EUCLIDES PERES FARIAS JUNIOR

ESTUDO COMPARATIVO ENTRE ALGORITMOS DE REGRAS DE ASSOCIAÇÃO DE FORMA NORMAL E INCREMENTAL DE DADOS

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática.

CURITIBA

2008

EUCLIDES PERES FARIAS JUNIOR

ESTUDO COMPARATIVO ENTRE ALGORITMOS DE REGRAS DE ASSOCIAÇÃO DE FORMA NORMAL E INCREMENTAL DE DADOS

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática.

Área de Concentração: *Ciência da Computação*

Orientador: Prof. Dr. Júlio Cesar Nievola

CURITIBA

2008

Dedicatória

À minha esposa Lucimari, minhas filhas Mariana e Joana pela compreensão e carinho, aos meus irmãos e irmãs, ao IAM (Instituto de Assistência ao Menor) que me formou para a vida e principalmente aos meus pais, que dentro das possibilidades, souberam me ensinar a viver e dedicaram-se o máximo para que eu obtivesse sucesso na vida.

Agradecimentos

A Deus por ter me direcionado os rumos e as oportunidades certas na vida. A minha família, esposa e filhas, pela compreensão, suporte e amor dedicado em todos os momentos da minha vida. Ao meu orientador, Professor Dr. Júlio Cesar Nievola, pela paciência, dedicação, incentivo e principalmente respeito sobre tudo o que produzi em prol do desenvolvimento deste trabalho. Aos meus amigos de profissão: Sr. José Rodrigues Filho, Marcelo Gabardo, Vinícius Martins, Lucilaine Pascucci, Heversom Valle e Marcelo Rea. Aos meus amigos Mestres e Mestrando: André Gustavo Uchôa, Andréia Marini, Helyane Borges, Luiz Gustavo Senko e Edson Kageyama pelas palavras de incentivo, pelos auxílios dispendido e coleguismo em todos os momentos. E todos aqueles que de alguma forma contribuíram para que esse trabalho fosse realizado com sucesso.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

Err($X \rightarrow Y$)	Erro da Regra de Associação
Acc()	Precisão da Regra
NegRel()	Confiança Negativa
Sens()	Sensitividade
Spec()	Especificidade
Cov()	Cobertura
Nov()	Novidade
Sat()	Satisfação
RSens()	Sensitividade Relativa
RSpec()	Especificidade Relativa
RNegRel()	Confiança Negativa Relativa
WRAcc()	Precisão Relativa com Peso
WRNegRel()	Confiança Negativa Relativa com Peso
WRSpec()	Especificidade Relativa com Peso
todb	*Total de registros da base de dados*
f_x	Freqüência relativa
P(X)	Probabilidade de X
|X|	Cadinalidade de X
+,-	Classes de Regras
$\delta(X)$	Comprimento de L(X)
N	Numero de elementos
C_k	Conjunto de Itens candidatos
L_k	Conjunto de Itens freqüentes
Δ	Taxa de variação
$A \in B$	A pertence a B
$\emptyset$	Conjunto Vazio
$X \cup Y$	X união com Y
$X \cup Y$	X união com Y
$X \cap Y$	X intersecção Y
$X \supset Y$	X contém Y
T_{ID}	Identificador único da transação
$t \subset I$	Conjunto de Item
D	Conjunto de transações
I	Conjunto de atributos binários
X_i , Y_j	Elementos Variando em i, j
minsup	Suporte mínimo
Conf	Confiança
Sup	Suporte
T	Tabela

$X \rightarrow Y$	Regras de Associação
X, Y	Elementos
i, j, k, q, p	Variáveis arbitrárias

Lista de Abreviaturas

KDD	*Knowledge Discovery in Database*
MD	*Mineração de Dados*
IA	*Inteligência Artificial*
AM	*Aprendizagem de Máquina*
MID	*Mineração Incremental de Dados*
E/S	o *Entrada e Saída*
tidlist	*Conjunto de Itens da Lista*
Hash	*Sumário de dados*
ANS	*Agência Nacional de Saúde Suplementar*
ASCII	*American Standard Code for Infrmation Interchange*

Resumo

Atualmente, o maior desafio da Tecnologia da Informação é vislumbrar mecanismos de exploração de dados de forma a apresentar uma análise detalhada com o objetivo de detectar novas descobertas de negócios ou tendências de mercado. Agregadas a esta necessidade, surgiram diversas técnicas computacionais que se propõem a efetuar este trabalho, denominadas Técnicas de Mineração de Dados. Dentre as técnicas de mineração de dados destacam-se as Regras de Associação, definidas como padrões descritivos representando a probabilidade de um conjunto de itens aparecerem em uma transação. Nesta dissertação foram explorados dois algoritmos de regras de associação, o algoritmo *APriori*, aplicado para a extração de regras de associação de forma convencional e o algoritmo *ZigZag*, utilizado para extração de regras de associação de forma incremental, cujo principal objetivo é efetuar um estudo comparativo entre estes algoritmos bem como a sua forma de utilização. Ambos os algoritmos, foram aplicados em duas bases de dados de uma empresa de planos de saúde de forma a demonstrar uma comparação dos resultados, medindo tempo, quantidade e qualidade das regras extraídas de duas bases de dados. Em todas as extrações, o algoritmo *ZigZag* mostrou-se eficaz em seu desempenho, em comparação às aplicações do algoritmo *APriori*. Entretanto a análise dos resultados não é uma tarefa trivial, tendendo a resultados subjetivos. Para isto, foram feitos diversos trabalhos de interpretação de resultados bem como aplicadas medidas de avaliação das regras, a fim de se obter a melhor qualidade dentre as regras extraídas de seus respectivos algoritmos e bases de dados.

Palavras-Chave: Regras de Associação, Mineração Incremental de Regras de Associação, Descoberta do Conhecimento.

Abstract

Now a days the biggest challenge to the Information Technology is to catch a glimpse of the data mining exploration mechanisms to present a detailed analysis with the objective to detect new discoveries in business or market tendencies. Aggregated to this necessity has surged several technicians proposing to realize this job, called Data Mining Technician. Among those, the data mining detach the Rules Association, defined as descriptive standards representing the probability of a item set to appear in a transaction. In this dissertation were explored two association rules algorithms, the APriori, applied to rules extraction in conventional way and the ZigZag algorithm, utilized to extract rules association in incremental way, which the main objective is to effectuate a comparative research between these algorithms as well as its utilization. Both was applied in two data base in health insurance company to demonstrate a comparison of the results, measuring time, amount and quality of the extracted rules from both data base. In all algorithms extractions the ZigZag has showed it self effective in its performance, in comparison to the APriori applications. However the results analysis is not a trivial duty, tending to subjective results. To this, was done many jobs of interpretation of the results as well applied measurements of the rules evaluations with the purpose to obtain the best quality within the extracted rules and its respective algorithms and data bases.

Keywords: Rules Association, Incremental Mining of Rules Association, Knowledge Discovery in Databases.

Capítulo I

Introdução

Este trabalho apresenta uma comparação entre algoritmos de Regras de Associação, um de forma não incremental e outra incremental em duas bases distintas, apresentando uma análise do ponto de vista de desempenho, qualidade e quantidade de regras geradas destes dois algoritmos.

1.1. Motivação

O avanço computacional consolidou-se nos tempos de hoje, como um marco na evolução humana, dispondo de recursos cada vez mais sofisticados e eficientes, capazes de produzir e gerar resultados expressivos a todos os segmentos: empresariais, sociais, científico e acadêmico. Em contribuição a isto, surgiu a necessidade de se manipular as informações, que possuem um papel vital e indispensável para a gestão e competitividade empresarial, podendo se consolidar como um ativo de um elevado grau de importância neste enquadramento.

Entretanto, toda esta gama de informação está alocada em grandes quantidades de dados que partindo deste princípio, propõe um grande e cada vez mais evoluído processo denominado Descoberta do Conhecimento em Base de Dados ou (*KDD*), expressa através de uma clássica definição de (FAYYAD, 1996), como: “Extração de conhecimento de base de dados, é o processo de identificação de padrões válidos, potencialmente úteis e compreensíveis embutidos nos dados”.

Como apoio e centro dos processos de *KDD*, destacam-se o processo de *Data Mining* ou Mineração de Dados (MD), definida como uma área multidisciplinar que incorpora técnicas utilizadas em diversas áreas como Inteligência Artificial (IA) e Aprendizagem de Máquina (AM) [MON03]. Com o objetivo de descobrir a relação entre produtos, efetuar classificação de consumidores, processo de previsão de vendas, definir potencialidades de lucros em novos mercados de trabalho ou novas filiais, tomadas de decisão e demais aplicações úteis em todos os segmentos que estejam ao alcance da manipulação de dados.

A MD é o principal processo do *KDD*, porém não é o último, pois todo o processo é desenvolvido de forma interativa e freqüente podendo ser refinado até que se possam extrair valiosos padrões. À primeira vista, estes processos parecem indicar uma forma de hierarquia, pois apresentam algo que começa em instâncias elementares, mesmo representando um volume de dados elevado e termina em um ponto bem concentrado, porém de forma extremamente valiosa para quem o utiliza.

A mineração de dados contém diversas tarefas e métodos, são elas: Classificação, Regressão, Previsão, Clusterização[1] e Associação que é a tarefa tema desta dissertação. Com o sucesso reconhecido, a MD vem nos últimos tempos, sofrendo diversificações e evoluções significativas para que a sua utilização seja cada vez mais disseminada e, é com maior freqüência, encontrado nos sistemas disponíveis em diversos segmentos.

Uma das evoluções da MD é a Mineração Incremental de Dados (MID), que surgiu da deficiência dos algoritmos convencionais de regras de associação em ter que minerar todos os dados constantes em uma base, demandando com isto um custo elevadíssimo de tempo e processamento em bases relativamente grandes tornando-se, na maioria das vezes, inviável a sua utilização.

A aplicação dos algoritmos incrementais é um método alternativo tendo em vista que a sua utilização comprovou significativa melhora na redução do custo computacional, que é considerado de grande valor para a mineração de dados, produzindo assim, resultados acima do esperado devido à sua dimensão de carga de dados a ser minerada.

Por outro lado, os algoritmos convencionais, tal como o *APriori*, são de grande eficiência com um reconhecimento notório nas tarefas de Mineração de Dados para Regras de Associação, devido à sua maturidade e referências de sucesso em diversos segmentos de mercado e publicações científicas. Com isto, é apresentado neste trabalho um estudo

[1]*Custerização* - Vem de *Cluster: significa aglomeração, quantidade de agrupamento, muito utilizada nas bibliografias de MD como "agrupamento".*

comparativo entre algoritmos de Regras de Associação *APriori* (Regra de Associação convencional), baseado na implementação de (AGRAWAL et al., 1996) e o algoritmo *ZigZag* (Regras de Associação de forma Incremental), também conhecido com um limiar positivo, para construir de maneira incremental uma grade de conjunto de itens freqüentes, como mineração incremental de dados[VEL02].

O objetivo da utilização desta base de dados empregada neste trabalho, é a identificação de possíveis trocas entre profissionais da área de saúde, para futuras investigações da real necessidade dos procedimentos médicos/hospitalares que são gerados por esta prática.

É fato comum *perceber* que existe indicação/encaminhamento de um profissional para outro profissional da área de saúde. Entretanto, existe uma dificuldade de apontar e medir a proporção de eventos advindos desta prática. O objetivo da aplicação dos algoritmos neste trabalho é encontrar qualquer associação com informações até o momento desconhecidas pela empresa, de forma a contribuir para uma possível ação no sentido de melhorar a gestão dos procedimentos praticados pelos profissionais deste segmento ou determinar uma nova sistemática, baseada nas informações novas aqui descoberta.

1.2. Objetivo Principal

O objetivo principal deste trabalho é comparar e analisar as técnicas dos algoritmos de Regras de Associação, aplicadas de forma convencional e outro incremental em duas bases de dados com tamanhos distintos. Com isto, pretende-se avaliar a quantidade e qualidade das regras extraídas de ambos os algoritmos.

1.3. Objetivos Específicos

- Analisar o comportamento do Algoritmo *APriori* (*Não Incremental*) nas duas bases de dados;
- Analisar o comportamento do Algoritmo *ZigZag* (*Incremental*) nas duas bases de dados;
- Efetuar uma comparação das regras extraídas entre os algoritmos;
- Avaliar os resultados obtidos através das técnicas de medidas de avaliação das regras de associação;
- Avaliar a qualidade das regras extraída pelos algoritmos;

1.4. Organização do Trabalho

Este trabalho está organizado em sete capítulos. O Capítulo II apresenta uma introdução à Mineração de Dados e as principais tarefas relacionadas bem como suas características. O capítulo III descreve os principais Fundamentos de Regras de Associação e os algoritmos empregados neste trabalho. O Capítulo IV apresenta as Medidas de Avaliação de Regras, o Capítulo V, a Metodologia proposta, o Capítulo VI apresenta os experimentos, a interpretação dos resultados e a aplicação das Medidas de Avaliação de Regras. O Capítulo VII descreve as Conclusões e Sugestões para Trabalhos Futuros.

Capítulo II

Introdução à Mineração de Dados ou (*KDD*)

Neste capítulo são apresentados os principais fundamentos da Descoberta de Conhecimento em Banco de Dados (*Knowledge Discovery in Database) ou (KDD*) bem como uma breve descrição de suas etapas, dando ênfase à mineração de dados e suas tarefas.

2.1. Descoberta do Conhecimento em Banco de Dados

A constante evolução dos mecanismos computacionais, bem como a evolução dos mecanismos de extração de informação, sugere hoje em dia um processo mais elaborado em suas aplicações computacionais. Para isto, usa-se o *(KDD)*, que surgiu devido ao grande volume de dados armazenados em bases de dados, guardando uma quantidade significativa de informações complexas e sem utilização. A habilidade de manipulação e entendimento dos dados ainda corre contra o tempo, no sentido de superar a quantidade em que se armazenam estes dados, pois ainda não se tem à prática de técnicas de aprendizagem e descoberta do conhecimento em grandes bases de dados.

Através disto, uma definição clássica é descrita como: "Extração de Conhecimento de Bases de Dados é o processo de identificação de padrões válidos, novos, potencialmente úteis e compreensíveis embutidos nos dados" (FAYYAD et al., 1996b).

A descoberta do conhecimento é utilizada em diversas áreas de aplicação tais como: bancária, científica, médica, comercial, engenharia, financeira, gerencial, *Internet*, segurança e etc.

O processo de *KDD* é composto por etapas que são apresentadas na Figura 2.1.:

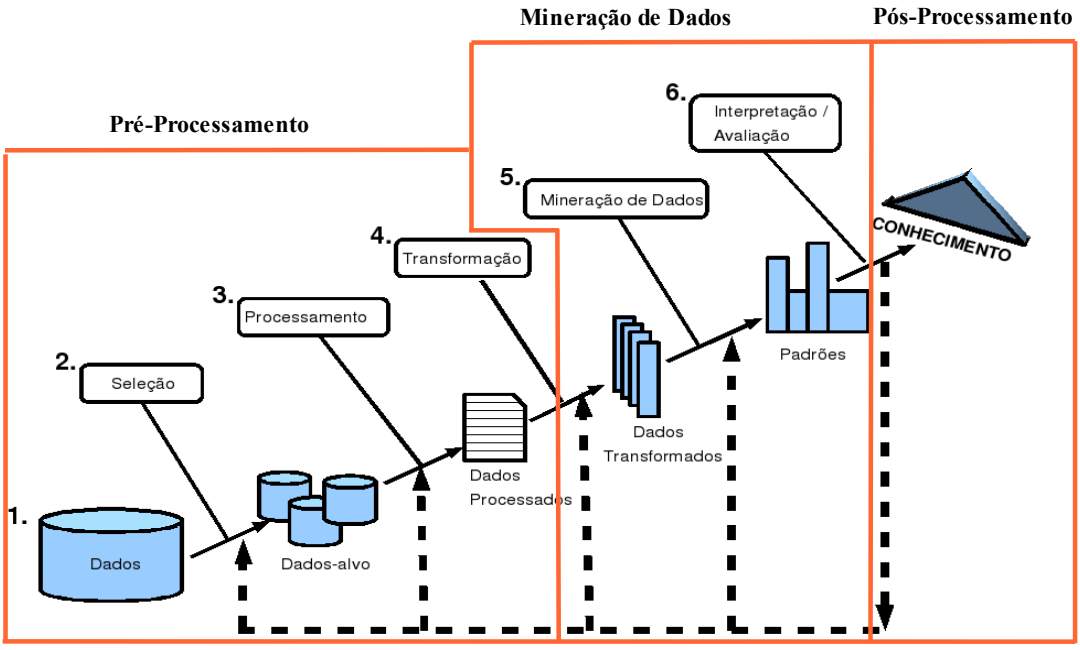


Figura 2.1 – Etapas do Processo de Descoberta do Conhecimento [FAY96].

2.1.1. Etapas do Processo de Descoberta do Conhecimento

O processo de *KDD* é definido por um conjunto de etapas, que as envolvem de forma a atender o domínio da aplicação até a interpretação dos resultados, conforme demonstrado na Figura 2.1, divido em três fases, Pré-processamento, Mineração de Dados e o Pós-processamento completando assim, as etapas do processo de *KDD*. Este processo é Interativo e Iterativo, ou seja, em qualquer momento o processo sofre uma intervenção por parte do engenheiro do conhecimento para efetuar a execução dos procedimentos como a seleção e a limpeza dos dados, objetivando os dados expressivos, envolvendo muitos passos e possíveis decisões a serem tomadas pelo especialista ou engenheiro [WON02]. Com isto, estes processos demonstram que cada etapa é fundamental para que se possam alcançar os objetivos estabelecidos, desde que utilizadas corretamente. Estas fases são divididas em etapas como segue:

1. Consolidação dos dados: Consiste em determinar uma lista preliminar de atributos, consolidando dados em base de dados de trabalho (fontes internas ou externas), eliminando ou estimando valores faltantes, removendo os *outliers* (exceções óbvias) e etc.;
2. Seleção: Consiste na escolha e seleção dos dados necessários à análise do problema em questão. Em geral, trata-se de dados operacionais das empresas,

dificultando assim a sua escolha. Caso existam diferentes estruturas e formatos de dados de uma mesma empresa, gera mais um obstáculo aplicado à esta etapa;

3. Pré-processamento: Nesta fase o objetivo é eliminar as inconsistências entre os dados selecionados, proporcionando uma correção prévia de possíveis erros na análise dos dados;
4. Transformação: Tem a função de reduzir o número de amostras, atributos, intervalos de atributos e normalizá-los;
5. Mineração de dados ou *Data Mining*: É a principal fase do *KDD*. Nesta fase, pode-se aplicar diversas técnicas de *DM*, tais como: Regras de Associação, Redes Neurais, Algoritmos Genéticos, Agrupamentos, Árvore de Decisão, e outras. Com o objetivo de extrair padrões, leva-se em conta que a escolha da técnica a ser utilizada em uma determinada análise depende muito das tarefas que se pretende realizar, tais como: características que os dados se apresentam, estimativa do significado através de formas estatísticas, resultados obtidos e também dos recursos tecnológicos disponíveis;
6. Avaliação: Através da produção dos resultados obtidos nas etapas anteriores, pode-se efetuar as interpretações e conclusões, procurando assim a melhor qualidade dos padrões extraídos, verificando a facilidade de interpretação do conhecimento obtido, de forma útil para a tomada de decisão e principalmente um resultado adequado na extração do conhecimento.

2.1.2. Tarefas de Mineração de Dados

O enriquecimento da descoberta do conhecimento está sem dúvida nenhuma, em todos estes conjuntos de etapas e tarefas objetivando dispor de recursos e técnicas avançadas para extrair o máximo conhecimento da melhor forma aliado à diversas técnicas e ferramentas disponíveis. Desta forma, dentre as principais tarefas envolvidas pelo processo de descoberta do conhecimento estão a associação, classificação e o agrupamento.

- **Associação:** Esta tarefa tem como objetivo principal encontrar, a partir de um conjunto de exemplos, um conjunto de regras de associação, isto é, através da freqüência dos atributos associados aos exemplos pode-se descobrir quais estão mais frequentemente associadas [FAY96].

Como particularidade desta tarefa, aplica-se um tipo de dado denominado "cesta de mercado" ou "*basket data*", especialmente conhecido, onde cada registro consiste de um conjunto de atributos denominados itens. Normalmente estes itens estão na forma binária em sua aplicação original. A partir daí, cada tupla corresponde a uma transação, onde um item pode assumir um valor verdadeiro ou falso, de acordo com sua presença ou não na transação.

- **Classificação:** Esta tarefa é muito conhecida devido a sua resolução comumente efetuada através de técnicas de mineração de dados. Isto é, sua utilização serve para prever a classe de um objeto [FAY96].

  Todos os dados utilizados na resolução desta tarefa consistem em um conjunto de atributos denominados previsores, de um atributo especial denominado meta, definindo assim a classe a que este registro pertence. A utilização de classificador é baseada na descoberta de um relacionamento, entre os atributos previsores e o atributo meta, utilizando os registros cuja classe é conhecida, para que após estes atributos previsores tenham condições de ser utilizado na previsão de uma classe de um registro cuja classe é desconhecida.

  Na utilização dos classificadores, todos os exemplos que ficam disponíveis para a geração de um modelo de classificação, são divididos em dois conjuntos exclusivos: treinamento e teste. No classificador o atributo meta do conjunto de teste fica indisponível. Após a previsão das classes dos exemplos do conjunto de teste, as classes previstas são então comparadas com as classes reais dos exemplos, definidas pelo atributo meta. Caso a classe prevista seja igual a classe real, a previsão foi correta, caso contrário a previsão está errada.

  O objetivo principal de uma tarefa de classificação é a maximização das taxas de classificação correta nos dados de teste, que corresponde à razão entre o número de exemplos corretamente classificados e o número total de exemplos disponíveis no conjunto de testes.

- **Agrupamento:** Tem o objetivo de particionar um conjunto de dados em um conjunto de grupos, cujos membros possuam algumas propriedades interessantes em comum. Outra possibilidade é o agrupamento *Bayesiano* que busca o número de classes que resulta em um melhor ajuste de uma distribuição de probabilidade aos dados [FAY96].

  Em resumo, os processos de *KDD* consistem em vários passos, iterativos e interativos, permitindo que padrões sejam extraídos do banco de dados de forma

automatizada. Dentre esses passos, cita-se como o mais importante a "*MD*", onde é aplicado o algoritmo de descoberta do conhecimento. Este algoritmo é específico para cada tipo de tarefa que se deseja aplicar, de acordo com sua necessidade, como mostra a tabela 2.1 das principais tarefas de MD, a seguir.

Tabela 2.1 – Principais Tarefas de Mineração de Dados [LAV03].

GRUPO DE TAREFAS	TAREFA	DESCRIÇÃO	TIPO DA TAREFA	MÉTODOS
Análise de Dependências	Associação	Consiste na descoberta de regras que descrevem dependências significativas entre os itens de um banco de dados que ocorrem na mesma transação.	Tarefa descritiva	Estatística, teoria dos conjuntos, modelos probabilísticos de gráficos de dependência.
	Detecção de Seqüências Temporais.	Consiste na descoberta de regras que descrevem dependências entre itens que ocorrem ao longo do tempo.	Tarefa descritiva	Estatística, teoria dos conjuntos.
Identificação de classes	Regressão	Consiste em identificar um padrão de comportamento dos itens do banco de dados, descobrindo uma função que possa mapear os novos itens em um valor retornado pela função descoberta.	Tarefa preditiva	Regressão linear, redes neurais, ajuste de curvas, árvores de decisão.
	Classificação	Consistem em identificar um padrão de comportamento dos itens do banco de dados, visando mapear os novos itens em classes pré-existentes.	Tarefa preditiva	Árvores de decisão, redes neurais, métodos baseados em exemplo.
	Clusterização	Consiste em gerar e identificar grupos, *clusters*, a partir do agrupamento natural de itens do banco de dados de acordo com a similaridade entre eles.	Tarefa descritiva	Árvore de decisão, redes neurais, métodos baseados em exemplo.
Descrição de classes	Sumarização	Consiste em descobrir regras que descrevam sucintamente uma classe.	Tarefa descritiva	Árvore de decisão, redes neurais, algoritmos genéticos.
Detecção de desvios	Detecção de desvios	Consistem em detectar desvios e anomalias dos itens em um banco de dados.	Tarefa descritiva	Teoria dos conjuntos, estatística.

2.2. Aprendizado de Máquina

Muito tem-se falado na capacidade computacional, que ao decorrer do tempo tem-se evoluído de forma a superar a capacidade humana. Porém, a aprendizagem de máquina ainda está longe de ser dominada na forma humana, que em sua natureza é de forma simplista, porém com um contexto extremante complicado a nível de conhecimento, ou seja, a aprendizagem é uma propriedade essencialmente humana, onde o significado de aprender é a

mudança que se propõe a fazer melhor, dentro de um determinado critério no momento em que uma situação similar ocorrer. Assim, aprendizagem não é definitivamente um processo de memorização, pois se fosse, os computadores não teriam nenhuma dificuldade e já teriam superado o homem, dado a sua facilidade de memorizar, bem como os diversos avançados e recursos de memória disponíveis. Assim, a dificuldade deste processo não é *memorizar* e sim *generalizar* um comportamento para uma nova situação.

O aprendizado pode ser visto de duas forma (MICHALSKI et al., 1986):

1. Refinamento de Habilidades;
2. Aquisição de conhecimento.

O refinamento de habilidades é representado pela evolução gradual das atividades motoras e cognitivas, despertadas através da prática. O refinamento das habilidades aprendidas, aprimora o aperfeiçoamento do processo de aprendizado, através da coordenação motora e mental, com o objetivo de corrigir as imperfeições. Por exemplo, quanto mais se pratica um esporte, melhor fica seu desempenho, tornando-se um especialista no assunto.

Já a aquisição do conhecimento, é a capacidade de adquirir informações através de símbolos em conjunto com o desenvolvimento da habilidade de efetuar a aplicação na informação de modo efetivo. É como aprender matemática, onde pode-se dizer que absorveu conceitos importantes, com um entendimento de seus significados relacionados ao mundo real.

A aprendizagem possui duas fases distintas, são elas:

1. **Treinamento (supervisionado)**: são apresentados exemplos ao sistema, de forma a aprender a partir destes exemplos e efetuar gradativamente seus parâmetros ajustáveis, afim de se obter uma saída próxima da desejada.
2. **Utilização**: Neste momento, há o surgimento de novos exemplos que nunca antes foram vistos. Assim, há um desejo que, o sistema automaticamente efetue a generalização do que apareceu.

Com isto, desmistifica-se a confusão entre *aprendizado* e *memorização*, dando uma falsa idéia de que o processo de aprendizagem de máquina é uma tarefa simples. Muito pelo contrário, além destas dificuldades, este processo possui diversos problemas cujos principais podem ser divididos em três partes:

1. **Classificação:** a determinação da *entrada* em uma certa classe, como por exemplo a capacidade de reconhecimento de uma certa imagem, dentre um número finito de

imagens, tendo em vista que, não foram exploradas todas as características desta imagem para o auxílio no processo de reconhecimento;

2. **Regressão:** o processo de predição através de um exemplo ou através de dados históricos de dias e meses anteriores, como por exemplo o valor da bolsa de valores para o dia seguinte;
3. **Estimação de Densidade:** Este processo é efetuado através de reconhecimento ou identificação de um exemplo de forma estimativa. Bem como a extração de *n* tipo de especialidade ou categoria que se propõe a identificar de forma quantitativa.

2.2.1. Modos do Aprendizado

Conforme (RUSSEL et, al., 1995) "*para alguns sistemas de aprendizado é necessário predizer que se uma certa ação irá fornecer uma certa saída*". Assim, pode-se classificar os sistemas de aprendizado de máquina da seguinte forma:

1. *aprendizado supervisionado:* é o aprendizado onde dado um conjunto de observações ou exemplos rotulados, ou seja, a classe de cada exemplo é conhecida, objetivando encontrar hipóteses com a capacidade de classificar novas observações entre as classes que já existem;
2. *aprendizado não-supervisionada* ou *clustering*: através de um conjunto de exemplos sem identificação, objetiva-se a buscar similaridade ou a existência de *clusters*.

Com isto, obtêm-se diversas formas de aprendizado, apoiado por uma gama significativa de algoritmos e aplicações que enriquecem a abordagem de aprendizado de máquina. Estes algoritmos estão divididos em paradigmas, são eles:

- **Paradigma Simbólico:** busca o aprendizado construindo representações simbólicas de um conceito através de exemplos e contra-exemplos desse conceito [MOR73];
- **Paradigma Estatístico:** é um método de classificação, também utilizado para construir árvore de decisão, tendendo a focar tarefas em que todos os atributos possuem valores contínuos e ordinais;
- **Paradigma *Instance-based***: ao lembrar de um caso que possua alguma similaridade, onde a classe é conhecida e admitir que o novo caso terá a mesma classe é uma forma de classificar um caso. Esta é uma forma simplista de exemplificar os sistemas *instance-based* [QUI93];

- **Paradigma Conexionista**: este paradigma surgiu da representação de uma rede neural, que possui diversas unidades interconectadas entre si. As redes neurais são construções matemáticas relativamente simples, que foram inspiradas pelo sistema nervoso do ser humano [MCC43];
- **Paradigma Genético:** dada população individual de elementos de classificação, competindo entre si para que se possa fazer a predição, é conhecido como um classificador genético. Este paradigma, avalia a força dos elementos, onde somente os que possuírem desempenhos mais fortes podem proliferar, baseando-se na teoria de Darwin, onde só sobrevivem os mais bem adaptados ao ambiente [GOL89];

2.2.2. Aprendizado Incremental e Não-Incremental

Dada a diversidade dos algoritmos, eles podem ser classificados de acordo com seus exemplos de duas formas:

1. *Não incremental*: neste formato, exige a necessidade de que todos os exemplos estejam disponíveis para que possam ser induzidos a um conceito. A vantagem deste algoritmo é utilizar em problemas de aprendizado onde a base de dados possua um porte pequeno ou médio e que não ocorram mudanças em seus exemplos [GOM02].
2. *Incremental*: neste formato, segue a sua definição, onde trabalha-se com respostas a cada novo exemplo de treinamento observado. Assim, o sistema considera os exemplos um a um. Ou seja, após o primeiro exemplo ser considerado, o mesmo é tomado como exemplo para construir uma determinada hipótese e após, considera-se um segundo exemplo, podendo fazer (ou não) mudanças na primeira hipótese, apenas baseando-se em como esta classifica o segundo exemplo, de forma a fazer as modificações no conceito, à medida que mais exemplos são apresentados. Algumas vantagens deste algoritmo é a possibilidade de ser atualizado a cada nova observação, bem como a sua utilização em bases de dados relativamente densas [GOM02].

2.3. Considerações Finais do Capítulo

Neste capítulo foram apresentadas algumas das principais características de Mineração de Dados tais como, Descoberta do Conhecimento, Etapas do processo do Conhecimento e Tarefas do Processo de Descoberta do Conhecimento, bem como uma breve descrição em Aprendizagem de Máquina, tendo em vista a sua enorme contribuição aplicada à Mineração de Dados.

No próximo capítulo são abordadas as Regras de Associação bem como os algoritmos tema deste trabalho, Algoritmos *APriori* e *ZigZag*.

Capítulo III

Regras de Associação

Neste capítulo são apresentados os principais fundamentos e aplicações das regras de associação de forma normal através do algoritmo *APriori* e de forma Incremental através do algoritmo *ZigZag*. Também é apresentada uma descrição detalhada dos algoritmos, bem como exemplos de aplicação.

3.1. Introdução à Regra de Associação

Conhecidamente, uma das mais populares e importantes tarefas de mineração de dados, as regras de associação se destacam devido a sua aplicabilidade em problemas de negócios reais e principalmente pela sua facilidade de compreensão até mesmo para pessoas com pouco conhecimento ou contato com *Data Mining* [HIP02]. Esta tarefa tem a capacidade de identificar grupos de atributos de um banco de dados tipicamente relacionados, representando assim a probabilidade de um grupo aparecer em uma transação, tendo em vista que um outro está presente na mesma [LAV03].

Mesmo que a aplicação das regras de associação possa ser utilizada em qualquer tipo de base de dados, a proposta inicial da sua aplicação se deu a partir de uma análise de cesta de produtos (*market basket analysis*). Sua composição é feita pelos itens adquiridos pelos clientes em uma determinada cesta de compra, isto é, em uma mesma transação. O conhecimento extraído através desta metodologia pode ser aplicado em diversos segmentos de mercados, tendências, propagandas promocionais, decisões de *marketing* etc.

Uma representação clássica das Regras de Associação é o formato condicional do tipo SE X ENTÃO Y, tipicamente conhecida no formato de implicação $X \rightarrow Y$, contendo dois elementos: o antecedente representado pelo X e o conseqüente que é representado pelo Y

[AGR93a]. Normalmente estas regras são associadas a valores denominados *Suporte* (*Sup*) e *Confiança* (*Conf*) dos elementos das regras. O valor de *Sup* tem o objetivo de indicar a porcentagem de ocorrências da associação em relação ao montante total de registros, isto é, a probabilidade de que uma transação satisfaça a condição *X*. Já o valor de *Conf*, tem a função de indicar todas as ocorrências em percentual do antecedente onde o item conseqüente está associado, ou seja, é a probabilidade de que uma transação satisfaça a condição *Y*, se ela satisfaz a condição *X* [LEV99].

Tabela 3.1: Exemplos de Regras de Associação

Regras	**Suporte (Sup)**	**Confiança (Conf)**
Compra(*x*, "fraudas") → compra(*x*, "cerveja")	0,5%	60%
Possui_plano_de_Saúde(*x*) ^ Sexo_Masculino(*x*) → Pessoa_Jurídica(*x*)	14,2403%	50,2764%

O funcionamento dos algoritmos para extração das regras se dá a partir da análise das combinações dos dados do conjunto a ser pesquisado. Com isto, a sua operação tem um crescimento exponencial em função do número de itens a ser comparado.

Existem diversas técnicas que foram originadas nas áreas de Estatística, Teoria dos conjuntos, Modelos Probabilísticos de gráficos de dependências, sendo consideradas técnicas mais usadas para a extração de regras de dependências [FAY96, BIG96].

3.2. Descrição Formal do Problema

Considerando o formato original, a tarefa de descoberta de regras de associação foi definida em um tipo especial de dados, usualmente chamado de "*market basket data*". A composição deste tipo de banco de dados se dá por um conjunto de itens. No ambiente relacional, este banco de dados, é representado por uma tabela *T* composta de atributos binários. Estes atributos correspondem aos itens e os registros dentro da tabela representam as transações. Com isto, os atributos binários assumem valor *1*, caso seja constatado a presença do item na transação, e caso não esteja presente, o valor assumido é *0* [LAV03].

Tabela 3.2: Tabela *T* de Itens Binários

Transações (T_{ID})	I_1	I_2	I_3
1	1	1	1
2	1	1	0
3	1	0	0
4	0	0	0
5	0	1	1
6	0	0	1

Uma regra de associação consiste em uma expressão da forma $X \rightarrow Y$, onde X e Y são conjuntos de itens [AGR93a]. Desta forma, a representação é descrita da seguinte forma: X_1, ^ X_2, ... X_n, onde X_i ($i \in \{1, m\}$) e Y_j($j \in \{1,n\}$) dão a representação dos itens, são os atributos do banco de dados [CHN96]. As regras significam que as transações do banco de dados que contém X tendem a conter Y.

A representação do problema de descoberta de regras de associação num modelo formal foi proposto, inicialmente, por Agrawal, Imielinski e Swami em 1993 [AGR93a]. Considere um conjunto de atributos binários chamados de itens I = $\{I_1, I_2, \ldots, I_m\}$. Seja D um conjunto de transações, onde cada transação $t \subset I$ é um *Conjunto de Itens*. Cada transação é associada a um identificador único, chamado de T_{ID}. Assim, seja $X \subset I$ um *conjunto de itens*, uma transação t contém X se $X \subset t$. Com isto, conclui-se que uma regra de associação é um relacionamento da forma $X \rightarrow Y$, onde $X \subset I$, $Y \subset I$ e $X \cap Y = \varnothing$.

Toda regra de associação $X \rightarrow Y$ possui um valor de *Sup* no conjunto de transações D se *Sup*% das transações em D contiver X e Y [AGR93a, AGR96]. O valor de *Sup* é a probabilidade de uma transação em D conter $X \cup Y$, isto é, a freqüência da regra. Com isto, o *Sup*($X \cup Y$) é definido como:

quantidade $(X \cup Y)$ / *tamanho (D)* = $|\{t \in D \mid X \subset t, Y \subset t\}| / |D|$ *ou*

$$Sup(A \rightarrow B) = \frac{\sum\{t \in D \mid (A \cup B) \subseteq t\}}{\sum\{t \in D\}} \qquad (3.1)$$

De forma análoga, a regra $X \rightarrow Y$ é válida no conjunto de transações D com um valor de *conf,* se *conf*% das transações do banco de dados que contêm X também contém Y. Isto é, o valor *Conf* é a probabilidade de Y ocorrer em uma transação de D tendo em vista que existe a ocorrência de X, indicando a "força" da regra. Assim, *Conf* de uma regra é representada por:

$$Conf(A \rightarrow Y) = Sup(X \cup Y) \,/\, Sup(X) \text{ ou}$$

$$Conf(A \rightarrow B) = \frac{\sum \{t \in D \mid (A \cup B) \subseteq t\}}{\sum \{t \in D \mid A \subseteq t\}} = \frac{Sup(A \rightarrow B)}{Sup(A)} \tag{3.2}$$

Dado o exemplo da Tabela 3.3, representado por cinco (05) itens de compra, temos 10 transações nas regras, com valores de suporte mínimo *Sup*=10%, gerando com isto 8 regras de associação, então pode-se concluir que o *Sup* é um percentual que determina a ocorrência de 1 item ou um conjunto de itens no total de transações. Então vê-se que o item 2 ocorre 7 vezes em 10 transações, determinando o valor de suporte *Sup*=70%. Já a confiança, é determinada pela ocorrência mútua de 2 ou mais itens em relação ao *Sup* do antecedente da regra. Assim, tem-se os itens 2 e 3 aparecendo juntos em 6 das 7 vezes que o item 2 aparece, determinando então *Conf*=85,7%.

Tabela 3.3: Exemplo de Aplicação de Extração de Regras de Associação.

Transações
1,2,3
1,4,5
2,3,4
1,2,3,4
2,3
1,2,4
4,5
1,2,3,4
3,4,5
1,2,3

Regras	**Suporte (*Sup*)**	**Confiança (*Conf*)**
3 → 2	70,0 %	85,7%
2 → 3	70,0%	85,7%
2 → 1 4 → 5	60,0% 30,0%	83,3% 100,0%
3 → 2 1	50,0%	80,0%
3 → 2 1 4 → 3 5	40,0% 10,0%	100,0% 100,0%
2 → 3 4 1	20,0%	100,0%

3.2.1. Mineração Incremental

O conceito de Mineração Incremental indica a dinâmica de constâncias nos fluxos de atualização de novos dados em uma base de dados, fazendo com que exista uma variância na associação de itens, que pode vir a se tornar fraca ou forte na medida em que se evolui a base de dados. Com isto, uma das formas para manter as regras de associação sempre atualizadas é utilizar D como ponto de partida, d^+ como acréscimo de um conjunto de novas transações e a operação de remoção das velhas transações é representada por d^-, i.e., $\Delta = (D \cup d^+)$ $-d^-$ [VEL02a]. Entretanto, este processo encontrará uma deficiência se a taxa de atualização da base de dados for muito alta, fazendo com que o custo computacional da replicação freqüente dos algoritmos seja excessivamente alta.

Assim, diversas técnicas de atualização incremental, foram desenvolvidas com o objetivo de prevenir que se obtenha um processo de retrabalho na mineração dos dados em toda a nova base de dados.

No processo de mineração incremental, ao constatar que uma base de dados sofre alterações, as regras descobertas anteriormente podem se tornar inválidas na nova base de dados, ou ocorrer o surgimento de novas regras que eram inválidas anteriormente. O *Conjunto de Itens* freqüentes que continuam sendo válidos na base atualizada, são chamados de *Conjuntos de Itens Retido*. Já aqueles *Conjunto de Itens* que se tornarem freqüentes após a atualização da base de dados, são chamados de *Conjunto de Itens Emergentes* [VEL01]. Então, pode-se afirmar que os algoritmos que trabalham de forma incremental com as regras de associação, possuem um mecanismo para a geração dos *Conjunto de Itens Retido* e dos emerge *Conjunto de Itens*, descartando os conjuntos freqüentes que se tornaram associações fracas [LAV03].

3.3. Algoritmo *APriori*

O algoritmo *APriori* foi proposto por Rakesh Agrawal e Ramakrishnan Srikant [AGR94] conforme apresentado na Figura 3.1. O princípio do algoritmo *APriori* é garantir que qualquer subconjunto de um *Conjunto de Itens* seja freqüente. Assim, para minerá-los, deve-se encontrar todos os *Conjunto de Itens* freqüentes (L_k) (conjunto de itens que possui um suporte mínimo). A isto, deve-se encontrar um subconjunto de um *Conjunto de Itens* que também tenha um *Conjunto de Itens* freqüente, i.e, se $\{AB\}$ é um *Conjunto de Itens* freqüente, ambos $\{A\}$ e $\{B\}$ devem ser *Conjunto de Itens* freqüentes. Já em sua iteração, deve-se

encontrar os *Conjunto de Itens* com cardinalidade de 1 até *k*, ou seja *k-Conjuntos de Itens* .

Assim, deve-se usar os *Conjunto de Itens* freqüentes para gerar as regras de associação.

Passo de Junção: C_k é gerado combinando L_{k-1} consigo

Passo de Poda: Qualquer (*k-1*)-*Conjunto de Itens* que não é freqüente não pode ser um subconjunto de um *k-Conjunto de Itens* freqüente

Pseudo-Código:

C_k: *Conjunto de Itens* candidatos de tamanho *k*

L_k: *Conjunto de Itens* freqüentes de tamanho *k*

L_1 = {itens freqüentes};
para (k = 1; L_k != *Ø*; k++) ***faça***
    C_{k+1} = candidatos gerados de L_k;
  para cada transação *t* na base de dados ***faça***
    atualiza o contador de todos os candidatos em C_{k+1} que estão contidos em *t*
  L_{k+1} = candidatos em C_{k+1} com *min_support*
end
return $\cup_k L_k$;

Figura 3.1: Algoritmo *APriori*

A premissa básica do algoritmo *APriori* é garantir que qualquer subconjunto de um *Conjunto de Itens* seja freqüente. Então, o conjunto de candidatos contendo *k* itens pode ser gerado fazendo uma combinação dos *Conjuntos de Itens* de tamanho (*k-1)* e, como conseqüência, eliminar os itens que contenham algum subconjunto que não seja freqüente [AGR94]. Para tanto, o algoritmo *APriori* utiliza um outro algoritmo para efetuar a geração do conjunto de itens candidatos, apresentado na Figura 3.2 [LAV03]. O passo 1, tem a função de realizar a combinação dos conjuntos em L_{k-1} para a geração de C_k. Já no passo 2, ocorre a retirada de C_k dos candidatos que possuem algum subconjunto de tamanho (*k-1)* que não seja freqüente, isto é, não façam parte de lista de L_{k-1}. Dentro do algoritmo *APriori* existe um mecanismo que assume que os itens de cada transação do banco de dados estão em ordem lexicográfica, então $p.item_1$, $p.item_2$, ..., $p.item_k$ e $q.item_1$, $q.item_2$, ..., $q.item_k$ assim, este algoritmo ordena os dados para a geração dos candidatos.

1) **auto-unindo** L_{k-1}

$C_k \leftarrow$ **selecione** $p.item_1, p.item_2, ..., p.item_{k-1}, q,item_{k-1}$

para cada $L_{k-1}\, p, L_{k-1}\, q$

para os quais $p.item_1 = p.item_1, ..., p.item_{k-2} = q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$

2) **poda**

Para todos os *Conjunto de Itens c em* C_k **faça**

Para todos (*k-1)- subconjuntos s* de tamanho de *c* **faça**

Se (*s* não está em L_{k-1}) **então**

Delete *c* de C_k

Figura 3.2: Algoritmo para a Geração dos Candidatos do *APriori* .

Dentre as particularidades que o algoritmo exige, tem-se o Cálculo do Suporte dos Candidatos. Como justificativa deste cálculo, tem-se:

- O número total de candidatos pode ser elevado de acordo com a quantidade de atributos da base de dados aplicada no algoritmo;
- De forma análoga, uma transação pode conter muitos candidatos.

A metodologia existente é a seguinte:

- Os *Conjuntos de Itens* candidatos são armazenados em uma ***hash tree***[2];
- **Nós folha** de uma *hash tree* contém uma lista de *Conjunto de Itens* e contadores;
- **Nós interiores** contêm uma tabela *hash*[3];
- **Função *SubConjunto***: encontra todos os subconjuntos em uma transação.

1. Para um exemplo da aplicação do algoritmo, considera-se que tem-se inicialmente uma base de dados *E* formado por 4 atributos {A, B, C, D}. Estes atributos ou registros obtêm em sua base um total de 5 (cinco) itens, distribuídos aleatoriamente na base de dados conforme a Figura 3.3. A partir daí, o algoritmo efetua uma análise da base de dados. Após isto, efetua a contagem da incidência dentro da base de forma a

[2] *HASH TREE* – Em criptografia e de ciência da computação *Hash* árvores ou *Merkle* árvores são um tipo de estrutura de dados que contém uma árvore de informações resumidas sobre um grande pedaço de dados por exemplo, um arquivo utilizado para verificar o seu conteúdo.

[3] *HASH* ou *TABELA HASH* - Em ciência da computação a tabela *hash* (de *hashing*, no inglês), também conhecida por tabela de espalhamento, é uma estrutura de dados especial, que associa chaves de pesquisa (*hash*) a valores. Seu objetivo é, a partir de uma chave simples, fazer uma busca rápida e obter o valor desejado. É algumas vezes traduzida como tabela de escrutínio.

ordenar o *Conjunto de Itens* e efetuar a contagem de cada item, gerando assim C_1. O próximo passo é efetuar a "poda" do item que não obtiver uma freqüência na base de dados, que para o exemplo é o item 4, contendo apenas uma incidência na base, gerando com isto L_1. Após, o algoritmo efetua mais uma checagem das incidências de itens compostos e os ordena, gerando com isto C_2. Mais uma vez o algoritmo efetua a análise da base *E* e esta análise irá mais uma vez excluir os itens que não possuírem freqüência suficiente na base, que neste caso são os itens {1 2} e {1 5}, gerando o L_2. Novamente se refaz a checagem das incidências para gerar um novo *Conjunto de Itens* C_3 que neste caso gerou {2 3 5}; o sistema retorna as análises da base *E* para gerar o L_3 e finaliza o processo, tendo em vista que não há mais conjunto a ser gerado contendo então o *Conjunto de Itens* L_3 {2 3 5}.

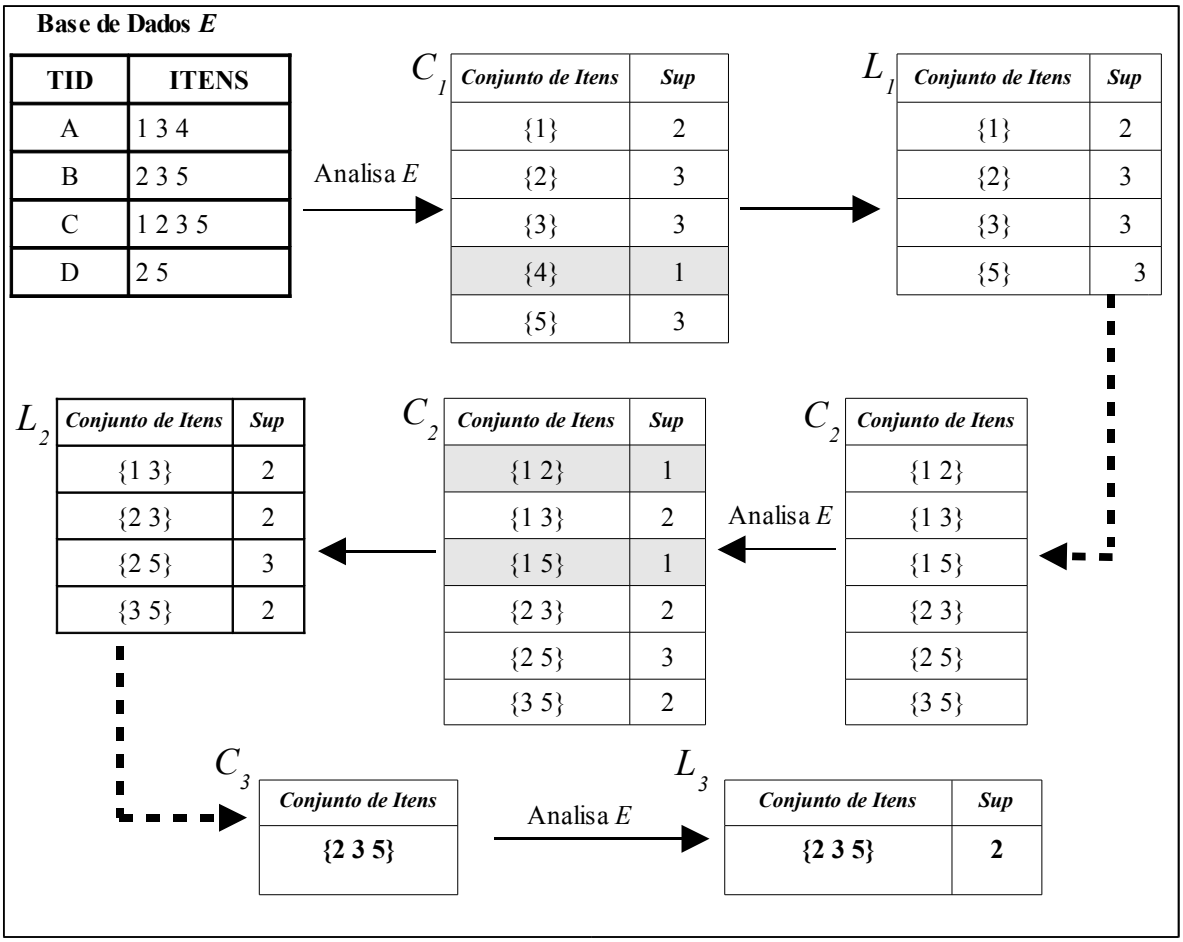


Figura 3.3: Exemplo de Aplicação do Algoritmo *APriori*.

3.3.1. Eficiência e Desempenho do Algoritmo *APriori*

A mineração de regras de associação é uma tarefa complexa e difícil por pelo menos duas razões: quando existirem muitos dados na base de dados e pela quantidade elevada de atributos. Fazendo com que o tempo de execução do algoritmo seja linear em relação ao tamanho da amostra, o número de passagens pela base de dados é um fator que reflete neste tempo. Além das *n* passagens pela base de dados, onde *n* é a quantidade de itens do maior conjunto de candidatos, faz com que cada passagem pelo banco de dados signifique uma rotina de *Entrada* e *Saída*. Um outro fator é o número de atributos que são freqüentes ajudando na queda de desempenho do algoritmo como descrito a seguir.

Um método que se propõe a efetuar uma melhora significativa na eficiência do algoritmo, é empregado como segue:

- **Armazenar os contadores de *Conjunto de Itens* em *hash-tables***: Este processo se dá através de um *k-Conjunto de Itens* cujo contador correspondente está abaixo do *limiar* não podendo ser freqüente;
- **Redução de transação:** Uma transação que não contiver um *Conjunto de Itens* freqüente é inútil para a análise dos *Conjuntos de Itens*;
- **Particionamento:** Qualquer *Conjunto de Itens* que é potencialmente freqüente na Base de Dados, deve ser freqüente em pelo menos uma partição da Base de Dados;
- **Amostragem**: Mineração de um subconjunto dos dados, suporte menor é mais um método para determinar a "completeza";
- **Contagem dinâmica do *itemset***: Adicione novos *Conjunto de Itens* candidatos somente quando todos os seus subconjuntos são estimados serem freqüentes.

O algoritmo *APriori* possui um núcleo que usa (*k-1*)-*Conjunto de Itens* freqüentes para gerar o conjunto de *k-Conjunto de Itens* de candidatos freqüentes. Assim, faz uma busca e comparação para efetuar a coleta do *score* dos *Conjunto de Itens* candidatos.

Outro ponto que faz o algoritmo reduzir seu desempenho, é a existência no algoritmo de um gargalo que é apresentado na geração de candidatos, onde sua complexidade se dá em grandes conjuntos de candidatos, como se segue:

- Necessita efetuar análise na Base de Dados de ($n + 1$) análises, onde *n* é o tamanho do maior padrão;

- Quando existe uma quantidade muito grande de itens dos conjuntos freqüentes, o desempenho do algoritmo *APriori* é degradada significativamente, devido ao custo computacional e não ao processo de entrada e saída (E/S);
- Com relação ao tamanho do valor do *Suporte Mínimo* (*minsup*), quanto menor for o valor especificado, maior será a freqüência de conjuntos de candidatos.

Com isto, diversos algoritmos foram propostos para resolver o problema encontrado pelo algoritmo *APriori*, dentre eles destacam-se:

- Algoritmos que se propõe a efetuar uma redução do número de passagens pelo banco de dados: *APrioriTid* [AGR94], *DHP* [PAR97], *DIC* [BRIN97], *Partition* [SAV95], *FPGroup* [HAMPEI00];
- Algoritmos que utilizam o processo de amostragem: [TOI96] e [DOM98];
- Algoritmos que se propõem a efetuar a redução dos candidatos: DHP [PAR97], [CAM00];
- Algoritmos que obtêm a alternativa para aplicação do valor de Suporte muito baixo: *MsAPriori* [LIU99];
- Algoritmo que se propõe a trabalhar com Bases de Dados densas: *MaxMiner* [BAY98], *A-close* [PAS99], *Closet* [HAN00] e *Charm* [ZAK02];
- Uma das primeiras abordagens incrementais, possui uma característica iterativa, de forma que a cada iteração realiza um acesso à base de dados *D+*. Este algoritmo requer *k* acessos à base de dados, onde *k* representa a cardinalidade do maior conjunto de itens freqüentes. É muito parecido com o algoritmo *DHP*: Algoritmo *FUP* (CHEUNG et al., 1996);
- Algoritmo Incremental de Dados, propõe-se a trabalhar com a *Máxima Freqüência de Itens* (MFI), algoritmo tema deste trabalho: algoritmo *ZigZag* (VELOSO et al., 2002a).

3.4. Algoritmo *ZigZag*

A idéia principal do algoritmo *ZigZag* evoluiu, a partir da idéia de Mineração Incremental para a extração de Regras de Associação, primeiramente introduzido por [CHN96]. O tema central da mineração incremental se deu através da possibilidade de se fazer a reutilização dos cálculos minerados, com o objetivo de melhorar o desempenho das futuras iterações. O nome do algoritmo *ZigZag*, foi inspirado no comportamento do algoritmo

que efetua uma procura de soluções por *Conjunto de Itens* com a máxima freqüência de uma forma ascendente e efetua a enumeração dos seus subconjuntos de uma forma descendente (VELOSO et al., 2002).

O algoritmo *ZigZag* tem a finalidade de manter somente o conjunto de itens freqüentes *maximal* (MFI), também conhecido como um limiar positivo, para construir de maneira incremental uma grade de *conjunto de itens freqüentes* (VELOSO et al., 2002). Este algoritmo usa o conhecimento descoberto nas fases anteriores para reduzir o custo da atualização dos *Conjuntos de Itens*. Os *MFI* são atualizados através de um processo de buscas retroativas efetuadas pelos resultados das minerações anteriores, fazendo com que se tenha um modo eficiente de determinar os *Conjunto de Itens* freqüentes. A idéia da busca retroativa para encontrar o *MFI* nas grandes bases de dados está relacionado a um algoritmo não-incremental desenvolvido inicialmente em um algoritmo chamado GENMAX [GRO98].

Pode-se destacar também a eficiência deste algoritmo no momento em que se desejar alterar alguns parâmetros como, por exemplo o suporte mínimo. A computação do suporte usado em *ZigZag* é baseada na associatividade dos conjuntos de itens (VELOSO et al., 2002a). Este algoritmo também suporta mineração incremental quando antigas transações são removidas das *BD* e novas são adicionadas, mantendo o conjunto de regras de associação coerente com os dados mais recentes, a fim de permitir com isto um elevado grau de interatividade.

O Algoritmo *ZigZag* é adequado para buscar regras de associação confiáveis por meio da função estável [LM01]. As regras estáveis não sofrem grandes mudanças ao longo do tempo, sendo assim mais confiáveis. O algoritmo também permite estabelecer um limiar de relaxamento para atualizar apenas os conjuntos de itens freqüentes em que a variação de popularidade excede esse limiar. A idéia é que, se existirem poucas mudanças nos dados relacionados a um *Conjunto de Itens*, as regras de seus subconjuntos provavelmente permanecerão muito próximas das regras originais. Por não efetuar a atualização completa na base de dados, o algoritmo aumenta a sua eficiência [CAV95]. O algoritmo *ZigZag* é composto de três etapas principais:

1. Gravar novas transações e descartar as transações obsoletas;
2. Atualizar o *Conjunto de Itens* com a máxima freqüência na atualização de base de dados;
3. Atualizar o *suporte* (*Sup)* dos outros *Conjuntos de Itens* freqüentes na base de dados.

3.4.1. Definições Preliminares

- Um *Conjunto de Itens X* é freqüente se o seu suporte não é mais utilizador de um determinado suporte mínimo (*minsup*);
- Um *k-Conjunto de Itens*, é um *Conjunto de Itens* que contém *k* itens;
- Uma transação *T* é um conjunto de itens (*Conjunto de Itens*);
- I é um conjunto de Itens;
- $T \subseteq I$ é somente identificado como *tid*;
- *D* é uma base de dados das transações;
- *tidlist,* é um conjunto de *tids* que contém um *Conjunto de Itens*;
- *X* é denotado como $\mathcal{L}_{\Delta}(X)$.
- $\Delta = (D \cup d^{+})\text{-}d^{-}$, usando um ponto de partida da base de dados *E*, um conjunto de novas transações d^{+} são adicionadas e um conjunto de antigas transações d^{-}, são removidas da base, formando uma nova transação;

Para um melhor entendimento é apresentado um exemplo descrito e implementado por (VELOSO et al., 2002a) na página seguinte.

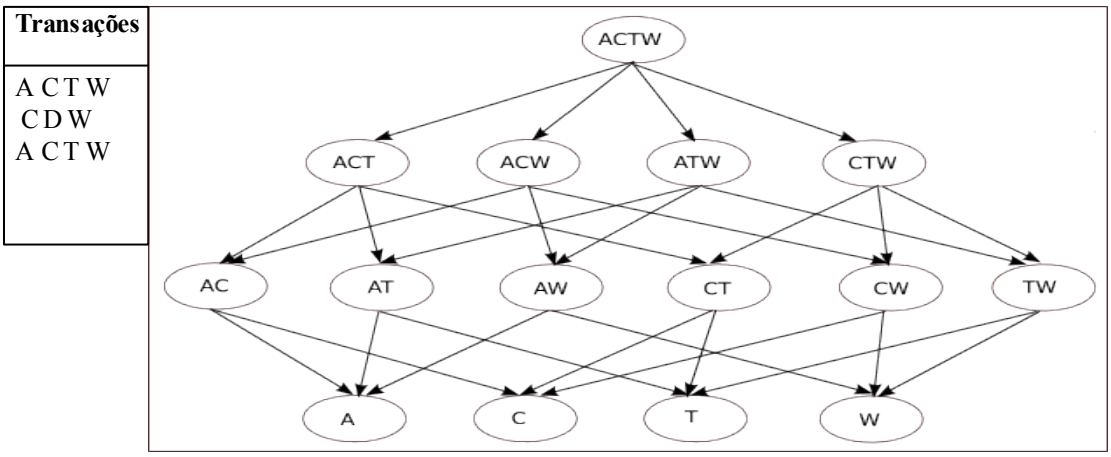


Figura 3.4: Arranjo da Freqüência de *Conjuntos de Itens A*pós as Três Primeiras Transações [VEL02].

Suporte (*Sup*)	*Conjunto de Itens* que aparecem
100%	C, W, CW
67%	A, T, AC, AT, AW, CT, TW, ACT, ACW, ATW, CTW, ***ACTW***

TidLists			
A	**C**	**T**	**W**
1	1	1	1
3	2	3	2
	3		3

Regra de Associação - Confiança (*Conf*) 100%
$A \Rightarrow C$, $A \Rightarrow T$, $A \Rightarrow W$, $A \Rightarrow C\ T$, $A \Rightarrow C\ W$, $A \Rightarrow TW$, $A\ CTW$, $A \Rightarrow CTW$, $AC \Rightarrow T$, $AC \Rightarrow W$, $AC \Rightarrow TW$, $AW \Rightarrow CT$, $AW \Rightarrow C$, $AW \Rightarrow T$, $ACW \Rightarrow T$, $C \Rightarrow W$, $T \Rightarrow A$, $T \Rightarrow C$, $T \Rightarrow W$, $T \Rightarrow AC$, $T \Rightarrow AW$, $T \Rightarrow CW$, $T \Rightarrow ACW$, $TA \Rightarrow C$, $TA \Rightarrow W$, $TC \Rightarrow A$, $TC \Rightarrow W$, $TW \Rightarrow A$, $TW \Rightarrow C$, $TA \Rightarrow CW$, $TC \Rightarrow AW$, $TAC \Rightarrow W$, $TW \Rightarrow AC$, $TAW \Rightarrow C$, $TCW \Rightarrow A$, $W \Rightarrow C$

Figura 3.5: Demonstração do Desenvolvimento do Algoritmo *ZigZag* [VEL02]

Transações
A C T W
C D W
A C T W
C D W
C D T W

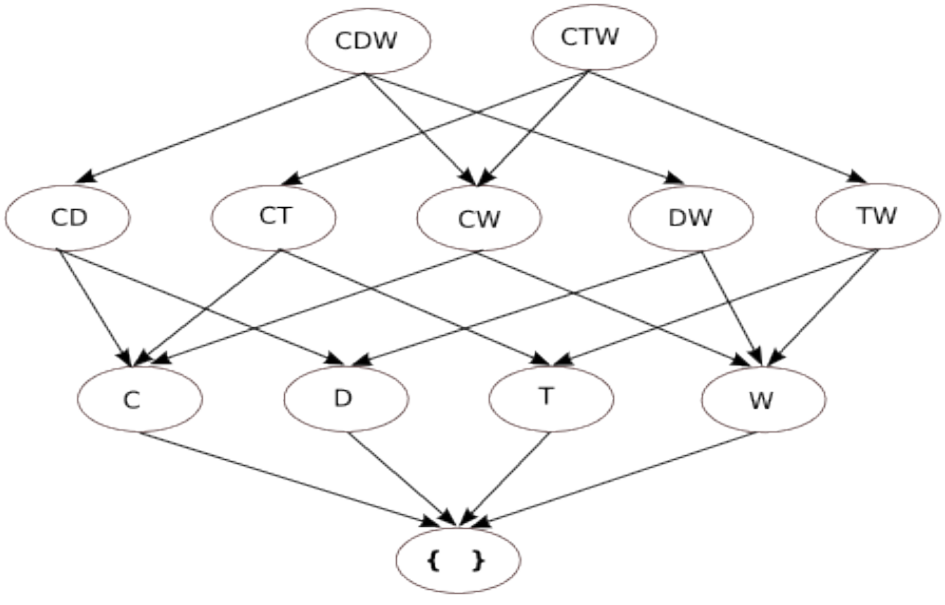


Figura 3.6. Arranjo de *Conjunto de Itens* Após o Incremento de Dados [VEL02].

Suporte (*Sup*)	***Conjunto de Itens* aparecem**
67 %	D, CD

Suporte (*Sup*)	***Conjunto de Itens* recusados**
33%	A, AC, AT, AW ACT, ATW, ACTW

Suporte (*Sup*)	***Conjunto de Itens R*etidos**
100%	C
83%	W, C W
67%	T, CT
50%	TW, ***CTW***

Figura 3.7: Desenvolvimento do Algoritmo *ZigZag* Após o Incremento de Dados [VEL02].

De acordo com a ilustração das principais operações do algoritmo nas Figuras 3.4 a 3.7, neste exemplo o valor de *Sup* é 50%. Os *Conjunto de Itens* protegidos são os *Conjunto de Itens* com a máxima freqüência em cada estrutura. A estrutura superior é obtida através da mineração da base de dados original *E* em três transações. A transação *ACTW* é o único *Conjunto de Itens* com a máxima freqüência. As regras com 100% de confiança são apresentadas na Figura 3.5. A parte inferior da estrutura mostra o que acontece quando um incremento de três novas transações for adicionado em *E*.

Aqui não existe nenhum decremento d^-, de forma que $\Delta = D \cup d^+$. Com o mesmo suporte mínimo, a base principal demonstra ACTW. Após a atualização do incremento tem-se CDW e CTW como os dois novos *Conjunto de Itens* freqüentes máximos. Deve-se observar que mesmo neste exemplo simples, a atualização afetou significativamente os resultados, permitindo que tenha-se as características importantes da atualização do processo. Primeiramente é possível que o conjunto de máxima freqüência de *Conjunto de Itens* mude completamente, devido a sua atualização, com as mudanças correspondentes nos *Conjunto de Itens* freqüentes. Segundo, pode haver itens completamente novos e *Conjunto de Itens* que se tornam freqüentes na atualização da base de dados.

Então, o principal objetivo do algoritmo é reduzir o número de *Conjunto de Itens* candidatos que são examinados na atualização da base de dados Δ, para decidir se são freqüentes ou não. Será apresentada a utilização armazenada em *P*, alguns *Conjunto de Itens* podem ser examinados apenas sobre d^+ e d^- em ordem a fim de descobrir o valor do suporte, reduzindo potencialmente a freqüência do custo computacional. Entretanto, uma importante característica na busca da base máxima empregada no algoritmo *ZigZag*, é não ser necessário

examinar cada conjunto de itens isolado, reduzindo em potencial o número dos candidatos encontrando todos os *Conjuntos de Itens*.

3.4.2. Gravação e Descarte de Transações

Nas gravações e descartes de transações, as novas transações são gravadas criando para cada item Δ, com seu *tidlist* em d^+, de modo que com isto, no fim do processo se tenha uma projeção vertical. Similarmente, descarta-se as transações obsoletas, examinando cada item em Δ criando seu *tidlist* em d^-, dando forma à projeção de decremento vertical na base de dados.

A projeção vertical para uma atualização da base de dados em Δ também deve ser atualizada com as novas transações. Portanto, para cada item em Δ aumenta-se seu *tidlist* com as transações em d^+ e deve-se remover os *tids* em d^-, o qual contém o item. Ou seja, para um item *X*, a atualização do *tidlist* se dá através de $\mathcal{L}_{\Delta}(X)=(\mathcal{L}_{D}(X) \cup \mathcal{L}_{d+}(X)) - \mathcal{L}_{d-}(X)$.

Este processo pode ser executado sempre que uma transação é adicionada ou removida da base de dados, onde é gravado dentro ou rejeitado das projeções. Esta estratégia tem diversas vantagens em curto prazo para a mineração incremental em regras de associação.

Primeiramente, o aumento dos *tidlists* torna-se fácil de determinar que a informação não é a operação atual da atualização, desde que os *tidlists* sejam sempre requisitados cronologicamente. Por esta razão, é também fácil determinar e rejeitar as informações obsoletas, executando a mineração em curto prazo. Segundo, é fácil identificar testes padrões emergentes e contrastes entre os *Conjunto de Itens* sobre as operações de mineração. Como d^+ reflete os eventos mais recentes, pode-se também verificar o impacto das decisões como personalizações, em aplicações de WEB, por exemplo [VEL02].

3.4.3. Determinando a Freqüência dos *Conjunto de Itens*

Neste momento, o algoritmo *ZigZag* após a gravação de novas transações e descartes dos itens obsoletos, determina a freqüência dos *Conjuntos de Itens* na atualização da base de dados Δ. O conjunto da freqüência de *Conjunto de Itens* em Δ é composta pela união dos conjuntos de *Conjunto de Itens* retidos e os conjuntos de *Conjunto de Itens* emersos, podendo com isto ser determinado pelos máximas freqüência dos *Conjunto de Itens* em Δ [GOU01].

Este é o passo em que o *ZigZag* emprega a procura para a máxima freqüência de *Conjunto de Itens* em Δ. Uma busca eficiente para satisfazer as duas propriedades são,

primeiramente a habilidade da execução computacionalmente rápida da freqüência, pois esta propriedade é associada com o custo de processamento dos *Conjunto de Itens* candidatos. Neste processo, para satisfazer a propriedade de aproximação geralmente utiliza-se a estrutura de compressão para a representação dos *tidlists*, como *diffsets* ou compressão de vetores [GOU01]. Segundo, a habilidade para remover facilmente os grandes filhos identificados do espaço de busca da consideração. Esta propriedade é associada a um número de candidatos gerados na procura. Para um pequenos número de candidatos gerados, a sua busca seria mais rápida. Com estas funcionalidades é que o algoritmo *ZigZag* obtém bons resultados em termos de desempenho computacional [VEL02].

3.4.4. Técnica da Freqüência Computacional

O algoritmo *ZigZag* possui uma particularidade, enquanto procura pela máxima freqüência de *Conjunto de Itens*, continuamente gera soluções parciais. E quando uma nova solução parcial surge, um novo conjunto de soluções deve ser computado para tornar possível este processo de extensões de soluções parciais. Assim, para se resolver o desempenho da freqüência computacional, o *ZigZag*, baseia-se na associatividade dos *Conjunto de Itens* são definidos como segue:

Dado X sendo um *k-Conjunto de Itens* de itens $X_1 \ldots X_k$, onde $X_i \in I$, $\mathcal{L}(X)$ é sua *tidlist* e $\delta(X) = |\mathcal{L}(X)|$ é o comprimento de $\mathcal{L}(X)$ e portanto o contador do *Sup* de X em um determinado banco de dados. Muitos *Conjunto de Itens* são obtidos pela junção dos itens individuais e o *Sup* é obtido pela intersecção do *tidlist* dos itens individuais (PARTHASARATHY et al., 1999). Desde que construiu a projeção vertical de d^+ e d^- e a atualização da projeção vertical para Δ, tem-se livre acesso aos *tidlists* para a procura de um item isolado em d^+, d^- e Δ, podendo assim, computar o *Sup* de alguns *Conjuntos de Itens* em d^+, d^- e Δ [VEL02].

O principal objetivo deste passo é maximizar os números de *Conjuntos de Itens* que tem a sua freqüência computada baseada apenas em d^+ e d^-. Assim, os *Conjunto de Itens* são chamados de *Conjunto de Itens* retidos e o contador de *Sup* em D são armazenados em P. Com isto, rapidamente descobre-se que um candidato é um *Conjunto de Itens* retido, então é armazenado os contadores de *Sup* para os *Conjunto de Itens* retidos usando uma tabela *hash*. Para um melhor entendimento, é descrita a rotina onde a freqüência para uma extensão é testada no *Combine_Set*.

Combine_Set(I_{l+1}, S_{l+1})
C = Ø
Para todo y ∈ S_{l+1} **faça**
    y' = y
Se $I_{l+1} \cup \{y\}$ for um *Conjunto de Itens* retido
    então $\sigma(y') = \sigma_D(I_{l+1} \cup \{y\}) + \sigma_{d+}(I_{l+1} \cup \{y\}) - \sigma_{d-}(I_{l+1} \cup \{y\})$
    senão, $\sigma(y') = \sigma_\Delta(I_{l+1} \cup \{y\})$
    Se $\sigma(y') \geq$ *minsup*
        então $C = C \cup \{y\}$
Retorne *C*

Figura 3.8: Geração Otimizada de *Conjuntos de Itens* [VEL02].

Uma explanação do algoritmo é uma solução parcial onde I_{l+1} é combinada com cada item *y* em um possível conjunto S_{l+1}, gerando assim uma extensão. Se a extensão for freqüente em *y*', então deve ser adicionada para a nova combinação de conjunto. Logo, para decidir se uma extensão é válida, deve-se verificar o seu suporte. Para executar de forma rápida computacionalmente uma freqüência, deve-se primeiramente verificar se a extensão $I_{l+1} \cup \{y\}$ é um *Conjunto de Itens* retido. Então como informado, a freqüência pode ser computada apenas em d^+, d^- e usando *P*, realçando dessa forma o processo computacional da freqüência [VEL02].

3.4.5. Técnica de poda

Nesta técnica, existem dois princípios para uma pesquisa eficiente usando *backtracking*[4]. São eles: 1 – é mais eficiente para fazer a próxima escolha de um ramo, para ser uma combinação de conjuntos com poucos itens. 2 – Pose-de remover o mais rápido possível um nó da árvore, da busca de um *backtracking*, podase eficazmente muitas filiais da consideração.

Reordenar os elementos da combinação de conjuntos corrente para atingir os objetivos, são meios eficazes de redução da busca de espaço. A heurística básica empregada por GENMAX [GRO98], é classificar a combinação de conjuntos no incremento ordenado para o suporte; este processo é usado para produzir uma combinação pequena de conjuntos no próximo nível, desde que os itens com valor de *Sup* baixo tenham uma possibilidade menor de produzir os *Conjunto de Itens* freqüentes.

[4]*Backtrack* – Recuo, utilizase esta nomenclatura devido a sua usabilidade comum neste tipo de literatura.

1. Guiado pela mineração prévia dos resultados, o algoritmo *ZigZag* requisita continuamente os elementos na combinação gerada dos conjuntos nos subseqüentes níveis de procura de espaços, já que os acessos livres para estimar o *Sup* de *Conjunto de Itens*, são gerados na procura, potencialmente capturando assim que possível algumas mudanças nas dependências entre a solução parcial e na combinação dos conjuntos. Isto permite que o *ZigZag* comece a requisitar os elementos para produzir ramos menores. De fato o *ZigZag* usa informações prévias para computar as *correlações mensuradas*, entre uma solução parcial e a combinação dos conjuntos de itens ao invés do *Sup*. A correlação pode ser usada para gerar dependências estatísticas para a presença e a ausência dos itens (i.e., itens da combinação de conjuntos) em um *Conjunto de Itens* (i.e., uma solução parcial), e este valor é computado por: $\frac{\sigma_D(\alpha \cup x)}{\sigma_D(\alpha) * \sigma_D(x)}$, em que α é uma solução parcial e x é um item dentro da combinação de conjunto. Se uma espécie combinar o conjunto em ordem crescente de correlação, a mesma produz uma pequena combinação de conjuntos no próximo nível, conduzindo a um elevado grau de poda. Sendo assim, a utilização desta otimização não afeta a exatidão do algoritmo desde a reordenação da combinação de conjunto, apenas muda a ordem em que a máxima freqüência dos *Conjunto de Itens* são gerados. Entretanto, estas otimizações melhoram drasticamente a eficiência da procura, reduzindo o numero de candidatos gerados, uma vez que a próxima escolha do ramo para explorar o *backtracking* na pesquisa, seja provável a boa aproximação da melhor escolha enquanto efetua a mineração em Δ. Em resumo, a principal idéia da eficiência da procura empregada no algoritmo *ZigZag*, decorre do fato de que o algoritmo elimina os ramos que são agrupados por um ramo já minerado com a máxima freqüência de *Conjunto de Itens*, e é muito provável que aqueles que são geradas anteriormente incluem um grande número de candidatos de *Conjunto de Itens*, que seriam gerados se a ordem em que estes padrões máximos gerados for diferente. A discussão é melhor entendida no algoritmo na figura 3.8.

Opt_Gen_Max(I_l, C_l, l)

Para cada $x \in C_l$ **faça**
$I_{l+1} = I \cup \{x\}$
$S_{l+1} = \{y : y \in C_l$ **e** $y > x\}$
se $I_{l+1} \cup S_{l+1}$ **então um super conjunto em MFI é retornado**
$C_{l+1} = Combine\_Set(I_{l+1}, S_{l+1})$
se C_{l+1} **está vazio**
então se I_{l+1} não tem super conjunto em MFI **então**
MFI = MFI $\cup$ I_{l+1}
senão
Ordenar C_{l+1} baseado na correlação entre I_{l+1} e cada item de
C_{l+1} Opt_Gen_Max(I_{l+1}, S_{l+1}, $l + 1$)

Figura 3.9: Busca Otimizada de MFI [VEL02].

3.4.6. Atualizando o Suporte (*Sup*) da Freqüência de *Conjunto de Itens*

Apenas determinando os MFI não é o suficiente para gerar todas as regras de associação, uma vez que não possuímos o *Sup* associado em cada subconjunto. Este é um dos principais aspectos negativos existente em toda a abordagem baseada em aproximação, uma vez que a busca completa ao longo de toda a base de dados é necessária para calcular a freqüência de cada subconjunto. Levando em consideração que em um ambiente dinâmico toda a base de dados é geralmente grande, este passo certamente pode ser consumir muito tempo.

Para evitar a busca em toda a base de dados a freqüência computacional de todos os *Sub Conjuntos*, novamente o algoritmo *ZigZag* faz uso de ambas as abordagens (máxima e incremental) aproximação, percorrendo a estrutura freqüente do *Conjunto de Itens* de forma invertida (*top-down*). Quebra cada *k-Conjunto de Itens* freqüente máximo em subconjunto *k* de tamanho (*k – 1*). Se a freqüência do subconjunto gerada for um *Conjunto de Itens* retido, a freqüência em Δ é computada apenas sobre d^+ e d^-, somando o seu *Sup* conhecido na contagem de *D*, para a contagem de *Sup* em d^+ e a subtração do *Sup* em d^-. Senão, se o subconjunto gerado é um *Conjunto de Itens* emerso, sua freqüência deve ser computada sobre toda base de dados atualizada. A aproximação incremental faz a enumeração invertida (*top-down*) de forma muito eficiente, desde que tenhamos os contadores de *Sup* em *D* para um possível número elevado de subconjunto gerado (i.e., todos os *Conjunto de Itens* retidos), evitando com isto, interseções de execução sobre a base de dados inteira para determiná-las. Com isto, precisamos apenas executar as interseções sobre d^- e d^+. Este processo itera gerando

subconjuntos menores, bem como a computação de suas freqüências até que não existam mais subconjuntos a serem verificados, em:

Update_Subsets_Support(θ)

Para cada *Sub Conjuntos* $\beta \subseteq \theta$ **faça**
  Se $\sigma(\beta)$ não for atualizado ainda **Então**
    Se β não for um *Conjunto de Itens* retido **Então** $\sigma(\beta) = \sigma_D(\beta) + \sigma_d + (\beta) - \sigma_d + (\beta)$
    Senão, $\sigma(\beta) = \sigma_\Delta(\beta)$
  Update_Subsets_Support(β)

Figura 3.10: Algoritmo de Processo de Atualização de *Suporte* [VEL02].

Apesar de todos os esforços desprendidos no algoritmo, para algumas bases de dados realmente densas, o processo de atualização incremental, pode torna-se impraticável a sua utilização, sabendo-se que o número de subconjunto pode ter seu crescimento muito elevado, degradando por completo a desempenho do algoritmo.

Neste cenário, uma possível estratégia é dedicado ao *relaxamento* da atualização do processo, como segue: Se a variação do suporte está acima de um *ponto inicial de relaxamento*, então ele é necessário para atualizar o subconjunto (*Sub Conjuntos*). Por um lado, se a quantidade populada do *Conjunto de Itens* não mudou muito, marca-se somente a sua popularidade de acordo com a mesma variação relativa observada em seu super-conjunto (S*uper Conjunto*), fazendo assim, o seu *suporte* como sendo uma boa aproximação para o seu real *suporte* em Δ. Mais tarde quando a superfície da mudança acontecer (i.e., exceder o ponto inicial de relaxamento), o conjunto, juntamente com o seu subconjunto, será atualizado com a sua freqüência exata, como segue:

ZigZag(*S, D,* d^+, d^-, *minsup, minconf*)
Procurar por *P e recuperar a informação coletada na mineração prévia.*
Procurar por d^+ *e* d^-. Cria o *Incremento* e *Decremento* na projeção vertical.
Atualizar a projeção Vertical de Δ.
F_1 = Item frequente em Δ.
Ordenar F_1 na ordem crescente de *suporte*.
MFI = ∅
Opt_Gen_Max(∅, F_1, *0*)
Para cada $\theta \in$ *MFI* ***faça***
  Update_Subsets_Support(θ)
Gerar as Regras de Associação com valor de ***confiança (Conf)*** ≥ ***a confiança mínima*** (***minconf***)

Figura 3.11: Descrição Geral do Algoritmo *ZigZag* [VEL02].

3.5. Considerações Finais do Capítulo

Neste capítulo foi apresentado dois algoritmos *APriori* e *ZigZag*. *APriori* inicialmente proposto em [AGR94] e o algoritmo *ZigZag*, inicialmente proposto por [Veloso et al. 2002A]. Ambos utilizados para a extração de regras de associação. Entretanto, cada algoritmo possui uma forma diferenciada para este fim, o *APriori* é utilizado de forma convencional, percorrendo toda a base de dados para a busca das regras. Já o algoritmo *ZigZag* é usado de forma incremental, o qual baseia-se na busca dos *MFIs*, utilizando o conhecimento descoberto nas fases anteriores para reduzir o custo da atualização dos *Conjuntos de Itens,* dando a este algoritmo uma maior eficiência em grandes bases de dados. A essência deste trabalho é efetuar um comparativo entre os algoritmos, afim de se avaliar a quantidade, qualidade e desempenho das regras extraídas pelos algoritmos, empregados aos valores de *suporte* (*Sup*) e *confiança (Conf)*. Entretanto, estes algoritmos não possuem medidas de avaliação da qualidade das regras, assunto este que será discutido no próximo capítulo.

Capítulo IV

Medidas de Avaliação de Regras

Neste capítulo são apresentadas as principais Medidas de Avaliação das Regras as quais representam um importante papel neste trabalho, pois é a partir das medidas de avaliação de regras que se faz uma análise qualitativa das regras extraídas pelos algoritmos.

4.1. Introdução à Medidas de Avaliação de Regras

O uso comum das medidas de avaliação de regras empregadas em trabalhos voltados para algoritmos de Classificação, bem como a sua eficácia comprovada em diversos trabalhos, permitiu que se adotasse como referência a metodologia empregada no trabalho de Gomes(2002), o qual efetuou um estudo da qualidade das regras de classificação. Tendo em vista que as medidas aplicadas são compatíveis e validam diversos tipos de regras, tais como Classificação e Associação, tema deste trabalho.

4.2. Regras

Usualmente utilizados nas literaturas, os sistemas de aprendizagem de máquina que induzem regras de conhecimento são denotados como Corpo → Cabeça, porém serão empregados neste trabalho os termos *Body* → *Head*, terminologia usual nos trabalhos voltados para extração de regras, denotados como $B \rightarrow H$.

As medidas de avaliação de regras tem o objetivo de dar uma indicação de forma hipotética da força de associação entre *Head* e *Body* demonstrada por uma regra. Assume-se que, dado um conjunto de exemplos identificados, pode-se determinar para cada possível *Head* e *Body*, a verdade ou não de *Head* e a verdade ou não de *Body* para um determinado exemplo.

A indução das regras preditivas podem ocorrer através do aprendizado proposicional ou de primeira ordem. Em se tratando de regras proposicionais, define-se *Body* como uma conjunção de pares atributo-valor e *Head* é designado pelo atributo previsto. Já no caso de aprendizado de primeira ordem, onde as regras são cláusulas *Prolog* determina-se que *Head* é um literal positivo e *Body* é uma conjunção de literais positivos e/ou negativos.

Um exemplo só pode ser coberto por uma regra $B \rightarrow H$ se *Body* for verdadeiro para esse exemplo. Já nas regras proposicionais, um exemplo só é coberto quando satisfaz as condições de uma regra , onde todas as condições de uma regra são avaliadas verdadeiras e de acordo com a descrição do exemplo.

> "Nas regras de primeira ordem, os exemplos descritos pelo(s) átomo(s) são combinados com a cabeça da regra, determinando assim a substituição θ pela qual as variáveis na cabeça da regra sejam substituídas pelos termos que constarem na descrição do exemplo. Assim a regra só pode cobrir o exemplo, se e somente se *Body* θ for avaliado como verdadeiro" [GOM02].

Através deste estudo, fica claro que este trabalho se dá através de regras induzidas de aprendizado proposicional ou de primeira ordem.

4.3. Matriz de Confusão

Uma matriz de confusão é considerada uma base padrão para se calcular as medidas de avaliação de hipóteses em problemas de classificação. Este procedimento também pode ser usado para avaliar a qualidade de regras de Associação. Assim, aplica-se a matriz de confusão ao conjunto como um todo, isto é, o conjunto é tratado como uma caixa preta. Isto difere da *tabela de contingência*, que refere-se a uma única regra que constitui o conjunto simbólico.

O objetivo de uma matriz de confusão é mostrar o número de previsões corretas em relação às esperadas para cada regra. Levando em consideração os problemas de classificação binária (composta por duas classes), denotadas como "+" e "-", as escolhas são estruturadas com a finalidade de predizer quando há ocorrência ou não-ocorrência de um evento [BAR00].

Dada uma regra representada por R em um exemplo $T_i = (x_i, y_i)$ com sua respectiva classe denotada por y_i. Assim, pode-se aplicar a regra ao exemplo a fim de se comparar o resultado previsto pela cabeça H da regra com a sua verdadeira classe y_i do exemplo descrito. Com isto, faz com que esta comparação resulte em uma das quatro possíveis situações:

1. Se B e H são verdade, o exemplo será coberto corretamente pela regra;
2. Se B é verdade mas H é falso, então o exemplo é coberto pela regra de forma incorreta;

3. *B* é falso e *H* também é falso, quando o exemplo não for coberto pela regra e a classe prevista por *H* da regra não for a mesma classe y_i do exemplo.
4. Se *B* falso e *H* é verdade, então o exemplo é coberto pela regra de forma incorreta.

Desta forma, ao aplicar a regra a um conjunto de testes *T*, que contenha *n* exemplos, deriva-se a cada regra a sua matriz de contingência conforme a Tabela 4.1.

Tabela 4.1: Tabela de Contingência para Uma Regra[GOM02].

	H	$\bar{H}$	
B	hb	$\bar{h}b$	b
$\bar{B}$	$h\bar{b}$	$\overline{hb}$	$\bar{b}$
	h	$\bar{h}$	n

hb = é o número de exemplos para os quais *H* é verdade e *B* é verdade;
$\bar{h}b$ = é o número de exemplos para os quais *H* é falso e *B* é verdade;
$h\bar{b}$ = número de exemplos para os quais *H* é verdade e *B* é falso;
$\overline{hb}$ = número de exemplos para os quais *H* é falso e *B* é falso;
b = número de exemplos para os quais *B* é verdade;
$\bar{b}$ = número de exemplos para os quais *B* é falso;
h = número de exemplos para os quais *H* é verdade;
$\bar{h}$ = número de exemplos para os quais *H* é falso;
n = número total de exemplos.

Em que *x* é a cardinalidade do conjunto *X*, isto é, $x = |X|$. Assim, *h* denota a cardinalidade do conjunto *H*, ou seja, $h = |H|$. O *h* representa o número de exemplos para os quais o *H* da regra é verdade. Associada à cardinalidade de *x*, a freqüência relativa f_x é utilizada como uma estimativa da probabilidade *P*(*X*), ou seja, $P(X) = f_x = \frac{x}{n}$. Como por exemplo, ao considerar a cardinalidade *hb*, a probabilidade *P*(*HB*) pode ser determinada da seguinte forma:

$$P(HB) = f_{bh} = \frac{hb}{n} \tag{4.1}$$

Analogamente, podem ser determinados os valores das probabilidades da seguinte forma: $P(\bar{H}B), P(\overline{HB}) e P(H\ \bar{B})$. Através destas probabilidades, os valores de $P(B), P(\bar{B}), P(H) e P(\bar{H})$ podem ser determinadas da seguinte forma:

$$P(B)=P(\bar{H}B)+P(HB)\ ou\ P(\bar{B})=P(H\ \bar{B})+P(\overline{HB}) \tag{4.2}$$

Essas medidas de avaliação de regras, utilizadas neste trabalho estão definidas em forma de estimativas de probabilidade ou freqüências relativas, oriundas da tabela de contingência. Com isto, pode-se observar que a tabela de contingência é uma generalização da matriz de confusão. Embora apresentada como referência, a Matriz de Confusão não será empregada neste trabalho.

4.4. Medidas de Avaliação de Regras

Diversas medidas foram pesquisadas e implementadas com o objetivo de auxiliar o usuário no entendimento e utilização do conhecimento adquirido pelos sistemas de aprendizagem de máquina. É através da tabela de contingência que se deriva a definição das medidas de avaliação de regras, tendo em vista que as mesmas são dadas em termos de freqüências relativas, onde algumas destas medidas foram propostas inicialmente por Piatetsky-Shapiro (1991), o qual detém a capacidade de trazer consigo alguma novidade.

4.4.1. Exemplo para Avaliação das Regras

Com o objetivo de ilustrar as medidas de avaliação das regras apresentadas a seguir, será considerado o exemplo de conjunto ilustrado na Tabela 4.2, adaptado de Quinlan (1993) por BARANAUSKAS e Monard (2000). Os conjuntos de dados apresentados, contém informações de medidas diárias de condições meteorológicas e uma classificação se o dia é apropriado para jogar golfe, determinada por (“go”) ou não (“dont_go”), exemplo bastante divulgado didaticamente no meio acadêmico, e em Gomes (2002) através do exemplo: Exemplos de Treinamento *Voyage.*

Para demonstrar um melhor entendimento, os atributos de cada exemplo são:

- ***outlook:*** contém os valores *sunny*, *overcast* ou *rain*;
- ***temperature:*** é um valor numérico, com a finalidade de indicar a temperatura em graus *Celsius*;
- ***humidity:*** é uma valor numérico indicando a umidade relativa do ar;

- ***windy:*** assume os valores *yes* ou *no*.

Através do conjunto de dados contidos na Tabela 4.2 como um conjunto de treinamento, foram induzidas as regras conforme demonstrada na Figura 4.1, a partir da qual foram selecionadas três das regras apresentadas.

Cada uma das três regras foi avaliada através do conjunto de testes descritos na Tabela 4.4. A tabela de contingência para cada uma das três regras está descrita na tabela 4.5.

4.5. Medidas Genéricas das Regras

Seguindo o exemplo descrito anteriormente (*voyage*), são apresentadas as figuras e as tabelas como segue: Tabela 4.2: Conjunto de Exemplos de Treinamento *voyage*, Figura 4.1: Regras Induzidas a partir do Conjunto de Exemplos de Treinamento *voyage*, Tabela 4.3: Conjunto de Exemplos de Teste *voyage* e Figura 4.2: Regras Selecionadas e Suas Tabelas de Contingência. Através das tabelas e figuras são discutidas todas as medidas como contribuição a este trabalho. Entretanto, são utilizados somente as medidas aplicadas às regras extraídas dos Algoritmos *APriori* e *ZigZag*.

Tabela 4.2: Conjunto de Exemplos de Treinamento *voyage* [BARANAUSKAS et. al, 2000].

Outlook	***Temperature***	***Humidity***	***Windy***	***Voyage?***
Rain	27	95	No	Go
Rain	20	70	Yes	dont_go
Rain	23	80	Yes	dont_go
Rain	25	81	No	Go
Rain	21	80	No	Go
Sunny	25	72	Yes	Go
Sunny	21	79	Yes	dont_go
Sunny	26	70	No	Go
Sunny	27	92	No	dont_go
Sunny	30	88	No	dont_go
Overcast	23	90	Yes	Go
Overcast	29	78	No	Go
Overcast	19	65	Yes	dont_go
Overcast	26	75	No	Go
Overcast	20	87	Yes	dont_go

```
IF outlook = overcast
THEN CLASS = go

IF outlook = sunny
                AND humidity ≤ 78
THEN CLASS = go

IF outlook = sunny
                AND humidity > 78
THEN CLASS = dont_go

IF outlook = rain
                AND windy = no
THEN CLASS = go

CLASS = go
```

Figura 4.1: Regras Induzidas a partir do Conjunto de Exemplos de Treinamento *voyage* (BARANAUSKAS et al., 2000).

Tabela 4.3: Conjunto de Exemplos de Teste *voyage* (BARANAUSKAS et al., 2000).

Outlook	***Temperature***	***Humidity***	***Windy***	***Voyage?***
Sunny	25	72	Yes	Go
Sunny	28	91	Yes	dont_go
Sunny	22	70	No	Go
Sunny	23	95	No	dont_go
Sunny	30	85	No	dont_go
Overcast	23	90	Yes	Go
Overcast	29	78	No	Go
Overcast	19	65	Yes	dont_go
Overcast	26	75	No	Go
Overcast	20	87	Yes	Go
Rain	22	95	Yes	Go
Rain	19	70	Yes	dont_go
Rain	23	80	Yes	dont_go
Rain	25	81	No	Go
Rain	21	80	No	go

R1	IF outlook = overcast THEN CLASS = go	4 1 5 5 5 10 9 6 15
R2	IF outlook = sunny AND humidity > 78 THEN CLASS = dont_go	3 0 3 3 9 12 6 9 15
R3	IF outlook = rain AND windy = no THEN CLASS = go	2 0 2 7 6 13 9 6 15

Figura 4.2: Regras Selecionadas e Suas Tabelas de Contingência (BARANAUSKAS et al., 2000).

Definição 4.4.1: *Precisão*:

$$Acc(B \rightarrow H) = P(H|B) = P\frac{(HB)}{P(B)} = \frac{f_{hb}}{f_b} = \frac{hb}{b} \tag{4.3}$$

A *precisão* de uma regra (*consistência ou confidência*) é uma medida do quanto uma regra é específica para o problema. Então, a precisão pode ser definida como a probabilidade condicional de *H* ser Verdade dado que *B* é Verdade [LAV99].

Definição 4.4.2: *Erro*:

$$Err(B \rightarrow H) = P(\bar{H}|B) = \frac{f_{\bar{h}b}}{f_b} \tag{4.4}$$

O *erro* de uma regra é definido como sendo (1 – *Acc*(R)), então pode-se afirmar que quanto maior o erro, menos precisamente a regra cobre corretamente os exemplos da sua classe. Portanto, pode-se melhor definir o erro como:

$$Err(R) = 1 - Acc(R) = 1 - P(H|B) = \frac{1 - P(HB)}{P(B)} = \frac{P(\bar{H}\ B)}{P(B)} = P(\bar{H}|B) \tag{4.5}$$

Considerando as regras R_1, R_2 e R_3 apresentadas na Figura 4.2, para a aplicação das medidas de precisão e erro, ambas resultam respectivamente em:

$$Acc(R_1)=\frac{4}{5}=0.800 \qquad Err(R_1)=1-Acc(R_1)=0.200$$

$$Acc(R_2)=\frac{3}{3}=1.000 \qquad Err(R_2)=1-Acc(R_2)=0.000$$

$$Acc(R_3)=\frac{2}{2}=1.000 \qquad Err(R_3)=1-Acc(R_3)=0.000$$

Pode-se dizer que a definição da *precisão de uma regra* tem o objetivo de avaliar regras individuais e, por essa razão, tende a favorecer a precisão de exemplos positivos. O que a difere da *precisão do conjunto*, tendo em vista que a *precisão de uma regra*, constitui a hipótese, definida em função da matriz de confusão.

A Tabela 4.4 apresenta um modelo de conjunto de exemplos de teste, com as regras *"go"* como positiva e a regra "*dont_go"* como negativa.

Tabela 4.4: Matriz de Confusão [GOM02]

Regra	**Preditos como "*go*"**	**Preditos como "*dont_go*"**
"go"	*Verdadeiros positivos* ***Tp*** = 8	*Falsos negativos* ***Fn***=1
"dont_go"	*Falsos positivos* ***Fp*** = 1	*Verdadeiros negativos* ***Tn*** = 5

A precisão do conjunto de regras é utilizada como medida padrão para avaliar hipóteses, representada pela seguinte fórmula:

$$AC(h)=Tp+\frac{Tn}{n} \tag{4.6}$$

Aplicada ao exemplo da matriz de confusão da Tabela 4.4, o valor da precisão da hipótese ou precisão da regra é:

$$AC(h)=8+\frac{5}{15}=0.867$$

Desta forma, pode-se notar que a precisão de uma regra, é de fato o mesmo que a *confidência* no aprendizado das regras de associação, sabendo que as regras de associação admitem que *Head* seja uma conjunção de atributos. Então, a precisão de uma regra pode ser

usada também para medir a *confiança* de uma regra na predição de casos positivos, tendo em vista que ela mede o quão foram corretos os resultados retornados.

Definição 4.4.3: *Confiança Negativa:* $NegRel(B \rightarrow H) = P(\bar{H} \mid \bar{B})$

A *confiança negativa* é o correspondente a precisão, todavia, para os exemplos que não são cobertos pela regra. Define-se como a probabilidade condicional de *H* ser falso dado que *B* também é falso, como apresentado na equação:

$$NegRel(R) = P(\bar{H} \mid \bar{B}) = \frac{P(\overline{HB})}{P(\bar{B})} = \frac{f_{\overline{hb}}}{f_{\bar{b}}} = \frac{\overline{hb}}{\bar{b}} \tag{4.7}$$

Para problemas de classificação binária, a confiança negativa da hipótese é computada da seguinte forma:

$$NegRel(h) = \frac{Tn}{Tn + Fn} \tag{4.8}$$

Aplicado ao exemplo das regras R_1, R_2 e R_3 da Figura 4.2, a confiança negativa de cada regra resulta respectivamente em:

$$NegRel(R_1) = \frac{5}{10} = 0.500$$

$$NegRel(R_2) = \frac{9}{12} = 0.750$$

$$NegRel(R_3) = \frac{6}{13} = 0.461$$

De acordo com a matriz de confusão da Tabela 4.4, o valor da confiança negativa da hipótese é:

$$NegRel(\mathrm{h}) = \frac{5}{5+1} = 0.833$$

Definição 4.4.4: *Sensitividade (Completeza ou Recall[5]):* $Sens(B \rightarrow H) = P(B|H)$

É uma medida do número (relativo) de exemplos da classe prevista em *H* cobertos pela regra. É semelhante ao *recall* de casos positivos usados em recuperação de informação. Esta medida define-se como a probabilidade condicional de *B* ser verdade dado que *H* é verdade. Quanto maior a sensitividade, mais exemplos são cobertos pela regra.

$$Sens(R) = P(B|H) = \frac{P(HB)}{P(H)} = \frac{f_{hb}}{f_h} = \frac{hb}{h} \tag{4.9}$$

Nos casos de classificação binária, conforme abordado anteriormente, a sensitividade da hipótese mede a fração de verdadeiros positivos que são corretamente classificados, isto é, um *recall* de casos positivos, dado por:

$$Sens(\mathrm{h}) = \frac{Tp}{Tp + Fn} \tag{4.10}$$

Definição 4.4.5: *Especificidade:* $Spec(B \rightarrow H) = P(\bar{B}|\bar{H})$

A *especificidade* de uma regra é o correspondente a completeza, todavia, para os exemplos que não são cobertos pela regra *R*. É definida como a probabilidade condicional de *B* ser falso dado que *H* é falso, ou seja:

$$Spec(R) = P(\bar{B}|\bar{H}) = \frac{P(\overline{HB})}{P(\bar{H})} = \frac{f_{\overline{hb}}}{f_{\bar{h}}} = \frac{\overline{hb}}{\bar{h}} \tag{4.11}$$

O *recall* dos casos negativos na recuperação de informação é dado por:

$$Sens(\mathrm{h}) = \frac{Tn}{Tn + Fn} \tag{4.12}$$

Considerando as regras R_1, R_2 e R_3 da Figura 4.2, a sensitividade e a especificidade de cada regra resulta em:

[5] *Recall* – Chamada de volta, recolha de produtos defeituosos, mandar de volta. Esta é uma expressão usual computacionalmente para a Definição: *Sensitividade.*

$$Sens(R_1)=\frac{4}{9}=0.444 \qquad Spec(R_1)=\frac{5}{6}=0.833$$

$$Sens(R_2)=\frac{3}{6}=0.500 \qquad Spec(R_2)=\frac{9}{9}=1.000$$

$$Sens(R_3)=\frac{2}{9}=0.222 \qquad Spec(R_3)=\frac{6}{6}=1.000$$

Para a matriz de confusão da Tabela 4.4, o valor da sensitividade e da especificidade da hipótese é:

$$Sens(h)=\frac{8}{8+1}=0.889$$

$$Spec(h)=\frac{5}{5+1}=0.833$$

Definição 4.4.6: *Cobertura:* $Cov(B \rightarrow H)=P(B)$

A *cobertura de uma regra* é uma medida do número (relativo) de exemplos cobertos pela regra *R*. Como tal, é uma medida de generalidade da regra. É definida como a probabilidade de *B* ser verdade, isto é, quanto maior a cobertura, maior o número de exemplos cobertos pela regra *R*.

$$Cov(R)=P(B)=f_b=\frac{b}{n} \tag{4.13}$$

Definição 4.4.7: *Suporte:* $Sup(B \rightarrow H)=P(HB)$

O suporte de uma regra é uma medida do número (relativo) de exemplos cobertos corretamente pela regra *R*. É definido como a probabilidade de *H* e *B* serem verdade. Quanto maior o suporte, maior o número de exemplos da regra em questão que são cobertos corretamente pela regra R.

$$Sup(R)=P(HB)=f_{hb}=\frac{hb}{n} \tag{4.14}$$

Cabe notar que, diferentemente das medidas anteriores, o suporte é simétrico em *H* e *B*, ou seja, $Sup(R)=P(HB)=P(BH)$.

Considerando as regras R_1, R_2 e R_3 da Figura 4.2, a cobertura e o suporte de cada regra resulta respectivamente em:

$$Cov(R_1)=\frac{5}{15}=0.333 \qquad Sup(R_1)=\frac{4}{15}=0.267$$

$$Cov(R_2)=\frac{3}{15}=0.200 \qquad Sup(R_2)=\frac{3}{15}=0.200$$

$$Cov(R_3)=\frac{2}{15}=0.133 \qquad Sup(R_3)=\frac{2}{15}=0.133$$

Definição 4.4.8: *Novidade:* $Nov(B \rightarrow H)=P(HB)-P(H)P(B)$

Esta medida tem a intenção de mostrar o quanto uma regra é interessante, nova ou fora do comum. A medida *Novidade*, foi inicialmente apresentada Piatetsky-Shapiro (1991), como sendo uma medida *PS*(*R*). Uma medida de novidade é definida através da estrutura da tabela de contingência, Tabela 4.1.

Então, define-se uma regra $B \rightarrow H$ como sendo nova se a probabilidade de *B* e *H* ocorrerem juntos e não puder ser inferida pelas probabilidades de *B* e *H* isoladamente, ou seja, *B* e *H* não são estatisticamente independentes.

Esta medida é obtida comparando o valor esperado *P(HB)* com os valores de *P(H)* e P(B). Quanto mais o valor esperado for diferente do valor observado, maior será a probabilidade de existir uma correlação verdadeira e inesperada entre *B* e *H*. Pode ser demonstrado que -0,25 ≤ *Nov*(*R*) ≤ 0,25, e quanto maior um valor positivo, mais forte é a associação entre *B* e $\bar{H}$. Se *Nov*(*R*) = 0, então a regra não é interessante, nem nova ou fora do comum.

Assim sendo, a novidade de uma regra será:

$$Nov(R)=P(HB)-P(H)P(B)=f_{hb}-f_h \cdot f_b=\frac{1}{n}\left(hb-\frac{h \cdot b}{n}\right) \qquad (4.15)$$

Para as regras R_1, R_2 e R_3 da Figura 4.2, a novidade de cada regra resulta respectivamente em:

$$Nov(R_1)=\frac{4}{15}-\left(\frac{9}{15}\cdot\frac{5}{15}\right)=0.067$$

$$Nov(R_2) = \frac{3}{15} - \left(\frac{6}{15} \cdot \frac{3}{15}\right) = 0.120$$

$$Nov(R_3) = \frac{2}{15} - \left(\frac{9}{15} \cdot \frac{2}{15}\right) = 0.053$$

Definição 4.4.9: *Satisfação:* $Sat(B \rightarrow H) = \frac{P(\bar{H}) - P(\bar{H}|B)}{P(\bar{H})}$

A *satisfação de uma regra* é medida pelo aumento relativo na precisão entre a regra $B \rightarrow$ verdade e a regra $B \rightarrow H$. Esta medida tem a capacidade de promover um equilíbrio entre regras com diferentes condições e conclusões.

$$Sat(R) = \frac{P(\bar{H}) - P(\bar{H}|B)}{P\,\bar{H}} = \frac{f_{\bar{h}} - \frac{f_{\bar{h}b}}{f_b}}{f_{\bar{h}}} = 1 - \left(\frac{\bar{h}b}{b} \cdot \frac{n}{\bar{h}}\right) \tag{4.16}$$

É relativamente simples demonstrar que $Sat(B \rightarrow H) = \frac{P(H|B) - P(H)}{1 - P(H)}$, uma vez que $P(\bar{H}) - P(\bar{H}|B) = (1 - P(H)) - (1 - P(H|B)) = P(H|B) - P(H)$. Observa-se que *Sat*(*R*) está relacionada diretamente com *Acc*(*R*), pois *Sat*(*R*) = *sss*[6] *Acc*(*R*) = 1. Porém, diferente da precisão de uma regra, a *satisfação* leva em conta o total da tabela de contingência, tratando a associação de diferentes *H* para um mesmo *B* de uma regra, transformando-se na regra mais indicada para as tarefas de descoberta de conhecimento [GOM02].

Considerando as regras R_1, R_2 e R_3 da Figura 4.2, a satisfação de cada regra resulta respectivamente em:

$$Sat(R_1) = \left(\frac{6}{15} - \frac{1}{5}\right) \subseteq \frac{6}{15} = 0.500$$

$$Sat(R_2) = \left(\frac{9}{15} - \frac{0}{3}\right) \subseteq \frac{9}{15} = 1.000$$

$$Sat(R_3) = \left(\frac{6}{15} - \frac{0}{2}\right) \subseteq \frac{6}{15} = 1.000$$

[6] *sss* – Se e somente se.

Para uma melhor avaliação da aplicação das regras, a Tabela 4.5 demonstra todos os valores das medidas genéricas de avaliação de regras obtidas para cada uma das três regras R_1, R_2 e R_3 da Figura 4.2, utilizadas para ilustração [LVR99].

4.6. Medidas Relativas de Avaliação de Regras

Nesta seção é apresentada a utilidade de um conjunto de medidas para o descobrimento de conhecimento proposto por Lavrac et. al, (1999), demonstrando uma síntese das medidas aplicadas. A medida que inspirou a elaboração desta síntese foi a medida *novidade*, pelo fato de ser considerada uma medida *relativa*, uma vez que compara o suporte esperado sobre a suposição de independência estatística, demonstrada na Definição: 4.4.7 da seção anterior.

Tabela 4.5: Valores das Medidas Genéricas de Avaliação das Regras [GOM02]

Medidas	***Regras***		
	R_1	R_2	R_3
Acc ($B \rightarrow H$)	0.800	1.000	1.000
Err($B \rightarrow H$)	0.200	0.000	0.000
NegRel($B \rightarrow H$)	0.500	0.750	0.461
Sens($B \rightarrow H$)	0.444	0.500	0.222
Spec($B \rightarrow H$)	0.833	1.000	1.000
Cov($B \rightarrow H$)	0.333	0.200	0.133
Sup($B \rightarrow H$)	0.267	0.200	0.133
Nov($B \rightarrow H$)	0.067	0.120	0.053
Sat($B \rightarrow H$)	0.500	1.000	1.000

Definição 4.4.10: *Precisão Relativa: RAcc(B→H) = P(H|B) – P(H)*

A *precisão relativa* de uma regra $R = B \rightarrow H$ é o ganho de precisão obtido em relação a uma regra padrão *true* $\rightarrow H$, isto é, uma regra que avalia B como verdade para todos os exemplos. Neste caso, uma regra só interessa se melhorar a precisão da regra padrão. Então a medida relativa é definida como:

$$RAcc(R) = P(H|B) - P(H) = \frac{f_{hb}}{f_b} - f_h = \frac{hb}{b} - \frac{h}{n} \tag{4.17}$$

Uma outra forma de ver a precisão relativa de uma regra é como uma medida de utilidade de conexão de um corpo *B* com uma dada cabeça *H*. Assim, considerando as regras R_1, R_2 e R_3 da Figura 4.2, a precisão relativa de cada regra resulta respectivamente em:

$$RAcc(R_1) = \frac{4}{5} - \frac{9}{15} = 0.200$$

$$RAcc(R_2) = \frac{3}{3} - \frac{6}{15} = 0.600$$

$$RAcc(R_3) = \frac{2}{2} - \frac{9}{15} = 0.400$$

Definição 4.4.11: *Confiança Negativa Relativa: RNegRel*$(B \rightarrow H) = P(\bar{H} \mid \bar{B}) - P(\bar{H})$

A *confiança negativa relativa* é o análogo à *precisão relativa* para os exemplos que não são cobertos pela regra. Nesse caso, a regra padrão é *falso* $\rightarrow \bar{H}$, ou seja:

$$RAcc(R) = P(H \mid B) - P(H) = \frac{f_{hb}}{f_b} - f_h = \frac{hb}{b} - \frac{h}{n} \tag{4.18}$$

Definição 4.4.12: *Sensitividade Relativa: RSens*$(B \rightarrow H) = P(B \mid H) - P(B)$

A *sensitividade relativa* tem o objetivo de medir o ganho de sensitividade obtido em relação à sensitividade de regra padrão é $B \rightarrow$ *verdade*, isto é, uma regra que avalia *H* como verdade para todos os exemplos.

$$RNegRel(R) = P(H \mid B) - P(H) = \frac{f_{\overline{hb}}}{f_b} - f_h = \frac{\overline{hb}}{\bar{b}} - \frac{\bar{h}}{n} \tag{4.19}$$

Definição 4.4.13: *Especificidade Relativa: RSpec*$(B \rightarrow H) = P(\bar{B} \mid \bar{H}) - P(\bar{B})$

Análoga à *medida sensitividade* relativa para os exemplos que não são cobertos pela regra. Então neste caso, a regra padrão é $\bar{B} \rightarrow$ *verdade*, ou seja:

$$RSens(R) = P(B \mid H) - P(B) = \frac{f_{hb}}{f_h} - f_b = \frac{hb}{h} - \frac{b}{n} \tag{4.20}$$

Desta forma, deve-se observar que a visão utilizada na construção dessas medidas, primeiramente desenvolve o *B* da regra para depois encontrar um *H* apropriado para esse *B*. Considerando as regras R_1, R_2 e R_3 da Figura 4.2, da *confiança negativa*, a sensitividade relativa e a especificidade relativa de cada regra resultam respectivamente em:

$$RNegRel(R_1) = \frac{5}{10} - \frac{6}{15} = 0.100$$

$$RNegRel(R_2) = \frac{9}{12} - \frac{9}{15} = 0.150$$

$$RNegRel(R_3) = \frac{6}{13} - \frac{6}{15} = 0.061$$

$$RSens(R_1) = \frac{4}{9} - \frac{5}{15} = 0.111$$

$$RSens(R_2) = \frac{3}{6} - \frac{3}{15} = 0.300$$

$$RSens(R_3) = \frac{2}{9} - \frac{2}{15} = 0.089$$

$$RSpec(R_1) = \frac{5}{6} - \frac{10}{15} = 0.167$$

$$RSpec(R_2) = \frac{9}{9} - \frac{12}{15} = 0.200$$

$$RSpec(R_3) = \frac{6}{6} - \frac{13}{15} = 0.133$$

O ponto mais importante referente as *medidas relativas* é que elas dispõem mais informações sobre a utilidade de uma regra do que as informações fornecidas pelas medidas absolutas. Como por exemplo, se em uma tarefa de predição a precisão de uma regra for menor que a freqüência relativa da regra que a prediz, então tal regra tem um desempenho ruim, independente da sua precisão absoluta [GOM02].

Entretanto, existe um complicador com relação a precisão relativa: pois é fácil obter uma alta precisão relativa para regras específicas, isto é, regras com baixa generalidade (generalidade = P(B)). Para que se possa contornar esse complicador, uma variante é proposta em Lavrac et al., (1999), onde é atribuído um peso para cada medida. Este peso promove um

balanceamento ("*trade off*") entre a generalidade e a relatividade destas medidas, conforme descrito a seguir.

4.7. Medidas Relativas de Regras com Peso

Definição 4.4.14: *Precisão Relativa com Peso:*

$$WRAcc(R) = P(B)(P(H \wedge B) - P(H)) = f_b\left(\frac{f_{hb}}{f_b} - f_h\right) = \frac{b}{n}\left(\frac{hb}{b} - \frac{h}{n}\right) \tag{4.21}$$

A precisão relativa com peso efetua um balanceamento entre generalidade e precisão relativa. Esta medida é muito utilizada na literatura como uma medida de ganho, a qual tem a função de avaliar a utilidade de um *literal L* considerado para estender o corpo de *B* de uma regra: $\frac{P(BL)}{P(B)}(P(H|BL) - P(H|B))$.

Desta forma, considerando as regras R_1, R_2 e R_3 da Figura 4.2, a precisão relativa com o peso de cada regra resulta respectivamente em:

$$WRAcc(R_1) = \frac{5}{15} \cdot \left(\frac{4}{5} - \frac{9}{15}\right) = 0.067$$

$$WRAcc(R_2) = \frac{3}{15} \cdot \left(\frac{3}{3} - \frac{6}{15}\right) = 0.120$$

$$WRAcc(R_3) = \frac{2}{15} \cdot \left(\frac{2}{2} - \frac{9}{15}\right) = 0.053$$

Objetivando disponibilizar um melhor entendimento de medidas de avaliação de regras, demonstra-se na seqüência o resultado, através de um teorema.

Teorema 4.4.1: *Racc(R) = Nov(R)*

Prova.

$$\begin{aligned} WRAcc(B \rightarrow H) &= P(B)(P(H|B) - P(H)) \\ &= (P(B)P(H|B) - P(B)P(H)) \\ &= P(HB) - P(H)P(B) \\ &= Nov(B \rightarrow H). \end{aligned}$$

O teorema 4.4.1, tem as seguintes implicações:

1. Regras com alta precisão relativa com peso também têm alta novidade e vice-versa;
2. Alta novidade é alcançada pelo balanceamento entre a generalidade e a precisão da regra, obtido em comparação com a regra trivial *true* → *H*. Assim pode-se afirmar que, uma ter uma alta precisão relativa não é o suficiente para concluir que uma regra seja interessante, uma vez que é necessário também que ela seja suficientemente genérica.

A importância deste teorema é estabelecer um elo entre as medidas de avaliação de regras. Deve-se observar também, que a precisão relativa com peso é uma *medida fundamental de avaliação de regras*, pois, dispõe de um balanceamento entre precisão e outras medidas supervisionadas, como será apresentado na seqüência.

Definição 4.4.15: *Confiança Negativa Relativa com Peso:*

$$WRNegRel(R) = P(\bar{B})(P(\bar{H}|\bar{B}) - P(\bar{H})) = f_{\bar{b}}\left(\frac{f_{\bar{h}\bar{b}}}{f_{\bar{b}}} - f_{\bar{h}}\right) = \frac{\bar{b}}{n}\left(\frac{\overline{hb}}{\bar{b}} - \frac{\bar{h}}{n}\right) \tag{4.22}$$

O uso do peso $P(\bar{B})$ é motivado pelo fato da maioria das regras muito gerais terem uma confiança negativa alta.

Definição 4.4.16: *Sensitividade Relativa com Peso:*

$$WRSens(R) = P(H)(P(B|H) - P(B)) = f_h\left(\frac{f_{hb}}{f_h} - f_b\right) = \frac{h}{n}\left(\frac{hb}{h} - \frac{b}{n}\right) \tag{4.23}$$

Definição 4.4.17: *Especificidade Relativa com Peso:*

$$WRSpec(R) = P(\bar{H})(P(\bar{B}|\bar{H}) - P(\bar{B})) = f_{\bar{h}}\left(\frac{f_{\bar{h}\bar{b}}}{f_{\bar{h}}} - f_{\bar{b}}\right) = \frac{\bar{h}}{n}\left(\frac{\overline{hb}}{\bar{h}} - \frac{\bar{b}}{n}\right) \tag{4.24}$$

Da mesma forma que nos casos anteriores, os pesos têm como objetivo a prevenção de soluções simples. Considerando as regras R_1, R_2 e R_3 da Figura 4.2 a confiança negativa, a sensitividade e especificidade relativas com peso de cada regra resultam respectivamente em:

$$WRNegRel(R_1) = \frac{10}{15} \cdot \left(\frac{5}{10} - \frac{6}{15} \right) = 0.067$$

$$WRNegRel(R_2) = \frac{12}{15} \cdot \left(\frac{9}{12} - \frac{9}{15} \right) = 0.120$$

$$WRNegRel(R_3) = \frac{13}{15} \cdot \left(\frac{6}{13} - \frac{6}{15} \right) = 0.053$$

$$WRSens(R_1) = \frac{9}{15} \cdot \left(\frac{4}{9} - \frac{5}{15} \right) = 0.067$$

$$WRSens(R_2) = \frac{6}{15} \cdot \left(\frac{3}{6} - \frac{3}{15} \right) = 0.120$$

$$WRSens(R_3) = \frac{9}{15} \cdot \left(\frac{2}{9} - \frac{2}{15} \right) = 0.053$$

$$WRSpec(R_1) = \frac{6}{15} \cdot \left(\frac{5}{6} - \frac{10}{15} \right) = 0.067$$

$$WRSpec(R_2) = \frac{9}{15} \cdot \left(\frac{9}{9} - \frac{12}{15} \right) = 0.120$$

$$WRSpec(R_3) = \frac{6}{15} \cdot \left(\frac{6}{6} - \frac{13}{15} \right) = 0.053$$

Percebe-se que estas medidas são as iguais para cada uma das três regras, a partir do que, é possível estabelecer uma relação entre as quatro medidas padrão de avaliação de regras supervisionadas, a partir do relacionamento destas com as suas variantes com peso [GOM02].

Teorema 4.4.2: *WRAcc(R) = WRSens(R) = WRSpec(R) = WRNegRel(R)*

prova.

$$\begin{aligned} WRAcc(B\rightarrow H) &= P(B)(P(H|B)-P(H)) \\ &= P(HB)-P(H)P(B) \\ &= P(H)(P(B|H)-P(B)) \\ &= WRSens(B\rightarrow H). \end{aligned}$$

$$\begin{aligned} WRAcc(B\rightarrow H) &= P(B)(P(H|B)-P(H)) \\ &= P(HB)-P(H)P(B) \\ &= (1-P(\bar{H}B)-P(H\,\bar{B})-P(\overline{HB}))-(1-P(\bar{B}))(1-P(\bar{H})) \\ &= (1-P(\bar{H})-P(\bar{B})+P(\overline{HB}))-(1-P(\bar{H})-P(\bar{B})+P(\bar{H})P(\bar{B})) \\ &= P(\overline{HB})-P(\bar{H})P(\bar{B}) \\ &= P(\bar{H})(P(\bar{B}|\bar{H})-P(\bar{B})) \\ &= WRSpec(B\rightarrow H) \end{aligned}$$

$$\begin{aligned} WRAcc(B\rightarrow H) &= WRSpec(B\rightarrow H) \\ &= \mathbf{WRSpec}(B\rightarrow H). \\ &= P(\bar{B})(P(\bar{H}|\bar{B})-P(\bar{H})) \\ &= WRnEGrEL(B\rightarrow H) \end{aligned}$$

Para melhor entendimento, a Tabela 4.6 mostra de forma condensada os valores das medidas relativas de avaliação de regras, com e sem peso, obtidas para cada uma das três regras empregadas, como exemplo de ilustração.

Tabela 4.6: Valores das Medidas Relativas de Avaliação das Regras

Medidas	***Regras***		
	R1	***R2***	***R3***
RAcc ($B \rightarrow H$)	0.200	0.600	0.400
RNegRel($B \rightarrow H$)	0.100	0.150	0.061
RSens($B \rightarrow H$)	0.111	0.300	0.089
RSpec($B \rightarrow H$)	0.167	0.200	0.133
WRAcc($B \rightarrow H$)*	0.833	1.000	1.000

*Medidas Relativas com Peso *WRAcc*($B \rightarrow H$) = *WRSens*($B \rightarrow H$) = *WRSpec*($B \rightarrow H$) = *WRNegRel*($B \rightarrow H$)

4.8. Considerações Finais do Capítulo

O objetivo das medidas de avaliação de regras é prover ao usuário uma forma de focalizar sua atenção sob aquelas regras que melhor se destacam de forma sustentada pelos dados, isto é, concentra a atenção sob as regras de maior qualidade. Com isto, estas medidas são apresentadas no próximo Capítulo como parte do tema deste trabalho.

Capítulo V

Metodologia

Neste capítulo é apresentada a metodologia empregada neste trabalho, bem como a utilização dos algoritmos propostos, a análise do desempenho que cada algoritmo obteve e a aplicação das medidas de avaliação de regras nas principais regras obtidas. Por fim a interpretação dos resultados obtidos.

5.1. Introdução da Metodologia Empregada

O presente trabalho trata da comparação de dois algoritmos de extração de regras de Associação, o Algoritmo *ZigZag* e o Algoritmo *APriori.* O que justificou a utilização dos algoritmos, foi a característica diferenciada do algoritmo *ZigZag*, o qual possui a viabilidade de poder ser utilizada em bases de dados de tamanho elevado para a extração de regras de associação de forma Incremental, que é o mesmo propósito do algoritmo *APriori*, porém de forma normal, permitindo uma análise de teor comparativo entre os resultados obtidos. Diante disto, o objetivo do trabalho é, além de analisar o desempenho dos algoritmos, também analisar a qualidade das regras obtidas.

A metodologia empregada neste trabalho, está dividida em 3 fases composta por 6 estágios distintos. A primeira fase é a Fase da Descoberta do Conhecimento, onde são empregadas as etapas do *KDD*, composta por três estágios distintos: O primeiro versa sobre a definição dos algoritmos empregados e seus valores de *suporte* e *confiança* para a extração das regras. Este estágio é essencial para a definição da viabilidade do trabalho proposto. No segundo estágio é realizada a definição das bases de dados empregadas afim de comparação dos resultados. Neste estágio, é definida a quantidade de bases de dados disponível a ser utilizada pelos algoritmos, a dimensão das bases de dados, o período de dados contido em

cada base, bem como suas características para a extração de regras. No terceiro estágio, é efetuada a aplicação dos algoritmos. Nele, as bases de dados devem ser adaptadas para que ambos os algoritmos possam interpretar os dados, para a extração das regras de associação. Finaliza-se assim, a primeira fase.

A segunda fase é a Fase Estatística. Nela são aplicadas diversas equações matemáticas, objetivando realizar um comparativo entre as regras obtidas dos algoritmos. Esta fase é composta por dois estágios: No primeiro estágio é efetuada a avaliação do desempenho obtida de cada algoritmo em cada regra extraída, bem como os artifícios utilizados para efetuar a comparação entre as regras. Neste estágio, são demonstrados os gráficos de desempenho do tempo e da quantidade de regras obtidas para cada valor de suporte e confiança obtido de cada algoritmo, bem como um comparativo da posição das regras obtidas entre os algoritmos de acordo com a sua ordem de classificação. O segundo estágio versa sobre a aplicação das medidas de avaliação de regras de associação. O objetivo é avaliar a qualidade das regras extraídas utilizando diversas medidas de avaliação que são referência nas bibliografias de avaliação de regras de associação, finalizando assim esta fase. Por fim, a terceira e última fase, que é a Fase de Conclusão, composta pela etapa de Interpretação dos Resultados, que tem o objetivo de efetuar a interpretação dos resultados obtidos, demonstrando a viabilidade e aplicabilidade dos algoritmos propostos. Com o objetivo de melhor apresentar as definições acima, a Figura 5.1 demonstra as três fases e suas etapas aplicadas neste trabalho.

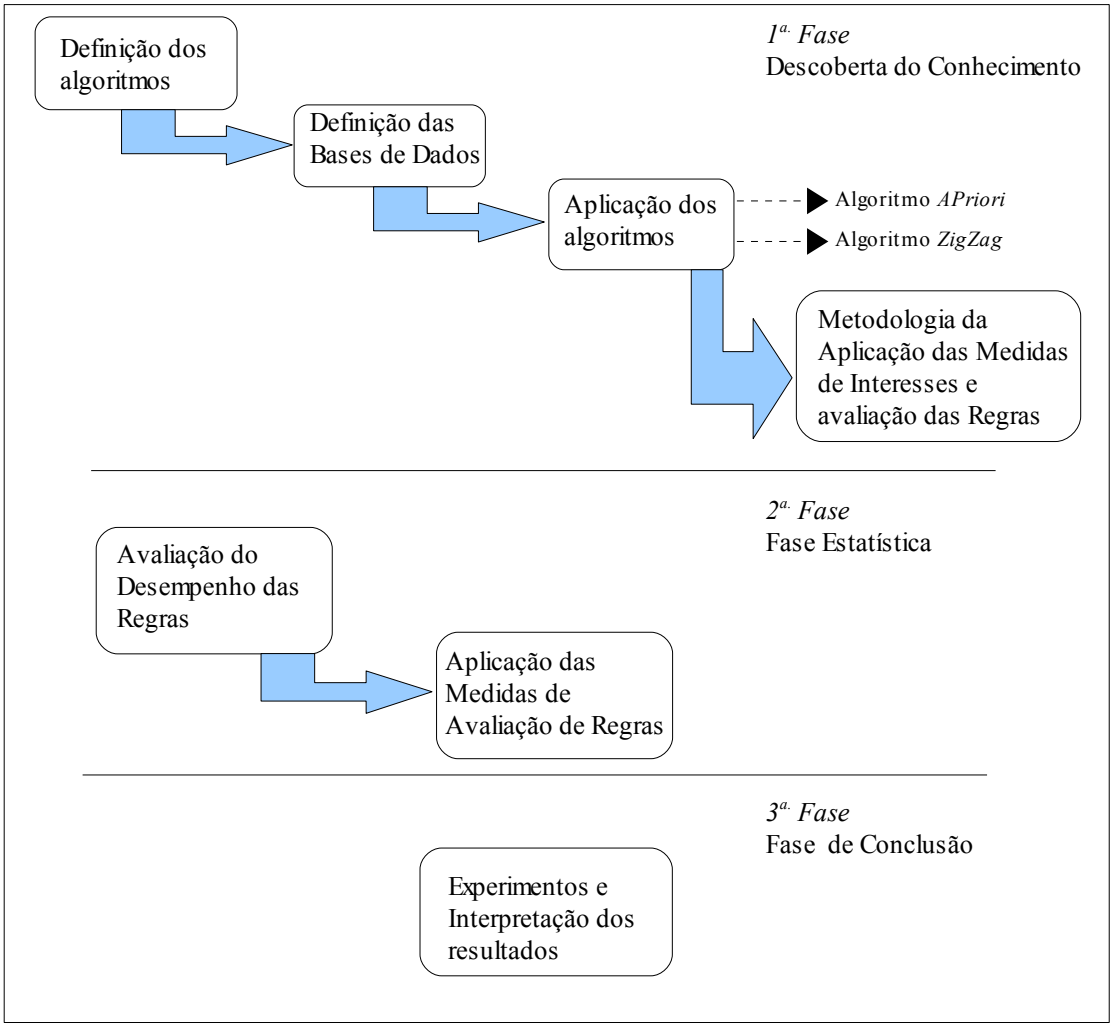


Figura 5.1: Fases Aplicadas à Comparação das Regras de Associação Extraídas pelos Algoritmos *APriori* e *ZigZag*.

5.2. Definição dos Algoritmos

Uma das mais complexas fases deste trabalho, é a definição dos algoritmos. Nela todo o processo pode ser comprometido, tendo em vista a característica, evolução, compatibilidade, tempo, aplicabilidade etc. Neste trabalho, optou-se por utilizar os algoritmos *APriori* e *ZigZag*, para melhor representar duas formas distintas de extração de regras de associação. O algoritmo *APriori* faz a extração de regras de associação de forma Normal e o algoritmo *ZigZag* de forma Incremental. O algoritmo *APriori* aqui utilizado foi uma implementação do Professor Bart Goethals[7] (junho de 2006), baseada na implementação de Agraval et al. (1996), e está disponível na internet, no endereço http://www.adrem.ua.ac.be/~goethals/software/.

[7]Bart Goethals – Professor in the departement of Mathematics and Computer Science at the University of Antwerp. – Middelheimlaan 1 B-2020 Antwerpen, Belgium.

Esta implementação foi uma das mais eficientes aplicada às bases de dados utilizadas neste trabalho, tendo em vista que dispõe do recurso de ser aplicada em bases de dados, tanto literais como numeral, base esta como pré-requisito para a compatibilidade no uso do algoritmo *ZigZag,* facilitando assim, a aplicação de ambos os algoritmos ao mesmo tipo de bases de dados. Uma outra particularidade deste algoritmo, foi a facilidade como ele se comporta mediante a uma base de dados relativamente densa.

Esta implementação do *APriori*, está dividida em duas partes distintas, a primeira, tem o objetivo de buscar os conjuntos de itens freqüentes na base de dados, e a segunda, utiliza o *conjunto de itens* freqüentes encontrados para a geração das regras.

Por motivo de não ser encontrada documentação da utilização deste algoritmo, é apresentado a sintaxe da utilização do programa neste trabalho, o qual foi empregada em um sistema operacional Linux conforme descrito na Seção 5.3, através de linha de comando, com o seguinte formato:

1. ./APriori <base de dados> <tipo de dados> <Valor de Suporte Mínimo> [Saída do arquivo de Conjuntos] - (para encontrar os *conjuntos de itens freqüentes*), e
2. ./rules < Arquivo de Conjuntos> <Valor de Confiança Mínima> [Saída dos Arquivos de Regras] – (para encontrar as regras de associação).

O outro algoritmo empregado neste trabalho é uma implementação baseada na idéia de Cheung et. al., (1996), disponibilizado por *e-mail* através do Sr. Adriano Alonso Veloso[8]. Este algoritmo tem como idéia principal, manter a máxima freqüência dos *conjuntos de itens*. Conhecido como um limiar positivo, para construir de forma incremental uma grande freqüência de *conjunto de itens*. Ele possui três etapas principais:

1. Gravar as novas transações e descartar as transações obsoletas;
2. Atualizar o *conjunto de itens* máximos freqüentes na atualização da base de dados;
3. A atualização do *suporte* para uma freqüência de *conjunto de itens* na atualização da base de dados.

[8]Adriano Alonso Veloso – Professor Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação. Belo Horizonte, Brasil.

De forma análoga à utilização do algoritmo *APriori*, não foi possível encontrar documentação sobre a utilização do algoritmo *ZigZag*. Dessa forma, é apresentada a sintaxe de utilização do programa para a utilização do algoritmo, foi empregada em um sistema operacional Linux, através de linha de comando, com o seguinte formato:

./zigzag -i <Base de Dados> -o <Saída do arquivo> -p <Saída do arquivo p/ futuras execuções> -s <valor de suporte mínimo> -c <valor de confiança mínima> -m <Número de transações>

A idéia inicial da utilização destes algoritmos, é utilizar implementações já consagradas na extração das regras, sem sofrer alterações do formato original de seus desenvolvedores. Fazendo com que os dados coletados para a comparação principalmente no sentido de desempenho, tenha uma maior credibilidade dos seus resultados.

5.3. Definição das Bases de Dados

Esta etapa consiste em preparar o ambiente para a execução dos algoritmos de regras de associação. Para tal, é necessário a preparação dos dados de forma a seguir as configurações básicas de cada algoritmo empregado neste trabalho. Levando em consideração que os dados foram extraídos de uma base real, estes dados sofreram uma alteração no sentido de preservar as informações reais da empresa.

Os dados constantes nesta base foram extraídos de um *Data Warehouse*[9] de forma numérica, definidos mediante a condição de que deveria conter duas bases de dados de tamanho e períodos distintos. Então, utilizou-se dados históricos do ano de 2005, formando duas bases com dois períodos: a primeira base contém dados pertinentes a seis (6) meses de informação (Janeiro à Junho), contendo 482.286 registros. A segunda base contém dados pertinentes aos 12 meses do ano de 2005 (Janeiro à Dezembro), contendo um total de 5.075.715 registros. Esta quantidade de registros é quantidade final, tendo em vista que ambas as bases passaram pelo processo de *KDD*, nas fases de consolidação, seleção, pré-processamento e transformação dos dados, ficando com 12 características demonstradas na Tabela 5.1 para a extração das regras. A diferença da quantidade de registros efetuados em um período, se deu por conta de um processo de transição na empresa, em que no primeiro

[9]*Data Warehouse* – (armazém de dados) é uma coleção de dados, organizados por assunto, integrados, não-voláteis, históricos, cujo propósito é fornecer suporte à tomada de decisão nas organizações.

período (janeiro a junho/2005), os processos eram manuais. A partir de julho, foi automatizado todo o mecanismo de controle destes processos, fazendo com que o número de registro fosse atualizado *on-line,* conforme mostra a Tabela 5.3 contendo as bases que foram definidas e homologadas para utilização dos algoritmos.

Tabela 5.1: Tabela das Características das Cases de Dados

Nome	Descrição
Tipo de pessoa	Física ou Jurídica;
Situação	Ativo, Desligado ou Inativo;
Código da Especialidade	Código da especialidade atendido na guia;
Código Prestador[10] Solicitante	Código do Prestador que solicitou o exame;
Código da especialidade do Prestador Executante	Código da especialidade do prestador que executou o exame;
Situação do Prestador executante	Situação do Prestador executante: Ativo, Inativo ou Desligado;
Código do Serviço	Código do serviço executado;
Código do Usuário	Código do usuário que utilizou os serviços;
Idade	Idade do Usuário;
Estado Civil	Estado civil do usuário: solteiro, casado, desquitado, divorciado, viúvo e outros.
Sexo	Sexo do usuário: masculino ou feminino;
Código do Plano	Código do Plano do usuário.

O campo pertinente à Idade, colocada nesta base de dados, seguiu as recomendações da ANS[11], que determina uma tabela de faixa etária apresentada na Tabela 5.2. e codificada para melhor representar na base de dados. Esta faixa de idade vem de encontro a uma das etapas do *KDD*, a Transformação, que tem a função de reduzir o número de amostras, atributos, intervalos de atributos e normalizá-los[12].

[10]*Prestador* – Nome usualmente utilizado para empresas que prestam serviços especializados, tal como laboratório de análises clínicas.
[11]*ANS* – Agência Nacional de Saúde Suplementar.
[12]*Normalizar* – tornar(-se) normal, regularizar(-se). Este termo aqui empregado, tem o sentido de ajuste dos dados de acordo com a necessidade do algoritmo proposto.

Tabela 5.2: Tabela de Faixa Etária Definida pela ANS

Faixa Etária	Código
00-18 anos	111
19-23 anos	222
24-28 anos	333
29-33 anos	444
34-38 anos	555
39-43 anos	666
44-48 anos	777
49-53 anos	888
54-58 anos	999
59 ou mais	1010

Tabela 5.3: Tabela de Base de Dados

Nome da Base	Quantidade de Registros	Tamanho da Base (*Mb*)	Período
BDDTS_janjun2005.Apr	482.286	37Mb	Janeiro à Junho/2005
BDDTS_jandez2005.Apr	5.075.715	395Mb	Janeiro à Dezembro/2005

5.4. Aplicação dos Algoritmos

Nesta fase foram aplicados os algoritmos mediante a definição dos valores de *suporte* e *confiança,* os quais são 10%, 30%, 45% e 60%. Estes valores foram empregados de forma alternada entre si, para cada arquivo de regras geradas e algoritmo empregado. Esta sistemática permite gerar um total de 16 experimentos conforme apresentado na Tabela 5.4 para cada algoritmo e apresentado na Figura 5.2, totalizando em 32 experimentos aplicados nas duas bases para cada algoritmo empregado.

Tabela 5.4: Tabela dos Valores para *Suporte* e *Confiança*

Experimentos	Nível de *Suporte*	Nível de *Confiança*
1	10	10
2	10	30
3	10	45
4	10	60
5	30	10
6	30	30
7	30	45
8	30	60
9	45	10
10	45	30
11	45	45
12	45	60
13	60	10
14	60	30
14	60	45
15	60	45
16	60	60

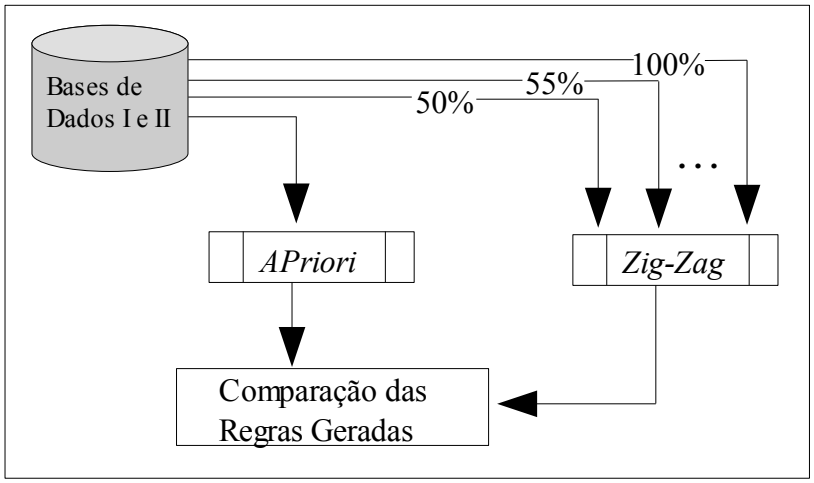


Figura 5.2: Sistemática da Comparação das Regras Geradas

Ambos os algoritmos foram executados na seguinte plataforma: Um servidor AMD *Opteron*™ 248, bi-processado 2GHz com 4Gb RAM, Sistema Operacional Linux – Ubuntu

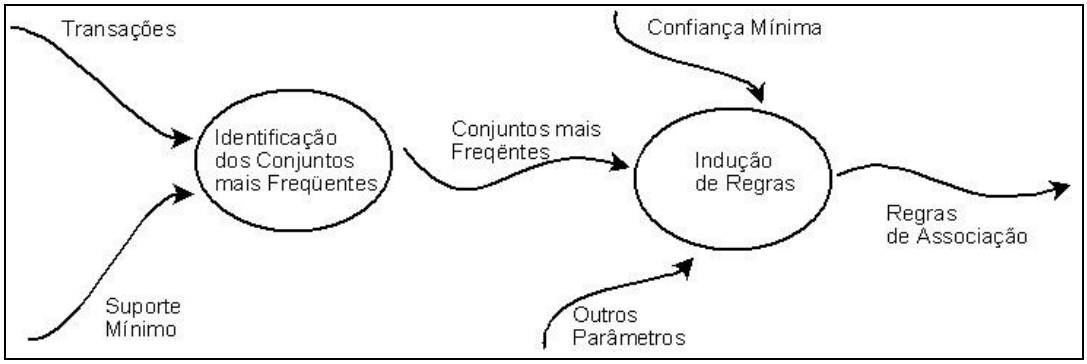
6.10 – 64bits, Kernel – linux Hercules 2.6.17-11-generic#2SMP, UTC 2007 x86_64 GNU/Linux e banco de dados MySQL 5.0.

5.4.1 Aplicação do Algoritmo *APriori*

O algoritmo *APriori* utilizado neste trabalho, é composto de dois programas. O primeiro programa denominado *APriori*, tem a função de gerar os conjuntos de itens e o segundo programa denominado *rules*, utiliza o conjunto gerado pelo programa anterior para extrair a combinação de regras. Assim, pode-se afirmar, conforme apresentado na Figura 5.3, a seguinte definição do algoritmo empregado que possui dois módulos principais:

I. Identificação dos conjuntos mais freqüentes;
II. Indução de regras de associação.

Figura 5.3: Estrutura do Programa *APriori*[SOU00]



Mediante estas definições, o algoritmo *APriori* foi empregado através da criação de quatro *scripts*[13], dois para a base de dados I e dois para a base de dados II, conforme descritos e apresentados abaixo:

- *gerar_APriori_Con* – Utilizado na primeira Base de dados conforme Tabela 5.3 para a identificação dos Conjuntos mais Freqüentes;
- *gerar_APriori_Con2* – Utilizado na segunda Base de dados conforme Tabela 5.3 para a identificação dos Conjuntos mais Freqüentes;

[13] *Scripts ou Shell Scripts*: Arquivos contendo instruções estruturadas para execução de processos pré-definidos. Muito utilizados em sistemas Operacionais Unix e Linux.

Cada *script* contém instruções para a geração dos conjuntos mais freqüentes, utilizando o programa *APriori* com os seguintes parâmetros:

APriori [base de dados] [tipo de dados] [suporte mínimo] [saída do arquivo]

Base de dados: As bases de dados utilizadas foram as descritas na Tabela 5.3 em formato numérico, atendendo os requisitos mínimo de ambos os algoritmos. Em especial as especificações do algoritmo *ZigZag* que não recomenda utilização de dados em formato literal ou binário.

Tipo de Dados: Este parâmetro é utilizado para definição do tipo de dados, como descrito acima, o formato utilizado foi o de número 3.

- tipo de dados: 1 – arquivo em formato binário;
- tipo de dados: 2 – arquivo em formato *ascii*[14];
- tipo de dados: 3 – qualquer formato de arquivo; (utilizado neste trabalho)
- tipo de dados: 4 – arquivos em formato *ascii* ou binário somente;

Suporte mínimo: Este valor foi empregado conforme descrito na seção 5.3, isto é, 10, 30, 45 e 60. Sempre respeitando a quantidade conforme a Tabela 5.4 da mesma seção.

Observação: O valor de suporte empregado neste algoritmo deverá ser obrigatoriamente o número de registros (representado na equação 5.1), por exemplo: se a base tem 200 registros e deseja-se um suporte de 30%, então o valor a ser passado como entrada é o valor igual a 60.

$$|a| = max\{a, -a\} \tag{5.1}$$

Saída do arquivo: Esta opção é a saída do arquivo padrão para que o próximo programa possa utilizar.

De forma análoga, foram empregados os dados no programa *rules* afim de extrair as regras de associação de acordo com os seguintes *scripts*:

- *gerar_APriori_Sup* – Utilizado na primeira Base de dados conforme Tabela 5.3 e utilizando os arquivos gerados conforme a Tabela 5.5, para a extração das regras de associação;

[14] *ASCII* (*American Standard Code for Information Interchange*). Código numérico usado para representar caracteres em arquivos texto em computadores e dispositivos de armazenamento eletrônico de dados.

- *gerar_APriori_Sup2* – Utilizado na segunda Base de dados conforme Tabela 5.3 e utilizando os arquivos gerados conforme a Tabela 5.5, para a extração das regras de associação;

Cada *script* contém instruções para a extração das regras, utilizando o programa *rules* com os seguintes parâmetros:

rules [conjunto de arquivos] [confiança mínima] [saída do arquivo]

Conjunto de arquivos: São os arquivos gerados conforme descrito na pelos programas *APriori* e *ZigZag*;

Confiança mínima: Este valor foi empregado conforme descrito na seção 5.3, isto é, 10%, 30%, 45% e 60%. Sempre respeitando a quantidade conforme a Tabela 5.4 da mesma seção.
Saída do arquivo: Esta opção é a saída do arquivo padrão das regras extraídas, apresentada na Tabela 5.5.

Tabela 5.5: Tabela dos Arquivos de Regras Extraídas do Algoritmo *APriori*

Experimentos	Nome do Arquivo de regras Base de Dados I	Nome do Arquivo de regras Base de Dados II
1	Regras_Apr1_s10c10.txt	Regras_Apr2_s10c10.txt
2	Regras_Apr1_s10c30.txt	Regras_Apr2_s10c30.txt
3	Regras_Apr1_s10c45.txt	Regras_Apr2_s10c45.txt
4	Regras_Apr1_s10c60.txt	Regras_Apr2_s10c60.txt
5	Regras_Apr1_s30c10.txt	Regras_Apr2_s30c10.txt
6	Regras_Apr1_s30c30.txt	Regras_Apr2_s30c30.txt
7	Regras_Apr1_s30c45.txt	Regras_Apr2_s30c45.txt
8	Regras_Apr1_s30c60.txt	Regras_Apr2_s30c60.txt
9	Regras_Apr1_s45c10.txt	Regras_Apr2_s45c10.txt
10	Regras_Apr1_s45c30.txt	Regras_Apr2_s45c30.txt
11	Regras_Apr1_s45c45.txt	Regras_Apr2_s45c45.txt
12	Regras_Apr1_s45c60.txt	Regras_Apr2_s45c60.txt
13	Regras_Apr1_s60c10.txt	Regras_Apr2_s60c10.txt
14	Regras_Apr1_s60c30.txt	Regras_Apr2_s60c30.txt
15	Regras_Apr1_s60c45.txt	Regras_Apr2_s60c45.txt
16	Regras_Apr1_s60c60.txt	Regras_Apr2_s60c60.txt

5.4.2 Aplicação do Algoritmo *ZigZag*

O algoritmo *ZigZag* utilizado neste trabalho, foi o algoritmo conforme descrito na seção 5.2 e empregado na plataforma Linux em linha de comando, contendo os seguintes parâmetros:

./*zigzag* -i <Base de Dados> -o <Saída do arquivo> -p <Saída do arquivo p/ futuras execuções> -s <valor de suporte mínimo> -c <valor de confiança mínima> -m <Número de transações>

-i <Base de Dados>: Esta é a opção que utiliza a base de dados que será aplicada para a extração das regras de acordo com a Tabela 5.3 desta seção;

-o <Saída do arquivo>: Esta opção e a saída do arquivo de Regras de Associação que foram encontradas pelo algoritmo, apresentada na Tabela 5.7 abaixo;
-p <Saída do arquivo p/ futuras execuções>: Esta opção é utilizada para que o algoritmo grave um arquivo para futuras execuções;

-s <valor de suporte mínimo>: É o valor de *suporte* mínimo empregado na extração de regras, de acordo com a Tabela 5.4 desta seção;

-c <valor de confiança mínima>: É o valor de *confiança* mínimo empregado na extração de regras, de acordo com a Tabela 5.4 desta seção;

-m <Número de transações>: É o número de transações "incremental" que deverá ser executado, como demonstra a Figura 5.4 da sistemática de incremento do algoritmo *ZigZag*.

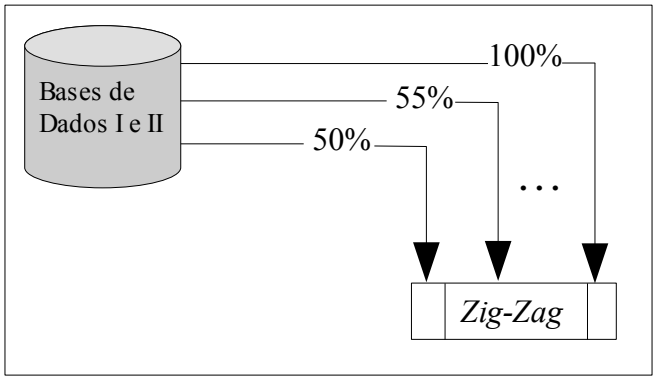


Figura 5.4: Sistemática da Aplicação Incremental do Algoritmo *ZigZag*.

Como a utilização do algoritmo *ZigZag* é de forma incremental como apresentado na Figura 5.4 acima, primeiramente convencionou-se que deve-se efetuar uma divisão do tamanho total da base em 2 (duas) partes distintas, o qual a primeira representa 50% do valor total, a outra parte do total da base, foi fracionada em 10 parte iguais representando cada fração 5% do total base de dados, como demonstrada da na equação 5.1.

$$\frac{todb}{2} = a \qquad todb_k = \frac{a}{10}; k = 1, 2, \ldots, 10 \tag{5.1}$$

$$I_k = k - \acute{e}sima\ fração\ de\ a \quad [15]$$

Descrição da equação:

todb = Total de registros da base de dados;

a = 50% de *todb*;

I_k = fração de incremento.

Com isto, os *scripts* tiveram que conter dados para cada base conforme abaixo:

$a = \frac{482.286}{2} = 241.142$ Primeira carga de dados para Incremento, utilizado na base de dados I;

$todb_k = \frac{241.142}{10} = 24.114$ Valor para cada fração de I_k utilizado como Incremento até o valor total da base de dados, totalizando 10 iterações.

[15] A representação da fração I_k é necessário devido a cada iteração obter um resultado de incremento.

Tabela 5.6: Tabela dos Arquivos de Regras Extraídas do Algoritmo *ZigZag*

Experimentos	Nome do Arquivo de regras Base de Dados I	Nome do Arquivo de regras Base de Dados II
1	Regras_ZIG1_s10c10.txt	Regras_ZIG2_s10c10.txt
2	Regras_ZIG1_s10c30.txt	Regras_ZIG2_s10c30.txt
3	Regras_ZIG1_s10c45.txt	Regras_ZIG2_s10c45.txt
4	Regras_ZIG1_s10c60.txt	Regras_ZIG2_s10c60.txt
5	Regras_ZIG1_s30c10.txt	Regras_ZIG2_s30c10.txt
6	Regras_ZIG1_s30c30.txt	Regras_ZIG2_s30c30.txt
7	Regras_ZIG1_s30c45.txt	Regras_ZIG2_s30c45.txt
8	Regras_ZIG1_s30c60.txt	Regras_ZIG2_s30c60.txt
9	Regras_ZIG1_s45c10.txt	Regras_ZIG2_s45c10.txt
10	Regras_ZIG1_s45c30.txt	Regras_ZIG2_s45c30.txt
11	Regras_ZIG1_s45c45.txt	Regras_ZIG2_s45c45.txt
12	Regras_ZIG1_s45c60.txt	Regras_ZIG2_s45c60.txt
13	Regras_ZIG1_s60c10.txt	Regras_ZIG2_s60c10.txt
14	Regras_ZIG1_s60c30.txt	Regras_ZIG2_s60c30.txt
15	Regras_ZIG1_s60c45.txt	Regras_ZIG2_s60c45.txt
16	Regras_ZIG1_s60c60.txt	Regras_ZIG2_s60c60.txt

5.5. Metodologia de Avaliação e Desempenho das Regras

A metodologia empregada nesta seção divide-se em duas etapas distintas. A primeira, destina-se a quantificar as regras extraídas e medir o tempo de processamento para cada regra aplicada em cada base de dados, cada valor de *suporte* e *confiança* e algoritmo empregado.

Com isto, é possível ter uma visão clara do desempenho de cada algoritmo, não no sentido de apresentar a qualidade, mas sim a quantidade e o tempo despendido que cada algoritmo obteve. Visando uma melhor explanação desta etapa, a Figura 5.5 apresenta toda a sistemática aplicada às regras neste trabalho, o qual se subdivide em duas fases, *a* para algoritmo *APriori* e *b* para algoritmo *ZigZag*.

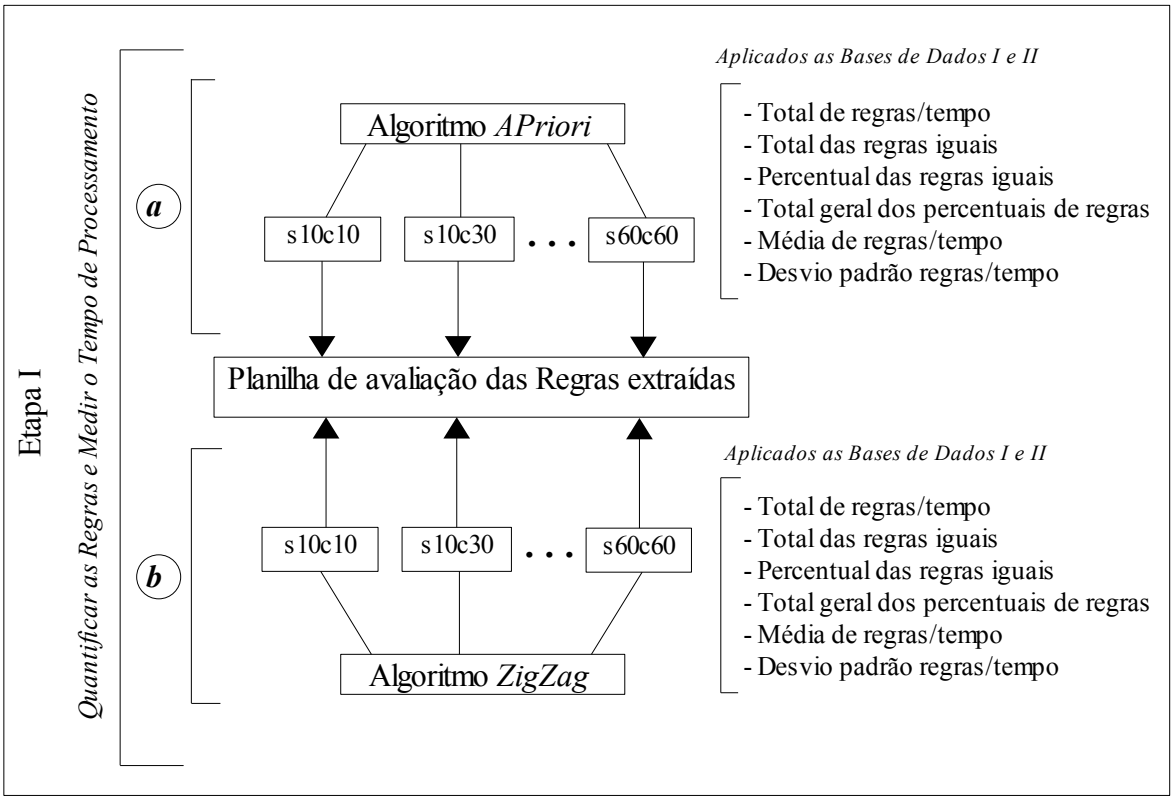


Figura 5.5: Fluxograma para Desenvolver uma Planilha dos Dados Pertinentes a Etapa I

A segunda etapa, destina-se a efetuar uma aplicação das medidas de avaliação das regras, através da extração de informações das regras na *tabela de contingência* demonstrada no Capítulo 4, Seção 4.1.2, Tabela 4.1 deste trabalho.

Inicialmente foi desenvolvido um programa com o objetivo de construir a *Tabela de Contingência* das regras extraídas dos algoritmos propostos, apresentado na Figura 5.6 através de um fluxograma da tabela de contingência. Este fluxograma, apresenta como foi possível construir a tabela de contingência, peça fundamental para a aplicação da medidas de interesse.

A tabela extraída foi exportada para uma planilha para possibilitar a aplicação das 21 medidas aplicadas para avaliação das regras (conforme descritas no Capítulo 4, Definições 4.4.1 à 4.4.17). Esta sistemática está melhor representada na Figura 5.7, como sendo a Etapa II desta seção.

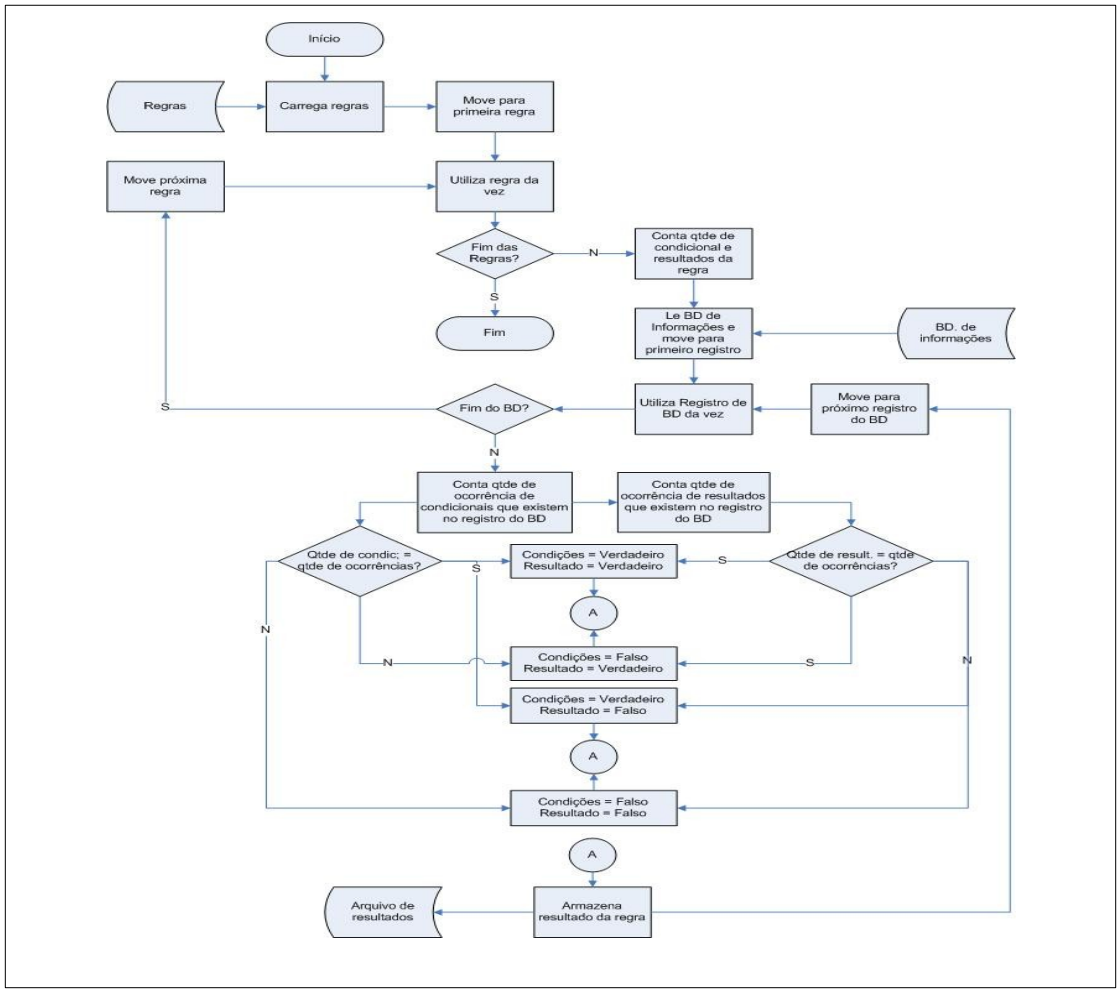


Figura 5.6: Fluxograma da Tabela de Contingência

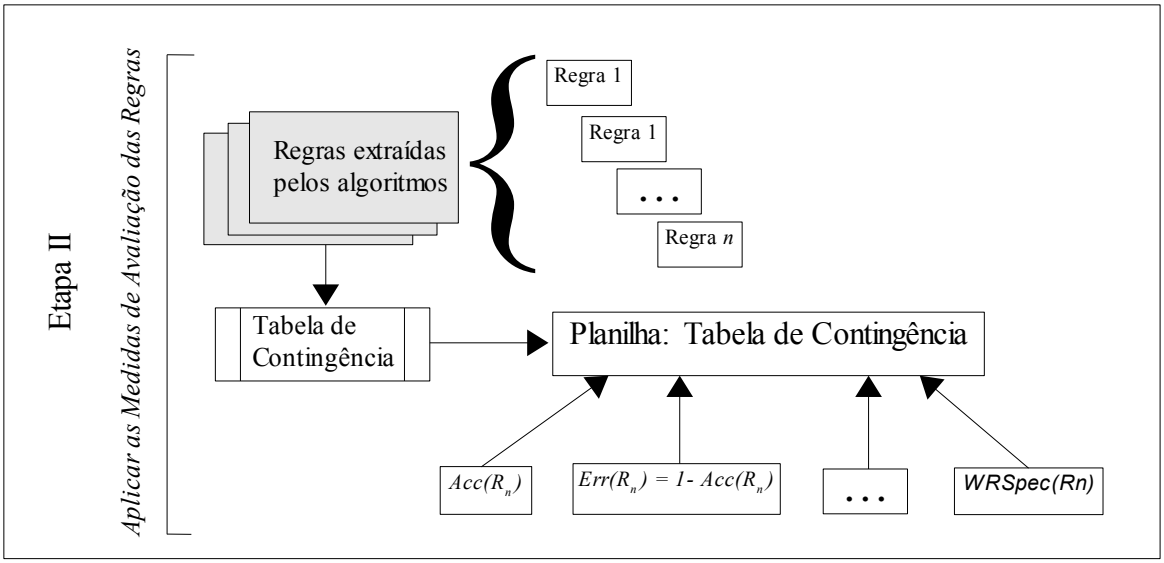


Figura 5.7: Figura da Etapa de Avaliação das Regras

Através destas etapas serão empregadas as seguintes medidas de avaliação, as quais estão dividas em três tipos:

- *Medidas Genéricas:*
  - *acc*(R): Precisão da regra
  - *err*(R): Erro da Regra
  - *negrel*(R): Confiança negativa da regra
  - *sens*(R): Sensitividade da regra
  - *spec*(R): Especificidade da regra
  - *cov*(R): Cobertura da regra
  - *sup*(R): Suporte da regra
  - *nov*(R): Novidade da regra
  - *sat*(R): Satisfação da regra

- *Medidas Relativas:*
  - *racc*(R): Precisão relativa com peso da regra
  - *rnegrel*(R): Confiança negativa relativa com peso da regra
  - *rsens*(R): Sensitividade relativa com peso da regra
  - *rspec*(R): Especialidade relativa com peso da regra

- *Medidas Relativas com peso:*
  - *wracc*(R): Precisão relativa com peso da regra
  - *wrnegrel*(R): Confiança negativa relativa com peso da regra
  - *wrsens*(R): Sensitividade relativa com peso da regra
  - *wrspec*(R): Especialidade relativa com peso da regra

5.6. Considerações Finais do Capítulo

Neste capítulo foi descrita a metodologia empregada neste trabalho, de forma estruturada. O próximo capítulo, versa sobre a aplicação da metodologia proposta afim de validar o comparativo dos algoritmos propostos neste trabalho.

Capítulo VI

Experimentos e Avaliação dos Resultados

Neste capítulo são apresentados os experimentos e a avaliação dos resultados obtidos, de acordo com a metodologia proposta no capítulo anterior. Apresentando de forma investigativa os resultados pertinentes ao desempenho e a qualidade que cada algoritmo (*APriori* e *ZigZag*) obteve, viabilizando com isto, a comparação dos resultados de forma a apontar qual a melhor indicação para o uso de cada algoritmo para o cenário proposto.

6.1. Introdução

As regras geradas a partir do Algoritmo *APriori* são regras tomadas como referência para as regras geradas pelo algoritmo *ZigZag*. Dessa forma, serão equacionadas estatisticamente todas as regras geradas por ambos os algoritmos para comprovar a sua qualidade. Uma das formas de mensurar as regras extraídas é utilizando as medidas de avaliação das regras de forma qualitativa para cada regra gerada pelos algoritmos (APriori e ZigZag). Os valores obtidos são calculados a partir da soma das informações referentes aos exemplos extraídos com valores conhecidos e desconhecidos de cada regra.

6.2. Bases de Dados

A realização dos testes dependeu de duas bases de dados de forma semelhantes, porém com um tamanho distinto. A primeira base contém informações de uma empresa de plano de saúde, pertinentes a um período de 6 meses (janeiro a junho) de 2005, totalizando 482.286 registros. A segunda base contém as mesmas informações da primeira, porém, com um tamanho de 5.075.715 registros, pertinentes a 12 meses (janeiro a dezembro) de 2005, como apresenta a Tabela 5.3 no Capítulo anterior.

A criação dessas bases de dados foi possível através de um *Datawarehouse* tendo em vista que se trata de uma base de dados real, que dispõe de informações históricas de um determinado ano, cujas as características extraídas foram definidas de acordo com a Tabela 5.1 do Capítulo anterior.

6.3. Aplicação do Algoritmo *APriori*

Através das bases de dados definidas conforme descritas na seção 6.1, iniciou-se a aplicação dos algoritmos, primeiramente pelo *APriori*. Neste momento procurou-se deixar o algoritmo induzir as regras de associação entre os procedimentos médicos executados por prestadores usados pelos beneficiários do plano de saúde. A única determinação legal que esta base possui, é o atributo da faixa etária conforme descrito no Capítulo 5, Tabela 5.2, demais atributos, foram escolhidos direcionados para a descoberta de associação entre prestadores.

6.3.1 Criação dos *Scripts* Para o Algoritmo *APriori*

Em virtude da quantidade dos experimentos, optou-se por criar *scripts* de execução, assumindo os parâmetros necessários para a geração das regras de acordo com o algoritmo empregado. Assim, cada *script* deve seguir a ordem de valores convencionados para *suporte* e *confiança* dispostos da seguinte forma: 10%, 30%, 45% e 60%. Estes valores alternam-se entre si, gerando um total de 16 regras para cada algoritmo de cada base de dados, totalizando 32 experimentos.

Os *scripts* utilizados para gerar regras através do algoritmo *APriori* estão divididos em duas etapas, as quais são:

Etapa I

- *gerar_APriori_Con* = tem o objetivo de extrair os conjuntos de itens freqüentes, de acordo com o valor de *confiança* aplicado;
- *gerar_ APriori_Con2* = segue a mesma sistemática do *script* anterior, porém direcionado para a segunda base de dados.

Estes *scripts* foram criados para executar o programa *rules*, cujo objetivo é buscar os conjuntos de itens freqüentes na base de dados, gerando assim, para cada valor de *confiança* um arquivo contendo os conjuntos de itens quais serão utilizados pelo programa *APriori*, para gerar as regras de associação.

Os parâmetros utilizados nestes *scripts* são os seguines:

./*rules* resultados/APriori/Cjtos_Apr_s10.txt 10 resultados/APriori/Regras_APR_s10c10.txt >> resultados/APriori/Temp_Apr_Reg_s10c10.txt

Onde:

- *rules*: é o programa que gera o conjunto freqüente de itens;
- resultados/APriori/Cjtos_Apr_s10.txt: é o arquivo destino gerado, como apresenta a Tabela 5.6;
- 10: é o valor de o valor de confiança mínima empregada ao algoritmo.
- >> resultados/APriori/Temp_Apr_Reg_s10c10.txt: é o resultado de saída para arquivo, da evolução do programa, adicionado ao *script* como um arquivo de *log*[16] de controle da aplicação do algoritmo.

Etapa II

- *gerar_APriori_Sup* = Tem o objetivo de gerar as regras de acordo com o valor de *suporte* empregado e os arquivos de conjuntos de itens freqüentes gerados na etapa anterior.
- *gerar_APriori_Sup2* = segue a mesma sistemática do *script* anterior, porém direcionado para a base de dados II.

Estes *scripts* tem a função de passar parâmetros para o programa que gera as regras de associação denominado *APriori*, baseados nos arquivos gerados pelos *scripts* anterior. Os parâmetros utilizados nestes *scripts* são:

./APriori Bases/BDDTS_janjun2005.Apr 3 289371.00 resultados/APriori/Cjtos_Apr_s60.txt > resultados/APriori/TempoCJ_Apr_Sup_60.txt

- *./APriori*: programa executável para gerar as regras;
- Bases/BDDTS_janjun2005.Apr: é o arquivo da base de dados que o *script gerar_APriori_Con* gera contendo o conjunto de itens freqüentes;
- 3: é o valor correspondente ao tipo de dados utilizado para a geração do algoritmo conforme descrito na seção 5.4.1 do capítulo anterior;
- 289371.00: é o valor de *suporte* empregado pelo algoritmo, este valor deve ser em formato de valor absoluto, ou seja, o sistema não entende um valor direto de 10%. Por exemplo: Caso queira executar o programa com um valor de 20% de 60, deve-se

[16] *LOG* – usualmente utilizado em computação, para armazenar os resultados afim de uma futura análise.

passar o valor absoluto de 12 e não 20%. Com isto, este valor representa 60% da base de dados I.

- resultados/APriori/Cjtos_Apr3_s60.txt > resultados/APriori/TempoCJ_Apr3_Sup_60.txt: é o resultado de saída para arquivo, da evolução do programa, adicionado ao *script* como um arquivo de *log* de controle da aplicação do algoritmo.

Então, a linha de comando para a geração das regras é a seguinte:

$./*gerar_APriori_Sup*

$./*gerar_APriori_Sup2*

$./*gerar_APriori_Con*

$./*gerar_APriori_Con2*

6.3.2 Dados de Entrada para o Algoritmo *APriori*

Como o algoritmo *APriori* trabalha somente com dados numéricos, os dados foram codificados de acordo com a Tabela 6.1, dessa forma os dados de entrada foram:

Tabela 6.1: Tabela de Legendas

Código	**Descrição**
210	Serviços Hospitalares
992	Pessoa Jurídica
1919	Situação do usuário
42042	Situação do Solicitante, “Ativo”
77777	Sexo Masculino
88888	Sexo Feminino
9991010	Faixa etária das pessoa que possuem idade acima de 59 anos
190000011	Estado Civil, “Casado”
190000001	Estado Civil, “Solteiro”

Dados de entrada:

992 1919 209 39462 50008 42042 28040783 18024890 9991010 190000090 77777 30000076

992 1919 209 39462 50008 42042 28040333 18024890 9991010 190000090 77777 30000076

.

.

.

991 1919 3 38474 50008 42042 39100022 39015734 9991010 190000011 88888 30000077

Onde cada coluna representa um atributo, como segue:

1. Tipo de pessoa;
2. Situação;
3. Código de Especialidade;
4. Código do Prestador Solicitante;
5. Código de Especialidade do Solicitante;
6. Situação do Solicitante;
7. Código do Serviço;
8. Código do Usuário;
9. Idade;
10. Estado Civil;
11. Sexo;
12. Código do Plano.

6.3.3 Resultados Obtidos do Algoritmo *APriori*

1919 => 210 30000030 (S 48675, C 10.1113%)

1919 => 210 992 30000030 (S 48675, C 10.1113%)

1919 => 42042 77777 500048 (S 48516, C 10.0783%)

1919 42042 => 992 77777 30000025 (S 48911, C 10.2672%)

.

.

.

210 9991010 => 992 (S 92486, C 100%)

Os resultados apresentados acima, referem-se aos valores empregados de *suporte* e *confiança* igual a 10%. Em função da quantidade de arquivos ser grande e principalmente do tamanho dos arquivos gerados, serão adotados para apresentação deste trabalho as seguintes regras:

BD I.: 210 992 1919 190000011 => 9991010	11,60	37,66
BD II: 992 1919 42042 190000011 => 9991010	12,80	34,54
BD I.: 992 1919 190000011 => 77777	19,45	44,34
BD II: 992 1919 190000011 => 77777	14,37	38,39

BD I.: 992 1919 9991010 => 88888 17,04 60,20

BD II: 992 1919 9991010 => 88888 13,07 64,54

6.3.4 Análise dos Resultados Obtidos pelo Algoritmo *APriori*

O algoritmo *APriori* utilizado neste trabalho, efetuou uma única interação para a extração das regras, aplicado a cada valor de suporte e confiança.

Regra 1: 992 1919 42042 190000011 => 9991010 16,99 39,21

Base de Dados I

Tradução da Regra:

- Das pessoas que possuem um plano de saúde tipo "pessoa jurídica", estão ativos no plano, utilizaram de algum recurso médico/hospitalar de um prestador[17] Ativo, estão com o seu plano com o *status* de ativo e são casados. 39,21% possuem uma idade igual ou superior a 59 anos. (*suporte* de 16,99% e *confiança* de 39,21%).
-

Regra 1: 992 1919 42042 190000011 => 9991010 12,81 34,54

Base de Dados II

Tradução da Regra:

- Das pessoas que possuem um plano de saúde tipo "pessoa jurídica", estão ativos no plano, utilizaram de algum recurso médico/hospitalar de um prestador Ativo, estão com o seu plano com o *status* de ativo e são casados. 39,21% possuem uma idade igual ou superior a 59 anos.(*suporte* de 12,81% e *confiança* de 34,54%).

Considerações:

- Esta é uma informação muito importante, pois tende a chamar a atenção das empresas no sentido de um trabalho de medicina preventiva. Pois além de utilizar o plano, esta pessoa necessitou de algum tipo de exame ou serviço especializado, em função da informação de um prestador de serviços médicos estar com o status Ativo para esta regra. Ou seja, a informação de um prestador só aparece, caso seja necessário a sua utilização.

Regra 2: 992 1919 190000011 => 77777 19,45 44,34

Base de Dados I

[17]Prestador: é um termo muito usual nas empresas de plano de saúde, para se referir a prestadores de serviços terceirizados.

Tradução da Regra:

- 44,34% dos usuários de um plano de saúde do sexo Masculino, possuem um plano de saúde do tipo pessoa jurídica, estão ativos neste plano e são casados.(*suporte* de 19,45% e *confiança* de 44,34%).

Regra 2: 992 1919 190000011 => 77777 14,37 38,39

Base de Dados II

Tradução da Regra:

- 38,39% dos usuários de um plano de saúde do sexo Masculino, possuem um plano de saúde do tipo pessoa jurídica, estão ativos no plano e são casados. (*suporte* de 14,37% e *confiança* de 38,39%).

Considerações:

- Esta regra, apresenta uma tendência das pessoas do sexo masculino terem um plano de saúde, principalmente por se tratar de um benefício que as empresas adotam ao adquirir um plano de saúde empresarial para seus funcionários.

Regra 3: 992 1919 9991010 => 88888 17,04 60,20

Base de Dados I

Tradução:

- Das pessoas que possuem um plano de saúde do tipo pessoa jurídica, estão ativos neste plano, tem idade maior ou igual a 59 anos, 60,20% são do sexo Feminino. (*suporte* de 17,04% e *confiança* de 60,20%).

Base de Dados II

Regra 3: 992 1919 9991010 => 88888 13,07 64,54

Tradução:

- Das pessoas que possuem um plano de saúde do tipo pessoa jurídica, estão ativos neste plano, tem idade maior ou igual a 59 anos, 64,54% são do sexo Feminino. (*suporte* de 13,07% e *confiança* de 64,54%).

Considerações:

- Esta regra, apresenta um fator muito importante para a área da saúde, pois apresenta um número significativo de pessoas com mais de 59 anos utilizando um plano de saúde. Isto só é possível dado ao incentivo das empresas em adquirir planos empresariais para seus funcionários. É sabido que as pessoas com mais de

59 anos, possuem um complicador com relação à saúde, pois todos os planos de saúde praticam preços elevados em decorrência da faixa etária. Porém estes preços são fiscalizados e administrados pela ANS (Agência Nacional de Saúde Suplementar).

6.3.5 Considerações Finais para o Algoritmo *APriori*

O estudo da aplicação do algoritmo *APriori* proposto neste trabalho, obteve um desempenho consistente, tendo em vista que apesar dos valores aplicados de *suporte* e *confiança* serem relativamente altos, o programa gerou regras plausíveis sempre tendendo a informações triviais, em decorrência da característica do algoritmo *APriori* se baseia na maior freqüência de ocorrência da associação dos fatos aplicados.

6.4. Apresentação dos resultados do Algoritmo *APriori*

Após extração das regras, foram computados os tempos de processamento e os totais de regras extraídas pelo algoritmo *APriori*, para cada arquivo de conjunto de itens freqüentes, arquivos de regras e base de dados utilizados, apresentados na Tabela 6.2 abaixo. Este dados são importantes em virtude da necessidade de comparação dos resultados dos dados extraídos.

Tabela 6.2: Quantidade de Regras e Tempo de Processamento Gerados pelo Algoritmo *APriori*.

Experimentos	***Suporte* (%)**	***Confiança* (%)**	**Base de Dados I** *Algoritmo APriori*		**Base de Dados II** *Algoritmo APriori*	
			Quantidade de Regras	**Tempo de Processamento** *(minutos)*	**Quantidade de Regras**	**Tempo de Processamento** *(minutos)*
1	10	10	3.748	12,67	2.284	125,90
2	10	30	2.653	12,67	1.529	125,91
3	10	45	1.803	12,66	1.082	125,90
4	10	60	1.205	12,67	807	125,90
5	30	10	406	8,14	174	69,12
6	30	30	406	8,14	174	69,12
7	30	45	315	8,14	144	69,12
8	30	60	224	8,14	107	69,12
9	45	10	98	6,06	46	63,79
10	45	30	98	6,06	46	63,79
11	45	45	98	6,06	46	63,79
12	45	60	74	6,06	39	63,79
13	60	10	50	5,58	22	49,71
14	60	30	50	5,58	22	49,71
15	60	45	50	5,58	22	49,71
16	60	60	50	5,58	22	49,71

A Tabela 6.2 tem o objetivo de apresentar de forma geral, o desempenho, a quantidade de regras extraídas e o tempo de processamento despendido pelo algoritmo *APriori*. Com a disposição dos dados na Tabela 6.2, pode-se observar que quanto maior a base de dados, menor a quantidade de regras extraídas, porém o tempo de processamento é proporcional ao tamanho da base de dados, isto não significa uma norma da relação do tamanho da base de dados com o tempo de processamento das regras. O que representa estes dados são as características empregadas e os valores de *suporte* e *confiança*.

A Figura 6.1 é um demonstrativo da relação entre a quantidade de regras geradas e dos valores de *suporte* e *confiança*, bem como o tempo de processamento empregados no algoritmo *APriori*. É visível a diferença entre as regras geradas com um suporte e confiança igual a 10% dos demais valores empregados. Isto só é possível em função do valor baixo para as regras, pois é sabido que quanto menor o valor de suporte, maior será a quantidade de regras extraída. De forma análoga, a Figura 6.2 é referente ao tempo de processamento, onde

a medida do tempo empregado neste algoritmo, varia em decorrência da capacidade que o servidor possui para processar as regras que são geradas de acordo com os valores de *suporte* e *confiança* e do tamanho das bases de dados adotado.

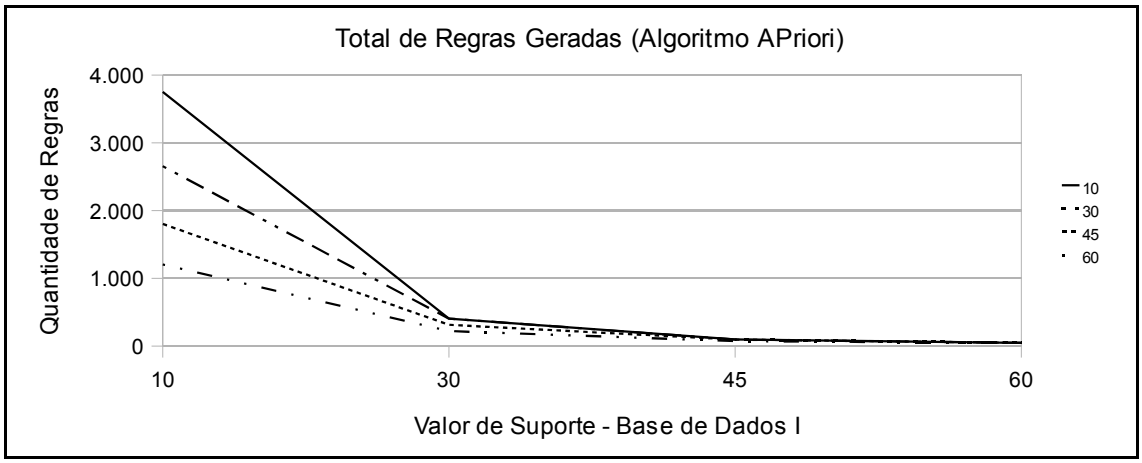


Figura 6.1: Quantidade de Regras Geradas pelo Algoritmo *APriori* (Base de Dados I)

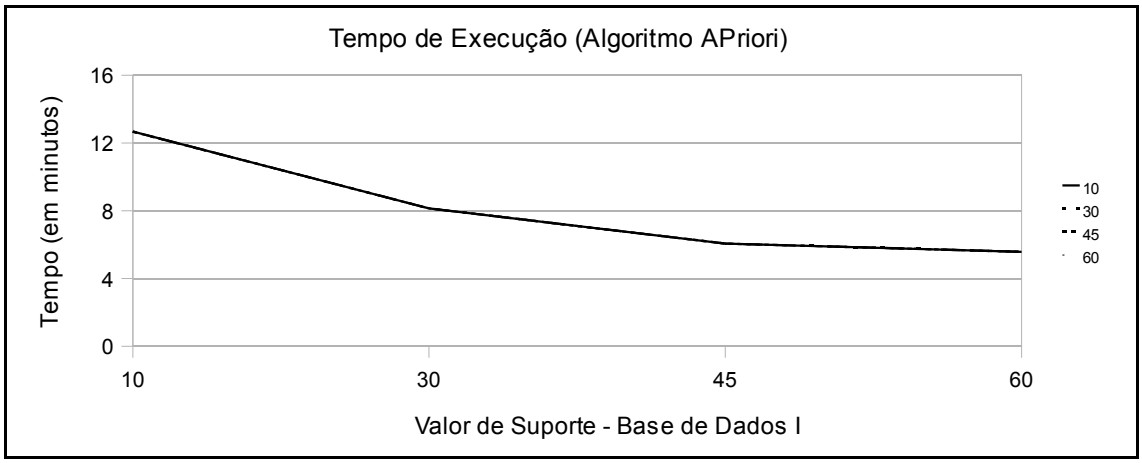


Figura 6.2: Tempo de Execução do Algoritmo *APriori* (Base de Dados I)

A Figura 6.3 apresenta os dados computados de forma estatística da quantidade de regras geradas pelo algoritmo *APriori*.

Para a representação dos dados que foram extraídos da base de dados II, serão apresentados de forma análoga aos gráficos apresentados nas Figuras 6.1, 6.2 e 6.3 referentes à base de dados I, são representados os gráficos com as informações pertinentes a base de dados II.

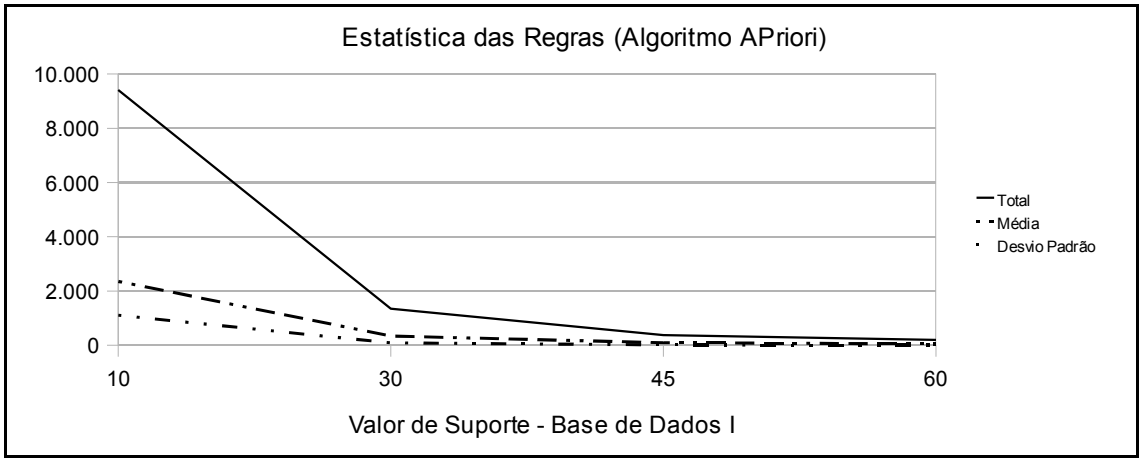


Figura 6.3: Estatística das Regras Geradas pelo Algoritmo *APriori* (Base de Dados I).

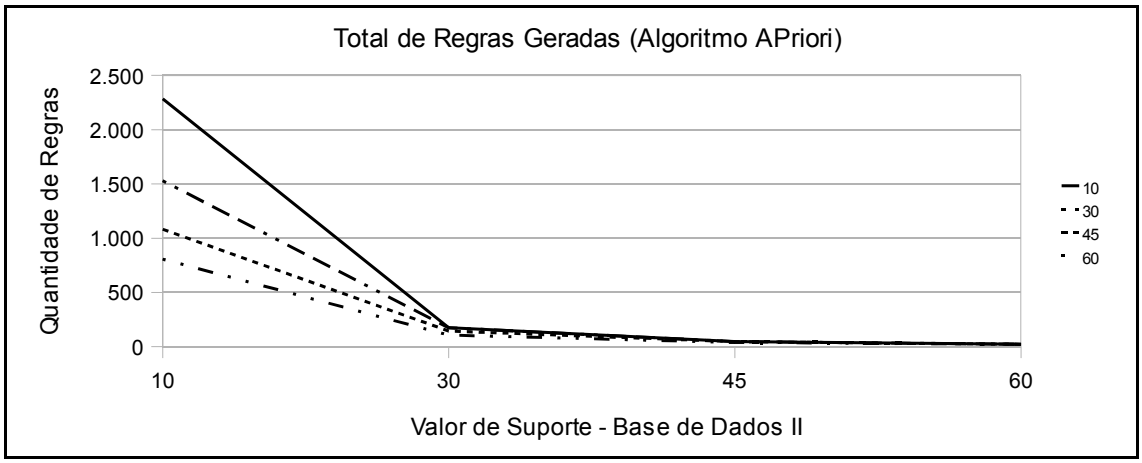


Figura 6.4: Quantidade de Regras Geradas pelo Algoritmo *APriori* (Base de Dados II).

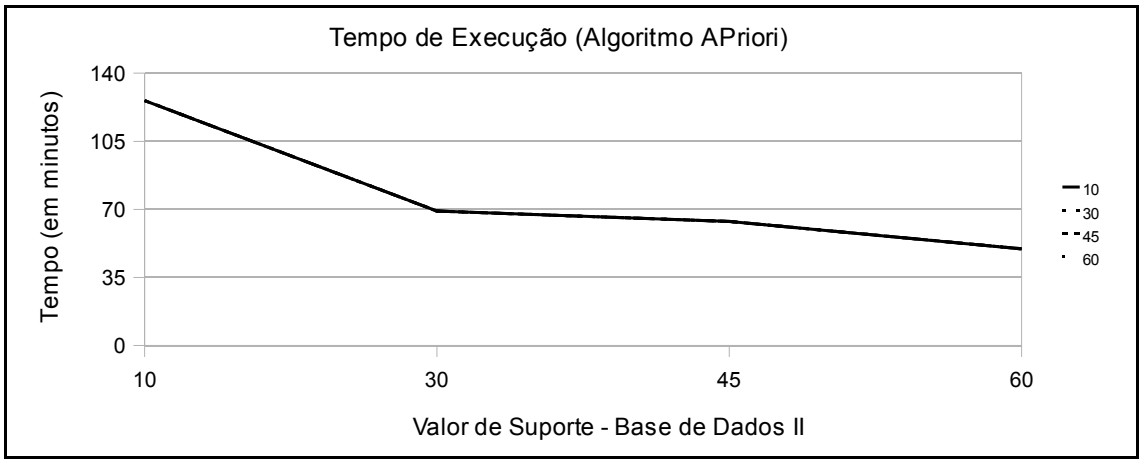


Figura 6.5: Tempo de Execução do Algoritmo *APriori* (Base de Dados II).

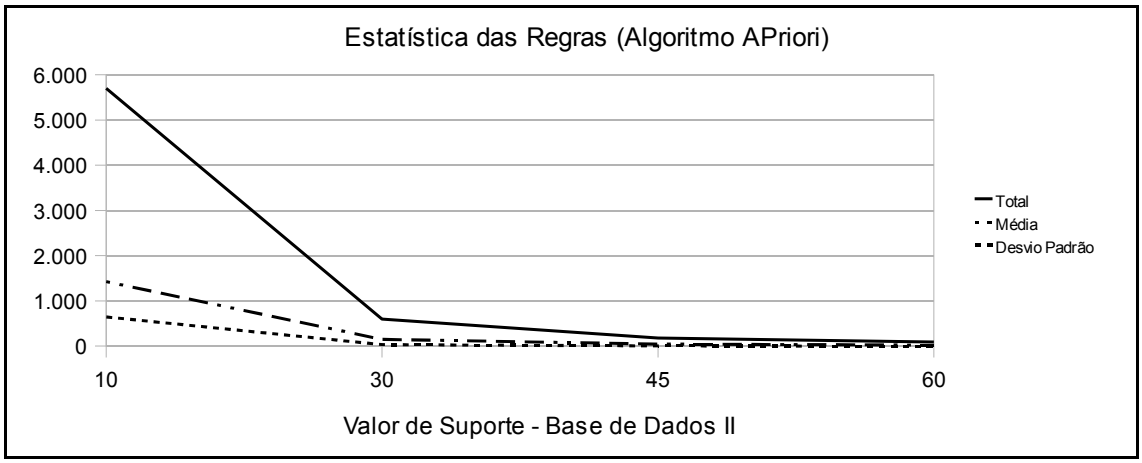


Figura 6.6: Estatística das Regras Geradas pelo Algoritmo *APriori* (Base de Dados II).

6.5. Aplicação do Algoritmo *ZigZag*

O algoritmo *ZigZag* foi empregado seguindo a metodologia e de forma análoga ao algoritmo descrito na seção 6.1.2 (*APriori*). Porém, este algoritmo possui a característica de extração de regras de forma Incremental, o que o difere do algoritmo *APriori* que efetua a extração de regras de forma normal.

Partindo desse princípio, levando em consideração que deve-se utilizar para empregar este algoritmo, as mesmas bases de dados bem como as mesmas formas definida para a extração das regras, descritas na seção 6.1. Também foi adotado o desenvolvimento de um *script* que atendesse os parâmetros exigentes pelo programa do algoritmo *ZigZag*.

Dado o formato incremental deste algoritmo, houve a necessidade de criar uma sistemática para que pudesse atender os parâmetros que o programa *ZigZag* exige. Desta forma, foi convencionado que as bases de dados seriam particionadas em duas partes iguais, onde a primeira parte representa a carga inicial dos dados contendo 50% do tamanho total da base, os outros 50% foram fracionados em 10 partes iguais, representando 5% do total da base de dados, como demonstrado na Figura 5.4 do Capítulo anterior.

Com isto, deu-se início à extração de regras através do algoritmo *ZigZag,* às quais são apresentadas na Tabela 6.2.

6.5.1 Criação dos *Scripts* para o Algoritmo *ZigZag*

Como mencionado acima, a justificativa para a geração de *scritp* foi a quantidade de arquivos que deverão ser gerados. Os *scripts* criados, devem assumir os parâmetros exigidos por parte do programa *zigzag* que gera as regras de associação. Desta forma, foi desenvolvido

um *script* denominado de *script pai*, o qual faz uma chamada para os 16 *scripts* denominados filhos referentes aos valores empregados de *suporte* e *confiança* 10%, 30%, 45% e 60%, dispostos da seguinte forma:

***scripts* pai:** executa_Zig1 e executa_Zig2

scripts filhos: s10c10_1, s10c30_1, s10c45_1, s10c60_1, s30c10_1, s30c30_1, s30c45_1, s30c60_1, s45c10_1, s45c30_1, s45c45_1, s45c60_1,s60c10_1, s60c30_1, s60c45_1, s60c60_1.

Os *scripts* denominados filhos são os responsáveis pela geração das regras de associação. É neste momento que deve-se fazer o processo incremental dos dados para aplicar o algoritmo. Partindo desta premissa, a execução do programa *zigzag* divide-se em duas partes, a primeira denominada carga inicial, representada por 50% do total da base de dados utilizada e as 10 partes incrementais, fracionadas do restante da base de dados que representa 50%. Entretanto, não existe uma sistemática diferente para aplicação das partes das bases de dados, o que muda é o tamanho da base que se passa como parâmetro, descrito da seguinte forma: tamanho de carga = 50% e tamanho incremental = 5% da segunda parte da base. Desta forma, cada *script* contém as seguintes instruções:

Bases de dados: Seguindo as definições de geração da base de dados, foram empregadas as mesmas bases utilizadas no algoritmo *APriori*. BDDTS_janjun2005.Apr contendo um total de 482.286 registros e BDDTS_jandez2005.Apr com um total de 5.075.715 de registros.

Fracionamento das bases:

Base de Dados I

Carga inicial .....: $\frac{482.286}{2} = 241.142 \quad 50\%.$

Parte incremental .....: $\frac{241.142}{10} = 24.114 \quad 5\%.$

Base de Dados II

Carga inicial .....: $\frac{5.075.715}{2} = 2.537.857 \quad 50\%.$

Parte incremental .....: $\frac{2.537.857}{10} = 253.786 \quad 5\%.$

A partir deste momento, deram início à execução do algoritmo:

./zigzag -i Bases/BDDTS_janjun2005.Apr -o resultados/ZigZag/Regras_ZIG_s10_c10.txt -p resultados/ZigZag/Res_ZigZag_s10c10_BDDTS_JanJun -s10 -c10 -m 241142.00 >> resultados/ZigZag/Historico_ZIG_s10_c10.txt

./zigzag: programa executável para a extração das regras;
-i Bases/BDDTS_janjun2005.Apr: parâmetro que define qual base de dados será aplicada, neste caso a primeira base de dados;
-o resultados/ZigZag/Regras_ZIG_s10_c10.txt: parâmetro que extrai as regras, direcionando para um arquivo definido;
-p resultados/ZigZag/Res_ZigZag_s10c10_BDDTS_JanJun: parâmetro que extrai os resultado das fases incrementais do algoritmo direcionando para um arquivo definido;
-s10: valor de suporte;
-c10: valor de confiança;
-m 241142.00**:** valor da parte Incremental. É neste parâmetro que se dá o processo incremental do algoritmo, em virtude dos valores passados como parâmetros até alcançar 100% do tamanho da base de dados que se deseja utilizar. Neste caso, 50% do valor da Base de dados I.
>> resultados/ZigZag/Historico_ZIG_s10_c10.txt: Opcional, apenas direciona o *display* do andamento do algoritmo para um arquivo. Neste caso, foi utilizado este direcional para um arquivo, devido às informações de tempo de processamento que forma computados.

A partir desta descrição, o algoritmo foi empregado alternando-se os valores adicionado para cada interação baseados no parâmetro **-m** e nos nomes dos arquivos a serem gerados, fazendo com isto 11 interações como demonstra o exemplo abaixo:

Primeira fase – *Carga de dados*
./zigzag -i Bases/BDDTS_janjun2005.Apr -o resultados/ZigZag/Regras_ZIG_s10_c10.txt -p resultados/ZigZag/Res_ZigZag_s10c10_BDDTS_JanJun -s10 -c10 -m 241142.00 >> resultados/ZigZag/Historico_ZIG_s10_c10.txt

Segunda fase – *Parte incremental - 01*

./zigzag -i Bases/BDDTS_janjun2005.Apr -o resultados/ZigZag/Regras_ZIG_s10_c10.txt -p resultados/ZigZag/Res_ZigZag_s10c10_BDDTS_JanJun -s10 -c10 -m 24114.00 >> resultados/ZigZag/Historico_ZIG_s10_c10.txt

.

.

.

Décima primeira fase – *Parte incremental - 10*

./zigzag -i Bases/BDDTS_janjun2005.Apr -o resultados/ZigZag/Regras_ZIG_s10_c10.txt -p resultados/ZigZag/Res_ZigZag_s10c10_BDDTS_JanJun -s10 -c10 -m 24114.00 >> resultados/ZigZag/Historico_ZIG_s10_c10.txt

Fim das iterações.

Observe que em todas estas fases, não houve mudança nos nomes dados de arquivos que recebem os valores computados. Cada um dos 16 *scripts* contém estas Etapas de geração, por isto que foram criados 18 *scripts* para este processo, os quais dois são *scripts* pai (um para cada base de dados) e o restante filhos de acordo com as bases de dados aplicada. Dessa forma a linha de comando utilizada para gerar as regras é a seguinte:

$executa_Zig1

//*Script* contendo interações pertinentes à primeira base de dados;

$executa_Zig2

//*Script* contendo interações pertinentes à primeira base de dados;

6.5.2 Dados de Entrada para o Algoritmo *ZigZag*

Nesta etapa, as bases de dados utilizadas foram as mesmas descritas na seção 6.1.2.1 de Dados de Entrada.

6.5.3 Resultados Obtidos

Seguindo o modelo empregado na seção anterior 6.1.2.3 Resultados Obtidos, os resultados foram:

1919 => 30000030 210(S 10.092643%, C 10.111344%)

1919 42042 => 77777 500048(S 10.059674%, C 10.184307%)

1919 42042 => 77777 992 30000025(S 10.141577%, C 10.267223%)
1919 => 500011(S 10.082690%, C 10.101373%)

.
.
.

9991010 88888 210 => 992(S 11.646091%, C 100.000000%)

Os dados extraídos acima, representam dados empregados à valores de suporte e confiança igual a 10%. Em função da quantidade de arquivos e principalmente do tamanho dos arquivos gerados, serão adotados para apresentação deste trabalho as seguintes regras iguais às empregadas no algoritmo APriori, tendo em vista que pretende-se efetuar uma comparação das regras extraídas. Entretanto, o algoritmo ZigZag ao gerar as mesmas regras, apenas as distribuem de forma diferente ao algoritmo APriori, sempre respeitando a condicional que é o fator importante para a interpretação das regras. Dessa forma, as regras são:

BD I.: 992 1919 190000011 210 => 9991010	11,60	37,66
BD II: 992 1919 42042 190000011 => 9991010	12,81	34,54
BD I.: 992 1919 190000011 => 77777	19,43	44,34
BD II: 992 1919 190000011 => 77777	14,37	38,39
BD I.: 992 1919 9991010 => 88888	17,05	60,20
BD II: 992 1919 9991010 => 88888	13,07	64,54

6.5.4 Legendas para o Algoritmo *ZigZag*

As legendas são as mesmas empregadas na seção 6.1.2.4 Legendas.

6.5.5 Análise dos Resultados Obtidos pelo Algoritmo *ZigZag*

O algoritmo *ZigZag* utilizado neste trabalho, efetuou uma única iteração para a extração das regras, aplicado a cada valor de suporte e confiança.

Regra 1: 992 1919 190000011 210 => 9991010 11,60 37,66

Base de Dados I

Tradução da Regra:

- Das pessoas que possuem um plano de saúde tipo “pessoa jurídica”, estão ativos no plano, utilizaram de algum recurso médico/hospitalar de um prestador Ativo, estão com o seu plano com o *status* de ativo e são casados. 37,66% possuem uma idade igual ou superior a 59 anos. (*suporte* de 11,60% e *confiança* de 37,66%).

Regra 1: 992 1919 42042 190000011 => 9991010 12,81 34,54

Base de Dados II

Tradução da Regra:

- Das pessoas que possuem um plano de saúde tipo “pessoa jurídica”, estão ativos no plano, utilizaram de algum recurso médico/hospitalar de um prestador Ativo, estão com o seu plano com o *status* de ativo e são casados. 34,54% possuem uma idade igual ou superior a 59 anos.(*suporte* de 12,81% e *confiança* de 34,54%).

Considerações:

- Esta é uma informação muito importante, pois tende a chamar a atenção das empresas no sentido de um trabalho de medicina preventiva. Pois além de utilizar o plano, esta pessoa necessitou de algum tipo de exame ou serviço especializado, em função da informação de um prestador de serviços médicos estar com o status Ativo para esta regra. Ou seja, a informação de um prestador só aparece, caso seja necessário a sua utilização.

Regra 2: 992 1919 190000011 => 77777 19,43 44,34

Base de Dados I

Tradução da Regra:

- 44,34% dos usuários de um plano de saúde do sexo Masculino, possuem um plano de saúde do tipo pessoa jurídica, estão ativos neste plano e são casados.(*suporte* de 19,43% e *confiança* de 44,34%).

Regra 2: 992 1919 190000011 => 77777 14,37 38,39

Base de Dados II

Tradução da Regra:

- 38,39% dos usuários de um plano de saúde do sexo Masculino, possuem um plano de saúde do tipo pessoa jurídica, estão ativos no plano e são casados. (*suporte* de 14,37% e *confiança* de 38,39%).

Considerações:

- Esta regra, demonstra uma tendência das pessoas do sexo masculinos terem um plano de saúde, principalmente por se tratar de um benefício que as empresas adotam ao adquirir um plano de saúde empresarial para seus funcionários.

Regra 3: 992 1919 9991010 => 88888 17,05 60,20

Base de Dados I

Tradução:

- Das pessoas que possuem um plano de saúde do tipo pessoa jurídica, estão ativos neste plano, tem idade maior ou igual a 59 anos, 60,20% são do sexo Feminino. (*suporte* de 17,05% e *confiança* de 60,20%).

Base de Dados II

Regra 3: 992 1919 9991010 => 88888 13,07 64,54

Tradução:

- Das pessoas que possuem um plano de saúde do tipo pessoa jurídica, estão ativos neste plano, tem idade maior ou igual a 59 anos, 64,54% são do sexo Feminino. (*suporte* de 13,07% e *confiança* de 64,54%).

Considerações:

- Esta regra, demonstra um fator muito importante para a área da saúde, pois demonstra um número significativo de pessoas com mais de 59 anos utilizando um plano de saúde. Isto só é possível dado ao incentivo das empresas em adquirir planos empresariais para seus funcionários. É sabido que as pessoas com mais de 59 anos, possuem um complicador com relação à saúde, pois todos os planos de saúde praticam preços elevados em decorrência da faixa etária. Porém estes preços são fiscalizados e administrados pela ANS (Agência Nacional de Saúde Suplementar).

Observação: A interpretação dos resultados acima descritos, são as mesmas descritas na seção 6.1.2.5 do algoritmo *APriori*. Preservando as regras para uma comparação dos resultados obtidos por ambos os algoritmos.

6.5.6 Considerações Finais da aplicação do Algoritmo *ZigZag*

O estudo da aplicação do algoritmo *ZigZag* proposto neste trabalho, obteve um desempenho consistente, apesar dos valores aplicados de *suporte* e *confiança* serem relativamente altos. Indo de encontro aos resultados obtidos pelo algoritmo APriori, o algoritmo *ZigZag* gerou resultados plausíveis e úteis, porém caso desejar melhores resultados, deverá ser efetuado um estudo preliminar das necessidades que se deseja alcançar, evidentemente que isto se aplica também ao algoritmo *APriori*, não esquecendo de considerar o volume de dados que é um fator complicador para a utilização deste algoritmo.

6.6. Apresentação dos Resultados do Algoritmo *ZigZag*

Como já era esperado, o algoritmo *ZigZag* possui um desempenho superior ao algoritmo *APriori*, gerando em sua maioria, a mesma quantidade de regras, como pode-se ver na Tabela 6.1. De forma análoga, ao empregar o algoritmo *ZigZag* à segunda,0 .60base de dados, também obteve um aumento significativo em seu tempo de processamento, porém houve uma redução na quantidade das regras extraídas, indo de encontro ao mesmo comportamento do algoritmo *APriori*. Evidentemente que não considera-se a qualidade das regras empregadas, assunto que será discutido no próximo capítulo. Uma particularidade é que estas regras geradas iguais, em sua maioria foram encontradas a partir do valor de *suporte* e *confiança* de 10% e 30%. Assim, pode-se dizer que ao empregar o algoritmo *ZigZag* objetivando uma comparação dos resultados, quanto menor for o valor de suporte e confiança, maior será a probabilidade de gerar regras extremamente iguais às regras geradas pelo algoritmo *APriori*. Dessa forma é apresentado na Tabela 6.3 a representação dos valores de quantidade de regras e tempo de processamento empregados neste algoritmo.

Tabela 6.3: Quantidade de Regras e Tempo de Processamento Gerados pelo Algoritmo *ZigZag*.

Experimentos	***Suporte* (%)**	***Confiança* (%)**	**Base de Dados I** *Algoritmo ZigZag*		**Base de Dados II** *Algoritmo ZigZag*	
			Quantidade de Regras	**Tempo de Processamento** *(minutos)*	**Quantidade de Regras**	**Tempo de Processamento** *(minutos)*
1	10	10	3.748	1,090	2.284	11,500
2	10	30	2.653	1,100	2.653	11,890
3	10	45	1.803	1,120	1.082	11,440
4	10	60	1.205	1,090	807	11,430
5	30	10	342	0,290	174	2,580
6	30	30	342	0,310	174	2,580
7	30	45	265	0,320	144	2,610
8	30	60	196	0,028	107	2,650
9	45	10	98	0,140	46	1,350
10	45	30	98	0,130	46	1,380
11	45	45	98	0,150	46	1,370
12	45	60	74	0,100	39	1,380
13	60	10	50	0,080	22	0,650
14	60	30	50	0,070	22	0,670
15	60	45	50	0,060	22	0,650
16	60	60	50	0,050	22	0,640

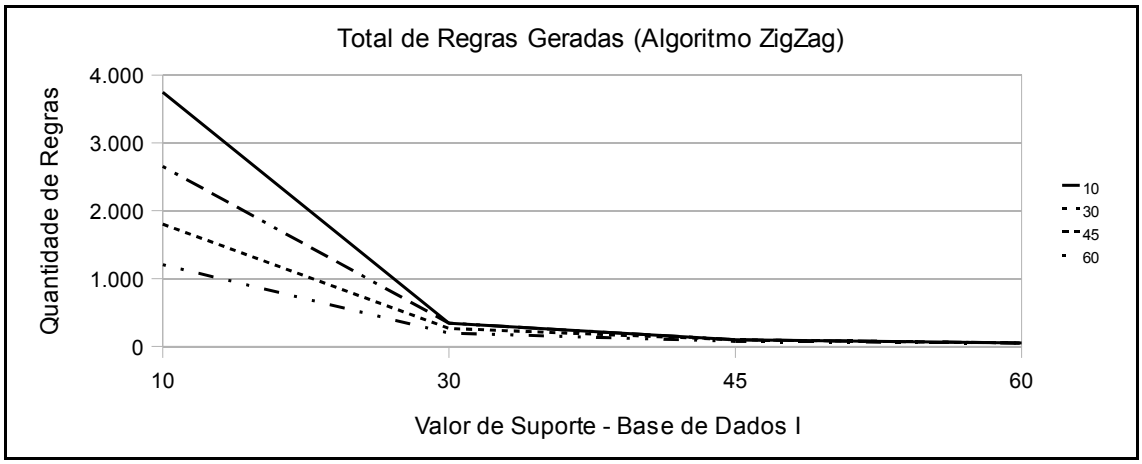


Figura 6.7: Quantidade de Regras Geradas pelo Algoritmo *ZigZag* (Base de Dados I)

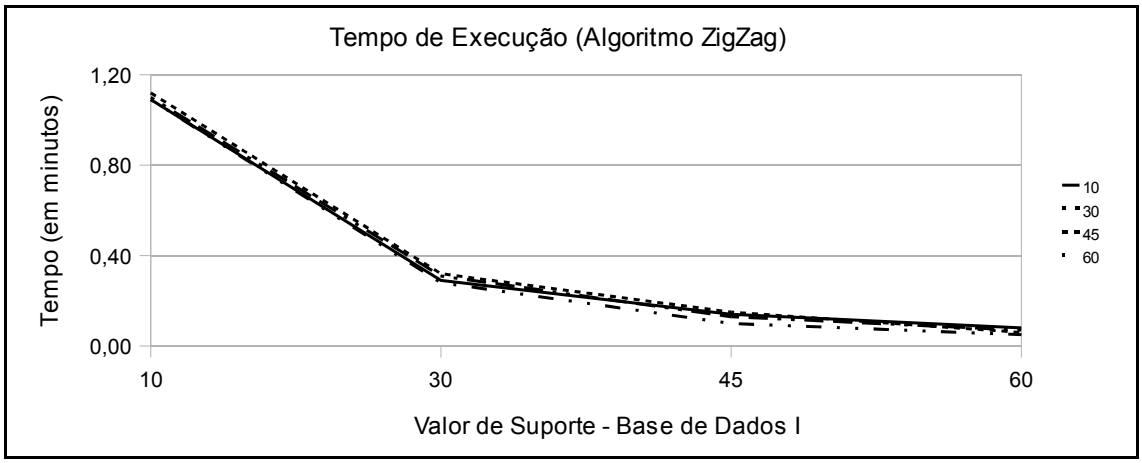


Figura 6.8: Tempo de Execução do Algoritmo *ZigZag* (Base de Dados I)

A Figura 6.7 é um demonstrativo da relação entre a quantidade de regras geradas e dos valores de *suporte* e *confiança*, bem como o tempo de processamento empregados no algoritmo *ZigZag*. Entretanto a diferença da quantidade das regras geradas com um *suporte* e *confiança* igual a 10%, não são tão expressivas quanto as regras extraídas pelo algoritmo *APriori*. Demonstrando com isto, que nas extrações das regras o algoritmo *ZigZag* por se tratar de um algoritmo que trabalha de forma linear com as máximas freqüência de itens, extrai de forma regular todas as regras que satisfazem as condições pré-estabelecidas na aplicação do algoritmo.

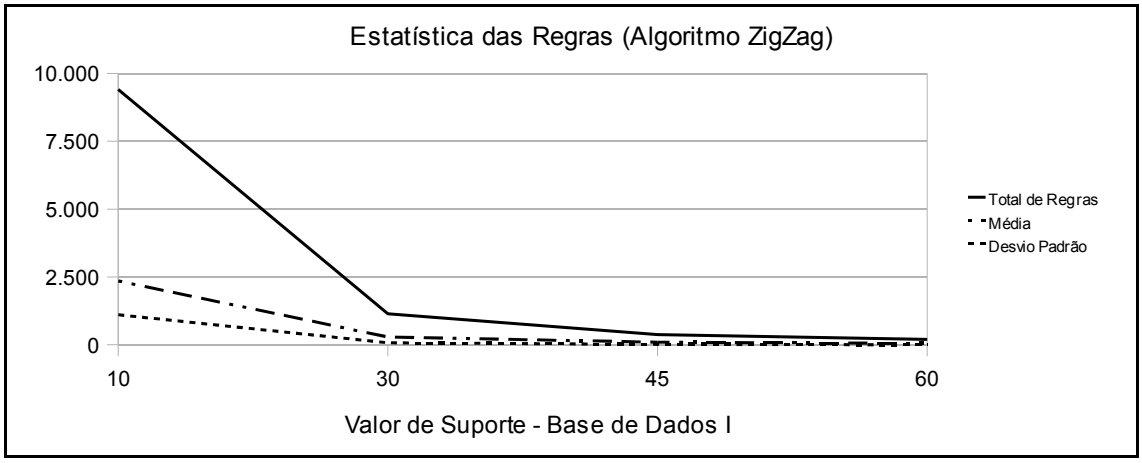


Figura 6.9: Estatística das Regras Geradas pelo Algoritmo *ZigZag* (Base de Dados II).

Para a representação dos dados que foram extraídos da base de dados II, serão demonstrados de forma análoga aos gráficos demonstrados nas Figuras 6.7, 6.8 e 6.9 referentes à base de dados I, são representados os gráficos com as informações pertinentes a base de dados II.

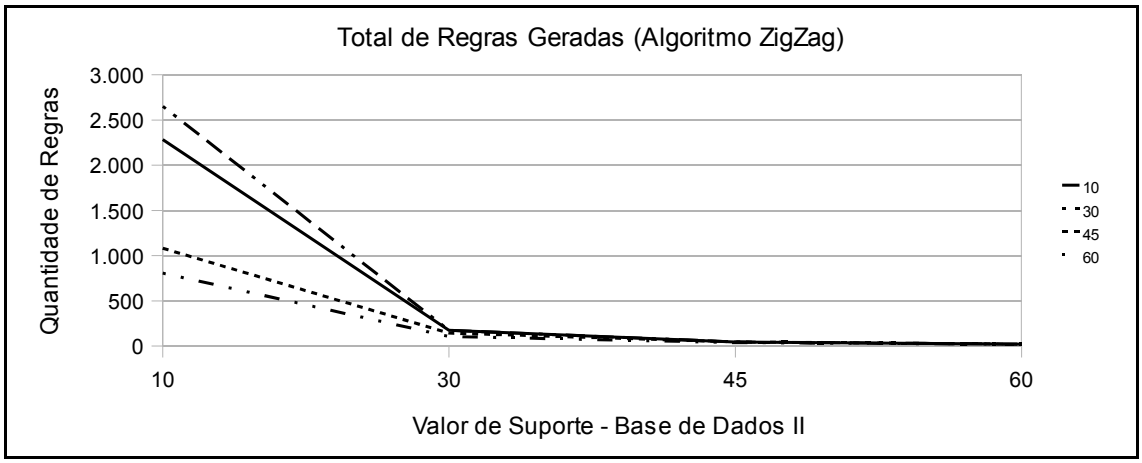


Figura 6.10: Quantidade de Regras Geradas pelo Algoritmo *ZigZag* (Base de Dados II)

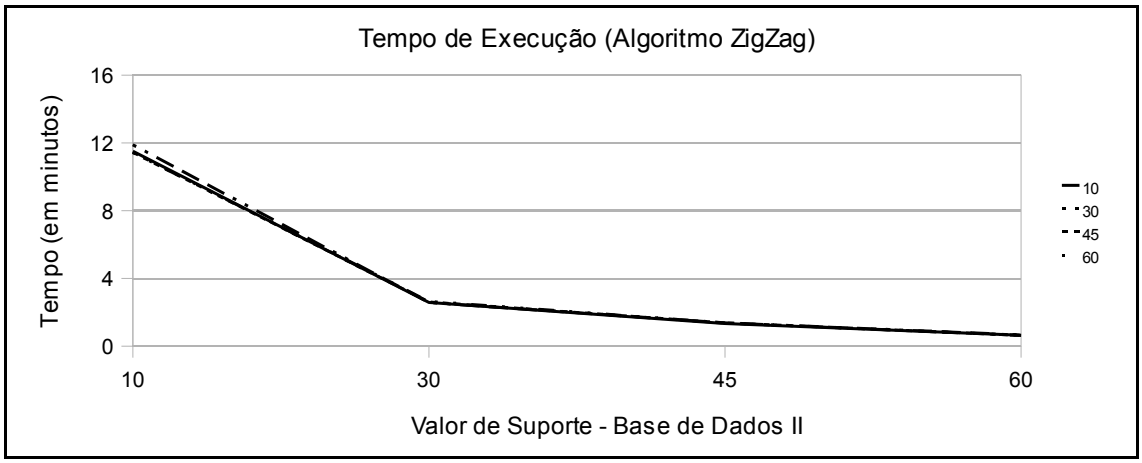


Figura 6.11: Tempo de Execução do Algoritmo *ZigZag* (Base de Dados II)

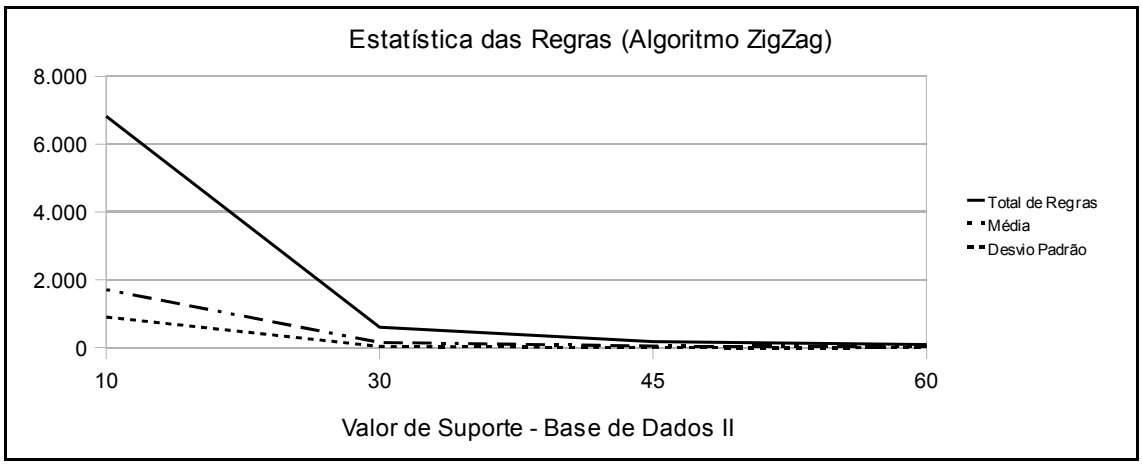


Figura 6.12: Estatística das Regras Geradas pelo Algoritmo *ZigZag* (Base de Dados II).

6.6.1. Comparação das Regras Extraídas dos Algoritmos *APriori* e *ZigZag*

Esta seção versa sobre uma comparação das regras extraídas pelos algoritmo *APriori* e *ZigZag*. A abordagem empregada nesta seção é de cunho comparativo, onde serão demonstrados os resultados obtidos por ambos os algoritmos. Em decorrência do objetivo de validar a utilização do algoritmo *ZigZag* levando como referência o algoritmo *APriori*.

Tabela 6.4: Tabela Comparativa dos Dados *APriori* e *ZigZag* (Base de Dados I).

BASE DE DADOS I		***Apriori***			***Zigzag***		
Qtde	**Medidas**	**Regras**	**Iguais**	**Percentual**	**Regras**	**Iguais**	**Percentual**
1	S10c10	3.748	1.294	34,53%	3.748	1.294	34,53%
2	S10c30	2.653	967	36,45%	2.653	967	36,45%
3	S10c45	1.803	708	39,27%	1.803	708	39,27%
4	S10s60	1.205	462	38,34%	1.205	462	38,34%
5	S30c10	406	198	48,77%	342	198	57,89%
6	S30c30	406	198	48,77%	342	198	57,89%
7	S30c45	315	157	49,84%	265	157	59,25%
8	S30c60	224	112	50,00%	196	112	57,14%
9	S45c10	98	74	75,51%	98	74	75,51%
10	S45c30	98	74	75,51%	98	74	75,51%
11	S45c45	98	74	75,51%	98	74	75,51%
12	S45c60	74	50	67,57%	74	50	67,57%
13	S60c10	50	26	52,00%	50	26	52,00%
14	S60c30	50	26	52,00%	50	26	52,00%
15	S60c45	50	26	52,00%	50	26	52,00%
16	S60c60	50	26	52,00%	50	26	52,00%
Total geral das Regras		**11.328**	**4.472**	**----**	**11.122**	**4.472**	**----**

Através da tabela 6.3 é possível se ter uma real idéia da evolução dos algoritmos empregados neste trabalho. A premissa básica desta Tabela 6.4 é de cunho comparativo, dessa forma, foram equacionados os valores obtidos pelos algoritmos *APriori* e *ZigZag*, e distribuídos da seguinte forma:

- Medidas: é o valor de *suporte* e *confiança* para cada regra;
- Regras: é o total de regras encontradas;
- Iguais: é o total de regras encontradas IGUAIS geradas pelos algoritmos. Esta nesta parte, foi utilizado um banco de dados contendo as regras, onde submeteu-se uma *query* que fizesse tal comparação.

- Percentual: é equação de regras geradas através dos campo Regras com o campo Iguais, apresentando um percentual das regras geradas pelos algoritmos com o total de regras encontradas idênticas.

Com isto, pôde-se observar que 25% das regras extraídas da Base de Dados I são diferentes, pois o Algoritmo *APriori* gerou uma quantidade maior para os valores de suporte igual a 30% conforme Tabela 6.5. Os demais 75% possuem a mesma quantidade, porém não necessariamente as mesmas regras.

Tabela 6.5: Quantidade de Regras Diferentes Extraídas Pelos Algoritmos

BASE DE DADOS I		***Apriori***			***Zigzag***		
Qtde	**Medidas**	**Regras**	**Iguais**	**Percentual**	**Regras**	**Iguais**	**Percentual**
5	S30c10	406	198	48,77%	342	198	57,89%
6	S30c30	406	198	48,77%	342	198	57,89%
7	S30c45	315	157	49,84%	265	157	59,25%
8	S30c60	224	112	50,00%	196	112	57,14%

De maneira análoga ao número de quantidade de regras extraídas pelos algoritmos, por motivo do algoritmo *ZigZag* ter gerado uma quantidade menor de regras idênticas, automaticamente o seu percentual também demonstrou elevação. Uma observação importante é que esta diferença, só aconteceu para os valores de suporte e confiança igual a 30% nesta base, em decorrência do algoritmo *APriori* efetuar uma busca geral da associação disponível, diferentemente do algoritmo *ZigZag* que só extrai dos itens máximos freqüentes.

Para um melhor entendimento, estes valores são demonstrados os gráficos nas Figura 6.13 e 6.14 os quais demonstram a evolução do total das regras geradas pelo algoritmo *APriori* aplicados nas duas bases de dados. Já na Figura 6.15, é apresentado o número total de regras geradas Iguais por ambos os algoritmos (*APriori* e *ZigZag*), esta figura tem por objetivo uma comparação na aplicação dos algoritmos para ambas as bases de dados (base de dados I e II). Na Figura 6.16, é apresentada uma estatística geral do tempo de processamento que ambos os algoritmos tiveram como resultado para a extração das regras.

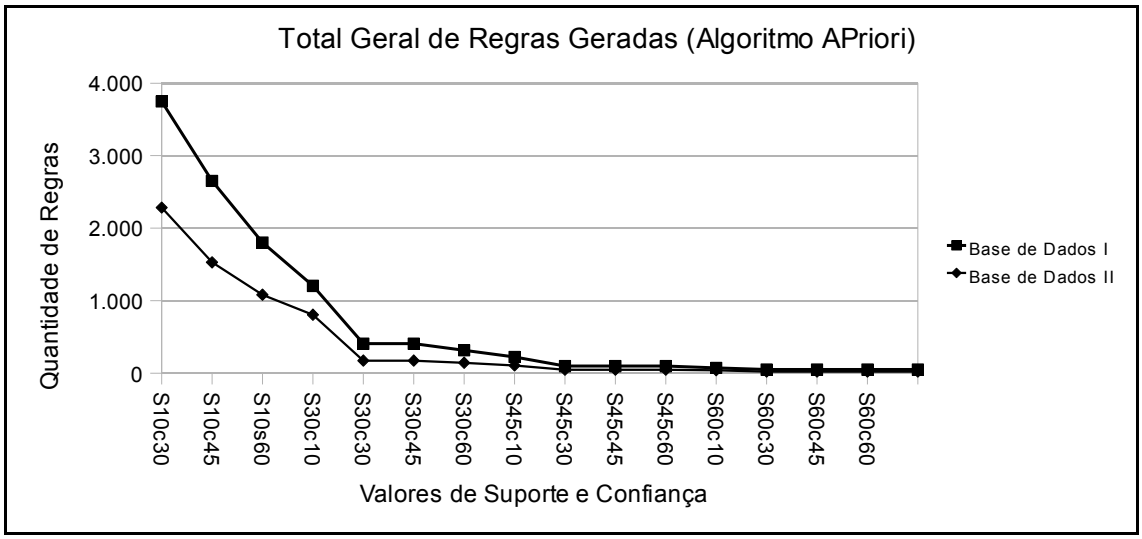


Figura 6.13: Número de Regras Geradas Pelo Algoritmo *APriori*

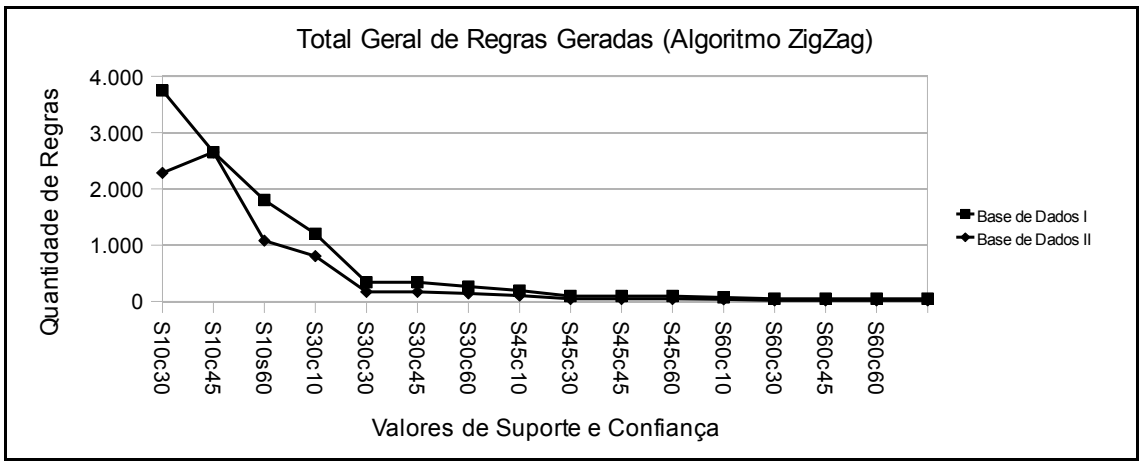


Figura 6.14: Número de Regras Geradas Pelo Algoritmo *ZigZag*

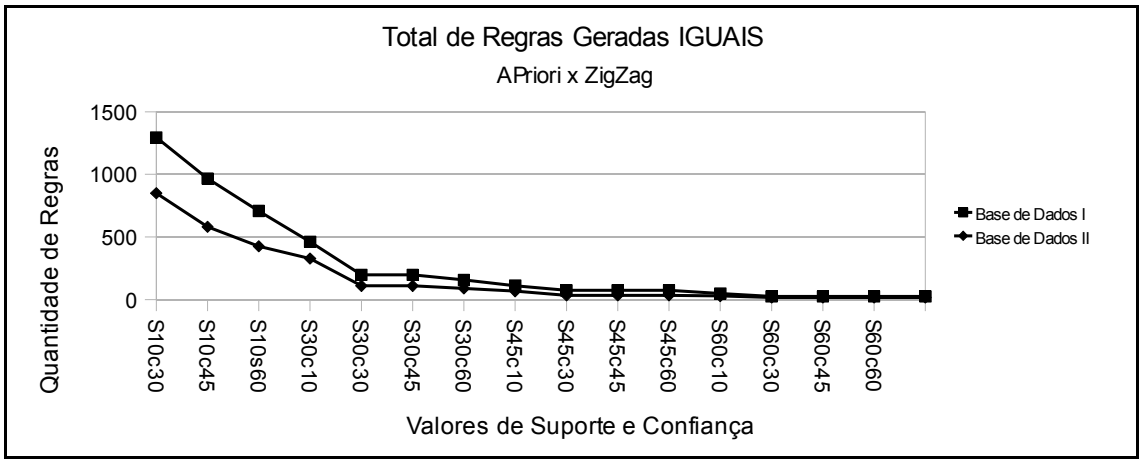


Figura 6.15: Número Total de Regras Iguais Geradas Pelo Algoritmo *APriori*.

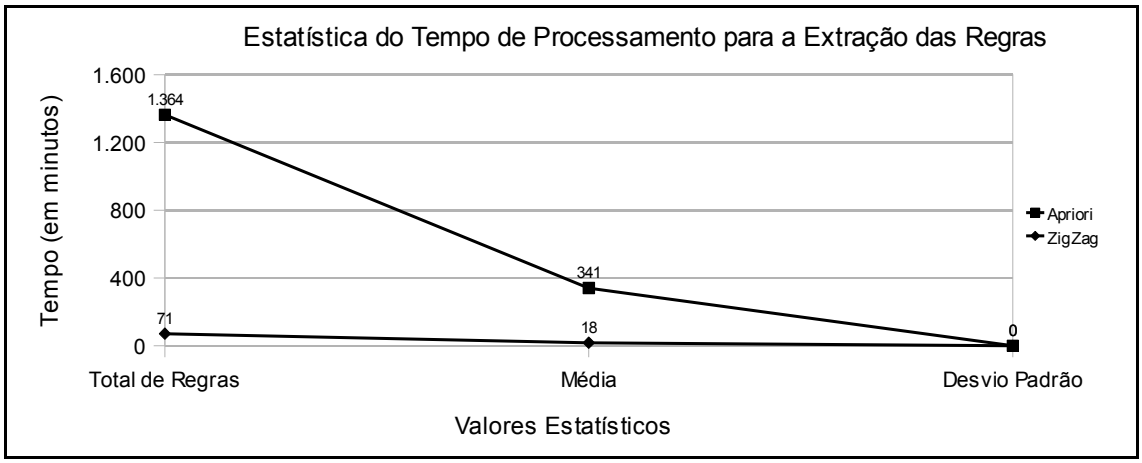


Figura 6.16: Estatística do Tempo de Processamento para a Extração das Regras

6.7. Considerações Finais da Seção

Nesta seção foram apresentados todos os dados gerados pelos algoritmos *APriori* e *ZigZag*, de forma a representar uma análise completa do desempenho de cada algoritmo. Este estudo demonstrou que o algoritmo *ZigZag* se mostrou coerente, em decorrência dos dados apresentados se aproximarem dos dados gerados pelo algoritmo *APriori* que é o referencial.

Assim, este estudo transcorreu nesta seção de forma quantitativa e não qualitativa, assunto este que será tratado na próxima seção.

6.8. Medidas de Avaliação das Regras

Nesta seção são apresentadas as medidas de avaliação de regras, as quais foram pré-selecionadas e descritas na seção 6.1.2.3. Dessa forma as regras aplicadas para este estudo estão dispostas da seguinte forma:

Tabela 6.6: Regras Selecionadas para Avaliação

Base de Dados	Algoritmo	Sup/Conf	Código da Regra	Regras	*Suporte*	Confiança
1	*APriori*	1010	1	992 1919 9991010 => 88888	17,05	60,20
1	*APriori*	1030	2	992 1919 9991010 => 88888	17,05	60,20
1	*APriori*	1045	3	210 992 1919 9991010 => 88888	11,65	60,73
1	*APriori*	1060	4	992 1919 9991010 => 88888	17,65	60,20
2	*APriori*	1010	5	992 1919 42042 190000011 => 9991010	12,81	34,54
2	*APriori*	1030	6	992 1919 190000011 => 77777	14,37	38,39
2	*APriori*	1045	7	992 1919 9991010 => 88888	13,07	64,54
2	*APriori*	1060	8	992 1919 9991010 => 88888	13,07	64,54
1	*ZigZag*	1010	9	992 1919 190000011 => 77777	19,43	44,34
1	*ZigZag*	1030	10	992 1919 9991010 => 88888	17,05	60,20
1	*ZigZag*	1045	11	992 1919 9991010 => 88888	17,05	60,20
1	*ZigZag*	1060	12	992 1919 9991010 => 88888	17,05	60,20
2	*ZigZag*	1010	13	992 1919 9991010 => 88888	1307	64,54
2	*ZigZag*	1030	14	992 1919 190000011 => 77777 42042 210	14,11	32,21
2	*ZigZag*	1045	15	992 1919 9991010 => 88888	13,07	64,54
2	*ZigZag*	1060	16	992 1919 9991010 => 88888	13,07	64,54

As regras apresentadas na Tabela 6.6 será representada pelo código de regras para uma melhor apresentação e distribuição das informações extraídas. Dessa forma, foram escolhidas aleatoriamente um conjunto de regras para a aplicação das medidas, sempre respeitando os valores de *suporte* e *confiança*. Com isto, os arquivos de regras pertinentes às regras descritas na tabela 6.4, foram aplicadas a um programa desenvolvido em java, cujo objetivo é gerar a Tabela de Contingência, descrita no capítulo 4 e a sistemática para geração da tabela, descrita no no capítulo 5. Esta tabela é fundamental para que se possa aplicar as medidas de avaliação das regras. A Figura 6.4 mostra as regras aplicadas à Tabela de contingência, bem como os valores computados.

Tabela 6.7: Tabela de Contingência

CodReg	Tabela de Contingência				Valores Absolutos				
	Hb	~hb	h~b	~h~b	B	~B	H	~H	N
1	82.229	54.371	198.363	147.322	136.600	345.685	280.592	201.693	482.285
2	82.229	54.371	198.363	147.322	136.600	345.685	280.592	201.693	482.285
3	56.167	36.319	224.425	165.374	92.486	389.799	280.592	201.693	482.285
4	82.229	54.371	198.363	147.322	136.600	345.685	280.592	201.693	482.285
5	649.970	1.232.085	743.337	2.450.323	1.882.055	3.193.660	1.393.307	3.682.408	5.075.715
6	729.354	1.170.460	1.039.785	2.136.116	1.899.814	3.175.901	1.769.139	3.306.576	5.075.715
7	663.352	364.393	2.643.224	1.404.746	1.027.745	4.047.970	3.306.576	1.769.139	5.075.715
8	663.352	364.393	2.643.224	1.404.746	1.027.745	4.047.970	3.306.576	1.769.139	5.075.715
9	211.318	0	270.967	0	211.318	270.967	482.285	0	482.285
10	136.600	0	345.685	0	136.600	345.685	482.285	0	482.285
11	136.600	0	345.685	0	136.600	345.685	482.285	0	482.285
12	136.600	0	345.685	0	136.600	345.685	482.285	0	482.285
13	1.027.745	0	4.047.970	0	1.027.745	4.047.970	5.075.715	0	5.075.715
14	723.269	1.176.545	1.026.825	2.149.076	1.899.814	3.175.901	1.750.094	3.325.621	5.075.715
15	1.027.745	0	4.047.970	0	1.027.745	4.047.970	5.075.715	0	5.075.715
16	1.027.745	0	4.047.970	0	1.027.745	4.047.970	5.075.715	0	5.075.715

Objetivando explorar os resultados obtidos pelas medidas aplicadas às regras extraídas, foram utilizadas as 17 medidas disponíveis para avaliação de regras conforme descritos no Capítulo 5, as quais estão divididas em três grupos distintos: medidas genéricas. medidas relativas e medidas relativas com peso.

Dessa forma, foram imputados os dados pertinentes a cada regra escolhida conforme demonstra a Tabela 6.4, os quais são apresentada na Tabela 6.8 como segue:

Tabela 6.8: Tabela de Medidas Genéricas

CodReg	$Acc(R_n)$	$Err(R_)$	$NegRel(R_n)$	$Sens(R_n)$	$Spec(R_n)$	$Cov(R_n)$	$Sup(R_n)$	$Nov(R_n)$	Sat(Rn)
1	0,602	0,398	1,078	0,293	0,525	0,283	0,170	0,118	0,048
2	0,602	0,398	0,426	0,293	0,730	0,283	0,170	0,118	0,048
3	0,607	0,393	0,424	0,200	0,820	0,192	0,116	0,080	0,061
4	0,602	0,398	0,426	0,293	0,730	0,283	0,170	0,118	0,048
5	0,345	0,655	0,767	0,466	0,665	0,371	0,128	0,269	0,098
6	0,384	0,616	0,673	0,412	0,646	0,374	0,144	0,244	0,054
7	0,645	0,355	0,347	0,201	0,794	0,202	0,131	0,071	-0,017
8	0,645	0,355	0,347	0,201	0,794	0,202	0,131	0,071	-0,017
9	1,000	0,000	0,000	0,438	Não Aplicado	0,438	0,438	0,000	Não Aplicado
10	1,000	0,000	0,000	0,283	Não Aplicado	0,283	0,283	0,000	Não Aplicado
11	1,000	0,000	0,000	0,283	Não Aplicado	0,283	0,283	0,000	Não Aplicado
12	1,000	0,000	0,000	0,283	Não Aplicado	0,283	0,283	0,000	Não Aplicado
13	1,000	0,000	0,000	0,202	Não Aplicado	0,202	0,202	0,000	Não Aplicado
14	0,381	0,619	0,677	0,413	0,646	0,374	0,142	0,245	0,055
15	1,000	0,000	0,000	0,202	Não Aplicado	0,202	0,202	0,000	Não Aplicado
16	1,000	0,000	0,000	0,202	Não Aplicado	0,202	0,202	0,000	Não Aplicado

Tabela 6.9: Tabela de Medidas Relativas

	MEDIDAS RELATIVAS			
CodReg	*Racc(R)*	*RnegRel(R)*	*Rsens(R)*	*Rspec(R)*
1	0,020	0,008	0,010	0,014
2	0,020	0,008	0,010	0,014
3	0,026	0,006	0,008	0,012
4	0,020	0,008	0,010	0,014
5	0,071	0,042	0,096	0,036
6	0,035	0,021	0,038	0,020
7	-0,006	-0,002	-0,002	-0,003
8	-0,006	-0,002	-0,002	-0,003
9	0,000	0,000	0,000	Não aplicado
10	0,000	0,000	0,000	Não aplicado
11	0,000	0,000	0,000	Não aplicado
12	0,000	0,000	0,000	Não aplicado
13	0,000	0,000	0,000	Não aplicado
14	0,036	0,021	0,039	0,021
15	0,000	0,000	0,000	Não aplicado
16	0,000	0,000	0,000	Não aplicado

Tabela 6.10: Tabela de Medidas Relativas com Peso

	MEDIDAS RELATIVAS COM PESO			
CodReg	*WRAcc(R)*	*WRNegRel(R)*	*WRSens(R)*	*WRSpec(R)*
1	0,006	0,006	0,006	0,006
2	0,006	0,006	0,006	0,006
3	0,005	0,005	0,005	0,005
4	0,006	0,006	0,006	0,006
5	0,026	0,026	0,026	0,026
6	0,013	0,013	0,013	0,013
7	-0,001	-0,001	-0,001	-0,001
8	-0,001	-0,001	-0,001	-0,001
9	0,000	0,000	0,000	Não aplicado
10	0,000	0,000	0,000	Não aplicado
11	0,000	0,000	0,000	Não aplicado
12	0,000	0,000	0,000	Não aplicado
13	0,000	0,000	0,000	Não aplicado
14	0,013	0,013	0,013	0,013
15	0,000	0,000	0,000	Não aplicado
16	0,000	0,000	0,000	Não aplicado

6.9. Interpretação dos Resultados das Medidas

Através das Tabelas 6.6 à 610 apresentadas acima, nesta seção, serão interpretados os valores para análise e identificação das regras induzidas às medidas propostas.

6.9.1 Medidas Genéricas

Conforme informado, a interpretação das regras genéricas descritas na Tabela 6.8, segue da seguinte forma:

Através desta definição e dos dados imputados na Tabela 6.6, inicialmente foi adotado dois modelos de experimentos para análise. O primeiro experimento denominado Exp_I, aplica-se regras 9 à 16, o segundo experimento denominado Exp_II, aplica-se às regras de 1 a 8, como segue:

Para o experimento Exp_I, afirmar que para este experimento, as regras que atenderam a condição estas medidas foram:

- **Regra 9 a 12:** 992 1919 190000011 => 77777

  É uma regra extraída pelo algoritmo *ZigZag* com valores de *suporte* igual a 10% e *confiança* igual a 10%, 30%, 45% e 60% aplicada à base de dados I.

  – *x*% dos usuários de um plano de saúde do sexo Masculino, possuem um plano de saúde do tipo pessoa jurídica, estão ativos neste plano e são casados.

- **Regra 13 a 16:** 992 1919 9991010 => 88888

É uma regra extraída pelo algoritmo ZigZag com valores de *suporte* igual a 10% e *confiança* igual a 10%, 30%, 45% e 60% aplicada à base de dados II.

- Das pessoas que possuem um plano de saúde do tipo pessoa jurídica, estão ativos neste plano, tem idade maior ou igual a 59 anos, *x*% são do sexo Feminino.

Das 8 regras geradas pelo algoritmo *ZigZag*, apenas duas atenderam as medidas de Genéricas para os valores de *suporte* e *confiança.* Onde a regra 992 1919 190000011 => 7777, atendeu somente para a base de dados I e a regra 992 1919 9991010 => 88888, atendeu somente para a base de dados II. Porém, todas as duas regras não atenderam a medida *Sat*(R), em decorrência dos valores gerados pela Tabela de Contingência não estarem zerados, isto é, não foi detectada nenhuma incidência nas bases de dados, os quais referem-se na tabela de contingência como:

$\bar{h}b$ = é o número de exemplos para os quais *H* é falso e *B* é verdade;

$\overline{hb}$ = número de exemplos para os quais *H* é falso e *B* é falso;

Para o experimento Exp_II, afirmar que para este experimento, as regras que atenderam a condição estas medidas foram:

- **Regra 1 a 4:** Esta faixa de regras compreendem às regras geradas pela base de dados I, dada por:

992 1919 9991010 => 88888 - É uma regra extraída pelo algoritmo *APriori* com valores de *suporte* igual a 10% e *confiança* igual a 10%, 30%, 45% e 60%.

- Das pessoas que possuem um plano de saúde do tipo pessoa jurídica, estão ativos neste plano, tem idade maior ou igual a 59 anos, *x*% são do sexo Feminino.

- **Regra 5 a 8:** Esta faixa de regras compreendem regras pertinentes à base de dados II, as quais são:

- 992 1919 9991010 => 88888 - É uma regra extraída pelo algoritmo *APriori* com valores de *suporte* igual a 10% e *confiança* igual a 10%, 30%, 45% e 60%.

- 992 1919 190000011 => 77777 - *x*% dos usuários de um plano de saúde do sexo Masculino, possuem um plano de saúde do tipo pessoa jurídica, estão ativos no plano e são casados.

Desta forma, a interpretação dos resultados para as medidas genéricas são:

- Somente as regras 1,2,3,4,8 atenderam os requisitos definidos pela medida *Acc(R)*, tendo em vista o valor alcançado para cada regra;

- Conseqüentemente, as regras 5 e 6, atenderam os requisitos da medida *Err(R);*

- De acordo com a definição, a única regra que atendeu a medida *NegRel(R)*, foi a regra número 1, em decorrência da alta probabilidade de que *H* e *B* são falso;

- Para a medida *Sens(R)*, as regras 5 e 6 foram as que mais atenderam, tendo em vista que quanto maior a sensitividade, um maior número de exemplos são cobertos pela regra;

- Para a medida *Spec(R)*, as regras 2,3,4,7 e 8, foram as que mais atenderam esta medida, demonstrando a completeza das regras para os exemplos que não são cobertos pelas regras;

- Para a medida *Cov(R)*, as regras que mais atenderam esta medidas foram as regras 5 e 6, tendo em vista o maior o número de exemplos cobertos na regra;

- Para a medida, as regras que mais atenderam esta medida foram as regras 1,2 e 4, tendo em vista de o maior o número de exemplos cobertos pela regra;

- Para a medida *Nov(R)*, esta medida refere-se à novidade das regras, assim, ambos os experimentos atendem a medida, porém com uma interpretação diferenciada, onde as regras 5 e 6, foram as que tiveram o maior valor positivo, ficando mais forte é a associação entre as

regras. Porém a regra de número 3 foi a regra que atendeu a medida de no sentido de quanto mais próximo a regra alcançar o valor negativo, mais forte é a associação entre B e $\bar{H}$;

- A regra que mais atende a medida Sat(R), foi a regra de número 1, tendo em virtude de ter o maior valor alcançado pela regra.

6.9.2 Medidas Relativas

Seguindo a mesma sistemática descrito na seção 6.9.1, ao aplicar as medidas pertinentes à Medidas Relativas, podemos afirmar que para este experimento, as regras que atenderam estas medidas foram:

- A regra que mais atendeu a medida *Racc(R)*, foi a regra de número 5, devido ao seu maior valor encontrado para este experimento;

- De forma análoga, a regra de número 1, obteve o maior valor para este experimento atendendo a medida *RnegRel(R),* que demonstrou que a regra padrão é falso;

- Para a medida Rsens(R), a regra que mais atendeu esta medida, foi a regra 5, pois com este valor, a sua sensitividade para os exemplos que não são cobertos pelas regras.

- Para a medida Rspec(R), a regra que mais atendeu esta medida, foi a regra 5, de forma análoga à medida Spec, porém direcionado para os exemplos que não são cobertos pela regra.

6.9.3 Medidas Relativas de Regras com Peso

Estas medidas trabalham de forma análoga às medidas Confiança negativa, Especificidade Relativa com Peso, dessa forma podemos afirmar que para os exemplos empregados obtivemos o seguinte:

- As regras que mais atenderam a medida WRAcc(R), foram as medidas 5 e 6, que apresentaram um maior valor, representando um alta precisão e peso, obtendo assim uma novidade relativamente alta.

- De forma análoga, as regras que mais atenderam todas as regras relativas com peso, WRAcc(R) , WRNegRel(R),WRSens(R), WRSpec(R), foram as regras 5 e 6, devido ao maior valor encontrado em relação às regras exportas a este experimento.

6.10. Considerações Finais do Capítulo

Neste capítulo foram apresentadas toda a plicação dos algoritmos nas bases de dados, bem como a sua interpretação. Dessa forna, este capítulo obteve uma divisão em duas seções a primeira (seção 6.6), versa sobre todos os dados gerados pelos algoritmos *APriori* e *ZigZag*, de forma a representar uma análise completa do desempenho de cada algoritmo. Este estudo demonstrou que o algoritmo *ZigZag* se mostrou coerente, em decorrência dos dados apresentados se aproximarem dos dados gerados pelo algoritmo *APriori* que é o referencial. Dessa forma, este estudo transcorreu nesta seção de forma quantitativa e não qualitativa, assunto este que será tratado na próxima seção. A segunda seção (6.8) apresentou as medidas de avaliação das regras são extremamente importantes para que se possa focalizar a intenção à regras que melhor se sustentar pelos dados, isto é, dispender a atenção para as regras que possuírem uma maior qualidade.

Entretanto, o fato das regras terem apresentados um bom desempenho, para uma ou várias medidas diferentes, os resultados podem demonstrar um valor óbvio, como foi o caso empregado na seção 6.9.3.

Foram utilizadas regras somente com valor de *suporte* igual a 10%, para aplicação das medidas, no entanto, o objetivo foi alcançado no sentido de avaliação das regras extraídas das bases de dados I e II, aplicados aos algoritmos *APriori* e *ZigZag* propostos.

Capítulo VII

Conclusões e Sugestões para Trabalhos Futuros

A pesquisa empregada neste trabalho compreendeu uma comparação dos algoritmos de regras de associação de forma normal, através do algoritmo *APriori* Bart Goethals(2006) e de forma Incremental através do algoritmo *ZigZag* [Veloso et al. 2002a], indo de encontro às pesquisas na área de Aprendizado de Máquina. Estas pesquisas contribuíram muito para o desenvolvimento de tecnologias direcionadas à Mineração de Dados. Uma das contribuições em relevância abordada neste trabalho, refere-se à comparação dos algoritmos de aprendizado de máquina na extração do conhecimento em grandes volumes de dados aplicados à um cenário real.

O Capítulo 2 deste trabalho, apresentou uma abordagem clássica de Mineração de Dados, bem como uma breve descrição de Aprendizado de Máquina, cujo objetivo é denotar informações pertinentes à Associação através de regras. A tarefa de associação, é uma das mais utilizadas em mineração de dados, pois sua compreensão é de fácil teor, embora o seu complicador seja na definição do que se deseja extrair e dependendo do recurso (algoritmo) utilizado, muitas das vezes torna-se inviável.

Dessa forma, são apresentados no capítulo 3, os algoritmos de regra de associação, compreendido especificamente em dois algoritmos com propósitos iguais, extração de regras de associação, porém de forma diferentes. O primeiro algoritmo, chamado *APriori*, discorre para uma abordagem normal de extração de regras de associação e o segundo, algoritmo denominado *ZigZag,* cujo formato de extração de regras se dá através de um mecanismo incremental.

A possibilidade desta comparação, se deu em decorrência do algoritmo *APriori* servir como base de referência, tendo em vista a sua grande utilização no meio científico e

comercial. Já o algoritmo *ZigZag*, o que se tem conhecimento é aplicação de cunho científico acadêmico, pois não houve a possibilidade de encontrar referências reais de aplicação prática deste algoritmo.

No Capítulo 4 foram discutidas as principais medidas de avaliação de regras de associação. Este assunto é de grande importância para a mineração de dados em geral, pois atende diversas tarefas e aplicações, focando na busca mensurável da qualidade das regras, usualmente focadas para regras de classificação; no caso deste trabalho, focou-se para as regras de associação.

A aplicação das medidas de avaliação foi possível neste trabalho através de duas ferramentas auxiliares. A primeira desenvolvida em linguagem java para gerar a tabela de contingência, considerada o coração do processo das medidas de avaliação. A segunda parte foi efetuada de forma planilhada, onde criou-e uma planilha, com base nos dados extraídos da tabela de contingência, possibilitando assim, um fácil manuseio na aplicação das medidas às regras geradas.

Dessa forma, foi possível visualizar através de tabelas ilustrativas o desempenho das regras, bem como facilitou o manuseio dos valores extraídos para comparação dos resultados obtidos de cada regra.

Dos valores demonstrados, os valores que apresentaram uma diferença significativa, foi constatado ao valor de *suporte* igual a 30% na quantidade geral das regras para cada valor de confiança. Entretanto na quantidade de regras geradas iguais, não houve diferença. O algoritmo que apresentou maior diferença foi o algoritmo *APriori*, este fato ocorreu devido a característica do algoritmo *APriori* efetuar combinações gerais da base de dados. Já o algoritmo *ZigZag*, para gerar as regras, utiliza o mecanismo dos máximos freqüentes, dessa forma estes valores não foram considerados para este algoritmo.

No capítulo 5 foram discutidas as metodologias empregadas, afim de efetuar a análise comparativa dos resultados apresentados pelos algoritmos, pelas medidas aplicadas e pelas comparações empregadas aos resultados. O Capítulo 6 discorreu sobre a aplicação dos algoritmos. É o mais importante, tendo em vista que dele, dissemina-se informações para todos os campos, tanto comparativos, como resultado esperados para uma condição.

Inicialmente foram definidas as bases de dados, cujo teor é igual, porém com informações de períodos distintos, fazendo com que o tamanho de cada uma das bases fosse diferentes. Após definidas as bases de dados, foram definidos os procedimentos e quantidade

de regras as quais se pretende trabalhar, onde sua criação se deu através de valores empregados de suporte e confiança os quais foram: 10%, 30%, 45% e 60%.

Estes valores foram determinados em função das bases serem as mesmas para ambos os algoritmos, o que comprova o baixo desempenho do algoritmo *APriori*, pois ao diminuir estas medidas, o tempo de processamento seria muito grande, comprometendo assim o desenvolvimento deste trabalho.

Com isto, adotou-se uma sistemática de intercalamento entre os valores de *suporte* e *confiança* aplicados entre si, possibilitando para cada base de dados 16 regras, um total de 32 geradas.

Após a extração das regras, foram efetuadas comparações entre os dados extraídos de cada algoritmo empregado e cada uma das base de dados utilizadas. Em geral os resultados foram semelhantes, confirmando o bom desempenho do algoritmo *ZigZag* em relação ao algoritmo *APriori* na extração das regras. Esta informação é muito importante, em virtude do objetivo de comparação ser atendida, uma vez que o algoritmo aqui empregado, é o fator de exemplo para a extração das regras.

De forma análoga, as medidas de avaliação de regras, foram incitadas para que se pudesse fazer a avaliação e apuração direcionadas às regras que mais se destacam, evitando assim, que um especialista tenha que fazer uma análise completa a cada regra do conjunto de regras individualmente, dessa forma que se transcorreu a aplicação das medidas descritas no Capítulo 6.

A partir das análises e comparações efetuadas, a grande contribuição deste trabalho se deu na intenção de medir e demonstrar a qualidade das regras extraídas dos algoritmos propostos, principalmente do algoritmo *ZigZag* em relação ao algoritmo *APriori*. Dessa forma, permite sugerir uma abordagem prática e também comparativa do algoritmo ZigZag com outros algoritmos já consagrados em extração de regras de associação de forma convencional e incremental de dados, para possível validação e utilização em todos os campos, tanto acadêmico como comercial.

Referências

[AGR93a] AGRAWAL, R.; IMIELINSKI, T.; SWAMI, A. *Mining Association Rules beween Sets of Items in Large Databases*. Proceedings of the ACM SIGMOD Conference. Washington, USA, May 1993.

[AGR93b] AGRAWAL, R.; IMIELINSKI, T.; SWAMI, A. *Database Mining: A desempenho perspective*. IEEE Trans. On Knowledge and Data Engineering. December, 1993.

[AGR94] AGRAWAL, R.; SRIKANT, R. *Fast Algorithms for Mining Association Rules*. Proc. Int´l Conf. Very Large Data Bases, pp. 487-499, Santiago, Chile, Sept. 1994.

[AGR96] AGRAWAL, R.; H. MANNILA; R. SRIKANT; H. TOIOVONEN, and VERKAMOA.I. Fast discovery of association rules. In U.M. Fayyad, G. Piatetsky- Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in KnowledgeDiscovery and Data Mining, pages 307–328. MIT Press, 1996.*

[BAR00] BARANAUSKAS, J. A; MONARD, M. C. *Reviwing some machine learning concepts and methods. Techical Report* 102, ICMC-USP.
Ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_102.ps.zip.

[BIG96] BIGUES, J.P. - *Data mining with Neural Networks*, 1996. McGrawHill.

[BRU00] BRUSSO, M. J. - *Access Miner*: uma proposta para a extração de regras de associação aplicada à mineração do uso da Web. Dissertação de Mestrado, Instituto de Informática, UFRGS, 2000.

[BAT94] BATISTTI, R. *Using Mutual Information for Selecting Features in Supervised Neural Net Learning.* IEEE Transactions on Neural Networks, Vol.5, no.4,1994, p.537-550.

[BAY98] BAYARDO, R. J. *Efficiently mining long patterns from databases*. In ACM SIGMOD Conference of Management of Data, 1998.

[BER93] BERCU, S.; LORETTE, G. *On-line handwritten word recognition: an approach based on hidden markov models.* IWFHR-3, 1993, p.385-390.

[BOZ89] BOZINOVIC, R.M. & SRIHARI, S.N. *Off-line cursive script recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence*. Vol.11, No. 1, 1989, p.68-83.

[BUR83a] BURR, D.J. *Designing a handwriting reader. IEEE Transactions on Pattern Analysis and Machine Intelligence,* PAMI-5(5). 1983, p.554-559.

[BUR83b] BURR, D.J. *Designing another handwriting reader. ACM Journal on Computer Technology***,** 4(2). 1983, p.324-329.

[CAM] CAMARGO, S. S. - Mineração de Regras de Associação no Problema de Cesta de Compras aplicada ao Comércio Varejista de Confecção. UFRGS, Instituto de Informática, VI Semana Acadêmica do PPGC, 2000. Disponível em: www.inf.ufrgs.br/pos/SemanaAcademica/Semana2000/SandroCamargo.

[CAV95] CAVALCANTI, F. T. *Incremental Mining Techniques, Universidade do Minho, Portugal*, 2005

[CHN96] CHEUNG, D; HAN, J; V. Ng, And C. Y. Wong: "*Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique". In Proceedings of the 12th International Conference On Data Engineering, February* 1996.

[CLA89] CLARK, P. And NIBLETT, T. . *The CN2 induction algorithm. Machine Learning* 3(4):261-283, 1989.

[DOM98] DOMINGO, C.; GAVALDÀ, R.; WATANABE, º – *On-line sampling Methods for Discovering Association Rules*. Universitat Politècnica de Catalunya, Departament de Llenguatges i Sistemes Informàtics, Report Number: LSI-99-4-R, 1998.

[FAY96a] FAYYAD, U., G. PIATETSKY-SHAPIRO & P. SMYTH. *From data mining to knowledge discovery: and overview. In Advances in Knowledge Discovery in Databases & Data Mining, pp. 1-34, 1996a.*

[GOL89] GOLDBERG, D. E. *Genetic Algoritms in Search, Optimization an Machine Learning*. Addison Wesley, Canadá, 1989.

[GOM02] GOMES A . K. Análise do Conhecimento Extraído de Classificadores Simbólicos Utilizando Medidas de Avaliação e de Interessabilidade. Dissertação de Mestrado, ICMC-USP. São Paulo – Brasil, 2002.

[GOU01] GOUDA, K.; ZAKI, M. - *Efficiently mining maximal frequent itemsets. In Proc. of the 1st IEEE Int'l Conference on Management of Data, 2001.*

[GRO98] GROTH, R. Data Mining: a hands-on approach for business *professionals. ISBN 0-13-756412-0. New Jersey, Prentice Hall,* 1998

[HAN00] HAN, MAO, R. - *Closet: An efficient algorithm for mining frequent closed itemsets.* In SIGMOD Int'l Workshop on Data Mining and Knowledge Discovery, 2000.

[HIP00] HIPP, J.; GUNTZER, U. and NAKHAEIZADEH, G.*, "Agorithms forAssociation Rule Mining – A General Survey and Comparason"*. SIGKDD2000.

[HIP02] HIPP, J.; GUNTZER, U.*, "Is pushing constraints deeply into the mining algorithms really what we want?: an alternativa appoach for association rule mining"*. SIGKDD Exploration. Vol. 4-1, June 2002.

[HOR99] HORST, P. S. Avaliação do conhecimento adquirido por algoritmos de aprendizado de máquina utilizando exemplos. Dissertação de Mestrado, ICMC-USP. 1999.

[HOR00] HOSRT, P. S. And MONARD, M. Um sistema computacional para avaliação de regras induzidas por algoritmos de aprendizado de máquina. *In Proceedings IBERAMIA-SBIA 2000 Open Discussion Track, pages* 167-176, Atibaia, SP, Brasil. 2000.

[JOR03] JORGE, A. M. – Material de Apoio do Mestrado em Análise de Dados e Sistema de Apoio à Decisão. Faculdade de Economia, Universidade do Porto. Disponível em: www.niaad.liacc.up.pt/~amjorge/Aulas/madsad/ecd2 . Acessado em 15/07/2007.

[LVR99] LAVRAC, P.; FLACH and ZUPAN, B. *Rules evaluation measures: a unifying view. In Proceedings of the Ninth International Workshop on Inductive Logic Programing.* LNAI, volume 1634, pages 74-185, 1999.

[LAV03] LAVÔR, R. M. P.. Implementação de serviços relacionados à regras de associação. XVIII, 152 p. 29 cm (NCE/IM/UFRJ, M.Sc. Informática, 2003).

[LEV99] LEVY, E. - *The Lowndown on Data Mining*. Teradatareview, Summer, 1999.

[LIU99] LIU, B.; HSU, W.; MA, Y. - *Mining Association Rules with Multiple MinimumSupports*. Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-99), pp. 337-341, 1999.

[LM01] LIU, B. And MA, Y. *"Analyzing the Interestingness of Association Rules from the Temporal Dimension". In 1st IEEE International Conference on Data Mining, November,* 2001.

[MCC43] McCULLOCH, W. S.; PITTS, W. *A logical calculus of the ideas immenet in nervous activity*. In Boulletin of Mathematical Biophisics, volume 5, page 115-133, 1943.

[MON03] MONARD, M. C. & J. A. BARANAUSKAS. Conceitos sobre aprendizado de máquina. In S. O. Rezende (Ed.), *Sistemas Inteligentes: Fundamentos e Aplicacões*, pp. 89–114. Manole. 2003.

[MOR73] J. MORGAN, R. MESSENGER.: *A sequencial search program for the analysis of nominal scale dependent variables. Technical report, Institute for Social Research, University of Michigan. Tecnical Report.* 1973.

[PAR97] PARK, J.S.; CHEN, M.-S. - *Using a Hash-Based Method with Transaction Trimming for Mining Association Rules*. IEEE transactions on Knowledge and Data Engineering, vol. 9, No 5, 1997.

[PIA91] PIATETSKY-SHAPIRO, G. *Discovery, analysis and presentation of strong rules.* In Piatetsky-Shapiro, G. and Frawley, W. J., editor, Knowledge Discovery in Databases, pages 229-248. 1991.

[PRT99] PARTHASARATHY, S.; ZAKI, M.; OGIHARA, M.; DWARKADAS, S. *Incremental and Interactive Sequence Mining, In 8th International Conference on Information and Knowledge Management,* November 1999.

[QUI93] QUINLAN, J. R. C4.5: Programs for Machine Learning. Morgan Kaufmann Publisher, Los Altos, California, USA. 1993.

[RUS95] RUSSEL, S. And NORVIG, P. *Artificial Intelligence: A Modern Approach. Prenticeall.* 1995.

[SAV95] SAVASERE, A.; OMIELINSKI, E.: NVATHE, S. - *An Efficient Algorithm for Mining Association Rules in Large Databases*. In Very Large Databases, 1995.

[SIE03] SIEBES, A. - *Association Rules. Institute of Information and Computing Sciences, Department of Mathematics and Computer Science, Universiteit Utrecht*. 2003.

[SOU00] SOUZA, O. R.M. Mineração de Dados de um Plano de Saúde para obter Regras de Associação. Dissertação de Mestrado PPEP- UFSC- Florianópolis - Brasil, 2000.

[SRI97] SRIKANT, R.; VU, Q.; AGRAWAL, R. - *Mining Association Rules with Item Constraints. IBM Almaden Research Center & American Association for Artificial Intelligence*, 1997.

[TAN00] P-N., Kumar V. (2000), *"Interestigness Measures for Association Patterns: A Perspective"*,. KDD 2000 *Workshop on Postprocessing in Machine Learning and Data Mining.*

[TOI96] TOINIVEN, H. - Sampling Large Databases for Association rules. Proceedings of the 22th VLDB Conference, 1996.

[TUR01] TURBAN, E. A. ARONSON; JAY, E., *Decision Suport Systems and Intelligent System*. New Jersey: Prentice Hall. 2001.

[VEL02] VELOSO, A.; MEIRA Jr, W.; BUNTE, M.; PARTHASARATHIY, And M. ZAKI: *"Mining frequent itemsets in evolving databases"***.** *In Proceedings of the 2nd SIAM International Conference on Data Mining, USA, 2002.*

[VEL03] VELOSO, M.; SILVA J. P. A. "Regras de associação aplicadas a um método de apoio ao planejamento de recursos humanos" - Porto : [s.n.], 2003. - VII, 150 f. : il. ; 30 cm http://purl.pt/6322.

[WON02] WONG, L. *Datamining: Discovering Information from Bio-data. Current Topics in Computational Biology*, edited by Tao Jiang, Ying Xu, and Michael Zhang, 2002, c. 13, p. 317-342, MIT Press, Cambridge, MA.

[ZAK00] ZAKI, M. J. - Parallel sequence mining on SMP machine. In Zaki and Ho, 2000.