de duas categorias. Sendo um algoritmo de classificação, o objetivo é calcular a que categoria pertence um novo caso, também ele representado por um ponto neste espaço. As SMV's calculam uma divisão entre os pontos da primeira categoria, e os pontos da segunda, de tal forma que, a distância entre a linha e os pontos seja a maximizada.

Nesta secção, vamos explorar os algoritmos que definem esta classe de ferramentas.

5.1.1 Regressão Logística e Intuição

Regressão logística é um modelo de regressão cujo objetivo é categorizar um dado caso. O resultado é portanto generalizado como 0 ou 1, dependendo da categoria a que pertence. Para tal é necessário um conjunto de casos exemplares devidamente classificados. Para tal, utilizamos uma função $h(x)$ em que x é um ponto num espaço vetorial. Colocando $h(x) = g(\theta x)$, para um dado vetor θ, pretendemos obter

$$g(z) = \frac{1}{1+e^{-z}}$$

Com esta formulação do problema, é possível mapear todos os possíveis valores de entrada para um alcance $[0, 1]$, em que caso $h(x)$ seja igual ou superior a 0.5, declaramos que o resultado é a classificação 1 e declaramos 0 caso contrário. O objetivo é, então, encontrar um valor para θ tal que para todos os casos que temos classificados, $\theta x \geq 0$ caso a classificação seja 1 e $\theta x < 0$ caso contrário.

Para aproximar θ do valor que melhor denote os casos que lhe inserimos classificados (ou seja, para tentar tornar o algoritmo preciso quando chegar um novo caso) existe uma função chamada a função de custo. Esta função possuí a propriedade de que o seu valor seja 0 quando todos os casos que temos na base de dados forem todos perfeitamente classificados pela função $h(x)$, ou seja $h(x) = 1$ se x for um caso positivo e $h(x) = 0$ caso contrário. Sendo um problema de *machine learning*, é potencialmente impossível atingir tal objetivo com a função de custo, e, sendo assim, o nosso objetivo é obter um θ que a minimize. A função para regressão logística é definida da seguinte forma:

$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} Cost(h_\theta(x^i), y^i)$, onde

$Cost(h_\theta(x), y) = -log(h_\theta(x))$ se $y = 1$

$Cost(h_\theta(x), y) = -log(1 - h_\theta(x))$ se $y = 0$

Utilizando cálculo, e a derivada da função de custo, podemos então progredir iterativamente para encontrar um valor mínimo (ou aproximado) para a função de custo.

5.1.2 Generalização para SMV's

Como o algoritmo de regressão logística classifica os resultados em 2 valores (existem algoritmos capazes de fazer uma classificação multivalor, que é um caso mais geral do problema aqui descrito), existe um hiperplano no espaço cartesiano em questão, que separa os valores classificados como 0 e os valores classificados como 1. A zona que separa os casos, pode ter uma área variada, sendo que o hiperplano escolhido para categorizar os valores pode variar. Por exemplo, na figura 5.1, temos 2 hiperplanos representados (neste caso, sendo este um espaço bidimensional, o hiperplano é uma simples linha) que separam os dois conjuntos de pontos. Por intuição, a linha mais escura é a linha que melhor separa os dois grupos, possuindo uma maior margem do que a linha cinzenta.

O algoritmo de regressão linear não permite sempre encontrar a melhor fronteira entre os dois grupos e é esta a característica que divide este algoritmo com as SMV's.

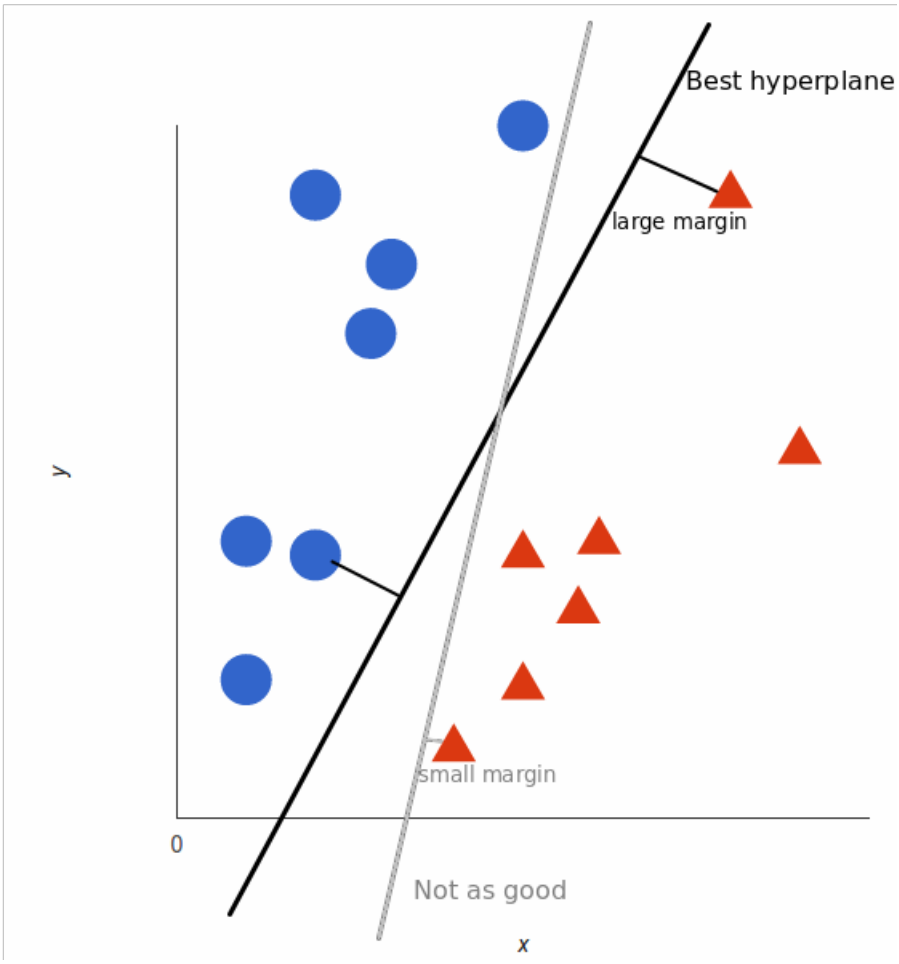

Figura 5.1: 2 linhas que separam os dois conjuntos de pontos no plano, sendo que a linha negra é a que, intuitivamente, os separa melhor.

Nas SMV's, a função de custo é alterada para a chamada decisão de fronteiras de SMV definida da seguinte forma:

$$min_\theta \; C \sum_{i=1}^{m} y^i Cost_1(h_\theta(x^i), y^i) + (1-y)^i Cost_0(h_\theta(x^i), y^i) + \frac{1}{2} \sum_{i=1}^{n} \theta_i^2$$

onde C é uma constante escolhida que afeta o processo de decisão, em particular, num sistema com *outliers*, se o valor de C for pequeno o suficiente, o sistema conseguirá detetar e ignorar os *outliers* devidamente.

5.1.3 Kernels

Nas subsecções anteriores, mencionamos a utilização das SMV's apenas nos casos em que o problema pode ser resolvido uma fronteira que se define por um hiperplano. Esta opção nem sempre é a melhor escolha para separar os nossos dados nos contextos reais com que as SMV's se defrontam. Tomemos como exemplo a divisão feita na figura 5.2. Neste caso a divisão é claramente um problema de encontrar uma circunferência que obtenha os requisitos de maximizar a distância a todos os pontos. Contudo, com o algoritmo linear das SMV's não é possível.

Uma solução para o problema seria definir uma nova dimensão z, onde definíamos o valor em z de cada ponto por $z_i = x_i^2 + y_i^2$, e procurar resolver o problema com as dimensões x e z (apresentado o gráfico na figura 5.3). Esta solução tem os seus limites computacionais, visto que nos casos reais, são utilizadas várias dimensões, e calcular a nova pode tornar-se computacionalmente pesado.

Com os *kernels*, podemos utilizar um ou mais pontos no nosso espaço vetorial, e tentar definir a fronteira, em função das distâncias dos casos em função destes pontos inseridos. Podemos então dizer que os *kernels* são funções de transformação aplicadas aos nossos casos classificados. Por exemplo, no sistema acima, podemos definir um ponto $(0, 0)$ e utilizar um *kernel* gaussiano para transformar descrever a distância dos casos ao nosso ponto, e assim, encontrar a solução para o nosso problema.

5.2 Aplicações

As SMV's são consideradas um dos mais poderosos algoritmos dentro da área de *machine learning* e por isso, os casos mais comuns de exemplos atuais no dia-a-dia de aplicações utilizam SMV's. Nesta secção vamos apresentar alguns dos exemplos de aplicações no mundo real destes algoritmos.

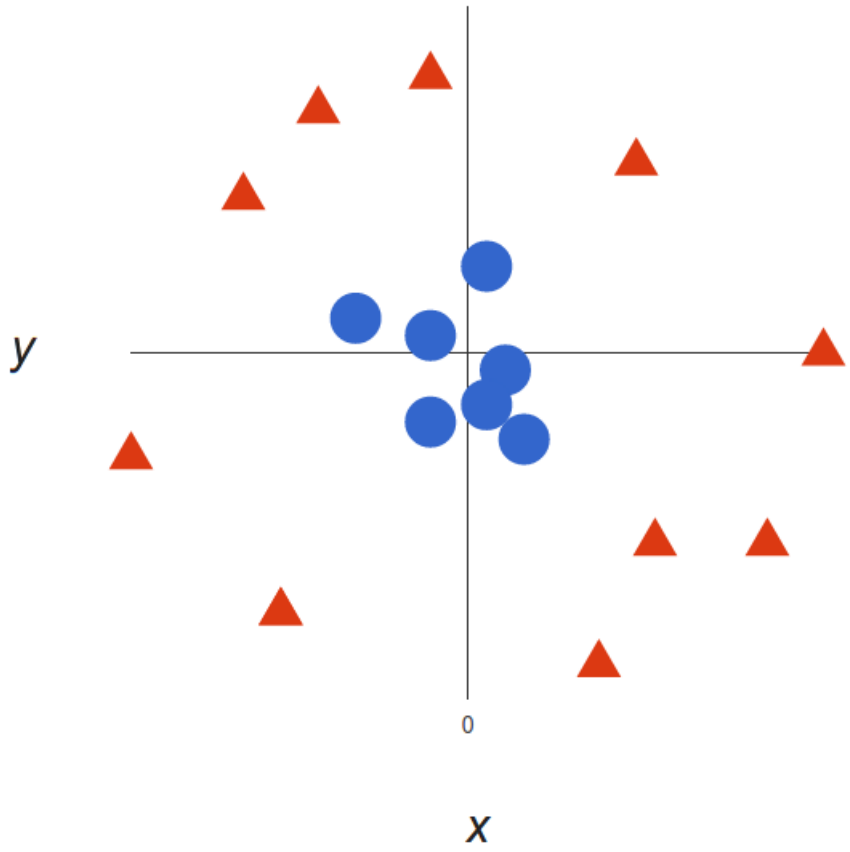

Figura 5.2: Problema de classificação que necessita de uma solução não linear de separação dos dados

5.2.1 Reconhecimento de Dígitos escritos à mão

Devido à sua capacidade de classificação, dado um conjunto de exemplos de dígitos escritos à mão, é possível utilizarmos SMV's para identificar um futuro dígito, e assim, permitir aos computadores reconhecerem escrita e documentos existentes em papel. Esta aplicação foi inicialmente apresentada em 1995 com uma margem de erro inferior a 12% (utilizando *kernels* polinomiais inferior a 4%).

5.2.2 Categorização de Textos

Categorização de textos é a tarefa de atribuir categorias predefinidas a documentos de texto. Sendo também esta uma área de classificação, podem ser utilizadas SMV's para tentar atribuir uma categoria a textos novos. Com SMV's é possível atingir performances superiores a 86%.

5.2.3 Identificação de Género utilizando imagens

Utilizando *kernels* lineares, é possível utilizar SMV's para identificar o género de uma pessoa utilizando um histórico de imagens previamente identificadas.

Capítulo 6

Conclusão

A constante atualização da base de casos, permite dar aos métodos de raciocínio baseado em casos uma melhoria contínua sobre os problemas, que não é trabalhada nos outros métodos aqui apresentados. Como a descrição de cada problema pode ser elaborada de forma adaptar-se melhor ao contexto do problema, este método é bastante útil nos problemas de situações reais como foi mostrado.

Através da utilização de árvores de decisão, pode-se construir um sistema de aprendizagem eficiente, no que toca à classificação de objetos. A sua representação permite que qualquer utilizador a possa utilizar e ter assim um output que possa ser medido e trabalho em qualquer caso ou circunstância. Através do conceito de entropia podemos medir qual o melhor atributo a escolher e assim, ter pelo menos uma garantia de que estamos a escolher o melhor atributo no momento para a construção ou ampliação da árvore.

As *support vector machines* são talvez os melhores métodos algorítmicos atuais para categorizar conjuntos de informação. Obtendo uma boa base de dados classificados, e utilizando uma boa aplicação dos kernels, podemos desenvolver sistemas poderosos, tal como as suas aplicações aqui mencionadas o demonstra.

Bibliografia

[1] http://scikit-learn.org/stable/modules/svm.html

[2] http://cs229.stanford.edu/notes/cs229-notes3.pdf

[3] https://en.wikipedia.org/wiki/Support_vector_machine

[4] http://data-flair.training/blogs/applications-of-svm/

[5] https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/

[6] http://www.brighthubpm.com/risk-management/48360-using-a-decision-tree-to-calculate-expected-monetary-value/

[7] http://www.aiai.ed.ac.uk/links/cbr.html

[8] http://www.iiia.csic.es/ mantaras/Emergent_CBR_Appl.pdf

[9] http://artint.info/html/ArtInt_190.html

[10] https://en.wikipedia.org/wiki/C4.5_algorithm

[11] https://en.wikipedia.org/wiki/Entropy_(information_theory)

[12] http://www.cs.umd.edu/ samir/498/10Algorithms-08.pdf

[13] Quinlan, J. R., "Induction of Decision Trees", Machine Learning 1: 81-106, Kluwer Academic Publishers, 1986