Universidade Federal de Minas Gerais
Departamento de Ciência da Computação

João Paulo Domingos Silva

Algoritmos de Classificação baseados em Análise Formal de Conceitos

Belo Horizonte, Minas Gerais, Brasil
Fevereiro de 2007

João Paulo Domingos Silva

Algoritmos de Classificação baseados em Análise Formal de Conceitos

Dissertação apresentada ao Programa de Pós-Graduação em Ciência Da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Newton José Vieira

Belo Horizonte, Minas Gerais, Brasil

Fevereiro de 2007

João Paulo Domingos Silva

Algoritmos de Classificação baseados em Análise Formal de Conceitos

---

Prof. Dr. Newton José Vieira
Departamento de Ciência da Computação - UFMG

Prof. Dr. José Lopes de Siqueira Neto
Departamento de Ciência da Computação - UFMG

Prof. Dr. Luis Enrique Zárate
Departamento de Ciência da Computação - PUC Minas

Belo Horizonte, Minas Gerais, Brasil
Fevereiro de 2007

Em livros de história, seremos a memória dos dias que virão...
Humberto Gessinger

Agradecimentos

Obrigado ao Newton, meu orientador, por confiar em mim e por me receber sempre com enorme presteza. Obrigado ao Zárate, com quem conheci muitos dos assuntos aqui abordados. Obrigado ao José, por ensinar-me bastante da carreira de docente. E obrigado ao Renato, colega de pesquisas e de viagens.

Dedicatória

Aos meus pais – Reginaldo e Anette –, ao Luiz Felipe e à Grazielle.
São pessoas que ajudam-me a vencer os momentos mais apertados.
São pessoas com quem aprendo mais do que em mil leituras, com suas constantes lições de amor.

Resumo

O ser humano sempre procura mais conhecimento. Esse é essencial em âmbitos pessoal e também profissional. Para conseguir mais conhecimento, o ser humano armazena volumosos repositórios de dados, dos quais tem que extrair informação. Contudo, a quantidade de dados e a complexidade desses podem prejudicar a extração de informação adequada. Para solucionar tal problema, recorre-se à chamada mineração de dados. Essa fornece um conjunto de métodos para processamento dos dados e extração de informação. Dentre tais métodos, está a chamada classificação, de especial interesse para este trabalho.

Existem diversas propostas de algoritmos de classificação. Algumas são clássicas (bastante conhecidas), como a chamada árvore de decisão. Outras, assunto principal deste trabalho, são propostas alternativas, baseadas por exemplo em análise formal de conceitos. Essas últimas propostas executam a tarefa de classificação por meio de estruturas chamadas reticulados de conceitos; e conseguem bons resultados. Este trabalho apresenta diferentes algoritmos para classificação baseados em análise formal de conceitos, sendo alguns desses propostas inéditas, publicados aqui pela primeira vez. Tais algoritmos têm boas precisões de classificação, mas apresentam alguns desafios, como os custos computacionais.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

Análise formal de conceitos e classificação são os assuntos abordados e combinados por este trabalho. O primeiro desses assuntos – análise formal de conceitos – é originário da matemática e fornece a interessante estrutura chamada reticulado de conceitos (baseada em ordenação parcial), enquanto o segundo – classificação – é oriundo da ciência da computação, em particular, da mineração de dados. Eles são aqui unidos para a tarefa de descoberta de conhecimento.

O problema que este trabalho aborda é: Como empregar análise formal de conceitos para classificação? Pesquisas em ciência da computação já usam os reticulados de conceitos em tarefas de mineração de dados [Pri05, Stu03, SWW98, Wil01], mas a maioria preocupa-se com regras de associação (um dos métodos de mineração de dados). A classificação é também assunto de alguns trabalhos aqui estudados [Oos88, LN90, Sah95, CR04], mas ainda existe espaço para novas propostas.

São muitas as motivações para abordar este problema. Primeiro, o ser humano precisa de informação. Conforme será mencionado nos próximos capítulos, em muitas situações, o ser humano encontra-se em cenários com excesso de dados (armazenados em volumosos repositórios), porém carentes de informação. É necessário, portanto, conseguir informação desses dados e, depois, conhecimento baseado na informação. É também motivação o potencial da análise formal de conceitos em tarefas da ciência da computação. Tal análise fornece, como já mencionado, os reticulados de conceitos, interessantes estruturas solidamente fundamentadas em matemática discreta, que podem contribuir com métodos já clássicos da mineração de dados. Por fim, embora muitos estudos em ciência da computação já considerem a análise formal de conceitos, esse assunto é conhecido por poucos e timidamente explorado; são raríssimas, por exemplo, as publicações sobre tal assunto em língua portuguesa.

Este trabalho tem dois objetivos principais: (*i*) estudar e aqui descrever algoritmos para solução do problema proposto – propostas que usem análise formal de conceitos para classificação; (*ii*) propor novo algoritmo para classificação com reticulado de conceitos.

Os dois objetivos enumerados foram cumpridos. Estudaram-se e codificaram-se (em programas Java) quatro algoritmos de classificação com reticulados. Alguns desses, devido às datas de suas propostas, não usam o reticulado de conceitos, mas um pseudo-reticulado (tais assuntos são detalhados nos capítulos a seguir). Foi também proposto e implementado um algoritmo inédito para a mesma tarefa, que conseguiu excelentes resultados. Escreveram-se programas também para algoritmos clássicos de classificação – árvore de decisão e conjunto de regras –, permitindo a comparação entre esses diferentes paradigmas de propostas.

As propostas para classificação aqui abordadas (exceto os métodos clássicos) são os seguintes: "Grand", "Rulearner", "Legal", "Galois", "Similares1", "Similares2". As duas primeiras propostas usam o pseudo-reticulado, enquanto as demais empregam o reticulado de conceitos. As três primeiras propostas produzem regras de classificação, enquanto as três últimas classificam usando o próprio reticulado de conceitos. São, portanto, seis algoritmos similares em alguns pontos, mas que, usando reticulados de conceitos (ou pseudo-reticulados) executam a classificação de diferentes maneiras. Importante destacar que os dois últimos algoritmos são propostas inéditas (na verdade, duas versões de uma nova proposta).

O trabalho foi organizado do seguinte modo: os capítulos 2 e 3 apresentam os fundamentos para este trabalho – o primeiro desses capítulos aborda classificação (são também descritos alguns métodos clássicos), enquanto o segundo aborda análise formal de conceitos; os seis algoritmos para classificação com reticulados são tema do capítulo 4; os diferentes experimentos com todos os algoritmos são apresentados no capítulo 5; por fim, o capítulo 6 encerra o trabalho, concluindo e debatendo muitas das informações apresentadas, em especial, comparando os algoritmos.

Como será mostrado, os algoritmos para classificação com reticulados têm vantagens e desvantagens, quando comparados à clássica mineração de dados. Eles conseguem resultados muitas vezes superiores aos métodos clássicos, contudo, tais bons resultados vêm acompanhados de demorados tempos de execução (essa é a principal desvantagem de algoritmos que trabalham com análise formal de conceitos). Questões como essa são debatidas durante este trabalho, que lança mão de diversos exemplos, a fim de auxiliar no entendimento dos assuntos aqui abordados.

Capítulo 2

Classificação

Este capítulo apresenta, sem muitas minúcias (pois seu conteúdo já é bastante explorado na literatura [TSK06, WF05]), a mineração de dados, em especial, o método de classificação. Organizou-se o capítulo do seguinte modo: a seção 2.1 introduz mineração de dados e classificação; a seção 2.2 aborda conjuntos de dados, usados como entradas para a classificação; árvores de decisão são resumidas na seção 2.3; e conjuntos de regras na seção 2.4.

Primando pela clareza do capítulo e do trabalho, convencionam-se aqui três importantes e recorrentes termos [DP97]:

Dado é apenas um símbolo, armazenado em repositório (como códigos armazenados em computador);

Informação é o dado já interpretado em um contexto (por exemplo: temperatura, em Celsius ou Fahrenheit, dependendo do receptor da informação);

Conhecimento é a informação interpretada em um contexto.

2.1 Introdução

O ser humano, embora possua hoje imensos repositórios de dados, encontra-se muitas vezes desprovido de informação e, conseqüentemente, também sem conhecimento adequado para tomar decisões, que poderiam ser baseadas naquela informação. O armazenamento desses grandes volumes de dados aconteceu durante as últimas décadas, devido aos constantes avanços na informática, como, por exemplo, melhoria da tecnologia de bancos de dados e diminuição dos custos de computadores e

relacionados. Contudo, como os repositórios passaram a ter tamanhos tão imensos e dados complexos, o ser humano não mais conseguiu, com os métodos que até então empregava, extrair informação de suas próprias coleções [HK06].

Dado um cenário onde se tem excesso de dados e carência de informação, tornou-se necessária, portanto, a elaboração de métodos de análise, que recebam como entrada tais dados e produzam informação, a fim de permitir o aumento de conhecimento. Essa análise não pode ser puramente humana, senão, devido a motivos já mencionados, esse seria um processo muito lento, caro e subjetivo, ou seja, extremamente ineficiente. Portanto, como os computadores permitiram o armazenamento de tamanhas quantidades de dados, eles que agora executem processos automáticos ou semi-automáticos, para análise de dados e produção de informação.

O chamado KDD (*knowledge discovery in databases*[1]) é um conhecido processo para extração de informação em bancos de dados [FPSS96]. A informação obtida durante tal processo precisa ser correta, inédita, útil e compreensível. O KDD contém diversos passos que primeiro processam os dados de entrada e depois tratam a informação obtida, a fim de permitir aumento de conhecimento. Entre esses dois conjuntos de passos, existe o passo mais importante neste trabalho, aquele no qual é produzida informação, o passo chamado mineração de dados. Os passos anteriores à mineração chamam-se pré-processamento (eles preparam a entrada), enquanto os posteriores chamam-se pós-processamento (visualização, por exemplo).

Mineração de dados é, portanto, um importante passo do processo KDD, que recebe como entrada um conjunto lapidado de dados e que produz o tipo de informação desejado naquele momento. Esse passo contém uma variedade de métodos que são executados de acordo com o interesse de quem executa o KDD, podendo produzir, portanto, informação em diferentes formatos, como regras de classificação, regras de associação, agrupamentos etc.

O método de mineração de dados abordado neste trabalho chama-se classificação. Para apresentar tal método, é preciso apresentar primeiro o formato de entrada recebido por ele, pois essa contém um aspecto importante. O método de classificação, assim como os demais, recebe um conjunto composto por objetos e atributos (tais termos são abordados em minúcias na próxima seção), sendo cada objeto formado por valores de atributos. O que diferencia a classificação dos demais métodos é a presença de um atributo especial. Doravante, esse atributo será chamado de classe, enquanto os demais serão chamados apenas de atributos.

[1] Do inglês: "descoberta de conhecimento em bancos de dados".

Classificação é um método que aprende uma função (essa, no sentido matemático) com valores de atributos em seu domínio e valores de classe em sua imagem. Em outras palavras, dado um conjunto de valores de atributos, tal função determina um valor de classe para esse conjunto. Essa função é também conhecida como modelo de classificação e será assim denominada doravante.

Recorre-se ao método de classificação com um dos seguintes propósitos: (*i*) descrever um conjunto de objetos, mostrando de que maneira os valores de atributos determinam o valor de classe e o que diferencia objetos de classes diferentes; (*ii*) determinar um valor de classe para objetos que tenham esse atributo sem conteúdo. Este trabalho aborda o segundo propósito apresentado – determinar valores de classe. São comuns as situações em que uma equipe de pessoas, munidos de uma coleção de objetos, tem que decidir algo que possa ser modelado como uma classe (este trabalho mostra, nas próximas páginas, diversos exemplos de conjuntos assim).

O método de classificação, para determinar valores de classe, tem uma primeira fase, na qual é aprendido o modelo de classificação. Para tal, usa-se um primeiro conjunto de objetos cujos valores de classe são conhecidos. Um algoritmo então produz, baseado em tal conjunto, um modelo, que pode ter, por exemplo, o formato de conjunto de regras ou árvore de decisão (tais modelos são apresentados nas próximas seções).

A segunda fase do método de classificação é a operação do modelo de classificação produzido. Para tal, usa-se um segundo conjunto de objetos, com os mesmos atributos do primeiro conjunto (os objetos têm os mesmos atributos, mas podem ter valores diferentes de atributos), mas com valores de classe desconhecidos. O modelo então, seja no formato de conjunto de regras ou árvore de decisão, classifica esses objetos, em função de seus valores de atributos.

Depois de aprendido, é interessante que o modelo seja avaliado. Metodologias de teste e avaliação são mostrados em capítulo futuro, mas adianta-se aqui uma conhecida medida de desempenho, chamada precisão. Dados um modelo pronto e um conjunto de objetos sem valores de classe, esses objetos podem ser classificados corretamente ou erroneamente por aquele modelo. Calcula-se então a precisão do modelo de classificação com a seguinte fórmula:

$$\text{precisão} = \frac{\text{número de acertos}}{\text{número de objetos classificados}}$$

2.2 Entradas para classificação

O conjunto de entrada para métodos de mineração de dados, embora os repositórios possam ser bastante complexos na prática, é representado como uma simples tabela. Suas linhas representam objetos, enquanto suas colunas representam atributos; no caso de classificação, uma dessas colunas denota a classe. As células da tabela contêm valores assumidos por objetos, para cada um dos atributos [2].

O objeto, composto por valores de atributos e classe, representa qualquer entidade existente no universo modelado por aquele conjunto de dados. Tais entidades geralmente são bastante intuitivas para o ser humano, como perfis de clientes, por exemplo, em um cenário empresarial. O conjunto de objetos será aqui denominado O; dele se originam $O' \subseteq O$ e $O'' \subseteq O$, subconjuntos usados nas fases de classificação – aprendizagem e operação.

O atributo, como também a classe, é qualquer propriedade cujos valores compõem os objetos. Convenciona-se aqui que A é o conjunto de atributos da coleção de dados, incluindo a classe; essa é denotada por $c \in A$, enquanto $B \subset A$ denota os demais atributos. Formalizando: $A = B \cup \{c\}$. Cada atributo, incluindo a classe, possui seu domínio (conjunto de valores); cada objeto pode ter, para cada atributo e para a classe, no máximo um conteúdo, pertencente ao respectivo domínio. Convenciona-se que A' contém a união dos domínios de atributos e classe. Sendo C' domínio da classe e B' a união dos domínios dos atributos, tem-se $A' = B' \cup C'$.

O modelo de classificação é, em outras palavras, uma função que seleciona um elemento de C', baseado em elementos de B', possuídos por um objeto. Importante destacar que deve ser possível determinar a que atributos referem-se os elementos de A'; no caso de valores homônimos de atributos, esses devem ser diferenciados ("atributo1.conteúdo" e "atributo2.conteúdo", por exemplo). Quando C' contém somente dois elementos, tem-se a chamada classificação binária.

Apesar da simplicidade para se formalizar conjuntos de dados, esses geralmente possuem problemas na prática. Exemplos comuns de problemas em conjuntos de dados são: anomalias (objetos muito diferentes dos demais ou com valores atípicos para alguns atributos); objetos que, por quaisquer motivos, tenham atributos em branco; inconsistências (objetos com valores errados para atributos, demandando,

[2] É importante observar que objetos e atributos, assim como outros conceitos neste trabalho, podem ser chamados de outras maneiras na literatura. Entretanto, primando pela coerência neste texto, convencionam-se aqui tais termos, pois eles são empregados em outros assuntos aqui abordados.

por exemplo, consulta a outras fontes para correção do problema) etc.

Como o conjunto pode conter problemas, é necessário que ele seja pré-processado, a fim de se fornecer uma entrada com mais qualidade ao método de mineração de dados. Alguns exemplos de pré-processamento são: seleção de objetos (seja unindo ou removendo alguns dos mesmos); seleção de atributos; correção de valores de atributos (tentando solucionar problemas como valores incorretos ou ausentes) etc. Por esse ser um extenso assunto, não será detalhado aqui.

É adequado e relevante apresentar aqui um exemplo de conjunto de dados, abordado até então principalmente com formalismos matemáticos. A Tabela 2.1 [WF05] representa um conjunto cujos dados descrevem diferentes situações de pacientes em exames oftalmológicos. O leitor observe a presença de um atributo especial, na última coluna, cujos valores são lentes de contato recomendadas, dependendo da situação oftalmológica do paciente.

As linhas da tabela são os objetos; o conjunto O contém, portanto, 24 elementos. Já as colunas são os atributos, exceto a primeira (índices de objetos, para auxiliar a referenciar os mesmos) e a última (essa é a classe). O conjunto B' contém todos os valores que aparecem nas células dos quatro atributos, enquanto o conjunto C' contém três possíveis valores de classe – podem-se recomendar lentes macias, rígidas ou nenhuma lente.

Esse conjunto é idealizado (não acontece na prática), pois além de não apresentar problemas, contém todas as combinações de valores dos quatro atributos. Os exemplos que usam tal tabela, durante este trabalho, lançam mão de apenas alguns objetos para concepção do modelo (primeira fase de classificação). Caso usassem todos, não existiriam objetos inéditos para classificar (segunda fase de classificação). Seleciona-se $O' = \{01, 08, 11, 14, 20, 22\}$. Em tal subconjunto, encontram-se todos os valores de atributos e de classe, em quantidades equilibradas; procurou-se, portanto, evitar que algum valor fosse muito recorrente, enquanto outro fosse raro.

Tabela 2.1: Conjunto de dados sobre lentes de contato.

	Idade	Prescrição	Astigmatismo	Lacrimejamento	Lentes
01	Juventude	Miopia	Não	Reduzido	Nenhuma
02	Juventude	Miopia	Não	Normal	Macias
03	Juventude	Miopia	Sim	Reduzido	Nenhuma
04	Juventude	Miopia	Sim	Normal	Rígidas
05	Juventude	Hipermetropia	Não	Reduzido	Nenhuma
06	Juventude	Hipermetropia	Não	Normal	Macias
07	Juventude	Hipermetropia	Sim	Reduzido	Nenhuma
08	Juventude	Hipermetropia	Sim	Normal	Rígidas
09	Pré-presbiopia	Miopia	Não	Reduzido	Nenhuma
10	Pré-presbiopia	Miopia	Não	Normal	Macias
11	Pré-presbiopia	Miopia	Sim	Reduzido	Nenhuma
12	Pré-presbiopia	Miopia	Sim	Normal	Rígidas
13	Pré-presbiopia	Hipermetropia	Não	Reduzido	Nenhuma
14	Pré-presbiopia	Hipermetropia	Não	Normal	Macias
15	Pré-presbiopia	Hipermetropia	Sim	Reduzido	Nenhuma
16	Pré-presbiopia	Hipermetropia	Sim	Normal	Nenhuma
17	Presbiopia	Miopia	Não	Reduzido	Nenhuma
18	Presbiopia	Miopia	Não	Normal	Nenhuma
19	Presbiopia	Miopia	Sim	Reduzido	Nenhuma
20	Presbiopia	Miopia	Sim	Normal	Rígidas
21	Presbiopia	Hipermetropia	Não	Reduzido	Nenhuma
22	Presbiopia	Hipermetropia	Não	Normal	Macias
23	Presbiopia	Hipermetropia	Sim	Reduzido	Nenhuma
24	Presbiopia	Hipermetropia	Sim	Normal	Nenhuma

2.3 Árvore de decisão

Árvore de decisão é um modelo abordado em diversos trabalhos [BFOS84, Qui79, Qui93, Kas80] e sua compreensão é bastante intuitiva. Durante a primeira fase de classificação, constrói-se uma árvore baseada em um conjunto de dados cujos valores de classe são conhecidos. Cada nó interno representa um atributo, enquanto as arestas que deixam aquele nó representam todos os valores de seu domínio (ou intervalos de valores, dependendo do atributo). Os nós terminais (também conhecidos como folhas) contêm valores de classe.

O algoritmo para construção da árvore de decisão, cujo pseudo-código é mostrado no Algoritmo 1, é recursivo. Para cada nó gerado, é checado se a recursão pode ser interrompida naquele ramo; em caso positivo, aquele nó torna-se terminal, cujo rótulo é algum valor de classe. Caso a árvore precise continuar a ser expandida, seleciona-se um atributo para aquele nó, geram-se tantas arestas quanto forem seus valores de domínio e continua-se a recursão.

Existem, no algoritmo apresentado, algumas questões que ganharão aqui mais detalhes, como, por exemplo: a condição de parada; a geração de arestas com valores de atributo; a seleção do atributo para o nó interno. Existem ainda outras questões merecedoras de maior debate, como o problema da árvore de decisão replicada (quando formam-se sub-árvores iguais), o que demanda uma poda na estrutura. Contudo, problemas como esse último excedem o escopo do trabalho.

Uma primeira questão quanto ao algoritmo é: Quando interromper a expansão da árvore de decisão? A primeira alternativa é cessar a recursão quando todos os objetos que chegam até aquele nó possuírem o mesmo valor de classe ou os mesmos valores de atributo. É possível também interromper a expansão da árvore quando existir um valor predominante de classe, ou seja, quando uma considerável quantidade (superior a determinado limiar) de objetos que chegam até aquele nó possuir o mesmo valor de classe. Qualquer que seja a alternativa adotada, o nó em questão torna-se terminal e é depois rotulado com o valor predominante de classe.

Uma segunda questão: Como particionar o domínio de um atributo, a fim de gerar arestas para um determinado nó? Essa questão parece simples, mas é preciso lembrar que existem atributos cujos domínios podem ser numéricos ou extremamente grandes, inviabilizando, portanto, existir uma aresta para cada valor de atributo. No caso de atributo numérico (ou mesmo categórico, porém de domínio grande), é adequado particionar seu domínio, agrupando seus valores, para que esses grupos

formem um número reduzido de arestas. Tais grupos podem, no caso de atributo numérico, ser entendidos como intervalos de valores de seu domínio.

Por fim, aborda-se aqui a questão: Como selecionar um atributo para rotular determinado nó e prosseguir a recursão? O primeiro critério é não selecionar novamente atributos que já apareçam em nós ascendentes. É interessante também selecionar um atributo cujas arestas de seus valores de domínio conduzam a filhos chamados puros, ou seja, que não apresentem grande mistura de valores de classe. A medida de impureza de nós aqui abordada chama-se informação; ela é baseada no cálculo de entropia (as fórmulas mostradas nos próximos parágrafos são baseadas em exemplos publicados na literatura [TSK06]).

Algoritmo 1 Pseudo-código para construção de árvore de decisão.

entradas: O', B, B', C' (conforme anteriormente definidos).
resultados: nó raiz de árvore de decisão.

```
se nó em questão é terminal então
  rotular nó com classe dominante
senão
  selecionar melhor atributo
  rotular nó com atributo
  determinar domínio do atributo
  para todo conteúdo no domínio execute
    adicionar descendente ao nó
    rotular aresta com conteúdo
    recomeçar algoritmo, analisando descendente
  fim para
fim se
retorne nó em questão
```

O leitor suponha que o algoritmo encontra-se no seguinte instante: foi constatado que a recursão deve continuar, sendo necessário, portanto, selecionar um atributo para o nó em questão. Sabe-se, em tal instante, que o nó considerado apresenta alguma impureza, ou seja, os objetos que o alcançam têm alguns valores iguais de atributos (aqueles já checados em nós ascendentes), mas não têm somente um valor de classe (senão, a recursão seria cessada).

A medida de impureza chamada informação (o leitor observe que essa palavra é aqui empregada em sentido diferente daquele definido no começo do capítulo) é

aplicada a qualquer nó da árvore e retorna quanta informação falta para se chegar a um nó terminal. Tal medida baseia-se na distribuição de valores de classe dos objetos que alcançam aquele nó; quanto mais uniforme for essa distribuição, mais alto o resultado da medida de impureza (nó mais impuro). Eis a fórmula para cálculo da impureza de um nó, em uma classificação binária ($c1$ e $c2$ são os números de objetos naquele nó, com cada um dos valores de classe):

$$\text{info}([c1, c2]) \;=\; -\; c1/(c1+c2) \times \log_2 c1/(c1+c2) \;-\; c2/(c1+c2) \times \log_2 c2/(c1+c2)$$

O algoritmo deve selecionar um atributo que não apareça em nós ascendentes ao nó considerado. Caso o algoritmo esteja apenas começando a árvore, tratando a raiz, todos os atributos são candidatos a rótulo daquele nó. Para cada atributo candidato a rótulo do nó em questão, testam-se que nós filhos ele produziria (por meio das arestas referentes a seu domínio) e calcula-se a impureza média desses filhos. Para um nó com dois filhos, calcula-se a impureza média desses com a seguinte fórmula ($o1$, $o2$, $o3$ são, respectivamente, os números de objetos no nó pai e em seus dois filhos, enquanto $c1a$, $c2a$, $c1b$, $c2b$ são as distribuições das duas classes – 1 e 2 – nos dois filhos – a e b):

$$\text{info}([c1a, c2a], [c1b, c2b]) \;=\; (o2/o1) \times \text{info}([c1a, c2a]) \;+\; (o3/o1) \times \text{info}([c1b, c2b])$$

Calculadas a impureza do nó em questão e a impureza média dos prováveis nós filhos, o algoritmo seleciona o atributo que, por meio das arestas de seu domínio, gere nós filhos menos impuros. Quando os objetos caminharem do nó pai para os filhos, seguindo as arestas adequadas, ocorrerá diminuição da impureza, ou seja, ocorrerá ganho de informação, pois os nós terminais estarão mais próximos. A seleção do atributo procura, portanto, maximizar o ganho de informação. Empregando os mesmos símbolos das fórmulas anteriores, eis como calcular o ganho de informação de determinado nó, com dois filhos e em uma classificação binária:

$$\text{gain}(.) \;=\; \text{info}([c1, c2]) \;-\; \text{info}([c1a, c2a], [c1b, c2b])$$

Exemplo 1: Mostram-se aqui os primeiros passos para a construção da árvore de decisão referente à Tabela 2.1. Como já mencionado, será empregado o subconjunto de objetos $O' = \{01, 08, 11, 14, 20, 22\}$, para essa construção.

O algoritmo começa checando se a árvore terá nós além da raiz. A resposta é afirmativa, pois os objetos têm distintos valores de classe. Calcula-se a impureza do nó raiz e observa-se que essa é máxima, pois os valores de classe encontram-se igualmente distribuídos entre os objetos – dois recomendam lentes macias, dois recomendam lentes rígidas e dois não recomendam quaisquer lentes. Eis o cálculo de impureza do nó raiz:

$$\begin{aligned} \text{info}([2,2,2]) &= -\,2/6 \times \log_3 2/6 \;-\; 2/6 \times \log_3 2/6 \;-\; 2/6 \times \log_3 2/6 \\ &= 1/3 \;+\; 1/3 \;+\; 1/3 \\ &= 1 \end{aligned}$$

Como o algoritmo está tratando o nó raiz, todos os atributos são candidatos a rótulo desse nó. Calcula-se, para cada atributo candidato, a impureza média de seus nós filhos. É necessário observar que tais cálculos são ponderados pelo número de objetos que chegam a esses nós. Eis as impurezas médias dos nós filhos, para cada atributo candidato (os números de valores de classes, entre colchetes, referem-se respectivamente a "nenhuma", "rígidas", "macias"):

Idade:

$$\begin{aligned} \text{info}([1,1,0],[1,0,1],[0,1,1]) &= (2/6) \times \text{info}([1,1,0]) \;+\; (2/6) \times \text{info}([1,0,1]) \\ &\quad + (2/6) \times \text{info}([0,1,1]) \\ &= (2/6) \times 0.63 \;+\; (2/6) \times 0.63 \;+\; (2/6) \times 0.63 \\ &= 0.63 \end{aligned}$$

Prescrição:

$$\begin{aligned} \text{info}([2,1,0],[0,1,2]) &= (3/6) \times \text{info}([2,1,0]) \;+\; (3/6) \times \text{info}([0,1,2]) \\ &= (3/6) \times 0.58 \;+\; (3/6) \times 0.58 \\ &= 0.58 \end{aligned}$$

Astigmatismo:

$$\begin{aligned} \text{info}([1,0,2],[1,2,0]) &= (3/6) \times \text{info}([1,0,2]) \;+\; (3/6) \times \text{info}([1,2,0]) \\ &= (3/6) \times 0.58 \;+\; (3/6) \times 0.58 \\ &= 0.58 \end{aligned}$$

Lacrimejamento:

$$\begin{aligned} \text{info}([2,0,0],[0,2,2]) &= (2/6) \times \text{info}([2,0,0]) \;+\; (4/6) \times \text{info}([0,2,2]) \\ &= (2/6) \times 0 \;+\; (4/6) \times 0.63 \\ &= 0.42 \end{aligned}$$

O quarto atributo, se selecionado como rótulo do nó raiz, produziria nós filhos mais puros, comparando-se aos filhos dos demais candidatos. Os cálculos mostram

que tal atributo produziria um descendente onde todos os objetos que nele chegassem teriam os mesmos valores de classe – dois objetos em "nenhuma" –, além de outro descendente, esse com alguma impureza. Para decidir essa questão, calcula-se então o ganho de informação, para cada atributo candidato:

$$\begin{aligned}
\text{gain(Idade)} &= \text{info}([2,2,2]) - \text{info}([1,1,0],[1,0,1],[0,1,1]) \\
&= 1 - 0.63 = 0.37 \\
\\
\text{gain(Prescrição)} &= \text{info}([2,2,2]) - \text{info}([2,1,0],[0,1,2]) \\
&= 1 - 0.58 = 0.42 \\
\\
\text{gain(Astigmatismo)} &= \text{info}([2,2,2]) - \text{info}([1,0,2],[1,2,0]) \\
&= 1 - 0.58 = 0.42 \\
\\
\text{gain(Lacrimejamento)} &= \text{info}([2,2,2]) - \text{info}([2,0,0],[0,2,2]) \\
&= 1 - 0.42 = 0.58
\end{aligned}$$

O quarto atributo proporciona maior ganho de informação, logo, "lacrimejamento" é escolhido como rótulo do nó raiz. Geram-se duas arestas, com os valores de tal atributo – "reduzido" e "normal". O algoritmo recomeça, recursivamente, em cada um dos nós filhos. Será constatado, embora o exemplo termine por aqui, que o nó "reduzido" será terminal e rotulado como "nenhuma", enquanto o nó "normal" continuará expandindo uma subárvore.

Ao fim do algoritmo, será produzida uma árvore de decisão, como aquela mostrada na Figura 2.1. Será então o término da primeira fase de classificação, com um modelo pronto para classificar objetos inéditos.

■

Durante a segunda fase de classificação, objetos caminham pela árvore e nela encontram valores para a classe, que até então eram desconhecidos. Cada um desses objetos começa seu percurso no nó raiz e, a cada nó interno visitado, confere o atributo ali encontrado como rótulo. O objeto então continua seu caminho, escolhendo a aresta cujo rótulo corresponda ao seu valor naquele atributo. Quando o objeto alcança um nó terminal, é classificado com seu rótulo.

Exemplo 2: Dadas a Tabela 2.1 e a árvore de decisão da Figura 2.1, escolhe-se naquela qualquer objeto, para nessa ser classificado. Embora conheçam-se os

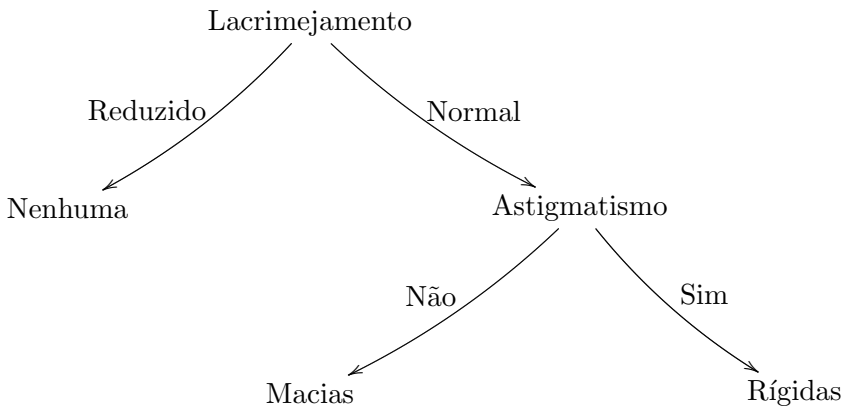

Figura 2.1: Árvore de decisão sobre lentes de contato.

valores de classe de todos aqueles objetos, eles são aqui ignorados; eles serão, na verdade, úteis na checagem de sucesso ou fracasso da árvore de decisão. Como qualquer objeto pode servir de exemplo, escolhe-se o segundo:

Idade	=	Juventude
Prescrição	=	Miopia
Astigmatismo	=	Não
Lacrimejamento	=	Normal

O algoritmo primeiro confere, no nó raiz, o atributo "lacrimejamento"; como o objeto tem valor "normal" em tal atributo, ele caminha pela aresta de mesmo rótulo. O algoritmo então confere o atributo "astigmatismo", no próximo nó interno; o objeto caminha pela aresta "não", pois é esse seu valor em tal atributo. O objeto alcança um nó terminal e tem sua classe preenchida com "macias". De fato, "macias" é o valor correto de classe; logo, sucesso! ■

2.4 Conjunto de regras

O modelo de classificação baseado em regras é também bastante simples e conhecido [Coh95, CB91, CN89, MMHL86]. Em tal modelo, produz-se, na primeira fase, um conjunto de regras no formato "se... então...". Esse formato contém o chamado antecedente ("se...") e o chamado conseqüente ("então..."). O antecedente é um conjunto de valores de atributos (é necessário distinguir a que atributos pertencem tais valores) unidos por conjunções, enquanto o conseqüente é, no caso de classificação, um valor de classe. Lêem-se as regras, portanto, do seguinte modo: "se o objeto tem os valores de atributos presentes no antecedente, então seu valor de classe é aquele do conseqüente".

A relação entre objetos e regras é denominada cobertura. Dados um objeto e uma regra, se aquele possui todos os valores de atributo presentes no antecedente dessa, afirma-se que a regra cobre o objeto. É importante destacar que, para existir tal cobertura, o antecedente da regra deve ser subconjunto dos valores de atributos do objeto, mas esse pode conter também outros valores.

Um conjunto de regras pode ser considerado, quanto à cobertura, mutuamente exclusivo ou exaustivo. No primeiro caso, qualquer objeto deve ser coberto por, no máximo, uma das regras. Já no segundo caso, qualquer objeto deve ser coberto por, no mínimo, uma das regras. É desejável que o conjunto de regras produzido na primeira fase de classificação seja mutuamente exclusivo e também exaustivo, mas, como tais propriedades nem sempre são atendidas, é interessante evitar possíveis problemas decorrentes disso. Em conjuntos de regras não exaustivos, insere-se uma regra com antecedente vazio e conseqüente padrão; desse modo, no mínimo tal regra cobrirá qualquer objeto. Para conjuntos de regras não mutuamente exclusivos, ordenam-se as regras e requer-se a leitura na ordem definida; desse modo, no máximo uma regra cobrirá qualquer exemplo. O conjunto ordenado de regras e com uma dessas padrão é mutuamente exclusivo e exaustivo.

O Algoritmo 2 mostra um simples pseudo-código para construção do conjunto de regras. Esse não é ordenado tampouco tem regra padrão. O algoritmo procura, a cada valor de classe, produzir regras que cubram o máximo possível de objetos. Ao fim da execução, todos os objetos são cobertos pelo conjunto de regras produzido. O leitor considere um valor de classe e os objetos que tenham tal valor (esses chamam-se objetos positivos). Como o algoritmo seleciona valores de atributos para formar um antecedente, de modo a cobrir o máximo dos objetos positivos? Para isso, procuram-se valores de atributos que proporcionem maior cobertura, que maximizem

a proporção:

$$\frac{\text{objetos positivos cobertos}}{\text{objetos cobertos}}$$

Algoritmo 2 Pseudo-código para construção de conjunto de regras.

---

entradas: O', B, B', C' (conforme aqui definidos).
resultados: conjunto de regras.
  para todo valor de classe **execute**
    enquanto existirem objetos positivos não cobertos **execute**
      produzir regra para valor de classe com antecedente vazio
      enquanto regra não cobrir todos os objetos positivos **ou**
      não existirem mais valores de atributos **execute**
        escolher valor de atributo que cubra máximo de objetos positivos
        adicionar valor de atributo ao antecedente
      fim enquanto
    fim enquanto
  fim para

Exemplo 3: Mostram-se aqui os primeiros passos para construção do conjunto de regras referente à Tabela 2.1. Como já mencionado, será empregado o subconjunto de objetos $O' = \{01, 08, 11, 14, 20, 22\}$, para essa construção.

O processo começa arbitrariamente com as regras de conseqüente "Nenhuma". Para formar o antecedente, analisam-se valores de atributos e suas coberturas, conforme cálculo já mencionado. As frações abaixo têm: em seus numeradores, número de objetos positivos cobertos por tais valores de atributos, e em seus denominadores, número de objetos cobertos por tais valores de atributos.

Juventude	1/2
Pré-presbiopia	1/2
Presbiopia	0/2
Miopia	2/3
Hipermetropia	0/3
Não	1/3
Sim	1/3
Reduzido	2/2
Normal	0/4

O antecedente recebe "Reduzido". Com esse valor de atributo no antecedente, cobrem-se todos os objetos com valor de classe "Nenhuma" e não é coberto nenhum com outro valor. Produz-se, portanto a regra:

$$\text{Reduzido} \Rightarrow \text{Nenhuma}$$

Como foram cobertos todos os objetos positivos e nenhum negativo, o processo agora considera o conseqüente "Macias". Analisam-se os valores de atributos para o antecedente:

Juventude	0/2
Pré-presbiopia	1/2
Presbiopia	1/2
Miopia	0/3
Hipermetropia	2/3
Não	2/3
Sim	0/3
Reduzido	0/2
Normal	2/4

Os valores de atributos "Hipermetropia" e "Não" permitiram as maiores proporções. Escolhe-se, arbitrariamente, o primeiro desses valores (caso os numeradores fossem diferentes, o maior seria preferível). Como, com tal antecedente, são também cobertos objetos negativos, é preciso procurar mais valores de atributos. Continua-se, portanto, a procura, considerando-se, nas frações abaixo, apenas os objetos com valor "Hipermetropia".

Juventude	0/1
Pré-presbiopia	1/1
Presbiopia	1/1
Não	2/2
Sim	0/1
Reduzido	0/0
Normal	2/3

Adiciona-se "Não" ao antecedente. Com tal antecedente, cobrem-se todos os objetos com valor "Macias" e nenhum negativo. Produz-se, portanto a regra:

$$\text{Hipermetropia} \wedge \text{Não} \Rightarrow \text{Macias}$$

O processo com o conseqüente "Rígidas" é muito similar ao último descrito, portanto, não será aqui narrado. Ao fim da primeira fase de classificação, tem-se o conjunto de regras mostrado na Figura 2.2.

$$\begin{array}{ll} \text{Reduzido} & \Rightarrow \text{Nenhuma} \\ \text{Hipermetropia} \wedge \text{Não} & \Rightarrow \text{Macias} \\ \text{Sim} \wedge \text{Normal} & \Rightarrow \text{Rígidas} \end{array}$$

Figura 2.2: Conjunto de regras sobre lentes de contato.

■

Na segunda fase de classificação, objetos têm seus valores de classe, até então desconhecidos, determinados pelo conjunto de regras. Cada objeto é classificado segundo os conseqüentes das regras que o cobrem. Para evitar conflitos entre conseqüentes diferentes, é desejável, como já mencionado, que o conjunto de regras seja mutuamente exclusivo e exaustivo. As regras aqui produzidas têm ambas as propriedades.

Exemplo 4: Dados a Tabela 2.1 e o conjunto de regras da Figura 2.2, escolhe-se naquela qualquer objeto, para nesse ser classificado. Escolhe-se, arbitrariamente, o quarto objeto:

$$\begin{array}{lcl} \text{Idade} & = & \text{Juventude} \\ \text{Prescrição} & = & \text{Miopia} \\ \text{Astigmatismo} & = & \text{Sim} \\ \text{Lacrimejamento} & = & \text{Normal} \end{array}$$

O objeto é coberto pela última regra e, portanto, corretamente classificado como "Rígidas". ■

Capítulo 3

Análise formal de conceitos

Análise formal de conceitos é um ramo da matemática aplicada, particularmente da denominada teoria de reticulados [DP90]. Esses podem ser sucintamente definidos como conjuntos parcialmente ordenados, com propriedades descritas durante este capítulo. Um interessante modo de começar o estudo da análise formal de conceitos é segmentando seu próprio nome: "análise" sugere observação e manipulação de dados fornecidos; "conceitos" determina unidades segundo as quais os dados são estruturados, correspondentes à interpretação humana da realidade (adiante serão apresentados os conceitos formais); "formal" sugere fundamentação matemática referente à análise e aos conceitos. Este capítulo apresenta análise formal de conceitos, baseado em trabalho de seus criadores [GW96].

Este capítulo, em diversos momentos, requer do leitor conhecimento de matemática discreta, em especial, noções de ordenação parcial. Exemplos são usados de modo a auxiliar no entendimento de formalismos matemáticos. O capítulo primeiro aborda os chamados contextos formais (seção 3.1), depois conceitos formais (seção 3.2) e, por fim, apresenta reticulados de conceitos, além de mencionar pseudo-reticulados (seção 3.3).

3.1 Contextos formais

Contexto formal mono-valorado é definição primordial para o estudo da análise formal de conceitos. Tal contexto tem a notação (O, A, R), sendo O e A conjuntos contendo elementos respectivamente denominados objetos e atributos e sendo $R \subseteq O \times A$ uma relação binária chamada incidência. Todo elemento da incidência tem a notação oRa, uma relação entre objeto e atributo, lida como "o objeto o tem o atributo a". Para quaisquer subconjuntos $E \subseteq O$ e $I \subseteq A$, podem ser determinados seus conjuntos derivados (que respectivamente são os atributos comuns aos objetos e os objetos que possuem os atributos), por meio das operações de derivação:

$$\begin{aligned} E^{\uparrow} &= \{a \in A \mid \forall o \in E \; oRa\} \\ I^{\downarrow} &= \{o \in O \mid \forall a \in I \; oRa\} \end{aligned}$$

A chamada tabela cruzada é uma maneira adequada de representar contextos formais mono-valorados. Em tal tabela, as linhas são objetos e as colunas são atributos. A incidência é representada nas células da tabela, por algum símbolo que indique existir relação entre objeto e atributo.

Contextos formais mono-valorados representam bem conjuntos cujos objetos apenas têm ou não atributos, ou seja, cuja relação entre esses dois elementos é "sim ou não". Contudo, os atributos de conjuntos de dados usados na prática costumam possuir diversos valores. É necessário, portanto, definir contexto formal multi-valorado. Esse tem a notação (O, A, A', R). O, A e A' são conjuntos de objetos, atributos e valores de atributos, respectivamente. $R \subseteq O \times A \times A'$ é, como mostra a notação, uma relação ternária entre esses conjuntos. Todo elemento $(o, a, b) \in R$ é lido como "o atributo a tem o valor b para o objeto o".

A conversão de contexto formal multi-valorado para mono-valorado é necessária, para adequá-lo ao estudo da análise formal de conceitos. Essa conversão acontece por meio da chamada escala conceitual. Existem diferentes escalas, mas a que pertence ao escopo deste trabalho é a chamada nominal. Em tal escala, os valores de atributos do contexto formal mono-valorado tornam-se, no multi-valorado, atributos. Existe, nesse último contexto, relação entre objeto e atributo quando essa também existir, no primeiro contexto, entre objeto e valor de atributo. Importante destacar que atributos numéricos no contexto formal multi-valorado precisam ser primeiro organizados em intervalos, para depois passarem pela escala nominal. Doravante, quando este trabalho mencionar contextos formais multi-valorados, será presumido,

mas não explicitado, que foi executada a escala nominal, sempre que necessária[1].

Exemplo 5: O conjunto da Tabela 2.1 possui objetos, atributos e valores para esses. Pode ser considerado, portanto, um contexto formal multi-valorado (O, A, A', R), com os seguintes conjuntos:

O = $\{01, \ldots 24\}$
A = {Idade,Prescrição,Astigmatismo,Lacrimejamento,Lentes}
A' = {Juventude,Pré-presbiopia,Presbiopia,Miopia,Hipermetropia,Não, Sim,Reduzido,Normal,Nenhuma,Macias,Rígidas}

Tal conjunto pode ser convertido para um contexto formal mono-valorado, por meio de uma escala nominal. Os valores de atributos do contexto de origem tornam-se, portanto, atributos no contexto de destino. A Tabela 3.1 mostra o contexto formal mono-valorado (O, A', R). Suas linhas são objetos, suas colunas são atributos e existe relação de incidência em células marcadas com $\times$. Como não existem, neste exemplo, atributos numéricos no contexto de origem, a conversão ocorre diretamente. É necessário mencionar que, caso existissem valores de atributos homônimos, esses deveriam ser diferenciados no contexto de destino, produzindo nesse atributos distintos.

Doravante, será suprimido o uso das palavras mono-valorado e multi-valorado, restando apenas contexto formal. Este exemplo será empregado durante todo o trabalho, tratado como (O, A', R) ou (O, A, A', R). Quando for usada essa última notação (pois, em muitos exemplos, é preciso distinguir atributos e seus valores), não será mencionada a escala nominal, mas, sempre que necessário, entende-se que essa foi aplicada. ■

[1] O leitor observe que os elementos do conjunto A' podem ser chamados de atributos ou de valores de atributos, dependendo do tipo de contexto formal em questão – atributos para mono-valorado, valores de atributos em caso contrário.

Tabela 3.1: Contexto formal sobre lentes de contato.

	Juventude	Pré-presbiopia	Presbiopia	Miopia	Hipermetropia	Não	Sim	Reduzido	Normal	Nenhuma	Macias	Rígidas
01	×			×		×		×		×		
02	×			×		×			×		×	
03	×			×			×	×		×		
04	×			×			×		×			×
05	×				×	×		×		×		
06	×				×	×			×		×	
07	×				×		×	×		×		
08	×				×		×		×			×
09		×		×		×		×		×		
10		×		×		×			×		×	
11		×		×			×	×		×		
12		×		×			×		×			×
13		×			×	×		×		×		
14		×			×	×			×		×	
15		×			×		×	×		×		
16		×			×		×		×	×		
17			×	×		×		×		×		
18			×	×		×			×	×		
19			×	×			×	×		×		
20			×	×			×		×			×
21			×		×	×		×		×		
22			×		×	×			×		×	
23			×		×		×	×		×		
24			×		×		×		×	×		

3.2 Conceitos formais

Um contexto formal (O, A', R) determina conceitos formais, cada um com notação (E, I), sendo $E \subseteq O$, $I \subseteq A'$, $E^{\uparrow} = I$ e $I^{\downarrow} = E$. O conjunto dos objetos do conceito formal é denominado extensão e o dos atributos, intensão. Cada elemento da extensão possui todos os da intensão, e é correta a afirmação análoga para a intensão em relação à extensão.

Existem algoritmos para encontrar todos os conceitos formais de um contexto formal, mas esses não são aqui abordados; a próxima seção apresenta reticulados de conceitos e nele, mostra-se um algoritmo que, para produzir tais reticulados, encontra os conceitos formais.

Exemplo 6: Eis alguns exemplos de conceitos formais (E, I) do contexto formal (O, A', R) da Tabela 3.1:

$(\{01, 03, 05, 07, 09, 11, 13, 15, 16, 17, 18, 19, 21, 23, 24\},$	{Nenhuma})
$(\{02, 06, 10, 14, 22\},$	{Não,Normal,Macias})
$(\{04, 08, 12, 20\},$	{Sim,Normal,Rígidas})

O leitor confirme que, em todos esses exemplos, cumprem-se as propriedades $E \subseteq O$, $I \subseteq A'$, $E^{\uparrow} = I$ e $I^{\downarrow} = E$. Em outros termos, as extensões contêm objetos, as intensões contêm atributos e, mais interessante, cada elemento das extensões possui todos os de suas intensões e vice-versa, como pode ser checado no contexto formal. ■

3.3 Reticulados de conceitos

Quando o conjunto de todos os conceitos formais de um contexto formal é hierarquicamente ordenado, recebe a denominação de reticulado de conceitos. Seus conceitos formais se relacionam como $(E_1, I_1) \leq (E_2, I_2)$, quando $E_1 \subseteq E_2$ e $I_2 \subseteq I_1$, sendo (E_1, I_1) chamado subconceito e (E_2, I_2) chamado superconceito.

Para o leitor mais interessado em formalismos matemáticos, apresenta-se aqui o reticulado de conceitos de outra maneira. Os conjuntos-potência $\mathfrak{P}(O)$ e $\mathfrak{P}(A')$ são parcialmente ordenados segundo o operador $\subseteq$. Graças à chamada conexão de Galois, é possível produzir o reticulado de conceitos, contendo as duas ordens em uma estrutura. A conexão de Galois são mapeamentos $\varphi : \mathfrak{P}(O) \rightarrow \mathfrak{P}(A')$

e $\psi : \mathfrak{P}(A') \rightarrow \mathfrak{P}(O)$ (no caso da análise formal de conceitos, tais mapeamentos podem ser as operações de derivação), de modo que, para quaisquer $O_1, O_2 \in \mathfrak{P}(O)$ e $A_1, A_2 \in \mathfrak{P}(A')$:

$$O_1 \subseteq O_2 \Rightarrow \varphi O_2 \subseteq \varphi O_1$$
$$A_1 \subseteq A_2 \Rightarrow \psi A_2 \subseteq \psi A_1$$
$$O_1 \subseteq \psi\varphi O_1$$
$$A_1 \subseteq \varphi\psi A_1$$

Tão adequado quanto representar um contexto formal por meio de uma tabela cruzada, é representar um reticulado por meio de um diagrama. O diagrama representa o reticulado como um grafo, cujos vértices denotam conceitos formais e cujas arestas denotam suas relações $(E_1, I_1) \leq (E_2, I_2)$. Quando dois conceitos formais se relacionam como subconceito e superconceito sem que haja qualquer outro conceito formal entre ambos, seus vértices devem ser conectados por uma aresta no diagrama. O vértice mais alto do diagrama representa o conceito formal cuja extensão contém todos os objetos, enquanto o mais baixo contém todos os atributos em sua intensão.

Para tornar mais confortável e eficiente a leitura de um diagrama, extensões e intensões não precisam ser escritas por completo. Nomes de objetos e de atributos são escritos no diagrama somente uma vez. Desse modo, os objetos têm seus nomes escritos no vértice mais baixo em que primeiro ocorrem, enquanto os nomes dos atributos são escritos no vértice mais alto em que primeiro ocorrem. Todos os atributos possuídos por um objeto (além dos escritos no rótulo de seu vértice) podem ser encontrados nas intensões dos conceitos superiores alcançados pelo vértice daquele objeto. Desse modo, o aspecto mais importante no diagrama é sua estrutura em níveis, ou seja, as alturas dos vértices devem ser consideradas, pois determinam subconceitos e superconceitos.

Exemplo 7: O contexto formal (O, A', R) da Tabela 3.1 produz muitas dezenas de conceitos formais, inviabilizando, devido ao espaço requerido, mostrar aqui seu reticulado de conceitos. Considera-se aqui, portanto, o subconjunto de objetos $\{01, 08, 11, 14, 20, 22\}$. A Figura 3.1 mostra o reticulado de conceitos produzido, em formato de diagrama. Rótulos de objetos aparecem abaixo dos vértices, enquanto os de atributos aparecem em cima.

Cada vértice do diagrama representa um conceito formal. Quando conectados por aresta, dois conceitos formais são subconceito e superconceito diretos. Como mencionado nesta seção, o diagrama tem rótulos em modo reduzido, ou seja, não

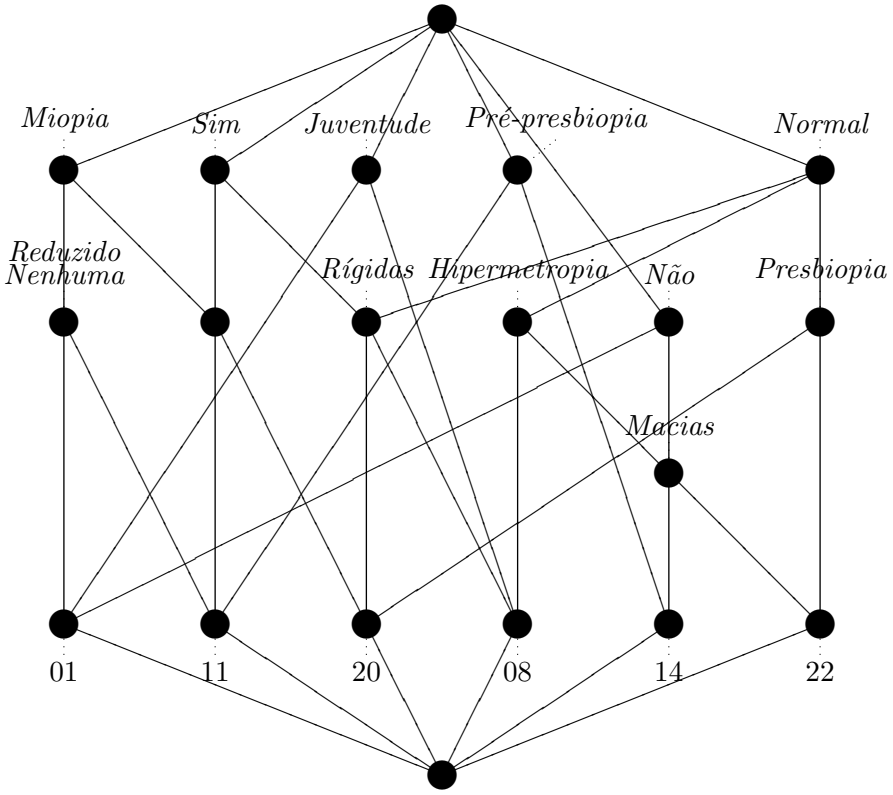

Figura 3.1: Reticulado sobre lentes de contato.

se escrevem extensões e intensões por completo. De qualquer modo, essas podem ser inferidas para qualquer conceito formal. Considerando, por exemplo, o conceito formal de rótulo "Reduzido, Nenhuma": sua extensão é $\{01, 11\}$ (objetos de vértices inferiores) e sua intensão é {Miopia,Reduzido,Nenhuma} (atributos de vértices superiores). De fato, se consultado o contexto formal, constata-se que tais objetos têm tais atributos em comum e vice-versa (tais atributos são compartilhados por tais objetos). ■

Dado um conjunto de conceitos formais de um reticulado, ínfimo e supremo são respectivamente o maior subconceito e o menor superconceito comuns a todos os elementos de tal conjunto. As definições de maior e menor, dentro do escopo de reticulados, dizem respeito à ordenação hierárquica. Em outras palavras, o primeiro

vértice no diagrama no qual se cruzam os caminhos de determinados conceitos formais é chamado ínfimo (se o caminho for para baixo) ou supremo (se o caminho for para cima). O teorema básico de reticulados de conceitos aborda ínfimo $\bigwedge$ e supremo $\bigvee$ de conceitos formais, cujos índices estão em T:

$$\begin{aligned} \bigwedge_{t\in T}(E_t, I_t) &= \left(\bigcap_{t\in T} E_t, \left(\bigcup_{t\in T} I_t \right)^{\downarrow\uparrow} \right) \\ \bigvee_{t\in T}(E_t, I_t) &= \left(\left(\bigcup_{t\in T} E_t \right)^{\uparrow\downarrow}, \bigcap_{t\in T} I_t \right) \end{aligned}$$

Todo reticulado de conceitos possui um vértice superior a todos e outro inferior também a todos; por esse motivo, tal reticulado é completo.

Concluindo o assunto de reticulados de conceitos, o Algoritmo 3 [CR04] mostra um pseudo-código para construção de tais estruturas.

Algoritmo 3 Pseudo-código para construção de reticulado de conceitos.

entradas: contexto formal (O', A, A', R).
resultados: reticulado de conceitos.

```
converter contexto para mono-valorado
inserir (O, O↑) no reticulado
conceitos ← {(O, O↑)}
enquanto conceitos ≠ ∅ execute
  seguintes ← ∅
  para todo (E, I) ∈ conceitos execute
    subconceitos ← subconceitos de (E, I)
    para todo (E1, I1) ∈ subconceitos execute
      inserir (E1, I1) no reticulado
      conectar (E, I) e (E1, I1)
      seguintes ← seguintes ∪ {(E1, I1)}
    fim para
  fim para
  conceitos ← seguintes
fim enquanto
```

Esse algoritmo constrói o reticulado de conceitos em níveis. Ele produz o primeiro conceito formal, seus subconceitos e as arestas necessárias. Depois, para cada conceito formal, o processo é repetido, produzindo-se mais subconceitos e mais arestas.

A operação para produzir subconceitos de (E, I) é a seguinte: para todo atributo x não presente em I, produz-se um candidato $((I \cup \{x\})^{\downarrow}, (I \cup \{x\})^{\downarrow\uparrow})$; dentre tais candidatos, retornam-se os verdadeiros subconceitos. Um candidato qualquer (E_1, I_1) é subconceito se, e somente se, todo atributo y presente em $I_1 \setminus I$ produzir o mesmo $(I \cup \{y\})^{\downarrow}$.

Esse algoritmo tem, no pior caso, complexidade $O(|\text{conceitos}||O||A'|^2)$. Para cada conceito formal (daí o $|\text{conceitos}|$), é preciso produzir seus subconceitos e percorrer os mesmos. Dessas duas operações, a primeira é mais onerosa. Para produzir subconceitos, recorre-se aos operadores de derivação (daí o $|O||A'|$), no máximo $|A'|$ vezes (para todo atributo x não presente em I, conforme descrito no parágrafo anterior). É interessante mencionar que o máximo número de conceitos formais no reticulado de um contexto formal (O, A', R) é $2^{|A'|}$; o reticulado de conceitos cresce, portanto, exponencialmente, dependendo da incidência do contexto formal.

Conclui-se esta seção com o chamado pseudo-reticulado. Esse tem estrutura similar ao reticulado de conceitos, mas é anterior à proposta da análise formal de conceitos. Tal assunto é relevante neste trabalho, pois alguns algoritmos do capítulo seguinte, por questão histórica, adotam tal estrutura.

O pseudo-reticulado, conforme descrito na literatura, tem algumas diferenças em relação ao reticulado de conceitos. Ele é também representado por diagrama, mas, como os vértices podem não ser conceitos formais (tais como descritos aqui), são denominados apenas nós. O pseudo-reticulado contém os seguintes nós (que costumam ter apenas as intensões escritas em rótulos, ignorando-se as extensões):

- nós-atributo contêm somente um elemento na intensão e são os mais superiores na estrutura;

- nós-objeto correspondem aos conceitos formais cujos rótulos reduzidos contêm objetos e são os mais inferiores na estrutura;

- nós-intermediários estão entre nós-objeto e nós-atributo.

Como já mencionado, tais nós não são sinônimos de conceitos formais. Alguns vértices do pseudo-reticulado são somente nós, enquanto alguns conceitos sequer figuram em tal estrutura. Nós-atributo nem sempre são conceitos formais, mas os demais sempre o são. O primeiro conceito formal, ou seja, $(O, O^{\uparrow})$, aparece no pseudo-reticulado somente se existir, no contexto formal, o chamado atributo universal (aquele tido por todos os objetos). O último conceito formal, ou seja, $(A'^{\downarrow}, A')$,

aparece no pseudo-reticulado se, em situação análoga, o contexto formal contiver o chamado objeto universal (aquele que possui todos os atributos).

Exemplo 8: O contexto formal (O, A', R) da Tabela 3.1 produz, para os objetos $\{01, 08, 11, 14, 20, 22\}$, o pseudo-reticulado mostrado na Figura 3.2. Tal estrutura também é escrita em modo reduzido, com os atributos aparecendo apenas uma vez (primando pela clareza, são também mostrados os objetos em rótulos).

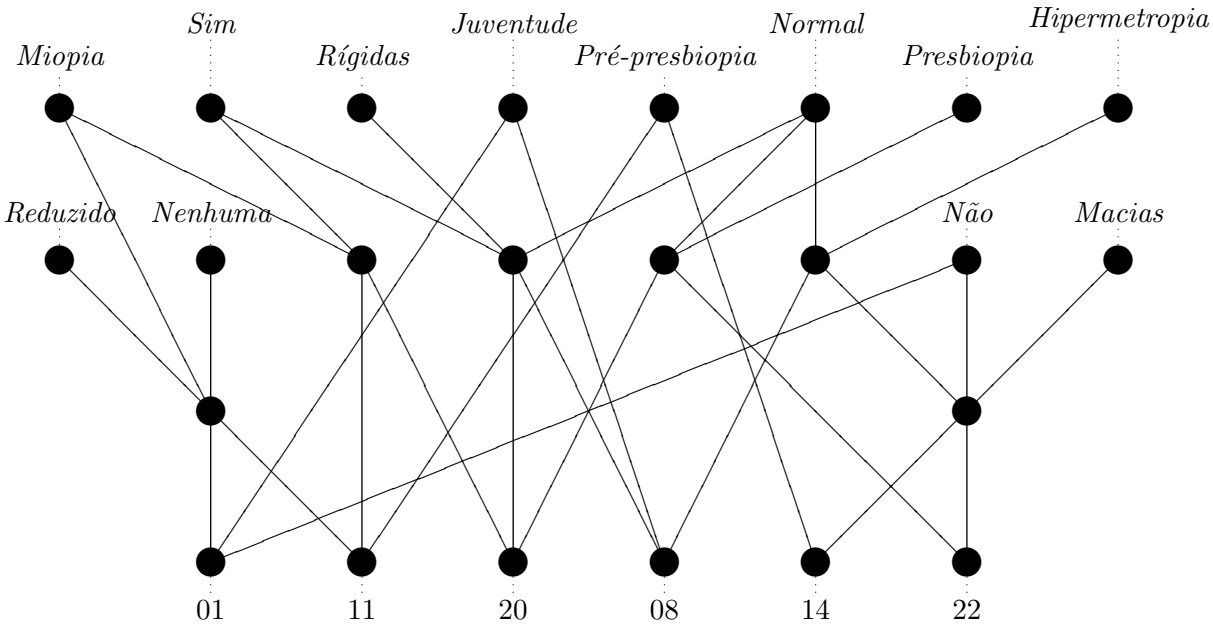

Figura 3.2: Pseudo-reticulado sobre lentes de contato.

Os nós-atributo são aqueles cujos rótulos têm atributos escritos; dentre tais nós, os de rótulos "Reduzido", "Nenhuma", "Macias", "Rígidas" não são conceitos formais. Os nós-objeto são aqueles cujos rótulos têm objetos escritos; todos são conceitos formais. Os demais nós também correspondem a conceitos formais. O leitor observe a ausência do primeiro e último conceitos formais, pois não existem objeto e atributo universais. ■

O Algoritmo 4 [Oos88] mostra o pseudo-código para construção do pseudo-reticulado. Ele primeiro produz todos os nós-atributo, depois, para cada nó-objeto, checa a necessidade de novos nós, calculando a interseção da intensão do nó-objeto

com as demais. O pseudo-código não explicita a produção de arestas, mas esse é um processo simples – primeiro, conectam-se os nós àqueles com elementos em comum na intensão, depois, removem-se as arestas redundantes (ou seja, aquelas com nós não diretamente superiores e inferiores).

Algoritmo 4 Pseudo-código para construção de pseudo-reticulado.

entradas: contexto formal (O', A, A', R).
resultados: pseudo-reticulado.

```
converter contexto para mono-valorado
para todo atributo de A' execute
  inserir nó-atributo na estrutura
fim para
para todo objeto de O execute
  inserir nó-objeto na estrutura
  conectar nó-objeto a nós-atributo adequados
  para todo nó presente no pseudo-reticulado execute
    calcular interseção das intensões de nó e nó-objeto
    produzir nó-intermediário
    se pseudo-reticulado não contém nó-intermediário então
      inserir nó-intermediário no pseudo-reticulado
      conectar nó-intermediário a nós adequados
    fim se
  fim para
  remover arestas redundantes
fim para
```

Capítulo 4

Classificação com reticulados: teoria

Apresentam-se aqui propostas que executam o método de classificação, baseada na análise formal de conceitos. Tal assunto tem seus fundamentos teóricos apresentados em capítulos anteriores. Embora tais capítulos já tenham formalizado notações para diversos elementos, enumeram-se essas aqui novamente, pois continuarão a ser usadas: O é o conjunto de objetos; as entradas das fases de classificação são $O' \subseteq O$ (primeira fase) e $O'' \subseteq O$ (segunda fase); A é o conjunto que se particiona em atributos B e classe $\{c\}$; A' é o conjunto particionado em domínios dos atributos e domínio da classe; $B' \subset A'$ contém os valores (domínios) de atributos e $C' \subset A'$, os valores (domínio) da classe.

Apresentam-se aqui seis propostas; as quatro primeiras já foram publicadas [FFNN04, NN05], enquanto as duas últimas (na verdade, duas versões baseadas na mesma intuição) foram elaboradas durante este trabalho. Duas dessas propostas empregam o pseudo-reticulado, enquanto o reticulado, tal como formalizado pela análise formal de conceitos, é a estrutura usada nos demais algoritmos. Algumas dessas propostas trabalham com regras, mas outras usam o reticulado para classificar.

O capítulo dedica uma seção para cada proposta: a seção 4.1 apresenta o algoritmo "Grand"; 4.2 descreve o chamado "Rulearner"; o algoritmo "Legal" é tema da seção 4.3; "Galois" é assunto de 4.4; e, por fim, as seções 4.5 e 4.6 apresentam os inéditos "Similares1" e "Similares2". Após descrever os algoritmos (primeira e segunda fases de classificação), as seções contêm exemplos de classificações bem e mal-sucedidas.

4.1 Grand

O algoritmo "Grand" [Oos88, Oos94] (nome baseado em "graph-based induction") é a mais antiga das propostas aqui apresentadas e seu funcionamento é bastante intuitivo. Ele começa a primeira fase de classificação construindo um pseudo-reticulado (o leitor observe que não se emprega aqui o reticulado da análise formal de conceitos) para o contexto formal (O', A, A', R). O algoritmo considera, como mostra a notação do contexto formal, a classe e seu domínio, além dos atributos e seus valores.

"Grand" produz regras de classificação baseadas no pseudo-reticulado. Uma vez pronta tal estrutura, o algoritmo nela encontra os nós-classe de intensões $classe \in C'$ e depois processa seus nós inferiores (não apenas imediatamente inferiores) de intensões $valores \subseteq A'$. Para cada nó-classe e para cada nó inferior seu, produz-se uma regra com $valores \setminus \{classe\}$ no antecedente e $classe$ no conseqüente, se respeitadas as condições: $classe \in (valores \setminus \{classe\})^{\downarrow\uparrow}$ e $valores$ for minimal.

Em outros termos, "Grand" produz regras que, com o mínimo de elementos em seus antecedentes, cubram o máximo de objetos. Procurando por nós descendentes de nós-classe com intensões $valores$ minimais, encontram-se os filhos desses nós-classe. Adicionando a condição $classe \in (valores \setminus \{classe\})^{\downarrow\uparrow}$, garante-se que as regras produzidas são corretas, pois todos os objetos com $valores \setminus \{classe\}$ têm também o valor $classe$. Essa segunda condição também faz, caso os filhos de nós-classe não atendam a mesma, a pesquisa continuar estrutura abaixo, à procura de nós com intensões minimais e que produzam regras corretas.

O Algoritmo 5 mostra o pseudo-código da primeira fase do "Grand". Depois de pronto o pseudo-reticulado, construído conforme o já apresentado Algoritmo 4, checam-se os filhos dos nós-classe. Caso tais filhos não produzam regras corretas, é necessário continuar, como já mencionado, a pesquisa pela estrutura. Quando se produz regra baseada em algum nó, a pesquisa não continua em seus filhos. É importante ter cuidado, quando programando tal algoritmo, para não se visitar repetidamente os nós, tarefa desnecessária e extremamente onerosa (por isso a expressão "não visitados" aparece no pseudo-código).

O pseudo-código contém um teste condicional, no qual é checado se o nó analisado pode produzir regra correta. Entretanto, na prática, tal trecho demanda mais linhas de código-fonte. Dado o nó analisado em tal trecho, é preciso remover de sua intensão o valor de classe, derivar esse conjunto (obtendo objetos) e depois derivar novamente (obtendo valores de atributos); isso para checar uma das condições aqui apresentadas – o valor de classe deve pertencer ao conjunto resultante das duas derivações. Tal

Algoritmo 5 Pseudo-código da primeira fase do algoritmo "Grand".

entradas: contexto formal (O', A, A', R).
resultados: pseudo-reticulado da entrada; conjunto de regras.

```
montar pseudo-reticulado do contexto formal
para todo nó-classe do pseudo-reticulado execute
  marcar filhos do nó-classe como candidatos
  enquanto existirem nós candidatos execute
    remover nó de candidatos
    se nó produz regra correta então
      produzir regra com nó
    senão
      marcar filhos do nó (não visitados) como candidatos
    fim se
  fim enquanto
fim para
```

processo mostra que, embora use um pseudo-reticulado (cujos nós não armazenam extensões), o algoritmo "Grand" ainda precisa ter acesso ao contexto formal, para executar as operações de derivação.

Após a construção do pseudo-reticulado e do conjunto de regras, o algoritmo "Grand" executa a segunda fase de classificação. Dado, como entrada, um contexto formal (O'', A, A', R) cuja tabela tenha apenas células em branco para os valores de classe, procuram-se regras que cubram esses objetos, classificando-os. Esse processo é simples, similar à tradicional classificação baseada em regras.

Existem aqui, como no método clássico de classificação baseada em regras, dois problemas que merecem destaque: um objeto pode não ser coberto por nenhuma regra; ou, pelo contrário, mais de uma regra pode cobrir algum objeto. "Grand" sugere, para o primeiro problema, classificar o objeto com o valor de classe mais votado, ou seja, o conseqüente mais freqüente em regras que cubram aquele objeto. Para o segundo problema, o objeto não é classificado.

Exemplo 9: O algoritmo "Grand" é executado para o contexto formal (O, A, A', R) convertido na Tabela 3.1, considerando-se $c =$"Lentes".

A primeira fase do algoritmo começa construindo o pseudo-reticulado para o contexto formal (O', A, A', R), onde $O' = \{01, 08, 11, 14, 20, 22\}$. Essa estrutura é mos-

trada na Figura 3.2. Continuando a primeira fase, começa-se a fase de construção do conjunto de regras. Encontram-se os nós-classe {Nenhuma}, {Macias}, {Rígidas}. Checam-se seus filhos, cujas intensões são {Miopia, Reduzido, Nenhuma}, {Hipermetropia, Não, Normal, Macias}, {Sim, Normal, Rígidas}. A pesquisa é interrompida, pois todos esses nós produzem regras corretas, mostradas na Figura 4.1.

Miopia $\wedge$ Reduzido	$\Rightarrow$	Nenhuma
Hipermetropia $\wedge$ Não $\wedge$ Normal	$\Rightarrow$	Macias
Sim $\wedge$ Normal	$\Rightarrow$	Rígidas

Figura 4.1: Conjunto de regras produzido pelo algoritmo "Grand".

Concluída a primeira fase, deseja-se executar a fase seguinte, classificando o conjunto $O'' = \{03, 04, 06\}$. O objeto "03", coberto pela primeira regra, é classificado como "Nenhuma"; o objeto "04" é classificado como "Rígidas", pois é coberto pela terceira regra; por fim, o objeto "06", coberto pela segunda regra, é classificado como "Macias". Todos esses objetos são corretamente classificados. ■

Exemplo 10: O algoritmo "Grand", em sua primeira fase, constrói o pseudo-reticulado para o contexto formal (O', A, A', R) (baseado em (O, A, A', R) convertido na Tabela 3.1, considerando-se $O' = \{01, 08, 11, 14, 20, 22\}$ e $c =$"Lentes"). Essa estrutura é mostrada na Figura 3.2. A primeira fase depois constrói o conjunto de regras, mostrado na Figura 4.1.

A segunda fase classifica o conjunto $O'' = \{02, 16, 24\}$. O objeto "02" não é coberto por nenhuma das regras e, por isso, não pode ser classificado. A terceira regra cobre o objeto "16" e o classifica como "Rígidas", contudo, seu valor correto de classe seria "Nenhuma". Por fim, o objeto "24", cujo valor correto de classe também seria "Nenhuma", é erroneamente classificado como "Rígidas" também pela terceira regra. Esses são exemplos de classificações mal-sucedidas. ■

4.2 Rulearner

O algoritmo "Rulearner" [Sah95] é uma proposta similar à da seção anterior. Ele também usa um pseudo-reticulado (o leitor novamente observe que não se trata

do reticulado da análise formal de conceitos) para produzir regras de classificação. Duas primeiras diferenças que se observam nesse algoritmo são, primeiro, o fato dele ignorar valores de classe na construção do pseudo-reticulado e, depois, o fato do algoritmo demandar um parâmetro de entrada.

A primeira fase de "Rulearner", embora diferente do algoritmo anterior, baseia-se na mesma estratégia: produzir regras que possuam poucos atributos e cubram muitos objetos, preferindo, portanto, nós mais ao topo do pseudo-reticulado. Procuram-se, para isso, nós que possuam, dentre seus descendentes, o máximo de nós-objeto, contanto que esses tenham algum valor predominante de classe (daí a necessidade do parâmetro: determinar tal predominância).

O Algoritmo 6 mostra o pseudo-código da primeira fase de "Rulearner". As entradas são o contexto formal (O', A, A', R) e o parâmetro de impureza. O algoritmo ignora, durante a construção do pseudo-reticulado, a classe e seu domínio, como se esses não pertencessem ao contexto formal. O parâmetro de impureza é detalhado a seguir. As saídas da primeira fase são o pseudo-reticulado e o conjunto de regras.

Algoritmo 6 Pseudo-código da primeira fase do algoritmo "Rulearner".

entradas: contexto formal (O', A, A', R); parâmetro de impureza.
resultados: pseudo-reticulado da entrada; conjunto de regras.

```
montar pseudo-reticulado do contexto formal (O', B, B', R)
marcar nós do pseudo-reticulado como ativos
rotular nós do pseudo-reticulado, usando parâmetro de impureza
enquanto existirem nós ativos execute
  encontrar nó ativo não-impuro com maior cobertura
  produzir regra com nó
  para todo nó-objeto ativo descendente do nó execute
    para todo nó ativo ascendente do nó-objeto execute
      decrementar cobertura do nó ascendente
      se cobertura do nó ascendente = 0 então
        marcar nó ascendente como inativo
      fim se
    fim para
    marcar nó-objeto como inativo
  fim para
fim enquanto
```

O pseudo-código primeiro constrói o pseudo-reticulado, usando, para isso, o Algoritmo 4. Uma vez pronta tal estrutura, marcam-se todos seus nós como ativos. Apenas nós ativos podem produzir regras; quando inativos, eles são ignorados em muitos passos do algoritmo (como se, a grosso modo, eles não mais existissem). Depois de ativados, os nós são rotulados com valores de classe ou como impuros, usando o parâmetro fornecido como entrada.

Para entender os rótulos dos nós, é importante primeiro definir alguns termos adotados nesse algoritmo. Dado qualquer nó do pseudo-reticulado, seus descendentes são os nós inferiores por ele alcançáveis, por caminhos de arestas. Definem-se ascendentes analogamente. Importante comentar que descendentes e ascendentes de um nó incluem ele próprio. Por fim, define-se cobertura de um nó como a quantidade de nós-objeto presente em seus descendentes. Embora cobertura seja uma palavra já empregada para regras de classificação, ela é também definida para nós do pseudo-reticulado, pois, como será mostrado, tais nós podem tornar-se regras (mantendo coerente o emprego de tal palavra).

Os nós são, como mencionado, rotulados com valores de classe ou como impuros. Nós não-impuros são aqueles que possuem, dentre seus descendentes, uma quantidade considerável de nós-objeto com mesmo valor de classe; os não-impuros têm como rótulo tal valor predominante de classe. Já com nós impuros, seus nós-objeto descendentes não definem um valor predominante de classe. O parâmetro de impureza, recebido como entrada, controla essa predominância; ele determina a porcentagem tolerada de outros valores de classe, dentre os nós-objeto descendentes. Observa-se então que, embora ignore a classe e seus valores para construir o pseudo-reticulado, o algoritmo precisa dos mesmos para rotular os nós, acessando, para isso, o contexto formal.

Com nós ativados e rotulados, o algoritmo encontra aquele não-impuro de maior cobertura, para produzir cada regra. O nó escolhido fornece sua intensão como antecedente da regra e seu rótulo (um valor de classe) como conseqüente. Caso mais de um nó não-impuro possua a maior cobertura, prefere-se aquele cuja intensão é menor. Com tais estratégias, produzem-se regras que, com poucos atributos (menor tamanho de intensão), cubram muitos objetos (maior cobertura).

Após produzir cada regra, o algoritmo desativa alguns nós do pseudo-reticulado. Encontram-se primeiro os nós-objeto descendentes do primeiro nó (aquele que deu origem à regra). Encontram-se depois os ascendentes de tais nós-objeto, decrementando em um suas coberturas; desativam-se os ascendentes cujas coberturas

chegarem a zero. Desativam-se, por fim, os nós-objeto em questão, para que não sejam considerados na procura por próximas regras.

A segunda fase de classificação de "Rulearner" é idêntica à do algoritmo anterior: procuram-se regras que cubram os objetos de O'', para que esses sejam classificados.

Exemplo 11: "Rulearner" é executado para o contexto formal (O, A, A', R) convertido na Tabela 3.1, considerando-se $c =$"Lentes", e para um parâmetro de impureza de 20% (ou seja, requer-se que, no mínimo, 80% de nós-objeto votem em um mesmo valor de classe, para produzir regras).

A primeira fase começa construindo o pseudo-reticulado para o contexto formal (O', B, B', R), onde $O' = \{01, 08, 11, 14, 20, 22\}$. Essa estrutura é mostrada na Figura 4.2.

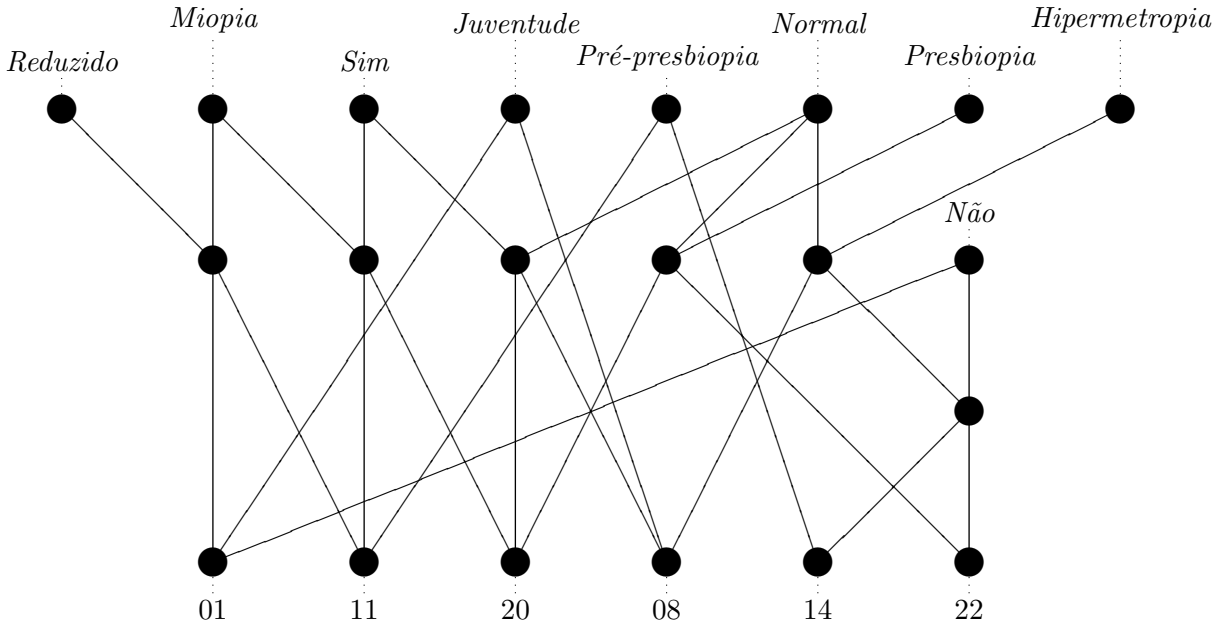

Figura 4.2: Pseudo-reticulado produzido pelo algoritmo "Rulearner".

Descreve-se aqui, a título de exemplo, a produção de uma das regras (mencionam-se os detalhes julgados relevantes). Procura-se primeiro o nó não-impuro com a maior cobertura; como existem quatro não-impuros com cobertura 2 – aqueles de intensões {Reduzido}, {Hipermetropia, Não, Normal}, {Sim, Normal}, {Miopia, Reduzido} –, prefere-se o nó com menor tamanho de intensão – de intensão {Reduzido}

e rótulo "Nenhuma". Produz-se então a regra "Reduzido ⇒ Nenhuma". Desativam-se depois os nós-objeto descendentes – de intensões {Juventude, Miopia, Não, Reduzido} e {Pré-presbiopia, Miopia, Sim, Reduzido} – e decrementam-se as coberturas de seus ascendentes (esses não são aqui enumerados, pois isso não traria informação relevante). O algoritmo continua produzindo outras regras e, após seu término, tem-se o conjunto de regras da Figura 4.3.

Reduzido	⇒	Nenhuma
Hipermetropia ∧ Não ∧ Normal	⇒	Macias
Sim ∧ Normal	⇒	Rígidas

Figura 4.3: Conjunto de regras produzido pelo algoritmo "Rulearner".

Concluída a primeira fase, executa-se a segunda, para os objetos $O'' = \{03, 04, 06\}$. O objeto "03", coberto pela primeira regra, é classificado como "Nenhuma"; o objeto "04" é classificado como "Rígidas", por ser coberto pela terceira regra; o objeto "06" é classificado como "Macias" pela segunda regra. Todos esses exemplos foram corretamente classificados. ■

Exemplo 12: O algoritmo "Rulearner", em sua primeira fase, constrói o pseudo-reticulado para o contexto formal (O', B, B', R) (baseado no contexto (O, A, A', R) convertido na Tabela 3.1, considerando-se $O' = \{01, 08, 11, 14, 20, 22\}$ e $c =$"Lentes"), com um parâmetro de impureza de 20%. Essa estrutura é mostrada na Figura 4.2. A primeira fase depois constrói o conjunto de regras, mostrado na Figura 4.3.

A segunda fase classifica o conjunto $O'' = \{02, 16, 24\}$. Como no algoritmo da seção anterior, esses são, aqui também, exemplos de classificações mal-sucedidas. O objeto "02" não é classificado, pois não é coberto por nenhuma regra; o objeto "16", cujo valor correto de classe seria "Nenhuma", é classificado como "Rígidas" pela terceira regra; o objeto "24", também coberto pela terceira regra, é classificado como "Rígidas", no entanto, "Nenhuma" seria o correto. ■

4.3 Legal

O algoritmo "Legal" [LN90, Ngu94] (nome baseado em "learning with Galois lattice") trabalha com um reticulado (não mais um pseudo-reticulado) e produz regras baseadas em tal estrutura. A grande vantagem de tal proposta é não construir o reticulado completo, o que garante ganhos em tempo, quando comparado aos demais algoritmos aqui apresentados. O reticulado construído possui o primeiro conceito formal (aquele mais ao topo), mas não necessariamente possui outros conceitos; em outras palavras, em tal estrutura, quaisquer conceitos formais certamente possuem um supremo, mas não existe certeza quanto a um ínfimo. A construção de tal reticulado incompleto é orientada por dois parâmetros de entrada, descritos mais à frente.

A grande desvantagem do algoritmo é requerer quatro parâmetros de entrada. Cada fase de classificação emprega dois desses parâmetros. O problema é determinar seus valores, tarefa não concluída sequer pelos autores da proposta; existem algumas sugestões de melhores valores, mas eles dependem bastante do conjunto de entrada [NN96]. O algoritmo tem mais uma desvantagem: realizar somente classificação binária.

A primeira fase de "Legal" recebe um contexto formal (O', A, A', R) e, depois de produzir um reticulado, produz também regras baseadas em tal estrutura. Como o algoritmo realiza somente classificação binária, é necessário que $|C'| = 2$. É também necessário rotular tais valores de classe como positivos e negativos, pois tal informação será empregada na construção do reticulado. Particiona-se aqui o conjunto O' de objetos em O^+ (objetos positivos) e O^- (objetos negativos), dependendo de seus valores de classe. É importante comentar que positivos e negativos são apenas rótulos; não indicam que um dos valores é mais importante.

O reticulado de tal algoritmo considera somente objetos de O^+ e ignora a classe e seus valores (afinal, todos os objetos têm valores positivos de classe). Ele é construído segundo o Algoritmo 3, com algumas alterações. O primeiro conceito formal (E, I) contém somente objetos positivos em sua extensão, ou seja, $E = O^+$. A segunda alteração é a derivação de valores de atributos, para formar conceitos formais; o conjunto de objetos resultante de tal derivação contém apenas elementos positivos. Em resumo, tal reticulado refere-se ao contexto formal (O^+, B, B', R).

O reticulado aqui empregado é incompleto, pois contém somente conceitos formais válidos. Dado o parâmetro $0 \leq \alpha \leq 1$ (um dos quatro recebidos como entrada), um conceito formal (E, I) de tal reticulado é chamado válido se $|E|/|O^+| \geq \alpha$. Ou

seja, conceitos formais válidos são aqueles cujas extensões possuem um tamanho mínimo desejado, ou melhor, são aqueles que cobrem um número mínimo de objetos positivos. Como as extensões de conceitos formais na região superior do reticulado têm mais elementos, o reticulado incompleto pode, dependendo de α, possuir somente tais conceitos. As regras formadas por conceitos válidos são também chamadas válidas.

Uma vez pronto o reticulado (com objetos positivos e conceitos formais válidos), conclui-se a primeira fase, produzindo-se regras pertinentes. Essas são formadas por conceitos formais também chamados pertinentes. Para definir conceitos pertinentes, define-se primeiro conceito coerente. Dado o parâmetro $0 \leq \beta \leq 1$, um conceito formal (E, I) do reticulado é chamado coerente se $|E|/|O^-| \leq \beta$. Ou seja, conceitos formais coerentes cobrem um número máximo de objetos negativos. Conceitos formais pertinentes são válidos e coerentes; definem-se regras coerentes analogamente.

Importante mencionar: Se um conceito formal é coerente, todos os seus descendentes também o são; mas se um conceito formal é, no entanto, válido, todos os seus ascendentes também o são.

O Algoritmo 7 mostra o pseudo-código da primeira fase de "Legal". O leitor observe que, no caso de $\alpha = 0$, o reticulado é completo.

Algoritmo 7 Pseudo-código da primeira fase do algoritmo "Legal".

entradas: contexto formal (O', A, A', R); parâmetros α e β.
resultados: reticulado incompleto; conjunto de regras pertinentes.

```
montar reticulado do contexto (O+, B, B', R), usando α
para todo conceito formal do reticulado execute
  se conceito formal é coerente, usando β então
    produzir regra com conceito formal
  fim se
fim para
```

A segunda fase de "Legal" recebe um contexto formal (O'', A, A', R), com objetos sem valores de classe, e os classifica com regras pertinentes. Tais regras têm intensões de conceitos formais em seus antecedentes e apresentam todas somente um conseqüente – o valor positivo de classe (afinal, o reticulado considera apenas objetos positivos).

Ao contrário das propostas que também produzem regras, este algoritmo usa parâmetros $0 \leq \lambda \leq 1$ e $0 \leq \gamma \leq 1$ (os dois restantes) para classificar os objetos.

Para qualquer objeto a ser classificado, sendo n o número de regras que cobrem tal objeto e m o tamanho do conjunto de regras, o objeto é classificado como positivo se $n/m \geq \lambda$ ou como negativo se $n/m < \gamma$; em caso contrário ($\gamma \leq n/m < \lambda$), o objeto não é classificado. Para evitar o problema de objetos não classificados, sugere-se $\lambda = \gamma$.

Exemplo 13: "Legal" é executado para o contexto formal (O, A, A', R) convertido na Tabela 3.1, com $c =$"Lentes". Converte-se tal problema para outro com dois valores de classe: "Macias" e "Rígidas" tornam-se o valor positivo, enquanto "Nenhuma" é considerado como negativo. Todo objeto que possua "Macias" ou "Rígidas" é, portanto, um objeto positivo. É também preciso determinar os quatro parâmetros; analisando-se tal exemplo, determinam-se $\alpha = 0.5$, $\beta = 0.1$, $\lambda = 0.4$, $\gamma = 0.4$, pois tal configuração produz bons resultados (tal análise é baseada em estudos dos autores do algoritmo [NN96]).

A primeira fase produz o reticulado incompleto do contexto formal (O^+, B, B', R), sendo $O' = \{01, 08, 11, 14, 20, 22\}$ e $O^+ = \{08, 14, 20, 22\}$. Essa estrutura é mostrada na Figura 4.4 (como o reticulado é mostrado em modo reduzido, usam-se somente os rótulos dos valores de atributos; o modo reduzido aqui não funcionaria corretamente com rótulos de objetos). O leitor observe que os conceitos formais são válidos, pois suas extensões (deduzíveis com auxílio do contexto formal) cobrem o número mínimo de objetos positivos.

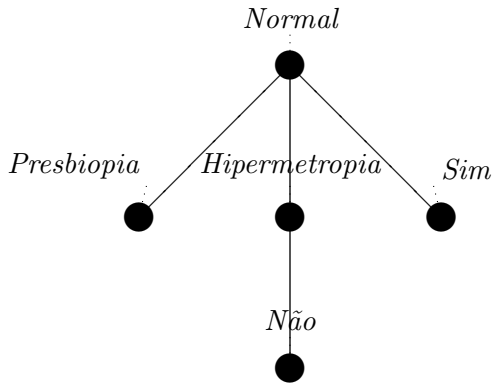

Figura 4.4: Reticulado incompleto produzido pelo algoritmo "Legal".

A primeira fase continua, produzindo um conjunto de regras pertinentes, mos-

trado na Figura 4.5. Como o reticulado possui apenas conceitos formais coerentes (além de válidos, evidentemente), todos eles produzem regras.

Normal	$\Rightarrow$	Classe Positiva
Normal $\wedge$ Presbiopia	$\Rightarrow$	Classe Positiva
Normal $\wedge$ Hipermetropia	$\Rightarrow$	Classe Positiva
Normal $\wedge$ Sim	$\Rightarrow$	Classe Positiva
Normal $\wedge$ Hipermetropia $\wedge$ Não	$\Rightarrow$	Classe Positiva

Figura 4.5: Conjunto de regras produzido pelo algoritmo "Legal".

A segunda fase classifica o conjunto $O'' = \{03, 06\}$. Com os parâmetros $\lambda = 0.4$ e $\gamma = 0.4$, cada objeto precisa ser coberto por, no mínimo, duas das cinco regras. O objeto "06", coberto por três regras (primeira, terceira e quinta), é corretamente classificado como positivo (sua classe original é "Macias"). O objeto "03" é corretamente classificado como negativo (sua classe original é "Nenhuma"), pois nenhuma das regras o cobre. ■

Exemplo 14: "Legal" é executado para o contexto formal (O, A, A', R) convertido na Tabela 3.1, com $c =$"Lentes". Converte-se tal problema para classificação binária, conforme descrito no exemplo anterior, e determinam-se $\alpha = 0.5$, $\beta = 0.1$, $\lambda = 0.4$, $\gamma = 0.4$. A Figura 4.4 mostra o reticulado do contexto formal (O^+, B, B', R), sendo $O' = \{01, 08, 11, 14, 20, 22\}$ e $O^+ = \{08, 14, 20, 22\}$. Por fim, a Figura 4.5 mostra o conjunto de regras pertinentes, resultado da primeira fase de classificação.

A segunda fase classifica o conjunto $O'' = \{02, 16\}$. Com os parâmetros $\lambda = 0.4$ e $\gamma = 0.4$, cada objeto precisa ser coberto por, no mínimo, duas das cinco regras. O objeto "02", coberto por uma regra, é erroneamente classificado como negativo (sua classe original é "Macias"). Já o objeto "06" (cuja classe original é "Nenhuma") é também erroneamente classificado, mas como positivo, pois é coberto por duas regras. ■

4.4 Galois

O algoritmo "Galois" [CR04] (esse nome é, na verdade, empregado na primeira versão do algoritmo [CR93], na qual baseia-se esta) usa o reticulado completo de conceitos e não mais trabalha com regras. Como mostrado a seguir, é requerido um parâmetro de entrada.

Em sua primeira fase, "Galois" recebe o contexto formal (O', A, A', R) e produz o reticulado, ignorando a classe e seus valores. O reticulado corresponde, logo, ao contexto formal (O', B, B', R). Para executar tal fase, recorre-se ao Algoritmo 3.

Em sua segunda fase, "Galois" recebe o contexto formal (O'', A, A', R) e também um parâmetro de impureza. O Algoritmo 8 mostra o pseudo-código da segunda fase de classificação.

Algoritmo 8 Pseudo-código da segunda fase do algoritmo "Galois".

entradas: reticulado da primeira fase;
contexto formal (O'', A, A', R);
parâmetro de impureza.
resultados: valores de classe para O''.

```
para todo objeto do contexto formal execute
  conceitos ← {primeiro conceito formal}
  seguintes ← ∅
  para todo conceito formal em conceitos execute
    se intensão do objeto contém intensão do conceito formal então
      considerar valores mais votados de classe, usando parâmetro
      seguintes ← subconceitos não visitados do conceito formal
      marcar seguintes como visitados
    fim se
    conceitos ← seguintes
    seguintes ← ∅
  fim para
  classificar objeto segundo valor de classe mais votado
fim para
```

Para classificar objetos sem valores de classe, "Galois" caminha pelo reticulado, usando o chamado parâmetro de impureza. Para cada objeto a ser classificado, o algoritmo considera conceitos formais que tenham certas propriedades e, baseando-se nos valores de classe sugeridos por tais conceitos, classifica aquele objeto. As

propriedades mencionadas são as seguintes: a intensão do conceito formal deve ser subconjunto dos valores de atributos do objeto; além disso, os valores de classe sugeridos pelo conceito formal devem, para serem considerados, ser eleitos por determinada proporção da extensão (a medida de impureza determina quantos outros valores de classe são tolerados na extensão). Ao fim do processo, o objeto é classificado segundo eleição dos valores de classe; ganha aquele mais votado nos conceitos formais (considerando as duas propriedades aqui descritas).

Com tal processo, "Galois" considera, a grosso modo, todas as combinações de valores de atributos, presentes no reticulado e pertencentes ao objeto a ser classificado. É importante comentar que, embora a classe e seus valores tenham sido ignorados na construção do reticulado, eles são requeridos na segunda fase. É adequado armazenar, em cada conceito formal, as quantidades de objetos com cada valor de classe, a fim de garantir mais eficiência ao algoritmo.

Exemplo 15: O algoritmo "Galois" é executado para o contexto formal (O, A, A', R) convertido na Tabela 3.1, considerando-se $c =$"Lentes". A primeira fase do algoritmo produz, para $O' = \{01, 08, 11, 14, 20, 22\}$, o reticulado da Figura 4.6, ignorando a classe e seus valores, para a construção.

Deseja-se, na segunda fase, classificar o objeto "03", com parâmetro de impureza 40%. Caminha-se no reticulado, pesquisando-se os conceitos formais (reparar que suas intensões são subconjuntos dos valores de atributos de "03"):

$(\{01, 02, 03, 04, 05, 06\},$	$\varnothing)$
$(\{01, 02\},$	{Juventude})
$(\{01, 03, 05\},$	{Miopia})
$(\{02, 03, 05\},$	{Sim})
$(\{03, 05\},$	{Miopia,Sim})
$(\{01, 03\},$	{Miopia,Reduzido})

Do primeiro, segundo e quinto conceitos formais, não se consideram valores de classe, pois suas extensões não elegem nenhum com mais de 60%. Do terceiro e sexto conceitos formais, considera-se o valor "Nenhuma", pois as extensões de ambos os conceitos elegem tal valor de classe com mais de 60%. O valor "Rígidas" é eleito pelo quarto conceito formal com mais de 60%. O objeto "03" é, portanto, corretamente classificado como "Nenhuma", valor de classe eleito dentre os seis conceitos formais pesquisados. ■

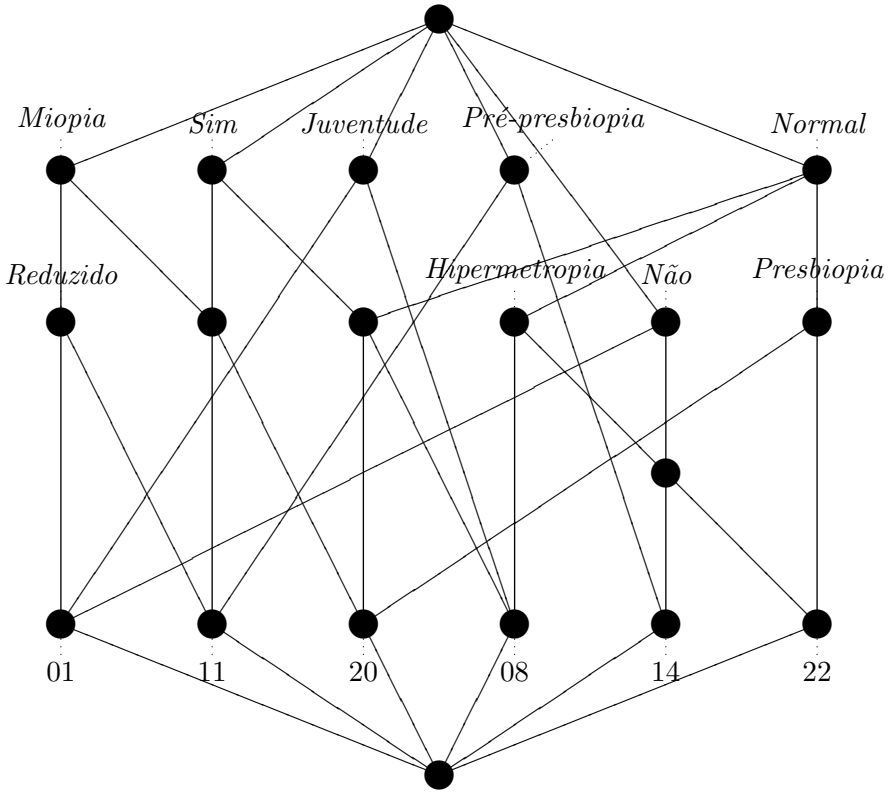

Figura 4.6: Reticulado produzido pelo algoritmo "Galois". Os algoritmos "Similares1" e "Similares2" também produzem essa estrutura.

Exemplo 16: O algoritmo "Galois" é executado para o contexto formal (O, A, A', R) convertido na Tabela 3.1, considerando-se $c =$"Lentes". A primeira fase do algoritmo produz, para $O' = \{01, 08, 11, 14, 20, 22\}$, o reticulado da Figura 4.6, ignorando a classe e seus valores, para a construção.

Deseja-se, na segunda fase, classificar o objeto "02", com parâmetro de impureza 40%. Caminha-se no reticulado, pesquisando-se os conceitos formais:

$(\{01, 02, 03, 04, 05, 06\},$	$\varnothing)$
$(\{01, 02\},$	$\{\text{Juventude}\})$
$(\{01, 03, 05\},$	$\{\text{Miopia}\})$
$(\{01, 04, 06\},$	$\{\text{Não}\})$
$(\{02, 04, 05, 06\},$	$\{\text{Normal}\})$

O terceiro conceito formal elege o valor de classe "Nenhuma", enquanto o quarto elege "Macias". Entretanto, os outros três conceitos formais não têm valores de classe considerados, pois não conseguem os mínimos 60%. O objeto "02" não pode ser classificado, devido ao impasse entre "Nenhuma" e "Macias". ■

4.5 Similares1

O algoritmo "Similares1" é uma proposta inédita, desenvolvida durante este trabalho. Ele usa apenas o reticulado da análise formal de conceitos e não requer nenhum parâmetro de entrada. Enquanto os algoritmos apresentados nas seções anteriores procuravam conceitos formais (ou nós, no caso do pseudo-reticulado) que cobrissem muitos objetos com poucos atributos, este algoritmo introduz a noção de similaridade, para classificar.

A primeira fase de "Similares1" recebe um contexto formal (O', A, A', R) e produz um reticulado, ignorando a classe e seu domínio. O reticulado corresponde, na verdade, ao contexto formal (O', B, B', R); a classe e seus valores são empregados em outras tarefas, não na construção da estrutura. Essa primeira fase, idêntica à do algoritmo anterior, emprega o Algoritmo 3.

A segunda fase de "Similares1" recebe um contexto formal (O'', A, A', R) cujos objetos não possuam nenhum dos valores de classe e, baseando-se no reticulado, classifica tais objetos. Em resumo, para cada objeto, encontram-se os conceitos formais mais similares a ele e, analisando-se tais conceitos, escolhe-se a classe. O Algoritmo 9 mostra o pseudo-código para executar essa segunda fase.

Uma primeira questão quanto ao algoritmo "Similares1" é: como calcular similaridade? O pseudo-código contém uma variável para armazenar a similaridade máxima e um conjunto para armazenar os conceitos formais mais similares. Durante a pesquisa pelo reticulado, começando no primeiro conceito formal (aquele do topo), se algum conceito for mais similar ao objeto do que a similaridade máxima já calculada, essa é trocada pela nova similaridade e começa-se um novo conjunto de conceitos formais mais similares; caso a nova similaridade seja idêntica à máxima já calculada, o conjunto mencionado apenas recebe mais um elemento. Descreveu-se aqui como encontrar os conceitos formais mais similares ao objeto que se quer classificar, mas ainda é preciso definir essa similaridade. Para isso, o leitor suponha o objeto $o \in O''$ e alguns conceitos formais (E_1, I_1), (E_2, I_2), (E_3, I_3) (inventa-se também um conjunto de valores de atributos $\{a_1, a_2, a_3, a_4\}$):

Algoritmo 9 Pseudo-código da segunda fase do algoritmo "Similares1".

entradas: reticulado da primeira fase; contexto formal (O'', A, A', R).

resultados: valores de classe para O''.

```
para todo objeto do contexto formal execute
    similaridade máxima = 0
    similares ← ∅
    conceitos ← {primeiro conceito formal}
    seguintes ← ∅
    para todo conceito formal de conceitos execute
        marcar conceito formal como visitado
        calcular similaridade do conceito formal com objeto
        se similaridade > similaridade máxima então
            similaridade máxima ← similaridade
            similares ← {conceito formal}
        senão se similaridade = similaridade máxima então
            adicionar conceito formal a similares
        fim se
        para todo subconceito não visitado do conceito formal execute
            se compensa ir para subconceito então
                adicionar subconceito a seguintes
            fim se
        fim para
        conceitos ← seguintes
        seguintes ← ∅
    fim para
    encontrar em similares conceito formal mais convicto
    classificar objeto segundo conceito formal mais convicto
fim para
```

$$
\begin{array}{ll}
o, & \text{com valores de atributos } \{a_1, a_2\} \\
(E_1, I_1), & \text{onde } I_1 = \{a_1, a_2, a_3, a_4\} \\
(E_2, I_2), & \text{onde } I_2 = \{a_1, a_3\} \\
(E_3, I_3), & \text{onde } I_3 = \{a_1, a_2, a_3\}
\end{array}
$$

A função de similaridade recebe como argumentos um objeto composto por valores de atributos e um conceito formal. O segundo argumento contém extensão e intensão, mas o primeiro contém apenas valores de atributos. Parece adequado, portanto, tentar elaborar uma medida de similaridade que trabalhe com os valores de atributos dos dois argumentos. Objeto e conceito formal podem ter valores de atributos iguais ou diferentes; chamam-se os primeiros de valores positivos de atributos e os últimos de valores negativos de atributos.

Uma primeira tentativa de medida de similaridade poderia ser a quantidade de atributos positivos de atributos, entre o objeto e o conceito formal. Caso se adotasse tal medida, os conceitos formais (E_1, I_1) e (E_3, I_3) seriam os mais similares ao objeto o; contudo, o primeiro desses conceitos contêm também muitos valores negativos de atributos. Uma segunda tentativa poderia, baseando-se no argumento apresentado, considerar mais similares os conceitos formais com menor quantidade de valores negativos de atributos. Com tal medida, os conceitos formais (E_2, I_2) e (E_3, I_3) seriam os mais similares ao objeto o; no entanto, o primeiro desses conceitos contém poucos valores positivos de atributos (menos até que o primeiro conceito formal, descartado por tal medida).

A função de similaridade mais adequada precisa, portanto, considerar ambos os valores positivos e negativos de atributos. O conceito formal precisa ser beneficiado por valores positivos em sua intensão, mas também penalizado por seus valores negativos de atributos. Desse modo, o conceito formal (E_3, I_3) é o mais similar ao objeto o, pois sua intensão contém mais valores positivos de atributos do que negativos, balanceados de maneira melhor do que nos demais conceitos em questão. Determina-se, portanto, que a função de similaridade do algoritmo "Similares1", entre um objeto $o \in O$ e um conceito formal (E, I), é a seguinte:

$$
\begin{array}{c}
\text{sim}(o, (E, I)) \; = \\
\text{número de valores positivos de atributos} - \\
\text{número de valores negativos de atributos}
\end{array}
$$

Uma segunda questão quanto ao algoritmo "Similares1" é: uma vez encontrados os conceitos formais mais similares ao objeto, como classificar este? De fato, é possível (e provável) que o conjunto de mais similares contenha mais que um conceito formal. Caso fosse sempre encontrado somente um conceito mais similar, o valor de classe sugerido por ele classificaria o objeto. No entanto, como o conjunto de mais similares não é garantidamente unitário, é necessário analisar os valores de classe sugeridos pelos conceitos formais mais similares.

Para classificar um objeto, o algoritmo escolhe, dentre os conceitos formais mais similares, aquele mais convicto quanto ao valor de classe. O conceito formal contém objetos em sua extensão que possuem valores de classe, naquele primeiro contexto formal empregado para construção do reticulado; essa extensão logo apresenta, segundo mesma idéia do algoritmo anterior, quantidades de votos dos objetos para cada valor de classe. Define-se aqui conceito formal mais convicto como aquele que, comparado a outros, apresenta a maior porcentagem de votos para algum valor de classe. A convicção mais alta de um conceito formal é 100%; isso acontece quando todos os objetos de sua extensão sugerem somente um valor de classe. Prefere-se, dentre os conceitos formais mais similares, aquele mais convicto, pois parece pouco adequado adotar valores de classe pouco votados em seus conceitos formais.

O algoritmo precisa, devido à adoção de tal estratégia, ter acesso aos valores de classe dos objetos que formaram o reticulado. Durante a construção de tal estrutura, na primeira fase, não foram consideradas a classe tampouco seu domínio e valores em objetos, mas agora eles são necessários. Uma solução seria, ao analisar os conceitos formais mais similares ao objeto a classificar, acessar valores de classe dos objetos no contexto formal, no entanto, esse seria um procedimento bastante oneroso. A solução mais adequada é, durante a construção do reticulado, armazenar em cada conceito formal as porcentagens de votos da extensão, para cada valor de classe.

A questão da escolha de valores de classe, solucionada por "Similares1" com a escolha do conceito formal mais convicto, pode ter soluções diferentes. Uma dessas é proposta e apresentada na próxima seção.

Uma última questão quanto ao algoritmo "Similares1" é: como percorrer o reticulado? O pseudo-código caminha por tal estrutura em níveis, usando, para isso, dois conjuntos; o primeiro contém os conceitos formais sob análise, o segundo contém os próximos a analisar – subconceitos não visitados dos conceitos sob análise. Durante essa pesquisa, também ocorrem, como já mencionado, os cálculos de similaridade e a manutenção do seu resultado mais alto. Caso o processo fosse somente esse,

o reticulado seria inteiramente percorrido, mas, como mostra o pseudo-código, os subconceitos passam por uma condição antes de serem adicionados ao conjunto de próximos conceitos formais.

O algoritmo não percorre o reticulado inteiro, pois isso é desnecessário. Existem algumas regiões de tal estrutura que, se percorridas, certamente não aumentariam a similaridade máxima já alcançada. O leitor suponha que o algoritmo encontra-se na seguinte situação: após calcular a similaridade de um conceito formal (E, I) a um objeto $o \in O''$, é preciso decidir se compensa caminhar para os subconceitos do conceito em questão. Caso esse caminho certamente não possa aumentar a similaridade máxima já alcançada, os subconceitos são ignorados como próximos conceitos formais. Para responder quando compensa avançar para os subconceitos (para o reticulado construído na primeira fase), efetua-se a seguinte análise:

$$\text{Compensa?} \begin{cases} \text{Sim}, & \text{sim}(o,(E,I)) + (|A'| - |I|) \geq \text{similaridade máxima} \\ \text{Não}, & \text{sim}(o,(E,I)) + (|A'| - |I|) < \text{similaridade máxima} \end{cases}$$

Esse raciocínio, bastante simples, diz o seguinte: caso, somando-se a similaridade calculada – $\text{sim}(o,(E,I))$ – ao número de valores de atributos restantes – $|A'| - |I|$ –, não for superada a similaridade máxima já calculada, não compensa caminhar para os subconceitos. A cada subconceito avançado, o tamanho da intensão aumenta, no mínimo, em um elemento (valores positivos ou negativos de atributos); o tamanho máximo de qualquer intensão no reticulado é $|A'|$ (valores de atributos usados na construção do reticulado). Logo, só compensa continuar se, com os valores restantes de atributos, for possível superar ou, no mínimo, alcançar a máxima similaridade calculada.

É possível melhorar mais a pesquisa no reticulado, com alguns cuidados simples. O primeiro, já mencionado em seções anteriores, consiste em marcar conceitos formais já visitados, a fim de evitar que tais estruturas seja analisadas repetidamente. Um segundo cuidado é interromper a pesquisa se for calculada uma similaridade igual ao número de valores de atributos do objeto a classificar. Quando isso acontece, foi encontrado um conceito formal cuja intensão é idêntica aos valores de atributos do objeto, logo, não é mais necessário caminhar pelo reticulado, pois nenhum outro conceito terá tal similaridade. Classifica-se o objeto segundo aquele conceito formal.

Concluindo o estudo de "Similares1", aborda-se o uso de valores negativos de atributos nos conceitos formais similares, pois, pela primeira vez, um algoritmo tolera tal situação. Permitem-se tais valores em conceitos formais considerados si-

milares, pois esses também possuem valores positivos de atributos. Contudo, é realmente perigoso adotar valores de classe de tais conceitos formais, pois eles podem ser determinados por valores negativos de atributos; o objeto seria classificado por valores de atributos que ele sequer possui (uma situação bastante indesejável). Uma solução para tal problema é tolerar esses valores negativos apenas se atendidos alguns critérios. Portanto, para qualquer conceito formal (E_1, I_1) e cada subconceito (E_2, I_2) que adicione somente valores negativos, esse último será visitado se ambos (E_1, I_1) e (E_2, I_2) sugerirem o mesmo valor de classe. Desse modo, procura-se tolerar valores negativos de atributos que não determinem valores de classe.

Exemplo 17: "Similares1" é executado para o contexto formal (O, A, A', R) convertido na Tabela 3.1, considerando-se $c =$"Lentes". A primeira fase do algoritmo produz, para $O' = \{01, 08, 11, 14, 20, 22\}$, o reticulado da Figura 4.6, ignorando a classe e seus valores, para a construção. A segunda fase do algoritmo encontra, para $O'' = \{04\}$, seus mais similares conceitos formais, com similaridade 2:

$(\{08\},$ {Juventude,Hipermetropia,Sim,Normal})
$(\{20\},$ {Presbiopia,Miopia,Sim,Normal})
$(\{08, 20\},$ {Sim,Normal})
$(\{11, 20\},$ {Miopia,Sim})

Os dois primeiros conceitos formais sugerem "Rígidas" com 100% de convicção, pois contém extensões unitárias; o terceiro conceito formal também sugere "Rígidas" com 100% de convicção; o quarto conceito formal nada sugere, pois "Nenhuma" e "Rígidas" têm 50% cada em sua extensão. O objeto "04" é, portanto, corretamente classificado como "Rígidas", pois tal valor foi sugerido com 100% de convicção. ■

Exemplo 18: "Similares1" é executado para o contexto formal (O, A, A', R) convertido na Tabela 3.1, considerando-se $c =$"Lentes". A primeira fase do algoritmo produz, para $O' = \{01, 08, 11, 14, 20, 22\}$, o reticulado da Figura 4.6, ignorando a classe e seus valores, para a construção. A segunda fase do algoritmo encontra, para $O'' = \{02\}$, um conceito formal com similaridade 2:

$(\{01\},$ {Juventude,Miopia,Não,Reduzido})

O conceito formal sugere, com 100% de convicção, classificar o objeto como "Nenhuma". Esse é erroneamente classificado, pois seu valor correto de classe seria

"Macias". Objeto e conceito formal possuem três valores positivos de atributos e somente um negativo, mas, apesar dessa similaridade, a classificação foi equivocada.
■

4.6 Similares2

O algoritmo "Similares2" é, na verdade, uma variação da proposta da seção anterior. Sua primeira fase também produz, usando o Algoritmo 3, um reticulado para o contexto formal (O', A, A', R), ignorando a classe e seus valores. Sua segunda fase também caminha pelo reticulado procurando conceitos formais segundo uma medida de similaridade e, baseando-se em tais conceitos, classifica os objetos do contexto formal (O'', A, A', R). O modo de percorrer a estrutura é idêntico (também evitando caminhos que não superem a similaridade máxima calculada, como aqueles com valores negativos determinantes) e a similaridade é também calculada da mesma maneira. A diferença entre os dois algoritmos está na escolha do valor de classe.

O algoritmo "Similares2" usa, para classificar um objeto, o valor de classe mais votado pelos conceitos formais similares. Essa estratégia, diferente de "Similares1", é similar à votação realizada por "Galois". Para classificar, não se considera somente um conceito formal que sugere com mais convicção, mas, pelo contrário, consideram-se as sugestões de todos, pois esses são conceitos com a mesma similaridade. No entanto, como alguns conceitos formais podem sugerir valores de classe com pouquíssima convicção, é adequado colher sugestões apenas daqueles que as dêem com uma convicção mínima. Essa estratégia é implementada neste trabalho de dois modos diferentes: (i) apenas conceitos formais similares com convicção superior a 50% têm suas sugestões consideradas na votação; (ii) requer-se, como em algumas propostas, um parâmetro de impureza, que aqui determina a não-convicção máxima tolerada.

Exemplo 19: "Similares2" é executado para o contexto formal (O, A, A', R) convertido na Tabela 3.1, considerando-se $c =$"Lentes". A primeira fase do algoritmo produz, para $O' = \{01, 08, 11, 14, 20, 22\}$, o reticulado da Figura 4.6, ignorando a classe e seus valores, para a construção. A segunda fase do algoritmo encontra, para $O'' = \{19\}$, seus mais similares conceitos formais, com similaridade 2:

$(\{11\},$ {Pré-presbiopia,Miopia,Sim,Reduzido})
$(\{20\},$ {Presbiopia,Miopia,Sim,Normal})
$(\{01, 11\},$ {Miopia,Reduzido})
$(\{11, 20\},$ {Miopia,Sim})

O primeiro conceito formal sugere "Nenhuma", com 100% de convicção; também com 100%, o segundo sugere "Rígidas"; "Nenhuma" é sugerida com 100% de convicção pelo terceiro conceito formal; por fim, o quarto tem 50% em "Nenhuma" e em "Rígidas". Usando ou não parâmetro de impureza (exigindo mais de 50% de convicção, no segundo caso), "Nenhuma" é mais votada (primeiro e terceiro conceitos formais) e classifica corretamente o objeto "19". ■

Exemplo 20: "Similares2" é executado para o contexto formal (O, A, A', R) convertido na Tabela 3.1, considerando-se $c =$"Lentes". A primeira fase do algoritmo produz, para $O' = \{01, 08, 11, 14, 20, 22\}$, o reticulado da Figura 4.6, ignorando a classe e seus valores, para a construção. A segunda fase do algoritmo encontra, para $O'' = \{07\}$, um conceito formal com similaridade 2:

$(\{08\},$ {Juventude,Hipermetropia,Sim,Normal})

O conceito formal mais similar sugere, com 100% de convicção, classificar como "Rígidas". Aceitando tal sugestão, o algoritmo classifica erroneamente o objeto "07", cujo valor correto de classe seria "Nenhuma". ■

Capítulo 5

Classificação com reticulados: experimentos

Este capítulo descreve experimentos com as propostas apresentadas no capítulo anterior. Enquanto o teor daquele era teórico, o conteúdo deste contém implementações e resultados práticos. A seção 5.1 descreve o cenário no qual efetuaram-se os experimentos – programas, computador, entradas, metodologia etc. As demais seções resumem os diversos experimentos efetuados: na seção 5.2, executam-se os algoritmos para diversos arquivos de repositório na internet; na seção 5.3, processa-se o método de avaliação chamado "holdout"; e na seção 5.4, o método de avaliação é a validação cruzada. A seção 5.5 debate os tempos de execução dos algoritmos.

5.1 Introdução

As seis propostas do capítulo anterior, além de dois algoritmos clássicos de classificação, foram implementadas neste trabalho, para experimentos e comparações. Os dois algoritmos clássicos são árvore de decisão e conjunto de regras, tais como aqui apresentados. Os oito algoritmos (dois do capítulo 2, seis do capítulo 4) foram escritos, de maneira fiel aos pseudo-códigos aqui mostrados, em linguagem de programação Java. Quanto às propostas que usam reticulado (ou pseudo-reticulado), tal estrutura foi armazenada em tabela "hash", para acesso mais rápido aos conceitos formais (ou nós). Esses foram escritos como classes, conectados aos seus descendentes diretos por meio de apontadores (referências, como dito em Java).

Os experimentos com os oito programas ocorreram em um computador com processador "AMD Athlon" de 2GHz, com 512MB de memória primária e com

sistema operacional "Suse" (distribuição "Linux").

Os arquivos de entrada para os oito programas, usados em trabalhos sobre classificação, foram coletados na internet [NHBM98]. Eles contêm, como mostrado a seguir, poucos objetos e atributos, se comparados a verdadeiras entradas para mineração de dados; usam-se tais arquivos de dimensões reduzidas, devido ao crescimento exponencial do reticulado em relação à entrada. Para as propostas que usam reticulado (ou pseudo-reticulado), os conjuntos de dados contidos nos arquivos foram primeiro convertidos para contextos formais, usando escalas nominais. O repositório acessado contém informações mais detalhadas sobre tais conjuntos; aqui apenas enumeram-se os mesmos (com os títulos originais):

BA: "balance scale weight and distance database";

HR: "Hayes-Roth and Hayes-Roth (1977) database";

LE: "database for fitting contact lenses";

M: "the monk's problems";

PO: "postoperative patient data";

SH: "space shuttle autolanding domain";

VO: "1984 United States congressional voting records database";

ZO: "zoo database".

Eis alguns detalhes quanto a tais conjuntos: HR está dividido em dois arquivos, um para cada fase de classificação; M está dividido em três problemas (M1, M2, M3), cada um composto por dois arquivos (o arquivo da segunda fase de classificação é idêntico nos três problemas); os atributos índice (únicos para cada objeto) são descartados neste trabalho; objetos com valores ausentes de atributos também são descartados.

Os algoritmos com parâmetro de impureza – "Rulearner", "Galois", "Similares1" – são executados com diversos valores para este (40%, 30%, 20%, 10%); "Similares2", quando sem parâmetro, requer consenso da maioria dos elementos da extensão (ou seja, no mínimo 50%), para considerar voto do conceito formal na eleição da classe. Quanto aos quatro parâmetros do algoritmo "Legal", testaram-se diversos valores para os mesmos e, em cada comparação entre os programas, usam-se os valores que causaram melhores resultados. O Anexo enumera os diversos testes de tais parâmetros – λ e γ sempre são 0.1, enquanto variam-se α e β.

5.2 Conjuntos originais

Os primeiros experimentos usam os arquivos tais como coletados no repositório. Os conjuntos HR e M (esse, dividido em três problemas) têm, como já mencionado, um arquivo para a primeira fase de classificação e outro para a segunda. Os demais conjuntos têm seus únicos arquivos usados nas duas fases de classificação. A Tabela 5.1 mostra os tamanhos – números de objetos em cada fase, números de atributos, tamanhos de domínios – dos arquivos dos conjuntos.

Tabela 5.1: Conjuntos originais.

	BA	HR	LE	M1	M2	M3	PO	SH	VO	ZO
$\|O'\|$	625	132	24	124	169	122	87	15	435	101
$\|O''\|$	625	28	24	432	432	432	87	15	435	101
$\|A\|$	5	5	5	7	7	7	9	7	17	17
$\|B'\|$	20	15	9	17	17	17	26	22	16	36
$\|C'\|$	3	3	3	2	2	2	3	2	2	7

Os oito programas foram executados para cada conjunto de dados. A Tabela 5.2 mostra a precisão de cada programa para cada conjunto[1]. São também mostrados os melhores e piores resultados para cada conjunto (considerando, primeiro, as propostas com reticulado ou pseudo-reticulado, depois, as oito propostas), bem como as precisões médias de cada algoritmo, para tais entradas.

Alguns algoritmos – "Grand", "Similares1", "Similares2", árvore de decisão, conjunto de regras – têm sempre precisão 100% para alguns conjuntos – BA, LE, SH, ZO. Esse bom desempenho já era esperado, pois tais conjuntos têm um só arquivo e tais algoritmos cobrem, sem tolerar impurezas, todos os objetos na primeira fase de classificação (seja com regras, caminhos na árvore ou conceitos formais similares). Mas os arquivos PO e VO, embora também tenham um só arquivo, não permitiram precisão 100%. Isso aconteceu por "perda de dados" durante a conversão para contexto formal; em PO, foi preciso tornar categórico um atributo numérico, enquanto em VO, os três valores ("sim", "não", "?") de cada atributo foram convertidos em dois ("sim", "não"), no contexto formal. Já para os conjuntos HR e M, que possuem dois arquivos, os algoritmos apresentam diferentes resultados.

[1]Os resultados do algoritmo árvore de decisão, para a entrada VO, não são mostrados em nenhuma das seções, pois em alguns experimentos ocorreu estouro de memória, não sendo possível concluir a execução.

Tabela 5.2: Precisões (%) com conjuntos originais. O algoritmo "Legal" recebeu os seguintes valores de parâmetros: $\alpha = 0.1$, $\beta = 0.6$, $\lambda = 0.1$, $\gamma = 0.1$. * O algoritmo executa só classificação binária. ** Estouro de memória.

	BA	HR	LE	M1	M2	M3	PO	SH	VO	ZO	Médias
Grand	100.00	46.43	100.00	91.44	78.47	87.96	83.91	100.00	50.80	100.00	83.90
Rulearner 40%	83.68	78.57	79.17	75.00	67.13	63.89	72.41	66.67	83.68	69.31	73.95
Rulearner 30%	86.72	71.43	91.67	73.15	63.89	80.56	72.41	80.00	91.49	85.15	79.65
Rulearner 20%	88.00	71.43	91.67	84.72	66.90	85.65	70.12	86.67	91.49	89.11	82.58
Rulearner 10%	97.76	53.57	100.00	86.11	70.60	85.65	85.06	100.00	89.43	97.03	86.52
Legal	*	*	*	65.74	67.13	80.09	*	100.00	76.32	*	77.86
Galois 40%	91.68	75.00	100.00	75.23	72.53	81.94	71.26	86.67	95.17	84.16	83.36
Galois 30%	94.56	60.71	100.00	81.02	74.31	83.80	71.26	86.67	94.94	82.18	82.95
Galois 20%	95.84	60.71	100.00	86.34	76.16	87.04	73.56	100.00	94.94	87.13	86.17
Galois 10%	100.00	60.71	100.00	90.28	77.55	88.89	85.06	100.00	95.40	95.05	89.29
Similares1	100.00	78.57	100.00	80.56	75.00	74.77	93.10	100.00	97.47	100.00	89.95
Similares2	100.00	82.14	100.00	83.10	76.85	72.69	93.10	100.00	97.47	100.00	90.54
Similares2 40%	100.00	82.14	100.00	83.10	76.85	72.69	93.10	100.00	97.47	100.00	90.54
Similares2 30%	100.00	82.14	100.00	83.10	76.85	72.69	93.10	100.00	97.47	100.00	90.54
Similares2 20%	100.00	82.14	100.00	83.10	76.85	72.69	93.10	100.00	97.47	100.00	90.54
Similares2 10%	100.00	82.14	100.00	83.10	76.85	72.69	93.10	100.00	97.47	100.00	90.54
Melhores	100.00	82.14	100.00	91.44	78.47	88.89	93.10	100.00	97.47	100.00	90.54
Piores	83.68	46.43	79.17	65.74	63.89	63.89	70.12	66.67	50.80	69.31	73.95
Árvore	100.00	67.86	100.00	78.00	64.12	94.44	83.91	100.00	**	100.00	87.59
Regras	100.00	64.29	100.00	82.64	63.66	86.57	90.80	100.00	100.00	100.00	88.80
Melhores	100.00	82.14	100.00	91.44	78.47	94.44	93.10	100.00	100.00	100.00	90.54
Piores	83.68	46.43	79.17	65.74	63.66	63.89	70.12	66.67	50.80	69.31	73.95

5.3 Método "holdout"

O método de avaliação chamado "holdout" consiste em dividir, em dois conjuntos disjuntos, uma coleção com valores conhecidos de classe. Usa-se o primeiro conjunto na construção do modelo de classificação, depois calcula-se a precisão desse modelo com o segundo conjunto. Algumas proporções sugeridas na divisão do conjunto são, por exemplo, 75%-25% e 50%-50% (primeira e segunda fases de classificação).

O método "holdout" é simples e possui diversas limitações. Primeiro, caso sejam poucos os objetos com valores conhecidos de classe, menos ainda serão aqueles usados na construção do modelo, devido à divisão do conjunto original em dois. Segundo, o modelo de classificação e sua precisão dependem bastante da divisão em dois conjuntos (como mostram os experimentos desta seção). Por fim, alguns valores de classe podem ser muito freqüentes em um dos arquivos, porém raros no outro.

Os primeiros experimentos "holdout" deste trabalho dividem os conjuntos na proporção 75%-25%. Para os conjuntos que já possuem dois arquivos – HR e M –, consideram-se, para essa nova divisão, aqueles com mais objetos (o da primeira fase, para HR, o da segunda fase, para M). Para evitar o último problema mencionado no parágrafo anterior, os conjuntos são divididos de diversas maneiras e, para cada uma, é registrado um experimento. Por isso são aqui mostradas quatro avaliações "holdout" 75%-25%. No experimento A, os conjuntos são divididos mantendo-se as proporções de valores de classe nos dois arquivos, enquanto nos demais experimentos, a divisão é aleatória. A Tabela 5.3 descreve os arquivos usados nesses experimentos.

Tabela 5.3: Conjuntos divididos em 75%-25%.

	BA	HR	LE	M	PO	SH	VO	ZO
$\|O'\|$	469	99	18	324	66	12	327	76
$\|O''\|$	156	33	6	108	21	3	108	25
$\|A\|$	5	5	5	7	9	7	17	17
$\|B'\|$	20	15	9	17	26	22	16	36
$\|C'\|$	3	3	3	2	3	2	2	7

A Tabela 5.4 mostra resultados do primeiro experimento "holdout" 75%-25%. "Grand" tem duas melhores precisões (LE e PO), mas também duas piores (M e VO). "Rulearner" (em seus diversos testes) tem a pior precisão média do primeiro grupo de algoritmos. "Legal" não figura entre os piores resultados tampouco entre os melhores. "Galois" (em seus diversos testes) tem três melhores precisões (HR, LE, PO) e somente uma pior (M). "Similares1" tem melhor precisão em quatro arquivos (BA, LE, M, VO) e nenhum pior resultado. "Similares2" (em seus diversos testes) conquista a melhor precisão média, com destaque para 100% em ZO.

Tabela 5.4: Precisões (%) com conjuntos 75%-25% A. O algoritmo "Legal" recebeu os seguintes valores de parâmetros: $\alpha = 0.2$, $\beta = 0.5$, $\lambda = 0.1$, $\gamma = 0.1$. * O algoritmo executa só classificação binária. ** Estouro de memória.

	BA	HR	LE	M	PO	SH	VO	ZO	Médias
Grand	60.26	70.97	80.00	16.67	76.19	66.67	55.56	91.67	64.75
Rulearner 40%	56.41	61.29	60.00	66.67	71.43	66.67	64.81	66.67	64.24
Rulearner 30%	72.44	87.10	80.00	22.22	71.43	66.67	84.26	87.50	71.45
Rulearner 20%	53.21	80.65	80.00	16.67	57.14	66.67	84.26	87.50	65.76
Rulearner 10%	51.28	74.19	80.00	16.67	66.67	66.67	86.11	87.50	66.14
Legal	*	*	*	50.00	*	66.67	82.41	*	66.36
Galois 40%	66.03	74.19	80.00	55.56	71.43	66.67	87.04	83.33	73.03
Galois 30%	60.90	87.10	80.00	33.33	71.43	66.67	85.19	83.33	70.99
Galois 20%	67.31	80.65	80.00	16.67	71.43	66.67	87.96	87.50	69.77
Galois 10%	60.90	74.19	80.00	16.67	76.19	66.67	87.96	91.67	69.28
Similares1	78.85	70.97	80.00	66.67	61.90	66.67	92.59	95.83	76.69
Similares2	52.56	61.29	80.00	66.67	71.43	66.67	89.81	100.00	73.55
Similares2 40%	58.97	80.65	80.00	66.67	71.43	66.67	89.81	100.00	76.78
Similares2 30%	63.46	83.87	80.00	66.67	76.19	66.67	89.81	100.00	78.33
Similares2 20%	68.59	77.42	80.00	66.67	76.19	66.67	88.88	100.00	78.05
Similares2 10%	68.59	77.42	80.00	66.67	76.19	66.67	89.81	100.00	78.17
Melhores	78.85	87.10	80.00	66.67	76.19	66.67	92.59	100.00	78.33
Piores	51.28	61.29	60.00	16.67	57.14	66.67	55.56	66.67	64.24
Árvore	46.15	61.29	40.00	33.33	61.90	66.67	**	91.67	57.29
Regras	49.36	77.42	80.00	66.67	66.67	66.67	92.59	91.67	73.88
Melhores	78.85	87.10	80.00	66.67	76.19	66.67	92.59	100.00	78.33
Piores	46.15	61.29	40.00	16.67	57.14	66.67	55.56	66.67	57.29

A Tabela 5.5 mostra resultados do segundo experimento "holdout" 75%-25%. Muitos dos algoritmos – "Grand", "Rulearner", "Galois", "Similares2" – têm precisões perfeitas para alguns arquivos (100% em SH), mas também resultados pífios para outros (0% em LE). "Legal" tem novamente desempenho mediano, alcançando 100% de precisão em SH, mas não saindo de 0% em M. "Similares1" tem bom desempenho, conseguindo, por exemplo, 100% de precisão para os conjuntos M e SH. A precisão média de "Rulearner" é novamente a pior entre os primeiros algoritmos, enquanto a de "Similares2" é novamente a melhor; tal algoritmo consegue 100% para três conjuntos (M, SH, ZO).

Tabela 5.5: Precisões (%) com conjuntos 75%-25% B. O algoritmo "Legal" recebeu os seguintes valores de parâmetros: $\alpha = 0.5$, $\beta = 0.1$, $\lambda = 0.1$, $\gamma = 0.1$. * O algoritmo executa só classificação binária. ** Estouro de memória.

	BA	HR	LE	M	PO	SH	VO	ZO	Médias
Grand	76.92	63.64	0.00	66.67	52.38	100.00	48.15	88.00	61.97
Rulearner 40%	84.62	66.67	0.00	66.67	61.90	66.67	53.70	84.00	60.53
Rulearner 30%	79.49	72.73	0.00	66.67	66.67	100.00	89.81	88.00	70.42
Rulearner 20%	73.72	72.73	0.00	66.67	61.90	100.00	90.74	88.00	69.22
Rulearner 10%	76.92	72.73	0.00	66.67	38.10	100.00	85.19	88.00	65.95
Legal	*	*	*	0.00	*	100.00	86.11	*	62.04
Galois 40%	92.31	48.48	33.33	66.67	66.67	100.00	90.74	80.00	72.28
Galois 30%	94.87	63.64	33.33	66.67	66.67	100.00	90.74	80.00	74.49
Galois 20%	76.92	63.64	16.67	66.67	61.90	100.00	91.67	80.00	69.68
Galois 10%	69.23	63.64	0.00	66.67	57.14	100.00	91.67	84.00	66.54
Similares1	78.21	54.55	33.33	100.00	47.62	100.00	91.67	96.00	75.17
Similares2	80.77	51.52	33.33	100.00	47.62	100.00	90.74	100.00	75.50
Similares2 40%	85.90	54.55	33.33	100.00	47.62	100.00	92.59	100.00	76.75
Similares2 30%	94.87	63.64	0.00	100.00	47.62	100.00	93.52	100.00	74.96
Similares2 20%	84.62	63.64	0.00	100.00	38.10	100.00	91.67	100.00	72.25
Similares2 10%	84.62	63.64	0.00	100.00	42.86	100.00	87.96	100.00	72.39
Melhores	94.87	72.73	33.33	100.00	66.67	100.00	93.52	100.00	76.75
Piores	69.23	48.48	0.00	0.00	38.10	66.67	48.15	80.00	60.53
Árvore	47.44	54.55	0.00	100.00	23.81	100.00	**	96.00	60.26
Regras	69.87	72.73	0.00	88.89	47.62	66.67	88.89	96.00	66.33
Melhores	94.87	72.73	33.33	100.00	66.67	100.00	93.52	100.00	76.75
Piores	47.44	48.48	0.00	0.00	23.81	66.67	48.15	80.00	60.26

A Tabela 5.6 mostra resultados do terceiro experimento "holdout" 75%-25%. "Grand" tem desempenho mediano, com piores precisões para três conjuntos (LE, SH, VO). Os resultados de "Rulearner" variam de acordo com seu parâmetro, conseguindo 100% para LE e M; tal algoritmo tem melhor precisão média, entre todos os demais, com parâmetro 30%. Já "Legal" tem pior precisão média, conseguindo 0% para M. "Galois", "Similares1" e "Similares2" têm desempenhos medianos; esses três algoritmos apresentam melhores precisões para alguns conjuntos ("Galois" para PO, "Similares1" para ZO, "Similares2" para BA e VO), mas também piores precisões ("Galois" para LE e SH, "Similares1" para SH, "Similares2" para HR e SH).

Tabela 5.6: Precisões (%) com conjuntos 75%-25% C. O algoritmo "Legal" recebeu os seguintes valores de parâmetros: $\alpha = 0.5$, $\beta = 0.1$, $\lambda = 0.1$, $\gamma = 0.1$. * O algoritmo executa só classificação binária. ** Estouro de memória.

	BA	HR	LE	M	PO	SH	VO	ZO	Médias
Grand	72.61	72.73	0.00	66.67	59.09	50.00	53.21	80.77	56.89
Rulearner 40%	80.25	75.76	0.00	83.33	86.36	50.00	86.24	57.69	64.95
Rulearner 30%	78.34	81.82	100.00	100.00	59.09	75.00	89.91	76.92	82.64
Rulearner 20%	75.80	75.76	100.00	100.00	59.09	75.00	89.91	73.08	81.08
Rulearner 10%	66.24	78.79	100.00	100.00	40.91	75.00	89.91	80.77	78.95
Legal	*	*	*	0.00	*	75.00	85.32	*	53.44
Galois 40%	81.53	72.73	50.00	33.33	90.91	50.00	92.66	69.23	67.55
Galois 30%	81.53	78.79	0.00	66.67	90.91	50.00	90.83	69.23	66.00
Galois 20%	76.43	78.79	0.00	66.67	90.91	50.00	89.91	73.08	65.72
Galois 10%	68.79	75.76	0.00	66.67	63.64	50.00	90.83	76.92	61.58
Similares1	73.25	72.73	83.33	66.67	50.00	50.00	93.58	92.31	72.73
Similares2	76.43	72.73	50.00	66.67	50.00	50.00	91.74	80.77	67.29
Similares2 40%	80.25	72.73	50.00	66.67	54.55	50.00	91.74	80.77	68.34
Similares2 30%	82.80	66.67	16.67	66.67	50.00	50.00	97.25	88.46	64.82
Similares2 20%	76.43	66.67	16.67	66.67	50.00	50.00	95.41	88.46	63.79
Similares2 10%	76.43	66.67	16.67	66.67	45.45	50.00	92.66	88.46	62.88
Melhores	82.80	81.82	100.00	100.00	90.91	75.00	97.25	92.31	82.64
Piores	66.24	66.67	0.00	0.00	40.91	50.00	53.21	57.69	53.44
Árvore	49.68	60.61	16.67	100.00	45.45	50.00	**	92.31	59.25
Regras	67.52	78.79	100.00	94.44	68.18	25.00	90.83	76.92	75.21
Melhores	82.80	81.82	100.00	100.00	90.91	75.00	97.25	92.31	82.64
Piores	49.68	60.61	0.00	0.00	40.91	25.00	53.21	57.69	53.44

O último experimento "holdout" 75%-25% é mostrado na Tabela 5.7. Destaque (negativo) para "Rulearner", com pior precisão média com parâmetro 40%; em seus diversos testes, tal algoritmo tem pior precisão para, no mínimo, um conjunto. Destaque (positivo) para "Galois", com melhor precisão média (quase 90%) com parâmetro 40%; tal algoritmo tem, em seus diversos testes, muitas das melhores precisões, em especial 100% para LE e SH.

Tabela 5.7: Precisões (%) com conjuntos 75%-25% D. O algoritmo "Legal" recebeu os seguintes valores de parâmetros: $\alpha = 0.1$, $\beta = 0.2$, $\lambda = 0.1$, $\gamma = 0.1$. * O algoritmo executa só classificação binária. ** Estouro de memória.

	BA	HR	LE	M	PO	SH	VO	ZO	Médias
Grand	67.95	81.82	66.67	94.44	47.62	33.33	67.59	88.00	68.43
Rulearner 40%	73.08	75.76	50.00	78.70	71.43	0.00	85.19	64.00	62.27
Rulearner 30%	69.23	72.73	83.33	81.48	66.67	0.00	89.81	72.00	66.91
Rulearner 20%	65.38	72.73	66.67	91.67	61.90	33.33	91.67	84.00	70.92
Rulearner 10%	64.74	87.88	66.67	94.44	47.62	33.33	93.52	88.00	72.03
Legal	*	*	*	65.74	*	66.67	72.22	*	68.21
Galois 40%	88.46	81.82	100.00	95.37	76.79	100.00	89.81	76.00	88.53
Galois 30%	75.64	81.82	100.00	85.19	76.19	33.33	91.67	76.00	77.48
Galois 20%	75.00	81.82	66.67	96.30	61.90	33.33	90.74	76.00	72.72
Galois 10%	73.08	81.82	66.67	94.44	52.38	33.33	91.67	84.00	72.17
Similares1	56.41	72.73	83.33	87.04	52.38	33.33	90.74	96.00	71.50
Similares2	58.97	72.73	83.33	82.41	47.62	33.33	90.74	96.00	70.64
Similares2 40%	67.31	72.73	83.33	81.48	57.14	33.33	90.74	96.00	72.76
Similares2 30%	69.23	75.76	100.00	84.26	52.38	66.67	89.81	96.00	79.26
Similares2 20%	69.23	75.76	83.33	85.19	47.62	66.67	90.74	96.00	76.82
Similares2 10%	69.23	75.76	83.33	85.19	52.38	66.67	90.74	96.00	77.41
Melhores	88.46	87.88	100.00	96.30	76.79	100.00	93.52	96.00	88.53
Piores	56.41	72.73	50.00	65.74	47.62	0.00	67.59	64.00	62.27
Árvore	39.74	75.76	66.67	94.44	47.62	33.33	**	96.00	64.79
Regras	51.92	84.85	66.67	100.00	52.38	33.33	94.44	88.00	71.45
Melhores	88.46	87.88	100.00	100.00	76.79	100.00	94.44	96.00	88.53
Piores	39.74	72.73	50.00	65.74	47.62	0.00	67.59	64.00	62.27

Os próximos experimentos "holdout" deste trabalho dividem os conjuntos na proporção 50%-50%. Essa divisão ocorre de maneira similar à última descrita (para "holdout" 75%-25%). Os conjuntos HR e M, embora possuam dois arquivos no repositório de origem, são novamente divididos conforme já descrito, considerando-se o maior arquivo. São novamente efetuadas diferentes divisões (de A a D); nos experimentos A, balanceiam-se as proporções dos valores de classe nos dois arquivos, enquanto, nos demais experimentos, a divisão em dois arquivos é aleatória. A Tabela 5.8 descreve as diferentes divisões dos conjuntos.

Tabela 5.8: Conjuntos divididos em 50% 50%.

	BA	HR	LE	M	PO	SH	VO	ZO
$\|O'\|$	313	66	12	216	44	8	218	51
$\|O''\|$	312	66	12	216	43	7	217	50
$\|A\|$	5	5	5	7	9	7	17	17
$\|B'\|$	20	15	9	17	26	22	16	36
$\|C'\|$	3	3	3	2	3	2	2	7

Como foi comentada cada tabela dos últimos experimentos e para não tornar cansativa a leitura, resumem-se, neste parágrafo, algumas observações sobre os experimentos "holdout" 50%-50%. A Tabela 5.9 mostra que, para o primeiro de tais experimentos, "Grand" tem pior desempenho, enquanto "Similares1" tem o melhor. Destaque para os algoritmos "Similares1" e "Similares2", com muitas das melhores precisões. No segundo experimento, mostrado na Tabela 5.10, o pior desempenho é de "Galois" e o melhor é de "Similares2". Destaque negativo para as péssimas precisões em LE, não saindo de 0% em alguns algoritmos, mas destaque positivo para as melhores precisões em M, SH, ZO, todas em "Similares1" e "Similares2". A Tabela 5.11, referente ao terceiro experimento, é similar à anterior em alguns pontos, como as baixíssimas precisões em LE. Em tal experimento, o pior resultado é de "Grand" e o melhor é de "Similares1". No último experimento, mostrado na Tabela 5.12, "Legal" tem o pior desempenho, enquanto "Galois" tem o melhor. Destaque para as precisões de 100% em ZO, com os algoritmos "Similares1" e "Similares2".

Tabela 5.9: Precisões (%) com conjuntos 50% 50% A. O algoritmo "Legal" recebeu os seguintes valores de parâmetros: $\alpha = 0.2$, $\beta = 0.5$, $\lambda = 0.1$, $\gamma = 0.1$. * O algoritmo executa só classificação binária. ** Estouro de memória.

	BA	HR	LE	M	PO	SH	VO	ZO	Médias
Grand	42.63	70.77	54.55	25.00	69.77	14.29	54.84	77.55	51.18
Rulearner 40%	62.50	50.77	54.55	75.00	69.77	57.14	66.36	65.31	62.68
Rulearner 30%	57.69	80.00	54.55	52.78	72.09	28.57	91.24	77.55	64.31
Rulearner 20%	31.41	73.85	54.55	25.00	72.09	14.29	91.24	79.59	55.25
Rulearner 10%	34.29	73.85	54.55	25.00	58.14	14.29	91.24	77.55	53.61
Legal	*	*	*	61.11	*	57.14	84.79	*	67.68
Galois 40%	64.73	64.62	54.55	63.89	72.09	57.14	90.32	73.47	67.60
Galois 30%	56.73	73.85	63.64	50.00	72.09	14.29	89.40	71.43	61.43
Galois 20%	50.00	75.38	63.64	25.00	72.09	14.29	90.78	77.55	58.59
Galois 10%	47.12	70.77	54.55	25.00	69.77	14.29	92.17	75.51	56.15
Similares1	91.84	69.23	63.64	75.00	58.14	71.43	92.63	91.84	76.72
Similares2	60.58	67.69	72.73	75.00	60.47	71.43	91.71	91.84	73.93
Similares2 40%	66.35	70.77	63.64	75.00	62.79	71.43	93.09	91.84	74.36
Similares2 30%	64.74	70.77	72.73	75.00	60.47	57.14	93.09	91.84	73.22
Similares2 20%	63.46	70.77	72.73	75.00	58.14	57.14	93.55	91.84	72.83
Similares2 10%	64.10	69.23	72.73	75.00	55.81	57.14	92.63	91.84	72.31
Melhores	91.84	80.00	72.73	75.00	72.09	71.43	93.55	91.84	76.72
Piores	31.41	50.77	54.55	25.00	55.81	14.29	54.84	65.31	51.18
Árvore	17.95	63.08	72.73	50.00	58.14	57.14	**	93.88	58.99
Regras	33.97	78.46	72.73	50.00	44.19	57.14	89.40	83.67	63.70
Melhores	91.84	80.00	72.73	75.00	72.09	71.43	93.55	93.88	76.72
Piores	17.95	50.77	54.55	25.00	44.19	14.29	54.84	65.31	51.18

Tabela 5.10: Precisões (%) com conjuntos 50% 50% B. O algoritmo "Legal" recebeu os seguintes valores de parâmetros: $\alpha = 0.2$, $\beta = 0.2$, $\lambda = 0.1$, $\gamma = 0.1$. * O algoritmo executa só classificação binária. ** Estouro de memória.

	BA	HR	LE	M	PO	SH	VO	ZO	Médias
Grand	79.49	66.67	0.00	0.00	53.49	57.14	70.05	80.00	50.86
Rulearner 40%	89.10	50.00	0.00	75.00	58.14	42.86	58.53	70.00	55.45
Rulearner 30%	85.90	62.12	0.00	25.00	58.14	42.86	72.35	74.00	52.55
Rulearner 20%	81.09	60.61	0.00	25.00	58.14	57.14	89.87	80.00	56.48
Rulearner 10%	80.77	63.64	0.00	25.00	51.16	57.14	90.78	80.00	56.06
Legal	*	*	*	50.00	*	57.14	84.33	*	63.82
Galois 40%	91.03	57.58	0.00	0.00	60.47	42.86	90.78	60.00	50.34
Galois 30%	83.33	60.61	0.00	0.00	60.47	42.86	91.24	72.00	51.31
Galois 20%	80.77	68.18	0.00	0.00	60.47	57.14	91.24	78.00	54.48
Galois 10%	76.92	71.21	0.00	0.00	58.14	57.14	92.63	82.00	54.76
Similares1	73.08	66.67	25.00	100.00	44.19	71.43	92.63	96.00	71.13
Similares2	78.85	53.03	25.00	100.00	46.51	71.43	93.09	96.00	70.49
Similares2 40%	93.59	53.03	25.00	100.00	48.84	71.43	93.55	96.00	72.68
Similares2 30%	85.90	57.58	25.00	100.00	48.84	71.43	93.55	96.00	72.29
Similares2 20%	85.90	65.15	25.00	100.00	46.51	71.43	94.01	96.00	73.00
Similares2 10%	85.90	63.64	25.00	100.00	46.51	71.43	94.01	96.00	72.81
Melhores	93.59	71.21	25.00	100.00	60.47	71.43	94.01	96.00	73.00
Piores	73.08	50.00	0.00	0.00	44.19	42.86	58.53	60.00	50.34
Árvore	49.68	62.12	25.00	100.00	51.16	57.14	**	88.00	61.87
Regras	67.95	69.70	0.00	83.33	55.81	42.86	91.24	90.00	62.61
Melhores	93.59	71.21	25.00	100.00	60.47	71.43	94.01	96.00	73.00
Piores	49.68	50.00	0.00	0.00	44.19	42.86	58.53	60.00	50.34

Tabela 5.11: Precisões (%) com conjuntos 50% 50% C. O algoritmo "Legal" recebeu os seguintes valores de parâmetros: $\alpha = 0.5$, $\beta = 0.1$, $\lambda = 0.1$, $\gamma = 0.1$. * O algoritmo executa só classificação binária. ** Estouro de memória.

	BA	HR	LE	M	PO	SH	VO	ZO	Médias
Grand	79.23	62.12	0.00	0.00	47.73	12.50	52.75	70.59	40.62
Rulearner 40%	73.16	60.61	0.00	75.00	61.36	62.50	69.27	72.55	59.31
Rulearner 30%	77.96	62.12	0.00	25.00	68.18	12.50	91.74	56.86	49.30
Rulearner 20%	77.64	72.73	0.00	25.00	56.82	62.50	91.74	60.78	55.90
Rulearner 10%	77.00	72.73	0.00	25.00	50.00	62.50	86.70	70.59	55.57
Legal	*	*	*	0.00	*	87.50	86.24	*	57.91
Galois 40%	79.23	56.06	0.00	0.00	79.55	50.00	93.58	60.78	52.40
Galois 30%	84.35	65.15	0.00	0.00	77.27	12.50	93.58	56.86	48.71
Galois 20%	81.79	66.67	0.00	0.00	63.64	12.50	92.66	68.63	48.24
Galois 10%	81.79	62.12	0.00	0.00	47.73	12.50	94.95	66.67	45.72
Similares1	72.84	65.15	25.00	100.00	54.55	62.50	94.50	90.20	70.59
Similares2	78.59	53.03	25.00	100.00	54.55	37.50	93.12	90.20	66.50
Similares2 40%	79.87	53.03	25.00	100.00	59.09	62.50	94.50	90.20	70.52
Similares2 30%	80.51	56.06	25.00	100.00	50.00	50.00	94.95	90.20	68.34
Similares2 20%	80.51	59.09	25.00	100.00	50.00	50.00	94.95	90.20	68.72
Similares2 10%	80.51	59.09	25.00	100.00	50.00	50.00	89.45	90.20	68.03
Melhores	84.35	72.73	25.00	100.00	79.55	87.50	94.95	90.20	70.59
Piores	72.84	53.03	0.00	0.00	47.73	12.50	52.75	56.86	40.62
Árvore	54.31	68.18	25.00	100.00	65.91	62.50	**	84.31	65.74
Regras	72.84	72.73	16.67	83.33	59.09	25.00	93.12	82.35	63.14
Melhores	84.35	72.73	25.00	100.00	79.55	87.50	94.95	90.20	70.59
Piores	54.31	53.03	0.00	0.00	47.73	12.50	52.75	56.86	40.62

Tabela 5.12: Precisões (%) com conjuntos 50% 50% D. O algoritmo "Legal" recebeu os seguintes valores de parâmetros: $\alpha = 0.5$, $\beta = 0.1$, $\lambda = 0.1$, $\gamma = 0.1$. * O algoritmo executa só classificação binária. ** Estouro de memória.

	BA	HR	LE	M	PO	SH	VO	ZO	Médias
Grand	69.23	59.09	50.00	96.30	58.14	57.14	65.90	86.00	67.73
Rulearner 40%	72.44	42.42	58.33	73.61	65.12	57.14	73.73	66.00	63.60
Rulearner 30%	72.44	54.55	50.00	71.76	65.12	57.14	88.02	70.00	66.13
Rulearner 20%	72.76	57.58	66.67	94.91	62.79	57.14	92.63	86.00	73.81
Rulearner 10%	63.46	62.12	58.33	95.83	58.14	85.71	87.56	86.00	74.64
Legal	*	*	*	0.00	*	85.71	83.87	*	56.53
Galois 40%	79.17	42.42	66.67	78.70	65.12	57.14	90.78	70.00	68.75
Galois 30%	78.21	51.52	66.67	86.11	65.12	85.71	89.40	80.00	75.34
Galois 20%	77.88	54.55	66.67	93.06	65.12	85.71	89.40	80.00	76.55
Galois 10%	69.55	59.09	50.00	95.83	65.12	57.14	89.40	84.00	71.27
Similares1	67.63	57.58	83.33	85.19	51.16	57.14	90.78	100.00	74.10
Similares2	68.59	42.42	75.00	82.87	55.81	85.71	90.78	100.00	75.15
Similares2 40%	75.00	48.48	66.67	82.87	58.14	57.14	91.71	100.00	72.50
Similares2 30%	72.44	48.48	75.00	85.65	55.81	57.14	91.24	100.00	73.22
Similares2 20%	72.76	46.97	75.00	86.57	51.16	57.14	90.78	100.00	72.55
Similares2 10%	72.76	53.03	66.67	86.57	51.16	57.14	91.24	100.00	72.32
Melhores	79.17	62.12	83.33	96.30	65.12	85.71	92.63	100.00	76.55
Piores	63.46	42.42	50.00	0.00	51.16	57.14	65.90	66.00	56.53
Árvore	50.96	62.12	75.00	79.63	55.81	85.71	**	92.00	71.60
Regras	59.94	56.06	66.67	86.57	51.16	57.14	92.17	92.00	70.21
Melhores	79.17	62.12	83.33	96.30	65.12	85.71	92.63	100.00	76.55
Piores	50.96	42.42	50.00	0.00	51.16	57.14	65.90	66.00	56.53

5.4 Método de validação cruzada

Esta seção mostra experimentos segundo o método chamado validação cruzada. Em tal método, divide-se, por exemplo, o conjunto de entrada em dois. Usa-se o primeiro conjunto na construção do modelo e o segundo em sua avaliação, depois trocam-se os papéis. Por fim, calcula-se a precisão média dos dois processamentos. O método descrito é a validação cruzada em duas iterações. De modo genérico, na validação cruzada em n iterações, divide-se o conjunto em n partes de tamanhos iguais, executa-se o algoritmo alternando-se os subconjuntos (a cada momento, um

dos subconjuntos é empregado na segunda fase, enquanto os demais formam a primeira) e, ao fim do processo, calcula-se a precisão média.

A validação cruzada soluciona algumas fraquezas do método "holdout" [TSK06]. Algumas de suas vantagens são o uso de mais objetos na construção do modelo (primeira fase de classificação) e também o uso de todo o conjunto, de modo alternado e sem repetição, na avaliação (segunda fase de classificação). Uma desvantagem é certamente o custo computacional de se cumprir tal método, executando diversas vezes o algoritmo de classificação.

O primeiro experimento da seção é uma validação cruzada em quatro iterações. Cada conjunto é, portanto, dividido em quatro arquivos de tamanhos iguais. O processo de divisão foi o seguinte: lê-se o arquivo original e cada quarto lido é copiado em um dos subconjuntos, alternado-se os mesmos. A Tabela 5.13 detalha a divisão dos conjuntos aqui usados em quatro partes. O leitor observe os números de objetos em cada iteração; 75% são usados na construção do modelo e 25% na avaliação do mesmo, sendo cada objeto empregado três vezes na fase de construção e somente uma vez na fase de avaliação.

Tabela 5.13: Conjuntos de validação cruzada 4. Enumeram-se números de objetos ($|O'|$ e $|O''|$) em cada iteração (1 a 4).

	BA	HR	LE	M	PO	SH	VO	ZO
$\|A\|$	5	5	5	7	9	7	17	17
$\|B'\|$	20	15	9	17	26	22	16	36
$\|C'\|$	3	3	3	2	3	2	2	7
$\|O'\|$ e $\|O''\|$	468 e 157	99 e 33	18 e 6	324 e 108	65 e 22	11 e 4	326 e 109	75 e 26
$\|O'\|$ e $\|O''\|$	469 e 156	99 e 33	18 e 6	324 e 108	65 e 22	11 e 4	326 e 109	76 e 25
$\|O'\|$ e $\|O''\|$	469 e 156	99 e 33	18 e 6	324 e 108	65 e 22	11 e 4	326 e 109	76 e 25
$\|O'\|$ e $\|O''\|$	469 e 156	99 e 33	18 e 6	324 e 108	66 e 21	12 e 3	327 e 108	76 e 25

As próximas tabelas mostram a validação cruzada para os oito programas aqui abordados. São exibidas as precisões dos algoritmos em cada iteração e para cada conjunto. Os algoritmos com parâmetro de entrada – "Rulearner", "Galois", "Similares2" – foram executados com diversos valores para esse, mas têm aqui exibidos somente os melhores resultados. O algoritmo "Legal" foi executado, para essa validação cruzada, com os seguintes valores para seus parâmetros (pois eles produziram alguns dos melhores resultados no método "holdout"): $\alpha = 0.2$, $\beta = 0.5$, $\lambda = 0.1$, $\gamma = 0.1$. É importante lembrar que, embora composta por iterações, a validação

cruzada tem seu resultado na precisão média das diversas execuções. O leitor confira, portanto, a última linha das próximas tabelas, onde são mostradas as precisões totais.

Tabela 5.14: Precisões (%) de "Grand" em validação cruzada 4.

	BA	HR	LE	M	PO	SH	VO	ZO	Médias
Iteração 1	42.04	66.67	83.33	33.33	27.27	50.00	58.72	96.15	57.19
Iteração 2	51.92	72.73	66.67	25.93	59.09	25.00	50.46	88.00	54.98
Iteração 3	56.41	66.67	66.67	29.63	68.18	25.00	47.71	84.00	55.53
Iteração 4	37.18	72.73	83.33	33.33	76.19	66.67	55.56	68.00	61.62
Médias	46.89	69.70	75.00	30.56	57.68	41.67	53.11	84.04	57.33

Tabela 5.15: Precisões (%) de "Rulearner" 30% em validação cruzada 4.

	BA	HR	LE	M	PO	SH	VO	ZO	Médias
Iteração 1	35.03	63.64	83.33	66.67	59.09	25.00	85.32	92.31	63.80
Iteração 2	71.15	84.85	66.67	35.19	72.73	0.00	91.74	76.00	62.29
Iteração 3	78.21	87.88	66.67	42.59	63.64	25.00	96.33	72.00	66.54
Iteração 4	51.92	81.82	83.33	66.67	61.90	33.33	83.33	64.00	65.79
Médias	59.08	79.55	75.00	52.78	64.34	20.83	89.18	76.08	64.60

Tabela 5.16: Precisões (%) de "Legal" em validação cruzada 4.

	M	SH	VO	Médias
Iteração 1	46.30	100.00	85.32	77.21
Iteração 2	64.81	25.00	84.40	58.07
Iteração 3	62.96	25.00	90.83	59.60
Iteração 4	48.15	0.00	81.48	43.21
Médias	55.56	37.50	85.51	59.52

Tabela 5.17: Precisões (%) de "Galois" 40% em validação cruzada 4.

	BA	HR	LE	M	PO	SH	VO	ZO	Médias
Iteração 1	50.32	60.61	83.33	44.44	68.18	25.00	92.66	92.31	64.61
Iteração 2	79.49	72.73	50.00	57.41	72.73	0.00	93.58	76.00	62.74
Iteração 3	76.28	75.76	83.33	59.26	72.73	25.00	97.25	72.00	70.20
Iteração 4	59.62	69.70	100.00	44.44	71.43	100.00	86.11	68.00	74.91
Médias	66.43	69.70	79.17	51.39	71.27	37.50	92.40	77.08	68.12

Tabela 5.18: Precisões (%) de "Similares1" em validação cruzada 4.

	BA	HR	LE	M	PO	SH	VO	ZO	Médias
Iteração 1	52.87	75.76	100.00	66.67	45.45	50.00	90.83	100.00	72.70
Iteração 2	76.28	72.73	83.33	83.33	40.91	25.00	91.74	100.00	71.67
Iteração 3	80.13	63.64	66.67	83.33	45.45	25.00	93.58	96.00	69.23
Iteração 4	70.51	66.67	83.33	66.67	66.67	33.33	91.67	84.00	70.36
Médias	69.95	69.70	83.33	75.00	49.62	33.33	91.96	95.00	70.99

Tabela 5.19: Precisões (%) de "Similares2" 40% em validação cruzada 4.

	BA	HR	LE	M	PO	SH	VO	ZO	Médias
Iteração 1	47.77	69.70	83.33	66.67	45.45	50.00	94.50	100.00	69.68
Iteração 2	63.46	72.73	66.67	83.33	54.55	25.00	95.41	100.00	70.14
Iteração 3	76.28	66.67	66.67	83.33	59.09	25.00	95.41	92.00	70.56
Iteração 4	73.72	69.70	83.33	66.67	71.43	100.00	88.89	80.00	79.22
Médias	65.31	69.70	75.00	75.00	57.63	50.00	93.55	93.00	72.40

Tabela 5.20: Precisões (%) da árvore de decisão em validação cruzada 4.

	BA	HR	LE	M	PO	SH	ZO	Médias
Iteração 1	42.04	57.58	66.67	33.33	59.09	75.00	96.15	61.41
Iteração 2	53.21	66.67	83.33	66.67	40.91	0.00	100.00	58.68
Iteração 3	48.72	57.58	66.67	66.67	45.45	50.00	92.00	61.01
Iteração 4	27.56	60.61	50.00	33.33	61.90	100.00	72.00	57.91
Médias	42.88	60.61	66.67	50.00	51.84	56.25	90.04	59.76

Tabela 5.21: Precisões (%) do conjunto de regras em validação cruzada 4.

	BA	HR	LE	M	PO	SH	VO	ZO	Médias
Iteração 1	42.04	69.70	83.33	66.67	68.18	75.00	95.41	96.15	74.56
Iteração 2	52.56	78.79	66.67	51.85	54.55	0.00	93.58	92.00	61.25
Iteração 3	53.85	75.76	66.67	51.85	50.00	50.00	94.50	92.00	66.83
Iteração 4	34.62	78.79	83.33	66.67	66.67	33.33	90.74	72.00	65.77
Médias	45.77	75.76	75.00	59.26	59.85	39.58	93.56	88.04	67.10

A Tabela 5.22 mostra os resultados dos algoritmos na validação cruzada de quatro iterações. "Grand" tem, devido a piores precisões para três conjuntos (BA, M, VO), o pior resultado de todos os algoritmos. "Rulearner" (em seus diversos testes) tem piores precisões para quatro conjuntos (LE, M, SH, ZO), mas um resultado mediano. "Legal" tem, como de costume, uma fraca precisão média. "Galois" (em seus diversos testes) apresenta melhor precisão para um conjunto (PO), mas pior para outro (M). "Similares1" tem melhores precisões para metade dos conjuntos (BA, LE, M, ZO). Contudo, é o algoritmo "Similares2" (em especial, com parâmetro 40%) que consegue a melhor precisão média de todos os algoritmos. As propostas clássicas – árvore de decisão e conjunto de regras – não apresentam piores tampouco melhores resultados médios.

Tabela 5.22: Precisões (%) em validação cruzada 4. * O algoritmo executa só classificação binária. ** Estouro de memória.

	BA	HR	LE	M1	PO	SH	VO	ZO	Médias
Grand	46.89	69.70	75.00	30.56	57.68	41.67	53.11	84.04	57.33
Rulearner 40%	50.72	71.22	62.50	75.00	67.86	31.25	81.34	65.19	63.14
Rulearner 30%	59.08	79.55	75.00	52.78	64.34	20.83	89.18	76.08	64.60
Rulearner 20%	53.30	74.25	79.17	31.48	62.18	33.33	91.25	76.12	62.63
Rulearner 10%	50.42	75.76	75.00	30.56	51.95	45.83	90.34	84.04	62.99
Legal	*	*	*	55.56	*	37.50	85.51	*	59.52
Galois 40%	66.43	69.70	79.17	51.39	71.27	37.50	92.40	77.08	68.12
Galois 30%	65.15	76.52	75.00	53.71	71.27	29.17	91.71	78.08	67.57
Galois 20%	57.14	74.25	79.17	31.48	69.00	35.42	92.17	76.08	64.34
Galois 10%	50.09	71.97	75.00	30.56	57.68	41.67	91.72	83.04	62.72
Similares1	69.95	69.70	83.33	75.00	49.62	33.33	91.96	95.00	70.99
Similares2	63.86	63.64	83.33	75.00	52.92	41.67	92.64	93.00	70.76
Similares2 40%	65.31	69.70	75.00	75.00	57.63	50.00	93.55	93.00	72.40
Similares2 30%	66.27	72.73	75.00	75.00	53.09	45.83	93.55	93.00	71.81
Similares2 20%	65.16	71.97	75.00	75.00	50.81	45.83	93.32	93.00	71.26
Similares2 10%	65.16	71.97	75.00	75.00	49.68	45.83	91.49	93.00	70.89
Melhores	69.95	79.55	83.33	75.00	71.27	50.00	93.55	95.00	72.40
Piores	46.89	63.64	62.50	30.56	49.62	20.83	53.11	65.19	57.33
Árvore	42.88	60.61	66.67	50.00	51.84	56.25	**	90.04	59.76
Regras	45.77	75.76	75.00	59.26	59.85	39.58	93.56	88.04	67.10
Melhores	69.95	79.55	83.33	75.00	71.27	56.25	93.56	95.00	72.40
Piores	42.88	60.61	62.50	30.56	49.62	20.83	53.11	65.19	57.33

A Tabela 5.23 descreve os arquivos para validação cruzada com dez iterações, o último e mais rigoroso experimento do trabalho. A descrição dessa tabela é análoga àquela dos arquivos para quatro iterações. Cada conjunto é dividido em dez arquivos; a cada iteração, nove desses subconjuntos são usados na construção do modelo e somente um em sua avaliação. É importante lembrar que cada subconjunto é usado uma vez na avaliação e nove na construção, sem repetição. Devido à divisão em dez arquivos, conjuntos menores têm poucos objetos na segunda fase de classificação, como, por exemplo, SH; em algumas iterações, existe somente um objeto, o que permite grandes variações de precisão.

Tabela 5.23: Conjuntos de validação cruzada 10. Enumeram-se números de objetos ($|O'|$ e $|O''|$) em cada iteração (1 a 10).

	BA	HR	LE	M	PO	SH	VO	ZO
$\|A\|$	5	5	5	7	9	7	17	17
$\|B'\|$	20	15	9	17	26	22	16	36
$\|C'\|$	3	3	3	2	3	2	2	7
$\|O'\|$ e $\|O''\|$	562 e 63	118 e 14	21 e 3	388 e 44	78 e 9	13 e 2	391 e 44	90 e 11
$\|O'\|$ e $\|O''\|$	562 e 63	118 e 14	21 e 3	388 e 44	78 e 9	13 e 2	391 e 44	91 e 10
$\|O'\|$ e $\|O''\|$	562 e 63	119 e 13	21 e 3	389 e 43	78 e 9	13 e 2	391 e 44	91 e 10
$\|O'\|$ e $\|O''\|$	562 e 63	119 e 13	21 e 3	389 e 43	78 e 9	13 e 2	391 e 44	91 e 10
$\|O'\|$ e $\|O''\|$	562 e 63	119 e 13	22 e 2	389 e 43	78 e 9	13 e 2	391 e 44	91 e 10
$\|O'\|$ e $\|O''\|$	563 e 62	119 e 13	22 e 2	389 e 43	78 e 9	14 e 1	392 e 43	91 e 10
$\|O'\|$ e $\|O''\|$	563 e 62	119 e 13	22 e 2	389 e 43	78 e 9	14 e 1	392 e 43	91 e 10
$\|O'\|$ e $\|O''\|$	563 e 62	119 e 13	22 e 2	389 e 43	79 e 8	14 e 1	392 e 43	91 e 10
$\|O'\|$ e $\|O''\|$	563 e 62	119 e 13	22 e 2	389 e 43	79 e 8	14 e 1	392 e 43	91 e 10
$\|O'\|$ e $\|O''\|$	563 e 62	119 e 13	22 e 2	389 e 43	79 e 8	14 e 1	392 e 43	91 e 10

Como na validação cruzada em quatro passos, as próximas tabelas mostram os resultados dos oito algoritmos, na validação cruzada em dez iterações. Mais uma vez, para as propostas que lançam mão de parâmetro de entrada, mostram-se os valores com os quais tais algoritmos tiveram melhor desempenho. O algoritmo "Legal" foi novamente executado com os seguintes valores de parâmetros: $\alpha = 0.2$, $\beta = 0.5$, $\lambda = 0.1$, $\gamma = 0.1$.

Tabela 5.24: Precisões (%) de "Grand" em validação cruzada 10.

	BA	HR	LE	M	PO	SH	VO	ZO	Médias
Iteração 1	63.49	78.57	66.67	29.55	33.33	50.00	52.27	90.91	58.10
Iteração 2	80.95	57.14	66.67	50.00	55.56	50.00	65.91	100.00	65.78
Iteração 3	63.49	69.23	100.00	58.14	44.44	100.00	34.09	90.00	69.92
Iteração 4	66.67	61.54	66.67	58.14	44.44	50.00	40.91	90.00	59.80
Iteração 5	66.67	76.92	50.00	34.88	44.44	50.00	63.64	90.00	59.57
Iteração 6	48.39	69.23	100.00	58.14	55.56	0.00	39.53	80.00	56.36
Iteração 7	67.74	53.85	100.00	58.14	77.78	100.00	62.79	90.00	76.29
Iteração 8	40.32	61.54	50.00	58.14	62.50	100.00	41.86	70.00	60.55
Iteração 9	74.19	76.92	50.00	30.23	75.00	100.00	60.47	80.00	68.35
Iteração 10	48.39	76.92	100.00	27.91	50.00	100.00	51.16	80.00	66.80
Médias	62.03	68.19	75.00	46.33	54.31	70.00	51.26	86.09	64.15

Tabela 5.25: Precisões (%) de "Rulearner" 20% em validação cruzada 10.

	BA	HR	LE	M	PO	SH	VO	ZO	Médias
Iteração 1	69.84	71.43	100.00	29.55	55.56	50.00	95.45	90.91	70.34
Iteração 2	79.37	64.29	100.00	59.09	55.56	50.00	93.18	90.00	73.94
Iteração 3	63.49	69.23	100.00	58.14	44.44	100.00	93.18	70.00	74.81
Iteração 4	53.97	69.23	66.67	58.14	66.67	0.00	84.09	90.00	61.10
Iteração 5	71.43	84.62	100.00	23.26	55.56	50.00	97.73	80.00	70.33
Iteração 6	66.13	76.92	100.00	55.81	55.56	0.00	93.02	80.00	65.93
Iteração 7	70.97	69.23	100.00	58.14	77.78	100.00	97.67	80.00	81.72
Iteração 8	64.52	76.92	50.00	58.14	62.50	100.00	90.70	70.00	71.60
Iteração 9	75.81	84.62	50.00	30.23	75.00	100.00	83.72	80.00	72.42
Iteração 10	62.90	76.92	100.00	23.26	75.00	100.00	86.05	80.00	75.52
Médias	67.84	74.34	86.67	45.38	62.36	65.00	91.48	81.09	71.77

Tabela 5.26: Precisões (%) de "Legal" em validação cruzada 10.

	M	SH	VO	Médias
Iteração 1	0.00	100.00	88.64	62.88
Iteração 2	63.64	50.00	86.36	66.67
Iteração 3	74.42	100.00	86.36	86.93
Iteração 4	74.42	50.00	72.73	65.72
Iteração 5	37.21	50.00	90.91	59.37
Iteração 6	27.91	0.00	93.02	40.31
Iteração 7	76.74	0.00	90.70	55.81
Iteração 8	76.74	0.00	90.70	55.81
Iteração 9	69.77	0.00	76.74	48.84
Iteração 10	0.00	0.00	79.07	26.36
Médias	50.09	35.00	85.52	56.87

Tabela 5.27: Precisões (%) de "Galois" 30% em validação cruzada 10.

	BA	HR	LE	M	PO	SH	VO	ZO	Médias
Iteração 1	69.84	71.43	66.67	27.27	55.56	50.00	97.73	90.91	66.18
Iteração 2	68.25	57.14	66.67	56.82	66.67	100.00	95.45	100.00	76.38
Iteração 3	73.02	76.92	100.00	58.14	88.89	100.00	90.91	70.00	82.24
Iteração 4	84.13	76.92	66.67	58.14	77.78	50.00	86.36	80.00	72.50
Iteração 5	77.78	76.92	50.00	60.47	66.67	50.00	95.45	80.00	69.66
Iteração 6	74.19	69.23	100.00	55.81	55.56	0.00	95.35	70.00	65.02
Iteração 7	75.81	61.54	100.00	58.14	77.78	100.00	97.67	90.00	82.62
Iteração 8	66.13	69.23	50.00	58.14	75.00	100.00	95.35	60.00	71.73
Iteração 9	75.81	84.62	50.00	30.23	87.50	100.00	83.72	70.00	72.74
Iteração 10	64.52	92.31	100.00	23.26	62.50	100.00	83.72	70.00	74.54
Médias	72.95	73.63	75.00	48.64	71.39	75.00	92.17	78.09	73.36

Tabela 5.28: Precisões (%) de "Similares1" em validação cruzada 10.

	BA	HR	LE	M	PO	SH	VO	ZO	Médias
Iteração 1	60.32	71.43	100.00	29.55	66.67	0.00	97.73	100.00	65.71
Iteração 2	44.44	71.43	100.00	86.36	33.33	100.00	88.64	100.00	78.03
Iteração 3	76.19	69.23	100.00	100.00	44.44	100.00	93.18	100.00	85.38
Iteração 4	66.67	61.54	66.67	100.00	66.67	50.00	86.36	100.00	74.74
Iteração 5	90.48	76.92	100.00	60.47	33.33	50.00	93.18	100.00	75.55
Iteração 6	87.10	61.54	100.00	55.81	55.56	100.00	93.02	90.00	80.38
Iteração 7	83.87	61.54	100.00	100.00	55.56	100.00	97.67	100.00	87.33
Iteração 8	79.03	61.54	50.00	100.00	25.00	100.00	90.70	80.00	73.28
Iteração 9	74.19	61.54	50.00	97.67	75.00	100.00	88.37	100.00	80.85
Iteração 10	72.58	76.92	100.00	23.26	37.50	0.00	86.05	80.00	59.54
Médias	73.49	67.36	86.67	75.31	49.31	70.00	91.49	95.00	76.08

Tabela 5.29: Precisões (%) de "Similares2" 30% em validação cruzada 10.

	BA	HR	LE	M	PO	SH	VO	ZO	Médias
Iteração 1	65.08	64.29	66.67	27.27	66.67	0.00	93.18	100.00	60.40
Iteração 2	60.32	78.57	100.00	86.36	44.44	100.00	93.18	100.00	82.86
Iteração 3	66.67	76.92	100.00	100.00	44.44	100.00	95.45	100.00	85.44
Iteração 4	79.37	61.54	100.00	100.00	55.56	50.00	90.91	100.00	79.67
Iteração 5	69.84	84.62	50.00	60.47	33.33	50.00	100.00	100.00	68.53
Iteração 6	72.58	76.92	100.00	55.81	55.56	100.00	97.67	90.00	81.07
Iteração 7	69.35	69.23	100.00	100.00	55.56	100.00	97.67	100.00	86.48
Iteração 8	70.97	69.23	50.00	100.00	50.00	100.00	93.02	80.00	76.65
Iteração 9	74.19	69.23	50.00	97.67	75.00	100.00	90.70	100.00	82.10
Iteração 10	74.19	84.62	100.00	23.26	37.50	0.00	86.05	90.00	61.95
Médias	70.26	73.52	81.67	75.08	51.81	70.00	93.78	96.00	76.51

Tabela 5.30: Precisões (%) da árvore de decisão em validação cruzada 10.

	BA	HR	LE	M	PO	SH	ZO	Médias
Iteração 1	42.86	64.29	100.00	100.00	55.56	50.00	100.00	73.24
Iteração 2	25.40	57.14	66.67	100.00	66.67	50.00	90.00	65.13
Iteração 3	49.21	53.85	100.00	100.00	66.67	100.00	100.00	81.39
Iteração 4	47.62	53.85	66.67	100.00	33.33	50.00	100.00	64.50
Iteração 5	42.86	69.23	100.00	100.00	55.56	50.00	100.00	73.95
Iteração 6	27.42	53.85	50.00	100.00	44.44	0.00	100.00	53.67
Iteração 7	40.32	46.15	100.00	100.00	55.56	100.00	100.00	77.43
Iteração 8	25.81	61.54	50.00	100.00	37.50	100.00	80.00	64.98
Iteração 9	43.55	61.54	50.00	100.00	75.00	0.00	90.00	60.01
Iteração 10	22.58	61.54	50.00	72.09	25.00	100.00	80.00	58.74
Médias	36.76	58.30	73.33	97.21	51.53	60.00	94.00	67.30

Tabela 5.31: Precisões (%) do conjunto de regras em validação cruzada 10.

	BA	HR	LE	M	PO	SH	VO	ZO	Médias
Iteração 1	60.32	71.43	66.67	100.00	55.56	50.00	97.73	72.73	71.81
Iteração 2	73.02	57.14	66.67	72.73	44.44	50.00	97.73	90.00	68.97
Iteração 3	58.73	84.62	66.67	100.00	33.33	50.00	95.45	90.00	72.35
Iteração 4	38.10	76.92	33.33	100.00	44.44	0.00	86.36	100.00	59.89
Iteração 5	66.67	76.92	50.00	100.00	33.33	0.00	97.73	90.00	64.33
Iteração 6	29.03	84.62	50.00	100.00	33.33	0.00	95.35	90.00	60.29
Iteração 7	70.97	61.54	100.00	100.00	44.44	100.00	97.67	100.00	84.33
Iteração 8	40.32	76.92	50.00	100.00	37.50	100.00	79.07	90.00	71.73
Iteração 9	70.97	84.62	50.00	100.00	87.50	0.00	88.37	80.00	70.18
Iteração 10	56.45	84.62	50.00	72.09	25.00	100.00	90.70	80.00	69.86
Médias	56.46	75.94	58.33	94.48	43.89	45.00	92.62	88.27	69.37

O último experimento – validação cruzada em dez iterações – tem seus resultados mostrados na Tabela 5.32. As propostas mais antigas de algoritmos – "Grand", "Rulearner", "Legal" – apresentam a maioria das piores precisões. "Grand" tem duas dessas (BA, VO) e um desempenho mediano. "Rulearner" (em seus diversos testes) tem piores precisões em metade dos conjuntos (HR, LE, M, ZO), mas também o melhor resultado em HR, com parâmetro 30%. "Legal" consegue, além da pior precisão em um dos conjuntos (SH), o pior resultado de todos os algoritmos. Já

"Galois" (em seus diversos testes) tem boas precisões médias (muitas acima de 70%) e melhores resultados em três conjuntos (BA, PO, SH). "Similares1" demonstra um considerável desempenho, mas a melhor precisão é novamente de "Similares2", em especial com parâmetro 30%. Esse último algoritmo (em seus diversos testes), embora apresente péssimo resultado em um dos conjuntos (PO), tem melhores precisões em quatro outros (LE, M, VO, ZO), com destaque para 96% em ZO. Quanto aos dois algoritmos restantes, apesar desses piorarem alguns dos resultados (BA, HR, LE, PO têm suas piores precisões reduzidas ainda mais), a árvore de decisão conseguiu a excelente precisão de 97% para M, melhor resultado para tal conjunto.

Tabela 5.32: Precisões (%) em validação cruzada 10. * O algoritmo executa só classificação binária. ** Estouro de memória.

	BA	HR	LE	M1	PO	SH	VO	ZO	Médias
Grand	62.03	68.19	75.00	46.33	54.31	70.00	51.26	86.09	64.15
Rulearner 40%	70.88	76.65	58.33	75.08	67.92	60.00	77.44	67.27	69.20
Rulearner 30%	75.82	78.19	68.33	41.69	64.45	65.00	89.63	82.09	70.65
Rulearner 20%	67.84	74.34	86.67	45.38	62.36	65.00	91.48	81.09	71.77
Rulearner 10%	63.65	63.65	78.33	44.23	48.47	55.00	88.75	85.09	65.90
Legal	*	*	*	50.09	*	35.00	85.52	*	56.87
Galois 40%	76.00	69.89	70.00	64.19	71.39	45.00	92.86	78.09	70.93
Galois 30%	72.95	73.63	75.00	48.64	71.39	75.00	92.17	78.09	73.36
Galois 20%	70.54	72.80	83.33	46.08	72.17	70.00	92.40	76.09	72.93
Galois 10%	65.58	72.80	75.00	45.86	55.42	70.00	92.41	82.09	69.89
Similares1	73.49	67.36	86.67	75.31	49.31	70.00	91.49	95.00	76.08
Similares2	63.89	65.16	90.00	75.08	51.81	70.00	92.17	96.00	75.51
Similares2 40%	67.37	69.78	78.33	75.08	52.92	70.00	93.55	96.00	75.38
Similares2 30%	70.26	73.52	81.67	75.08	51.81	70.00	93.78	96.00	76.51
Similares2 20%	70.41	72.75	78.33	75.08	44.86	70.00	92.86	96.00	75.04
Similares2 10%	70.41	72.75	78.33	75.78	46.11	70.00	90.80	96.00	75.02
Melhores	76.00	78.19	90.00	75.78	72.17	75.00	93.78	96.00	76.51
Piores	62.03	63.65	58.33	41.69	44.86	35.00	51.26	67.27	56.87
Árvore	36.76	58.30	73.33	97.21	51.53	60.00	**	94.00	67.30
Regras	56.46	75.94	58.33	94.48	43.89	45.00	92.62	88.27	69.37
Melhores	76.00	78.19	90.00	97.21	72.17	75.00	93.78	96.00	76.51
Piores	36.76	58.30	58.33	41.69	43.89	35.00	51.26	67.27	56.87

5.5 Tempos de execução

As tabelas a seguir comparam os tempos consumidos pelos programas aqui implementados – seis usam reticulados, enquantos dois são métodos clássicos de mineração. Embora tempos de execução dependam bastante de detalhes de implementação e do computador empregado (a configuração desse é descrita no capítulo anterior), eles são aqui mostrados para permitir a comparação entre os diversos algoritmos estudados. O leitor observe que, para auxiliar a análise, os valores são arredondados de 5 ms em 5 ms. É também importante mencionar que 0 ms representa tempo desprezível de execução. Por fim, as tabelas mostram os tempos de execução agrupados em cada fase de classificação (aprendizagem e operação).

A Tabela 5.33 mostra os tempos de execução dos algoritmos para os conjuntos originais (seus tamanhos são descritos na Tabela 5.1). Na primeira fase, quando classificação binária, os melhores tempos são do algoritmo "Legal", senão, os melhores são da árvore de decisão. Na segunda fase, o algoritmo "Rulearner", com seus reduzidos conjuntos de regras, é o mais rápido na classificação, para a grande maioria dos conjuntos.

O método "holdout" 75%-25% tem os tempos consumidos dos algoritmos (médias dos experimentos A-D) mostrados na Tabela 5.34 (os tamanhos dos arquivos estão na Tabela 5.3). Mais uma vez, na primeira fase, "Legal" e árvore de decisão têm os melhores tempos (o primeiro desses algoritmos, sempre que a classificação é binária). Quanto aos piores tempos na primeira fase, esses são, na maioria dos conjuntos, do algoritmo "Grand", devido à cara construção do pseudo-reticulado. Na segunda fase, "Rulearner" tem, mais uma vez, os melhores tempos em muitos dos conjuntos. Quanto aos piores tempos da segunda fase, esses são de "Similares1" e "Similares2".

O método "holdout" 50%-50% tem os tempos consumidos dos algoritmos (médias dos experimentos A-D) mostrados na Tabela 5.35 (os tamanhos dos arquivos estão na Tabela 5.8). Os melhores tempos na primeira fase são, mais uma vez, de "Legal" (sempre que a classificação é binária) ou árvore de decisão (em caso contrário). Já os piores pertencem a diferentes algoritmos. Devido à divisão do conjunto original em dois arquivos de tamanhos similares (50%-50%), a segunda fase requer mais tempo. "Similares1" e "Similares2" são os algoritmos mais lentos na fase de classificação, enquanto, por exemplo, "Grand" e "Rulearner", além dos métodos clássicos, têm os melhores valores.

Tabela 5.33: Tempos de execução (ms) para conjuntos originais. Na primeira fase, mostram-se o pior ($\alpha = 0$) e o melhor ($\alpha = 1$) tempos de execução de "Legal". * O algoritmo executa só classificação binária. ** Estouro de memória.

	BA	HR	LE	M1	M2	M3	PO	SH	VO	ZO
Grand	4545	185	85	2290	4835	2200	3865	60	117840	28085
Rulearner	3550	155	80	2305	4190	2335	2815	50	118595	58435
Legal (pior)	*	*	*	505	830	450	*	55	9285	
Legal (melhor)	*	*	*	15	15	10	*	5	30	
Galois	2060	190	75	850	1245	845	1525	80	14840	11770
Similares1	2080	195	80	915	1265	845	1555	80	14895	11790
Similares2	2070	195	75	915	1260	850	1545	80	14775	11775
Árvore	225	30	10	45	70	25	145	10	**	45
Regras	119365	490	15	380	2275	355	525	15	19990	340
Melhores	225	30	10	15	15	10	145	5	30	45
Piores	119365	490	85	2305	4835	2335	3865	80	118595	58435
Grand	160	5	0	115	170	115	25	0	570	10
Rulearner	15	0	0	5	5	5	0	0	5	0
Legal	*	*	*	40	35	40	*	0	185	
Galois	405	20	15	530	655	540	315	5	6880	4745
Similares1	1090	35	30	1395	2600	1480	680	15	9890	2750
Similares2	2510	55	45	3700	6115	3785	1510	20	27450	6540
Árvore	10	0	0	5	10	10	5	0	**	10
Regras	25	0	0	5	15	10	5	0	10	5
Melhores	10	0	0	5	5	5	0	0	5	0
Piores	2510	55	45	3700	6115	3785	1510	20	27450	6540

Tabela 5.34: Tempos de execução (ms) para conjuntos 75%-25%. Na primeira fase, mostram-se o pior ($\alpha = 0$) e o melhor ($\alpha = 1$) tempos de execução de "Legal". * O algoritmo executa só classificação binária. ** Estouro de memória.

	BA	HR	LE	M	PO	SH	VO	ZO
Grand	2565	135	70	9500	2175	40	46805	17990
Rulearner	2070	130	65	10620	1575	30	45830	37490
Legal (pior)	*	*	*	1600	*	50	5270	
Legal (melhor)	*	*	*	15	*	10	20	
Galois	1480	150	65	2385	995	75	9650	7635
Similares1	1495	155	65	2460	1000	75	9750	7660
Similares2	1495	155	65	2455	1000	75	9750	7665
Árvore	145	15	10	50	75	10	**	40
Regras	45065	240	15	1195	260	15	6335	170
Melhores	145	15	10	15	75	10	20	40
Piores	45065	240	70	10620	2175	75	46805	37490
Grand	20	0	0	20	0	0	25	0
Rulearner	10	0	0	0	0	0	5	0
Legal	*	*	*	15	*	0	50	
Galois	95	15	0	175	55	0	1280	710
Similares1	280	35	10	680	165	5	2485	715
Similares2	200	35	10	680	170	0	2495	715
Árvore	5	0	0	5	0	0	**	0
Regras	10	0	0	5	0	0	5	0
Melhores	5	0	0	0	0	0	5	0
Piores	280	35	10	680	170	5	2495	715

Tabela 5.35: Tempos de execução (ms) para conjuntos 50%-50%. Na primeira fase, mostram-se o pior ($\alpha = 0$) e o melhor ($\alpha = 1$) tempos de execução de "Legal". * O algoritmo executa só classificação binária. ** Estouro de memória.

	BA	HR	LE	M	PO	SH	VO	ZO
Grand	1200	115	50	3830	620	20	16850	4655
Rulearner	1310	110	30	1650	540	25	32240	11165
Legal (pior)	*	*	*	785	*	30	2455	
Legal (melhor)	*	*	*	15	*	5	15	
Galois	970	125	45	795	610	60	4875	3925
Similares1	975	130	45	815	615	60	4880	3965
Similares2	980	130	45	815	615	60	4875	3965
Árvore	75	25	5	25	55	20	**	35
Regras	11960	85	10	450	90	15	1620	75
Melhores	75	25	5	15	55	5	15	35
Piores	11960	130	50	3830	620	60	32240	11165
Grand	20	0	0	20	0	0	30	0
Rulearner	15	0	0	5	0	0	5	0
Legal	*	*	*	5	*	0	40	
Galois	150	20	5	5	85	0	1955	820
Similares1	490	45	20	660	220	5	3375	780
Similares2	330	50	20	570	220	5	3310	835
Árvore	5	5	0	5	0	0	**	0
Regras	15	0	0	5	0	0	5	0
Melhores	5	0	0	5	0	0	5	0
Piores	490	50	20	660	220	5	3375	835

Por fim, a Tabela 5.36 mostra os tempos consumidos na validação cruzada de dez passos. Os conjuntos, divididos em 90%-10%, são descritos na Tabela 5.23. A árvore de decisão cumpre a primeira fase com os melhores tempos, para a grande maioria dos conjuntos. Os piores tempos da primeira fase são divididos entre os diversos algoritmos. Já na segunda fase de classificação, algoritmos exceto "Similares1" e "Similares2" são os que mais conseguem melhores tempos.

Tabela 5.36: Tempos de execução (ms) para conjuntos 90%-10%. * O algoritmo executa só classificação binária. ** Estouro de memória.

	BA	HR	LE	M	PO	SH	VO	ZO
Grand	3670	160	75	14415	2930	50	95980	21840
Rulearner	2924	145	75	14955	2125	40	97880	45520
Legal	*	*	*	90	*	30	600	
Galois	1780	175	70	3340	1330	75	12650	9760
Similares1	1835	180	75	3400	1360	80	12705	9785
Similares2	1835	180	75	3400	1360	80	12710	9820
Árvore	225	20	10	40	115	10	**	45
Regras	84575	360	15	2780	380	15	14130	285
Melhores	225	20	10	40	115	10	600	45
Piores	84575	360	75	14955	2930	80	97880	45520
Grand	15	0	0	15	0	0	20	0
Rulearner	10	0	0	0	0	0	0	0
Legal	*	*	*	0	*	0	10	
Galois	40	10	0	100	40	0	625	405
Similares1	125	25	5	315	110	0	1185	395
Similares2	90	30	5	265	110	0	1180	390
Árvore	5	0	0	0	0	0	**	0
Regras	10	0	0	0	0	0	5	0
Melhores	5	0	0	0	0	0	0	0
Piores	125	30	5	315	110	0	1185	405

Uma rápida análise das tabelas já permite algumas conclusões quanto aos tempos demandados pelos algoritmos. Na fase de construção do modelo de classificação, os algoritmos "Legal" (dependendo do parâmetro) e árvore de decisão sempre figuram entre os melhores tempos. Contudo, o primeiro desses algoritmos não apresenta precisões aceitáveis, como já mostrado no capítulo anterior. Os algoritmos "Galois", "Similares1" e "Similares2" consomem tempos bastante similares, pois os três produzem o reticulado de conceitos. Os piores tempos da primeira fase são dos algoritmos "Grand" e "Rulearner", devido à onerosa construção do pseudo-reticulado. Já na segunda fase de classificação, os algoritmos que trabalham com regras – "Grand", "Rulearner", "Legal" – e aqueles já clássicos – árvore de decisão e conjunto de regras – são evidentemente mais rápidos na classificação de objetos. Algoritmos como "Galois", "Similares1" e "Similares2" são mais demorados devido à pesquisa pelo reticulado de conceitos, que pode ter um grande número de conceitos formais (exponencial em relação ao contexto formal).

Capítulo 6

Conclusão

Análise formal de conceitos fornece a interessante estrutura chamada reticulado de conceitos. Essa mostra, em formato de diagrama, relações entre objetos e valores de atributos. Essa propriedade motiva usar tais reticulados em, por exemplo, mineração de dados, pois tais estruturas explicitam o espaço de procura em métodos como classificação. Diversos algoritmos para classificação com reticulados (alguns, com o denominado pseudo-reticulado) são apresentados e comparados neste trabalho, além de também comparados a métodos clássicos de mineração de dados. Contudo, os benefícios do uso do reticulado de conceitos vêm acompanhados dos onerosos custos de sua construção.

Conforme já mencionado, o tamanho (número de conceitos formais) do reticulado de conceitos cresce de modo exponencial em relação à entrada. A entrada, no caso, é o contexto formal, composto por objetos, atributos, valores de atributos e relação de incidência. O tamanho do reticulado de conceitos depende mais dessa última do que dos demais conjuntos do contexto formal – objetos, atributos e valores de atributos. A relação de incidência, caso contenha, por exemplo, todas as combinações de valores de atributos (ou apenas atributos, em contextos formais mono-valorados), implica no máximo número de conceitos formais. Por outro lado, uma relação de incidência mais homogênea permite menores tamanhos de reticulados de conceitos, a despeito dos números de objetos, atributos e seus valores. Portanto, os onerosos custos para construção e manipulação de reticulados de conceitos são o grande revés dos algoritmos aqui apresentados [KO01, FN03].

As tabelas a seguir permitem um estudo das precisões dos algoritmos. Mostram-se, para cada conjunto e considerando as médias, as propostas que conseguem melhores precisões em diversos experimentos: “holdout” 75%-25%, “holdout” 50%-50%,

validação cruzada em quatro passos e em dez passos. Os resultados aqui mostrados sumariam as diversas tabelas do capítulo anterior. Em cada linha e em cada coluna, nas próximas tabelas, enumeram-se os algoritmos com melhor precisão, em determinado experimento (linha) e para determinado conjunto (coluna). As duas próximas tabelas têm também, na última linha, os melhores algoritmos mais freqüentes nos experimentos A-D.

"Similares1" e "Similares2" são os algoritmos que conseguem melhores precisões médias. O primeiro desses algoritmos tem o melhor desempenho no experimento "holdout" 50%-50%, enquanto o segundo é o melhor, segundo precisões médias, nos demais experimentos – "holdout" 75%-25% e nas validações cruzadas. "Rulearner" e "Galois" também conseguem as melhores precisões médias em alguns experimentos – "holdout" 75%-25% C-D, por exemplo. Alguns algoritmos sempre aparecem dentre os melhores para alguns conjuntos (para as duas primeiras tabelas a seguir, considera-se a última linha): "Rulearner" para HR; "Galois" para PO; "Similares2" para LE, M, VO.

Tabela 6.1: Melhores algoritmos para "holdout" 75%-25%.

	BA	HR	LE	M	PO	SH	VO	ZO	Médias
75%-25% A	Similares1	Rulearner Galois	Grand Rulearner Galois Similares1 Similares2 Regras	Rulearner Similares1 Similares2 Regras	Grand Galois Similares2	Grand Rulearner Legal Galois Similares1 Similares2 Árvore Regras	Similares1 Regras	Similares2	Similares2
75%-25% B	Galois Similares2	Rulearner Regras	Galois Similares1 Similares2	Similares1 Similares2 Árvore	Rulearner Galois	Grand Rulearner Legal Galois Similares1 Similares2 Árvore	Similares2	Similares2	Similares2
75%-25% C	Similares2	Rulearner	Rulearner Regras	Rulearner Árvore	Galois	Rulearner Legal	Similares2	Similares1 Árvore	Rulearner
75%-25% D	Galois	Rulearner	Galois Similares2	Galois Regras	Galois	Galois	Rulearner Regras	Similares1 Similares2	Galois
Melhores	Galois Similares2	Rulearner	Galois Similares2	Rulearner Similares1 Similares2 Árvore Regras	Galois	Rulearner Legal Galois	Similares2 Regras	Similares2	Similares2

Tabela 6.2: Melhores algoritmos para "holdout" 50%-50%.

	BA	HR	LE	M	PO	SH	VO	ZO	Médias
50%-50% A	Similares1	Rulearner	Similares2 Árvore Regras	Rulearner Similares1 Similares2	Rulearner Galois	Similares1 Similares2	Similares2	Similares1 Similares2 Árvore	Similares1
50%-50% B	Similares2	Galois	Similares1 Similares2 Árvore	Similares1 Similares2 Árvore	Galois	Similares1 Similares2	Similares2	Similares1 Similares2	Similares2
50%-50% C	Galois	Rulearner Regras	Similares1 Similares2 Árvore	Similares1 Similares2 Árvore	Galois	Legal	Galois Similares2	Similares1 Similares2	Similares1
50%-50% D	Galois	Rulearner Árvore	Similares1	Grand	Rulearner Galois	Rulearner Legal Galois Similares2 Árvore	Rulearner	Similares1 Similares2	Galois
Melhores	Galois	Rulearner	Similares1 Similares2 Árvore	Similares1 Similares2	Galois	Similares2	Similares2	Similares1 Similares2	Similares1

Tabela 6.3: Melhores algoritmos para validação cruzada em 4 passos.

	BA	HR	LE	M	PO	SH	VO	ZO	Médias
Cruzada 04	Similares1	Rulearner	Similares1 Similares2	Rulearner Similares1 Similares2	Galois	Similares2 Árvore	Similares2 Regras	Similares1	Similares2

Tabela 6.4: Melhores algoritmos para validação cruzada em 10 passos

	BA	HR	LE	M	PO	SH	VO	ZO	Médias
Cruzada 10	Galois	Rulearner	Similares2	Similares2 Árvore	Galois	Galois	Similares2	Similares2	Similares2

Análise formal de conceitos, com seus reticulados de conceitos, tem potencial para contribuir em métodos como classificação. Essa área tem como principais vantagens a fundamentação matemática e os diagramas de reticulados de conceitos, que, como já mencionado, explicitam o espaço de procura dos algoritmos. Os algoritmos aqui mostrados, na maioria dos experimentos, conseguem melhores desempenhos que métodos clássicos de classificação. Tais algoritmos têm algumas propriedades comuns: alguns produzem regras, enquanto outros classificam percorrendo conceitos formais; alguns usam o pseudo-reticulado, mas outros, o reticulado de conceitos; alguns dos algoritmos requerem parâmetros de entrada, enquanto outros os dispensam. O algoritmo aqui proposto, em suas versões "Similares1" e "Similares2", além de propôr uma medida de similaridade para conceitos formais, consegue excelentes precisões em muitos experimentos. Contudo, apesar do bom desempenho dos algoritmos de classificação com reticulados, eles têm o ônus do tamanho exponencial de tais estruturas. É necessário diminuir os tempos de execução desses algoritmos, preservando-se as precisões alcançadas, para que seja incentivada a aplicação da análise formal de conceitos em problemas verdadeiros de mineração de dados, onde os repositórios possuem até mesmo milhões de registros (afinal, o tamanho dos conjuntos é uma das motivações para mineração de dados).

Apêndice A

Parâmetros de "Legal"

O algoritmo "Legal" requer quatro parâmetros de entradas, cujos melhores valores sequer seus autores recomendam com precisão. Embora alguns experimentos [NN96] sugiram $\alpha = 0.5$, $\beta = 0.1$, $\lambda = 0.1$, $\gamma = 0.1$, tais valores nem sempre permitiram melhores resultados aqui, como mostram as tabelas a seguir. Estes anexos mostram diversos testes de valores para tais parâmetros.

Como são quatro parâmetros, testar todas as suas possibilidades de valor (de 0 a 1, com incremento de 0.1) demandaria quase 15 mil execuções do programa para cada conjunto... tarefa inviável. Decidiu-se, portanto, o seguinte: λ e γ recebem valores 0.1, enquanto checam-se diferentes valores para α e β. Primeiro, mantém-se $\beta = 1$ e altera-se α (de 0 a 1, com incremento de 0.1), depois, mantém-se $\alpha = 0$ e altera-se β (de 0 a 1, com incremento de 0.1). Usar os valores $\beta = 1$ e $\alpha = 0$ é equivalente a ignorar tais parâmetros, como se eles sequer existissem no algoritmo.

As tabelas a seguir mostram experimentos com diferentes valores de α e β. Tais experimentos são: conjuntos originais, "holdout" 75%-25% (A a D), "holdout" 50%-50% (A a D). O leitor observe que, a exemplo da unidade de medida adotada em precisão, as tabelas mostram os valores de parâmetro também em porcentagem. As últimas linhas de cada tabela comparam os melhores valores de α e β com $\alpha = 0.5$, $\beta = 0.1$.

Tabela A.1: Precisões (%) de "Legal" para conjuntos originais.

α (%)	β (%)	M1 (%)	M2 (%)	M3 (%)	SH (%)	VO (%)	Médias (%)
0	100	55.32	32.87	60.19	46.67	48.74	48.76
10	100	65.74	67.13	80.09	46.67	75.4	67.01
20	100	52.08	67.13	49.07	46.67	85.75	60.14
30	100	50	67.13	47.22	46.67	84.83	59.17
40	100	50	67.13	47.22	46.67	84.14	59.03
50	100	52.78	63.89	55.56	46.67	84.14	60.61
60	100	66.67	0	80.56	46.67	82.99	55.38
70	100	0	0	0	46.67	85.52	26.44
80	100	0	0	0	46.67	87.36	26.81
90	100	0	0	0	46.67	0	9.33
100	100	0	0	0	46.67	0	9.33
0	0	50.46	32.87	54.86	93.33	43.68	55.04
0	10	52.31	32.87	58.1	93.33	47.82	56.89
0	20	52.78	32.87	58.33	93.33	48.51	57.16
0	30	53.94	32.87	59.26	93.33	48.74	57.63
0	40	54.17	32.87	59.49	93.33	48.51	57.67
0	50	54.63	32.87	59.72	93.33	48.51	57.81
0	60	55.32	32.87	60.19	100	48.74	59.42
0	70	55.32	32.87	60.19	100	48.74	59.42
0	80	55.32	32.87	60.19	100	48.74	59.42
0	90	55.32	32.87	60.19	46.67	48.74	48.76
0	100	55.32	32.87	60.19	46.67	48.74	48.76
50	10	0	0	80.56	86.67	85.75	50.6
10	60	65.74	67.13	80.09	100	76.32	77.86

Tabela A.2: Precisões (%) de "Legal" para conjuntos 75%-25% A.

α (%)	β (%)	M (%)	SH (%)	VO (%)	Médias (%)
0	100	50	66.67	51.85	56.17
10	100	38.89	66.67	74.07	59.88
20	100	55.56	66.67	80.56	67.6
30	100	50	33.33	79.63	54.32
40	100	50	33.33	79.63	54.32
50	100	50	33.33	79.63	54.32
60	100	61.11	33.33	79.63	58.02
70	100	0	33.33	81.48	38.27
80	100	0	33.33	82.41	38.58
90	100	0	33.33	0	11.11
100	100	0	33.33	0	11.11
0	0	50	66.67	46.3	54.32
0	10	50	66.67	50.93	55.87
0	20	50	66.67	50.93	55.87
0	30	50	66.67	50.93	55.87
0	40	50	66.67	50.93	55.87
0	50	50	66.67	51.85	56.17
0	60	50	66.67	51.85	56.17
0	70	50	66.67	51.85	56.17
0	80	50	66.67	51.85	56.17
0	90	50	66.67	51.85	56.17
0	100	50	66.67	51.85	56.17
50	10	0	66.67	81.48	49.38
20	50	50	66.67	82.41	66.36

Tabela A.3: Precisões (%) de "Legal" para conjuntos 75%-25% B.

α (%)	β (%)	M (%)	SH (%)	VO (%)	Médias (%)
0	100	33.33	33.33	57.41	41.36
10	100	50	33.33	76.85	53.39
20	100	50	33.33	87.04	56.79
30	100	66.67	33.33	84.26	61.42
40	100	66.67	33.33	85.19	61.73
50	100	66.67	33.33	82.41	60.8
60	100	50	33.33	81.48	54.94
70	100	50	33.33	84.26	55.86
80	100	0	33.33	87.96	40.43
90	100	0	33.33	0	11.11
100	100	0	33.33	0	11.11
0	0	33.33	100	54.63	62.65
0	10	33.33	100	56.48	63.27
0	20	33.33	100	57.41	63.58
0	30	33.33	100	58.33	63.89
0	40	33.33	100	58.33	63.89
0	50	33.33	100	57.41	63.58
0	60	33.33	100	58.33	63.89
0	70	33.33	100	58.33	63.89
0	80	33.33	100	58.33	63.89
0	90	33.33	33.33	57.41	41.36
0	100	33.33	33.33	57.41	41.36
50	10	0	100	86.11	62.04
40	30	33.33	66.67	85.19	61.73

Tabela A.4: Precisões (%) de "Legal" para conjuntos 75%-25% C.

α (%)	β (%)	M (%)	SH (%)	VO (%)	Médias (%)
0	100	66.67	50	47.71	54.79
10	100	83.33	50	72.48	68.6
20	100	33.33	50	84.4	55.91
30	100	33.33	50	86.24	56.52
40	100	33.33	50	85.32	56.22
50	100	33.33	50	85.32	56.22
60	100	83.33	50	80.73	71.35
70	100	0	50	87.16	45.72
80	100	0	50	88.99	46.33
90	100	0	50	0	16.67
100	100	0	50	0	16.67
0	0	66.67	75	39.45	60.37
0	10	66.67	75	45.87	62.51
0	20	66.67	75	45.87	62.51
0	30	66.67	75	45.87	62.51
0	40	66.67	50	45.87	54.18
0	50	66.67	50	46.79	54.49
0	60	66.67	50	46.79	54.49
0	70	66.67	50	46.79	54.49
0	80	66.67	50	46.79	54.49
0	90	66.67	50	46.79	54.49
0	100	66.67	50	47.71	54.79
50	10	0	75	85.32	53.44
60	10	0	75	85.32	53.44

Tabela A.5: Precisões (%) de "Legal" para conjuntos 75%-25% D.

α (%)	β (%)	M (%)	SH (%)	VO (%)	Médias (%)
0	100	46.3	66.67	50	54.32
10	100	84.26	100	72.22	85.49
20	100	53.7	100	84.26	79.32
30	100	53.7	100	85.19	79.63
40	100	53.7	100	85.19	79.63
50	100	54.63	100	82.41	79.01
60	100	68.52	100	84.26	84.26
70	100	0	100	84.26	61.42
80	100	0	100	84.26	61.42
90	100	0	100	0	33.33
100	100	0	100	0	33.33
0	0	46.3	66.67	43.52	52.16
0	10	46.3	66.67	49.07	54.01
0	20	46.3	66.67	50	54.32
0	30	46.3	66.67	50	54.32
0	40	46.3	66.67	50	54.32
0	50	46.3	66.67	50	54.32
0	60	46.3	66.67	50	54.32
0	70	46.3	66.67	50	54.32
0	80	46.3	66.67	50	54.32
0	90	46.3	66.67	50	54.32
0	100	46.3	66.67	50	54.32
50	10	0	66.67	85.19	50.62
10	20	65.74	66.67	72.22	68.21

Tabela A.6: Precisões (%) de "Legal" para conjuntos 50%-50% A.

α (%)	β (%)	M (%)	SH (%)	VO (%)	Médias (%)
0	100	50	57.14	59.45	55.53
10	100	58.33	57.14	76.96	64.14
20	100	75	57.14	85.25	72.46
30	100	50	57.14	82.49	63.21
40	100	50	57.14	82.49	63.21
50	100	50	57.14	81.57	62.9
60	100	66.67	57.14	81.11	68.31
70	100	0	57.14	82.95	46.7
80	100	0	57.14	84.79	47.31
90	100	0	57.14	0	19.05
100	100	0	57.14	0	19.05
0	0	50	57.14	48.85	52
0	10	50	57.14	58.53	55.22
0	20	50	57.14	58.99	55.38
0	30	50	57.14	58.99	55.38
0	40	50	57.14	58.99	55.38
0	50	50	57.14	59.45	55.53
0	60	50	57.14	59.45	55.53
0	70	50	57.14	59.45	55.53
0	80	50	57.14	59.45	55.53
0	90	50	57.14	59.45	55.53
0	100	50	57.14	59.45	55.53
50	10	0	57.14	85.25	47.46
20	50	61.11	57.14	84.79	67.68

Tabela A.7: Precisões (%) de "Legal" para conjuntos 50%-50% B.

α (%)	β (%)	M (%)	SH (%)	VO (%)	Médias (%)
0	100	50	57.14	55.76	54.3
10	100	50	57.14	78.34	61.83
20	100	50	57.14	85.71	64.28
30	100	50	57.14	83.41	63.52
40	100	50	57.14	82.49	63.21
50	100	50	57.14	82.03	63.06
60	100	50	57.14	81.57	62.9
70	100	50	57.14	82.95	63.36
80	100	50	57.14	84.79	63.98
90	100	50	57.14	0	35.71
100	100	50	57.14	0	35.71
0	0	50	57.14	53.46	53.53
0	10	50	57.14	55.76	54.3
0	20	50	57.14	56.22	54.45
0	30	50	57.14	56.22	54.45
0	40	50	57.14	55.76	54.3
0	50	50	57.14	55.76	54.3
0	60	50	57.14	55.76	54.3
0	70	50	57.14	55.76	54.3
0	80	50	57.14	55.76	54.3
0	90	50	57.14	55.76	54.3
0	100	50	57.14	55.76	54.3
50	10	0	57.14	85.71	47.62
20	20	50	57.14	84.33	63.82

Tabela A.8: Precisões (%) de "Legal" para conjuntos 50%-50% C.

α (%)	β (%)	M (%)	SH (%)	VO (%)	Médias (%)
0	100	50	37.5	50.92	46.14
10	100	50	37.5	72.02	53.17
20	100	50	37.5	85.78	57.76
30	100	50	37.5	86.24	57.91
40	100	50	37.5	85.78	57.76
50	100	50	37.5	85.78	57.76
60	100	50	37.5	83.49	57
70	100	50	37.5	87.61	58.37
80	100	50	37.5	89.91	59.14
90	100	50	37.5	0	29.17
100	100	50	37.5	0	29.17
0	0	50	87.5	42.2	59.9
0	10	50	87.5	52.75	63.42
0	20	50	87.5	53.21	63.57
0	30	50	87.5	53.21	63.57
0	40	50	62.5	52.75	55.08
0	50	50	62.5	52.29	54.93
0	60	50	62.5	52.29	54.93
0	70	50	62.5	52.29	54.93
0	80	50	62.5	51.83	54.78
0	90	50	62.5	50.92	54.47
0	100	50	37.5	50.92	46.14
50	10	0	87.5	86.24	57.91
80	20	0	0	89.91	29.97

Tabela A.9: Precisões (%) de "Legal" para conjuntos 50%-50% D.

α (%)	β (%)	M (%)	SH (%)	VO (%)	Médias (%)
0	100	48.15	42.86	50.69	47.23
10	100	68.98	42.86	72.35	61.4
20	100	51.85	42.86	83.41	59.37
30	100	51.85	42.86	84.33	59.68
40	100	51.85	42.86	83.87	59.53
50	100	54.63	42.86	83.87	60.45
60	100	63.89	42.86	82.03	62.93
70	100	63.89	42.86	84.33	63.69
80	100	0	42.86	85.71	42.86
90	100	0	42.86	0	14.29
100	100	0	42.86	0	14.29
0	0	48.15	85.71	46.54	60.13
0	10	48.15	85.71	53	62.29
0	20	48.15	85.71	51.61	61.82
0	30	48.15	85.71	52.53	62.13
0	40	48.15	85.71	51.61	61.82
0	50	48.15	85.71	51.61	61.82
0	60	48.15	85.71	51.61	61.82
0	70	48.15	85.71	50.69	61.52
0	80	48.15	42.86	51.69	47.57
0	90	48.15	42.86	52.69	47.9
0	100	48.15	42.86	53.69	48.23
50	10	0	85.71	83.87	56.53
70	10	0	0	81.57	27.19

O algoritmo "Legal", dependendo do parâmetro α, insere mais ou menos conceitos formais no reticulado. Como o crescimento desse é exponencial em relação à entrada (em especial, ao conjunto incidência do contexto formal), tal parâmetro proporciona grandes diferenças de tempo, na execução do programa. As próximas tabelas mostram os tempos consumidos (em milissegundos), para diversos valores de α. O leitor observe que, quanto mais rigoroso o parâmetro α (ou seja, mais alto), menor o reticulado (o que não implica, necessariamente, em melhoria na precisão).

Tabela A.10: Tempos (ms) de "Legal" para conjuntos originais, com diferentes α.

α (%)	M1 (ms)	M2 (ms)	M3 (ms)	SH (ms)	VO (ms)
0	505	830	450	55	9285
10	145	140	145	40	1500
20	65	65	75	30	625
30	50	45	55	30	340
40	30	35	35	25	230
50	25	30	25	25	135
60	20	15	25	20	95
70	15	15	10	5	65
80	15	15	10	5	45
90	15	15	10	5	30
100	15	15	10	5	30

Tabela A.11: Tempos (ms) de "Legal" para conjuntos 75%-25%, com diferentes α.

α (%)	M (ms)	SH (ms)	VO (ms)
0	1600	50	5270
10	230	35	1090
20	110	35	495
30	70	25	280
40	50	25	195
50	45	20	120
60	30	20	90
70	15	15	60
80	15	15	40
90	15	10	20
100	15	10	20

Tabela A.12: Tempos (ms) de "Legal" para conjuntos 50%-50%, com diferentes α.

α (%)	M (ms)	SH (ms)	VO (ms)
0	785	30	2455
10	190	25	675
20	85	25	335
30	65	25	210
40	45	15	145
50	40	10	95
60	25	10	70
70	15	10	45
80	15	5	25
90	15	5	15
100	15	5	15

Referências Bibliográficas

[BFOS84] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and regression trees*. Chapman and Hall, 1984.

[CB91] P. Clark and R. Boswell. Rule induction with CN2: Some recent improvements. In *Machine learning: Proceedings of the 5th European Conference*, 1991.

[CN89] P. Clark and T. Niblett. The CN2 induction algorithm. *Machine learning*, 1989.

[Coh95] W. Cohen. Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning*, 1995.

[CR93] C. Carpineto and G. Romano. GALOIS: An order-theoretic approach to conceptual clustering. In *10th International Conference on Machine Learning*, 1993.

[CR04] C. Carpineto and G. Romano. *Concept data analysis: Theory and applications*. John Wiley and Sons, 2004.

[DP90] B. Davey and H. Priestley. *Introduction to lattices and order*. Cambridge University Press, 1990.

[DP97] T. Davenport and L. Prusak. *Information ecology - Mastering the information and knowledge environment*. Oxford University Press, 1997.

[FFNN04] H. Fu, H. Fu, P. Njiwoua, and E. Nguifo. A comparative study of FCA-based supervised classification algorithms. In *2nd International Conference on Formal Concept Analysis*, 2004.

[FN03] H. Fu and E. Nguifo. How well go lattice algorithms on currently used machine learning testbeds? In *1st International Conference on Formal Concept Analysis*, 2003.

[FPSS96] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, 1996.

[GW96] B. Ganter and R. Wille. *Formal concept analysis: Mathematical foundations*. Springer Verlag, 1996.

[HK06] J. Han and M. Kamber. *Data mining - Concepts and techniques*. Morgan Kaufmann, 2 edition, 2006.

[Kas80] G. Kass. An explanatory technique for investigating large quantities of categorical data. In *Applied statistics*. 1980.

[KO01] S. Kuznetsov and S. Obiedkov. Comparing performance of algorithms for generating concept lattices. In *ICCS workshop on Concept Lattices for Knowledge Discovery in Databases*, 2001.

[LN90] M. Liquière and E. Nguifo. LEGAL: un système d'apprentissage de concepts à partir d'exemples. *Journées Françaises sur l'Apprentissage*, 1990.

[MMHL86] R. Michalski, I. Mozetic, J. Hong, and N. Lavrac. The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. In *Proceedings of the 5th National Conference on Artificial Intelligence*, 1986.

[Ngu94] E. Nguifo. Galois lattice: A framework for concept learning. In *6th International Conference on Tools with Artificial Intelligence*, 1994.

[NHBM98] D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[NN96] P. Njiwoua and E. Nguifo. Back from experimentation : A study of learning bias in LEGAL. In *Benelux Conference on Machine Learning*, 1996.

[NN05] E. Nguifo and P. Njiwoua. Treillis de concepts et classification supervisée. *Techniques et Sciences Informatiques*, 2005.

[Oos88] G. Oosthuizen. *The use of a lattice in knowledge processing*. PhD thesis, University of Strathclyde, Escócia, 1988.

[Oos94] G. Oosthuizen. The application of concept lattices to machine learning. Technical report, University of Petroria, África do Sul, 1994.

[Pri05] U. Priss. Formal concept analysis in information science. *Annual Review of Information Science and Technology*, 2005.

[Qui79] J. Quinlan. Discovering rules by induction from large collection of examples. In Edinburgh University Press, editor, *Expert systems in the micro electronic age*. 1979.

[Qui93] J. Quinlan. *C4.5: Programs for machine learning*. Morgan-Kaufmann Publishers, 1993.

[Sah95] M. Sahami. Learning classification rules using lattices. In *8th European Conference on Machine Learning*, 1995.

[Stu03] G. Stumme. Conceptual knowledge discovery and processing. *International Journal of Human and Computer Studies*, 2003.

[SWW98] G. Stumme, R. Wille, and U. Wille. Conceptual knowledge discovery in databases using formal concept analysis methods. In *2nd European Symposium on Principles of Data Mining and Knowledge Discovery*, 1998.

[TSK06] P. Tan, M. Steinbach, and V. Kumar. *Introduction to data mining*. Addison Wesley, 2006.

[WF05] I. Witten and E. Frank. *Data mining - Practical machine learning tools and techniques*. Morgan Kaufmann, 2 edition, 2005.

[Wil01] R. Wille. Why can concept lattices support knowledge discovery in databases? In *Concept Lattices Based Knowledge Discovery in Databases Workshop*, 2001.