



Universidade do Porto

Faculdade de Engenharia

**FEUP**



Rui Manuel Santos Rodrigues Leite

# Combinação de Métodos de Pré-Processamento e Aprendizagem Simbólica

**Rui Manuel Santos Rodrigues Leite**

**Combinação de Métodos de Pré-Processamento  
e Aprendizagem Simbólica**



ESTAMPA E DEDICATIVA  
destinada ao autor da tese  
A. M. S. R. L.

*Tese submetida à Faculdade de Engenharia da Universidade do Porto para obtenção do grau de Mestre  
em Inteligência Artificial e Computação*

**Faculdade de Engenharia da Universidade do  
Porto – 2000**

004(043)/LE1n/COM

UNIVERSIDADE DO PORTO
Faculdade de Engenharia
<b>BIBLIOTECA</b>
N.º 51937
CDU 004(043)
Data 6 14 2001

Dedico esta tese  
à Fernanda,  
à minha mãe e ao meu pai



## RESUMO

O recente aumento da dimensão das bases de dados geradas pelas actividades industriais, governamentais, científicas entre outras, ultrapassaram em muito a nossa capacidade de análise e de interpretação de dados, necessárias à tomada de decisões. É por esse motivo que as técnicas de redução de dados investigadas na área de Extracção de conhecimento de Bases de Dados (ECBD) têm ganho certa relevância.

A ECBG é essencialmente um processo que pretende procurar identificar padrões válidos, anteriormente desconhecidos, úteis e inteligíveis, e é constituído por várias fases. A selecção e preparação de dados, redução de dimensões, procura de padrões, avaliação e interpretação dos resultados, constituem as várias fases do processo ECBG. A procura de padrões, realizada pelos algoritmos de aprendizagem e estatísticos, é efectuada tendo como base o tipo de problema (ex. classificação, regressão, agrupamento entre outros) e resulta num determinado modelo. Na fase de redução de dimensões pretende-se diminuir (segundo o número de casos, atributos e valores) o tamanho do conjunto de dados de forma a que este seja eficientemente processado pelos algoritmos de aprendizagem.

Nesta Tese de Mestrado abordamos dois métodos de redução de dimensões. O primeiro envolve utilizar apenas um subconjunto de casos, e o outro utilizar apenas um subconjunto de atributos.

Apresentamos um método de redução de casos, que designámos de algoritmo de processamento por partições progressivas (PPP), que visa construir um modelo utilizando apenas um subconjunto do conjunto de dados inicial, de forma a poupar tempo de processamento, não sacrificando os resultados finais do modelo.

Quanto à selecção de atributos desenvolvemos um método que designámos de SAPPP que utilizando os resultados parciais do algoritmo PPP, combina método de selecção de atributos com o método de redução do número de casos.

Avaliamos empiricamente os dois métodos apresentados utilizando vários conjuntos dados e verificámos ser útil aplicá-los em conjuntos de dados com um elevado número de casos, confirmando a nossa expectativa inicial. Este resultado promissor justifica que se dedique mais esforço para comprovar os resultados iniciais e melhorá-los.

## SUMMARY

Recent increase in the size of databases generated as a side effect of industrial, governmental, scientific activities often surpass our capacity to analyze and interpret this data and advance thus with decision making. This is why the techniques of data reduction studied within the area of Knowledge Discovery in Databases (KDD), have gained relevance.

The objective of Knowledge Discovery in Databases is to search for, and identify, valid and previously unknown patterns, that are intelligible to the user and also useful. This process consists of various phases, such as selection and preparation of data, dimensionality reduction, search for patterns and evaluation and interpretation of the results. The search for patterns, carried out often with the help of Machine Learning and Statistical algorithms. This process takes into account the problem type (e.g. classification, regression clustering among others) and produces a particular specialized model (e.g. a classifier). The objective of dimensionality reduction is to reduce the size of one (or more) dimensions of the data, such as number of cases, attributes or number of values.

In this M.Sc. Thesis we investigate two methods of dimensionality reduction. The first one involves using only a subset of the given cases (case reduction), and the other a subset of attributes (features). We present a method of case reduction, designated as an *algorithm for processing progressive partitions* (PPP), whose objective is to construct a model using only a subset of given cases, and in that way save time, without sacrificing much the overall performance. As for attribute reduction method, we have developed a method referred to as SAPPP, which combines *attribute selection* with the method of processing progressive partitions (PPP) referred to earlier.

We have carried evaluation of both dimensionality reduction methods on several datasets. We have verified that they are in general useful in datasets with large number

of cases, confirming thus our initial expectation. This promising results justifies that more effort is dedicated to this issue. The aim should be not only verify that the results hold in other situations, but also, to improve the methods further.

## RÉSUMÉ

L'augmentation récente de la taille des bases de données produites par les activités industrielles, gouvernementales, scientifiques et autres, surpassent souvent notre capacité d'analyser et interpréter ces données et d'avancer ainsi avec la prise de décision. C'est à cause de cela que les techniques de réduction de données, étudiées dans le domaine de la découverte de connaissance dans les bases de données (DCBD), ont gagné la pertinence.

L'objectif de la découverte de la connaissance dans les bases de données est de rechercher des configurations (formes) valides, précédemment inconnues, qui sont intelligibles et utiles pour l'utilisateur. Ce processus se compose de diverses phases, telles que la sélection et la préparation des données, la réduction de dimensionnalité, la recherche des configurations et l'évaluation et interprétation des résultats. La recherche des configurations est effectuée avec l'aide d'algorithmes d'apprentissage et des algorithmes statistiques. Ce processus tient compte du type de problème (par exemple discrimination, classification, régression, entre d'autres) et produit un modèle spécialisé particulier (par exemple un classificateur). L'objectif de la réduction de dimensionnalité (selon le nombre d'objets, attributs et valeurs) est de réduire la taille de l'ensemble de données de façon à que cet ensemble soit processé d'une manière efficiente.

Dans cette thèse de DEA, nous étudions deux méthodes de réduction de dimensionnalité. Le premier implique d'utiliser seulement un sous-ensemble des objets (réduction des objets), et l'autre un sous-ensemble d'attributs (variables).

Nous présentons une méthode de réduction des objets, indiquée comme algorithme pour procésser par des partitions progressives (PPP), dont l'objectif est de construire un modèle en utilisant seulement un sous-ensemble de l'ensemble total des données; et cela de façon à réduire le temps de procésser sans sacrifier les résultats finals du modèle.

Quant à la sélection d'attributs, nous avons développé une méthode désignée sous le nom de SAPPP, qui combine la sélection d'attributs avec la méthode de réduction d'objets (PPP).

Nous avons porté l'évaluation des deux méthodes de réduction de dimensionnalité sur plusieurs ensembles de données. Nous avons vérifié qu'ils sont en général utiles dans les ensembles de données avec un grand nombre d'objets, ce qui confirme notre expectative initiale. Ce résultat prometteur justifie que plus d'effort soit consacré à cette issue. Le but devrait être non seulement de constater que les résultats se vérifient dans d'autres situations, mais également, d'améliorer les méthodes.

## Prefácio

Num mundo cada vez mais competitivo em que as pessoas e as organizações têm que definir de forma racional as suas estratégias, conseguir extrair informações úteis dos dados é certamente uma importante vantagem competitiva.

O recente aumento da dimensão das bases de dados geradas pelas actividades industriais, governamentais, científicas entre outras, ultrapassaram em muito a nossa capacidade de análise e de interpretação de dados, necessária à tomada de decisões. É por esse motivo que as técnicas de redução de dados investigadas na área de Extracção de Conhecimento de Bases de Dados (ECBD) têm ganho certa relevância.

Fayyad [Fay96] define o processo ECBD como um processo não trivial de identificar (procurar) nos dados padrões válidos, anteriormente desconhecidos, potencialmente úteis e interpretáveis. Este processo é constituído por várias fases, nomeadamente; a selecção e preparação de dados; a redução da dimensão dos dados; a extracção de padrões (modelos) utilizando algoritmos de aprendizagem, que expliquem os dados e que possam ser usados em previsões; e a avaliação e interpretação dos modelos obtidos.

Quando a dimensão da base de dados que pretendemos tratar é extremamente elevada, as limitações de memória e de processamento dos computadores constituem um obstáculo, e assim os algoritmos de aprendizagem, muitas vezes não conseguem completar o processamento. Também o tempo de execução poderá ser um factor a ter em conta, já que algumas bases de dados poderão demorar demasiados dias ou semanas a processar, o que pode ser impraticável para certas aplicações.

Do ponto do utilizador é muitas vezes útil ir obtendo em períodos de tempo pequenos, alguns modelos prévios com resultados razoáveis, isto é com taxas de acertos aceitáveis, sabendo que o sistema vai tentar construir modelos progressivamente melhores (filosofia “anytime results”).

O objectivo desta Tese é procurar soluções que cumpram os objectivos referidos nos parágrafos anteriores. Por esse motivo abordámos a fase de redução de dados do processo ECBD, focando as técnicas de redução do número de casos (“case reduction”) e as técnicas de selecção de atributos (“feature selection”). Além disso a nossa abordagem foi restringida aos problemas de classificação.

Pretende-se com a redução do número de casos construir modelos utilizando apenas um subconjunto de dados, admitindo um pequeno decréscimo na taxa de acertos obtidas (razão entre as classificações correctas e o número total de casos), e com isso ter a possibilidade de tratar bases de dados de maiores dimensões em menos tempo.

O objectivo da selecção de atributos é semelhante ao da redução do número de casos só que a redução é feita através da eliminação de atributos irrelevantes.

Apresentaremos dois métodos de redução de dados, um para redução do número de casos, e o outro para selecção de atributos. Ambos os métodos foram avaliados, e assim apresentamos também as nossas conclusões acerca da sua utilidade.

A organização dos capítulos desta Tese, bem como uma descrição resumida dos seus conteúdos, é apresentada nos seguintes pontos:

- O Capítulo 1 contém uma visão geral sobre alguns conceitos da área em que se insere o nosso trabalho.
- O Capítulo 2 apresenta o problema da aprendizagem em conjuntos com um elevado número de casos junto com algumas soluções exploradas por alguns autores. Além disso apresentamos uma descrição do algoritmo que usámos para tentar resolver este problema.
- O Capítulo 3 contém a avaliação experimental do método descrito no capítulo anterior, apresentando os vários resultados, a análise dos mesmos e ainda as conclusões.
- O Capítulo 4 apresenta uma descrição do problema da aprendizagem em conjuntos com um elevado número de atributos irrelevantes junto com algumas soluções exploradas por alguns autores. Neste capítulo

apresentamos também uma descrição do algoritmo concebido para resolver este problema.

- O Capítulo 5 contém a avaliação experimental do método descrito no capítulo anterior, apresentando os vários resultados, a análise dos mesmos e ainda as conclusões.
- No Capítulo 6 apresentamos as conclusões finais; e aponta alguns desenvolvimentos das abordagens apresentadas em trabalhos futuros.

## Agradecimentos

Gostaria de agradecer ao meu Orientador, Professor Doutor Pavel Brazdil, por todo apoio que me deu ao longo deste trabalho, nomeadamente pelas suas dicas preciosas, pelo aconselhamento nas leituras mais relevantes, pelas ajudas na correção do texto, e especialmente pelo encorajamento que me deu em todas as fases deste trabalho, não me deixando desaninar.

Também gostaria de agradecer ao Joaquim Pinto da Costa, João Gama, Luis Torgo, Alípio Jorge, Carlos Leandro e Carlos Soares, pela ajuda que me deram, pelos artigos que me aconselharam bem como por observações relevantes para o meu trabalho.

Agradeço à Alzira Mota e ao José Novais pelas ajudas que me deram na fase da escrita.

Agradeço ainda ao Professor Doutor Altamiro da Costa Pereira por todo o apoio e compreensão que me deu no momento em que concorri a este Curso de Mestrado.

Agradeço finalmente, e com muito afecto, à minha mulher Fernanda, aos meus pais e meus sogros.



# Índice

<b>RESUMO .....</b>	<b>5</b>
<b>SUMMARY .....</b>	<b>7</b>
<b>RÉSUMÉ .....</b>	<b>9</b>
<b>Prefácio .....</b>	<b>11</b>
Agradecimentos .....	13
<b>Índice.....</b>	<b>15</b>
<b>Índice de tabelas.....</b>	<b>19</b>
<b>Índice de figuras.....</b>	<b>21</b>
<b>1 Extracção de Conhecimento de Bases de Dados (ECBD).....</b>	<b>23</b>
1.1 O processo de ECBD e suas fases .....	23
1.2 Prospecção de dados .....	24
1.3 Tipos de classificadores.....	25
1.4 Modelos na forma de árvores de decisão.....	26
1.5 Redes neurais.....	27
1.6 Avaliação dos modelos (contexto da classificação) .....	31
<b>2 Algoritmo de processamento por partições progressivas .....</b>	<b>35</b>
2.1 Motivação e trabalhos relacionados .....	36
2.2 Estratégias de redução do número de casos (estáticas versus dinâmicas).....	37
2.3 Estratégia de redução do número de casos dinâmica .....	37
2.4 Selecção aleatória e selectiva de casos .....	38
2.5 Que sequência de proporções devemos usar?.....	39
2.6 Que critério de paragem devemos usar?.....	40

2.7	O que concluir das curvas de aprendizagem?.....	41
2.7.1	Análise duma curva real .....	42
2.7.2	Resultados e interpretação .....	43
2.7.3	Critério de paragem .....	45
2.7.4	Outro critério utilizável.....	47
2.8	O algoritmo PPP .....	47
<b>3</b>	<b>Avaliação do algoritmo de processamento de partições progressivas.....</b>	<b>51</b>
3.1	Metodologia de avaliação.....	52
3.2	Conjuntos de dados de grande dimensão.....	55
3.2.1	Conjunto <i>TASKI</i> .....	55
3.2.1.1	Análise dos resultados obtidos .....	55
3.2.2	Conjunto sobre Cobertura Florestal .....	58
3.2.2.1	Análise dos resultados obtidos .....	58
3.2.3	Conjunto <i>Census</i> .....	60
3.2.3.1	Análise dos resultados obtidos .....	60
3.2.4	Conjunto <i>KDD-Cup 2000_CL</i> .....	61
3.2.4.1	Análise dos resultados obtidos .....	61
3.3	Conjuntos de dados de dimensão menor .....	62
3.3.1	Descrição dos conjuntos de dados .....	63
3.3.2	Análise dos resultados obtidos.....	63
3.4	Conjuntos de dados de dimensão menor (base MLP) .....	65
3.4.1	Análise dos resultados no conjunto <i>waveform</i> .....	66
3.4.2	Análise dos resultados no conjunto <i>letter</i> .....	67
3.5	Conclusões.....	68
<b>4</b>	<b>Selecção de atributos utilizando resultados do algoritmo PPP .....</b>	<b>71</b>
4.1	O problema dos atributos irrelevantes .....	71

4.2	Algoritmos mais e menos sensíveis aos atributos irrelevantes.....	72
4.3	Métodos de selecção de atributos .....	73
4.3.1	Selecção integrada no algoritmo (“embedded approach”) .....	73
4.3.2	Selecção anterior ao algoritmo (“Filter approach”) .....	74
4.3.3	Selecção guiada pelo algoritmo (“wrapper approach”) .....	74
4.4	O algoritmo de selecção de atributos baseado no algoritmo PPP.....	75
4.5	Considerações sobre o algoritmo.....	82
<b>5</b>	<b>Avaliação do algoritmo de selecção de atributos SAPPP .....</b>	<b>83</b>
5.1	Introdução.....	83
5.2	Metodologia de avaliação.....	84
5.3	Conjunto <i>TASK1</i> .....	86
5.3.1	Análise dos resultados obtidos.....	87
5.4	Conjunto sobre Cobertura Florestal.....	89
5.4.1	Análise dos resultados obtidos.....	89
5.5	Conjunto <i>Census</i> .....	91
5.5.1	Análise dos resultados obtidos.....	91
5.6	Conjunto <i>KDD-Cup2000_CL</i> .....	92
5.6.1	Análise dos resultados obtidos.....	92
5.7	Comparação do algoritmo SAPPP+PT com o algoritmo PPP.....	92
5.8	Conclusões.....	94
<b>6</b>	<b>Conclusões Finais .....</b>	<b>97</b>
6.1	Introdução.....	97
6.2	Motivação, objectivos e técnicas apresentadas.....	97
6.3	Conclusões.....	98
6.4	Trabalho futuro .....	101
	<b>REFERÊNCIAS.....</b>	<b>103</b>



# Índice de tabelas

Tabela 2-1 — Comparação do máximo global com o obtido pelo critério .....	46
Tabela 3-1 — Dados <i>TASK1</i> -Médias das taxas de acertos.....	57
Tabela 3-2 — Dados <i>Forest Cover Type</i> – médias das taxas de acertos.....	59
Tabela 3-3 — Dados <i>Census</i> – Médias dos tempos consumidos .....	60
Tabela 3-4 — Dados Census – Médias das taxas de acertos .....	61
Tabela 3-5 — Dados <i>KDD-Cup 2000_CL</i> – Médias dos tempos consumidos.....	62
Tabela 3-6 — Dados <i>KDD-Cup 2000_CL</i> – Médias das taxas de acertos .....	62
Tabela 3-7 — Descrição dos conjuntos de dados de menor dimensão.....	63
Tabela 3-8 — Médias das percentagens (de casos) escolhidas pelo algoritmo PPP .....	64
Tabela 3-9 — Dados <i>waveform</i> – Médias dos tempos .....	66
Tabela 3-10 — Dados <i>waveform</i> – Médias das taxas de acertos.....	66
Tabela 3-11 — Dados <i>letter</i> – Médias dos tempos.....	67
Tabela 3-12 — Dados <i>letter</i> – Médias das taxas de acertos .....	67
Tabela 5-1 — Dados <i>TASK1</i> -Médias das taxas de acertos .....	88
Tabela 5-2 — Dados <i>Forest Cover Type</i> – Médias das taxas de acertos.....	90
Tabela 5-3 — Dados <i>Census</i> – Médias dos tempos .....	91
Tabela 5-4 — Dados <i>Census</i> – Médias das taxas de acertos .....	91
Tabela 5-5 — Dados <i>KDD-Cup2000_CL</i> – Médias dos tempos.....	92
Tabela 5-6 — Dados <i>KDD-Cup2000_CL</i> – Médias das taxas de acertos .....	92
Tabela 5-7 — Comparação dos algoritmos SAPPP+PT e PPP .....	93



# Índice de figuras

Figura 1-1 — O processo de ECBD .....	23
Figura 1-2 — Exemplo de uma árvore de decisão.....	26
Figura 1-3 — “Caixa negra” representando uma rede com 5 entradas e 2 saídas .....	28
Figura 1-4 — Exemplo de uma rede neuronal multicamada .....	28
Figura 1-5 — Exemplo de um neurónio .....	29
Figura 2-1 — Algoritmo de “sampling” incremental .....	38
Figura 2-2 — Curva de aprendizagem.....	41
Figura 2-3 — Curva de aprendizagem em que sucede “overfitting” .....	42
Figura 2-4 —Curvas de aprendizagem .....	44
Figura 2-5 — Diferentes situações que podem surgir com um critério de 3 pontos .....	46
Figura 2-6 — Algoritmo de processamento por partições progressivas.....	48
Figura 3-1 — Dados <i>TASK1</i> – Rácio dos tempos ( $T_{PPP}/T_{PT}$ ).....	56
Figura 3-2 — Dados <i>TASK1</i> – Comparação dos tempos consumidos.....	56
Figura 3-3 — Dados <i>TASK1</i> – comparação das taxas de acertos .....	57
Figura 3-4 — Dados <i>TASK1</i> – Rácio das taxas de acertos ( $A_{PPP}/A_{PT}$ ) .....	57
Figura 3-5 — Dados <i>Forest Cover Type</i> – Rácio dos tempos ( $T_{PPP}/T_{PT}$ ) .....	58
Figura 3-6 — Dados <i>Forest Cover Type</i> – Curva de típica deste conjunto .....	59
Figura 3-7 — Ráciros dos tempos para os 8 conjuntos de dimensão menor .....	64
Figura 3-8 — Ráciros das taxas de acertos para os 8 conjuntos de dimensão menor ....	65
Figura 4-1 — Atributos seleccionados por uma sequência de Modelos.....	77
Figura 4-2 — Esquema de funcionamento do algoritmo de selecção de atributos (SAPP) .....	78

Figura 4-3 — Algoritmo de Seleção de Atributos (SAPPP).....	79
Figura 4-4 — Eliminação de atributos numa árvore de decisão.....	80
Figura 4-5 — Algoritmo para o cálculo do erro .....	81
Figura 5-1 — Dados <i>TASK1</i> - Rácio dos tempos .....	87
Figura 5-2 — Dados <i>TASK1</i> – comparação das taxas de acertos .....	88
Figura 5-3 — Dados <i>Forest cover type</i> – rácios dos tempos.....	89
Figura 5-4 — Forest cover type – taxa de acertos .....	90

# 1 Extracção de Conhecimento de Bases de Dados (ECBD)

Neste capítulo apresentamos alguns conceitos da área de ECBD por forma a melhor enquadrar o trabalho que realizámos nesta Tese. Vamos focar com maior pormenor os conceitos (e técnicas) que estão mais relacionados com os objectivos desta Tese, apenas referindo outros de forma muito sucinta. Os conceitos relacionados com a redução do número de casos bem como com a selecção de atributos não são apresentados neste capítulo, sendo introduzidos respectivamente nos capítulos 2 e 4.

## 1.1 O processo de ECBD e suas fases

O processo de ECBD segundo Fayyad [fay96] é um processo não trivial de identificar (procurar) nos dados padrões válidos, anteriormente desconhecidos, potencialmente úteis e interpretáveis. Este processo é constituído por várias fases, nomeadamente; a selecção e preparação de dados; transformação de dados incluindo a redução da dimensão dos dados; a prospecção de dados que consiste na extracção de padrões (utilizando os algoritmos de aprendizagem) que expliquem os dados e forneçam modelos de previsão; a avaliação e interpretação dos modelos obtidos. Podemos observar as fases do processo ECBD na seguinte Figura [fay96]

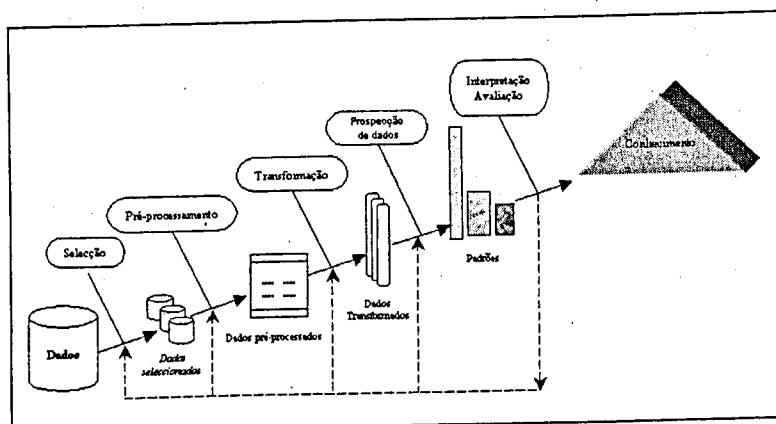


Figura 1-1 — O processo de ECBD

A fase inicial consiste em seleccionar da base de dados os dados alvo, em conformidade com o que pretendemos analisar nas fases seguintes do processo de ECBD. Segue-se

uma fase de pré-processamento que realiza a remoção de ruído e valores anormais (“outliers”), bem como tratamento de valores desconhecidos. Quanto à transformação de dados, essas incluem as operações de redução da dimensionalidade de dados, através da redução do número de casos (“case reduction”), e a selecção de atributos (variáveis) mais relevantes (“feature selection”) dependendo do objectivo pretendido e redução do número de valores possíveis por atributo (“value reduction”). Além da selecção de atributos, poderão ser usados métodos de construção de novos atributos (utilizando os já existentes) como os métodos de indução constructiva e ainda o método das componentes principais.

Na fase seguinte estão os métodos de prospecção de dados (“data mining”), que procuraram padrões interessantes nos dados. A interpretação e avaliação dos padrões encontrados é realizada na última fase. Os critérios de avaliação podem diferir de uma aplicação para outra e incluir, por exemplo, uma medida da capacidade dos padrões encontrados para efectuar previsões correctas, bem como serem inteligíveis pelos utilizadores. O processo pode voltar a fases anteriores em função dos resultados da avaliação não serem ainda satisfatórios.

Nesta Tesé vamos focar a fase de transformação e particularmente os métodos úteis na redução da dimensionalidade de dados. Antes disso vamos descrever em maior pormenor as fases de prospecção e de avaliação nas duas próximas secções.

## 1.2 Prospecção de dados

A prospecção de dados é às vezes associada à procura de padrões nos dados. Em função do tipo de problema que pretendemos resolver, a procura de padrões subdivide-se em vários tipos de tarefas.

São exemplos (entre outros) de tarefas da prospecção de dados, a classificação, a regressão, o agrupamento (“clustering”) e procura de relações de dependência (regras de associação), e o resultado é às vezes chamado “modelo”.

Uma das principais aplicações dos modelos construídos na fase de prospecção de dados é a previsão. O objectivo da previsão é prever o valor de um determinado atributo alvo, em situações em que é desconhecido (ex. previsão da temperatura para o dia seguinte).

A classificação e a regressão são tarefas que normalmente estão associadas ao objectivo da previsão. Alguns autores utilizam o termo “previsão” para referir o resultado da aplicação de um sistema de regressão para prever um valor numérico. A previsão de valores nominais é muitas vezes referido de *classificação*.

No caso da classificação o atributo alvo tem um número finito de valores nominais (sem ordenação), e no caso da regressão o mesmo assume valores num domínio contínuo (valores reais). Na próxima secção e porque nesta Tese apenas abordamos os problemas da classificação, vamos descrever quais os tipos principais de classificadores.

### 1.3 Tipos de classificadores

Existem vários tipos de classificadores para os quais existem algoritmos também de tipo diferente. Esses algoritmos são divididos por três tipos [WeiIn98;BerHand99] que descrevemos nos seguintes pontos:

- Os estatísticos descrevem o modelo de classificação através de operações matemáticas aos valores dos atributos dos casos a classificar. São exemplos destes métodos o discriminante linear e as redes neuronais [BerHand99].
- Os algoritmos baseados em medidas de distância realizam a classificação de um caso novo procurando casos similares (utilizando uma medida de distância) no conjunto dos já conhecidos, usando estes últimos para determinar a classificação. A aprendizagem utilizando estes algoritmos é denominada preguiçosa (“lazy learning”) [Mitch97], pois o classificador só é construído quando pretendemos classificar um novo caso. Um algoritmo deste tipo é o chamado algoritmo dos k-Vizinhos mais próximos (“k- Nearest neighbors”).
- Os algoritmos simbólicos (de aprendizagem automática “Machine Learning”) descrevem o modelo de classificação através de expressões condicionais que resultam em valores de verdadeiro ou falso, e cujos valores servem para discriminar os casos por classes. São exemplos destes classificadores as árvores de decisão, as regras de decisão e as tabelas de decisão [Mitch97;WeiIn98].

Nas duas próximas secções descrevemos os modelos dos tipos árvore de decisão e rede neuronal.

#### 1.4 Modelos na forma de árvores de decisão

Uma árvore de decisão (ver exemplo na Fig. 1-2) é constituída por *nós de decisão* (representada por rectângulos) e *folhas* (representadas por círculos). Para classificar um determinado caso é preciso percorrer a árvore de cima para baixo, partindo do nó do topo até atingir uma folha. A folha atingida por este percurso contém a classe atribuída a este caso.

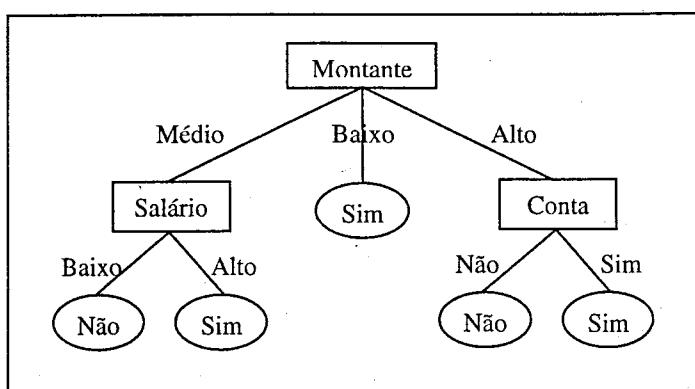


Figura 1-2 — Exemplo de uma árvore de decisão

O caminho descendente é determinado pelo valores nos atributos do caso a classificar e pelas condições existentes nos nós de decisão.

Em cada nó de decisão é testado o valor de um atributo (ex. no nó do topo testa-se o valor do montante). Se o atributo testado é *discreto* então existem tantas derivações quantos os valores possíveis para esse atributo, e é escolhido o ramo correspondente ao valor existente nesse atributo (no caso que pretendemos classificar). Se o atributo é *contínuo* então o teste correspondente tem apenas dois ramos que dividem os valores em dois intervalos definidos por um determinado valor de corte C, um para os valores menores que C e outro para os valores maiores ou iguais a C. Se estamos a testar um atributo contínuo escolhemos o ramo de acordo com o valor do atributo testado.

Uma árvore de decisão é em geral construída por um processo denominado de partição recursiva (“recursive partitioning”) dos casos. Em cada passo deste processo procura-se encontrar o atributo que melhor discrimina os casos do conjunto de dados. O método habitualmente usado na procura é o “subir a colina” (“hill climbing”).

## Capítulo 1

Depois de escolhido o atributo (no caso de ser contínuo procura-se também o melhor valor de corte), os dados são divididos pelos valores do teste desse atributo, e para cada subconjunto repete-se novamente este passo.

Uma forma de procurar o atributo que melhor discrimina os casos do conjunto de Treino utiliza uma medida baseada no *ganho de informação* [Qui93;Mit97].

Se E representar o conjunto de dados para o qual pretendemos encontrar o atributo que melhor discrimina os casos (em classes diferentes) e A representar um atributo genérico então o ganho de informação é definido por

$$Ganho(E, A) \equiv entropia(E) - \sum_{v \in valores(A)} \frac{|E_v|}{|E|} entropia(E_v)$$

em que a entropia é definida por

$$entropia(E) \equiv \sum_i^c - p_i \log_2 p_i$$

e  $p_i$  representa a proporção de casos no conjunto  $E$  que pertencem à classe  $i$ , e  $p_i$  representa a proporção de casos no conjunto  $E$  que pertencem à classe  $i$ ,  $valores(A)$  representa o conjunto de valores possíveis para o atributo  $A$  e  $E_v$  é o subconjunto de  $E$  no qual o atributo  $A$  tem o valor  $v$ .

O atributo escolhido para testar corresponde ao que tem um ganho de informação mais elevado.

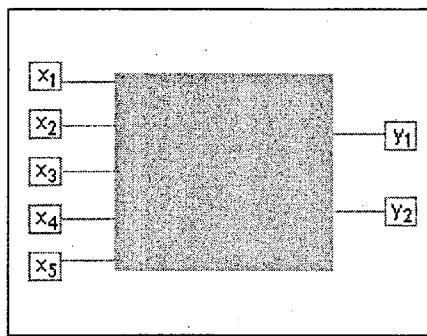
### 1.5 Redes neurais

As redes neurais são outro tipo de modelo que poderá ser utilizado tanto no contexto da classificação como no contexto da regressão.

Existem vários tipos de redes neurais que naturalmente utilizam algoritmos de aprendizagem específicos [PrincipeJ00]. Nesta Tese vamos abordar apenas um tipo de rede neuronal que se poderá denominar de rede perceptrões em multicamadas (“*Multilayer perceptron*”), que designaremos de forma abreviada por MLP.

As redes neuronais têm analogias com o sistema nervoso biológico e consiste num conjunto de unidades, denominadas de neurónios, que se encontram interligadas.

Uma rede neuronal é uma função que mapeia um conjunto de entradas num conjunto de saídas. Na próxima figura representamos através duma caixa negra o que representa uma rede neuronal, sendo as entradas representadas por  $x_1, x_2, x_3, x_4, x_5$  e as saídas por  $y_1$  e  $y_2$ .

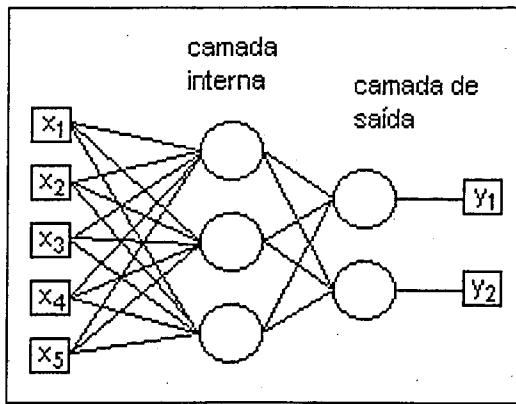


**Figura 1-3 — “Caixa negra” representando uma rede com 5 entradas e 2 saídas**

A rede neuronal é no fundo uma função  $F$  tal que  $F(x_1, x_2, x_3, x_4, x_5) = (y_1, y_2)$ .

A seguir vamos observar o que está no interior da caixa negra.

A Figura 1-4 ilustra um exemplo de uma rede neuronal, do tipo MLP, constituída por 5 neurónios.



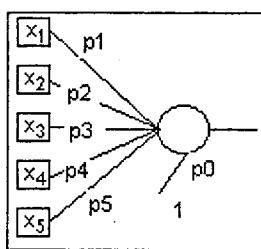
**Figura 1-4 — Exemplo de uma rede neuronal multicamada**

Neste tipo de redes existem entradas (na figura  $x_1, \dots, x_5$ ), saídas (na figura  $y_1$  e  $y_2$ ) e algumas unidades (chamadas neurónios) dispostos em várias camadas. A última camada de neurónios é denominada de camada de saída, sendo todas as outras denominadas camadas internas ( $1^{\text{a}}$  camada interna,  $2^{\text{a}}$  camada interna,...). As entradas da rede

neuronal estão ligadas à 1<sup>a</sup> camada interna, e se houver mais camadas, cada camada i é ligada à camada i+1.

A unir os neurónios existem ligações que possuem determinados pesos (ex. p<sub>1,3</sub> é o peso da ligação entre o neurónio 1 e o neurónio 3). Nas redes do tipo MLP os neurónios dumha camada encontram-se ligados aos neurónios da camada seguinte, as entradas encontram-se ligados aos neurónios da 1<sup>a</sup> camada e os neurónios da última camada estão ligados às saídas.

O valor de saída num determinado neurónio depende dos valores de saída dos neurónios que o antecedem. Para mostrar como se pode calcular os valores de saída de um determinado neurónio, usamos a seguinte figura para descrever como se processa para um neurónio em particular.



**Figura 1-5 — Exemplo de um neurónio**

Para calcular qual o valor de saída deste neurónio começamos determinar a soma pesada dos valores  $x_i$  com os pesos respectivos  $p_i$ . A esta soma acrescenta-se ainda um valor  $p_0$  (normalmente referido por *bias*). Depois usa-se esta soma como argumento de uma função que se denomina de função de activação. Normalmente como função de activação é utilizada a função sigmoide que se define da seguinte forma:

$$\text{sigmoide}(x) = \frac{1}{1 + e^{-x}}$$

Poderão ser ainda utilizadas outras funções de activação, por exemplo a função *tanh* (tangente hiperbólica) ou uma *função linear*. Referimos apenas que no caso toda a rede neuronal usar funções de activação lineares implica que a rede representava uma função cujas saídas seriam combinações lineares dos valores de entrada, o que impediria a utilização da rede em situações de classificação quando as classes não são linearmente separáveis.

Se a função de activação do neurónio que descrevemos anteriormente for a sigmoidal então a sua saída seria:

$$\text{saída do neurónio} = \frac{1}{1 + e^{-\left(p_0 + \sum_{i=1}^5 x_i p_i\right)}}$$

O exemplo que acabámos de mostrar é perfeitamente extensível (naturalmente com outras entradas e pesos) a outros neurónios existentes na rede neuronal que estamos a focar. É fácil determinar o valor das saídas sendo dados os valores das entradas, basta para isso calcular o valor de saída para cada neurónio. No caso de um neurónio ter como entrada a saída de outro neurónio precisamos de a calcular previamente.

O objectivo da aprendizagem numa determinada rede consiste na determinação dos pesos das ligações (e o peso  $p_0$  “bias”) entre os neurónios de forma a melhor se ajustar a um determinado conjunto de exemplos de treino. A aprendizagem consiste em procurar no espaço dos pesos um ponto que minimize o erro cometido pela rede considerando os exemplos de treino.

Perante o conjunto de exemplos do conjunto de treino podemos determinar o erro (total dos desvios quadráticos) cometido pela rede usando a seguinte expressão:

$$\text{erro} = \sum_{i=1}^{n \text{ de exemplos}} (f(x_i) - o_i)^2$$

em que  $X_i = (x_{i,1}, x_{i,2}, \dots, x_{i,n})$  representa os valores dos vários atributos do exemplo  $i$ , e  $o_i$  representa o valor do atributo objectivo (que queremos representar como função dos primeiros atributos).

A aprendizagem neste tipo de redes consiste num processo de minimização do erro anteriormente definido. Para um determinado conjunto de exemplos, o erro pode ser definido como uma função cujo domínio tem como dimensão o número de pesos existentes na rede. Se as funções de activação forem diferenciáveis então também o erro é uma função diferenciável, pelo que podemos procurar minimizar essa função através

dum processo de descida utilizando o gradiente. O algoritmo de aprendizagem mais comum para “treinar” redes neuronais MLP é o algoritmo de retropropagação do erro (“*Backpropagation*”) e baseia-se no método de descida pelo gradiente. O algoritmo consiste em ir ajustando os pesos segundo a direcção contrária ao gradiente, isto é, se a componente do gradiente correspondente ao peso  $p_i$  é  $d_i$  então o peso é ajustado usando a seguinte fórmula:

$$p_i \leftarrow p_i - \eta \cdot d_i$$

em que  $\eta$  é um parâmetro positivo próximo de zero.

A mesma transformação é feita para todos os pesos, e repetida até que o erro seja tão baixo quanto o pretendido. O algoritmo vai ajustando os pesos ao longo de várias iterações, através de pequenas modificações devido ao valor parâmetro  $\eta$  ser reduzido. Por este facto o algoritmo precisa normalmente de muitas iterações para aprender uma rede, e por isso é *um algoritmo bastante lento*.

Além do problema do tempo o algoritmo de aprendizagem sofre ainda do inconveniente do processo de minimização poder ficar aprisionado num mínimo local. Este problema é causado pelo facto do gradiente num determinado ponto dar a direcção da maior descida na vizinhança do ponto e portanto ser zero se o ponto for um mínimo local, sendo pois impossível distinguir um mínimo local de um mínimo global.

Quanto aos domínios de aplicação as redes neuronais têm sido bastante eficientes em problemas de aprendizagem muito complexos como o reconhecimento de imagens e de voz. Em resumo os dois principais inconvenientes destes métodos de aprendizagem são o elevado tempo despendido no processamento, e o facto de serem de difícil interpretação pois são descritos por um elevado número de parâmetros numéricos (pesos).

## 1.6 Avaliação dos modelos (contexto da classificação)

A avaliação dos modelos tem como objectivo principal determinar a probabilidade de classificar correctamente um determinado caso que não é utilizado na construção do modelo de classificação. No fundo estamos a verificar o sucesso que o modelo tem na previsão da classe de um caso novo. O sentido que damos a palavra *previsão* não é necessariamente a previsão de um caso no futuro. Algumas bases de dados constituem

apenas amostras do verdadeiro conjunto que pretendemos analisar, isto é, um caso novo é simplesmente um caso não existente no conjunto de dados que foi utilizado na aprendizagem.

A questão que se pode colocar é como podemos quantificar o possível sucesso do classificador em novos casos? Existem vários métodos de avaliação da capacidade predictiva de um determinado modelo de classificação.

Os métodos baseiam-se em medições empíricas (através de experiências) da *taxa de acertos*.

Um princípio que se assume é que não devemos utilizar para medir a taxa de acertos o mesmo conjunto de dados que usámos para construir o modelo. Se o fizermos estaremos a obter estimativas demasiado optimistas (“enviesadas”) da taxa de acertos real (probabilidade de acertar).

Vamos descrever nos seguintes pontos algumas das metodologias de avaliação [Mitch97;BerHand99] mais usadas:

- Método de TREINO e de TESTE.

A ideia é dividir o conjunto de dados inicial em dois subconjuntos, um chamado conjunto de TREINO que é utilizado para construir o modelo, e outro chamado conjunto de TESTE que é utilizado para testar o modelo. Os casos do conjunto de teste são classificados utilizando o modelo previamente construído e são comparadas as classificações previstas com as verdadeiras classificações. Este método tem a vantagem de consumir menos tempo que o método de validação cruzada apresentado a seguir.

- Método de Validação cruzada (“cross validation”)

O conjunto de dados é dividido em n subconjuntos ( $A_1, A_2, \dots, A_n$ ), cada subconjunto com aproximadamente o mesmo número de casos, sendo disjuntos dois a dois e tal que a reunião cobre todo o conjunto de dados.

A avaliação é feita através de um ciclo ( $i=1$  até  $n$ ) que percorre todos os subconjuntos  $A_i$ , utilizando  $A_i$  como conjunto de teste e a reunião dos outros

subconjuntos como conjunto de treino. Obtemos assim n estimativas para a taxa de acertos que normalmente são agregadas através da média aritmética.

- Método Deixa-um-de-fora (“leave-one-out”)

Este método é um caso particular da validação cruzada em que consideramos tantos subconjuntos como o número de casos.

Neste trabalho usamos quase sempre o método de “validação cruzada” para avaliar os modelos.

No próximo capítulo vamos descrever o problema do processamento de conjuntos de dados com elevado número de casos. O objectivo principal é apresentar o algoritmo de redução do número de casos que foi utilizado no nosso trabalho.



## 2 Algoritmo de processamento por partições progressivas

O processamento de conjuntos de dados com um elevado número de casos constitui um problema a ter em conta quando utilizamos os algoritmos de aprendizagem. Alguns algoritmos são demasiado lentos, tornando o processamento de grandes conjuntos um processo muito ineficiente. Mesmo os algoritmos mais rápidos poderão consumir demasiado tempo no processamento, pois há aplicações que necessitam resultados rápidos. Também os problemas determinados pelos limites de memória dos computadores poderão constituir um obstáculo intransponível, quando os conjuntos são muito grandes.

Para resolver o problema citado acima podemos utilizar métodos que reduzam o número de casos, diminuindo dessa forma o esforço de processamento necessário aos algoritmos de aprendizagem.

Uma solução desejável, do ponto de vista do utilizador, consiste em obter em períodos de tempo pequenos alguns modelos prévios com resultados razoáveis, isto é com taxas de acertos aceitáveis, sabendo que o sistema pode tentar construir modelos progressivamente melhores (filosofia “anytime results”).

Neste capítulo vamos abordar o problema do processamento de conjuntos de dados com um elevado número de casos, referindo algumas soluções já exploradas, e ainda descrever um método escolhido para fazer face a este problema.

Na próxima secção abordamos as principais motivações dos métodos de redução de dados, nomeadamente a redução do número de casos (“sampling”) bem como alguns trabalhos relacionados com este problema.

## 2.1 Motivação e trabalhos relacionados

Iremos a partir deste ponto considerar que os métodos do processo ECBD referidos no Capítulo 1 como *métodos de transformação*, são também uma forma de *métodos de pré-processamento* visto que correm antes da fase de prospecção de dados (“data mining”). Nesse sentido podemos afirmar que, com a crescente necessidade de aplicação de algoritmos de aprendizagem a grandes conjuntos de dados, a utilização de métodos de pré-processamento que reduzam a dimensão, tem vindo a tornar-se num problema fundamental neste processo.

Os grandes conjuntos de dados constituem para os algoritmos de aprendizagem um sério problema, visto que estes por vezes demoram demasiado tempo a tratar os dados, e outras vezes, por via de problemas de memória, nem conseguem concluir a construção do modelo.

A dimensão de um conjunto de dados pode ser vista segundo três facetas [WeiIn98]: o número de casos, o número de atributos, e o número de valores que os atributos podem tomar.

O problema que vamos abordar neste capítulo é a redução do número de casos (“sampling”), isto é, se considerarmos a habitual forma tabelar das bases de dados tratadas pelos algoritmos de aprendizagem, o problema consiste em reduzir o número de linhas sem afectar a *performance* do modelo construído.

Focamos o nosso problema apenas na classificação e portanto nos algoritmos que constroem modelos classificadores.

Uma questão pertinente é saber se realmente podemos construir modelos de classificação utilizando um subconjunto próprio dos dados disponíveis, e se, com esse subconjunto conseguimos obter, de forma mais rápida, um classificador que tenha uma boa taxa de acertos. Esclareçamos que esta taxa de acertos é obtida através da avaliação do modelo num subconjunto dos dados (TESTE) que não foi utilizado para gerar (“aprender”) o modelo, e calcula-se, dividindo o número de casos bem classificados pelo número total de casos.

## 2.2 Estratégias de redução do número de casos (estáticas versus dinâmicas)

Outra questão não menos importante é como podemos encontrar tal subconjunto, ou seja, que estratégia de redução do número de casos vamos usar. Duas categorias de estratégias de foram descritas e comparadas por John e Langley [John96], que avaliaram a estratégia estática *versus* a dinâmica, concluindo com a preferência pela segunda.

A ideia fundamental da *estratégia estática* é determinar se o subconjunto é “suficientemente similar”, do ponto de vista estatístico, ao conjunto inicial. Utiliza-se nesta estratégia uma forma de quantificar se o subconjunto considerado provém da mesma população que o conjunto inicial. A designação *estática* provém do facto de o processo não depender do algoritmo de aprendizagem que vai ser usado na fase posterior.

Na *estratégia dinâmica*, são gerados vários subconjuntos (candidatos) e para cada um deles é construído um modelo usando o algoritmo de aprendizagem. O processo de escolha do subconjunto “ideal” é guiado pelos resultados (taxa de acertos) dos modelos construídos, sendo portanto, ao contrário da estratégia anterior, dependente do algoritmo de aprendizagem, designando-se por isso estratégia dinâmica. Estes tipos de abordagens são normalmente designadas pelo termo “wrapper approach” [Kohavi95a].

## 2.3 Estratégia de redução do número de casos dinâmica

Um exemplo de processo de redução do número de casos, usando esta estratégia, é o chamado “sampling” incremental e é expresso de forma simples na Figura 2-1 (adaptada de [WeiIn98]).

Neste processo são gerados subconjuntos progressivamente maiores e é feita uma comparação das taxas de acertos obtidas pelos modelos gerados. Esta comparação é feita entre dois subconjuntos consecutivos e serve para decidir se o processo pára, isto é, serão gerados subconjuntos com proporções progressivamente maiores enquanto a taxa de acertos for aumentando. O subconjunto a usar será o que conduziu à maior taxa de acertos.

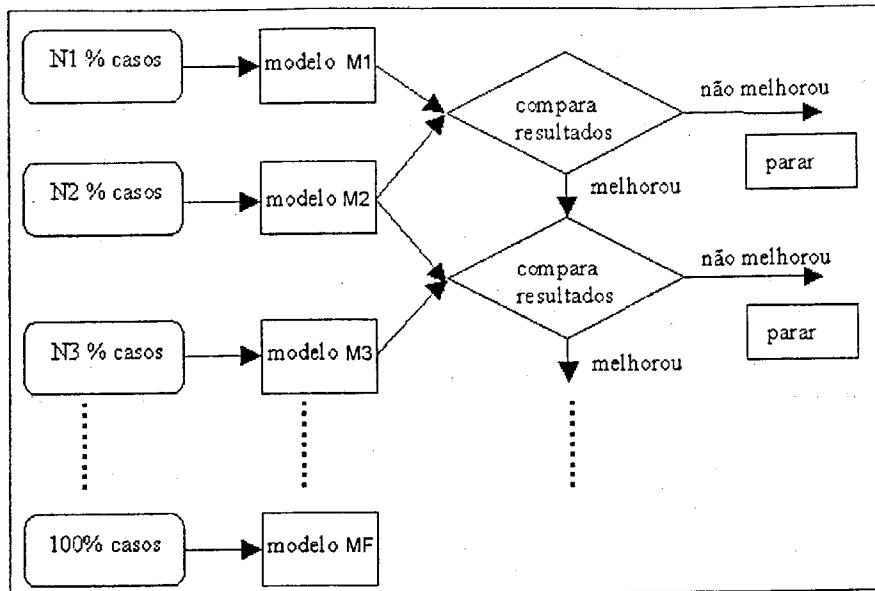


Figura 2-1 — Algoritmo de “sampling” incremental

Se  $S_i$  forem os subconjuntos gerados,  $M_i$  os modelos respectivos e  $A_i$  as taxas de acertos dos modelos  $M_i$ , as condições “compara resultados” que aparecem no esquema poderiam ser, por exemplo, da forma  $A_{i+1}-A_i < 0$  (a taxa começou a descer), ou mesmo  $A_{i+1}-A_i < \varepsilon$  (a taxa está subir mas pouco), em que  $\varepsilon$  é uma pequena margem de tolerância. O algoritmo “sampling” incremental vai procurar encontrar um valor  $P_{\min}$  que representa a proporção mínima de casos a usar, isto é, para subconjuntos de proporções maiores que  $P_{\min}$  a taxa de acertos não “melhora”.

## 2.4 Selecção aleatória e selectiva de casos

A geração de um subconjunto é geralmente obtida por amostragem aleatória. Outra forma de geração, denominada *estratificada*, é utilizada quando existe um grande desequilíbrio na distribuição das classes na base de dados. Neste caso a selecção é aleatória mas com a imposição que o subconjunto gerado tenha uma distribuição de classes o mais próxima possível à do conjunto inicial.

Para além da escolha aleatória de casos, existem outras formas mais selectivas de geração dos subconjuntos. Poderíamos, por exemplo, usar o “*windowing*” [Quin93] que

privilegia a selecção dos casos mal classificados num classificador previamente construído.

Poderíamos ainda usar a técnica que é referida por Mamitsuka e Abe em [Mamit00], que utiliza a noção de “uncertainty sampling”, que consiste em escolher os casos com maior “incerteza” nas futuras classificações.

## 2.5 Que sequência de proporções devemos usar?

A sequência de proporções (scheduling) N1% N2% N3% ... 100%, apresentada na Figura 2-1, é uma sequência crescente e poderá ser de vários tipos: progressão aritmética, geométrica ou ainda qualquer outra progressão crescente.

A questão de saber que sequência de proporções devemos usar relaciona-se com a eficácia do algoritmo de redução do número de casos. O tempo consumido pelo algoritmo (da Figura 2-1), que designaremos de *tempo acumulado*, é a soma dos tempos necessários para construir todos os modelos da sequência de subconjuntos, isto é, a sequência gerada até ao processo parar. Este tempo vai ser extremamente dependente da sequência de proporções que usarmos. Mas que sequência de proporções devemos usar de forma a que o processo seja eficiente? Devemos usar uma sequência do tipo *aritmética*, isto é da forma

$$p_n = p_0 + (n-1) r$$

em que  $p_0$  representa a proporção inicial e  $r$  o incremento da progressão, ou devemos usar uma progressão *geométrica*, isto é da forma

$$p_n = p_0 r^{n-1}$$

em que  $p_0$  representa a proporção inicial e  $r$  a razão da progressão, ou devemos usar uma sequência crescente geral?

Provost, Jensen e Oates no artigo “Efficient Progressive Sampling” [Prov99] respondem à pergunta anterior sugerindo a utilização de sequências segundo uma progressão geométrica, quando o algoritmo de aprendizagem tem complexidade temporal superior

ou igual à linear ( $O(n)$ ). Demonstraram que este processo progressivo utilizando este tipo de sequência é assintoticamente (mesma ordem) tão rápido como a aprendizagem utilizando o conhecimento prévio da proporção de dados a usar, isto é, assintoticamente o tempo não depende senão do número “mínimo” de exemplos necessários à obtenção de modelos de classificadores bons no contexto pretendido.

Pelo motivo anterior optámos pela progressão geométrica da proporções dos subconjuntos. Sendo assim, temos que determinar que valores devem ter os parâmetros  $p_0$  e  $r$  na sequência de proporções geométrica referida.

A questão da determinação, e discussão, dos “melhores” valores para os parâmetros  $p_0$  e  $r$  a utilizar é uma questão que não tem uma resposta fácil. É intuitivo que a optimalidade destes parâmetros vai depender bastante do conjunto de dados que pretendemos tratar, e que tentar obter valores dos parâmetros óptimos para a globalidade dos conjuntos implicaria correr muitas experiências em muitos conjuntos de dados diferentes. Por esse motivo decidimos fazer um estudo mais limitado, para podermos fixar os parâmetros que iremos utilizar no algoritmo de redução de casos.

Vamos utilizar o valor 2 para  $r$ , isto é a proporção cresce para o dobro em cada iteração, e ainda o valor 1,5.

Quanto ao valor de  $p_0$  (proporção inicial) vamos considerar 1%, isto é 0,01.

Usamos a sigla PPP para identificar o algoritmo que temos vindo a referir neste capítulo. O significado de PPP é algoritmo de *processamento por partições progressivas* (PPP).

## 2.6 Que critério de paragem devemos usar?

Com o intuito de obter um critério simples de paragem para o algoritmo que estamos a considerar, conduzimos um estudo preliminar para observar curvas de aprendizagem reais. A simplicidade do critério foi auto imposta, visto que o objectivo de Tese era a combinação de métodos de pré-processamento, e queríamos sobretudo combinar métodos de redução do número de casos com métodos de selecção de atributos.

Na secção seguinte apresentamos a experiência realizada, bem como o critério de paragem que iremos utilizar no algoritmo de processamento por partições progressivas que vamos apresentar neste capítulo.

## 2.7 O que concluir das curvas de aprendizagem?

Consideramos o problema da “aprendizagem” de um classificador, utilizando um dado algoritmo de indução, aplicado a um conjunto de dados. Uma *curva de aprendizagem* representa a evolução da taxa de acertos à medida que utilizamos subconjuntos com proporção cada vez maiores [Prov99]. A avaliação da taxa de acertos é feita utilizando um conjunto de teste diferente do conjunto de dados utilizado na aprendizagem. Em aplicações reais, quando os conjuntos de dados são grandes, a curva de aprendizagem em geral tem uma forma semelhante à ilustrada pela Figura 2-2.

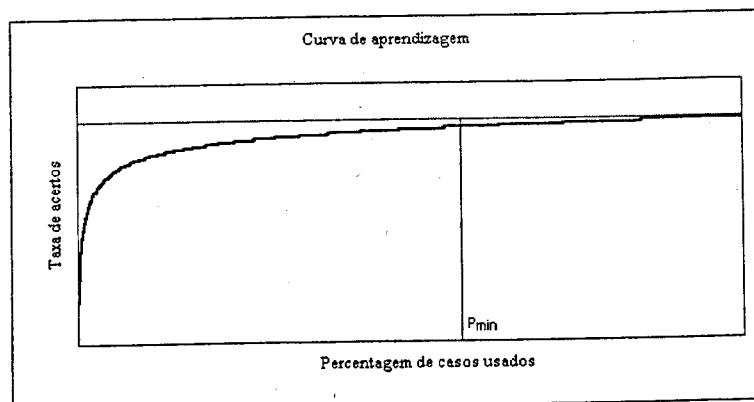


Figura 2-2 — Curva de aprendizagem

Nestas curvas existem três fases de comportamento. Uma primeira fase de subida rápida, uma segunda fase de subida mais lenta e uma terceira fase, denominada planalto, em que a taxa de acertos aumenta pouco. Esta fase de planalto (ou máximo) corresponde ao gráfico posterior a  $P_{\min}$  (Figura 2-2) em que a taxa de acertos “aumenta pouco”. É exactamente em curvas deste género que faz sentido reduzir o número de casos, pois se não existisse a fase de planalto, deveríamos usar 100% dos dados. Observemos que o significado que damos à expressão “aumenta pouco” é um critério bastante dependente da aplicação. A título de exemplo podemos referir que para algumas aplicações pode ser pouco relevante ganhos de 1% na taxa de acertos, enquanto noutras poderão ser importantes ganhos de 0,001%.

Alternativamente poderá acontecer que a curva de aprendizagem não apresente um verdadeiro planalto mas antes um máximo (Figura 2-3). Nestes casos depois de atingido o máximo a taxa de acertos diminui. A explicação para esta diminuição é o chamado sobreajustamento (“overfitting”), ou seja, o taxa de acertos no conjunto de teste vai aumentando até a um certo ponto em que começa a haver um sobreajustamento ao conjunto de treino.

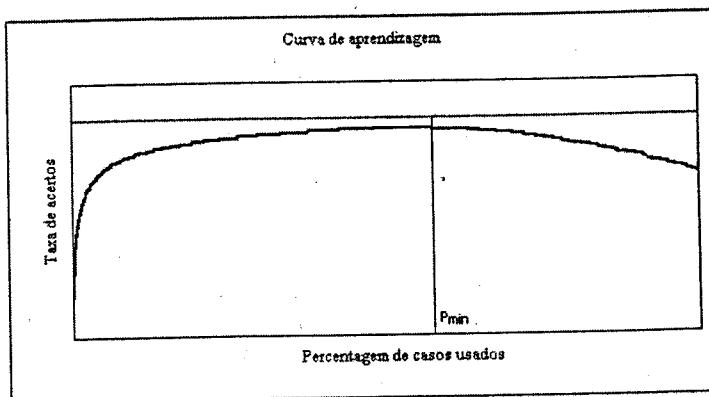


Figura 2-3 — Curva de aprendizagem em que sucede “overfitting”

A questão que se coloca é como detectar o planalto ou o máximo? Antes de tentar conceber o critério de paragem do método, decidimos analisar curvas reais.

### 2.7.1 Análise duma curva real

Para traçar a curva de aprendizagem, aplicámos o C5.0 [Quin98], conhecido algoritmo de indução de árvores de decisão, a uma sucessão de subconjuntos dum conjunto de dados chamado *TASK1*, e observámos a taxas de acertos obtidas.

O conjunto *TASK1* é uma das tabelas que constituem a base de dados SISYPHUS. Esta base provém dum repositório de dados (“data-warehouse”) chamado MASY, e é utilizada na recolha e análise de dados sobre seguros de vida da companhia de seguros *Swiss Life*. Este conjunto (*TASK1*) é constituído por 111077 casos, cada com 72 atributos, havendo 1 atributo objectivo (9 classes), 26 atributos de domínio contínuo, e 45 de domínio discreto

## Capítulo 2

Começámos por dividir este conjunto em dois subconjuntos, TREINO e TESTE, obtidos por “amostragem” aleatória estratificada, com proporções  $\frac{2}{3}$  e  $\frac{1}{3}$  respectivamente.

O conjunto TESTE serviu para calcular as taxas de acertos obtidas. O conjunto de TREINO serviu como base para a extracção dos subconjuntos de várias proporções. Extraímos do conjunto TREINO, de forma aleatória e estratificada, 10 subconjuntos com 1%, 5%, 10%, 20%, 30%, 40%, 50%, 60%, 70% e 80% de casos. A cada um desses subconjuntos aplicámos o C5.0 obtendo 10 classificadores. Avaliamos a taxa de acertos desses classificadores, utilizando o conjunto TESTE. A sequência de proporções utilizadas não é uma progressão geométrica, o que neste caso não é muito importante, pois o objectivo é apenas desenhar a curva de aprendizagem.

Repetimos este procedimento 5 vezes, embaralhando o conjunto *TASK1* e obtendo assim diferentes conjuntos TREINO e TESTE. Traçámos desta forma 5 curvas de aprendizagem utilizando os resultados das experiências.

### 2.7.2 Resultados e interpretação

As curvas que obtivemos estão expressas na Figura 2-4. Podemos observar que as mesmas apresentam muitas variações. De onde provém estas variações?

A taxa de acertos associada a um determinado modelo (classificador) representa a probabilidade de acerto na classificação de um novo caso. As taxas de acertos que obtivemos na experiência são apenas *estimativas* dessas probabilidades de acerto, e têm por isso associadas um determinado erro. Na realidade poderíamos enquadrar a verdadeira taxa de acertos através do seguinte intervalo com 95% de confiança:

$$\text{taxa de acertos} = [\text{taxa estimada} - \delta, \text{taxa estimada} + \delta]$$

$$\text{em que } \delta = 1,96 \times \sqrt{\frac{\text{taxa estimada} \times (1 - \text{taxa estimada})}{N}}$$

N é o número de casos do conjunto de teste

Além desta explicação para a variação existe ainda o argumento que o algoritmo de aprendizagem utilizado (aprendizagem de árvores de decisão) é instável no sentido em

que pequenas variações no conjunto de treino podem resultar em grandes variações [Breiman96] no classificador construído.

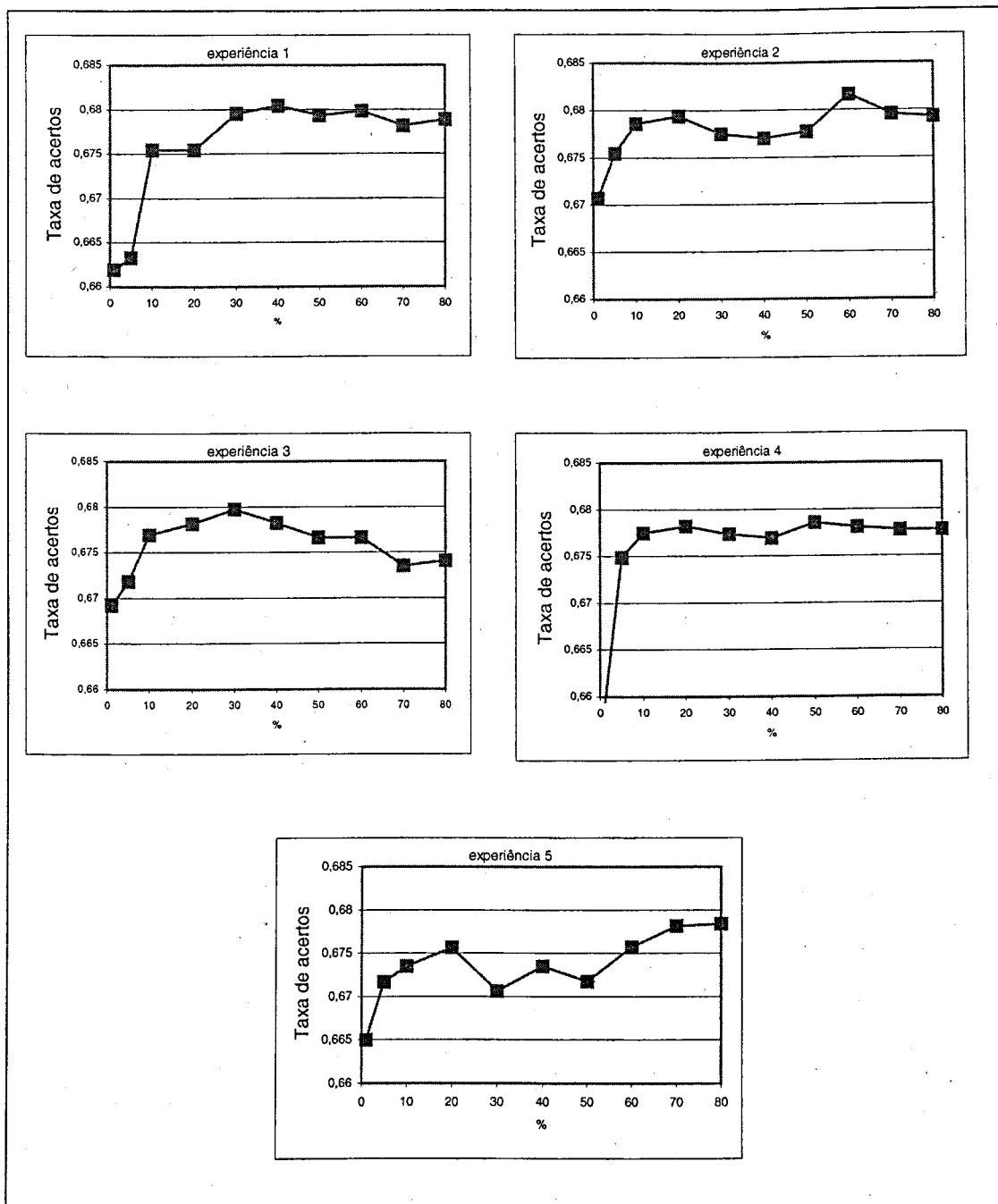


Figura 2-4 —Curvas de aprendizagem

Nas curvas obtidas das experiências individuais, tanto a “experiêncial”, como a “experiênci4” têm um comportamento bastante mais estável que as outras, reconhecendo-se um máximo no ponto 40% na “experiêncial” e no ponto 50% na “experiênci4”.

Na "experiência2" o máximo é atingido em 60%. Na "experiência3", talvez devido a sobreajustamento, observa-se que é atingido o máximo em 30%, e que depois a curva desce até ao fim. Na "experiência5" o máximo é atingido com 100% dos casos.

Podemos dizer que o nosso critério de paragem terá que identificar o máximo ou planalto de curvas que podem apresentar bastantes variações. Este processo, como se pode prever, será bastante arriscado. No fundo o critério irá procurar uma espécie de máximo local na curva de aprendizagem.

### 2.7.3 Critério de paragem

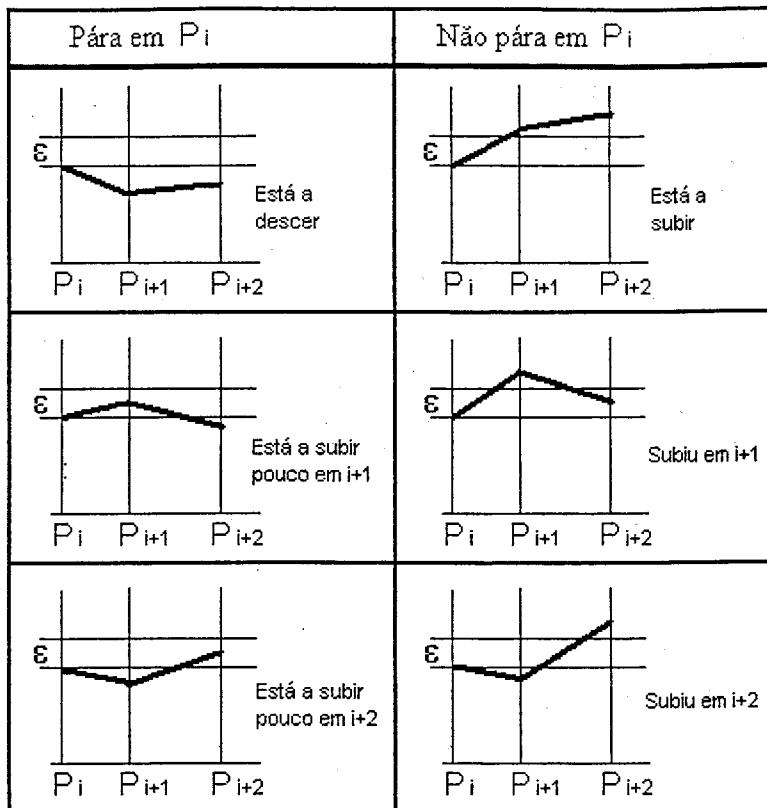
O método deverá decidir se já a curva já atingiu o máximo baseando-se apenas num segmento inicial da sequência de proporções.

O critério para detectar o máximo consiste em observar em que ponto surge uma subida demasiado pequena ( $<\varepsilon$ ) ou uma descida. O critério que vamos utilizar usa três valores consecutivos da taxa de acertos,  $A_i, A_{i+1}, A_{i+2}$ , e a detecção é feita se  $A_i + \varepsilon \geq A_{i+1}$  e  $A_i + \varepsilon \geq A_{i+2}$ , isto é, se nesta janela houver uma descida em relação ao primeiro ponto (considera-se que desceu) ou se as subidas forem pequenas (considera-se que subiu pouco) (ver Figura 2-5).

Apresentamos uma generalização do critério para n pontos utilizando a seguinte expressão:

$$\forall k \in \{1, 2, \dots, n-1\} \quad A_i + \varepsilon \geq A_{i+k}$$

O algoritmo pára se nenhum dos pontos a seguir ao ponto inicial ( $i$ ) apresentar uma subida maior que  $\varepsilon$ .



**Figura 2-5 — Diferentes situações que podem surgir com um critério de 3 pontos**

A escolha de uma janela com três pontos e não dois representa um compromisso. Por um lado, serve para tornar o critério um pouco mais robusto às variações da taxa de acertos, evitando que o processo páre na primeira descida.

Podemos comparar os resultados obtidos com a utilização deste critério simples com os obtidos observando toda curva de aprendizagem.

A Tabela 2-1 mostra as proporções escolhidas utilizando o critério descrito (considerando  $\epsilon=0$ ) e as proporções ideais observando a totalidade da curva.

**Tabela 2-1 — Comparação do máximo global com o obtido pelo critério**

Experiência	Máximo local usando o critério	Máximo global
1	40%	40%
2	20%	60%
3	30%	30%
4	20%	50%
5	20%	80%

Podemos observar que o máximo determinado pelo critério descrito é sempre atingido para uma proporção menor de casos.

#### 2.7.4 Outro critério utilizável

Outra variante do critério de paragem podia basear-se na modelação dos pontos da curva de aprendizagem através duma função e dessa forma prever em que ponto a função atinge o máximo.

Frey e Fisher no artigo “Modeling Decision Tree Performance with the Power Law” [Frey99], demonstram a adequação das funções do tipo

$$f(x) = ax^b$$

em que a e b são parâmetros a ajustar, para modelar curvas de aprendizagem típicas das árvores de decisão.

Depois de se obter um modelo (função) para a curva, utilizando os N primeiros pontos, poderíamos prever em que proporção era atingido o máximo. Além disso, o modelo da curva de aprendizagem poderá ser utilizado também para determinar qual a proporção a utilizar a seguir, utilizando-se dessa forma uma sequência de proporções adaptativa. Devemos salientar no entanto que esta abordagem necessita de bastantes pontos para que a curva seja modelizada com uma boa aproximação. Mesmo assim, podia-se conduzir um estudo para verificar se esta abordagem teria alguma vantagem.

## 2.8 O algoritmo PPP

Nesta secção apresentamos o algoritmo de redução do número de casos que implementámos. Este algoritmo é um processo dinâmico (significado já referido neste capítulo) e pode incluir-se no seu interior qualquer algoritmo de aprendizagem de classificadores.

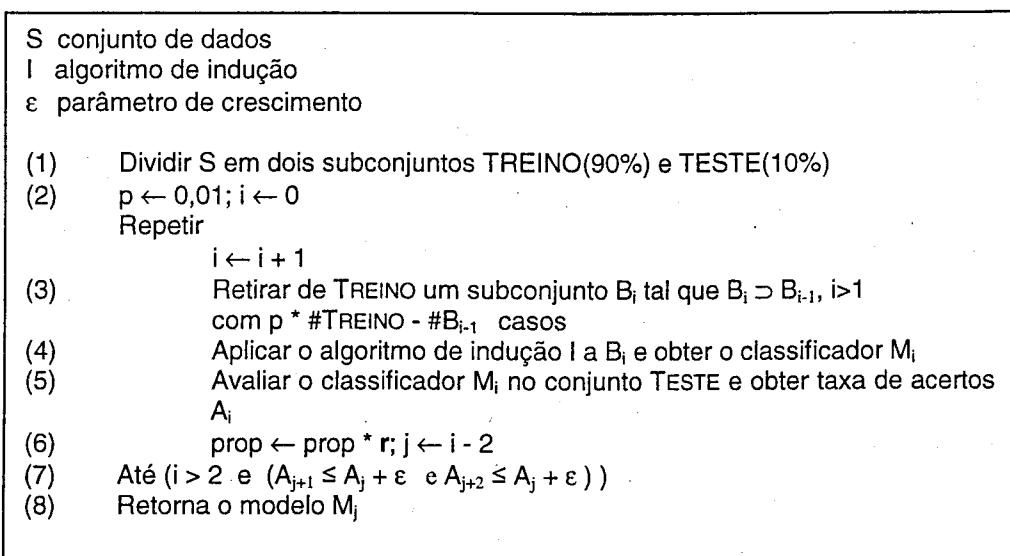
Podíamos obter melhores resultados utilizando algoritmos de aprendizagem *incrementais*, isto é, cujo esforço de aprendizagem dependesse apenas dos casos novos

acrescentados ao conjunto anterior (ex. Naive bayes) [Mitch97]. No entanto, a maioria dos algoritmos não são incrementais e consequentemente não especializámos o algoritmo PPP para estes casos.

Devemos referir que a sequência  $B_1 B_2 \dots B_n$  de subconjuntos utilizada neste algoritmo é construída de forma a que  $B_i \subset B_j$  ( $i < j$ ), isto é cada subconjunto contém todos subconjuntos anteriormente construídos. Os tamanhos dos subconjuntos estão em progressão geométrica.

O critério de paragem que utilizamos é o que foi descrito no final da secção anterior.

O algoritmo PPP está definido na seguinte figura.



**Figura 2-6 — Algoritmo de processamento por partições progressivas**

No passo (1) dividimos, de forma aleatória, o conjunto S em dois subconjuntos, TREINO e TESTE, e obrigámos a que a distribuição de classes dos três conjuntos fosse o mais próxima possível.

No passo (3) construímos um subconjunto  $B_i$  (estratificado), com  $p\%$  de casos do TREINO. Isto é feito acrescentando de forma a que o subconjunto  $B_k$  contém todos os subconjuntos anteriores  $B_{k-1}, B_{k-2}, \dots, B_1$ .

O passo (7) representa o critério de paragem que utilizámos.

No próximos capítulo vamos descrever as experiências que realizámos para avaliar o algoritmo que acabámos de descrever, bem como a análise dos resultados obtidos e conclusões.



### **3 Avaliação do algoritmo de processamento de partições progressivas**

Como já foi adiantado no Capítulo anterior o desenvolvimento de métodos que reduzam a necessidade de processar grandes quantidades de casos é uma tarefa bastante importante no âmbito do processo de extracção de conhecimento em bases de dados (ECBD). Para grandes conjuntos de dados, pretende-se ter a possibilidade de construir modelos de classificação com uma boa taxa de acertos, utilizando apenas um subconjunto de dados, gastando dessa forma menos tempo. Existem situações em que o conjunto de dados é tão grande que o algoritmo de aprendizagem não consegue completar a construção do modelo de classificação. Nestas situações a utilização de métodos de redução do número de casos (“sampling”) é essencial.

No capítulo anterior descrevemos uma solução para este problema, um algoritmo, que denominamos de PPP, que processa os dados por partições progressivamente maiores. Neste capítulo vamos avaliar este algoritmo comparando-o com o algoritmo PT (processamento total), que consiste no algoritmo base (algoritmo de aprendizagem) aplicado à totalidade dos dados.

O que pretendemos com esta avaliação é comparar os resultados do algoritmo PPP com os resultados obtidos pelo algoritmo PT. Realizámos, para esse efeito, várias experiências de comparação sobre 12 conjuntos de dados.

Na próxima secção descrevemos a metodologia de avaliação que utilizámos para comparar o nosso algoritmo com o algoritmo PT. O objectivo desta avaliação é comparar não só as taxas de acertos obtidas pelo algoritmo PT com a obtida utilizando o algoritmo de PPP, como também, os tempos gastos por cada um dos algoritmos e determinar se houve alguma poupança de tempo utilizando o método que implementámos.

### 3.1 Metodologia de avaliação

Para avaliar o algoritmo PPP utilizámos 12 conjuntos de dados e para cada um deles realizámos experiências com o objectivo de verificar se o mesmo, não perdendo muito na taxa de acertos, permitia poupar no tempo de execução em relação ao algoritmo PT.

Quisemos sobretudo experimentar o algoritmo PPP em grandes conjuntos de dados, no entanto devido à dificuldade de obtê-los tivemos também que utilizar alguns de dimensão menor.

Nas nossas experiências utilizámos como algoritmos base o C5.0 [Quin98], algoritmo de construção de árvores de decisão, e também um algoritmo de redes neurais para redes de perceptrões em multi-camadas (“Multi-layer perceptron”) implementado no pacote de software *Clementine* [Clementine]. Vamos referir o algoritmo de redes neurais pela sigla MLP e a implementação que utilizámos por clemMLP.

Sempre que não referirmos explicitamente o algoritmo base, que utilizámos no interior do algoritmo PPP, estaremos a considerar o C5.0, e referiremos explicitamente quando as experiências considerarem como algoritmo base o MLP.

Os valores dos parâmetros que usámos no nosso algoritmo, considerando o C5.0 como algoritmo base, foram para o  $r$  (regula o crescimento do tamanho das partições) o valor 2 e 1,5, para o parâmetro  $\epsilon$  (utilizado no critério de paragem) o valor 0,001 e utilizámos um critério que observa uma janela de 3 pontos.

Quanto aos mesmos parâmetros mas considerando o algoritmo base MLP, considerámos os valores  $r=1,5$  e  $\epsilon=0,01$ . Neste caso utilizámos como critério de paragem uma janela de apenas 2 pontos, isto é, ocorre uma paragem sempre que a taxa de acertos não suba pelo menos  $\epsilon$  entre duas iterações sucessivas do algoritmo PPP.

Os valores que observámos ao longo das experiências, e toda a metodologia são iguais tanto no caso do algoritmo base ser o C5.0 como o MLP. Descrevemos nos próximos parágrafos o que comparámos, que métodos de avaliação utilizámos, que valores observámos bem como agregámos os vários resultados.

Para comparar, em cada conjunto de dados, o algoritmo PPP com o algoritmo PT precisamos de ter estimativas tanto das taxas de acertos obtidas, como dos tempos consumidos pelos dois algoritmos.

Para obter estimativas mais fiáveis, tanto para as taxas de acertos como para os tempos, utilizámos o procedimento de validação cruzada (“cross validation” [Kohavi95b;BerHand99]) com 10 partes (“folds”). Com este procedimento obtivemos, para cada algoritmo, 10 medições para taxa de acertos e para os tempos.

Pretendemos quantificar as diferenças nas taxas de acertos, bem como no tempo consumido pelos dois algoritmos. Para tal vamos observar as diferenças entre as taxas de acertos (e tempos) obtidas pelos dois métodos dentro de cada iteração i da validação cruzada, as médias das taxas de acertos (e tempos) obtidas por cada método ao longo das 10 iterações e ainda os rácios entre as taxas de acertos (e tempos) obtidas pelo algoritmo PPP e pelo algoritmo PT. A vantagem da utilização dos rácios é o facto de tornar relativas as diferenças. Considerando que  $A_{PPP(i)}$  e  $A_{PT(i)}$  representam as taxas de acertos (o ‘A’ abrevia taxa de acertos) obtidas na iteração i da validação cruzada pelos algoritmos PPP e PT, e que  $T_{PPP(i)}$  e  $T_{PT(i)}$  os tempos (o ‘T’ abrevia tempos) consumidos pelos respectivos algoritmos nas mesmas iterações, então as medidas anteriores (diferenças, rácios e médias) são definidas da seguinte forma:

$$\text{Diferença nas taxas de acertos na iteração } i = A_{PPP(i)} - A_{PT(i)}$$

Rácio das taxas de acertos obtidas pelos algoritmos na iteração i :

$$AR_i = \frac{A_{PPP(i)}}{A_{PT(i)}}$$

Rácio dos tempos consumidos pelos algoritmos na iteração i :

$$TR_i = \frac{T_{PPP(i)}}{T_{PT(i)}}$$

Média das taxas de acertos obtidas pelo algoritmo PPP (e PT) ao longo das 10 iterações:

$$\bar{A}_{PPP} = \frac{1}{10} \sum_{i=1}^{10} A_{PPP(i)} \quad \text{e} \quad \bar{A}_{PT} = \frac{1}{10} \sum_{i=1}^{10} A_{PT(i)}$$

Média dos tempos consumidos pelo algoritmo PPP (e PT) ao longo das 10 iterações:

$$\bar{T}_{PPP} = \frac{1}{10} \sum_{i=1}^{10} T_{PPP(i)} \quad \text{e} \quad \bar{T}_{PT} = \frac{1}{10} \sum_{i=1}^{10} T_{PT(i)}$$

Para agregar os 10 rácios (das taxas e dos tempos) obtidos decidimos utilizar a média geométrica. A razão de usar a média geométrica e não a média aritmética é que esta última presta-se a interpretações inconsistentes dependendo da forma como se calculam os rácios. Esta inconsistência resulta de em geral

$$\frac{1}{n} \sum_{i=1}^n \frac{a_i}{b_i} \neq \left( \frac{1}{n} \sum_{i=1}^n \frac{b_i}{a_i} \right)^{-1}$$

(dados  $a_i \neq 0$  e  $b_i \neq 0$ ).

Contrariamente à média aritmética, a média geométrica é consistente quando calculamos os rácios doutra forma, isto resulta da equação

$$\sqrt[n]{\prod_{i=1}^n \frac{a_i}{b_i}} = \left( \sqrt[n]{\prod_{i=1}^n \frac{b_i}{a_i}} \right)^{-1}$$

ser sempre verdadeira (dados  $a_i \neq 0$  e  $b_i \neq 0$ ).

A observação destas médias geométricas dos rácios das taxas de acertos e dos tempos dão-nos uma ideia global do comportamento dos dois algoritmos ao longo das 10 iterações de validação cruzada.

Definimos estas medidas da seguinte forma:

Média geométrica dos Rácios das taxas de acertos (e tempos) obtidas pelos algoritmos:

$$\overline{AR} = \sqrt[10]{\prod_{i=1}^{10} \frac{A_{PPP(i)}}{A_{PT(i)}}} \text{ e } \overline{TR} = \sqrt[10]{\prod_{i=1}^{10} \frac{T_{PPP(i)}}{T_{PT(i)}}}$$

Um bom resultado do algoritmo PPP implica que  $\overline{AR}$  está muito próximo de 1 e  $\overline{TR}$  é menor que 1. Quanto menor for  $\overline{TR}$  maior será o tempo poupadado pelo algoritmo PPP.

Na próxima secção apresentamos os resultados que obtivemos em conjuntos de dados de grande dimensão utilizando o C5.0 como algoritmo base de PPP.

### 3.2 Conjuntos de dados de grande dimensão

Todos os resultados apresentados a seguir foram obtidos com o algoritmo base C5.0.

#### 3.2.1 Conjunto *TASK1*

O conjunto *TASK1* é uma das tabelas que constituem a base de dados SISYPHUS. Esta base provém dum repositório de dados (“data warehouse”) chamado MASY, que é utilizada na recolha e análise de dados sobre seguros de vida da companhia de seguros Swiss Life.

Este conjunto de dados é constituído por 111077 casos e tem 72 atributos. Dos 72 atributos existentes 26 são de domínio contínuo, 45 de domínio discreto e 1 é o atributo com o valor da classe do caso. Cada caso pode ter 1 de 9 classes possíveis.

##### 3.2.1.1 Análise dos resultados obtidos

Em relação aos tempos consumidos pelos dois processos, como se observa nas figuras 3-1 e 3-2, o algoritmo PPP foi mais rápido que o algoritmo PT.

Os melhores resultados foram obtidos quando o algoritmo PPP foi usado com o parâmetro  $r=1,5$ , conseguindo-se no melhor caso (fold=1) um tempo 270 vezes menor do que o consumido pelo algoritmo PT. Nos piores casos, que aconteceram para  $r=2$ , o algoritmo PPP gastou no máximo 0,64 do tempo gasto pelo algoritmo PT.

As médias geométricas dos rácios dos tempos foram de 0,0472 e 0,0172 quando corremos o algoritmo PPP usando os parâmetros  $r=2$  e  $r=1,5$  respectivamente. Podemos dizer então que para  $r=2$  o algoritmo PPP foi 21 vezes mais rápido que o algoritmo PT. Para  $r=1,5$  o algoritmo PPP foi 58 vezes mais rápido que o algoritmo PT.

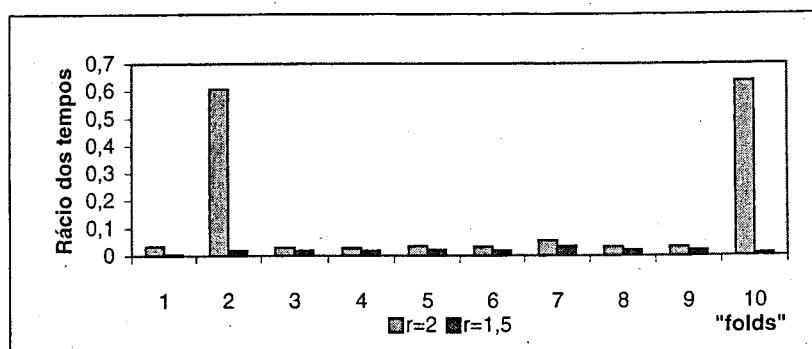


Figura 3-1 — Dados TASKI – Rácio dos tempos ( $T_{PPP}/T_{PT}$ )

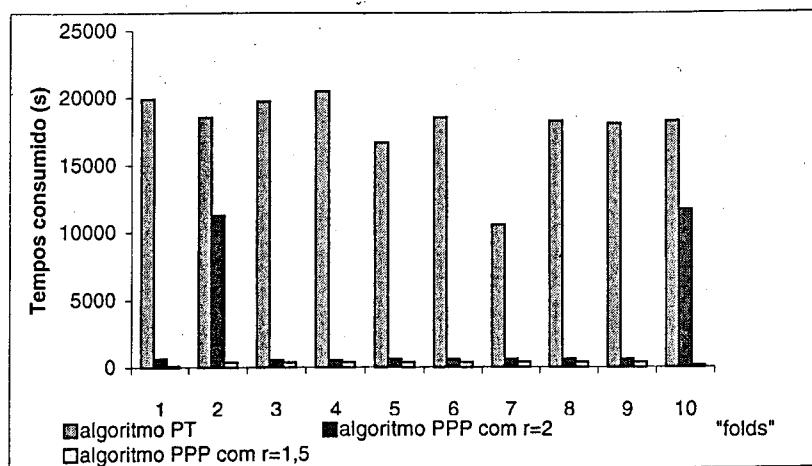


Figura 3-2 — Dados TASKI – Comparaçao dos tempos consumidos

Quanto às taxas de acertos obtidas com o algoritmo PPP, observa-se que foram muito próximas, embora ligeiramente inferiores às obtidas usando o algoritmo PT. A diferença entre as médias das taxas (em percentagens) foi de -0,2 para  $r=1,5$  e de -0,41 para  $r=2$  (ver Tabela 3-1).

Tabela 3-1 — Dados TASKI -Médias das taxas de acertos

	Médias aritméticas das Taxas de acertos (em percentagens)		Diferença $\bar{A}_{PPP} - \bar{A}_{PT}$
	$\bar{A}_{PPP}$	$\bar{A}_{PT}$	
r=1,5	67,62 ± 0,32	67,82 ± 0,27	-0,20
r=2	67,41 ± 0,44	67,82 ± 0,27	-0,41

As seguintes figuras mostram as taxas de acertos ao longo das 10 iterações, bem como o rácio entre a taxa obtida pelo algoritmo PPP e a taxa de acertos obtida pelo algoritmo PT. Na Figura 3-4 observamos que em 8 das 10 iterações os valores dos rácios, tanto para r=1,5 como r=2, muito próximos de 1, embora ligeiramente menores. Nas iterações 2 (para r=1,5) e 10 (para os dois valores de r) o algoritmo PPP obtém taxas de acertos maiores que as obtidas pelo algoritmo PT. Este facto poderá corresponder a uma situação potencial de sobreajustamento (“overfitting”, ver secção 2-7) que o algoritmo PPP evitou, mas poderá também resultar da variância.

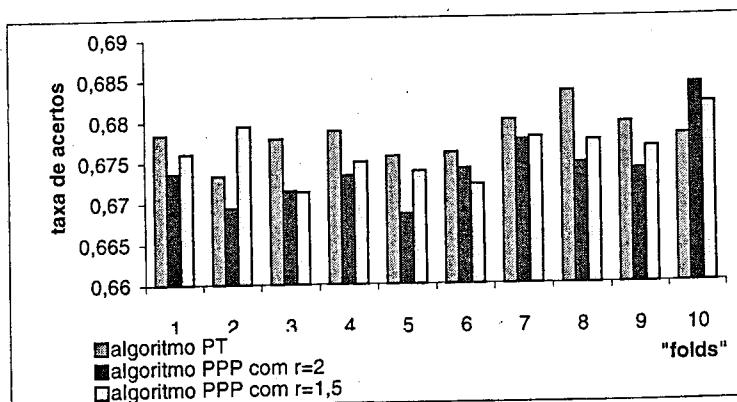
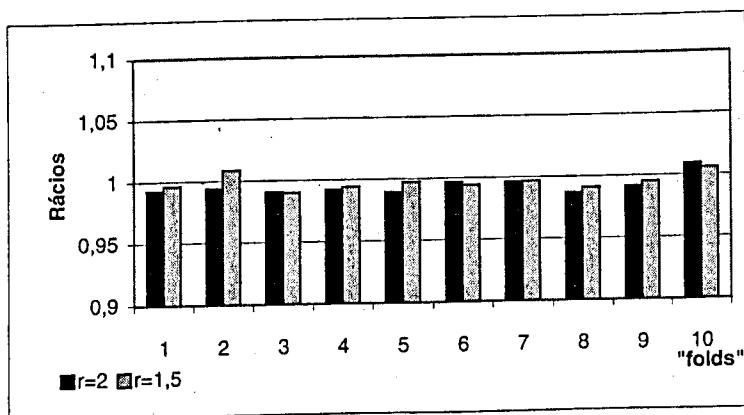


Figura 3-3 — Dados TASKI – comparação das taxas de acertos

Figura 3-4 — Dados TASKI – Rácio das taxas de acertos ( $A_{PPP}/A_{PT}$ )

Concluímos que o algoritmo PPP foi neste caso bastante útil, permitindo-nos poupar bastante tempo e obtendo taxas bastante comparáveis às obtidas pelo algoritmo PT.

### 3.2.2 Conjunto sobre Cobertura Florestal

Este conjunto de dados reúne informações sobre variáveis cartográficas e ainda uma variável com o tipo de cobertura florestal, sendo esta última a que contém o valor da classe. A tarefa de classificação consiste em prever o tipo de cobertura florestal dados os valores das variáveis cartográficas.

Este conjunto é constituído por 581012 casos e tem 55 atributos. Dos 55 atributos existentes 10 são de domínio contínuo, 44 de domínio discreto (neste caso binárias) e por fim 1 atributo com o valor da classe do caso. Cada caso pode ter 1 de 7 classes possíveis.

#### 3.2.2.1 Análise dos resultados obtidos

Quanto aos tempos consumidos pelos dois algoritmos observamos (ver Gráfico 3-5) que o algoritmo PPP foi mais lento.

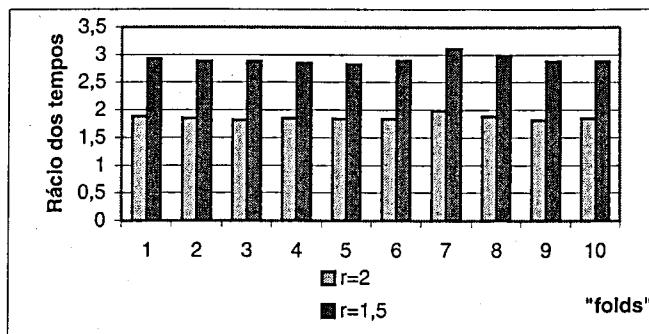


Figura 3-5 — Dados *Forest Cover Type* – Rácio dos tempos ( $T_{\text{PPP}}/T_{\text{PT}}$ )

O facto do algoritmo PPP ter sido mais lento que o algoritmo PT pode ser explicado pela observação da curva de aprendizagem obtida usando o C5.0 neste conjunto. A curva de aprendizagem neste conjunto (ver Figura 3-6) apresenta na fase final, subidas relativamente elevadas, o que obriga o método PPP a prosseguir até aos 100%, pois o

valor de  $\epsilon = 0,001$  não é suficientemente grande para obrigar a uma paragem mais precoce.

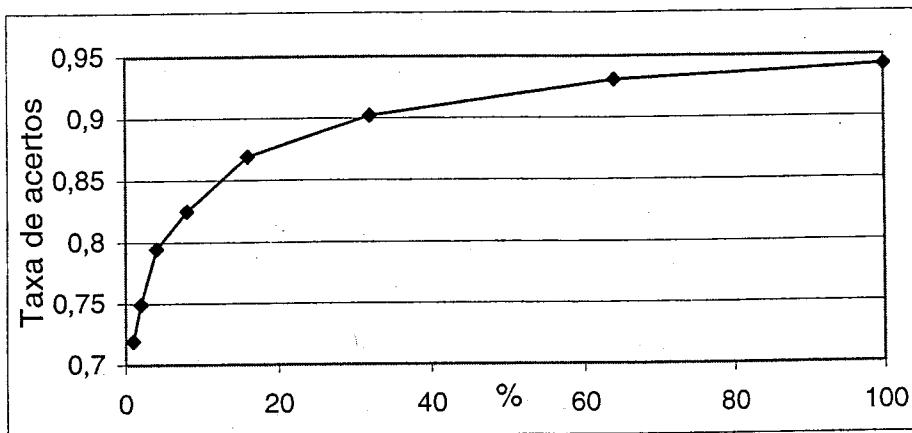


Figura 3-6 — Dados *Forest Cover Type* — Curva de típica deste conjunto

Se tivéssemos fixado o parâmetro  $r$  com um valor mais elevado ( $>2$ ) diminuíramos o número de iterações necessárias e dessa forma o tempo consumido pelo algoritmo PPP. No entanto o facto da taxa de acertos subir tanto ( $>1\%$  na última iteração) na fase final só poderia ser contornado por um valor do parâmetro  $\epsilon$  mais elevado, mas assistiríamos evidentemente a uma descida mais pronunciada nas taxas de acertos dos classificadores construídos.

No que diz respeito às taxas de acertos obtidas utilizando o algoritmo PPP observamos que foram ligeiramente inferiores às obtidas usando o algoritmo PT, sendo a diferença entre as médias das taxas em escala de percentagens igual a 0,21 (ver Tabela 3-2).

Tabela 3-2 — Dados *Forest Cover Type* — médias das taxas de acertos

	Médias aritméticas das Taxas de acertos (em percentagens)		$\bar{A}_{PPP} - \bar{A}_{PT}$
	$\bar{A}_{PPP}$	$\bar{A}_{PT}$	
$r=1,5$	$94,34 \pm 0,12$	$94,55 \pm 0,13$	0,21
$r=2$	$94,34 \pm 0,12$	$94,55 \pm 0,13$	0,21

As médias geométricas dos rácios entre as taxas de acertos obtidas pelo algoritmo PPP e as taxas de acertos obtidas pelo algoritmo PT têm, para os dois valores de  $r$  (1,5 e 2) o

valor 99,78, isto é o nosso algoritmo perdeu em taxa de acertos apenas 0,2% da taxa obtida pelo algoritmo PT. Estas médias mostram a proximidade relativa das taxas de acertos obtidas pelos dois algoritmos.

Em conclusão, embora a redução da taxa de acertos ter sido pequena o algoritmo PPP não conseguiu poupar tempo em relação ao algoritmo PT.

### 3.2.3 Conjunto *Census*

O *Census* é um conjunto que reúne informações sobre o rendimento anual de habitantes dos Estados Unidos da América e ainda outros atributos nomeadamente a raça, habilitações académicas, sexo, entre outros. O atributo rendimento tem dois valores possíveis, o primeiro representa rendimentos menores ou iguais a 50000 dólares e o outro valores maiores. O problema de classificação consiste em prever este último atributo utilizando os valores nos outros atributos. Este conjunto é constituído por 199523 casos e tem 55 atributos. Dos 55 atributos existentes 10 são contínuos, sendo os outros discretos.

As experiências neste conjunto foram realizadas utilizando apenas o valor do parâmetro  $r=1,5$ .

#### 3.2.3.1 Análise dos resultados obtidos

A Tabela 3-3 mostra os resultados que obtivemos em relação ao tempo consumido pelos algoritmo PPP e PT aplicados a este conjunto.

O ganho de tempo é muito significativo sendo o tempo consumido pelo algoritmo PPP cerca de 10 vezes menor que o tempo consumido pelo algoritmo PT.

**Tabela 3-3 — Dados *Census* – Médias dos tempos consumidos**

Médias aritméticas dos tempos consumidos	
$\bar{T}_{PPP}$	$\bar{T}_{PT}$
$26 \pm 22$	$270 \pm 64$

A Tabela 3-4 mostra os resultados que obtivemos no conjunto *Census* no que diz respeito às taxas de acertos. Podemos observar que a perda na taxa de acertos, apesar de

maior do que as observadas nos conjuntos TASK1 e Forest Cover Type, é menor do que 1%, o que poderá ser aceitável para algumas aplicações.

**Tabela 3-4 — Dados Census – Médias das taxas de acertos**

Médias aritméticas das Taxas de acertos (em percentagens)		Diferença
$\bar{A}_{PPP}$	$\bar{A}_{PT}$	$\bar{A}_{PPP} - \bar{A}_{PT}$
$94,6 \pm 0,14$	$95,3 \pm 0,13$	0,7

O algoritmo PPP foi 10 vezes mais rápido que o algoritmo PT, embora tenha perdido 0,7% na taxa de acertos.

### 3.2.4 Conjunto *KDD-Cup 2000\_CL*

O conjunto que utilizámos é baseado no conjunto KDD-Cup 2000, mas não é o conjunto tal qual apresentado no desafio [KDDCup2000]. Este conjunto foi preparado por Carlos Leandro do LIACC, a partir de KDD\_Cup2000, e é por esse motivo identificado com o sufixo “CL”. Designamos este conjunto por “KDD-Cup2000\_CL” e apresentamos sobre ele apenas as características que dizem respeito ao número de casos e número de atributos.

O conjunto é constituído por 90797 casos e 112 atributos. Dos 112 atributos 100 são contínuos, sendo os restantes discretos.

As experiências neste conjunto foram realizadas utilizando apenas o valor do parâmetro  $r=1,5$ .

#### 3.2.4.1 Análise dos resultados obtidos

A Tabela 3-5 mostra os resultados que obtivemos o que diz respeito ao tempo consumido pelos algoritmo PPP e PT aplicados a este conjunto, e permitem-nos concluir que neste conjunto não houve grande utilidade em utilizar o algoritmo PPP.

**Tabela 3-5 — Dados KDD-Cup 2000\_CL – Médias dos tempos consumidos**

Médias aritméticas dos tempos consumidos	
$\bar{T}_{PPP}$	$\bar{T}_{PT}$
$254 \pm 166$	$129 \pm 4$

O tempo gasto pelo algoritmo PPP é cerca do dobro do gasto pelo algoritmo PT. Se observarmos o desvio padrão das observações do tempo de PPP podemos concluir que neste conjunto o algoritmo PPP foi muito instável.

A Tabela 3-6 mostra os resultados que obtivemos neste conjunto no que diz respeito às taxas de acertos. Podemos observar que a perda na taxa de acertos é um pouco elevada, o que não é compensado, como vimos, pelo tempo gasto pelo algoritmo PPP.

**Tabela 3-6 — Dados KDD-Cup 2000\_CL – Médias das taxas de acertos**

Médias aritméticas das Taxas de acertos (em percentagens)		Diferença
$\bar{A}_{PPP}$	$\bar{A}_{PT}$	$\bar{A}_{PPP} - \bar{A}_{PT}$
$94,18 \pm 1,6$	$95,33 \pm 0,34$	1,15

Na próxima secção descrevemos os resultados que obtivemos em experiências com conjunto de dimensão menor.

### 3.3 Conjuntos de dados de dimensão menor

Embora o algoritmo PPP, que descrevemos no capítulo anterior, tenha sido desenvolvido para tratar preferencialmente grandes conjuntos de dados iremos nesta secção descrever os resultados que obtivemos em experiências com conjuntos de dimensão mais reduzida.

Os conjuntos de dados que utilizámos nestas experiências estão apresentados na Tabela 3-7, que contém informações sobre o número de casos entre outros parâmetros.

As experiências descritas nesta secção consideram o C5.0 como algoritmo base de PPP.

Contrariamente ao que fizemos nas experiências anteriores, em relação ao ponto inicial do algoritmo (proporção inicial), para cada conjunto, utilizámos a percentagem correspondente a 1000 casos.

### 3.3.1 Descrição dos conjuntos de dados

A seguinte tabela descreve de forma sucinta algumas características dos conjuntos de dados de dimensão menor que utilizámos na avaliação do algoritmo PPP. Contém informações sobre o domínio dos conjuntos e também as suas dimensões.

**Tabela 3-7 — Descrição dos conjuntos de dados de menor dimensão**

Conjunto	nº casos	nº de Atributos		nº classes	Domínio
		Conti.	Discr.		
<i>Adult</i>	32562	6	8	2	Censos sobre rendimentos.
<i>Satimage</i>	6435	36	0	6	Imagens de satélite.
<i>Letter</i>	20000	16	0	26	Reconhecimento de letras.
<i>Taskb_hhold</i>	12934	13	30	3	Seguros.
<i>Pyrimidines</i>	6996	54	0	2	Produtos farmaceuticos
<i>Shuttle</i>	58000	9	0	7	Space Shuttle.
<i>Waveform</i>	5000	40	0	3	Dados gerados artificialmente.
<i>Nursery</i>	12960	0	8	5	Admissão em escolas de enfermagem.

### 3.3.2 Análise dos resultados obtidos

No que concerne ao tempo, o algoritmo PPP foi em geral mais lento do que o algoritmo PT. A Figura 3-7 representa, para cada conjunto, a média geométrica dos rácios dos tempos gastos pelo algoritmo PPP e pelo algoritmo PT, utilizando o algoritmo PPP com os parâmetros  $r=2$  e  $r=1,5$ .

No que diz respeito ao tempos, observamos que o pior resultado foi obtido no conjunto de dados *nursery*, em que o tempo do algoritmo PPP foi 5 vezes superior ao utilizado pelo algoritmo PT. Nos restantes casos o tempo, embora superior, nunca chegou a 3 vezes o tempo do algoritmo PT, e na maior parte dos casos o rácio esteve muito próximo de 2 (sendo por vezes menor).

Observa-se que apesar de termos experimentado o nosso algoritmo em conjuntos de dimensão menor, que não são os conjuntos alvo do algoritmo PPP, em dois conjuntos (*shuttle*, *taskb\_hhold*), conseguimos tempos menores usando este algoritmo.

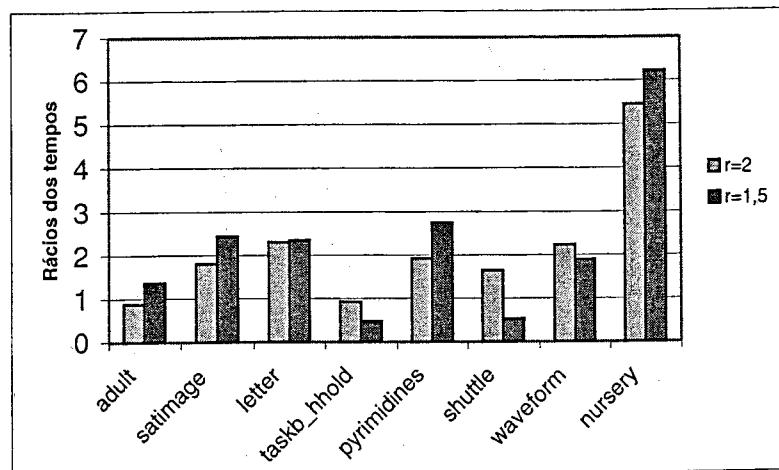


Figura 3-7 — Ráios dos tempos para os 8 conjuntos de dimensão menor

O facto do tempo utilizado pelo algoritmo PPP ter sido maior que o necessário ao algoritmo PT em quase todos os conjuntos, justifica-se pela observação que em praticamente todos a proporção de dados escolhida pelo algoritmo PPP ter sido 100%. A Tabela 3-8 representa para cada conjunto de dados a percentagem média de casos escolhida pelo algoritmo. Esta média é calculada, para cada conjunto, fazendo a média aritmética das percentagens escolhidas pelo algoritmo PPP em cada iteração de validação cruzada.

Tabela 3-8 — Médias das percentagens (de casos) escolhidas pelo algoritmo PPP

Conjuntos	Percentagens de casos	
	r=2	R=1,5
Adult	97%	84%
Satimage	87%	87%
Letter	100%	100%
Taskb	28%	13%
Pyrimidines	98%	93%
Shuttle	29%	4%
Waveform	87%	63%
Nursery	100%	100%

As taxas de acertos que obtivemos utilizando o nosso algoritmo são, em geral, muito próximas das obtidos pelo algoritmo PT. A Figura 3-8 representa, para cada conjunto de dados, as médias geométricas dos rácios entre as taxas de acertos obtidas pelos algoritmos PPP e PT.

Utilizando o algoritmo PPP, apenas nos conjuntos *satimage* e *waveform* obtivemos perdas maiores do que 1% (rácio <0,99) nas taxas de acertos obtidas utilizando o algoritmo PT. Esta perdas aconteceram quando utilizámos o parâmetro  $r$  com o valor 2.

Nos conjuntos *adult*, *letter* e *shuttle* a perda não atingiu os 0,5% da taxa de acertos do algoritmo PT, isto é o algoritmo PPP conseguiu 99,5% da taxa de acertos obtida pelo algoritmo PT.

Na experiência com o conjunto *waveform* observámos que quando utilizámos o algoritmo PPP com  $r=2$  obtivemos taxas de acertos maiores.

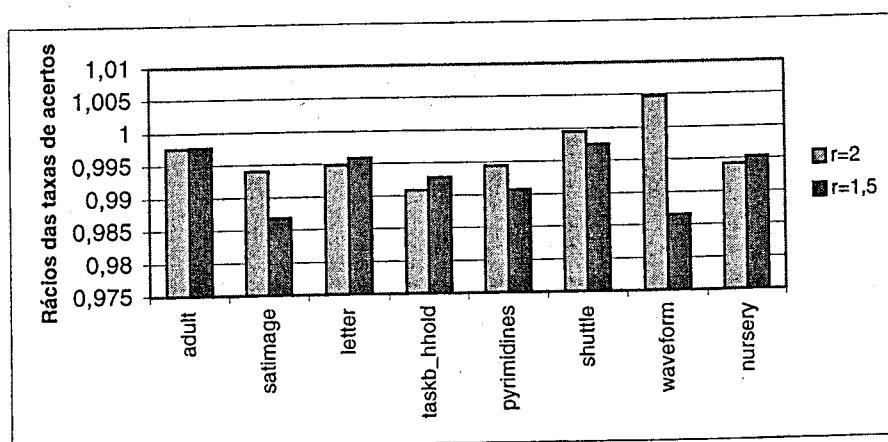


Figura 3-8 — Rácios das taxas de acertos para os 8 conjuntos de dimensão menor

### 3.4 Conjuntos de dados de dimensão menor (base MLP)

Como já foi referido no Capítulo 1, os algoritmos de aprendizagem de redes neurais são bastante lentos. O método PPP permite obter resultados parciais que poderão ser úteis (filosofia “anytime results”) e por esse motivo decidimos avaliar o algoritmo PPP utilizando o algoritmo base MLP.

O algoritmo PPP tendo no seu interior o algoritmo MLP foi comparado usando uma metodologia semelhante à das experiências anteriores. Relembremos que o critério de paragem foi neste casos baseado numa janela de apenas dois pontos e que os parâmetros utilizados são  $r=1,5$  e  $\epsilon=0,01$ .

Utilizámos 2 conjuntos de dados para avaliar esta configuração do algoritmo PPP. As comparações que fazemos têm como ponto de referência o algoritmo PT, isto é, o algoritmo MLP consumindo a totalidade dos dados.

Os conjuntos que usámos foram o *waveform* e o *letter*, conjuntos já descritos anteriormente. Nas duas próximas secções descrevemos os resultados que obtivemos.

### 3.4.1 Análise dos resultados no conjunto waveform

Quanto aos tempos gastos pelos dois algoritmos observamos que o algoritmo PPP gasta cerca de 2,8 vezes menos tempo que o algoritmo PT (ver Tabela 3-9). Devemos acrescentar que o conjunto é constituído por apenas 5000 casos, não sendo portanto o tipo de conjunto para o qual o algoritmo PPP foi projectado.

**Tabela 3-9 — Dados *waveform* – Médias dos tempos**

Médias aritméticas dos tempos consumidos	
$\bar{T}_{PPP}$	$\bar{T}_{PT}$
$37 \pm 13$	$103 \pm 21$

Em relação às taxas de acertos (ver Tabela 3-10), observa-se uma redução relativamente elevada (3,6%), quando utilizámos o algoritmo PPP, o que se poderá justificar pelo valor elevado do parâmetro  $\epsilon=0,01$ .

**Tabela 3-10 — Dados *waveform* – Médias das taxas de acertos**

Médias aritméticas das Taxas de acertos (em percentagens)		$\bar{A}_{PPP} - \bar{A}_{PT}$
$\bar{A}_{PPP}$	$\bar{A}_{PT}$	
$79,9 \pm 3,64$	$83,5 \pm 7,91$	-3,6

No entanto, podemos referir que as taxas de acertos obtidas pelo algoritmo PPP são melhores que as obtidas pelo algoritmo C5.0 consumindo a totalidade dos dados. O C5.0 obtém 75,46% de taxa de acertos, enquanto o algoritmo PPP utilizando o clemMLP como algoritmo base obtém 79,9%.

Em resumo o algoritmo PPP dá-nos uma taxa de acertos bastante melhor que o C5.0. Notámos ainda que o algoritmo PPP, utilizando o clemMLP como base, gasta mais tempo que o C5.0 consumindo a totalidade dos dados, mas menos que o algoritmo clemMLP consumindo a totalidade dos dados. O tempo gasto em média pelo C5.0 consumindo a totalidade dos dados foi de 4,27 segundos, e tempo do algoritmo PPP (base clemMLP) foi de 37 segundos.

### 3.4.2 Análise dos resultados no conjunto *letter*

Quanto aos tempos gastos pelos dois algoritmos (ver Tabela 3-11), observamos que o algoritmo PT é ligeiramente mais lento que o algoritmo PPP. O algoritmo PPP gasta 57% do tempo gasto pelo algoritmo PT.

Tabela 3-11 — Dados *letter* – Médias dos tempos

Médias aritméticas dos tempos consumidos	
$\bar{T}_{PPP}$	$\bar{T}_{PT}$
$1317 \pm 834$	$2300 \pm 205$

Em relação às taxas de acertos (ver Tabela 3-12) observou-se uma redução relativamente elevada, o que se poderá justificar pelo valor elevado do parâmetro  $\varepsilon=0,01$ .

Tabela 3-12 — Dados *letter* – Médias das taxas de acertos

Médias aritméticas das Taxas de acertos (em percentagens)		$\bar{A}_{PPP} - \bar{A}_{PT}$
$\bar{A}_{PPP}$	$\bar{A}_{PT}$	
$75,2 \pm 0,5$	$79,3 \pm 0,5$	-4,1

Ao contrário do que se passou com o conjunto *waveform*, no conjunto *letter* as taxas de acertos obtidas pelo C5.0 foram melhores que as obtidas usando redes neuronais (clemMLP). O C5.0 teve em média a taxa de acertos 88% e gastou 5,8 segundos.

As experiências que realizámos com o algoritmo PPP, utilizando o algoritmo base MLP, não foram conclusivas quanto às vantagens da utilização do algoritmo PPP. No entanto o algoritmo permite obter um conjunto de resultados preliminares em menos tempo, de acordo com a estratégia de “anytime results”. De qualquer forma experiências com conjuntos de dados de maior dimensão poderiam esclarecer melhor este ponto.

### 3.5 Conclusões

As experiências que realizámos revelam que o algoritmo PPP poderá ser aplicado a grandes conjuntos dados.

Com o conjunto *TASK1* obtivemos resultados surpreendentes no ganho de tempo utilizando este algoritmo. No melhor caso (“fold” 1), com o parâmetro  $r=1,5$ , o algoritmo PPP consome 270 vezes menos tempo que o algoritmo PT. Com o conjunto *Census* conseguiu-se reduzir o tempo cerca de 10 vezes à custa de uma redução de apenas 0,7% na taxa de acertos.

As experiências com os conjuntos *Forest Cover Type* e *KDD\_Cup2000\_CL* não revelaram ganhos por parte do algoritmo PPP. No caso do conjunto *Forest Cover Type* observámos que a curva de aprendizagem apresentava subidas bastante significativas até aos 100% dos dados, o que explica os maus resultados. É natural que se o ponto de paragem do algoritmo PPP estiver muito adiantado, isto é próximo dos 100% dos dados, então este não consegue poupar tempo.

A aplicação do algoritmo PPP para conjuntos de dados de dimensão menor, utilizando o algoritmo base C5.0, não revelou nenhuma vantagem, facto que esperávamos que acontecesse, pois nestes conjuntos o ponto de paragem deverá estar muito próximo dos 100% dos casos.

Quanto aos resultados obtidos com algoritmo PPP utilizando o algoritmo base MLP, conseguimos alguns ganhos no tempo, mas à custa de redução da taxa de acertos. Vale a

pena experimentar esta versão do algoritmo PPP em conjuntos de dimensão maior. Fica no entanto esta avaliação para trabalhos futuros.

Podemos mais uma vez referir que o algoritmo PPP, porque vai construindo modelos com taxas de acertos sucessivamente maiores, disponibiliza-nos resultados preliminares que podem ser bastante úteis. Do ponto do utilizador é muitas vezes bastante útil ter em menos tempo alguns modelos com resultados razoáveis, sabendo que o sistema pode tentar melhorar os modelos (filosofia “anytime results”).



## 4 Selecção de atributos utilizando resultados do algoritmo PPP

A dificuldade de processamento de grandes conjuntos de dados poderá ser causada não só pelo elevado número de casos mas também pela existência de muitos atributos irrelevantes e também pelo número de valores possíveis para cada atributo.

De acordo com [WeiIn98] o número de casos, número de atributos e número de valores constituem as três dimensões pela qual medimos o “tamanho” do conjunto de dados. Existem métodos que pretendem reduzir o “tamanho” do conjunto de dados segundo cada uma dessas dimensões. Nesta Tese pretendemos combinar métodos de pré-processamento que reduzam o número de casos (“sampling”) com métodos de pré-processamento que seleccionem os atributos relevantes (“feature selection”).

Deixámos os métodos de redução do número de valores dos atributos (designada discretização) fora dos objectivos desta Tese.

Descrevemos no Capítulo 2 um método que faz a redução do número de casos, isto é, constrói um modelo a partir dum subconjunto do conjunto treino tal que a sua taxa de acertos não seja significativamente (não no sentido estatístico) menor que a obtida do modelo construído a partir do conjunto inicial que contém a totalidade dos casos.

Neste capítulo iremos descrever o problema da selecção dos atributos mais relevantes e algumas soluções que já foram exploradas. Além disso, iremos descrever um algoritmo que combina um algoritmo de selecção de atributos com o algoritmo de redução de casos descrito no Capítulo 2, e que denominamos com a sigla SAPPP.

### 4.1 O problema dos atributos irrelevantes

Processar directamente, um conjunto de dados com um número elevado de atributos irrelevantes, aplicando um algoritmo de aprendizagem, implica não só um grande

esforço de computação, que se reflecte no tempo consumido, mas também na quantidade de memória necessária para correr esse algoritmo. Algumas implementações dos algoritmos de aprendizagem poderão mesmo não suportar conjuntos de grandes dimensões (nºcasos, nºatributos, nº valores) devido às necessidades de memória.

A existência de atributos irrelevantes num determinado conjunto de dados poder-se-á também reflectir na qualidade (taxa de acertos) dos modelos construídos pelo algoritmo de aprendizagem e resultar em diminuição da taxa de acertos.

#### 4.2 Algoritmos mais e menos sensíveis aos atributos irrelevantes

Alguns algoritmos são mais perturbados que outros, na qualidade medida pela taxa de acertos dos modelos que constroem, quando no conjunto de dados existem bastantes atributos irrelevantes. A classe de algoritmos designada k-vizinhos mais próximos (“kNN k-Nearest Neighbours”) sofre bastante com a presença de muitos atributos irrelevantes. Por outro lado os algoritmos que geram modelos simbólicos são mais robustos a este problema. [WeiIn98]

Eliminar atributos irrelevantes dum conjunto de dados melhora, em geral, os resultados obtidos pelo algoritmo dos k-vizinhos mais próximos. A razão para este facto é a forma como funciona este algoritmo. Este algoritmo classifica um novo caso procurando os k vizinhos mais próximos (mais semelhantes) no conjunto de treino e seguidamente combina as classificações destes casos (por exemplo através da classe mais frequente) obtendo a classificação para esse caso.

A proximidade dos casos é avaliada através da distância definida entre dois casos, sendo comum utilizar a seguinte função para distância:

$$d(i, j) = \sum_{k \in \text{Atributos}} \delta(\text{caso}(i, k), \text{caso}(j, k)), \text{ em que } \delta(a, b) = \begin{cases} 1 & \text{se } a \neq b \\ 0 & \text{se } a = b \end{cases}$$

$d(i, j)$  representa a distância do caso i ao caso j, e  $\text{caso}(i, k)$  representa o valor do atributo k no caso i

Mesmo que dois casos sejam de facto bastante semelhantes podem diferir nos atributos irrelevantes o que implica uma distância “artificialmente” grande, isto é, o vizinho mais

próximo não é bem encontrado, e portanto o algoritmo pode não dar muito bons resultados.

Existem variações ao algoritmo em que a distância é uma soma pesada, isto é cada atributo tem um determinado peso. Mas o problema de determinar esses pesos implica, de facto, que alguns atributos são mais relevantes que outros, isto é estamos a determinar quais são os atributos relevantes (pesos grandes) e irrelevantes (pesos pequenos).

Embora os algoritmos que geram árvores de decisão sejam mais robustos à existência de tributos irrelevantes, o tempo consumido e a memória gasta por estes aumenta se existirem muitos atributos irrelevantes.

Este problema é ainda mais evidente se os algoritmos que pretendemos utilizar são muito lentos (ex. redes neurais), ou sobretudo, se a complexidade do algoritmo depende bastante do número de atributos (ex. quadrática em relação ao número de atributos).

Na próxima secção vamos descrever algumas abordagens utilizadas para proceder à selecção de atributos relevantes.

### **4.3 Métodos de selecção de atributos**

Existem muitos métodos que procuram encontrar, para um determinado conjunto de atributos, um subconjunto (menor possível) com os atributos mais relevantes.

Estes métodos podem ser divididos segundo três abordagens diferentes [BlumLang97] referidas nas três seguintes secções.

#### **4.3.1 Selecção integrada no algoritmo (“embedded approach”)**

Nesta abordagem a selecção de atributos é realizada dentro do próprio algoritmo de aprendizagem. Algoritmos que constroem árvores de decisão são o exemplo mais evidente de métodos em que a selecção de atributos é feita internamente. No fundo estes algoritmos ao construírem o modelo estão a explorar o espaço dos subconjuntos de atributos, e a escolher os atributos mais relevantes.

#### 4.3.2 Selecção anterior ao algoritmo (“Filter approach”)

Nesta abordagem a selecção de atributos é realizada antes da fase em que o algoritmo de aprendizagem constrói o classificador, sendo portanto independente do algoritmo de aprendizagem.

Um exemplo simples de tal método é o que se baseia na relevância isolada de cada atributo, isto é, assenta na capacidade que um determinado atributo tem para discriminar as várias classes, sem considerar os outros atributos. É comum utilizar, para este fim, medidas baseadas na teoria da informação, como por exemplo, o ganho de informação. Seleccionar um conjunto de atributos relevantes, utilizando estas medidas, corresponde a ordenar os atributos por ordem decrescente da sua relevância e depois escolher os n primeiros, ou todos até um valor mínimo pré-estabelecido.

Além do método anterior, existe ainda a possibilidade de utilizar uma árvore de decisão para seleccionar atributos. O conjunto de atributos seleccionados consiste nos atributos que ocorrem na árvore de decisão. Pela forma como é construída uma árvore de decisão, este processo representa uma generalização do método baseado no ganho de informação, pois captura algumas dependências condicionais entre os atributos.

Uma sub-abordagem mais geral consiste em associar a cada subconjunto de atributos um valor de relevância e realizar uma procura no espaço dos subconjuntos de atributos tentando obter um subconjunto com um mínimo de atributos e um valor máximo de relevância.

#### 4.3.3 Selecção guiada pelo algoritmo (“wrapper approach”)

Esta abordagem [Kohavi95a] parte do princípio que a taxa de acertos obtida pelo modelo, gerado pelo algoritmo de aprendizagem, pode ser usada para decidir sobre a relevância dos atributos.

Os algoritmos de selecção deste tipo percorrem o espaço dos subconjuntos possíveis, e para cada subconjunto (candidato) constroem um modelo que é avaliado quanto à sua taxa de acertos. A ideia é encontrar um subconjunto pequeno mas com uma taxa de acertos o mais alta possível. O critério de escolha poderá ter outras variações que

considerem por exemplo a simplicidade do modelo, se esse for um objectivo importante.

A forma de percorrer o espaço é determinado pelo tipo de procura implementada, que determina qual o critério de paragem, estado inicial (subconjunto inicial) e como gerar o próximo estado da procura. Uma possibilidade de geração do próximo subconjunto (estado) a avaliar é através das operações *acrescentar um atributo* ou *retirar um atributo*, seguindo a direcção do *maior aumento* (procura “hill climbing”) da taxa de acertos.

Um critério de paragem poderá ser a não observação de melhoria na taxa de acertos. O ponto inicial poderá ser um subconjunto aleatório, um conjunto vazio (por convenção a taxa de acertos terá o valor zero) ou o conjunto total dos atributos.

Também estratégias de procura baseadas nos algoritmos genéticos [BlumLang97] foram usadas para percorrer o espaço dos subconjuntos de atributos, e dessa forma evitarem os máximos (mínimos) locais em que caiem muitas das estratégias de procura.

A principal desvantagem desta abordagem é o esforço computacional que necessita.

#### 4.4 O algoritmo de selecção de atributos baseado no algoritmo PPP

O algoritmo que vamos descrever parte da ideia de utilizar uma árvore de decisão para seleccionar atributos, mas evitando ter que utilizar a totalidade dos casos para obter essa árvore de decisão. Se utilizarmos todos os casos do conjunto para aprender uma árvore de decisão, e se o número de casos for muito elevado, será que podemos utilizar árvores construídas com menos casos para seleccionar atributos? Ainda que a resposta possa ser afirmativa, não temos à partida nenhuma ideia de quantos casos precisamos para que a árvore de decisão, construída com os mesmos, nos seleccione um bom conjunto de atributos. São estas as ideias de partida que utilizámos no algoritmo que apresentamos nesta secção.

O objectivo é seleccionar um subconjunto A de atributos relevantes utilizando o algoritmo de processamento de partições progressivas PPP. Pretendemos que o conjunto de atributos a seleccionar seja obtido observando os modelos construídos com subconjuntos do conjunto de dados. Evitamos assim, que a selecção de atributos seja

feita considerando a totalidade dos dados o que nos permitirá poupar tempo de execução.

O algoritmo PPP constrói vários modelos que nos poderão indicar os atributos mais relevantes. Decidimos observar os atributos que foram escolhidos pelos modelos gerados pelo algoritmo PPP.

Outra questão que se pode colocar é, como combinar os vários subconjuntos de atributos seleccionados pelos vários modelos gerados?

Para podermos elaborar um algoritmo, decidimos observar, para um determinado conjunto de dados os vários subconjuntos de atributos escolhidos pelos modelos gerados pelo algoritmo PPP.

Utilizando o conjunto *TASK1* e aplicando algoritmo PPP, obtivemos vários modelos (um em cada iteração do algoritmo PPP) e observámos os atributos seleccionados por esses.

A Figura 4-1 mostra os atributos que foram escolhidos em cada modelo. Cada linha representa um modelo, sendo ordenados por ordem crescente do tamanho da proporção utilizada para os construir. Cada coluna representa um atributo, os números na linha inicial representam o número dos atributos (existem 71) e constituem uma legenda para as colunas. Um símbolo ‘X’ significa que o atributo apareceu pela primeira vez, enquanto que ‘x’ significa que o atributo já apareceu num modelo anterior.

Utilizando a Figura 4-1 observamos o seguinte:

- O número de atributos seleccionados nos vários modelos (28 atributos) representam uma pequena parte do número total de atributos (71).
- Alguns atributos aparecem constantemente em todos os modelos (ex. atributos 2 e 3).
- Atributos escolhidos por um modelo podem não estar presentes noutras modelos (ex. atributo 1).

Nº	Propor.	Atributos																			Nº atr		
		1	2	3	4	5	6	8	9	10	23	29	33	35	37	38	39	40	42	43	54		
1	1%	X	X																		X	3	
2	2%	X	x	x		X												X	X	X	x	x	10
3	4%	x	x	x													X	x	x	x	X	8	
4	8%	x	x	x	X		X				X					X	x			x		9	
5	16%	x	x		x			X	X		X	X	X	X	x		x		x	x	X	13	
6	32%	x	x		X	x	x					X				x	x		X	x		10	
7	64%	x	x		x	x		X	X		X	x				x	x		x			11	

Figura 4-1 — Atributos seleccionados por uma sequência de Modelos

Pretendemos decidir a “relevância” dos atributos pela relevância que tiveram nos vários modelos construídos com subconjuntos diferentes do conjunto de dados inicial. Para não perder nenhum atributo que poderá ser potencialmente relevante a forma mais segura é seleccionar todos atributos que foram seleccionados em pelo menos um dos modelos. Vamos considerar como relevantes os atributos pertencentes à *reunião* dos conjuntos de atributos que cada modelo selecciona.

Quantos modelos devemos construir para seleccionar o subconjunto de atributos relevantes?

Resolvemos esta questão definindo um *critério de estabilidade* que permite ao algoritmo de selecção de atributos baseado no processamento por partições progressivas (usamos a sigla SAPPP para este algoritmo) não ter que considerar todo o conjunto de treino. Esse critério baseia-se na ideia de estabilidade de um dado conjunto de atributos em relação à reunião dos conjuntos de atributos previamente encontrados. Consideremos que  $Reunião_i$  é o conjunto de atributos seleccionados nas iterações até i (inclusive), isto é,

$$Reunião_i = \bigcup_{k \leq i} A_k$$

em que  $A_k$  é o conjunto de atributos seleccionado pelo modelo construído na iteração k.

Adoptámos a definição de estabilidade, que estabelece o que aconteceria ao modelo  $M_i$  se se retirassem os atributos novos. Vamos tentar prever o maior aumento de erro que poderia acontecer se ao modelo  $M_i$  retirássemos todos os atributos que não pertencem a

$Reunião_{i-1}$ , se esse aumento for menor que um dado valor (parâmetro *estabilidade*) decidimos parar o algoritmo escolhendo o conjunto de atributos  $Reunião_i$ .

Representámos na Figura 4-2 o funcionamento do algoritmo de selecção de atributos baseado em processamento por partições progressivas (SAPP).

Em cada iteração  $i$  do algoritmo é necessário um subconjunto de dados de treino e um conjunto de atributos a que chamámos  $Reunião$ . Em cada uma das 3 primeiras iterações o algoritmo apenas constrói modelos baseados no subconjunto de treino respectivo e acrescenta os atributos novos (que não aparecem na cobertura) do modelo ao conjunto  $Reunião$ .

A partir da 4<sup>a</sup> iteração, o algoritmo vai utilizar o critério de paragem de forma a evitar a construção de modelos com subconjuntos grandes. O critério de paragem utilizado observa a diferença no erro da árvore, construída com o subconjunto de treino, e a árvore resultante da anterior se forem retirados os novos atributos. O algoritmo pára se a diferença de erro for inferior ao parâmetro estabilidade.

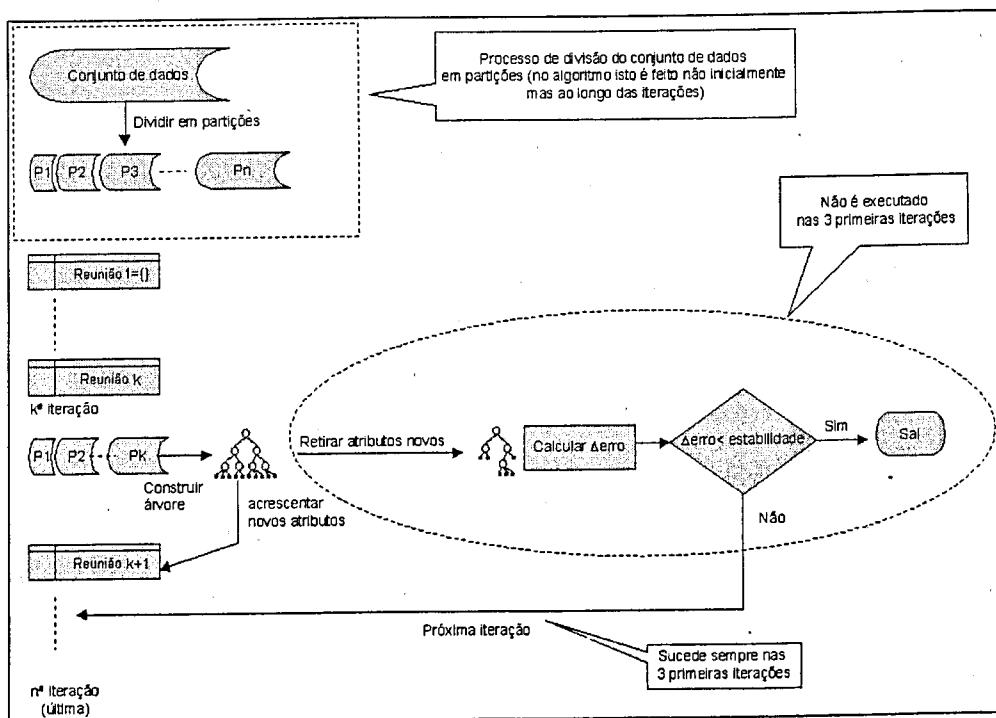


Figura 4-2 — Esquema de funcionamento do algoritmo de selecção de atributos (SAPP)

Em seguida apresentamos o algoritmo SAPPP com alguns pormenores adicionais.

### ALGORITMO

Parâmetros:

$x_0$  - proporção inicial

$r$  - razão da progressão geométrica

*estabilidade* – valor máximo admitido para a descida na taxa de acertos eliminando novos atributos

### INÍCIO

$p \leftarrow x_0$

$Reunião \leftarrow \{\}$

$i \leftarrow 1$

### Repetir

- (a) Retirar um subconjunto B (estratificado) de S com proporção p.
- (b) Usar o algoritmo de aprendizagem de árvores de decisão A para construir um modelo M para B  
Atr  $\leftarrow$  atributos utilizados no modelo M;  
Novos  $\leftarrow$  Atr  $- (Atr \cap Reunião)$
- (c) Retirar de M os nós (ficam folhas) que testem atributos  $\in$  Novos
- (d)  $\Deltaerro \leftarrow$  acréscimo de erro em M após retirar atributos  
 $Reunião \leftarrow Reunião \cup$  Novos  
Se  $i \geq 3$  e  $\Deltaerro \leq$  *estabilidade* então sai do ciclo  
 $p \leftarrow p * r$   
Se  $p > 1$  e  $p/r < 1$  então  $p \leftarrow 1$   
 $i \leftarrow i + 1$

Até  $p > 1$

Retorna o conjunto  $Reunião$  (conjunto de atributos seleccionado)

FIM

Figura 4-3 — Algoritmo de Seleção de Atributos (SAPPP)

Os passos (c) e (d) do algoritmo são ilustrados no exemplo apresentado nos seguintes parágrafos.

Suponhamos que num determinado ponto do algoritmo SAPPP o conjunto  $Reunião$  é constituído pelos atributos A1, A2, A3, A5, A7, A8, A9 e A10. Suponhamos ainda que o modelo M, construído no passo (b), é a árvore de decisão (“árvore inicial”) representada na Figura 4-4, na qual os nós estão representados com círculos e as folhas com quadrados. Todos os atributos que aparecem no modelo M (árvore inicial na Figura 4-4), exceptuando A4 e A6, estão no conjunto  $Reunião$ , isto é apareceram dois atributos novos (representados nos nós sombreados)

O que vamos fazer é cortar esses atributos novos da “árvore inicial”, obtendo uma árvore que designamos “árvore após cortar atributos”. Se num dado nó da árvore um

atributo novo é usado (na condição) então é transformado numa folha e todos os seus nós e folhas descendentes desaparecem.

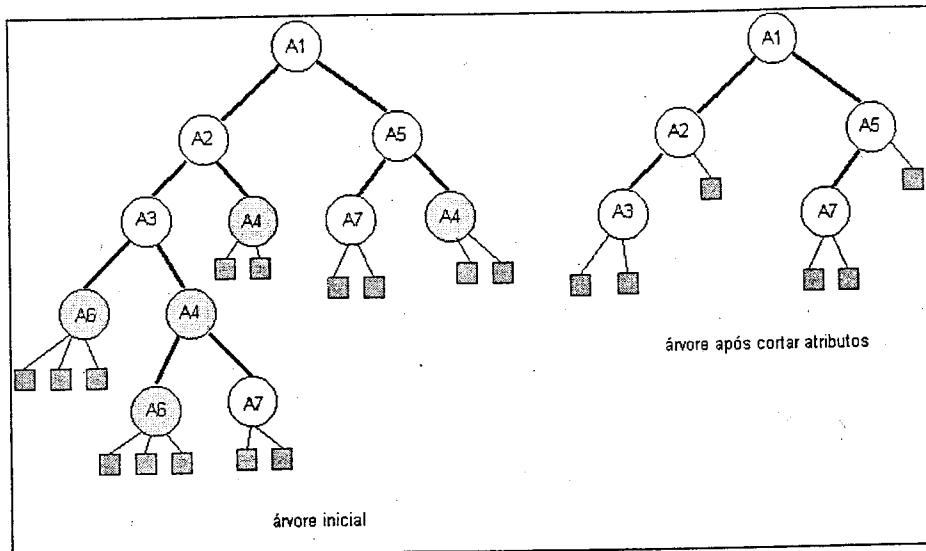


Figura 4-4 — Eliminação de atributos numa árvore de decisão

Para calcular o aumento do erro (redução da taxa de acertos) temos que calcular a taxa de erro na “árvore inicial” e na “árvore após cortar atributos”.

Na árvore de decisão inicial, em cada folha, encontra-se o número de exemplos (do conjunto de treino) que pertencem a cada classe. Representamos esses números utilizando a seguinte notação:

num caso particular temos,

$\#exemplos(1,1)$  ... nº de exemplos na folha 1 que pertencem à classe 1

e em geral,

$\#exemplos(f,c)$  ... nº de exemplos na folha f que pertencem à classe c

Assim, temos a distribuição das classes dos exemplos (embora não representada na Figura 4-4) que “caem” em cada folha e nó (soma das folhas descendentes). Essas distribuições vão-nos servir para calcular o erro cometido pela “árvore após cortar atributos” no conjunto que foi utilizado para construir a “árvore inicial”.

Para calcular o erro numa determinada árvore percorremos todas as folhas e contamos os erros cometidos em cada folha, utilizando a informação sobre o número de exemplos por folha e classe. Em seguida dividimos o total de erros das folhas pelo número total de casos que “caem” nessas folhas. A *diferença entre os erros* na “árvore inicial” e na “árvore após cortar atributos” é utilizada para decidir se o conjunto dos atributos novos *são muito importantes*. Se a diferença for inferior ou igual ao parâmetro estabilidade consideraremos que não vale a pena continuar o processo, isto é o conjunto de atributos seleccionados é a *Reunião* mais o conjunto dos novos atributos.

Em seguida apresentamos o processo que utilizámos para medir os erros (no treino) nas árvores baseando-nos nas distribuições de casos por classe que temos em cada folha.

Considerando a notação que já apresentámos para  $\#exemplos(f,c)$ , o algoritmo que utilizámos para calcular o erro das árvores de decisão encontra-se apresentado na Figura 4-5. No fundo o algoritmo vai percorrer todas as folhas, e para cada uma vai calcular quantos exemplos não pertencem à classe maioritária (classificação dada), que no fundo representa o número de erros que ocorrem nessa folha.

```
TotalAcertos ← 0
TotalCasos ← 0
Para cada folha f fazer
    Maior ← 1
    Para cada classe c fazer
        Se  $\#exemplos(f,c) > \#exemplos(f,Maior)$  então Maior ← c
        TotalCasos ← TotalCasos +  $\#exemplos(f,c)$ 
    Fim para
    TotalAcertos ← TotalAcertos +  $\#exemplos(f,Maior)$ 
Fim para
Erro ←  $(TotalCasos - TotalAcertos) / TotalCasos$ 
```

Figura 4-5 — Algoritmo para o cálculo do erro

O subconjunto dos atributos seleccionados pelo algoritmo SAPPP pode então ser utilizado para diminuir a dimensão do conjunto de dados inicial, havendo em seguida a possibilidade de:

- utilizar o mesmo algoritmo de aprendizagem (algoritmo base) para construir um modelo usando todos os casos do conjunto de dados mas sem os atributos não seleccionados. Podemos referir esta combinação

como sendo um algoritmo, e representamos este algoritmo pela abreviatura SAPPP+PT.

- utilizar um outro algoritmo de aprendizagem para construir um modelo usando conjunto de dados reduzido pela eliminação de atributos.
- aplicar o algoritmo PPP ao conjunto de dados reduzido. Na realidade isto constitui uma combinação do algoritmo PPP com o algoritmo SAPPP. É previsível que a redução do número de atributos restrinja a procura do modelo feita pelo algoritmo de aprendizagem, levando a que o número de casos necessário para construir bom modelo diminua. Podemos referir esta combinação como sendo um algoritmo, e representamos este algoritmo pela abreviatura SAPPP+PPP.

#### 4.5 Considerações sobre o algoritmo

O algoritmo SAPPP não selecciona necessariamente o menor conjunto de atributos relevantes considerando o conjunto de dados e o algoritmo de aprendizagem (algoritmo base). Se no conjunto de atributos existirem dois atributos com exactamente os mesmos valores para cada caso, então certamente não precisaríamos dos dois para construir o modelo. Esse dois atributos poderiam no entanto aparecer no conjunto de atributos seleccionados.

O que obtemos no final é apenas um subconjunto de atributos que não deverá conter atributos irrelevantes. Esperamos que reduzindo desta forma o conjunto de atributos, o tempo de execução necessário para o algoritmo de aprendizagem construir o modelo seja também reduzido.

Uma outra consequência da redução do número de atributos é diminuição da complexidade dos modelos construídos, o que traz a vantagem dos modelos serem mais facilmente interpretáveis.

No próximo Capítulo iremos avaliar este algoritmo de selecção de atributos, comparando os resultados, em tempo de execução e taxa de acertos, entre os algoritmos PT (algoritmo base processando todos os dados), o algoritmo SAPPP+PT e o algoritmo SAPPP+PPP.

## 5 Avaliação do algoritmo de selecção de atributos SAPPP

### 5.1 Introdução

No Capítulo anterior descrevemos um método de selecção de atributos, que designámos algoritmo de selecção de atributos baseado no processamento por partições progressivas (SAPPP). Este método observa os atributos seleccionados pelos modelos construídos utilizando uma versão do algoritmo PPP (com um algoritmo de aprendizagem de árvores de decisão como algoritmo base). É gerada uma sequência de modelos (neste caso árvores de decisão) construídos com conjuntos de treino progressivamente maiores. São observados os atributos seleccionados por estes modelos, e esses atributos são acumulados (acrescentados) num conjunto que chamámos *Reunião*. O algoritmo SAPPP contém um critério de paragem que decide em cada iteração se o conjunto de atributos seleccionados até ao momento (*Reunião*) é estável. Este conceito de estabilidade baseia-se em quantificar a diminuição na taxa de acertos (no treino) quando retiramos ao último modelo construído os atributos que não pertencem (i.e. os atributos novos) ao conjunto *Reunião*. Se a perda em taxa de acertos, quando retiramos os atributos novos do último modelo, for inferior ao parâmetro *estabilidade* o algoritmo SAPPP pára, e retorna todos os atributos seleccionados até ao momento como conjunto de atributos relevantes.

O algoritmo SAPPP pode ser usado para seleccionar atributos reduzindo dessa forma o conjunto de dados apenas aos atributos seleccionados.

Para avaliar este algoritmo vamos ver como se comportam os dois algoritmos que resultam da combinação deste com os algoritmos PT (algoritmo base processando a totalidade do conjunto de Treino) e PPP.

Representamos as combinações atrás referidas, que podemos ainda designar de algoritmos ou variantes da seguinte forma:

- SAPPP+PT – Corresponde a utilizar o algoritmo SAPPP para seleccionar atributos, e depois, com o conjunto de dados restrito a esses atributos usar o algoritmo PT para construir o modelo.
- SAPPP+PPP – Corresponde a utilizar o algoritmo SAPPP para seleccionar atributos, e depois, com o conjunto de dados restrito a esses atributos usar o algoritmo PPP para construir o modelo.

A ideia é verificar se o algoritmo SAPPP combinado com os algoritmos PT e PPP ajuda no nosso objectivo de aprender modelos em menos tempo sem diminuir muito a taxa de acertos. Por essa razão iremos comparar o algoritmo SAPPP+PT com o algoritmo PT, e também o algoritmo SAPPP+PPP com o algoritmo PT.

Realizámos, para esse efeito, várias experiências de comparação sobre 4 conjuntos de dados, que escolhemos por terem simultaneamente um número relativamente elevado de atributos e casos.

Na próxima secção descrevemos a metodologia de avaliação que utilizámos para comparar as variantes acima mencionadas com o algoritmo PT. O objectivo desta avaliação é comparar não só, as taxas de acertos obtidas pelos algoritmos SAPPP+PT e SAPPP+PPP com as obtidas pelo algoritmo PT, mas também, os tempos gastos por cada um dos algoritmos.

## 5.2 Metodologia de avaliação

Para avaliar o algoritmo SAPPP decidimos comparar os algoritmos SAPPP+PT e SAPPP+PPP com o nosso ponto de referência, isto é, o algoritmo PT.

Utilizámos 4 conjuntos de dados e para cada um deles realizámos experiências com o objectivo de comparar os resultados (em taxa de acertos e tempos consumidos) entre os algoritmos SAPPP+PT e PT, bem como entre os algoritmos SAPPP+PPP e PT.

O algoritmo base que utilizámos foi o C5.0 [Quin98], algoritmo de indução de árvores de decisão. Os valores dos parâmetros que usámos no algoritmo SAPPP foram para o  $r$  (regula o crescimento do tamanho das partições) o valor 1,5, e para o parâmetro

*estabilidade* (utilizado no critério de paragem) o valor 0,001. Para o algoritmo PPP (na combinação SAPPP+PPP) utilizámos os parâmetros  $r=1,5$  e  $\varepsilon=0,001$ .

Sobre cada conjunto utilizámos o procedimento de validação cruzada para comparar os diferentes algoritmos mencionados acima. Com o procedimento anterior obtemos, para cada algoritmo, 10 medições para taxa de acertos e para os tempos.

Para comparar os algoritmos, segundo os pares já referidos, observámos as medidas definidas nos seguintes pontos:

Usamos o símbolo X para designar tanto o algoritmo PPP como o algoritmo PT.

Dessa forma estamos a definir através de cada expressão que contenha o símbolo X uma expressão considerando  $X=PPP$  e outra com  $X=PT$ . O simbolo Y significa qualquer um dos algoritmos SAPPP+PT, SAPPP+PPP, e PT.

- Diferença nas taxas de acertos na iteração i :

$$A_{(SAPPP+X)_i} - A_{PT_i}$$

- Rácio das taxas de acertos obtidas pelos algoritmos na iteração i :

$$AR_{(SAPPP+X)_i} = \frac{A_{(SAPPP+X)_i}}{A_{PT_i}}$$

- Rácio dos tempos consumidos pelos algoritmos na iteração i :

$$TR_{(SAPPP+X)_i} = \frac{T_{(SAPPP+X)_i}}{T_{PT_i}}$$

- Média das taxas de acertos obtidas pelo algoritmo SAPPP+Y ao longo das 10 iterações:

$$\bar{A}_Y = \frac{1}{10} \sum_{i=1}^{10} A_{Y(i)}$$

- Média dos tempos consumidos pelo algoritmo SAPPP+PT (SAPPP+PPP e PT) ao longo das 10 iterações:

$$\bar{T}_Y = \frac{1}{10} \sum_{i=1}^{10} T_{Y(i)}$$

Para agregar os 10 rácios (das taxas e dos tempos) obtidos decidimos utilizar a média geométrica. Definimos estas medidas da seguinte forma:

Média geométrica dos Ráios das taxas de acertos (e tempos) obtidas pelos algoritmos SAPPP+X em relação à obtida pelo algoritmo PT são definidas da seguinte forma:

$$\overline{AR}_{SAPPP+X} = \sqrt[10]{\prod_{i=1}^{10} \frac{A_{(SAPPP+X)(i)}}{A_{PT(i)}}}$$

para o caso dos taxas de acertos e

$$\overline{TR}_{SAPPP+X} = \sqrt[10]{\prod_{i=1}^{10} \frac{T_{(SAPPP+X)(i)}}{T_{PT(i)}}}$$

para o caso dos tempos.

Um bom resultado do algoritmo SAPPP+X implica que  $\overline{AR}_{SAPPP+X}$  está muito próximo de 1 e  $\overline{TR}_{SAPPP+X}$  é menor que 1. Quanto menor for  $\overline{TR}_{SAPPP+X}$  maior será o tempo pouparado pelo algoritmo SAPPP+X.

Na próxima secção apresentamos os resultados que obtivemos.

### 5.3 Conjunto **TASKI**

Já descrevemos este conjunto de dados no Capítulo 3. Relembramos apenas que este contém 111077 casos, 72 atributos incluindo o atributo com o valor da classe.

### 5.3.1 Análise dos resultados obtidos

Em relação aos tempos consumidos pelos três algoritmos, como se observa na Figura 5-1, o algoritmo SAPPP+PPP foi o mais rápido dos três algoritmos.

Podemos observar que o algoritmo SAPPP+PT demora pelo menos o dobro do tempo do algoritmo SAPPP+PPP.

A média geométrica dos rácios dos tempos foram  $\overline{TR}_{SAPPP+PT} = 0,0048$ , e  $\overline{TR}_{SAPPP+PPP} = 0,0015$ . Estes valores significam que o algoritmo SAPPP+PT era 208 vezes mais rápido que o algoritmo PT, e o algoritmo SAPPP+PPP era 649 vezes mais rápido que o algoritmo PT.

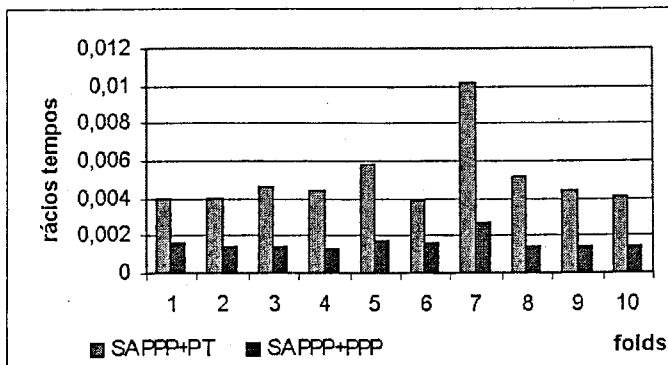


Figura 5-1 — Dados TASKI - Rácio dos tempos

Em relação a este conjunto de dados podemos concluir que o algoritmo SAPPP+PPP foi o mais rápido. O algoritmo SAPPP conseguiu reduzir o tempo gasto pelo algoritmo PT para ambas as combinações SAPPP+PT e SAPPP+PPP.

Quanto às taxas de acertos obtidas, tanto a combinação SAPPP+PT, como a SAPPP+PPP provocam um decréscimo na taxa de acertos (menor que 1%) em relação à obtida pelo algoritmo PT, provavelmente motivada pela eliminação de alguns atributos significativos. A Tabela 5-1 mostra as médias das taxas de acertos pelos obtidos pelos três algoritmos.

Tabela 5-1 — Dados *TASKI* -Médias das taxas de acertos

Médias aritméticas das Taxas de acertos (em percentagens)		Diferenças		
$\bar{A}_{SAPPP+PT}$	$\bar{A}_{SAPPP+PPP}$	$\bar{A}_{PT}$	$\bar{A}_{SAPPP+PT} - \bar{A}_{PT}$	$\bar{A}_{SAPPP+PPP} - \bar{A}_{PT}$
$66,85 \pm 0,45$	$66,87 \pm 0,50$	$67,82 \pm 0,26$	-0,97	-0,95

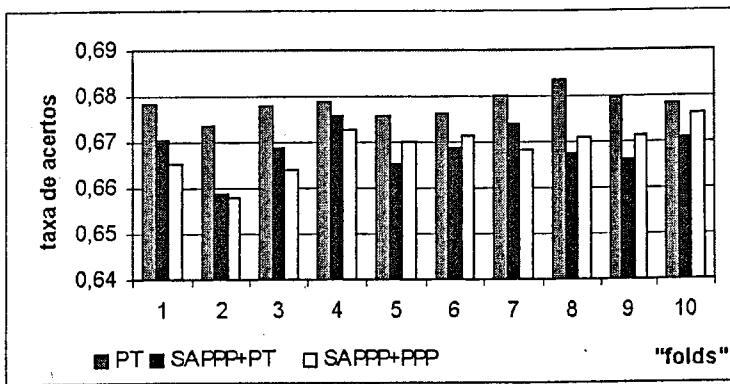
Na Figura 5-2 estão representadas as taxas de acertos ao longo das 10 iterações para os três algoritmos. Observamos que o algoritmo PT teve melhores resultados, em relação aos outros algoritmos, em todas as iterações da validação cruzada.

A média geométrica do rácio das taxas entre SAPPP+PT e PT foi 0,9857, o que quer dizer que o algoritmo SAPPP+PT obtém 98,57% da taxa obtida pelo algoritmo PT.

A média geométrica do rácio das taxas entre SAPPP+PPP e PT foi 0,9861, o que quer dizer que o algoritmo SAPPP+PPP obtém 98,61% da taxa obtida pelo algoritmo PT.

Podemos observar que em algumas iterações o algoritmo SAPPP+PPP obteve taxas de acertos ligeiramente maiores que as obtidas pelo algoritmo SAPPP+PT, o que se reflete nas médias descritas na Tabela 5-1.

Em resumo ambas as versões SAPPP+PT como SAPPP+PPP ganharam bastante tempo em relação a PT, à custa duma pequena diminuição da taxa de acertos.

Figura 5-2 — Dados *TASKI* – comparação das taxas de acertos

## 5.4 Conjunto sobre Cobertura Florestal

Já descrevemos no Capítulo 3 este conjunto de dados (“Forest cover type”). Relembramos apenas que este contém 581012 casos, 55 atributos incluindo o atributo com o valor da classe.

### 5.4.1 Análise dos resultados obtidos

Os tempos consumidos pelos três algoritmos, como se observa na Figura 5-3, mostram que o algoritmo SAPPP+PT foi o mais rápido dos três algoritmos, já que os rácios entre os tempos deste e os tempos do algoritmo PT foram todos menores que 1.

A média geométrica dos rácios dos tempos, entre o algoritmo SAPPP+PT e o algoritmo PT foi 0,87.

Tal como se observou no Capítulo 3, neste conjunto o algoritmo PPP não conseguiu poupar tempo, e assim também o algoritmo SAPPP+PPP foi sempre mais lento que o algoritmo PT, sendo a média geométrica dos rácios dos tempos, entre este e o algoritmo PT, igual a 2,7.

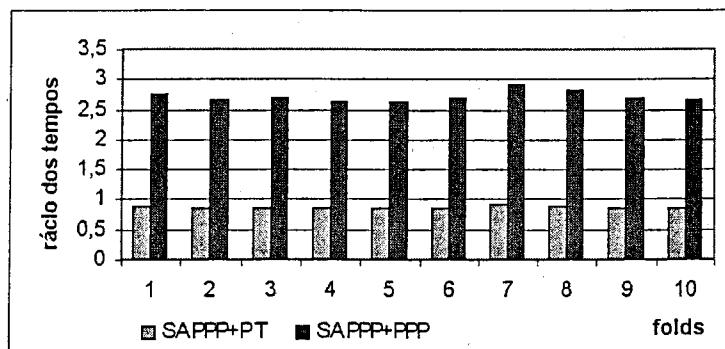


Figura 5-3 — Dados *Forest cover type* – rácios dos tempos

Em relação às taxas de acertos obtidas pelos algoritmo SAPPP+PT e SAPPP+PPP, notamos que são muito próximas, embora ligeiramente inferiores, às obtidas pelo algoritmo PT.

Tabela 5-2 — Dados Forest Cover Type – Médias das taxas de acertos

Médias aritméticas das Taxas de acertos (em percentagens)			Diferenças	
$\bar{A}_{SAPPP+PT}$	$\bar{A}_{SAPPP+PPP}$	$\bar{A}_{PT}$	$\bar{A}_{SAPPP+PT} - \bar{A}_{PT}$	$\bar{A}_{SAPPP+PPP} - \bar{A}_{PT}$
$94,34 \pm 0,12$	$94,34 \pm 0,12$	$94,55 \pm 0,13$	-0,11	-0,11

Na Figura 5-4 estão representadas as taxas de acertos ao longo das 10 iterações para os três algoritmos. Observamos que o algoritmo PT teve melhores resultados, em relação aos outros algoritmos.

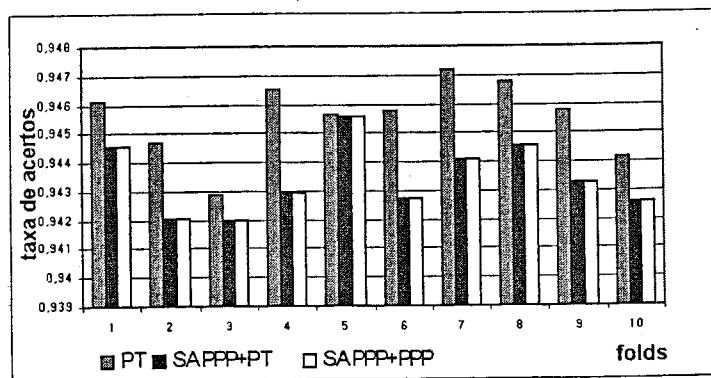


Figura 5-4 — Forest cover type – taxa de acertos

A média geométrica do rácio das taxas entre SAPPP+PT e PT, bem como a média geométrica do rácio das taxas entre SAPPP+PPP e PT, tiveram ambas o valor 0,9977, o que quer dizer que os algoritmos SAPPP+PT e SAPPP+PPP obtém 99,77% da taxa obtida pelo algoritmo PT.

Concluímos que para este conjunto o algoritmo SAPPP+PT teve bons resultados na taxa de acertos (não diminuiu muito) e houve uma modesta poupança de tempo pois algoritmo SAPPP+PT gastou apenas 87% do tempo gasto pelo algoritmo PT.

O algoritmo SAPPP+PPP não conseguiu nenhum ganho de tempo, mas também não perdeu muito na taxa de acertos (cerca 0,11%). O facto do algoritmo PPP, executado após o algoritmo SAPPP, não ter ganho tempo em relação a PT, justifica-se pelo facto deste ter escolhido como ponto de paragem a proporção 100% (ver resultados para o mesmo conjunto no Capítulo 3).

## 5.5 Conjunto *Census*

Em seguida vamos descrevemos os resultados que obtivemos com o conjunto *Census*, conjunto que já foi descrito no Capítulo 3. Vamos apresentar os resultados de forma mais resumida apresentando apenas as médias das taxas de acertos e tempos consumidos para os algoritmo SAPPP+PPP, SAPPP+PT e PT.

### 5.5.1 Análise dos resultados obtidos

Em relação aos tempos consumidos (ver Tabela 5-3) pelos três algoritmos neste conjunto observamos que a redução de tempo é bastante considerável em ambos os casos. O algoritmo SAPPP+PT consome apenas 30% do tempo consumido pelo algoritmo PT, e o algoritmo SAPPP+PPP consome apenas 9% do tempo consumido pelo algoritmo PT.

Tabela 5-3 — Dados *Census* – Médias dos tempos

Médias aritméticas dos tempos consumidos		
$\bar{T}_{SAPPP+PT}$	$\bar{T}_{SAPPP+PPP}$	$\bar{T}_{PT}$
$83 \pm 6,4$	$23 \pm 0,1$	$270 \pm 64$

Quanto às taxas de acertos podemos observar (Tabela 5-4) as reduções em relação ao algoritmo PT, quando utilizámos os algoritmos SAPPP+PT e SAPPP+PPP. A diferença é 0,18% para o primeiro, e de 0,77 para o segundo.

Tabela 5-4 — Dados *Census* – Médias das taxas de acertos

Médias aritméticas das Taxas de acertos (em percentagens)			Diferenças	
$\bar{A}_{SAPPP+PT}$	$\bar{A}_{SAPPP+PPP}$	$\bar{A}_{PT}$	$\bar{A}_{SAPPP+PT} - \bar{A}_{PT}$	$\bar{A}_{SAPPP+PPP} - \bar{A}_{PT}$
$95,14 \pm 0,17$	$94,55 \pm 0,17$	$95,32 \pm 0,14$	-0,18	-0,77

Concluímos que tanto os algoritmos SAPPP+PT como SAPPP+PPP obtém uma redução apreciável, em relação ao algoritmo PT, no tempo de processamento à custa de uma pequena redução na taxa de acertos.

## 5.6 Conjunto *KDD-Cup2000\_CL*

Em seguida vamos descrevemos os resultados que obtivemos com o conjunto *KDD-Cup2000\_CL*, conjunto que já foi descrito no Capítulo 3. Vamos apresentar os resultados de forma mais resumida apresentando apenas as médias das taxas de acertos e tempos consumidos para os algoritmo SAPPP+PPP, SAPPP+PT e PT.

### 5.6.1 Análise dos resultados obtidos

Em relação aos tempos consumidos (ver Tabela 5-5) pelos três algoritmos neste conjunto observamos que existe uma redução de tempos quando utilizamos os nossos algoritmos. O algoritmo SAPPP+PT consome 70% do tempo consumido pelo algoritmo PT, e o algoritmo SAPPP+PPP consome apenas 17% do tempo consumido pelo algoritmo PT.

**Tabela 5-5 — Dados *KDD-Cup2000\_CL* – Médias dos tempos**

Médias aritméticas dos tempos consumidos		
$\bar{T}_{SAPPP+PT}$	$\bar{T}_{SAPPP+PPP}$	$\bar{T}_{PT}$
$91 \pm 6,4$	$23 \pm 1,4$	$129 \pm 4,2$

No que diz respeito às taxas de acertos (Tabela 5-6), o algoritmo SAPPP+PT perdeu 0,57% na taxa de acertos em relação ao algoritmo PT, valor não muito elevado. A perda na taxa de acertos do algoritmo SAPPP+PPP em relação ao algoritmo PT foi um bocado elevada, assim como aconteceu nos resultados neste conjunto usando o algoritmo PPP (ver Capítulo 3)

**Tabela 5-6 — Dados *KDD-Cup2000\_CL* – Médias das taxas de acertos**

Médias aritméticas das Taxas de acertos (em percentagens)			Diferenças	
$\bar{A}_{SAPPP+PT}$	$\bar{A}_{SAPPP+PPP}$	$\bar{A}_{PT}$	$\bar{A}_{SAPPP+PT} - \bar{A}_{PT}$	$\bar{A}_{SAPPP+PPP} - \bar{A}_{PT}$
$94,76 \pm 0,54$	$91,97 \pm 0,43$	$95,33 \pm 0,34$	-0,57	-3,36

## 5.7 Comparação do algoritmo SAPPP+PT com o algoritmo PPP

Como pudemos observar nos resultados anteriores deste capítulo, o algoritmo SAPPP+PT foi o mais útil para reduzir o tempo de processamento em relação ao

algoritmo PT. O algoritmo SAPPP+PPP embora tivesse funcionado em alguns casos, outros não conseguiu poupar tempo em relação ao algoritmo PT.

Uma questão que se poderá colocar é como se comparam os algoritmos PPP, apresentado no Capítulo 2, e o algoritmo SAPPP+PT.

Será preferível usar o algoritmo PPP ou o algoritmo SAPPP+PT?

Utilizando os resultados deste capítulo e os resultados do Capítulo 3, considerando apenas os conjuntos em comum, podemos observar que o algoritmo SAPPP+PT parece melhor na maior parte dos casos.

A Tabela 5-7 mostra os resultados que obtivemos nos 4 conjuntos para ambos os métodos. Os resultados apresentados mostram as médias aritméticas das taxas de acertos bem como as médias dos tempos consumidos. Representamos em letra carregada os valores que representam vantagens de um algoritmo em relação ao outro (no conjunto *TASK1* em relação aos tempos observa-se que 85 é menor 312, logo 85 é favorável e aparece em letra carregada).

O tempo gasto pelo algoritmo SAPPP+PT foi menor em quase todos os conjuntos. Apenas no conjunto *KDD-Cup2000\_CL* o tempo consumido pelo algoritmo PPP foi menor. No entanto a taxa de acertos é maior no caso do algoritmo SAPPP+PT.

Em relação às taxas de acertos, o algoritmo SAPPP+PT obteve melhores taxas em dois dos conjuntos (*Census* e *KDD-Cup2000\_CL*), iguais no conjunto *Forest Cover Type* e taxas piores apenas para o conjunto *TASK1*. Mesmo no conjunto *TASK1*, em que a taxa de acertos foi pior para o algoritmo SAPPP+PT, o tempo poupadado pelo algoritmo SAPPP+PT (3,6 vezes mais rápido) poderá justificar a preferência por este algoritmo em vez de PPP.

**Tabela 5-7 — Comparação dos algoritmos SAPPP+PT e PPP**

Conjuntos	Taxas de acertos		Tempos	
	$\bar{A}_{SAPPP+PT}$	$\bar{A}_{PPP}$	$\bar{T}_{SAPPP+PT}$	$\bar{T}_{PPP}$
<i>TASK1</i>	66,9	<b>67,6</b>	<b>85</b>	312
<i>Forest Cover Type</i>	94,3	94,3	<b>1380</b>	4627
<i>Census</i>	<b>94,8</b>	94,2	<b>91</b>	254
<i>KDD-Cup2000_CL</i>	<b>95,1</b>	94,6	83	<b>26</b>

O algoritmo SAPPP+PT foi em geral mais útil na redução do tempo de processamento. O algoritmo PPP, quando aplicado aos conjuntos *Forest Cover Type* e *KDD\_Cup2000\_CL*, não conseguiu reduzir o tempo do processamento. Pelo contrário como vimos neste capítulo o mesmo não se passou com o algoritmo SAPPP+PT. Concluimos que é preferível em geral utilizar o algoritmo SAPPP+PT porque poderão existir riscos em utilizar o algoritmo PPP em alguns conjuntos de dados.

## 5.8 Conclusões

O objecto de aplicação do algoritmo SAPPP são os conjuntos com um elevado número de casos. As experiências que realizámos revelam que o algoritmo SAPPP poderá ser aplicado em conjuntos neste tipo de conjuntos.

Em conjuntos em que a dimensão do número de casos for baixa é natural que a eliminação de atributos tenha um impacto maior sobre a taxa de acertos. No entanto não testámos o algoritmo SAPPP neste tipo de conjuntos.

Para testar o nosso algoritmo utilizámos 4 conjuntos de dados com um elevado número de casos, e avaliámos a perda na taxa de acertos, bem como a poupança em tempo de processamento, do algoritmo SAPPP combinado com o algoritmo PT e com o algoritmo PPP.

A combinação SAPPP+PT teve resultados positivos nos 4 conjuntos de dados considerados nas experiências anteriores, sendo o resultado mais positivo o obtido com o conjunto *TASK1* em que a uma perda de 0,97% na taxa de acertos corresponde uma diminuição de tempo de 208 vezes.

No conjunto *Census* a redução de tempo é cerca de 70% do tempo em relação ao tempo gasto pelo algoritmo PT. Também no conjunto *KDD-Cup2000\_CL* observámos uma redução modesta de 30% no tempo em relação ao algoritmo PT.

No conjunto *Forest cover type* a perda na taxa de acertos foi 0,11% e o tempo de processamento do algoritmo SAPPP+PT foi 83% do tempo gasto pelo algoritmo PT.

A combinação SAPPP+PPP teve resultados positivos apenas nos conjunto *TASK1* e no conjunto *Census*, ambos com razoáveis reduções no tempo de processamento. O resultado mais impressionante foi obtido com o conjunto *TASK1* em que a redução de

tempo corresponde a 649 vezes menos tempo que o algoritmo PT. No entanto, esta versão parece ter alguns riscos. No conjunto *Forest cover type* o algoritmo SAPPP+PPP demorou mais tempo que o algoritmo PT, o que poderíamos adivinhar pelo resultado do algoritmo PPP neste mesmo conjunto (ver Capítulo 3).

Concluímos que a combinação SAPPP+PT poderá ser aplicada em grandes conjuntos de dados reduzindo-se inicialmente o número de atributos, e depois utilizando o algoritmo base com a totalidade dos casos. Se houver uma redução suficiente do número de atributos o algoritmo SAPPP+PT deverá conseguir poupar algum tempo em relação ao algoritmo PT.

No próximo capítulo apresentaremos as conclusões finais desta Tese.



## 6 Conclusões Finais

### 6.1 Introdução

Neste capítulo pretendemos recapitular os objectivos e motivações principais do nosso trabalho, as técnicas que foram apresentadas para cumprir esses objectivos, bem como as principais conclusões da avaliação destas técnicas, relatadas nos Capítulos 3 e 5. Descrevemos também as linhas de trabalho futuro perspectivadas aquando da conclusão deste trabalho.

### 6.2 Motivação, objectivos e técnicas apresentadas

Actualmente as organizações industriais, governamentais, científicas entre outras, produzem e armazenam grandes quantidade de dados. No processo de tomada de decisões é essencial ter informações correctas, úteis e actuais, que permitam efectuar previsões bem como explicar as relações menos evidentes existentes nos dados .

A dimensão das bases de dados das quais pretendemos obter o conhecimento necessário às nossas decisões tem vindo a aumentar bastante. Este facto impulsionou o aparecimento de novas técnicas inteligentes de análise de dados, que constituem o substrato do novo campo de conhecimento denominado de Extracção de Conhecimento de Bases de Dados (ECBD).

Fayyad [Fay96] define o processo ECBD como um processo não trivial de identificar (procurar) nos dados padrões válidos, anteriormente desconhecidos, potencialmente úteis e interpretáveis. Este processo é constituído por várias fases, que incluem entre outras a redução da dimensão dos dados seguida pela extracção de padrões (construção de modelos) utilizando algoritmos de aprendizagem.

Em algumas situações pode ser impossível, devido a limitações de tempo e memória, o processamento de uma grande quantidade de dados utilizando os algoritmos de aprendizagem. Estes algoritmos ao processarem uma base de dados de grande dimensão

poderão demorar mais tempo que o admissível para aplicação pretendida, ou consumir toda a memória disponível no computador e portanto não obter nenhuma solução.

O objectivo principal deste trabalho era, no contexto dos problemas de classificação, estudar métodos de redução de dados e ainda combinar métodos de redução de dados. Focámos os nossos objectivos nos métodos de redução do número de casos e nos métodos de selecção de atributos relevantes.

Apresentámos um método de redução de casos, que designámos de algoritmo de processamento por partições progressivas (PPP), que visa construir um modelo utilizando apenas um subconjunto do conjunto de dados inicial, de forma a poupar tempo de processamento, não sacrificando os resultados finais do modelo. Podemos mais uma vez referir que o algoritmo PPP, porque vai construindo modelos sucessivamente mais correctos (com taxas de acertos maiores), aproxima-se em taxa de acertos do algoritmo PT (que processa todos os dados), e desta forma disponibiliza ao utilizador resultados parciais úteis. Do ponto do utilizador é muitas vezes útil ir obtendo, em períodos de tempo pequenos alguns modelos prévios com resultados razoáveis, sabendo que o sistema vai tentar construir modelos progressivamente melhores (filosofia “anytime results”).

Quanto à selecção de atributos desenvolvemos um método que designámos de algoritmo de selecção de atributos baseado no processamento por partições progressivas (SAPPP), que utiliza os resultados parciais do algoritmo PPP, combinando o método de redução do número de casos com um método de selecção de atributos (baseado nos atributos seleccionados pelas árvores de decisão).

### **6.3 Conclusões**

Os resultados que obtivemos nas nossas experiências de avaliação (Capítulo 3 e 5) dos métodos apresentados revelaram-se promissores na prossecução dos nossos objectivos e confirmaram as nossas expectativas iniciais.

No que diz respeito ao método PPP as experiências verificaram ser útil aplicá-lo a conjuntos de dados com um elevado número de casos. Relembramos que o algoritmo

PPP utiliza o parâmetro  $r$  para regular o crescimento das proporções dos subconjuntos, utilizados para construir os modelos intermédios no processo de procura do modelo final.

Utilizámos duas variantes do algoritmo PPP, uma utilizando o C5.0 como algoritmo base e outra que utiliza o algoritmo clemMLP que é uma implementação de redes MLP no pacote de software *Clementine*.

Nas experiências de avaliação envolvendo o algoritmo PPP considerámos os parâmetros  $\epsilon=0,001$ ,  $r=1,5$  e  $r=2$ , quando o algoritmo base utilizado foi o C5.0, e considerámos os parâmetros  $\epsilon=0,01$  e  $r=1,5$ , quando o algoritmo base utilizado foi o clemMLP.

No caso em que utilizámos o algoritmo clemMLP como algoritmo base modificámos o critério de paragem de PPP para que considerasse apenas uma janela com dois pontos (ver Capítulo 2). Para esta versão do critério de paragem o algoritmo PPP pára quando entre duas iterações sucessivas do algoritmo não se registe um aumento superior ou igual a  $\epsilon$  na taxa de acertos. A motivação para as experiências do algoritmo PPP com redes neurais reside no elevado tempo de processamento consumido por este tipo de algoritmos de aprendizagem.

Os resultados descritos nos próximos parágrafos dizem respeito às experiências realizadas utilizando o C5.0 como algoritmo base.

Na avaliação em conjuntos com um elevado número de casos (cerca de 100000 casos ou mais) o método PPP obteve bons resultados nos conjuntos *TASK1* e *Census*, mas o mesmo não se verificou nos conjuntos *Forest cover type* e *KDD-Cup2000\_CL*.

Nas experiências com o conjunto *TASK1* obtivemos um resultado bastante positivo, conseguindo-se sem perder muito na taxa de acertos obter uma redução de tempo de pelo menos 20 vezes.

Nas experiências com o conjunto *Forest cover type* o algoritmo PPP não conseguiu ser eficiente, para nenhum dos valores do parâmetro  $r$  (valor 1,5 e 2). No entanto viemos a observar que dificilmente um método de redução de casos poderia ser eficiente neste conjunto, visto que a sua curva de aprendizagem apresentava na parte final subidas bastante significativas.

Nas experiências com o conjunto *Census* os resultados obtidos são satisfatórios, observando-se que o algoritmo PPP é cerca de 10 vezes mais rápido que o algoritmo PT, perdendo no entanto 0,7% na taxa de acertos.

Nas experiências com o conjunto *KDD-Cup2000\_CL* os resultados obtidos não revelaram vantagens no uso do algoritmo PPP. Neste conjunto o algoritmo PPP foi mais lento (cerca do dobro do tempo) e com menor taxa de acertos (menos 1,15%) em relação ao algoritmo PT.

As experiências realizadas com conjuntos de dados de dimensão menor, um domínio de aplicação para o qual o algoritmo PPP não é adequado, revelaram que o algoritmo PPP não era em geral eficiente, observando-se que o algoritmo PT (algoritmo base processando a totalidade dos dados), gastava menos tempo a processar os dados que o algoritmo PPP. Apesar disto o algoritmo PPP, conseguiu para dois conjuntos de dados (*taskb\_hhold* e *shuttle*) ser modestamente mais rápido (cerca de 2 vezes) que o algoritmo PT (para  $r=1,5$ ).

Em relação à avaliação do algoritmo PPP utilizando clemMLP como algoritmo base realizámos experiências com os conjuntos *waveform* e *letter*.

No conjunto *letter* o tempo gasto pelo algoritmo PPP é cerca de 56% do tempo gasto pelo algoritmo PT, e a perda na taxa de acertos é cerca de 4%.

No conjunto *waveform* o algoritmo PPP gasta cerca de 40% do tempo gasto pelo algoritmo PT mas perde 3,6% na taxa de acertos. O resultado neste último conjunto tem o ponto positivo de ser melhor que o obtido pelo C5.0 consumindo a totalidade dos dados. Observámos que o algoritmo PPP dá-nos uma taxa de acertos bastante melhor que o C5.0 consumindo a totalidade dos dados (75,46% para o C5.0 e .79,9% para o PPP com clemMLP como base).

No que diz respeito ao algoritmo SAPPP as experiências (Capítulo 5) verificaram ser útil aplicá-lo a conjuntos de dados com um elevado número de casos.

As experiências de avaliação do algoritmo SAPPP basearam-se em 4 conjuntos de dados com um elevado número de casos. Os conjuntos utilizados foram o *TASK1*, o *Forest Cover Type*, o *KDD-Cup2000\_CL* e o *Census*.

Testamos duas aplicações do algoritmo SAPPP. A primeira que designámos por algoritmo SAPPP+PT consiste numa primeira fase em que escolhe o subconjunto de atributos relevantes usando SAPPP e depois, usando a totalidade do conjunto de dados mas considerando apenas os atributos seleccionados, constrói um modelo usando o algoritmo base. A segunda aplicação que designámos SAPPP+PPP consiste em aplicar o algoritmo PPP após o algoritmo SAPPP ter escolhido um conjunto de atributos.

A combinação mais positiva foi a SAPPP+PT, conseguindo-se para os 4 conjuntos de dados diminuições no tempo de processamento à custa de uma pequena redução na taxa de acertos. O resultado mais impressionante foi obtido no conjunto *TASK1* em que a redução de tempo foi na ordem de 208 vezes. No conjunto *Census* a redução de tempo é cerca de 70% do tempo em relação ao tempo gasto pelo algoritmo PT. Também no conjunto *KDD-Cup2000\_CL* observámos uma redução modesta de 30% no tempo em relação ao algoritmo PT. No conjunto *Forest cover type* a perda na taxa de acertos foi 0,11% e o tempo de processamento do algoritmo SAPPP+PT foi 83% do tempo gasto pelo algoritmo PT.

A combinação SAPPP+PPP teve resultados positivos apenas nos conjunto *TASK1* e no conjunto *Census*, ambos com razoáveis reduções no tempo de processamento. O resultado mais impressionante foi obtido com o conjunto *TASK1* em que a redução de tempo corresponde a 649 vezes menos tempo que o algoritmo PT.. Com os conjuntos *Forest cover type* e *KDD-Cup2000\_CL* não obtivemos resultados positivos, o que se justifica pelo facto de estarmos a tentar usar o algoritmo PPP (embora depois de SAPPP), que como vimos não funcionou nestes conjuntos.

Além das conclusões anteriores devemos referir que comparámos os algoritmo SAPPP+PT com PPP e retiramos como conclusão que é geral melhor utilizar o algoritmo SAPPP+PT (ver Capítulo 3).

#### 6.4 Trabalho futuro

Nesta secção apresentamos o trabalho futuro que poderá ser feito para enriquecer o trabalho que já realizamos, tanto nos objectivos que fixamos para esta Tese, como outros objectivos mais latos. Apresentamos nos seguintes pontos as várias opções de trabalho futuro:

1. Reavaliar os algoritmos que descrevemos em novos conjuntos de dados de grande dimensão, para comprovar os já promissores resultados iniciais. Entre outras experiências seria interessante aplicar o algoritmo SAPPP em conjuntos de dados em que o número de atributos é extremamente elevado, como por exemplo os que dizem respeito à extracção de padrões em textos (“text mining”), em que o número de atributos podem ser na ordem dos milhares, e verificar se é possível utilizar o método apresentado sem alterações.
2. Acrescentar ao algoritmo PPP uma forma inteligente de selecção de casos, e portanto mais sofisticada que a actual selecção aleatória de casos em subconjuntos de proporções crescentes. Como referimos no Capítulo 3 podemos usar os conceitos de selecção baseada nas incertezas de classificação (“uncertainty sampling”) [Mamit00]. Achamos que esta técnica poderá ser útil para obter resultados melhores em alguns conjuntos de dados.
3. Experimentar outras técnicas de selecção de atributos para além da selecção por árvores de decisão que foi utilizada no algoritmo SAPPP.
4. Utilizar métodos de combinação de predições (ou de classificadores) para tentar aproveitar os modelos construídos durante a execução do algoritmo PPP, e dessa forma tentar aumentar a taxa de acertos. Uma possibilidade é utilizar todos os modelos construídos para construir um novo modelo através da combinação de modelos. A outra possibilidade consiste em decidir qual a classificação final através da combinação de predições. Podemos entre outros utilizar o método descrito em [Mot00], que contém um procedimento de integração de modelos em forma de regras.



## REFERÊNCIAS

- [BerHand99] M. Berthold, D. J. Hand (eds.) “*Intelligent Data Analysis*”. Springer-Verlag 1999.
- [BlumLang97] Blum A., Langley P.: *Selection of relevant features and examples in machine learning*, Journal of Artificial Intelligence, Vol 97, Nos.1-2, pp 245-271, Elsevier.
- [Breiman96] L. Breiman. Bias, variance, and arcing classifiers. Technical report 460, Statistics Department, University of California, 1996.
- [Clementine] SPSS/ISL Clementine 5.0.4 1997. <http://www.isl.co.uk>
- [Fay96] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy (eds.). “*Advances in Knowledge Discovery and Data Mining*”. AAAI/MIT Press 96.
- [Frey99] Frey, L. J., Fisher, D. H. “Modeling decision tree performance with the power law”. In *Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics (1999)*. D. Heckerman and J. Whittaker, Eds., San Francisco, CA: Morgan Kaufmann.
- [John96] John, G. H., & Langley, P. (1996). “Static vs. dynamic sampling for data mining”, *Proceedings of the Second International Conference of Knowledge Discovery and Data Mining* (pp. 367--370). Portland: AAAI Press.
- [KDDCup2000] Cup part of SIGKDD 2000 – International Conference on Knowledge Discovery & Data Mining.  
In <http://www.ecn.purdue.edu/KDDCUP>, 2000.

- [kohavi95a] Ron Kohavi and Dan Sommerfield. Feature subset selection using the wrapper method: Overfitting and dynamic search space. AAAI Press, 1995.
- [kohavi95b] R. Kohavi. "A study of cross-validation and bootstrap for accuracy estimation and model selection". In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*. San Mateo, CA: Morgan Kaufmann, 1995.
- [Mamit00] Hiroshi Mamitsuka, Naoki Abe. "Efficient Mining from Large Databases by Query Learning". *Proceedings of the Seventeenth International Conference on Machine Learning*. Stanford University 2000.
- [Mitch97] Mitchell T. M., "Machine Learning", The McGraw-Hill Companies, Inc, 1997.
- [Mot00] Mota, A., "Construção de Modelos em Forma de Regras através da Integração de Teorias Parciais". Tese de Mestrado 2000.
- [PrincipeJ00] Principe J., Euliano N., Lefebvre W., "Neural and Adaptive Systems – Fundamentals Through Simulations", John Wiley & Sons, Inc, 2000.
- [Prov99] Provost, F., D. Jensen, and T. Oates (1999). Efficient progressive sampling. Technical report 99-14, Department of Computer Science, University of Massachusetts/Amherst.
- [Quin93] Quinlan, R. (1993) "C4.5: Programs for Machine Learning". Morgan Kaufmann.
- [Quin98] Quinlan, R. (1993) C5.0: "An Informal Tutorial. RuleQuest.  
<http://www.rulequest.com/see5-unix.html>
- [Weiln98] Weiss S. M., Indurkhya, N. "Predictive Data Mining – a practical guide". Morgan Kaufmann Publishers, Inc, 1998.

