Instituto de Ciências Matemáticas e de Computação

ISSN - 0103-2569

Uma Descrição dos Arquivos e Parâmetros Utilizados na Execução do Sistema AutoClass

Edson Takashi Matsubara
Claudia Aparecida Martins
Maria Carolina Monard

Nº 170

RELATÓRIOS TÉCNICOS DO ICMC

São Carlos
Agosto/2002

Uma Descrição dos Arquivos e Parâmetros Utilizados na Execução do Sistema AutoClass*

Edson Takashi Matsubara[†]

Claudia Aparecida Martins[†‡]

Maria Carolina Monard[†]

[†]Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação e Estatística
C.P. 668, 13560-970 - São Carlos, SP - Brasil
e-mail: {edsontm, cam, mcmonard}@icmc.usp.br

[‡]Universidade Federal de Mato Grosso
Instituto de Ciências Exatas e da Terra
Departamento de Ciência da Computação
Av. Fernando Corrêa da Costa s/n
78060-900, Cuiabá, MT - Brasil

Resumo: Algoritmos de *clustering* basicamente são utilizados para agrupar objetos (exemplos) em clusters (grupos). AutoClass é um algoritmo de *clustering* robusto, bastante referenciado na comunidade científica, que utiliza técnicas probabilísticas e o modelo de Bayes. Entretanto, a documentação disponível que auxilia o usuário na execução do AutoClass deixa a desejar. Assim, este trabalho tem como objetivo a descrição, de uma forma simplificada e organizada, dos arquivos de entrada e saída do AutoClass, bem como a descrição dos parâmetros mais importantes/utilizados para execução do mesmo, sem entrar em detalhes sobre a teoria probabilística que o suporta, a qual está bem documentada em diversas publicações.

Palavras Chaves: AutoClass, Aprendizado não Supervisionado, *Clustering*

Agosto 2002

*Trabalho realizado com auxílio do CNPq e CAPES.

Sumário

1 Introdução **1**

2 Técnicas de *Clustering* **2**

3 AutoClass - Uma Visão Geral de Execução **3**

3.1 Modo Busca . . . 3

3.1.1 Arquivo de Cabeçalho . . . 5

3.1.2 Arquivo de Dados . . . 7

3.1.3 Arquivo de Modelos . . . 8

3.1.4 Arquivo de Parâmetros de Busca . . . 9

3.2 Modo Relatórios . . . 11

3.2.1 Relatório de Referência Cruzada por Cluster . . . 11

3.2.2 Relatório de Referência Cruzada por Exemplo . . . 12

3.2.3 Relatório de Influência dos Atributos . . . 13

3.3 Modo Predição . . . 13

4 Conclusão **14**

Referências **15**

Lista de Figuras

1 Técnicas de *Clustering*: (a) Otimização (b) *Clumping* (c) Probabilística (d) Hierárquica . . . 2

2 Esquema para Busca `(search)` e `Relatório (report)` . . . 4

3 Esquema de Predição `(predict)` . . . 4

4 Arquivo de Cabeçalho `(.hd2)` . . . 7

5 Arquivo de Dados `(.db2)` . . . 7

6 Arquivo de Modelos `(.model)` . . . 8

7 Arquivo de Modelos `(.model)` `Exemplo Extra` . . . 9

8 Relatório `(.class-text-n)` . . . 12

9 Relatório `(.case-text-n)` . . . 12

10 Conjunto de Dados Teste . . . 13

11 Relatório `.class-text-n (Teste)` . . . 14

Lista de Tabelas

1 Descrição dos Arquivos de Entrada `(search)` . . . 5

2 Tabela de Tipos de Atributos . . . 6

3 Descrição dos Modelos . . . 8

[1] Este documento foi elaborado com o LaTeX, sua bibliografia é mantida com o auxílio da ferramenta BibView e gerada automaticamente pelo BibTeX

1 Introdução

Na atual era da *Internet*, *Intranets* e *Data Warehouses*, os paradigmas fundamentais da análise de dados clássica estão sofrendo modificações. A quantidade de dados armazenada em bancos de dados de empresas, órgãos governamentais, universidades e em diversas organizações é cada vez maior e, com isso, um novo conjunto de necessidades para manipulação de dados tem surgido, tornando-se necessário o desenvolvimento de métodos e tecnologias específicos para atender essas necessidades.

As novas tecnologias computacionais envolvidas no processo de extração de conhecimento, ou *Data Mining*, em grandes quantidades de dados estão cada vez mais buscando características em algoritmos de Aprendizado de Máquina.

Os algoritmos de Aprendizado de Máquina (AM) podem ser caracterizados em algoritmos de aprendizado supervisionado e algoritmos de aprendizado não supervisionado. No primeiro caso, padrões são encontrados através da exploração de exemplos (dados) já conhecidos e devidamente rotulados com o valor de um atributo especial denominado classe, produzindo assim um modelo preditivo e, em alguns casos, um modelo descritivo. Já no aprendizado não supervisionado, os exemplos não estão explicitamente rotulados com uma classe, padrões de dados são encontrados a partir de alguma caracterização de regularidade, agrupando-os em clusters (Decker & Focardi, 1995). Neste caso o modelo produzido é somente descritivo.

Em Aprendizado de Máquina, o problema de criar uma classificação a partir de exemplos não rotulados, geralmente, é conhecido como processo de *clustering*. *Clustering* é usualmente visto como um processo que agrupa objetos (exemplos) físicos ou abstratos em clusters (grupos), de acordo com uma medida de similaridade previamente estabelecida, nas quais exemplos pertencentes a um mesmo cluster são mais semelhantes entre si, que exemplos pertencentes a clusters distintos.

O AutoClass (Cheeseman & Stutz, 1990) é uma ferramenta de *clustering* baseada na técnica probabilística que usa o modelo de Bayes, desenvolvido pelo grupo de Bayes no *Ames Research Center*. AutoClass é um software robusto de domínio público bastante utilizado e citado na literatura. A teoria probabilística que suporta o AutoClass encontra-se bem documentada em diversas publicações (Hanson et al., 1991; Cheeseman et al., 1988). Entretanto, a descrição dos parâmetros e manual de uso para executar o AutoClass deixa a desejar. Essa descrição encontra-se espalhada em diversos arquivos texto que acompanham o software e isso dificulta o entendimento da correta execução e do uso dos diversos parâmetros que podem ser utilizados para executar o AutoClass, o que motivou o desenvolvimento deste trabalho. Assim este trabalho tem como objetivo descrever de uma forma simplificada e organizada os arquivos e parâmetros mais importantes, ou mais utilizados, na execução do AutoClass, porém sem entrar em detalhes da teoria probabilística que suporta o mesmo.

Este trabalho está organizado da seguinte forma: na Seção 2 são descritas resumidamente as diferentes técnicas utilizadas no processo de *clustering*, na Seção 3 é apresentada uma visão geral de execução do AutoClass considerando seu modo de busca, de geração de relatórios e predição de novos exemplos. Finalmente, na Seção 4 são apresentadas algumas conclusões.

2 Técnicas de *Clustering*

As diferentes técnicas de *clustering* são classificadas de acordo com os resultados obtidos pelos algoritmos. A Figura 1 ilustra a classificação dessas técnicas, as quais são descritas a seguir (Witten & Frank, 2000; Chiara, 2001; Martins, 2001).

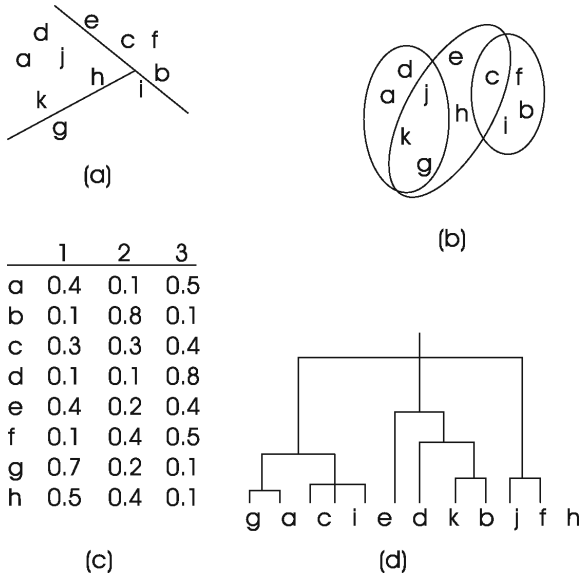

Figura 1: Técnicas de *Clustering*: (a) Otimização (b) *Clumping* (c) Probabilística (d) Hierárquica

Técnicas de otimização: visam formar uma k-partição ótima sobre os exemplos, isto é, divide o conjunto em k clusters mutuamente exclusivos, em que k é dado pelo usuário/especialista. As técnicas de otimização são computacionalmente caras, pois fazem uma busca exaustiva pela k-partição ótima. Por isso, seu uso é restrito a pequenas quantidades de exemplos e/ou pequenos valores de k.

Técnicas de grupo (*clumping*): retornam clusters nos quais seus elementos (exemplos) podem se sobrepor. O problema com essas técnicas é que um mesmo exemplo pode pertencer a vários clusters.

Técnicas probabilísticas: associam, para cada exemplo, a probabilidade dele estar em cada cluster.

Técnicas hierárquicas: criam árvores comumente chamadas de dendrogramas. As folhas da árvore representam exemplos individuais e os nós internos representam grupos (clusters) de exemplos. Esses métodos podem ser subdivididos em aglomerativos e divisivos. Os métodos aglomerativos constroem a árvore de baixo para cima (das folhas para a raiz), enquanto os divisivos constroem a árvore de cima para baixo (da raiz para as folhas). As técnicas hierárquicas são computacionalmente mais baratas que as técnicas de otimização.

Deve ser observado que esta não é a única maneira de classificar métodos de *clustering*. Outras possíveis classificações podem ser encontrados em (Michalski & Stepp, 1987; Jain et al., 1999).

3 AutoClass - Uma Visão Geral de Execução

Basicamente, AutoClass descreve clusters a partir de uma distribuição probabilística sobre os atributos dos exemplos, considerando que existe independência condicional nos dados. Antes de apenas particionar os exemplos, como a maioria das técnicas de *clustering* fazem, a abordagem Bayesiana busca em um espaço do modelo a "melhor"descrição dos clusters. Uma boa classificação busca o ponto entre precisão preditiva *versus* complexidade dos clusters sem *overfit*[1] dos dados. Tais clusters são também *fuzzy*, ou seja, cada caso (exemplo) tem uma probabilidade de pertencer a cada um dos diferentes clusters, e não apenas a um cluster — técnica probabilística. AutoClass tem várias características tais como: determinar o número de clusters automaticamente; usar exemplos com valores discretos e reais em um mesmo conjunto de dados; manipular valores ausentes/desconhecidos; utilizar tempo de processamento que é, aproximadamente, linear à quantidade de dados; entre outras.

O AutoClass possui três modos de execução, descritos a seguir e ilustrado nas Figuras 2 e 3:

1. Busca (`search`): consiste em agrupar os dados colocando os resultados em três arquivos com extensões `.search`, `.results[-bin]` e `.log`, alguns dos quais de difícil interpretação. O arquivo `.search` contém registros das tentativas da busca, enquanto que o arquivo `.results[-bin]` terá as "melhores"classificações encontradas por AutoClass. Já o arquivo `.log` contém a informação do que foi impresso na tela durante a execução.

2. Relatórios (`reports`): dois arquivos de saída do modo de busca (`.search` e `.results [-bin]`) são necessários para a execução do AutoClass no modo de relatórios. Neste modo, por sua vez, são criados mais quatro arquivos `.class-text-n`, `.case-text-n`, `.influ-o-text-n` e `.rlog`. Os arquivos com extensão `.class-text-n` e `.case-text-n`, mostram como foram agrupados os dados, enquanto que nos arquivos `.influ-o-text-n` e `.rlog` encontram-se detalhes da execução.

3. Predição (`predict`): este modo consiste na predição de possíveis clusters para um novo conjunto de dados (dados de teste). São utilizados os dois arquivos, `.search` e `.results [-bin]` criados no modo de busca (`search`). que possuem informações de como os exemplos podem ser agrupados — Figura 3.

Como pode ser observado, o AutoClass trabalha com diversos arquivos, tanto de entrada como de saída. Neste trabalho os arquivos serão identificados pela sua extensão. Por exemplo, o arquivo com resultados das tentativas de busca serão referenciados como `<nome_de_arquivo>.search`, ou mais resumidamente, como `.search`. O arquivo `.results` pode ser gerado como um arquivo binário, o que agiliza o acesso, sendo referenciado como `.results[-bin]`. A seguir são descritos com mais detalhes cada um dos três modos de execução do AutoClass.

3.1 Modo Busca

O modo de busca (`search`) de AutoClass é o modo de execução mais "pesado", podendo demorar algumas horas dependendo do tipo e tamanho do conjunto de exemplos de treinamento. Todos os cálculos necessários para agrupar os exemplos em clusters são feitos nessa fase.

[1] Superajuste.

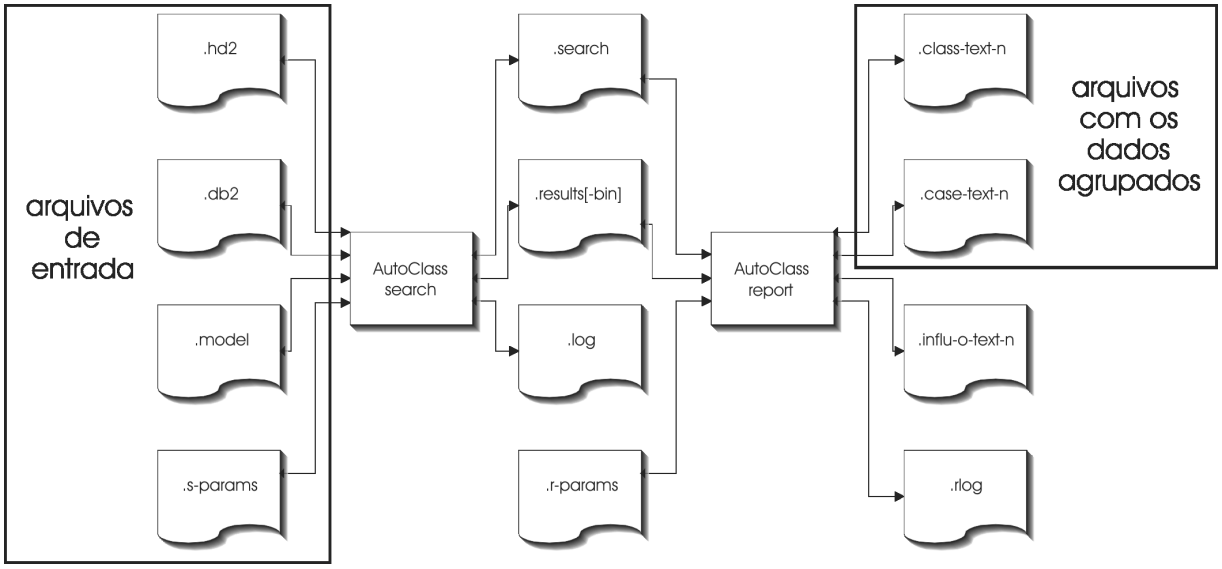

Figura 2: Esquema para Busca (`search`) e Relatório (`report`)

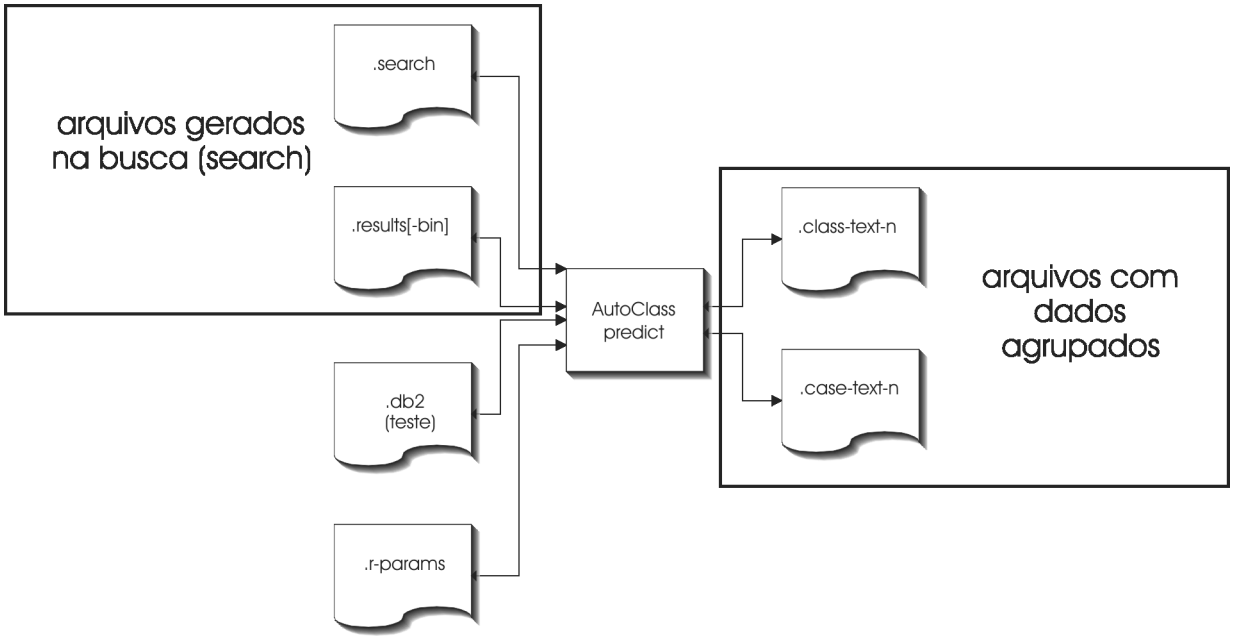

Figura 3: Esquema de Predição (`predict`)

A busca (`search`) de AutoClass é executada invocando a seguinte linha de comando:

```
autoclass -search <arquivo1>.db2[-bin] <arquivo2>.hd2
          <arquivo3>.model <arquivo4>.s-params
```

no qual `<arquivo1>..<arquivo4>` geralmente possuem o mesmo nome sendo diferenciados apenas pelas extensões `.db2`, `.hd2`, `.model`, `.s-params` respectivamente, e referem-se aos quatro arquivos de entrada, mostrados esquematicamente na Figura 2 e descritos na Tabela 1.

Segue uma discussão detalhada do conteúdo de cada um desses arquivos.

Tipo de Arquivo	Extensão	Descrição
Arquivo de cabeçalho	`.hd2`	Especifica o formato do arquivo de dados e os tipos dos atributos
Arquivo de dados	`.db2`	Contém os dados
Arquivo de modelos	`.model`	Define o modelo probabilístico correspondente a cada atributo
Arquivo de parâmetro de busca	`.s-params`	Especifica os parâmetros de busca

Tabela 1: Descrição dos Arquivos de Entrada `(search)`

3.1.1 Arquivo de Cabeçalho

O arquivo de cabeçalho (`.hd2`) contém a especificação dos dados de entrada (conjunto de treinamento) para AutoClass. Essa especificação consiste de duas partes:

1. especificação da definição;
2. definição do tipo de atributos.

A especificação da definição contém os seguintes parâmetros obrigatórios e parâmetros opcionais:

- `num_db2_format_defs` (obrigatório)
  Número de linhas da especificação da definição. O AutoClass não usa um separador para dividir as duas partes (especificações da definição e a definição do tipo dos atributos). Portanto, para separar essas duas partes do arquivo, deve-se contar o número de linhas da especificação da definição considerando apenas os parâmetros válidos, isto é, excluindo as linhas de comentário.

- `number_of_attributes` (obrigatório)
  Número de atributos. É necessária a declaração explícita da quantidade de atributos presentes em cada exemplo (registro).

- `separator_char` (opcional)
  Caracter separador entre atributos.
  O valor *default* é ' ' (espaço).

- `comment_char` (opcional)
  Caracter que define comentário em uma linha. Este caracter, quando utilizado, comenta uma linha inteira, ou seja, não se pode usar metade da linha para uma outra função e o restante para comentário.
  O valor *default* é ';' (ponto e vírgula).

- `unknown_token` (opcional)
  Caracter que representa valor desconhecido.
  O valor *default* é '?' (interrogação).

A segunda parte deste arquivo é utilizada para definição do tipo dos atributos. Os atributos são descritos utilizando a seguinte sintaxe:

1	2		3	4
`<zero-based att#>`	`<att_type>`	`<att_sub_type>`	`<att_description>`	`<att_param_pairs>`

no qual:

1. `<zero-based att#>` é o índice, a partir do zero, dos atributos presentes em cada exemplo, começando na linha 1;
2. `<att_type>` `<att_sub_type>` são descrições relacionadas diretamente com os tipos de atributo (real e discreto). As combinações destes dois atributos são mostradas na Tabela 2;
3. `<att_description>` é um identificador (sem espaços em brancos) ou uma *string* menor que 40 caracteres que descreve o atributo. Este valor não é computado, portanto, pode ser qualquer coisa que tenha significado para o usuário;
4. `<att_param_pairs>` é uma propriedade correspondente a cada tipo de atributo, como mostrado na Tabela 2. Por exemplo, `range` 10, significa que o atributo contém 10 valores discretos diferentes.

`<att_type>`	`<att_sub_type>`	`<att_param_pairs>`	descrição dos tipos
`dummy`	`none/nil`	–	Qualquer tipo de dado, porém, serão armazenados como 0 (zero) no banco de dados interno.
`discrete`	`nominal`	`range`	Podem ser strings, símbolos ou inteiros. No parâmetro `range` deverá ser especificado o número de valores discretos distintos. Campos com valores desconhecidos ('?') também devem ser contados.
`real`	`location`	`error`	Números reais positivos e negativos. Por exemplo, saldo de sua conta bancária.
`real`	`scalar`	`zero_point` `rel_error`	Números reais positivos. Por exemplo, altura de um objeto.

Tabela 2: Tabela de Tipos de Atributos

O valor `zero_point` contém o menor valor possível do atributo. Por exemplo, para um atributo que representa a altura de um objeto que varia de 1.5 a 2, `zero_point` terá o valor 1.5.

O AutoClass usa os erros, `error` e `rel_error`, como o limite mais baixo na largura da distribuição normal[2]. Assim, quanto maior o valor do erro maior será a sobreposição dos clusters e, quanto menor o valor do erro menor será a sobreposição. Portanto, esses valores precisam de um cuidado especial por parte do usuário para obter assim o agrupamento desejado.

A Figura 4 ilustra um arquivo de cabeçalho no formato *default* do AutoClass, considerando que o conjunto de dados contém 3 atributos, sendo um discreto e dois contínuos.

[2]O modelo da distribuição normal é também chamado de distribuição *Gaussiana*.

```
num_db2_format_defs 1
number_of_attributes 3

;;<zero-based att#> <att_type> <att_sub_type> <att_description> <att_param_pairs>
0 discrete nominal "sexo   (m/f)"      range 2
1 real     scalar  "altura (1.5 a 2)"  zero_point 1.5 rel_error .01
2 real     scalar  "peso   (47 a 120)" zero_point 47 rel_error .01
```

Figura 4: Arquivo de Cabeçalho (`.hd2`)

3.1.2 Arquivo de Dados

O arquivo de dados contém um conjunto de dados, semelhante a saída de um *select* em `SQL`, seguindo a especificação definida no arquivo de cabeçalho. Os caracteres de separador de atributos, valores desconhecidos e comentário são por *default*, respectivamente: ' ' , '?' e ';' e cada exemplo é separado por nova linha (`'\n'`).

Por exemplo, considere um arquivo de dados que contenha 12 exemplos e 3 atributos, mostrado na Figura 5, utilizando o arquivo de cabeçalho definido na Figura 4.

```
h 1.7  60
m 1.62 56
h 1.8  80
h 1.60 50
h 1.9  100
h 2    ?
h 1.75 80
h 1.78 90
m 1.6  47
m 1.5  ?
m 1.8  55
h 1.61 60
```

Figura 5: Arquivo de Dados (`.db2`)

Deve ser observado que os arquivos de dados desenvolvidos em ambiente DOS possuem os *carriage returns* (`^M`) no final da linha, os quais devem ser removidos caso seja utilizado o ambiente UNIX. Neste caso, pode ser utilizado o programa *dos2unix*[3] para fazer a conversão.

Como já mencionado, o AutoClass suporta atributos com valores discretos, como o atributo sexo; reais, como o atributo altura e valores desconhecidos. Portanto, é preciso especificar para cada atributo o modelo a ser usado no arquivo de modelos (`.model`).

[3] http://rpmfind.net/linux/RPM/ASP/blackcat-linux/6.2/i386/RedHat/RPMS/dos2unix-2.2-4bc.i386.html

3.1.3 Arquivo de Modelos

AutoClass é um software probabilístico, dessa forma, é necessário especificar no arquivo de modelos (`.model`) um modelo matemático para cada atributo de acordo com os valores do atributo.

No arquivo de modelos do AutoClass podem ser utilizados quatro tipos de modelos matemáticos, apresentados na Tabela 3. É possível, também, ignorar um ou mais atributos, utilizando o parâmetro `ignore` na definição do tipo de atributo no arquivo de modelos.

Tipos de Modelos	Modelo	Descrição
`single_multinomial`	multinomial	Valores discretos e valores desconhecidos ('?')
`single_normal_cn`	normal	Valores reais
`single_normal_cm`	normal	Valores reais e valores desconhecidos ('?')
`multi_normal_cn`	normal covariante	Valores reais
`ignore`	-	Ignora o atributo

Tabela 3: Descrição dos Modelos

Na primeira linha válida desse arquivo deve aparecer a expressão `model_index n m`, na qual `n` é o índice do atributo, a partir do zero, e `m` é o número de linhas usadas nas definições dos modelos.

Por exemplo, considere um arquivo `.model` como o especificado na Figura 6. Neste arquivo são considerados três atributos com índices 0, 1, 2. O atributo com índice 0 será ignorado, o atributo com índice 1 contém valores reais com distribuição normal e sem valores desconhecidos. O atributo com índice 2 contém valores reais, porém aceita valores desconhecidos ou ausentes.

```
1  model_index 0 3
2  ignore 0
3  single_normal_cn 1
4  single_normal_cm 2
```

Figura 6: Arquivo de Modelos (`.model`)

Entretanto, não é necessário ter uma linha para cada índice, ou seja, pode-se especificar na mesma linha um modelo que aceita vários índices de atributos ou, ainda, usar a palavra `default`, que considera todos os atributos que não foram ainda especificados, como mostrado na Figura 7. Neste arquivo são considerados 9 atributos com índices de 0 a 8. Os atributos com índices 0 e 1 serão ignorados (`ignore`), os atributos com índices 2 e 3 e contém valores reais com distribuição normal e sem valores desconhecidos (`single_normal_cn`), os atributos com índices 7 e 8 contém valores reais porém aceita valores desconhecidos ou ausentes (`single_normal_cm`) e os atributos restantes (4, 5, 6) contém valores reais (`multi_normal_cn`).

```
model_index 0 4
ignore 0 1
single_normal_cn 2 3
single_normal_cm 7 8
multi_normal_cn default
```

Figura 7: Arquivo de Modelos (`.model`) `Exemplo Extra`

3.1.4 Arquivo de Parâmetros de Busca

Existem várias maneiras de agrupar um mesmo conjunto de dados. Essas variações são determinadas pelos parâmetros definidos no arquivo de parâmetros de busca (`.s-param`), por exemplo, duração da busca, número de clusters a serem encontrados, entre outros.

Alguns dos vários parâmetros que podem ser definidos neste arquivo são:

- `fixed_j`
  Se 0 (zero), o AutoClass define o número de clusters.
  Se > 0, o número de clusters é o valor definido pelo usuário.
  O valor *default* é 0.

- `n_save`
  Especifica o número de tentativas de agrupamentos que serão gravados nos arquivos `.search` e `.results[-bin]`.
  O valor *default* é 2.

- `log_file_p`
  Se `false`, não cria o arquivo `.log`.
  O valor *default* é `true`.

- `search_file_p`
  Se `false`, não cria o arquivo `.search`.
  O valor *default* é `true`.

- `results_file_p`
  Se `false`, não cria o arquivo `.results[-bin]`.
  O valor *default* é `true`.

- `min_save_period`
  Para proteção contra possíveis imprevistos, o AutoClass gravará os cálculos periodicamente.
  O valor *default* é 1800 (segundos).

- `max_n_store`
  Especifica o número máximo de classificações armazenadas internamente.
  O valor *default* é 10.

- `n_final_summary`
  Especifica o número de tentativas que serão impressas no final da busca.
  O valor *default* é 10.

- `n_data`
  Se 0 (zero), lê todos os exemplos definidos no arquivo de dados (`.db2`).
  Se > 0, lê somente o número de exemplos definidos aqui (do primeiro ao n-ésimo).
  O valor *default* é 0.

- `save_compact_p`
  Se `true`, grava a busca em arquivos binários (`.results-bin e .chkpt-bin`).
  Se `false`, grava em arquivos texto (`.results e .chkpt`).
  O valor *default* é `true`.

- `randomize_random_p`
  Se `false`, usa o valor 1 para inicializar a função que gera números aleatórios, fornecendo assim sempre os mesmos números na repetição de testes.
  Se `true`, usa o relógio do computador para inicializar a função.
  O valor *default* é `true`.

- `force_new_search`
  Se `true`, ignora as buscas anteriores, ou seja, sobrescreve os arquivos `.search e .results[-bin]`.
  O valor *default* é `false`.

- `checkpoint_p`
  Se `true`, faz *checkpoints* da busca corrente a cada `min_checkpoint_period` segundos.
  O valor *default* é `false`.

- `min_checkpoint_period`
  Se `checkpoint_p = true`, faz *checkpoints* da busca corrente.
  O valor *default* é 10800 (segundos).

- `reconverg_type`
  Este parâmetro pode ser "chkpt"ou "results".
  Se "chkpt", continua a convergência da busca a partir do arquivo `.chkpt[-bin]` , mas isso ocorre somente se `checkpoint_p` for `true`.
  Se "results", continua a convergência da busca a partir do arquivo `.results[-bin]`, mas isso ocorre somente se `checkpoint_p` for `false`.
  O valor *default* é " ".

- `screen_output_p`
  Se `false`, não existe nenhuma saída direta para a tela do computador.
  O valor *default* é `true`.

- `interactive_p`
  Se `false`, executa indefinidamente.
  Se `true`, interrompe a execução após pressionar <q>.
  O valor *default* é `true`.

- `max_duration`
  Se 0 (zero), executa sem restrição de tempo.
  Se > 0, restringe o tempo de busca executando até o tempo (em segundos) definido.
  O valor *default* é 0.

- `max_n_tries`
  Se 0 (zero), executa sem restrição de tentativas.
  Se > 0, restringe o número de tentativas de execução ao número especificado.
  O valor *default* é 0.

Vale ressaltar que, dentre os parâmetros de parada, `interactive_p, max_duration` e `max_n_tries`, pelo menos um deles deve estar ativo. Caso contrário, o AutoClass executará indefinidamente. Para executar utilizando todos os parâmetros com o valor *default*, cria-se o arquivo de parâmetros de busca com uma linha contendo apenas um parâmetro de parada.

3.2 Modo Relatórios

No modo de busca, o AutoClass gera três arquivos de saídas: `.search`, `.results[-bin]` e `.rlog` — Figura 2, os quais contém informações sobre como os exemplos devem ser agrupados. Porém, essas informações estão em um formato de difícil interpretação para o usuário.

Utilizando dois arquivos de saída de busca, os arquivos `.search` e o `results[-bin]`, o AutoClass gera simultaneamente três tipos de relatórios invocando a seguinte linha de comando:

```
autoclass -reports <arquivo1>.results[-bin] <arquivo2>.search <arquivo3>.r-params
```

Como pode ser observado, é necessário especificar um novo arquivo, o arquivo de parâmetros de relatórios (`.r-params`). Neste arquivo alguns parâmetros podem ser definidos, ou podem ser utilizados os parâmetros *default* do AutoClass, neste caso, basta usar apenas um arquivo texto em branco.

Os três relatórios gerados são:

1. referência cruzada por cluster;
2. referência cruzada por exemplo;
3. valor de influência dos atributos.

Esses três relatórios são descritos a seguir.

3.2.1 Relatório de Referência Cruzada por Cluster

Este relatório mostra os clusters (`Class`) separadamente, listando os exemplos e a probabilidade do exemplos pertencerem a cada cluster. Por exemplo, considere um conjunto de dados com 12 exemplos. Um possível agrupamento dos dados poderia ser como apresentado na Figura 8[4]. Neste agrupamento, oito exemplos (1, 2, 3, 5, 6, 8, 11, 12) pertencem ao cluster 0 (`CLASS = 0`) e quatro (4, 6, 9, 10) pertencem ao cluster (`CLASS = 1`). Ainda, como pode ser observado, o exemplo 1, 2, 4, 6, 11 e 12 não tem 100% de probabilidade de pertencer a apenas um cluster. Neste caso, o exemplo 1 pertence ao cluster 0 com probabilidade 0.996 e tem probabilidade 0.004 de pertencer ao cluster 1.

[4] O cabeçalho do relatório foi omitido nesta ilustração, a qual mostra os dados do relatório a partir da linha 16 até a linha 45 do correspondente arquivo de relatório.

```


                              CLASS = 0



     Case #      Prob         (Class  Prob) (Class  Prob) (Class  Prob) (Class  Prob)
-------------------------------------------------------------------------------------

          1     0.996            1   0.004
          2     0.966            1   0.034
          3     0.999
          5     0.999
          7     0.999
          8     1.000
         11     0.986            1   0.014
         12     0.982            1   0.018


                              CLASS = 1



     Case #      Prob         (Class  Prob) (Class  Prob) (Class  Prob) (Class  Prob)
-------------------------------------------------------------------------------------

          4     0.799            0   0.201
          6     0.603            0   0.397
          9     1.000
         10     1.000
```

Figura 8: Relatório (`.class-text-n`)

3.2.2 Relatório de Referência Cruzada por Exemplo

Neste tipo de relatório, ilustrado na Figura 9, os exemplos listados são ordenados pelo índice (`Case #`), com seus respectivos clusters (`Class`) e probabilidades (`Prob`). Como um exemplo pode pertencer a mais de um cluster, neste relatório o exemplo será apresentado no cluster ao qual pertence com maior probabilidade.

```


     Case #  Class  Prob         Case #  Class  Prob         Case #  Class  Prob
-------------------------------------------------------------------------------------
          1    0   0.996             5    0   0.999             9    1   1.000
          2    0   0.966             6    1   0.603            10    1   1.000
          3    0   0.999             7    0   0.999            11    0   0.986
          4    1   0.799             8    0   0.999            12    0   0.982
```

Figura 9: Relatório (`.case-text-n`)

3.2.3 Relatório de Influência dos Atributos

O relatório de influência dos atributos tenta fornecer as medidas relativas considerando a "influência" que os atributos tiveram nos clusters encontrados. O relatório mostra os atributos ordenados pela sua influência na determinação dos clusters utilizando uma medida heurística da influência de cada atributo, do conjunto de treinamento, na construção desses clusters.

3.3 Modo Predição

O AutoClass pode ser usado como um software de classificação, ou seja, pode predizer a qual cluster os novos exemplos pertencem. Para isso é necessário trabalhar com dois conjuntos de dados (`.db2`):

- o conjunto de dados de treinamento, usado na execução da busca (`search`);
- o conjunto de dados de teste, usado na predição.

A predição é feita invocando a seguinte linha de comando:

```
autoclass -predict <arquivo1>.db2 <arquivo2>.results[-bin]
        <arquivo3>.search <arquivo4>.r-params
```

Durante a execução desse modo são gerados dois arquivos (`.class-text-n` e `.case-text-n`), — Figura 3 na página 4, semelhantes aos arquivos gerados no modo de relatórios. O nome destes arquivos de saída são baseados no arquivo do conjunto de dados (`.db2`). Por exemplo, se o arquivo de dados tem o nome `teste.db2` os arquivos gerados serão `teste.class-text-1` e `teste.case-text-1`.

Como ilustração, considere o conjunto de dados de treinamento mostrado anteriormente na Figura 5 na página 7. AutoClass gerou o relatório apresentado na Figura 8 na página oposta. Com os arquivos de saída `.search` e `.results[bin]` gerados no modo de busca e um novo conjunto de dados (teste), mostrado na Figura 10, AutoClass gera o relatório apresentado na Figura 11.

```
1  h 1.7  60
2  m 1.62 ?
3  h 1.8  80
4  h 1.60 50
5  h 1.6  56
6  h 2    100
7  h 1.75 80
8  h 1.78 90
9  m 1.6  47
10 m 1.5  50
11 m 1.8  ?
12 h 1.61 60
```

Figura 10: Conjunto de Dados Teste

```
18
19
20
21                                CLASS = 0
22
23
24
25      Case #      Prob         (Class  Prob) (Class  Prob) (Class  Prob) (Class  Prob)
26 ----------------------------------------------------------------------------------------
27
28           1     0.996            1   0.004
29           3     0.999
30           5     0.930            1   0.070
31           6     0.999            1   0.001
32           7     0.999
33           8     1.000
34          11     0.658            1   0.342
35          12     0.982            1   0.018
36
37
38                                CLASS = 1
39
40
41
42      Case #      Prob         (Class  Prob) (Class  Prob) (Class  Prob) (Class  Prob)
43 ----------------------------------------------------------------------------------------
44
45           2     0.656            0   0.344
46           4     0.799            0   0.201
47           9     1.000
48          10     1.000
```

Figura 11: Relatório `.class-text-n (Teste)`

Este relatório mostra que os exemplos (1, 3, 5, 6, 7, 8, 11 e 12) do arquivo de teste Figura 10 são classificados como pertencentes ao cluster 0 (`CLASS = 0`), enquanto que os exemplos (2, 4, 9 e 10) como pertencentes ao cluster 1 (`CLASS = 1`). Como pode ser observado, alguns desses exemplos tem probabilidade de pertencer a outro cluster.

4 Conclusão

Em um processo cujo objetivo é o agrupamento de dados em clusters utilizando medidas de similaridade, existem várias técnicas consolidadas, entre elas a probabilística utilizada pelo sistema AutoClass. O AutoClass é um software robusto e bastante referenciado pela comunidade científica. Os arquivos gerados pelo AutoClass contém várias informações estatísticas relacionadas aos atributos e aos clusters encontrados. Entretanto, a documentação fornecida descrevendo os arquivos de entrada/saída bem como os diversos parâmetros que podem ser utilizados na execução de AutoClass deixam a desejar.

Assim, o objetivo deste trabalho consiste na descrição, de maneira simples e resumida, dos arquivos de entrada e saída, bem como dos principais parâmetros utilizados na execução do sistema AutoClass. Alguns exemplos são apresentados com o intuito de facilitar o entendimento e tornar a tarefa de execução mais simples para o usuário.

Atualmente estamos utilizando AutoClass conjuntamente com algoritmos de aprendizado supervisionado simbólico para tentar explicar, conceitualmente, os clusters por ele encontrados. Já foram realizados diversos experimentos utilizando bases de dados reais e os resultados obtidos mostram que é interessante usar essa combinação de aprendizado supervisionado e não supervisionado a fim de procurar por uma explicação conceitual dos clusters encontrados por (AutoClass) (Martins et al., 2002).

Referências

Cheeseman, P. & Stutz, J. (1990). Bayesian classification (Autoclass): Theory and results advances in knowledge discovery and data mining. http://ic.arc.nasa.gov/ic/projects/bayes-group/autoclass-c-program.html.

Cheeseman, P., Stutz, J., Self, M., Kelly, J., Taylor, W., & Freeman, D. (1988). Bayesian classification. In *Proceedings of the Seventh National Conference of Artificial Intelligence (AAAI-88)*. St. Paul, MN.

Chiara, R. (2001). Aplicação de data mining em logs de servidores web. Exame de Qualificação de Mestrado, ICMC-USP.

Decker, K. M. & Focardi, S. (1995). Technology overview: A report on data mining. Technical report, CSCS-ETH, Swiss Scientific Computing Center.

Hanson, R., Stutz, J., & Cheeseman, P. (1991). Bayesian classification theory. Technical report, NASA Ames Research Center. http://ic.arc.nasa.gov/ic/projects/bayes-group/images/tr-fia-90-12-7-01.ps.

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323.

Martins, C. A. (2001). Interpretação de clusters em aprendizado de máquina. Exame de Qualificação de Doutorado, ICMC-USP.

Martins, C. A., Monard, M. C., & Halembeck, G. C. (2002). Combining clustering and inductive learning to find and interpret patterns from datasets. In *International Conference on Computer Science, Software Engineering, Information Technology, e-Business, and Applications (CSITeA02)*, Foz do Iguaçú, Brazil.

Michalski, R. S. & Stepp, R. E. (1987). Clustering. *Encyclopedia of Artificial Intelligence*. John Wiley & Sons.

Witten, I. H. & Frank, E. (2000). *Data Mining, Practical Machine Learning Tools and Techniques with Java Implementations*, volume 1. Morgan Kaufmann. http://www.cs.waikato.ac.nz/ml/weka/index.html.