Universidade de Coimbra
Faculdade de Ciências e Tecnologia
Departamento de Engenharia Electrotécnica e de Computadores

Carlos André Ferreira Santos Fernandes

Algoritmo do Tipo Filter-Wrapper de Seleção de Features para Utilização na Seleção de Genes

Dissertação de Mestrado Integrado em Engenharia Electrotécnica e de Computadores.

Coimbra

Setembro de 2017

Universidade de Coimbra

Universidade de Coimbra
Faculdade de Ciências e Tecnologia
Departamento de Engenharia Electrotécnica e de Computadores

Algoritmo do Tipo Filter-Wrapper de Seleção de Features para Utilização na Seleção de Genes

por

Carlos André Ferreira Santos Fernandes

Dissertação de Mestrado Integrado em Engenharia Electrotécnica e de Computadores, ramo de especialização em Automação.

Orientador:	Prof. Doutor Rui Alexandre de Matos Araújo
Co-Orientador:	Prof. Doutor Francisco Alexandre Andrade de Souza

Júri

Presidente:	Prof. Doutor Fernando Manuel dos Santos Perdigão
Vogais:	Prof. Prof. Doutor João Filipe de Castro Cardoso Ferreira
	Prof. Doutor Rui Alexandre de Matos Araújo

Setembro de 2017

"When you know how to think it empowers you far beyond those who know only what to think."
Neil deGrasse Tyson

Agradecimentos

Dedico este agradecimento não só aos que me acompanharam no presente curso, mas a todos os que me acompanharam na minha vida académica e pessoal, e ajudaram na realização deste objetivo.

Em primeiro lugar gostaria de agradecer aos meus pais Carlos e Fernanda pelo esforço, apoio e incentivo ao longo da caminhada.

Um agradecimento especial à minha namorada Flávia pelo apoio incondicional.

Agradeço ao meu orientador Professor Doutor Rui Araújo e ao meu co-orientador Professor Doutor Francisco Souza pelos conselhos, opiniões, dedicação, rigor científico e paciência que permitiram o desenvolvimento desta dissertação.

Ao meu amigo João (Ninja) pelo apoio e companheirismo durante o desenvolvimento deste trabalho.

Aos meus restantes amigos e familiares, por todos os momentos proporcionados ao longo da minha vida académica, e pela sua amizade e apoio ao logo da vida.

Agradeço ainda a todas as outras pessoas que de forma direta ou indireta me ajudaram na realização desta dissertação.

Resumo

A sequenciação de genes através de microarrays de Ácido desoxirribonucleico (ADN) permite conhecer a ordem de milhares de genes. Esta técnica enquadra-se num grupo de técnicas de sequenciação de alta eficiência que chegam a gerar dados na ordem dos Terabytes (Tb). Neste contexto justifica-se a importância da aplicação de técnicas de seleção de genes e avaliação da sua importância, que permitam aos investigadores Bioinformáticos focarem-se nos genes com mais influência na existência de doenças ou anomalias.

Estas técnicas tentam reconhecer padrões que revelem a importância de cada gene na presença de doenças ou condições, conseguindo um duplo efeito de selecionar os mais importantes ou eliminar os menos importantes, e consequentemente diminuindo a dimensionalidade dos dados. Estes dois efeitos revelam-se importantes, pois, os dados selecionados levam a melhorias no tempo de treino de modelos preditivos e na exatidão de predição, devido à exclusão de dados redundantes.

No entanto, a seleção de genes e a avaliação da sua importância revelam ser um problema do ponto de vista Matemático, pois, o número de variáveis é muito maior do que o número de amostras, o que se torna num problema com muitas soluções possíveis.

Na literatura, existem vários algoritmos propostos para a resolução do problema baseando-se em abordagens estatísticas ou em aprendizagem máquina, sendo estas abordagens designadas como sendo do tipo filtro (*filter*) ou do tipo embrulho (*wrapper*), respetivamente [Guyon *et al.*, 2008], [Peng *et al.*, 2005].

Este trabalho propõe uma abordagem conjunta, que explora ambas as técnicas, usando a estatística da informação mútua e vários algoritmos de aprendizagem máquina, como o Naive Bayes, Máquinas de Vetores de Suporte (*Support Vector Machines*), árvores de classificação (*Classification Trees*) e k-Vizinhos-Mais-Próximos (*k-Nearest-Neighbor*).

Para avaliar a importância do método proposto, este é aplicado com técnicas de reamostragem e os genes são ordenados por ordem de seleção. Os dados utilizados são oriundos de bases de dados públicas, e o algoritmo proposto é comparado com algoritmos existentes no estado da arte.

Palavras-chave: Seleção de genes de *Microarrays*, Importância de genes, Seleção de *features*.

Abstract

The DNA microarray for gene sequencing allows the screening of thousands of genes simultaneously, and with the advent of the next generation sequencing (NGS) technology, the number of genes available for analysis are much larger than before, where in NGS the generated data can get into the order of terabytes (Tb).

In this context, the gene selection and gene importance evaluation are important tools, since they allow Bioinformatic researchers to focus on promising gene candidates that actively contribute to some disease or anomaly.

These techniques try to recognize patterns that reveal the importance of each gene in the presence of diseases or conditions, achieving a double effect of selecting the most important ones or eliminating the less important ones, which on the other hand decreases the dimensionality of the data. These two effects are important because the selected data leads to less time spent on training predictive models and a more accurate prediction due to the exclusion of redundant data.

However, the gene selection and gene importance evaluation are problematic from the mathematical point of view, since the number of gene/features is much larger than the number of samples/users, making it a problem with many available solutions.

In the literature, there are many available algorithms proposed to solve the problem of gene selection and gene importance evaluation, where some of then are based on statistical methods, and other are based on machine learning approaches [Guyon *et al.*, 2008], [Peng *et al.*, 2005], and these two approches are also called as filter (statistical approaches) and wrapper approaches (machine learning approaches).

This work proposes a hybrid approach, which explores both types of techniques, the filter and wrapper, using the mutual information statistics and several machine learning algorithms, such as the Naive Bayes, Support Vector Machines, Classification Trees and k-Nearest-Neighbor classifiers. To assess the importance of each feature, the proposed method is going to be applied in several boostraped version of data and the genes are going to be ranked according to their frequency of being selected. The proposed approach is going to be applied in public benchmark datasets and compared to current state of art algorithms.

Keywords: *Microarray gene selection*, *Gene importance*, *Feature selection.*

Abreviaturas e Símbolos

Lista de Acrónimos

ADN	Ácido desoxirribonucleico
ARN	Ácido ribonucleico
CC	Coeficiente de Correleção
DT	*Decision Tree*
ICA	*Independent Components Analysis*
IM	Informação mútua
kNN	*k Nearest Neighbor*
LDA	*Linear Discriminant Analysis*
LOOCV	*Leave One Out Cross Validation*
mRMR	*minimum Redundancy Maximum Relevance*
NB	*Naive Bayes*
NGS	*Next Generation Sequencing*
PCA	*Principal Components Analysis*
PDF	*Probabilitie Density Function*
SBFS	*Sequential Backward Floating Selection*
SBS	*Sequential Backward Selection*
SFFS	*Sequential Forward Floating Selection*
SFS	*Sequential Forward Selection*
SVM	*Support Vector Machine*

Símbolos Gerais

L	número de *features* adicionadas por iteração de procura
M	número de amostras
N	número de variáveis
$nSets$	número de *bootstraps*
$p(a)$	Pdf da variável a
$p(a\|b)$	Pdf condicional da variável a dada a variável b
R	número máximo de *features* eliminadas por iteração de procura

S	conjunto de *features* selecionadas
X	conjunto de *features* de entrada
x_j	variável de entrada com índice j
X_k	*Feature* candidata
Y	variável de saída
Z	conjunto de *features* candidatas

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Motivação

Atualmente as investigações da área Biomédica estão numa boa fase da sua evolução, em particular na área da genética. Os métodos de transcrição do Ácido desoxirribonucleico (ADN) para Ácido ribonucleico (RNA), também conhecidos como métodos de sequenciação do genoma humano, tais como os *microarrays* ou o método *Next Generation Sequencing* (NGS), conseguem atualmente fornecer dados que podem chegar à unidade do milhão [Liu and Motoda, 2007]. Estes são usados para estudos sobre a relação entre genes, e a relação entre genes e fatores exteriores. Um desses estudos é a identificação dos genes com influência na presença de células cancerígenas num tecido. A análise destes dados requer métodos avançados capazes de lidar com grandes quantidades de informação e com a sua especificidade. Assim as propostas de recolha de informação são:

- extrair regras ou padrões sobre os dados;
- reduzir a dimensionalidade dos dados;
- agrupar/classificar que acordo com as suas caraterísticas.

Este estudo pode ser comparado à identificação de padrões em imagens, pelo que faz sentido utilizar técnicas oriundas dessa área (*patern recognition* e *machine learning*). Deste modo utilizam-se modelos preditivos ou classificadores com o objetivo de ajudar a recolher essa informação. No entanto, mantém-se o problema da quantidade de dados a serem analisados e processados, ou seja, a tarefa de reduzir a dimensionalidade dos dados.

Aqui se enquadra a técnica de seleção de variáveis que tenta resolver o problema de encontrar um subconjunto das variáveis disponíveis, o mais compacto e simultaneamente informativo possível, de modo a melhorar a eficiência do processamento e armazenamento de dados. A definição de vetores de variáveis agrupados numa matriz, continua a ser a

maneira mais comum e eficiente de representar dados para problemas de classificação e regressão. A informação pode ser armazenada em simples tabelas (com linhas a representar entradas/amostras, e colunas a representar caraterísticas/variáveis/*features*). Cada caraterística resulta de uma medição quantitativa ou qualitativa, ou seja, representa um atributo ou variável.

O interesse em modelos preditivos fez com que após vários anos de esforço paralelo entre investigadores das áreas de *soft-computing*, estatística, aprendizagem máquina, e descoberta de conhecimento, estes concentrassem esforços no problema de seleção de variáveis. Os recentes avanços na área de *soft-sensors* tornaram possível a criação de sistemas de reconhecimento que executam tarefas que não seriam possíveis de executar no passado, e a seleção de variáveis encontra-se no centro destes avanços, tendo aplicação em áreas como a industria médico-farmacêutica, indústria petrolífera, reconhecimento de vozes, biotecnologia, internet, marketing direcionado e outras aplicações.

1.2 Estado da arte

A seleção de *features* consiste numa técnica de pré-processamento de dados. Esta tenta identificar as *features* mais relevantes de um *dataset* e/ou remover as irrelevantes ou redundantes, de modo a melhorar a *performance* dos algoritmos de *machine learning*. Este processo de seleção de variáveis é composto por 2 partes: (i) o método de procura, no caso deste trabalho *plus-l-minus-r* (PLMR); e (ii) a função de custo que indica qual a *feature* a adicionar ou remover preferencialmente em relação a outras, ou qual o melhor subconjunto de *features*.

Assim o processo de seleção de *features* pode ser categorizado de acordo com a relação entre a função de custo utilizada na seleção de *features* e o classificador utilizado, categorizando-se em métodos *wrapper*, *filter*, *embedded*, *hybrid* e *ensemble*. Estes métodos possuem as suas vantagens e desvantagens, sendo que alguns combinam as vantagens de outros de modo a obter melhores resultados [Ang *et al.*, 2016]. Os últimos a serem desenvolvidos foram os métodos *hybrid* e *ensemble*. Estes métodos serão explorados no Capítulo 2, no entanto, na Tabela 1.1 é apresentado um resumo das suas vantagens e desvantagens. O elevado custo computacional envolvido na análise de grandes conjuntos de dados faz com que se dê preferência a métodos computacionalmente menos exigentes, assim se justifica a utilização de métodos que utilizem características do tipo *filter*, como é o caso da utilização de um método do tipo *embedded* para a análise de *microarrays* de genes neste trabalho.

No caso dos métodos do tipo *wrapper* os algoritmos mais populares são: *decision tree* (DT), *support vector machines* (SVMs), *Naive Bayes* (NB), *K-nearest neighbor* (KNN), *artificial neural networks* (ANNs), e *linear discriminant analysis* (LDA).

Em relação aos métodos do tipo *filter* diferentes abordagens foram propostas ao longo do tempo, incluindo medidas baseadas na teoria da informação, medidas de correlação, medidas de distância e medidas de consistência [Xue *et al.*, 2016]. Estes métodos são definidos por

Tabela 1.1: Resumo das vantagens e desvantagens dos diferentes tipos de seleção

Métodos	**Vantagens**	**Desvantagens**
Filter	• Mais rápido do que *Wrapper*; • Escalável; • Independente do classificador; • Menor complexidade computacional do que o *wrapper*;	• Ignora interações entre classificadores; • Ignora a dependência entre *features*;
Wrapper	• Interage com o classificador; • Considera a dependência entre *features*; • Melhor exatidão na *performance* do que o *Filter*;	• Mais propenso a sobreajustamento; • Específico para cada classificador; • Custo computacional;
Embedded	• Interage com o classificador; • Menor complexidade computacional do que o *Wrapper*; • Melhor exatidão na *performance* do que o *Filter*; • Menos propenso a sobreajustamento do que o *Wrapper*; • Considera a dependência entre *features*;	• Específico para cada classificador;
Hybrid	• Melhor exatidão na *performance* do que *Filter*; • Menor complexidade computacional do que o *Wrapper*; • Menos propenso a sobreajustamento do que o *Wrapper*;	• Específico para cada classificador;
Ensemble	• Menos propenso a sobreajustamento; • Mais flexibilidade e robustez com dados de alta dimensionalidade;	• Dificuldade na compreensão de um conjunto de classificadores.

um critério referido como função de custo, *relevance index* ou *scoring* [Duch, 2006]. Este critério tenta medir o potencial de utilidade de uma *feature*, ou um subconjunto destas, num classificador. Uma das abordagens a este critério que tem revelado mais interesse consiste na Informação Mútua. Esta é uma medida de independência estatística, que contém duas características relevantes. A primeira consiste na possibilidade de medir o grau de relação entre variáveis, para qualquer tipo de relação existente entre as variáveis, incluindo relações não lineares. A segunda consiste na sua invariância quando acontecem transformações no espaço das *features* que sejam invertíveis ou diferenciais, tais como, translações, rotações ou qualquer outra transformação que preserve a ordem original dos elementos dos vetores de *features* [Vergara and Estévez, 2014].

Esta abordagem verificou muitos avanços ao longo dos tempos em relação ao trabalho pioneiro de Battiti [Battiti, 1994], pelo que os mais importantes se encontram referenciados na Tabela 1.2. Devido à limitação de espaço, segue-se uma breve descrição dos mais conhecidos. Estes métodos justificam a sua utilização com diferentes argumentos, no entanto, todos se baseiam na ideia de aumentar a relevância de *features* e diminuir a sua redundância.

Em 1994 Battiti propôs o método *Mutual Information Based Feature Selection* (MIFS), no qual um termo corresponde à relevância e o outro à redundância, no entanto, não utiliza nenhum termo condicional.

Em 2005 Peng propôs o método *Maximum-Relevance Minimum-Redundancy* (MRMR). Este é baseado no método de Battiti, apenas com a alteração do termo referente à redundância em que é usada uma média deste. À semelhança do MIFS não é usado nenhum termo condicional.

Em 1999 Yang e Moody propuseram o método *Joint Mutual Information* (JMI), no qual é considerada a informação entre a saída (Y) e a junção da *feature* candidata (X_k) com cada uma das selecionadas. Este recolhe toda a informação relativa à redundância, mas utiliza apenas o valor médio, pelo que revela uma relação próxima com o MRMR.

Em 2002 Kwak e Choi propuseram uma melhoria no MIFS, denominada MIFS-U. Estes afirmam que o seu método se encontra melhor preparado para problemas onde a informação esta distribuída uniformemente no espaço.

Em 2004 Fleuret propôs um método baseado na *Conditional Mutual Information Maximization* (CMIM). Este examina a informação entre cada *feature* selecionada e a saída (Y) condicionada à atual *feature* candidata (X_k), que resulta num termo que pode ter valor positivo ou negativo, em que um valor negativo indica que a informação partilhada entre S_j e Y diminui quando incluída a *feature* candidata X_k. O critério CMIM toma o menor valor quando identifica que X_k interage mal com pelo menos uma das *features* selecionadas.

Em 2009 Brown apresentou uma medida de informação mais geral usando a informação mútua entra cada par de *features* e a informação mútua condicional, efetuando uma aproximação em que usa apenas os termos de informação mútua de primeira ordem.

Tabela 1.2: Critérios baseados na Teoria da Informação existentes na literatura

Critério	**Nome Completo**	**Autores**
MIM	Mutual Information Maximisation	Lewis (1992)
MIFS	Mutual Information Feature Selection	Battiti (1994)
KS	Koller-Sahami metric	Koller and Sahami (1996)
JMI	Joint Mutual Information	Yang and Moody (1999)
MIFS-U	MIFS-'Uniform'	Kwak and Choi (2002)
IF	Informative Fragments	Vidal-Naquet and Ullman (2003)
FCBF	Fast Correlation Based Filter	Yu and Liu (2004)
AMIFS	Adaptive MIFS	Tesmer and Estevez (2004)
CMIM	Conditional Mutual Info Maximisation	Fleuret (2004)
MRMR	Max-Relevance Min-Redundancy	Peng et al. (2005)
ICAP	Interaction Capping	Jakulin (2005)
CIFE	Conditional Infomax Feature Extraction	Lin and Tang (2006)
DISR	Double Input Symmetrical Relevance	Meyer and Bontempi (2006)
MINRED	Minimum Redundancy	Duch (2006)
IGFS	Interaction Gain Feature Selection	El Akadi et al. (2008)
FOU	First-Order Utility	Brown (2009)
SOA	Second Order Approximation	Guo and Nixon (2009)
CMIFS	Conditional MIFS	Cheng et al. (2011)
MGIG	Maximizing Global Information Gain	Shang et al. (2013)
CMICOT	Conditional Mutual Information with Complementary and Opposing Teams	Shishkin, Alexander, et al. (2016)
RelaxMRMR	RelaxMRMR	Vinh, Nguyen Xuan, et al. (2016)

1.3 Objetivos

Este trabalho tem como objetivo o desenvolvimento de um algoritmo de seleção de *features* que tente melhorar a exatidão de classificação de conjuntos de dados genéticos, em relação a alguns exemplos existentes na literatura. Para tal foram usadas técnicas de procura e critérios de escolha já existentes, nomeadamente a informação mútua. Foram também usados classificadores já implementados na plataforma de desenvolvimento do algoritmo (MATLAB).

A avaliação da exatidão do algoritmo foi efetuada através do erro de classificação do modelo utilizado em função do número de *features*, tanto para o conjunto de dados de treino como de teste. Para a avaliação da robustez do algoritmo foi usado 10 *fold cross validation* na classificação dos dados de treino, e vários níveis de *bootstrap*, o que levou à seleção de vários conjuntos de *features*. Finalizando, foi proposto um método de angariação do conjunto

final das *features* a partir dos vários criados pelos *bootstraps*. O algoritmo proposto é do tipo *embedded*, pois pretende conjugar o algoritmo de procura sequencial *plus-l-minus-r* com a seleção de variáveis através da informação mútua utilizando diferentes classificadores.

1.4 Organização do Documento

A dissertação está estruturada em sete capítulos, organizados da seguinte forma:

- No Capítulo 2, é explorada a importância da seleção de genes, abordado o tema da seleção de *features*, as suas características e problemas;
- No Capítulo 3, é abordado o tema da seleção de *features* utilizando técnicas de *Machine Learning*, descrevendo brevemente os classificadores usados e as bases teóricas necessárias à implementação deste trabalho;
- No Capítulo 4, é feita uma introdução à seleção de *features* com base em métodos da Teria da Informação, explicando brevemente as bases teóricas;
- No Capítulo 5, é explicada a metodologia desenvolvida e os diferentes métodos implementados;
- No Capítulo 6, é explicada a implementação do trabalho desenvolvido, e apresentados os resultados obtidos assim como a sua discussão;
- No Capítulo 7, são apresentadas as conclusões acerca do trabalho desenvolvido, assim como possíveis trabalhos futuros.

Capítulo 2

Seleção de Genes

2.1 Sequenciação de genes

A sequenciação de ADN consiste no processo de determinação da sequência das bases azotadas presentes num conjunto de nucleótidos, que formam os ácidos nucleicos (ADN e ARN). Até 1975 este processo era relativamente dispendioso, longo e levantava questões relativamente à sua segurança, devido ao uso de nucleótidos marcados com isótopos radioativos, de forma a medir a sua atividade.

Com o avanço da tecnologia novos processos surgiram, que garantem mais segurança, menos custo e menos tempo despendido. Assim em 1975 Fred Sanger apresentou um novo método usado posteriormente no Projeto Genoma Humano, cujo objetivo era o mapeamento completo do genoma humano, e identificação de todos os nucleótidos que o compõem. O método de Sanger tornou-se padrão por permitir obter sequências de alta fiabilidade para pedaços relativamente longos de ADN (>500 pares de bases) [Sanger and Coulson, 1975].

Mais recentemente surgiu um conjunto de técnicas denominadas NGS, que se caraterizam pelo paralelismo das reações e pela reduzida escala, o que permite a sequenciação de grandes quantidades de ADN de forma rápida e barata [Shendure and Ji, 2008].

2.1.1 *Microarrays* de ADN

Um *microarray* consiste numa matriz de pontos microscópicos dispostos numa lamela de microscópio ou numa base de silicone, cuja quantidade pode chegar à unidade dos milhões, em que estes representam os níveis de expressão de ARNm (ARN mensageiro) de genes. Este processo utiliza uma amostra de referência e uma amostra alvo, onde são aplicados corantes de diferente cor, por norma verde e vermelho. De seguida as amostras são hibridadas com ADN, e usando a técnica de fluoroscopia os níveis de ARNm são medidos, normalmente resultando em valores que variam entre -6 a 6, que representam a o nível de expressão de cada gene da amostra alvo em relação à amostra de referência. [Friedman *et al.*, 2001].

É comum estudar os diferentes níveis de expressão de conjuntos de genes específicos ad-

Tabela 2.1: *Microarray* típico

	Gene 1	Gene 2	$\cdots$	Gene N	Classe
Amostra 1	f_{11}	f_{12}	$\cdots$	f_{1N}	c_a
Amostra 2	f_{21}	f_{22}	$\cdots$	f_{2N}	c_b
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
Amostra M	f_{M1}	f_{M2}	$\cdots$	f_{MN}	c_z

quirindo medições de diferentes amostras submetidas a diferentes condições, e eventualmente guardadas numa única matriz de dados. Assim um *microarray* consiste numa matriz de dados onde se guarda a informação sobre os níveis de expressão dos genes, e tem tipicamente o formato da Tabela 2.1, onde cada coluna representa um gene e cada linha representa uma amostra, que por norma corresponde a um paciente. Cada valor de f_{ij} consiste no valor de expressão medido do gene j na amostra i, onde i=1,...,M (nº de amostras) e j=1,...,N (nº de *features*). A cada amostra previamente classificada, pertence uma classe $c_{a...z}$, que pode ser quantitativa ou qualitativa.

Em tarefas de desenvolvimento de algoritmos com classificadores, o *microarray* ou *dataset* utilizado tem de incluir a classe correspondente a cada amostra, de modo ser possivel utilizar-se uma parte para o treino do modelo, e outra para o teste da sua exatidão, de modo a garantir que este tem a capacidade de classificar amostras não classificadas. Tipicamente um *microarray* tem as seguintes características:

- alta dimensionalidade devido aos número elevado de genes;
- número limitado de amostras, devido à dificuldade e custo de obtenção;
- abundância de genes redundantes;

No caso da mudança do nível de expressão de um gene estar correlacionado com uma mudança no fenótipo, um grande número de outros genes podem ser regulados no sentido oposto, o que representa um desafio para tarefas de classificação. As teorias de aprendizagem computacional sugerem que o espaço de procura é exponencialmente crescente pelo valor de N e o número de amostras necessárias para uma aprendizagem fiável sobre certos fenótipos é na escala de $O(2^N)$. No entanto, a regra prática da estatística (M = 10 x N) é praticamente impraticável para dados de *microarrays*. A conjugação de um número limitado de amostras, com um conjunto elevado de genes leva à possibilidade de formação de demasiadas hipóteses estatísticas relevantes e igualmente válidas na interpretação de um *dataset*. Deste modo a seleção de um pequeno número de genes que seja representativo é essencial para o sucesso da classificação.

2.2 Seleção de Genes

Do ponto de vista prático, a seleção de um subconjunto dentro dos genes disponíveis, ajuda a identificar os genes mais relevantes na causa ou consequência de uma doença. Um conjunto de genes compacto é também desejável por parte dos biologistas devido ao custo elevado associado ao acompanhamento biológico ou validação clínica de genes selecionados [Liu and Motoda, 2007].

O problema de seleção de genes pretende determinar o modo como os genes se influenciam e interagem entre si, com o meio exterior e com os diferentes tipos de células existentes. Parte desse estudo pretende determinar a relação entre a presença de genes específicos e certas doenças, de modo a identificar a propensão de uma pessoa vir a desenvolver uma certa doença.

Esta é uma técnica comparada ao estudo de reconhecimento de padrões em imagens, em que se verificam padrões em certos locais. Assim como no caso das imagens, é necessário treinar o classificador, usando genes conhecidos *a priori* como detentores de certa doença, para posteriormente se tentar prever o mesmo resultado para genes não classificados, em relação à detenção da doença. Isto permite identificar a propensão da pessoa vir a desenvolver uma certa doença. No entanto, há casos de doenças hereditárias que se revelam difíceis de reconhecer devido a factores como um número reduzido de pacientes (amostras), influência de fatores não genéticos (ex: ambiente e fatores epigenéticos), ou ainda o facto da correlação da existência da doença se verificar apenas na combinação de muitos locais, ou seja, muitos genes tornando difícil a sua identificação ou estudo.

2.3 Seleção de Variáveis

O problema de seleção de genes parte do problema geral de redução da dimensionalidade, cujo objetivo é selecionar o melhor subconjunto (S) de *features* dentro do conjunto disponível (X).

$$S \subseteq X \tag{2.1}$$

Este é comum a áreas como a Visão por Computador e a *Big Data Analysis*, onde são utilizados grandes quantidades de dados a ser analisados, pelo que várias abordagens do ponto de vista de *data mining* são propostas (e.g. regras de associação, seleção de variáveis, *clustering*, modelos preditivos). Neste trabalho foca-se a atenção na seleção de variáveis, que no caso em estudo são genes, no entanto, as técnicas utilizadas podem ser aplicadas a qualquer *dataset* que cumpra a especificações de cada técnica, o que faz com sejam de certo modo genéricas.

A seleção de variáveis parte do princípio que nem todas as variáveis presentes nas amostras determinam a classe a que pertencem, ou seja, que existem variáveis redundantes. Isto leva a que a seleção das mais relevantes se torne um passo importante para:

- diminuir a dimensão do conjunto de dados de entrada;
- remover variáveis ruidosas;
- consequentemente diminuir o custo computacional do treino e teste de classificadores com esses dados;
- detetar quais as variáveis mais relevantes para a determinação da classe a que pertencem;
- melhorar a robustez e exatidão de classificação.

Assim enquadra-se como uma etapa no conjunto de técnicas de *Data Mining*. Este conjunto de técnicas tem como objetivo a procura de padrões e dependências estatísticas, ou seja, a extração de informação em grandes conjuntos de dados, através do seu tratamento estatístico. Verifica-se uma definição semelhante para o termo *Machine Learning*, pois, embora tenham origens diferentes, representam hoje em dia processos semelhantes com o mesmo objetivo. O número de técnicas de *Data Mining* usada num conjunto de dados depende do que se conhece e pode admitir sobre os dados, podendo se definir as seguintes etapas ou técnicas de *Data Mining*, não usadas necessariamente nesta ordem:

- *Data cleaning* – Remoção de dados inconsistentes ou irregulares;
- *Data integration* – Combinação de dados de várias fontes;
- *Data selection* – Selecionar apenas os dados que se sabe que serão relevantes;
- *Data transformation* – Transformar os dados para permitir a aplicação de outras técnicas ou comparação entre dados (ex: normalização);
- *Data mining* – procura de padrões nos dados;
- *Pattern evaluation* – Identificação de padrões realmente interessantes ou informativos;
- *Knowledge representation* – Representação da informação adquirida para outros utilizadores.

Assim se verifica que a seleção de *features* se enquadra numa etapa inicial em que posteriormente podem ser aplicados diferentes métodos. Deste modo a seleção de *features* pode ser realizada de modo direcionado para os processos aplicados posteriormente, como classificação ou *clustering*. Por exemplo, a seleção de *features* supervisionada é guiada pelo objetivo de obter do modelo preditivo o maior grau de exatidão de predição possível.

Todos os métodos de seleção de *features* são constituídos por dois componentes principais [Bishop, 1995]. O primeiro componente é o critério de avaliação da qualidade do subconjunto, também chamado de função de custo, que define se um subconjunto é melhor que outro ou qual a *feature* a adicionar ou eliminar. O segundo componente é o método de procura dentro

de um conjunto de *features* candidatas. Assim o critério de seleção pode ser categorizado em três diferentes tipos: métodos do tipo *filter*, *wrapper*, e *embedded* [Kohavi and John, 1997] [Guyon *et al.*, 2008]. Deste modo a relação entre a exatidão do classificador e o subconjunto de *features* selecionado pode ser avaliado independentemente ou em função do classificador.

Tendo o conjunto de dados de entrada D, caracterizado por ter M amostras, N *features* $X = \{x_i, i = 1, ..., N\}$, e as classificações Y, o problema de de seleção de *features* é definido como o problema de procura no espaço de observação N-dimensional, R^N, de um subespaço de m *features*, R^n, que caracterize 'otimamente' Y. Daqui se verifica que o número total de subespaços é 2^N, e que o número de subespaços com dimensão igual ou inferior a n é $\sum_{i=1}^{n} \binom{i}{N}$. Assim torna-se difícil procurar o melhor subespaço exaustivamente devido ao custo temporal/computacional elevado, mesmo no caso do número de *features* total ser moderado. Alternativamente foram desenvolvidos várias abordagens práticas mais simples, com o objetivo de limitar o custo e a complexidade computacional, como a procura sequencial, como a procura da melhor *feature* individual, a procura sequencial progressiva, ou a procura sequencial progressiva flutuante.

A condição de caracterização 'ótima' significa por norma erro de classificação mínimo, no entanto, uma seleção direcionada apenas para este objetivo (do tipo *wrapper*) levanta questões sobre o sobreajustamento do modelo aos dados de treino, o que pode levar a pouca robustez perante dados desconhecidos. Desta forma as medidas do tipo *filter* tentam cumprir a condição de caracterização procurando o subconjunto R^n que tenha máxima dependência com a classificação Y. O critério de máxima dependência pode ser difícil de calcular pelo que foram definidas abordagens alternativas como a máxima relevância.

2.3.1 Métodos de seleção - *Filter* e *Wrapper*

A seleção de um subconjunto de variáveis pode ser dividida em duas diferentes categorias principais, que são diferenciadas pelo facto do critério de seleção incluir o modelo preditor (ou classificador), ou não, e que podem ser usadas em conjunto formando uma nova categoria (*embedded*). Se o critério é independente do classificador, caracteriza-se como sendo do tipo *filter* ou filtro estatístico, e significa que a seleção é feita *a priori* do treino, pois baseia-se apenas na informação estatística dos dados, como a relevância e a redundância. Caso contrário é denominado de *embedded* quando depende do classificador usado e também usa informação estatística, e *wrapper* quando o critério de seleção é baseado apenas no comportamento do classificador utilizado quando sujeito aos subconjuntos selecionados. Daqui se verifica que no caso dos *wrappers* o subconjunto selecionado vai ser direcionado para classificador utilizado.

Os métodos do tipo *filter* usam métodos estatísticos (e.g. coeficiente de correlação (CC), informação mútua (IM)) para quantificar a qualidade de um subconjunto, sendo independentes do classificador usado. Por outro lado, os métodos do tipo *wrapper* usam critérios como o erro médio quadrático (*mean squared error*), o critério de informação de Akaike, as medidas

estatísticas de Mallows, ou a exatidão de classificação. O outro tipo de métodos, *embedded*, combina os dos dois anteriores, usando características dos classificadores para definir uma função de custo. Os métodos do tipo *wrapper* e *filter* serão explorados posteriormente nos Capítulos 3 e 4, respetivamente, pelo que se segue uma breve explicação dos restantes tipos.

2.3.1.1 *Embedded*

Os algoritmos de seleção do tipo *embedded* formam uma categoria de seletores de *features* onde a seleção está embutida no processo de forma a que depende do modelo de aprendizagem. As suas características são similares aos algoritmos do tipo *wrapper*, pelo que por vezes a sua distinção se torna confusa.

No entanto, a principal diferença dos métodos do tipo *embedded*, em relação aos *wrapper*, é que a sua dependência em relação a um modelo preditivo específico não permite a sua implementação em combinação com outros modelos [Souza, 2014].

São muito parecidos aos *wrappers* mas prosseguem mais eficientemente para um subconjunto através da otimização de uma função objetivo com duas partes, a penalização para os maiores subconjuntos e a utilização de um termo relativo à qualidade da relação do subconjunto com o classificador.

2.3.1.2 *Hybrid*

Os métodos do tipo híbrido são caracterizados pela combinação de diferentes métodos para realizar a seleção de *features*.

2.3.1.3 *Ensemble learning*

O conceito de aprendizagem conjunta (*Ensemble Learning*) parte do princípio que a combinação dos resultados de vários modelos representa um melhor resultado do que apenas de um modelo. Este princípio pode ser aplicado à seleção de *features*, em que se destacam dois diferentes tipos de abordagem [Seijo-Pardo *et al.*, 2017]:

- homogêneo - consiste em usar o mesmo método de seleção em diferentes dados de treino distribuindo o *dataset* por diferentes nós;
- heterogêneo - consiste em usar diferentes métodos de seleção de *features* no mesmo conjunto de dados de treino.

Ambas as abordagens baseiam-se na combinação do *ranking* das *features* selecionadas, que por sua vez se encontram ordenadas. O resultado final previsto resulta da combinação dos resultados de todos os modelos usados anteriormente, usando por exemplo uma média ponderada. No entanto, este tipo de processos é pouco atrativo devido à sua exigência computacional, em que é necessário processar e armazenar vários modelos [Souza, 2014].

2.3.2 Problemas - Dimensionalidade, *Bias* e *Overfitting*

2.3.2.1 Dimensionalidade

O problema da dimensionalidade de um *dataset* revela-se como uma das razões para a seleção de variáveis, de modo a reduzir a dimensionalidade de forma informativa, melhorando a *performance* de classificação quer em termos de exatidão, quer de custo computacional.

No entanto, este também se revela um problema da seleção de variáveis, denominado *curse of dimensionality* [Bellman, 1961]. Este ocorre apenas em espaços de alta dimensionalidade, ou seja, quando o número de amostras necessário para representar os dados de entrada cresce exponencialmente com o número de variáveis existentes. Isto faz com que por vezes não seja possível estimar uma PDF (*Probability Density Function*) dos dados de entrada, ou que, a exigência computacional seja muito elevada.

Este problema é principalmente verificado em *datasets* do tipo *microarray*, pois estes contêm por norma um vasto número de expressão de genes e um número limitado de amostras, o que leva ao problema seguinte.

2.3.2.2 *Overfitting*

Um dos maiores obstáculos na análise de dados de alta dimensionalidade como os *microarrays* tem a haver com o risco de *overfitting*. O facto de se possuir poucas amostras e muitas variáveis leva a que os *datasets* usados sejam pouco genéricos, o que por suas vez leva a que os modelos desenvolvidos apresentem um sobreajustamento aos dados de treino pouca generalidade, e portanto pouca robustez de previsão em dados desconhecidos.

2.3.2.3 *Bias*/Resultados tendenciosos

Outro obstáculo da análise de dados de alta dimensionalidade prende-se com a tendenciosidade dos resultados. Esta pode acontecer caso não haja uma separação bem delineada dos dados de treino e de teste, o que nem sempre é possível devido ao número reduzido de amostras. O cenário ideal seria o de ter um conjunto de amostras genérico para cada fase de desenvolvimento do modelo, i.e., um conjunto para seleção de variáveis, um conjunto para treino do modelo, e um conjunto para teste de *performance* do modelo. Na prática verifica-se que apenas a separação em conjunto de treino e de teste é quase sempre cumprida, isto deve-se ao facto da existência de métodos *hybrid* e *embedded* em que a fase de treino e seleção estão relacionadas.

2.4 Seleção e Validação do modelo

2.4.1 *Error rate*

A validação ou generalização de um modelo de predição relaciona-se com a sua capacidade de predição quando aplicado em dados independentes, de teste, ou seja, dados não utilizados no treino e desconhecidos pelo modelo ate então, o que nos permite obter um erro de classificação, usado como medida da qualidade do modelo. No entanto, aplicando validação cruzada nos dados de treino já é possível obter uma medida razoável da exatidão do modelo.

Assim, para validar a *performance* de um classificador deve-se começar por garantir que os dados de teste não são utilizados na seleção de variáveis nem no treino do classificador. No caso do conjunto de dados ser muito pequeno e impossibilitar esta separação, verifica-se que vai existir classificação tendenciosa pelo que devem ser utilizados métodos de redução desta. De qualquer das formas estes métodos podem ser utilizados para dar mais garantias sobre a *performance* do classificador.

A etapa de verificação da *performance* torna-se muito importante, sendo usada em duas ocasiões: seleção do modelo (escolha do modelo com melhor *performance* quando na disposição de vários); e avaliação do modelo final (estimação do erro generalizado de predição) em dados de teste independentes.

Daqui se verifica que antes de qualquer processamento ou utilização dos dados deve-se ter em conta a situação de se possuir apenas um conjunto de dados para a construção do classificador, isto porque existem duas etapas para as quais são necessários dados diferentes. Estas são a etapa de construção ou treino do classificador, em que se usam dados e a sua respetiva classificação para a construção do classificador, e a etapa de teste, em que se usa o classificador para prever a classificação de dados não classificados, de modo a se poder comparar a classificação obtida com a classificação esperada, e assim obter informação sobre a avaliação do comportamento do classificador.

2.4.2 Prevenção de resultados tendenciosos

Os resultados tendenciosos têm origem na utilização de apenas um conjunto de dados para várias etapas de seleção de variáveis. Intuitivamente se percebe que se for utilizado o mesmo conjunto de dados para treino e para teste, os resultados serão aparentemente melhores do que se utilizar um conjunto de dados para treino e outro para teste.

O método mais comum e direto de diminuir classificações tendenciosas é a separação dos dados para treino e teste, selecionando as amostras de ambos aleatoriamente. No entanto, para não obter um erro de predição demasiado otimista pode-se tentar corrigir este através de métodos como a validação cruzada e o *bootstrap*, externamente ao processo de seleção de variáveis. Estes métodos devem ser aplicados principalmente quando os conjunto de dados disponível é pequeno, de modo a prevenir o *overfiting* [Ambroise and McLachlan, 2002].

2.4.2.1 Validação Cruzada *Cross-validation*

A validação cruzada consiste em dividir o conjunto de dados em questão em M subconjuntos mutuamente exclusivos, utilizando-se uns para treino do classificador e outros para teste. Existem diversas formas de dividir o conjunto de dados sendo as mais utilizadas as seguintes:

2.4.2.1.1 ***Holdout*** Neste método divide-se o conjunto de dados em dois subconjuntos, um para treino e outro para teste. A divisão pode ser feita em partes iguais ou não, dependendo da quantidade de dados disponível.

2.4.2.1.2 ***k-fold*** O método *k-fold* consiste em dividir o conjunto em k subconjuntos do mesmo tamanho, e a partir dai efetuar k vezes o cálculo da exatidão de modo a testar cada um dos subconjuntos como dados de teste e os restantes como dados de treino.

2.4.2.1.3 **LOOCV** O método *Leave One Out Cross Validation* representa um caso específico do *k-fold* em que se utiliza k igual ao número total de amostras M, efetuando-se o cálculo para cada uma das amostras como dado de teste e as restantes como dados de treino. Deste modo teria que se treinar e testar o modelo M vezes o que pode representar um custo computacional muito elevado.

2.4.2.2 *Bootstrap*

A técnica de *bootstrap* apresenta-se como uma técnica de *ensemble learning*, pois a sua aplicação resulta em diferentes conjuntos de *features* selecionadas, que necessitam de algum tipo de combinação de modo a se obter o conjunto de *features* selecionadas final. Assim representa uma mais-valia na verificação da *performance* do modelo e consequentemente na sua validação.

A ideia base da técnica do *bootstrap* consiste em inferir uma população através de um conjunto de amostras. A população obtém-se replicando aleatoriamente um conjunto de amostras, ou seja, através da reamostragem aleatória [Efron, 1992] [Friedman *et al.*, 2001]. A criação dos vários conjuntos de dados a partir de um é feita da seguinte maneira: supondo que temos um conjunto de dados original X com M amostras, $X = \{x_1, \ldots, x_M\}$, cada um dos novos conjuntos de dados X_B é criado através da escolha aleatória e com repetição de N dados de X, ou seja, da recolha de M dados de X efetuando reposição. Deste modo há a possibilidade de existirem dados repetidos em X_B, e também dados de X não presentes em X_B. A avaliação do modelo é feita para cada um dos conjuntos de dados criados, o que permite adquirir dados sobre a precisão do modelo como a variância, intervalos de confiança e tendências.

2.4.2.2.1 ***Bagging*** Obtendo vários modelos a partir dos diferentes *datasets* de *bootstrap*, torna-se necessário agregar as predições obtidas e/ou as variáveis selecionadas (Bootstrap ag-

gregating). No caso da classificação o método de agregação preferencial é o voto maioritário. Enquanto que no caso da validação cruzada, o método preferencial é da média das exatidões obtidas. No caso da seleção de variáveis, considera-se que cada conjunto selecionado se apresenta numa ordem, pelo que por norma esta é utilizada para agregar os diferentes conjuntos. No entanto, podem ser aplicadas outras métricas como a frequência de seleção.

2.4.3 Interpretabilidade do Modelo

A interpretabilidade de um modelo preditivo está relacionada com a facilidade de identificação por parte do utilizador dos critérios usados e decisões tomadas por um modelo na predição de dados. Assim se percebe que o modelo preditivo mais interpretável é a árvore de decisão, sendo considerado um modelo do tipo caixa branca, devido à possibilidade de verificação do critério usado em cada nó, ou mesmo de visualização no caso de se construir o diagrama representativo. Por outro lado, todos os classificadores não lineares são considerados não interpretáveis, pois funcionam como caixa preta, desconhecendo-se exatamente os critérios usados em cada decisão.

2.4.4 Flexibilidade

O critério da flexibilidade de um modelo apresenta-se como o compromisso entre *overfitting* do modelo e a sua variância. Isto significa que quanto maior for a flexibilidade do modelo melhor será a capacidade que este possui para se ajustar aos dados de teste, no entanto, esta variância elevada pode levar a previsões erradas. Por outro lado, os modelos mais complexos tendem a apresentar flexibilidade mais limitada, ocorrendo um risco maior de *overfitting*.

Capítulo 3

Técnicas de *Machine Learning*

Como já foi explicado, o objetivo de selecionar variáveis é reduzir a dimensionalidade, sendo que para chegar a esse objetivo são adquiridas e usadas informações sobre as *features* mais relevantes. Esta etapa é por norma realizada também com o intuito de posteriormente fornecer os dados das variáveis selecionadas a um modelo preditivo ou classificador, de modo a prever classificações de amostras não classificadas, no entanto, a seleção de variáveis pode utilizar ou mesmo basear-se no comportamento dos classificadores.

A classificação consiste numa técnica de *machine learning* do tipo *supervised learning*, cujo objetivo é prever o valor ou classe de uma amostra após serem fornecidos dados sobre a amostra. Por outro lado, nas técnicas de *unsupervised learning* não há nenhuma tentativa de previsão, assim o objetivo é juntar ou organizar amostras em função das associações e padrões nos dados sobre a amostra.

A palavra supervisionado (*supervised*) justifica-se devido ao facto que durante a fase de treino, a previamente conhecida classe das amostras direcionar o processo de aprendizagem. Por outro lado, as técnicas não supervisionadas *unsupervised* implicam a inexistência de classificação nas amostras de treino. A seleção de *features* com técnicas deste tipo não faz parte dos objetivos deste trabalho, pelo que não serão discutidas em detalhe.

3.1 *Supervised Learning*

Neste tipo de aprendizagem os dois objetivos são a classificação, que pretende atribuir aos padrões nos dados de entrada diferentes categorias (classes), ou a regressão, que pretende prever um valor real associado aos dados de entrada. Ou seja, as técnicas de classificação pretendem prever respostas discretas (classes), que podem ser quantitativas ou qualitativas, enquanto que as técnicas de regressão pretendem prever respostas contínuas ou valores.

Do ponto de vista prático um algoritmo de aprendizagem supervisionada corresponde a um modelo matemático, que de acordo com os dados fornecidos, cria uma função que se aproxime o melhor possível aos dados introduzidos, de modo produzir saídas semelhantes ás dos dados de treino. Assim, na fase de treino, estes têm a propriedade de alterar a sua

função (ou relação entre os dados de entrada e saída) de acordo com a diferença entre a saída gerada e a saída esperada. A este processo chama-se aprendizagem por exemplos. No fim do treino espera-se que as saídas geradas e a saídas esperadas estejam suficientemente próximas de modo a se considerar o modelo útil para ser utilizado em quaisquer dados de entrada encontrados na prática.

Verifica-se assim que no caso da saída y poder apenas tomar valores $\{0,1\}$ se trata de um problema de classificação binária. No caso de y poder tomar mais de 3 valores diferentes (considerando apenas valores/números naturais) se trata de um problema de classificação multiclasse. E ainda no caso de y poder tomar qualquer valor real se trata de um problema de regressão.

3.1.1 Classificação

3.1.1.1 Naive Bayes

Este classificador denomina-se de *naive* ou ingénuo por assumir que a contribuição de uma *feature* para a classificação de uma amostra é independente de todas as outras *features*. Apesar de esta assunção não ser sempre verdade, este classificador demonstra uma *performance* relativamente boa a efetuar classificações, inclusive em dados multiclasse. Assim este classificador assenta no Teorema de Bayes, que é definido da seguinte forma:

$$p(c_j \mid d) = \frac{p(d \mid c_j)\, p(c_j)}{p(d)}$$

- $p(c_j \mid d)$ = probabilidade *a posteriori* ou condicionada do evento c_j condicional a d;
- $p(d \mid c_j)$ = probabilidade *a posteriori* ou condicionada do evento d condicional a c_j;
- $p(c_j)$ = probabilidade *a priori* do evento c_j;
- $p(d)$ = probabilidade *a priori* do evento d;

O classificador combina este modelo com uma regra de decisão, de modo a escolher a hipótese mais provável de uma nova amostra $D = d_1, d_2, ..., d_n$ pertencer a uma classe c_j da seguinte forma:

$$classify(D) = \arg\max p(C = j) \prod_{i=1}^{n} p(D = d_i | C = c_j) \tag{3.1}$$

As vantagens deste classificador são a rapidez, que depende apenas do número de classes; a facilidade de lidar com dados desconhecidos; a facilidade de combinar variáveis contínuas com variáveis discretas; e a *performance* prática, que mesmo quando se assume independências irrealistas entre as variáveis apresenta bons resultados.

Este classificador foi implementado no trabalho desenvolvido usando a função *fitcnb(X,Y)* disponível no MATLAB.

3.1.1.2 SVM

O classificador *Support Vector Machine* tenta encontrar a fronteira de decisão linear (hiperplano) que separa todos os pontos de uma classe dos pontos de outras classes. De todos os hiperplanos possíveis, o melhor que este classificador tenta encontrar é o que tem a maior margem entre as classes que separa. No entanto, no caso dos dados não permitirem uma separação linear, é usada uma função de perda que penaliza os pontos no lado errado do hiperplano. Estes podem ainda usar transformações ao nível do kernel, para transformar dados não separáveis linearmente em dados de dimensionalidade superior onde é possível encontrar uma fronteira linear.

Este classificador foi implementado no trabalho desenvolvido usando a função *fitcsvm(X, Y)* disponível no MATLAB.

3.1.1.3 Árvore de Decisão

Os classificadores de árvore de decisão consistem em dividir recursivamente os dados em regiões que contém apenas uma classe. Visualmente representa-se através de um diagrama com estrutura de árvore em que os últimos nós ou folhas representam a classe atribuída e o último ramo representa o teste efetuado para a escolha da classe.

Este é um classificador popular pela facilidade de interpretação dos resultados, ou seja, análise visual de um diagrama de árvore. Tem também vantagens como: não necessitar de preparação dos dados como a normalização; consegue lidar com dados numéricos e categóricos; a facilidade em lidar com problemas de classificação multi-classe; a rapidez de construção e predição; o modelo é do tipo caixa-branca em que cada decisão pode ser analisada, contrariamente os classificadores não lineares são do tipo caixa preta, o que pode levar a maior dificuldade na interpretação dos dados; possibilita a validação do modelo usando testes estatísticos; no entanto, requer conhecimento sobre os dados utilizados. Para além disso, permite simplificar o modelo efetuando uma "poda" na árvore, em que se seleciona o número de ramos e um peso correspondente.

Este classificador foi implementado no trabalho desenvolvido usando a função *fitctree(X, Y)* disponível no MATLAB, sendo referido neste documento como TREE.

3.1.1.4 KNN

O classificador k-vizinhos-mais-próximos (*K Nearest Neighbors*), atribui uma classe a um objeto de acordo com a classe, já conhecida dos dados de treino, com mais frequência nos seus k vizinhos mais próximos. Assim k é um hiperparâmetro escolhido pelo utilizador, em que quanto mais alto for o seu valor, menor será a influência de amostras ruidosas na atribuição da classe. Na classificação binária também é costume utilizar-se valores de k ímpar de modo a evitar empates na decisão. Os desafios deste classificador são a rapidez da predição, a escolha do hiperparâmetro k, a escolha da função de cálculo das distâncias, como

a distância euclideana e a distância de Mahalanobis, e os pesos em função das dimensões.

Neste trabalho foi escolhido o valor de 7 vizinhos, tendo sido implementado no trabalho desenvolvido usando a função *fitcknn(X,Y,'NumNeighbors',7)* disponível no MATLAB, em que a distância utilizada por defeito é a distância euclideana.

3.2 *Wrapper*

Os métodos do tipo *wrapper* usam os modelos preditivos como caixa preta, e a sua *performance* como função de custo para avaliar os subconjuntos. Deste modo o problema de seleção assenta na escolha de um modelo preditivo adequado e no tipo de procura. Visto que o problema da procura do melhor subconjunto dentro de todos os possíveis representa uma dificuldade, foram desenvolvidas técnicas de forma a tentar encontrar subconjuntos quase ótimos através de métodos sequenciais e heurísticos. Os algoritmos de seleção sequencial começam com um conjunto de *features* selecionadas vazio (ou cheio, com todas as *features* disponíveis) e posteriormente adicionam (ou removem) *features* até se obter o melhor resultado da função de custo ou se obter o número de *features* pretendido. Os métodos de procura heurísticos avaliam diferentes subconjuntos para otimizar a função objetivo. Os diferentes subconjuntos são gerados procurando no conjunto de dados de entrada ou através de tentativas de solução do problema de otimização.

No caso deste trabalho os métodos de seleção do tipo *wrapper* baseiam o seu critério de seleção na *performance* do classificador. Assim a aplicação deste método resultou na seleção de *features* com base no critério de exatidão de classificação. No entanto, como já foi explicado a exatidão resultante de classificações em que são usadas *features* baseadas nesse mesmo critério pode revelar-se tendenciosa. Isto levou à utilização do método de validação cruzada, o que resultou na utilização da média das exatidões obtidas na aplicação da validação cruzada como critério de seleção.

3.3 Procura Sequencial

Esta técnica é usada com vista a selecionar um subgrupo razoavelmente bom, isto é, é considerada um método subóptimo de obter um subgrupo. A seleção sequencial de variáveis ou seleção 'gulosa' (*greedy*), tem como base a tentativa de melhorar o subconjunto através iterações que verificam se adicionando ou eliminando variáveis ao subconjunto a sua *performance* tende a melhorar.

3.3.1 Procura Sequencial Progressiva - SFS *Sequential Forward Selection*

Também referido como *Sequential growing* este método começa com o subgrupo de variáveis selecionadas vazio, e adicionando variáveis após cada etapa [Whitney, 1971]. No primeiro passo cada uma das variáveis disponíveis é adicionada individualmente, e o subgrupo é avaliado. No final efetua-se a adição da variável que levou o subgrupo à melhor *performance*. O algoritmo repete estes passos até se atingir um número de variáveis selecionadas pré-especificado ou até não se verificarem mais melhoramentos na *performance* do subgrupo. Começando com o conjunto de variáveis selecionadas vazio.

Algorithm 3.1 Algoritmo SFS genérico

1: *Input: Conjunto de variáveis de entrada X e a saída Y;*
2: *Output: Conjunto das melhores features S;*
3: *Inicialização: $S = \emptyset$; $Z = X$; $k = 0$;*
4: *Encontrar Z_j que minimiza (ou maximiza) o critério $\mathcal{T}(S, Z_{j*}; Y)$:* $Z_j = \arg\min_{Z_{j*} \in Z} \mathcal{T}(S, Z_{j*}; Y)$;
5: *Atualizar $S := S \cup Z_j$;*
6: *Atualizar $Z := Z \setminus Z_j$;*
7: *k:=k+1;*
8: *Repetir de 4 a 7 até algum critério interromper ou até $k = d$.*

3.3.2 Procura Sequencial Regressiva - SBS *Sequential Backward Selection*

Esta técnica foi introduzida como *Sequential Pruning* por Marrill and Green [Marill and Green, 1963], e foi revista como sendo a primeira tentativa de procura de um subgrupo de variáveis.

Começando com todas as variáveis disponíveis, o primeiro passo consiste em considerar a eliminação de cada uma das variáveis individualmente. Os resultados de cada eliminação são comparados usando a função $J(.)$, que avalia a *performance* do subgrupo para cada uma das eliminações consideradas, e no final efetua-se a eliminação da variável que levou a melhores resultados. Estes passos são repetidos até o subgrupo ficar com um número de variáveis definido previamente, ou, até os resultados piorarem demasiado [Guyon *et al.*, 2008].

Tipicamente, o algoritmo SFS é executado mais rapidamente do que o SBS. Isto tem a haver com o facto de um começar com o subconjunto selecionado vazio e o outro cheio. Pois, verifiquemos a situação de ambos procurarem n variáveis num total de m (m>n). No início o algoritmo SFS vai avaliar um subgrupo de variáveis muito mais pequeno do que o SBS. É verdade que no final da procura o SFS tem de avaliar subgrupos quase tão grandes como o

SBS no início, e o SBS tem de avaliar subgrupos mais pequenos, mas já existem tão poucas opções a considerar que o tamanho dos subgrupos já não tem grande significado.

3.3.3 Outras abordagens de Procura Sequencial

Os métodos descritos anteriormente partem do conjunto de variáveis inicial e prosseguem numa única direção para o conjunto final selecionado. Isto significa que após a adição ou remoção de uma variável nunca se considere a possibilidade de anular este passo. Este efeito tem o nome de *nesting effect*. De modo a contrariar este efeito desenvolveu-se a ideia de *backtrack* que consiste em permitir que se elimine *features* do conjunto de selecionadas, e que se adicione *features* anteriormente eliminadas.

Uma implementação desta ideia combina os dois métodos apresentados (SFS e SBS), denominada *plus-l-take-away-r* (ou *plus-l-minus-r* PLMR) [Stearns, 1976]. Este método consiste em aplicar o SFS l vezes seguindo-se a aplicação do SBS r vezes, repetindo estes passos de adição e eliminação até se atingir o número total de *features*. Assim permitindo que *features* anteriormente adicionadas possam ser eliminadas, evitando o *nesting effect*. Este método permite o retrocesso de tamanho fixo definido pelos valores de l e r. Apesar de evitar o problema anterior surge um novo problema: determinar os valores apropriados para r e l de modo a obter bons resultados com um custo computacional aceitável.

Assim foi introduzido um novo conceito de *floating feature search* [Pudil *et al.*, 1994] e dois novos métodos, SFFS (*Sequential Forward Floating Selection*) and SBFS (*Sequential Backward Floating Selection*). Estes métodos "flutuantes" estão ligados ao *plus-l-take-away-r*, no entanto, o número de passos de progressão e de regressão são controlados dinamicamente em vez de serem fixos. Basicamente, no caso da procura sequencial, o algoritmo começa com um conjunto de *features* selecionadas vazio, e a cada iteração a melhor *feature*, i.e., a *feature* com o melhor valor da função de custo, a cumprir um certo critério é selecionada, ou seja, um passo do algoritmo SFS é efetuado, e de seguida é verificada a possibilidade de melhoria no caso de se eliminar uma *feature*, ou seja, é efetuado um passo do algoritmo SBS.

Apesar de nenhum dos métodos do tipo *floating* garantir a seleção do melhor subconjunto de *features*, a sua *performance* demonstra bons resultados quando comparados a outros métodos [Yusta, 2009]. Para alem disso, estes não efetuam tantos passos como o SFS e o SBS, pois, selecionam mais do que uma variável de cada vez, conseguindo um determinado número de variáveis usando um número de passos inferior. No entanto, o crescimento combinacional na complexidade de cada passo ultrapassa a redução no número de passos no que diz respeito ao custo computacional. Este elevado custo é apenas justificado em aplicações nas quais existem grupos de variáveis que sozinhas não apresentam nenhum melhoramento, mas em conjunto possuem informação importante para uma boa *performance* do classificador. No entanto, não garantem que não aconteça o caso de ficarem presos em soluções locais ótimas mesmo que o critério definido seja monotónico e a escala do problema seja reduzida.

Capítulo 4

Métodos baseados na Teoria da Informação

A teoria da informação foi originalmente proposta Shannon em 1948. Esta estuda a quantificação, armazenamento e comunicação da informação, tendo como conceito chave a entropia. Este conceito quantifica a quantidade de incerteza envolvida no valor de uma variável aleatória. Outro conceito muito importante nesta área é o de informação mútua, que se define como a quantidade de incerteza que é reduzida numa dada variável, a partir do conhecimento de outra. Deste modo estes métodos revelam ser importantes para seleção de variáveis a partir de uma abordagem estatística.

4.1 *Filter*

Este tipo de métodos pode ser considerado como *unsupervised learning* ou *supervised learning*, pois, o que fazem muitas vezes é colocar as variáveis segundo um *ranking*. Este é usado como mecanismo principal ou secundário de seleção de variáveis, devido à sua simplicidade, e sucesso empírico verificado. Os métodos estatísticos mais utilizados para determinar o *ranking*, como a correlação e a informação mútua, tentam avaliar dependências entre variáveis (quer seja entre *features* ou entre *features* e a classificação).

Assim filtragem de *features* pretende dispensar aquelas que tenham uma pequena hipótese de serem úteis na análise dos dados, i.e., pretende identificar as mais relevantes e também as mais redundantes para a classificação. Estes métodos são baseados na avaliação da *performance*, usando métricas calculadas diretamente com os dados, sem a interferência do comportamento dos classificadores que iram posteriormente ser usados nos dados após a seleção das *features*. Por norma estes algoritmos são computacionalmente menos dispendiosos do que os do tipo *wrapper* [Guyon *et al.*, 2008].

Em relação ao critério de seleção várias abordagens são possíveis. Talavera [Talavera, 1999] desenvolveu uma versão de um filtro que seleciona *features* em função da sua dependência, afirmando que *features* irrelevantes não dependem das outras. Manoranjan [Dash

et al., 2002] introduziu uma abordagem para o critério de seleção, baseada na entropia da distância entre os pontos representativos dos dados, em que observou que quando os dados estão agrupados, a entropia das distâncias num sub-espaço tem tendência a ser baixa. He, Cai, e Niyogi [He *et al.*, 2006] tomaram a abordagem de selecionar *features* com base no seu *Laplacian Score* ou poder de preservação da localidade, que é baseado na premissa de que dois pontos que estão próximos têm uma probabilidade elevada de pertencer a mesma classe.

4.1.1 *Ranking* - Ordenação

Os métodos do tipo *filter* usam a ordenação de variáveis como o principal critério de seleção. Estes são usados devido à sua simplicidade e sucesso reportado em aplicações práticas. Este tipo de seleção é feita através da ordenação das *features* consoante um critério selecionado e da definição de um limite (relacionado com o critério de ordenação ou com o número de *features* que se pretende selecionar), a partir do qual se define quais eliminar. Os métodos de ordenação são considerados do tipo *filter*, pois, são aplicados antes da classificação, de modo a filtrar as *features* menos relevantes.

Uma propriedade básica de uma *feature* é a sua capacidade de conter informação útil sobre as classes das amostras. Esta propriedade pode ser definida como a relevância de uma *feature*, e tenta medir a sua utilidade em discriminar as diferentes classes. Assim se levanta o problema da definição e medição da relevância. Na literatura existem várias definições e medidas de relevância, destacando-se a seguinte definição: "*A feature can be regarded as irrelevant if it is conditionally independent of the class labels.*" [Law *et al.*, 2004]. Esta definição indica que para uma *feature* ser relevante ela pode ser independente das restantes, no entanto, não pode ser independente das classes das amostras.

Daqui se verifica que as relações entre *features* podem desempenhar um papel importante. No entanto, em aplicações práticas esta distribuição é desconhecida, e apenas medida pela exatidão do classificador. Posto isto, um subconjunto de *features* ótimo pode não ser único, isto porque um classificador pode atingir a mesma exatidão de classificação usando diferentes subconjuntos de *features*.

As vantagens da ordenação de *features* relacionam-se com a leveza computacional e o facto de evitar *overfitting*. Os métodos do tipo *filter* não dependem de modelos preditivos o que evita resultados tendenciosos. Uma das desvantagens tem a haver com o facto de não se discriminar as variáveis em termos de correlação entre estas, o que pode levar à escolha de um subconjunto redundante. Isto leva a que *features* importantes possam ser descartadas, pois, quando sozinhas são pouco informativas, mas quando combinadas com outras tornam o conjunto informativo.

Um dos métodos de ordenação mais simples corresponde coeficiente de correlação de Pearson, definido como:

$$R(i) = \frac{cov(x_i, Y)}{\sqrt{var(x_i) * var(Y)}} \tag{4.1}$$

Onde x_i corresponde à i-ésima variável, Y à variável com as classes, $cov()$ à covariância e $var()$ à variância. No entanto, esta medida de correlação apenas consegue detetar dependências lineares entre a variável e as classes.

4.2 Medidas de Informação

Para introduzir o tema das medidas de informação é necessário rever os conceitos de medidas de informação de Shannon. Para tal são usadas variáveis aleatórias discretas nas equações deste Capítulo, pois, na prática a maioria das variáveis usadas são discretas por natureza ou por quantificação.

As medidas de dependência ou medidas de correlação quantificam a capacidade de prever um valor de uma variável usando o valor de outra variável. O coeficiente de correlação representa uma medida de dependência usada para medir a correlação entre uma *feature* e a classificação. No caso da *feature* X_1 medir uma correlação com a classe Y maior do que a medida de correlação da *feature* X_2 com a classe Y, significa que X_1 é preferida a X_2. Uma variação deste método consiste em determinar a dependência de uma *feature* com outras, cuja medida indica o nível de redundância entre as *features*. Todas as funções de custo baseadas em medidas de dependência podem ser divididas em medidas de distância e medidas de informação.

Dentro do conjunto de medidas de independência entre variáveis aleatórias, a informação mútua é caracterizada pela sua origem na teoria da informação [Friedman *et al.*, 2001]. Em contraste com o coeficiente de correlação linear, esta é sensível a dependências que não se manifestam na covariância. Em relação à IM, esta assume o valor 0 apenas no caso de duas variáveis aleatórias serem independentes. Este valor também se verifica para quantidades baseadas na entropia de Renyi, e estes são por norma mais fáceis de estimar (em particular se a sua ordem for igual ou superior a 2) [Kraskov *et al.*, 2004]. No entanto, a MI tem ligações únicas à entropia de Shannon que lhe conferem vantagens teóricas, mas também é verdade que estimar a MI nem sempre é tarefa fácil.

As medidas de informação típicas determinam a quantidade de informação ganha com uma *feature*. O ganho de informação da *feature* X_1 é calculado através da diferença entre a incerteza anterior e a incerteza posterior à utilização da *feature* X_1. Assim a *feature* X_1 é preferida a X_2 se o seu ganho de informação for superior. Um exemplo destas medidas é a entropia.

4.3 Entropia

Começando pelo conceito de entropia $H(x)$ de uma variável aleatória x, onde $p(x)$ representa a função de densidade de probabilidade da variável x como:

$$H(x) = -\sum_{i=1}^{n} p(x(i))\, log\, p(x(i)) \tag{4.2}$$

O conceito de entropia está diretamente relacionado com a probabilidade de ocorrência de um evento. Esta medida caracteriza a informação (simétrico da entropia) que está contida numa variável, ou seja, a incerteza removida quando revelado o real valor de X.

O conceito seguinte é de entropia conjunta de duas variáveis aleatórias, que se revela similar ao conceito de entropia de uma variável aleatória. Assim a entropia conjunta $H(X,Y)$ de um par de variáveis aleatórias X e Y é definido por:

$$H(X,Y) = -\sum_{x,y} p(x,y)\, log\, p(x,y) \tag{4.3}$$

Por fim, tendo a as variáveis X e Y, o conceito de entropia condicional de Y sabendo X é definido por:

$$H(Y|X) = -\sum_{x,y} p(y|x)\, log\, p(x,y) \tag{4.4}$$

Podendo ser reescrito por:

$$H(Y|X) = \sum_{x} p(x) \left[-\sum_{y} p(y|x)\, log\, p(y|x) \right] \tag{4.5}$$

Sabendo que o somatório interno corresponde à entropia de Y condicionada a um valor fixo de $x \in S_X$, pode-se reescrever da seguinte forma:

$$H(Y|X) = \sum_{x} p(x) H(Y|X = x) \tag{4.6}$$

Onde:

$$H(Y|X = x) = -\sum_{y} p(y|x)\, log\, p(y|x) \tag{4.7}$$

Semelhantemente para $H(Y|X,Z)$ escreve-se:

$$H(Y|X,Z) = \sum_{z} p(z) H(Y|X, Z = z) \tag{4.8}$$

Onde:

$$H(Y|X, Z = z) = -\sum_{x,y} p(x,y|z)\, log\, p(y|x,z) \tag{4.9}$$

Considera-se também as seguintes proposições:

$$H(X,Y) = H(X) + H(Y|X) \tag{4.10}$$

$$H(X,Y) = H(Y) + H(X|Y) \tag{4.11}$$

Esta proposição tem a seguinte interpretação. Considere-se a revelação do resultado de um par de variáveis aleatórias X e Y em dois passos: primeiro o resultado de X e depois o resultado de Y. Então a proposição diz que a quantidade total de incerteza removida em consequência da revelação de ambas as variáveis X e Y é igual à soma da incerteza removida em consequência da revelação de X (incerteza removida no primeiro passo) com a incerteza removida em consequência da revelação de Y após o conhecimento de X (incerteza removida no segundo passo).

4.4 Informação Mútua

A informação mútua entre duas variáveis X e Y é definida da seguinte forma:

$$I(X;Y) = \sum_{x,y} p(x,y)\, log\, \frac{p(x,y)}{p(x)p(y)} \tag{4.12}$$

Verifica-se que a informação mútua entre duas variáveis $I(X;Y)$ é simétrica entre X e Y. Também é possível verificar que a informação mútua entre uma variável aleatória X e ela própria é igual à sua entropia, i.e.:

$$I(X;X) = H(X) \tag{4.13}$$

A entropia de uma variável aleatória X pode ser chamada de informação própria. Verifica-se as seguintes proposições em relação à informação mútua:

$$I(X;Y) = H(X) - H(X|Y) \tag{4.14}$$
$$I(X;Y) = H(Y) - H(Y|X) \tag{4.15}$$

e

$$I(X;Y) = H(X) + H(Y) - H(X,Y) \tag{4.16}$$

Da equação 4.14 interpreta-se $I(X;Y)$ como a redução na incerteza de X quando Y é dado, ou, a quantidade de informação sobre X que Y consegue dar. Visto que $I(X;Y)$ é simétrico em X e Y, pode-se interpretar 4.15 como a quantidade de informação sobre Y dada por X.

As relações entre a entropia, a entropia condicional, a entropia conjunta e a informação mútua entre duas variáveis aleatórias X e Y estão representadas na figura 4.1.

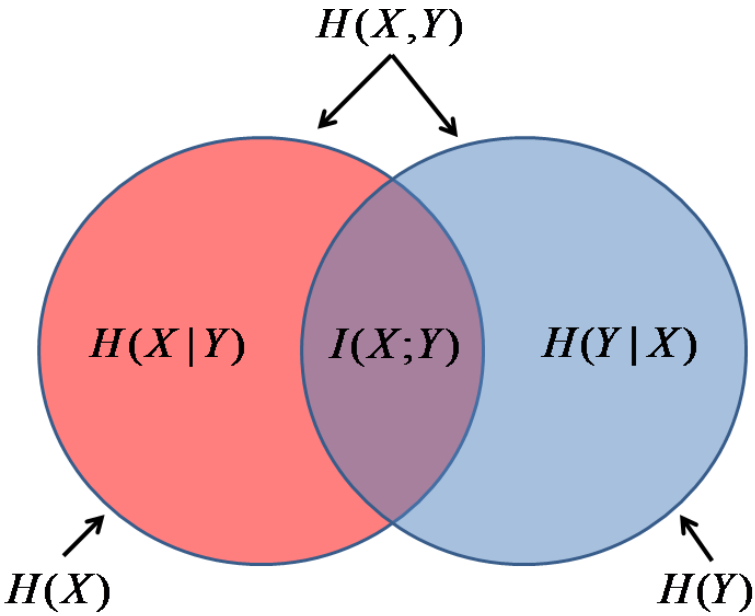

Figura 4.1: Diagrama de Venn ilustrando as relações entre Entropia e IM. Imagem retirada de [Vergara and Estévez, 2014].

É também possível reparar que as relações entre as medidas de informação de Shannon são consistentes com as relações das proposições das equações 4.14, 4.15, 4.10 e 4.11.

Analogamente à entropia, existe uma versão condicionada da informação mútua, denominada informação mútua condicionada. Para as variáveis aleatórias X, Y e Z, a informação mútua entre X e Y condicionada a Z é dada por:

$$I(X;Y|Z) = \sum_{x,y,z} p(x,y,z)\, log\, \frac{p(x,y|z)}{p(x|y)p(y|z)} \tag{4.17}$$

Verifica-se que $I(X;Y|Z)$ é simétrico em X e Y. Analogamente à entropia condicional podemos escrever:

$$I(X;Y|Z) = \sum_{z} p(z) I(X;Y|Z=z) \tag{4.18}$$

onde

$$I(X;Y|Z=z) = \sum_{x,z} p(x,y|z) log \frac{p(x,y|z)}{p(x|z)p(y|z)} \tag{4.19}$$

Similarmente, quando condicionado a duas variáveis escreve-se:

$$I(X;Y|Z,T) = \sum_{t} p(t) I(X;Y|Z,T=t) \tag{4.20}$$

onde

$$I(X;Y|Z,T=t) = \sum_{x,y,z} p(x,y,z|t) log \frac{p(x,y|z,t)}{p(x|z,t)p(y|z,t)} \tag{4.21}$$

Por fim verifica-se que a informação mútua entre uma variável aleatória X e ela própria, condicionada a uma variável aleatória Z é igual à entropia de X condicionada a Z:

$$I(X;X|Z) = H(X|Z) \tag{4.22}$$

Verificando-se as seguintes proposições sobre a informação mútua condicionada:

$$I(X;Y|Z) = H(X|Z) - H(X|Y,Z) \tag{4.23}$$

$$I(X;Y|Z) = H(Y|Z) - H(Y|X,Z) \tag{4.24}$$

Para concluir esta secção demonstra-se que todas as medidas de informação de Shannon são casos especiais de informação mútua condicional. Considerando Φ como sendo uma variável aleatória degenerada, i.e, Φ tem um valor constante com probabilidade 1. Considere-se a informação mútua de $I(X;Y|Z)$. Quando $X = Y$ e $Z = \Phi$, $I(X;Y|Z)$ transforma-se na entropia $H(X)$. Quando $X = Y$, $I(X;Y|Z)$ transforma-se na entropia condicional $H(X|Z)$. Quando $Z = \Phi$, $I(X;Y|Z)$ transforma-se na informação mútua $I(X;Y)$ [Yeung, 2008].

4.4.1 Densidade de probabilidade

Para calcular a informação mútua é necessário conhecer funções de densidade de probabilidade (PDF) das variáveis de entrada e de saída. Este processo revela-se difícil de implementar na prática, pois, o método mais simples de estimar PDFs é o método do histograma, que necessita de um espaço de memória muito grande. Por exemplo, num problema em que se pretende selecionar k *features*, se a saída for composta por K_Y classes e se dividir a j-ésima *feature* em P_j partições de modo a obter o histograma, são necessárias $K_Y \times \prod_{j=1}^{k} P_j$ células para calcular $I(x_j, S; Y)$. Neste caso, para um simples problema de seleção de 10 *features*, são necessárias $K_Y \times 10^{10}$ células de memória, se cada *feature* for dividida em 10 partições. Assim foram propostos métodos alternativos que usam apenas PDFs conjuntas de duas variáveis para calcular IMs [Battiti, 1994].

No trabalho desenvolvido foram usados métodos que permitem a utilização de variáveis continuas, para a estimação de PDFs, no entanto, ambos os métodos [Peng *et al.*, 2005] e [Brown *et al.*, 2012], procedem à discretização das variáveis de entrada de forma a efetuar a estimação através de métodos baseados em histogramas.

4.4.1.1 Estimador de densidade usando *Parzen Window*

O estimador de densidade *Parzen Window* pode ser usado para aproximar a densidade de probabilidade $p(x)$ de um vetor de variáveis aleatórias contínuas X. Este envolve a superposição de uma função de janela normalizada e centrada num conjunto de variáveis aleatórias. Dado um conjunto de n vetores de treino com dimensionalidade d, $D = x_1, x_2, ..., x_n$, a estimação da densidade usando *Parzen Window* é dada por:

$$\hat{p}(x) = \frac{1}{n} \sum_{i=1}^{n} \phi(x - x_i, h) \tag{4.25}$$

Onde $\phi(\cdot)$ representa a função de janela e h o parâmetro de largura. Parzen demonstrou que $\hat{p}(x)$ converge para a verdadeira densidade se $\phi(\cdot)$ e h forem selecionados corretamente.

Requer que a função de janela seja uma função de densidade de valor finito não negativo onde

$$\int \phi(y,h)dy = 1 \tag{4.26}$$

com o parâmetro de largura em função de n, tal que

$$\lim_{n\to\infty} h(n) = 0) \tag{4.27}$$

e

$$\lim_{n\to\infty} nh^d(n) = \infty \tag{4.28}$$

As funções de janela mais usadas são a retangular e a Gaussiana. A Gaussiana é dada por

$$\phi(z,h) = \frac{1}{(2\pi)^{d/2}h^d\left|\sum\right|^{1/2}} exp(-\frac{z^T\sum^{-1}z}{2h^2}) \tag{4.29}$$

onde $\sum$ representa a matriz de covariância de um vetor d dimensional de variáveis aleatórias z [Kwak and Choi, 2002].

A implementação deste estimador não revelou melhorias no cálculo da estimação da densidade de probabilidade, revelando também ser mais dispendioso computacionalmente do que outros métodos, pelo que a sua utilização foi desconsiderada.

4.4.2 Redundância, Relevância e Complementaridade

As abordagens de seleção de variáveis do tipo *filter* são baseadas no conceito de relevância, que será explorado de seguida. Basicamente o problema passa por achar o menor subconjunto de *features* possível que preserve a informação contida no conjunto de todas as *features* em relação à classificação das amostras Y. Este problema é normalmente resolvido encontrando as *features* relevantes e descartando as redundantes e irrelevantes. De seguida irá ser feita uma revisão das diferentes definições de relevância, redundância e complementaridade encontradas na literatura.

4.4.2.0.1 Relevância Intuitivamente, uma *feature* é relevante quando quer individualmente quer em conjunto com outras, revelar conter informação sobre a classificação Y.

Na literatura encontram-se várias definições de relevância, incluindo diferentes níveis de relevância. Kohavi and John [Kohavi and John, 1997] usaram uma base probabilística para definir três níveis de relevância: forte, fraca e irrelevante. *Features* fortemente relevantes fornecem informação única sobre Y, i.e., não podem ser substituídas por outras. *Features* com fraca relevância fornecem informação sobre Y, no entanto, podem ser substituídas por outras sem que se perca informação sobre Y. *Features* irrelevantes não fornecem informação sobre Y, pelo que podem ser dispensadas sem que se perca informação. Uma desvantagem da abordagem probabilística é a necessidade de testar a independência condicional de todos os subconjuntos de *features* possíveis, assim como estimar as funções de densidade de

probabilidade. Alternativamente existe uma definição de relevância baseada na informação mútua. A vantagem desta abordagem é que existem alguns bons métodos para estimar a IM. No entanto estas definições levantam algumas questões ou mesmo desvantagens sumarizadas da seguinte forma:

- Para classificar uma *feature* x_j como relevante, é necessário verificar todos os subconjuntos possíveis de $S\neg x_j$. Assim este método está sujeito a "maldição" da dimensionalidade.
- A definição de *feature* com forte relevância é muito restritiva. Se duas *features* fornecerem informação sobre Y mas forem redundantes então ambas serão descartadas de acordo com este critério.
- A definição de *feature* com fraca relevância não é suficiente para descartar uma *feature* do conjunto de *features* ótimo, sendo necessário discriminar se esta é redundante ou não.

4.4.2.0.2 Redundância Yu e Liu [Yu and Liu, 2004] propuseram outra tipologia de *features* dividindo-as em: com relevância fraca e redundantes; e com relevância fraca e não redundantes. Assim estes autores definiram o conjunto de *features* ótimo como sendo composto por *features* fortemente e fracamente relevantes e não redundantes. O conceito de redundância está associado ao nível de dependência entre duas ou mais *features*. Este nível de dependência entre uma *feature* x_j e o subconjunto $S \subseteq \neg x_j$ pode ser medido usando a informação mútua ($I(x_j, S)$). Esta medida de informação teórica sobre a redundância satisfaz as seguintes propriedades: é simétrica, não linear, não negativa, e não diminui quando se adicionam mais *features*. No entanto, usando esta medida não é possível determinar concretamente com qual das *features* em S é x_j redundante.

4.4.2.0.3 Complementaridade O conceito de complementaridade tem sido redescoberto várias vezes. Recentemente, tornou-se mais relevante devido ao desenvolvimento de técnicas mais eficientes para estimar informação mútua em espaços de dados de alta dimensionalidade. Complementaridade, também conhecida como sinergia, mede o grau de interação entre uma *feature* individual e um subconjunto de *features* S dado Y, através da expressão $I(x_j; S|C)$. Para demonstrar o conceito de complementaridade vamos começar por expandir a multi-informação mútua entre x_j, Y e S. Decompondo a multi-informação mútua nas suas três possibilidade temos:

$$I(x_i; S; Y) = \begin{cases} I(x_i; S|Y) - I(x_i; S) \\ I(x_i; Y|S) - I(x_i; Y) \\ I(S; Y|x_i) - I(S; Y) \end{cases} \tag{4.30}$$

De acordo com a equação 4.30, verifica-se na primeira linha que a multi-informação pode ser expressa como a diferença entre complementaridade $I(x_i; S|Y)$ e redundância $I(x_i; S)$.

Assim um valor positivo da multi-informação significa uma prevalência da complementaridade em relação a redundância. Analisando a segunda linha verifica-se que a expressão resulta em valores positivos quando a informação que x_i contém sobre Y é maior quando interage com o subconjunto S do que quando isso não se verifica. Este efeito denomina-se complementaridade. A terceira linha dá outra perspetiva sobre o efeito de complementaridade. A multi-informação toma valores positivos quando a informação que S contém sobre Y interage com x_i em relação ao caso em que não o faz. Assumindo que o efeito da complementaridade é dominante em relação à redundância a figura 4.2 ilustra um diagrama de Venn com as relações entre complementaridade, redundância e relevância [Vergara and Estévez, 2014].

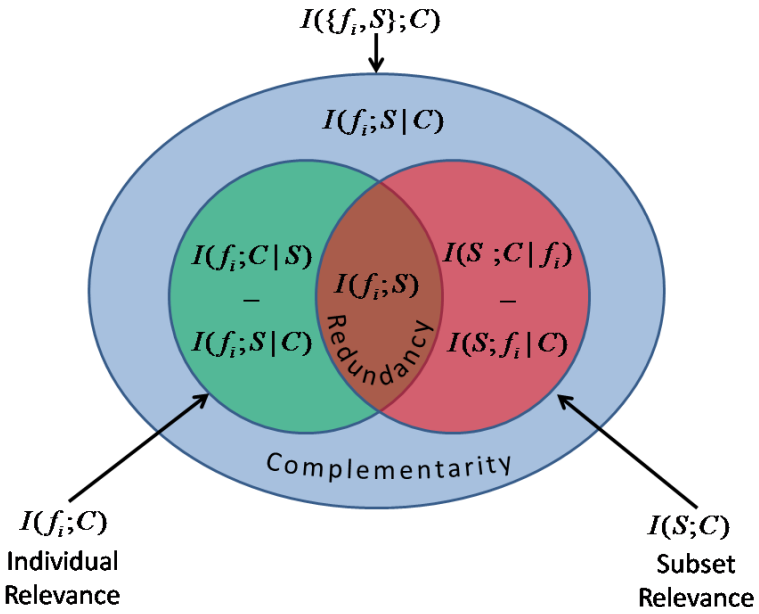

Figura 4.2: Diagrama de Venn ilustrando as relações entre complementaridade, redundância e relevância, assumindo que a multi-informação entre $f_i = x_j$, S e $C = Y$ é positiva. Imagem retirada de [Vergara and Estévez, 2014].

4.5 mRMR

Das várias abordagens de seleção de *features* do tipo *filter* existentes na literatura, a abordagem de Peng é a que mais se destaca na qualidade da sua *performance*, pelo que foi foi implementada neste trabalho de modo a servir de termo comparativo.

Em [Peng *et al.*, 2005] Peng começa por definir que um bom conjunto de *features* pode ser escolhido pelo critério de maximização da dependência estatística baseada na informação mútua. Devido à dificuldade na implementação direta da condição de máxima dependência, em [Peng *et al.*, 2005] derivou-se uma forma equivalente, o mRMR, para seleção de *features* incremental de primeira ordem. Aliada a outros seletores de *features* (e.g., *wrappers*) demonstrou bons resultados na seleção de conjuntos bons e compactos a custo reduzido. Por este

motivo é um dos métodos implementados, de modo comparativo ao algoritmo desenvolvido.

A condição de caracterização ótima significa muitas vezes na implementação prática erro de classificação mínimo. Numa situação *unsupervised* os classificadores não são especificados, pelo que erro mínimo requer dependência estatística máxima entre o subespaço R^m e a variável de saída Y (classes). Esta condição de máxima dependência pode ser abordada de várias formas, sendo que na seleção de variáveis a abordagem mais comum é a relevância máxima, em que as *features* selecionadas são as que apresentam a maior relevância com as classes Y. Por sua vez, a relevância é por norma caracterizada em termos de correlação ou informação mútua, sendo a última a mais comum.

Na abordagem de relevância máxima, para a seleção de uma *feature* é necessário que esta tenha a maior informação mútua com as classes $I(x_i, Y)$, refletindo a maior dependência. Em termos de procura sequencial, a seleção das melhores m *features*, é feita selecionando as primeiras m *features* após a ordenação destas de forma descendente de informação mútua $(I(x_i, Y))$.

No entanto, é reconhecido que na seleção de *features* a combinação das que se revelem individualmente boas, não significa boas *performances* de classificação. Por outras palavras "*the m best features are not the best m features*" [Cover, 1974], ou seja, as m melhores *features* podem não corresponder ao conjunto das melhores m *features*.

Aqui entram as medidas de redução de redundância entre *features* e a seleção com base na redundância mínima. Em [Ding and Peng, 2005], foi proposto um método de seleção de *features* tento em *datasets* contínuos como discretos, combinando as ideias de minimização da redundância e maximização da relevância (mRMR). Posteriormente, em [Peng *et al.*, 2005], os mesmos autores fizeram uma análise teórica demonstrando a equivalência entre o mRMR e a máxima dependência para seleção de *features* de primeira ordem, revelando mais eficiência no método proposto. Nesse trabalho foi feita uma aproximação da máxima dependência à maximização da informação mútua, no entanto, verificou-se a dificuldade em obter uma estimação precisa da função de densidade de probabilidade multivariada para $p(x_1, ..., x_m)$ e para $p(x_1, ..., x_m, Y)$, devido a duas dificuldades verificadas no espaço de alta dimensionalidade: 1) o número de amostras ser insuficiente e 2) a estimação da densidade multivariada por norma necessitar da computação da matriz de covariância, também ela de alta dimensionalidade.

Verificando-se a dificuldade de implementação da seleção de *features* com base na máxima dependência, em [Peng *et al.*, 2005] optou-se pela aproximação de máxima relevância. Deste modo são procuradas *features* que satisfaçam 4.31, que aproxima a dependência entre o conjunto de *features* selecionadas e as classes $(D(S, Y))$, com a média de todos os valores de informação mútua entre cada uma das *features* selecionadas $(x_i \in S)$ e a variável das classes Y.

$$Max\, D(S, Y),\; D = \frac{1}{|S|} \sum_{x_i \in S} I(x_i; Y) \tag{4.31}$$

No entanto, neste modo de seleção é presumível que o conjunto de *features* selecionadas

tenha muita redundância, ou seja, que a dependência entre estas seja elevada. Quando duas *features* têm uma grande dependência entre elas, o poder de discriminarem uma classe não varia muito quando uma delas é removida. Deste modo foi implementada uma condição de mínima redundância (R), de modo a selecionar *features* mutuamente exclusivas:

$$min\, R(S),\; R = \frac{1}{|S|^2} \sum_{x_i, x_j \in S} I(x_i; x_j) \tag{4.32}$$

A combinação das condições 4.31 e 4.32 resulta no critério *minimal-redundancy-maximal-relevance* (mRMR), que as combina de forma simples resultando no operador $\phi(D, R)$:

$$Max\, \Phi(D, R),\; \Phi = D - R \tag{4.33}$$

Assim é possível aplicar métodos de procura incremental usando o operador $\Phi(.)$. Supondo que já estão selecionadas $m-1$ *features*, no conjunto S_{m-1}, a tarefa consiste em selecionar a m-ésima *feature* do conjunto de candidatas $\{X - S_{m-1}\}$, através da seleção da *feature* que maximiza $\Phi(.)$, a que corresponde a condição:

$$max_{x_j \in X - S_{m-1}} \left[I(x_j; c) - \frac{1}{m-1} \sum_{x_i \in S_{m-1}} I(x_j; x_i) \right] \tag{4.34}$$

Capítulo 5

Metodologia Proposta

O trabalho desenvolvido começou com a identificação e estudo das bases teóricas, revistas nos Capítulos anteriores, necessárias para chegar ao algoritmo proposto. Esta primeira fase de aprendizagem levou à elaboração de 3 tipos de algoritmos de seleção, que variam no tipo de procura. Assim foram implementados algoritmos com tipo de procura: *forward*, *backward* e *plus-l-minur-r*; em que foram usados os classificadores descritos no Capítulo 3, combinados com seleção do tipo *wrapper* e *filter*. Nesta primeira fase foi também implementada a validação cruzada com 10 *folds*.

Numa segunda fase foi implementado o algoritmo proposto, cuja abordagem é explorada de seguida. Por fim foi implementada a técnica de *bootstrap*, e consequentemente 2 técnicas de agregação dos conjuntos de *features* selecionadas, de modo a chegar a um conjunto final, e uma técnica de fusão das classificações obtidas. As técnicas de agregação serão exploradas no Capítulo 6.

5.1 *Forward Search*

A primeira fase de desenvolvimento consistiu na implementação de um algoritmo de seleção de *features* com procura do tipo *forward*. Deste modo o objetivo consiste em selecionar N_f *features* de um total de N. A primeira implementação deste algoritmo foi realizada com uma seleção do tipo *wrapper*. Deste modo o algoritmo implementado corresponde ao algoritmo 3.1 do Capítulo 3, em que a função de custo ($\mathfrak{J}$) corresponde à média das exatidões de classificação da validação cruzada dos dados de treino. Assim a cada iteração é calculada a função de custo para a adição cada uma das *features* candidatas ($X_k \in Z$) e adicionada a *feature* que maximiza a função.

A segunda implementação deste tipo de procura foi realizada com seleção do tipo *filter*, em que o critério usado foi a maximização informação mútua entre a *feature* candidata e a saida Y. Deste modo a função de custo usada corresponde a:

$$X_k = \arg \max_{X_k \in Z} I(X_k; Y) \tag{5.1}$$

Assim a cada iteração é calculada a função de custo ($\mathcal{T} = I$) para a cada uma das *features* candidatas ($X_k \in Z$) e adicionada a *feature* que maximiza a informação mútua entre esta e a saída, ou seja, a *feature* que mais informação contém sobre a saída, de acordo com o algoritmo 5.1

Algorithm 5.1 Algoritmo SFS com seleção do tipo *filter*

1: *Input: Conjunto de variáveis de entrada* X*, saída* Y*, e número de features a selecionar* N_f*;*
2: *Output: Conjunto de features selecionadas* S*;*
3: *Inicialização:* $S = \emptyset$*;* $Z = X$*;* $k = 0$*;*
4: *Encontrar* X_k *que maximiza o critério* $I(X_k; Y)$*:* $X_k = \arg\max_{X_k \in Z} I(X_k; Y)$
5: *Atualizar* $S := S \cup X_k$*;*
6: *Atualizar* $Z := Z \setminus X_k$*;*
7: *k:=k+1;*
8: *Repetir de* 4 *a* 7 *até* $k = N_f$*.*

Finalizando a primeira fase foi implementado outro algoritmo de procura *forward search* com seleção do tipo *filter*. Nesta implementação foi seguido o trabalho de Peng [Peng *et al.*, 2005], sendo utilizado o critério mRMR descrito Capítulo 4. Esta implementação segue o conjunto de etapas do algoritmo 5.1, com a diferença que a função de custo corresponde à equação 4.34.

5.2 *Backward Search*

A segunda fase de desenvolvimento consistiu na implementação do tipo de procura *backward search*. Assim o objetivo consiste em eliminar N_f *features* do conjunto inicial que corresponde ao conjunto total de *features* ($S = X$). Por sua vez o conjunto inicial de candidatas, neste caso a serem eliminadas, também corresponde ao conjunto de *features* disponíveis ($Z = X = S$). Similarmente ao caso da implementação da procura *forward search* a primeira implementação deste tipo de procura foi feita com uma seleção da *feature* a eliminar do tipo *wrapper*, em que a função de custo corresponde à média das exatidões de classificação da validação cruzada dos dados de treino. Algoritmo de eliminação do tipo *wrapper* está descrito no algoritmo 5.2.

Visto de outra perspetiva pode-se dizer que o critério corresponde à seleção do melhor subconjunto ($S \setminus X_k$), em vez da seleção da *feature* X_k a eliminar.

A segunda implementação da procura do tipo *backward search* foi efetuada com seleção do *filter*. Similarmente ao implementado no *forward search* foi utilizado um critério simples, em que a função de custo utilizada corresponde igualmente à informação mútua entre cada uma das *features* selecionadas S e a saída Y. Assim o algoritmo implementado corresponde ao algoritmo 5.2, com uma função de custo diferente e sendo esta minimizada e não maximizada,

Algorithm 5.2 Algoritmo SBS com seleção do tipo *wrapper*

1: *Input: Conjunto de variáveis de entrada* X*, saída* Y*, e número de features a eliminar* N_f*;*
2: *Output: Conjunto de features selecionadas* S*;*
3: *Inicialização:* $S = X$*;* $k = 0$*;*
4: *Encontrar* X_k *que minimiza o critério* $\mathfrak{T}(S \setminus X_k; Y)$*:* $X_k = \arg\max_{X_k \in S} \mathfrak{T}(X_k; Y)$;
5: *Atualizar* $S := S \setminus X_k$*;*
6: *k:=k+1;*
7: *Repetir de* 4 *a* 6 *até* $k = N_f$*.*

de acordo com a equação 5.2.

$$X_k = \arg \min_{X_k \in S} I(X_k; Y) \tag{5.2}$$

Por fim foi implementado novamente o critério mRMR, seguindo os passos do algoritmo 5.2, com a diferença na função de custo que corresponde à minimização da equação 4.34.

5.3 PLMR *search*

Na terceira fase de desenvolvimento foi implementado o tipo de procura PLMR. Como explicado no Capítulo 3 este tipo de procura consiste em aplicar o SFS seguido do SBS, de modo a permitir que *features* selecionadas possam ser eliminadas, e que *features* eliminadas possam ser adicionadas, evitando o *nesting effect*. Assim a implementação deste tipo de procura consistiu em juntar a etapa de adição (SFS) com a etapa de eliminação (SBS) para cada tipo de seleção.

Esta implementação foi adaptada em relação à sua definição original, isto porque foi imposta a condição de não eliminar *features* que tenham sido adicionadas na iteração anterior. Desta forma podia-se considerar que a procura seria do tipo *floating*, no entanto, a definição deste tipo indica que apenas há eliminação no caso de se melhorar a função de custo, o que não se verifica nesta implementação, pelo que foi definido como sendo do tipo PLMR.

5.4 Algoritmo Final

O algoritmo proposto neste trabalho é baseado no tipo de procura flutuante do tipo *plus-l-minus-r* em que o número de *features* adicionadas é definido por (L) e o número de *features* eliminadas tem definido o limite máximo de (R) a cada iteração do ciclo de procura. O algoritmo proposto tem características específicas no modo como interage com o classificador, que o categorizam a o seu tipo de seleção como sendo do tipo *embedded*. Como vai ser demonstrado posteriormente, na etapa de adição o método usa a predição de classes devolvida pelo classificador, de modo a calcular o erro de predição, que por sua vez é uma variável usada

na função de custo desta etapa. Na etapa de eliminação a iteração consiste na utilização da probabilidade posterior devolvida pelo classificador na predição de classes após se treinar o classificador sem a *feature* candidata a ser eliminada.

5.4.1 Etapa de adição - *Forward Search*

A etapa de adição seleciona a *feature* (X_k) com a máxima IM no contexto do conjunto de *features* selecionadas atual (S), retirando-a do conjunto de candidatas a seleção (Z). As operações realizadas são:

$$\begin{aligned} X_k &= \arg\max_{X_k \in Z_t} I(X_k, S_t|Y), \\ S_{t+1} &\leftarrow S_t \cup X_k \\ Z_{t+1} &\leftarrow Z_t \setminus X_k \end{aligned} \tag{5.3}$$

Este método de seleção heurística adiciona a *feature* que apresenta o maior incremento possível na probabilidade condicional, representando uma maximização iterativa ávida. Verificando-se:

$$\begin{aligned} I(X_k;Y|S) &= \sum p(X_k,Y,S)\, log\, \frac{p(X_k,Y|S)}{p(X_k|S)\, p(Y|S)} \\ &= H(X_k|S) + H(Y|S) - H(X_k,Y|S) \\ &= H(X_k,S) - H(S) + H(Y,S) - H(S) - H(X_k,Y,S) + H(S) \\ &= H(Y|S) - H(Y|X_k,S) = I(X_k,Y|S) \end{aligned} \tag{5.4}$$

Daqui é possível provar que $I(X_k,Y|S) = I(X_k,S;E)$ da seguinte forma:

$$\begin{aligned} & I(X_k,Y|S) \\ &= H(Y|S) - H(Y|X_k,S) \\ &= \sum_S p(S=S^n)\, H(Y|S=S^n) \\ &= \sum_{S^n \in S} p(S=S^n)\, H(f(S=S^n)+\epsilon|S=S^n) \\ &= \sum_{S^n \in S} p(S=S^n)\, H(\epsilon|S=S^n) \\ &= H(E|S) - H(E|X_k,S) \\ &= H(E) - H(E|X_k,S) \\ &= I(X_k,S;E) \end{aligned} \tag{5.5}$$

Em que ϵ corresponde ao erro de classificação binária:

$$\epsilon = \begin{cases} 1,\ Y \neq f(s) \\ 0,\ otherwise \end{cases} \tag{5.6}$$

No entanto, a função de custo $I(X_k, S; E)$ verifica a seguinte relação:

$$I(X_k, S; E) = I(X_k; E) + I(S; E|X_k) \tag{5.7}$$

O que resulta na seleção de *features* de acordo com:

$$X_k = \arg\max_{X_k \in Z} I(X_k, E) + I(S, E|X_k) \tag{5.8}$$

Na primeira iteração não existem *features* selecionadas ($S = \emptyset$), o que significa que o conjunto de *features* candidatas é o conjunto de todas as *features* disponíveis ($Z = X$), e que a primeira *feature* selecionada é a que contém mais informação com a saída Y:

$$X_k = \arg\max_{X_k \in Z} I(X_k|Y) \tag{5.9}$$

Na segunda iteração, a seleção da segunda *feature* já é feita com base na equação 5.8, pois já é possível treinar um classificador com esta e encontrar um vetor de erro de classificação (E). Verifica-se também que tendo em conta as atualizações de S e Z referidas em 5.3, $|S| = 1$.

Nas seguintes iterações torna-se necessário encontrar dentro das *features* selecionadas (S) a que melhor representa o grupo. Esta é selecionada através da maximização da informação entre cada uma das selecionadas (S) e o erro (E) condicionada à *feature* candidata (X_k): da seguinte forma:

$$S_1 = \arg\max_{S_1 \in S} I(S_1; E|X_k) \tag{5.10}$$

O que significa que este cálculo é efetuado para cada *feature* candidata, ou seja, $|Z|$ vezes. Assim, as *features* seguintes são feitas da seguinte forma:

$$X_k = \arg\max_{X_k \in Z} I(X_k, E) + I(S_1, E|X_k) \tag{5.11}$$

Na etapa de adição de L *features* ao conjunto de *features* selecionadas S começa-se por verificar as *features* candidatas a serem selecionadas (Z), através da verificação das *features* existentes ainda não contidas em S ($Z = X \notin S$). De seguida verifica-se se o parâmetro L cumpre a condição de ser menor ou igual ao número de *features* candidatas ($L <= |Z|$), altera-se para o número de *features* candidatas caso não cumpra a condição. Finalizando o ciclo de adição, no final de cada iteração é calculada a exatidão de classificação com a validação cruzada para os dados de treino, e a exatidão para os dados de teste.

5.4.2 Etapa de eliminação - *Backward Search*

Na etapa de eliminação, tendo definido $R = 1$, é selecionada uma *feature* para ser eliminada. Assim é considerada a utilidade de cada *feature* candidata (X_k) através da informação mútua com a saída (classes), condicionada a todas as *features* já selecionada (S) sem a atual

candidata (X_k). As operações efetuadas são:

$$\begin{aligned} X_k &= \arg\max_{X_k \in S_t} I(S_t \setminus X_k; Y), \\ S_{t+1} &\leftarrow S_t \setminus X_k \\ Z_{t+1} &\leftarrow Z_t \cup X_k \end{aligned} \tag{5.12}$$

Desta forma é selecionado o conjunto que obtenha maior IM com a saída, i.e., é eliminada a *feature* (X_k) cujo conjunto restante ($S_t \setminus X_k$) tenha mais relevância com a saída (Y).

Tendo em conta as seguintes relações:

$$I(S \setminus X_k; Y) = H(Y) - H(Y|S \setminus X_k) \tag{5.13}$$

$$H(Y|S \setminus X_k) = -\sum p(Y|S \setminus X_k)\, log\, p(Y|S \setminus X_k) \tag{5.14}$$

A probabilidade condicionada $p(Y|S \setminus X_k)$ corresponde à probabilidade posterior devolvida pelo classificador, na fase de classificação. Por outro lado, a entropia da variável das classes ($H(Y)$) é calculada da seguinte forma:

$$H(Y) = -log(\frac{1}{2}) \tag{5.15}$$

Visto que se considera apenas classificação binária, ou seja, 2 classes.

No entanto, o cálculo da probabilidade condicionada $p(Y|S \setminus X_k)$, representa uma necessidade de treinar o classificador $|S|$ vezes, de modo a se considerar a possibilidade de eliminação de cada uma das *features* selecionadas. Posto isto, em vez de efetivamente eliminar cada uma das *features* candidatas ($X_k \in Z, Z = S$) de modo a se calcular a informação mútua entre a saída (Y) e o conjunto de selecionadas sem cada uma destas, adotou-se uma alternativa que demonstrou ter o mesmo efeito. Seguindo o trabalho de Yang [Yang *et al.*, 2009], na vez da eliminação procedeu-se à permutação aleatória de todas as amostras da *feature* que se pretende eliminar. Assim só depois da seleção da *feature* X_k é que se procede à sua eliminação do conjunto de selecionadas S.

Para finalizar após a eliminação da *feature* selecionada através da equação 6.1, é calculada a exatidão de classificação com a validação cruzada para os dados de treino, e a exatidão para os dados de teste.

Capítulo 6

Resultados e Discussão

6.1 Implementação

A aplicação do algoritmo proposto passou pelas várias etapas já abordadas, pelo que a explicação que segue aborda a estrutura do código do algoritmo final, pela ordem em que está escrito, e consequentemente pela ordem que é executado, e não necessariamente pela ordem que foi implementado. Assim a primeira parte do código aborda o problema de pré-processamento dos dados, em que é necessário adaptar os dados de entrada ao algoritmo desenvolvido.

6.1.1 Pré-processamento dos dados

No caso deste algoritmo, desenvolvido em volta do problema de classificação binário, em relação aos dados de entrada é necessário introduzir na variável X todas as amostras disponíveis, com dados relativos a todas as *features* candidatas a serem selecionadas ou eliminadas, e introduzir na variável Y as classes correspondentes.

No caso do *dataset* utilizado no desenvolvimento isto significou eliminar ou não utilizar uma coluna de dados correspondente à identificação das amostras, introduzindo as restantes colunas menos a última, correspondentes ás *features*, na variável X, e a última coluna, correspondente as classes na variável Y.

6.1.2 Dados de treino e de teste

Sabendo que um problema de classificação tem uma fase de treino e uma fase de testes, torna-se essencial dividir o conjunto de todos os dados existentes em duas partes, para garantir a realização dessas duas fases. Assim a primeira função desenvolvida realiza esse procedimento, recebendo como parâmetros o X e o Y.

Neste trabalho optou-se por uma divisão por igual, com seleção aleatória, resultando num total de quatro subconjuntos de dados que no caso do número total de amostras ser par têm igual comprimento, caso contrario diferem uma unidade entre os dois conjuntos. Estes são

os conjuntos de treino: conjunto de amostras X_train as correspondentes classes Y_train; e os conjuntos de teste: conjunto de amostras X_test as correspondentes classes Y_test. Esta foi utilizada nos 4 *datasets* iniciais (grupo 1) descritos mais a frente neste Capítulo.

Para os *datasets* do grupo 2 foi efetuada uma divisão de 70% para treino e 30% para teste, devido ao número de amostras limitado.

6.1.3 Validação cruzada

Tendo verificado em secções anteriores deste trabalho, os problemas que podem existir na utilização de classificadores, como *overfitting*, decidiu-se implementar o método da validação cruzada neste algoritmo. Assim este método pretende salvaguardar o algoritmo desses problemas, tentando garantir a fiabilidade dos resultados apresentados e a robustez do algoritmo desenvolvido.

Neste trabalho optou-se pelo uso de 10 "dobras" (*folds*) de validação, pelo que no passo seguinte do algoritmo passou pela implementação da função do MATLAB *crossvalind* de modo a obter para cada conjunto (amostra e correspondente classe) um índice aleatório, de 1 ate 10, que corresponde ao índice a que o *fold* vai pertencer. Estes são usados posteriormente nas funções implementadas para cada classificador, em que se realiza a validação cruzada. Por exemplo, no caso do *Naive Bayes* a função tem o nome de $error\_kfcv\_nb$. Assim as funções de validação cruzada devolvem o erro de classificação de cada dobra, sendo depois aplicada uma média. Criando os índices que vão corresponder cada amostra a uma dobra externamente faz com que cada classificador seja treinado e testado com as mesmas amostras em cada iteração do processo de validação cruzada.

Na fase de desenvolvimento final em que foram utilizados os *datasets* do grupo 2 optou-se por validação cruzada do tipo LOOCV. Deste modo são testadas todas as amostras individualmente.

6.1.4 *Bootstrap*

O passo seguinte consiste na aplicação do segundo método de generalização do modelo, o *bootstrap*. Para tal começa-se por definir o número total de *bootstraps* ou *datasets* que se pretende usar, guardando o valor na variável $nSets$, sabendo que esse número menos uma unidade corresponde ao número de novos *datasets* criados a partir do *dataset* original. Consequentemente no caso de se definir o valor unitário utiliza-se o *dataset* original sem repetições de amostras, e sem aplicar a técnica de *bootstrap*.

Este passo é implementado através da linha de código $id = ceil(rand(M, nSets) * M)$, em que a variável M corresponde ao número de amostras dos dados de treino, assim obtém-se uma matriz de tamanho $[M; nSets]$, i.e., $nSets$ colunas, em que cada coluna tem M índices que correspondem a amostras do *dataset* original. A partir deste ponto a aplicação deste método corresponde a correr $nSets$ vezes os algoritmos de seleção de variáveis com cada um

dos *nSets datasets* criados a partir do original.

6.1.5 Parâmetros de procura

O ciclo de procura não tem limitação de iterações, sendo finalizado apenas quando se alcança o número de *features* pretendidas, chamando a cada iteração a função correspondente à etapa de eliminação e a função correspondente à etapa de adição.

O parâmetro de procura L corresponde ao número de *features* a serem adicionadas em cada iteração do ciclo de procura, i.e., a cada etapa de adição, a menos que este valor seja maior do que número de *features* candidatas, nesse atribui-se a L o número de *features* candidatas.

No caso do parâmetro R, este corresponde ao número máximo de *features* a serem eliminadas por ciclo de procura, isto porque existem uma condição que pode fazer com que o número de *features* eliminadas não corresponda ao valor de R. Esta condição indica que nenhuma *feature* pode ser eliminada se pertencer ao conjunto das *features* adicionadas na última fase de adição. Assim, no final da fase de eliminação as *features* que pertençam ao grupo das ultimas *features* adicionadas e também ao grupo das últimas *features* eliminadas, são repostas no grupo de *features* selecionadas.

De modo a comparar e obter os melhores parâmetros de procura testou-se o algoritmo com diferentes valores de números de *features* a serem adicionadas e eliminadas R em cada iteração. Assim foram testadas as seguintes combinações presentes na Tabela 6.1. Anali-

Tabela 6.1: Combinações de L (nº de adições) e R (nº de elimnações) testadas

L=1	R=1
L=2	R=1
L=2	R=2
L=3	R=1
L=3	R=2
L=3	R=3

sando a Tabela 6.1 verifica-se que os valores testados cumprem a condição $L >= R$, ou seja, que o número de *features* adicionadas a cada iteração é sempre maior ou igual ao número de *features* eliminadas, de modo a que o tamanho do conjunto de *features* selecionadas seja crescente. No entanto, mesmo com a condição de eliminação implementada por vezes verificou-se um problema em que o algoritmo entrava num ciclo infinito no caso específico de $L = R$, isto pois apesar de não ser possível acontecer que as *features* eliminadas sejam as mesmas que as últimas adicionadas, acontecia que as *features* adicionadas correspondiam consequentemente às penúltimas *features* eliminadas. Para resolver este problema implementou-se numa primeira fase, a diminuição de uma unidade no valor de R, quando se verifica que o número de *features* selecionadas é o mesmo durante três iterações, de modo a impedir a criação de

um *deadlock*.

Posteriormente, tendo em conta os problemas verificados na escolha dos parâmetros de procura, tomou-se a decisão de usar os valores $L = 3$ e $R = 1$. A escolha destes valores deve-se ao facto de evitarem os problemas referidos, e de garantirem um rápido crescimento do número de *features* selecionadas, e consequentemente um menor esforço computacional.

Atingindo o número de *features* que se pretende selecionar, o algoritmo de procura é finalizado efetuando a avaliação final da exatidão das *features* selecionadas, com a validação cruzada dos dados de treino e com os dados de teste. Assim considera-se que a ordem das *features* no conjunto de selecionadas contém significado na relevância das *features*, e efetua-se a avaliação da exatidão para cada subconjunto de *features* por ordem de relevância, i.e., desde a primeira, até ao conjunto de todas as *features* selecionadas.

6.2 Resultados

Das etapas de implementação do algoritmo desenvolvido destacam-se a implementação da etapa de adição e a etapa de eliminação. Deste modo foram testadas e comparados os resultados dessas etapas independentemente e posteriormente em conjunto, do qual resultou o algoritmo final. Nos testes efetuados selecionou-se todas as *features* disponíveis, o que resultou em conjuntos de *features* selecionadas de tamanho igual ao número de *features* total ($|S| = |X|$). Daqui se percebe que no fundo se procedeu a uma ordenação da *features*, de acordo com a sua ordem de seleção. Deste modo usou-se 3 tipos de seleção para termos comparativos:

- *Wrapper* - baseada na *performance* do classificador utilizado;
- algoritmo baseado no mRMR - utilizando a toolbox de Peng para o cálculo da informação mútua;
- algoritmo desenvolvido - utilizando a toolbox de Pocock para o cálculo da informação mútua;

Assim os resultados apresentados são discriminados por método de procura utilizado, método de seleção, e classificador utilizado. Deste modo estes serão apresentados em função do tipo de procura utilizado. Da figura 6.1 se percebe que isto corresponde a que apresentando resultados de apenas um tipo de procura, este contenha 9 diferentes resultados de exatidão correspondentes à combinação dos métodos de seleção com os classificadores utilizados. Deste modo dá-se mais ênfase aos resultados do tipo de procura PLMR, visto que este combina os outros dois métodos de procura.

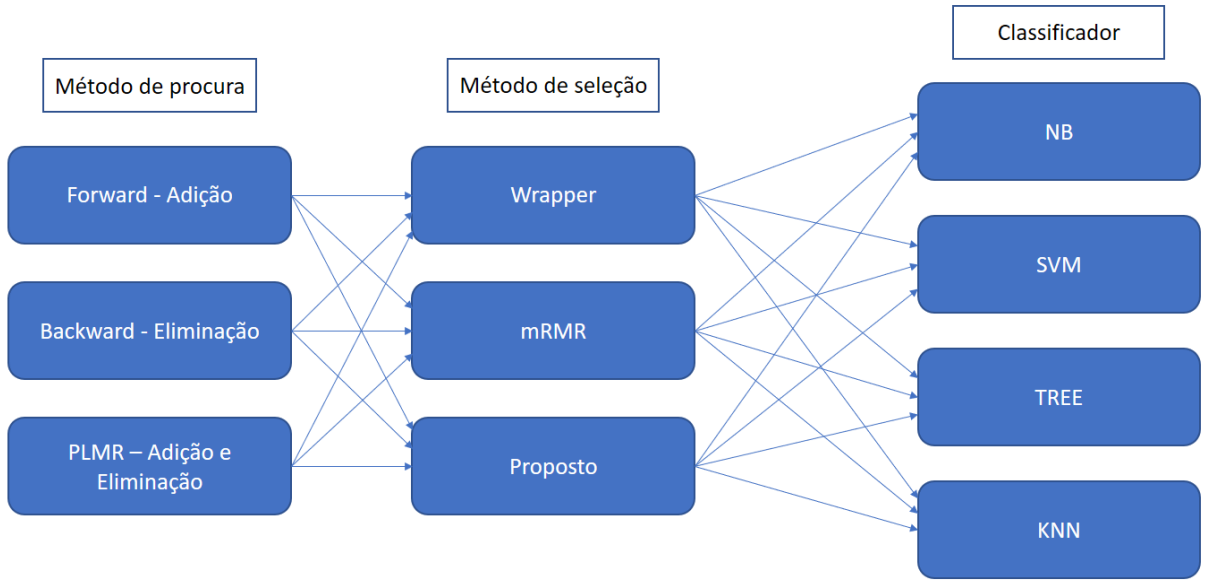

Figura 6.1: Esquema da combinação de métodos e classificadores

6.2.1 Definição de métricas

A métrica utilizada para avaliar o desempenho dos algoritmos testados foi a exatidão de classificação dos dados de teste e a exatidão de classificação da validação cruzada dos dados de treino, sendo que as apenas as medidas de teste são consideradas para conclusões. Esta é descrita pela seguinte equação:

$$ACC = \frac{|Y_i == \hat{Y}_i|}{m} \tag{6.1}$$

onde Y corresponde ao vetor de classificações reais das amostras, $\hat{Y}$ corresponde ao vetor de classificação obtido pela classificador, e m corresponde ao número total de amostras.

Deste modo, quando se usa a técnica de *bootstrap* os valores de exatidão demonstrados correspondem à média dos valores obtidos em cada resultado quando aplicado um *dataset* do *bootstrap*.

6.2.2 *Datasets*

Os *datasets parkinson*, *ionosphere*, *sonar* e *breast cancer* (grupo 1) foram adquiridos no repositório de *Machine Learning* da Universidade da Califórnia em Irvine, tendo sido escolhidos por cumprirem as seguintes condições: não possuírem valores omissos; os seus atributos ou *features* serem do tipo inteiro ou real; e serem de classificação binária, ou seja, possuírem apenas 2 classes. No entanto, não correspondem a *datasets* de *microarrays* de genes, tendo como função apenas comprovar a eficácia e capacidade de generalização do algoritmo.

O *dataset parkinson* contém 23 *features* de 197 amostras com valores reais, referentes a medidas de voz destinadas ao diagnostico da doença de Parkinson. O *dataset ionosphere*

contém 34 *features* de 351 amostras com dados reais e inteiros, referentes a dados fornecidos por um radar obtidos com o objetivo de identificar algum tipo de estrutura na ionosfera. O *dataset sonar* contém 60 *features* de 208 amostras com dados reais, referentes a dados fornecidos por um sonar obtidos com o objetivo de identificar se a estrutura em causa corresponde a uma rocha ou mina. O *dataset breast cancer* contém 32 *features* de 569 amostras com dados reais, referentes a medições efetuadas em células de tecido mamário com o objetivo de diagnosticar cancro da mama.

O segundo conjunto de *datasets*, *colon*, *leukemia* e *breast2* (grupo 2) foram obtidos na página do método mRMR, tendo sido utilizados por Peng para testar o seu método proposto. Estes correspondem a *microarrays* de genes, com dados discretizados em 3 estados, com 2 classes, adquiridos com o objetivo de diagnosticar cancro do cólon, leucemia e cancro da mama, respectivamente. Estes *datasets* têm as seguintes características: *colon* possui 62 amostras de 2000 *features*; *leukemia* possui 72 amostras de 7070 *features*; *breast2* possui 77 amostras de 4869 *features*.

De seguida são apresentados os resultados obtidos para os *datasets* do grupo 1, sendo apresentado os resultados dos *datasets* do grupo 2 apenas na secção seguinte.

6.2.3 Resultados utilizando *datasets* do grupo 1

Os seguintes resultados correspondem à média da exatidão de classificação de todos os subconjuntos de *features* selecionadas, que neste caso foram todas as *features* disponíveis. Por sua vez a exatidão de classificação para cada subconjunto também corresponde à média de classificação obtida para o subconjunto com o respectivo número de *features* em cada iteração do *bootstrap*, num total de 4 iterações, ou seja, 4 conjuntos de treino criados pelo *bootstrap*. Para cada combinação *dataset*-método utilizado são apresentadas as exatidões de classificação médias dos dados de teste e da média da validação cruzada dos conjuntos de treino.

6.2.3.1 Resultados com procura *Forward search*

Os resultados da etapa de adição foram obtidos usando o algoritmo final de *floating search*, definindo o parâmetro de *backward search*, i.e., o parâmetro que define o número de *features* a eliminar (R) com o valor 0. Deste modo o algoritmo apenas realiza a etapa de adição, até chegar ao número pretendido, funcionando com uma procura do tipo *forward search*. Esta implementação foi efetuada depois de se verificar os mesmos resultados nestes métodos (forward e PLMR com $R = 0$). Como referido no início deste Capítulo, tendo aplicada a técnica de *bootstrap* os resultados resultam da média das 4 iterações realizadas pelo algoritmo, cujos resultados correspondem à média de exatidão para todos os subconjuntos de *features*.

A Tabela 6.2 apresenta a média das exatidões de classificação dos dados teste para cada combinação: método de seleção com classificador; e também a média por classificador e por

método. Estes resultados correspondem à média dos resultados para cada um dos 4 *datasets* do grupo 1.

Tabela 6.2: Comparação das médias de exatidão de classificação dos dados de teste dos diferentes métodos usando *forward search* para *datesets* do grupo 1

	Wrapper	mRMR	Proposto	Média
NB	0,874	0,858	0,837	0,857
SVM	0,862	0,847	0,834	0,848
TREE	0,822	0,830	0,832	0,828
KNN	0,826	0,778	0,781	0,795
Média	0,846	0,8285	0,821	

6.2.4 Resultados com procura *Backward search*

A Tabela 6.3 apresenta a média das exatidões de classificação dos dados teste para cada combinação: método de seleção com classificador; e também a média por classificador e por método. Estes resultados correspondem à média dos resultados para cada um dos 4 *datasets* do grupo 1.

Tabela 6.3: Comparação das médias de exatidão de classificação dos dados de teste dos diferentes métodos usando *backward search* para *datesets* do grupo 1

	Wrapper	mRMR	Proposto	Média
NB	0,835	0,821	0,827	0,828
SVM	0,845	0,829	0,828	0,834
TREE	0,838	0,831	0,801	0,823
KNN	0,801	0,823	0,778	0,801
Média	0,830	0,826	0,808	

6.2.4.1 Resultados com procura PLMR

Estes resultaram da aplicação dos 3 métodos de seleção implementados, combinados com o tipo de procura *plus-l-minus-r*, tendo sido definido um número de *features* adicionadas de $L = 3$, e um número máximo de *features* eliminadas de $R = 1$.

A Tabela 6.4 apresenta a média das exatidões de classificação dos dados teste para cada combinação: método de seleção com classificador; e também a média por classificador e por método. Estes resultados correspondem à média dos resultados para cada um dos 4 *datasets* do grupo 1.

Tabela 6.4: Comparação das médias de exatidão de classificação dos dados de teste dos diferentes métodos usando procura PLMR para *datesets* do grupo 1

	Wrapper	mRMR	Proposto	Média
NB	0,824	0,817	0,816	0,819
SVM	0,830	0,804	0,784	0,806
TREE	0,821	0,817	0,806	0,815
KNN	0,801	0,793	0,795	0,796
Média	0,819	0,808	0,800	

6.2.4.2 Comparação de métodos

A Tabela 6.5 apresenta a média das exatidões de classificação dos dados teste para cada combinação, método de seleção com classificador, para cada *dataset* do grupo 1, e também a média por combinação, método de seleção com classificador.

Tabela 6.5: Comparação das médias de exatidão de classificação dos dados de teste dos diferentes métodos para *datesets* do grupo 1

Procura	Seleção	Parkinson	Ionosphere	Sonar	Breast Cancer	Média
Forward	Wrapper	0,831	0,854	0,694	0,899	0,819
	mRMR	0,801	0,840	0,681	0,912	0,808
	Proposto	0,791	0,820	0,681	0,911	0,801
Bacward	Wrapper	0,847	0,911	0,719	0,931	0,852
	mRMR	0,843	0,897	0,684	0,927	0,838
	Proposto	0,841	0,893	0,661	0,925	0,830
PLMR	Wrapper	0,801	0,875	0,709	0,936	0,830
	mRMR	0,816	0,850	0,708	0,931	0,826
	Proposto	0,799	0,842	0,672	0,922	0,809

6.2.5 Resultados usando *datasets* de genes (grupo 2)

Tendo verificado a eficácia, poder de generalização e correta implementação do método de procura PLMR, usou-se apenas este método para seleção de *features* (genes) dos *datasets* do grupo 2. A implementação deste método de procura deve-se a dois fatores: (*i*) o primeiro fator deve-se ao facto de se ter verificado a correta implementação da etapa de adição no PLMR, cuja confirmação foi obtida após se verificar os mesmos resultados no caso da utilização deste e do método de procura *forward*; (*ii*) o segundo fator prende-se com o facto de os *datasets* do grupo 2 corresponderem a *microarrays* de genes, e portanto apresentarem características expectáveis, i.e., um conjunto de amostras reduzido e um número de *features* elevado, o que não permitiu obter resultados para o método de procura *backward search*

devido ao custo computacional, visto que no melhor dos casos (menor número de *features*), para o *dataset colon* a seleção de 50 *features* significaria a eliminação de 1950.

Como tem vindo a ser descrito neste Capítulo a implementação dos *datasets* do grupo 2 sofreu alterações em relação ao grupo 1, devido à diferença nas características dos dados. No entanto, estas alterações não representam mudanças na implementação dos métodos de procura ou seleção, mas apenas na robustez dos resultados apresentados.

Assim os resultados seguintes foram recolhidos sem a aplicação do *bootstrap*, e com uma validação cruzada do tipo LOOCV. No entanto, para garantir a robustez dos resultados foram recolhidos 30 resultados para a seleção de 50 *features* em cada um dos *datasets*, em que para cada um destes, os 70% de amostras de treino e 30% de amostras de teste são selecionadas aleatoriamente.

A Tabela 6.6 apresenta a média das exatidões de classificação dos dados teste para cada combinação: método de seleção com classificador; e também a média por classificador e por método. Estes resultados correspondem à média dos resultados para cada um dos 3 *datasets* do grupo 2.

6.2.5.1 Resultados com procura *forward search*

Tabela 6.6: Comparação das médias de exatidão de classificação dos dados de teste dos diferentes métodos usando *forward search* para *datesets* do grupo 2

	mRMR	Proposto	Média
NB	0,813	0,739	0,776
SVM	0,791	0,733	0,762
TREE	0,773	0,713	0,743
KNN	0,810	0,741	0,775
Média	0,796	0,732	

6.2.5.2 Resultados com procura PLMR

A Tabela 6.7 apresenta a média das exatidões de classificação dos dados teste para cada combinação: método de seleção com classificador; e também a média por classificador e por método. Estes resultados correspondem à média dos resultados para cada um dos 3 *datasets* do grupo 2.

Tabela 6.7: Comparação das médias de exatidão de classificação dos dados de teste dos diferentes métodos usando procura PLMR para *datesets* do grupo 2

	mRMR	Proposto	Média
NB	0,800	0,723	0,761
SVM	0,794	0,637	0,716
TREE	0,765	0,633	0,699
KNN	0,811	0,693	0,752
Média	0,792	0,671	

6.2.5.3 Comparação de métodos

A Tabela 6.8 apresenta a média das exatidões de classificação dos dados teste para cada combinação, método de seleção com classificador, para cada *dataset* do grupo 2, e também a média por combinação, método de seleção com classificador.

Tabela 6.8: Comparação das médias de exatidão de classificação dos dados de teste dos diferentes métodos para *datesets* do grupo 2

		Colon	Leukemia	Breast2	Média
Forward	mRMR	0,771	0,945	0,662	0,792
	Proposto	0,671	0,759	0,585	0,671
PLMR	mRMR	0,781	0,945	0,655	0,794
	Proposto	0,724	0,891	0,580	0,732

6.2.6 Seleção do conjunto de *features* final

Visto que a utilização da técnica de *bootstrap* cria *nSets datasets* a partir de um original, isto leva a que o algoritmo de seleção tenha *nSets* iterações, o que resulta em *nSets* conjuntos de *features* selecionadas. Assim foram usados os dois principais métodos conhecidos para a obtenção de um conjunto de *features* final.

O primeiro baseia-se na posição das *features* no conjunto selecionado. Assim é dado um peso a cada *feature* correspondente à sua posição no conjunto, e no final juntam-se os diferentes conjuntos de *features* somando os respetivos pesos atribuídos, resultando num conjunto final.

O segundo baseia-se na frequência de seleção, no entanto, este só é aplicável quando se pretende selecionar um conjunto de *features* inferior de dimensão inferior ao número de *features* disponíveis, visto que não dando importância à localização das *features* dentro do conjunto de selecionadas, no caso de se selecionar todas as *features* disponíveis, o resultado em função da frequência de seleção não tem significado. Na imagem 6.2 está representado um exemplo de histograma de ocorrências das primeiras 15 *features* selecionadas, em 6 *bootstraps*.

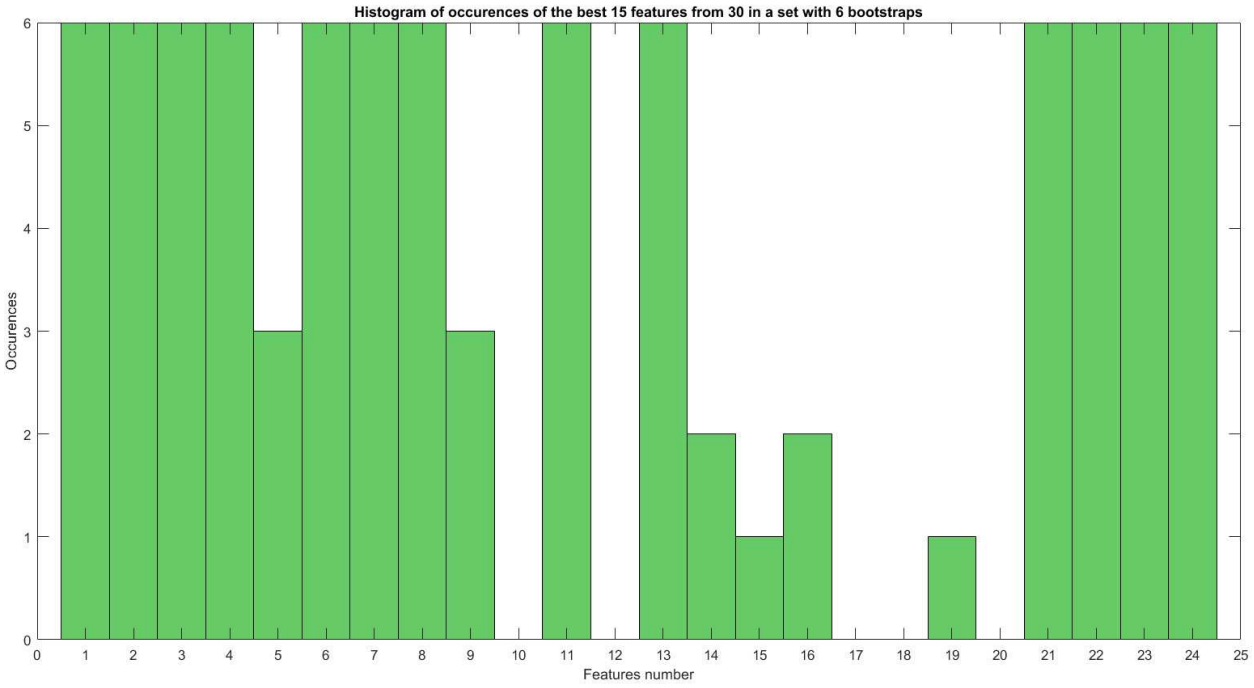

Figura 6.2: Exemplo de histograma de *features* selecionadas

6.3 Discussão de Resultados

Os resultados apresentados concentram a atenção na exatidão de classificação média dos dados de teste, pois, este é o fator decisivo na avaliação da *performance* de um método. Isto deve-se ao facto da validação cruzada dos dados de treino, apesar de poder ser usada como valor representativo da *performance*, apresentar na maioria dos casos resultados melhores do que os resultados dos dados de teste, o que significa que são resultados tendenciosos devido ao uso das mesmas amostras para as fases de treino e de obtenção de resultados.

O outro indicador de *performance* é o custo computacional/temporal, no entanto, devido ao facto de estes serem relativamente idênticos (o método proposto e o método mRMR), estes não foram considerados na avaliação da *performance*, encontrando-se disponíveis no apêndice A.

A comparação de resultados incidiu-se na comparação entre métodos, dando menos relevo à comparação entre classificadores, visto que os resultados obtidos dos diferentes classificadores podem ser combinados.

Dos resultados obtidos utilizando os *datasets* do grupo 1 verifica-se que o método proposto tem a capacidade de generalização. Isto porque estes dados não representam *microarrays* de genes, mas sim dados de problemas classificação. Assim, apesar do método proposto não apresentar, em qualquer dos diferentes tipos de procura, melhorias em relação ao *wrapper* ou ao mRMR, verifica-se que os resultados são muito próximos do método mRMR, pelo que provam a sua eficácia na seleção de *features*.

Analisando os resultados utilizando *datasets* de genes (grupo 2), retiram-se conclusões se-

melhantes, verificando-se que em qualquer das combinações, método de procura com método de seleção, o método mRMR é superior ao proposto.

Capítulo 7

Conclusão e Trabalhos Futuros

Nesta dissertação foi proposto o desenvolvimento de um algoritmo de seleção de *features* baseado na maximização da informação mútua entre as *features* selecionadas e as classificações existentes, com um tipo de procura *plus-l-minus-r*. Este método foi desenvolvido de modo a ser genérico para qualquer tipo de *dataset*, desde que os as suas variáveis sejam reais (não categóricas), que não contenha valores desconhecidos, e que contenha apenas 2 classes, ou seja, referentes a problemas de classificação binária.

De modo a obter termo comparativo foram ainda implementados dois métodos de seleção, um do tipo *wrapper* e um do tipo *filter* (mRMR).

Para validar e avaliar o desempenho do método proposto, começou-se pela aplicação de 4 *datasets* de uma base de *datasets* para problemas de *machine learning*. A qualidade da *performance* dos métodos implementados foi avaliada principalmente com base na média de exatidão de classificação dos dados de teste. Com vista à obtenção de resultados mais robustos foi também aplicada a técnica de *bootstrap* usando um total de 4 *datasets* para cada *dataset* original.

Os resultados obtidos demonstram que o método proposto tem capacidade de selecionar *features* com uma *performance* aceitável. Isto porque apresenta boa média de exatidão de classificação, dentro de um tempo de computação satisfatório. No entanto, não foram conseguidas melhorias em relação aos métodos implementados comparativamente.

Em relação à seleção de genes o método proposto revelou capacidade de realizar esta tarefa, apresentando resultados próximos do método mRMR, mas em todos os casos inferiores. Assim não foram verificadas melhorias, pelo que a abordagem implementada revela necessitar de uma abordagem diferente de modo a superar os algoritmos utilizados na seleção de genes existentes na literatura.

Apesar disso o método proposto demonstra viabilidade na seleção *features*, e consequentemente na redução do custo computacional de treino de classificadores com estas.

7.1 Trabalho Futuro

Visto que o método proposto implementado não conseguiu verificar melhorias em relação ao existente na literatura, seria interessante melhorar mais a sua eficácia e a eficiência. Seria também interessante obter mais resultados sobre *performance* do algoritmo para *datasets* de *microarrays* de genes.

Num olhar mais geral, ao entrar na era de *Big Data*, existe uma necessidade urgente de desenvolver métodos de seleção de características muito rápidos capazes de trabalhar com milhões de variáveis e biliões de amostras. Um desafio importante é o desenvolvimento de métodos mais eficientes para estimar MI em espaços de alta dimensão. Outro aspeto que merece reflexão é o tamanho ideal do subconjunto de *features* selecionadas, visto que não existe consenso para tal, e consequentemente não existindo um método de paragem de seleção geral para além de se atingir um número de *features* pré-determinado ou de não se verificar melhorias na exatidão de classificação.

Apêndice A

Resultados completos

Tabela A.1: Comparação das médias de exatidão de classificação dos diferentes métodos usando o classificador NB para os datasets do grupo 1

NB		PLMR			Forward			Back	
		Wrapper	mRMR	Proposto	Wrapper	mRMR	Proposto	Wrapper	mR
Parkinson	Treino	0,869	0,829	0,815	0,815	0,752	0,756	0,794	0,7
	Teste	0,825	0,789	0,807	0,830	0,815	0,811	0,789	0,7
Ionosphere	Treino	0,947	0,942	0,922	0,903	0,873	0,841	0,911	0,8
	Teste	0,881	0,889	0,876	0,948	0,957	0,929	0,929	0,9
Sonar	Treino	0,902	0,835	0,800	0,842	0,758	0,732	0,827	0,7
	Teste	0,710	0,711	0,703	0,773	0,724	0,677	0,691	0,6
Breast cancer	Treino	0,958	0,947	0,947	0,967	0,936	0,932	0,965	0,9
	Teste	0,881	0,881	0,881	0,949	0,937	0,935	0,934	0,9

Tabela A.2: Comparação das médias de exatidão de classificação dos diferentes métodos usando o classificador SVM para os datasets do grupo 1

SVM		PLMR			Forward			Back	
		Wrapper	mRMR	Proposto	Wrapper	mRMR	Proposto	Wrapper	mR
Parkinson	Treino	0,842	0,816	0,810	0,827	0,802	0,798	0,884	0,8
	Teste	0,852	0,763	0,742	0,898	0,882	0,881	0,807	0,8
Ionosphere	Treino	0,927	0,896	0,874	0,856	0,809	0,784	0,916	0,8
	Teste	0,849	0,798	0,767	0,894	0,858	0,841	0,851	0,8
Sonar	Treino	0,741	0,745	0,663	0,797	0,713	0,688	0,821	0,7
	Teste	0,707	0,708	0,680	0,720	0,717	0,679	0,770	0,7
Breast cancer	Treino	0,946	0,962	0,959	0,969	0,956	0,956	0,980	0,9
	Teste	0,913	0,948	0,947	0,938	0,934	0,937	0,954	0,9

Bibliografia

[Ambroise and McLachlan, 2002] Christophe Ambroise and Geoffrey J McLachlan. Selection bias in gene extraction on the basis of microarray gene-expression data. *Proceedings of the national academy of sciences*, vol. 99, no. 10, pp. 6562–6566, 2002. (Citado na página 14).

[Ang *et al.*, 2016] Jun Chin Ang, Andri Mirzal, Habibollah Haron, and Haza Nuzly Abdull Hamed. Supervised, unsupervised, and semi-supervised feature selection: a review on gene selection. *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 13, no. 5, pp. 971–989, 2016. (Citado na página 2).

[Battiti, 1994] Roberto Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on neural networks*, vol. 5, no. 4, pp. 537–550, 1994. (Citado nas páginas 4, e 29).

[Bellman, 1961] Richard Bellman. *Adaptive control process: a guided tour*. Princeton University Press, 1961. (Citado na página 13).

[Bishop, 1995] Christopher M Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995. (Citado na página 10).

[Brown *et al.*, 2012] G. Brown, A. Pocock, M.-J. Zhao, and M. Luján. Conditional Likelihood Maximisation: A Unifying Framework for Information Theoretic Feature Selection. *The Journal of Machine Learning Research*, vol. 13, pp. 27–66, 2012. (Citado na página 29).

[Cover, 1974] Thomas M Cover. The best two independent measurements are not the two best. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-4, no. 1, pp. 116–117, 1974. (Citado na página 33).

[Dash *et al.*, 2002] Manoranjan Dash, Kiseok Choi, Peter Scheuermann, and Huan Liu. Feature selection for clustering-a filter solution. In: *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pp. 115–122. IEEE, 2002. (Citado na página 23).

[Ding and Peng, 2005] Chris Ding and Hanchuan Peng. Minimum redundancy feature selection from microarray gene expression data. *Journal of bioinformatics and computational biology*, vol. 3, no. 02, pp. 185–205, 2005. (Citado na página 33).

[Duch, 2006] Włodzisław Duch. Filter methods. *Feature Extraction*, pp. 89–117, 2006. (Citado na página 4).

[Efron, 1992] Bradley Efron. Bootstrap methods: another look at the jackknife. In: *Breakthroughs in statistics*, pp. 569–593. Springer, 1992. (Citado na página 15).

[Friedman *et al.*, 2001] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, vol. 1. Springer series in statistics New York, 2001. (Citado nas páginas 7, 15, e 25).

[Guyon *et al.*, 2008] I. Guyon, S. Gunn, M. Nikravesh, and L.A. Zadeh. *Feature Extraction: Foundations and Applications*. Studies in Fuzziness and Soft Computing. Springer Berlin Heidelberg, 2008. ISBN 9783540354888. (Citado nas páginas iii, v, 11, 21, e 23).

[He *et al.*, 2006] Xiaofei He, Deng Cai, and Partha Niyogi. Laplacian score for feature selection. In: *Advances in neural information processing systems*, pp. 507–514. 2006. (Citado na página 24).

[Kohavi and John, 1997] Ron Kohavi and George H John. Wrappers for feature subset selection. *Artificial intelligence*, vol. 97, no. 1-2, pp. 273–324, 1997. (Citado nas páginas 11, e 30).

[Kraskov *et al.*, 2004] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical review E*, vol. 69, no. 6, p. 066138, 2004. (Citado na página 25).

[Kwak and Choi, 2002] Nojun Kwak and Chong-Ho Choi. Input feature selection by mutual information based on Parzen window. *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 12, pp. 1667–1671, 2002. (Citado na página 30).

[Law *et al.*, 2004] Martin HC Law, Mario AT Figueiredo, and Anil K Jain. Simultaneous feature selection and clustering using mixture models. *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 9, pp. 1154–1166, 2004. (Citado na página 24).

[Liu and Motoda, 2007] Huan Liu and Hiroshi Motoda. *Computational methods of feature selection*. CRC Press, 2007. (Citado nas páginas 1, e 9).

[Marill and Green, 1963] Thomas Marill and D Green. On the effectiveness of receptors in recognition systems. *IEEE transactions on Information Theory*, vol. 9, no. 1, pp. 11–17, 1963. (Citado na página 21).

[Peng *et al.*, 2005] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005. (Citado nas páginas iii, v, 29, 32, 33, e 36).

[Pudil *et al.*, 1994] Pavel Pudil, Jana Novovičová, and Josef Kittler. Floating search methods in feature selection. *Pattern recognition letters*, vol. 15, no. 11, pp. 1119–1125, 1994. (Citado na página 22).

[Sanger and Coulson, 1975] Fred Sanger and Alan R Coulson. A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase. *Journal of molecular biology*, vol. 94, no. 3, pp. 441IN19447–446IN20448, 1975. (Citado na página 7).

[Seijo-Pardo *et al.*, 2017] Borja Seijo-Pardo, Iago Porto-Díaz, Verónica Bolón-Canedo, and Amparo Alonso-Betanzos. Ensemble feature selection: Homogeneous and heterogeneous approaches. *Knowledge-Based Systems*, vol. 118, pp. 124–139, 2017. (Citado na página 12).

[Shendure and Ji, 2008] Jay Shendure and Hanlee Ji. Next-generation DNA sequencing. *Nature biotechnology*, vol. 26, no. 10, p. 1135, 2008. (Citado na página 7).

[Souza, 2014] Francisco Alexandre de Souza. *Computational Intelligence Methodologies for Soft Sensors Development in Industrial Processes*. Ph.D. thesis, UC, 2014. (Citado na página 12).

[Stearns, 1976] S. D. Stearns. On selecting features for pattern classifiers. In: *Proceedings of the 3rd International Conference on Pattern Recognition (ICPR 1976)*, pp. 71–75. 1976. (Citado na página 22).

[Talavera, 1999] Luis Talavera. Feature selection as a preprocessing step for hierarchical clustering. In: *ICML*, vol. 99, pp. 389–397. 1999. (Citado na página 23).

[Vergara and Estévez, 2014] Jorge R Vergara and Pablo A Estévez. A review of feature selection methods based on mutual information. *Neural Computing and Applications*, vol. 24, no. 1, pp. 175–186, 2014. (Citado nas páginas xiii, 4, 28, e 32).

[Whitney, 1971] A Wayne Whitney. A direct method of nonparametric measurement selection. *IEEE Transactions on Computers*, vol. 100, no. 9, pp. 1100–1103, 1971. (Citado na página 21).

[Xue *et al.*, 2016] Bing Xue, Mengjie Zhang, Will N Browne, and Xin Yao. A survey on evolutionary computation approaches to feature selection. *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 4, pp. 606–626, 2016. (Citado na página 2).

[Yang *et al.*, 2009] Jian-Bo Yang, Kai-Quan Shen, Chong-Jin Ong, and Xiao-Ping Li. Feature selection for MLP neural network: The use of random permutation of probabilistic outputs. *IEEE Transactions on Neural Networks*, vol. 20, no. 12, pp. 1911–1922, 2009. (Citado na página 40).

[Yeung, 2008] Raymond W Yeung. *Information theory and network coding*. Springer Science & Business Media, 2008. (Citado na página 29).

[Yu and Liu, 2004] Lei Yu and Huan Liu. Efficient feature selection via analysis of relevance and redundancy. *Journal of machine learning research*, vol. 5, no. Oct, pp. 1205–1224, 2004. (Citado na página 31).

[Yusta, 2009] Silvia Casado Yusta. Different metaheuristic strategies to solve the feature selection problem. *Pattern Recognition Letters*, vol. 30, no. 5, pp. 525–534, 2009. (Citado na página 22).