



**Universidade do Estado do Rio de Janeiro**  
Centro de Tecnologia e Ciência  
Faculdade de Engenharia

Ricardo Ferreira Vieira de Castro

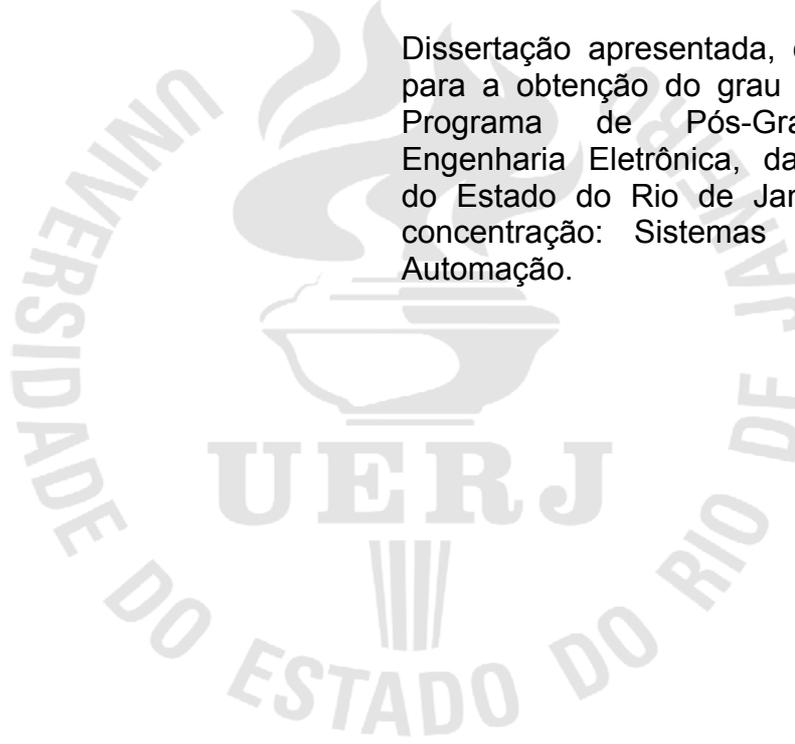
**Análise de desempenho dos algoritmos Apriori e *Fuzzy*  
Apriori na extração de regras de associação aplicados a  
um Sistema de Detecção de Intrusos**

Rio de Janeiro  
2014

Ricardo Ferreira Vieira de Castro

**Análise de desempenho dos algoritmos Apriori e *Fuzzy* Apriori na extração de regras de associação aplicado a um Sistema de Detecção de Intrusos**

Dissertação apresentada, como requisito para a obtenção do grau de Mestre, ao Programa de Pós-Graduação em Engenharia Eletrônica, da Universidade do Estado do Rio de Janeiro. Área de concentração: Sistemas Inteligentes e Automação.



Orientadora: Prof<sup>a</sup>. Dr<sup>a</sup>. Maria Luiza Fernandes Velloso  
Coorientador: Prof. Dr. Nival Nunes de Almeida

Rio de Janeiro  
2014

CATALOGAÇÃO NA FONTE  
UERJ/REDE SIRIUS/BIBLIOTECA CTC/G

S586

Castro, Ricardo Ferreira Vieira de.

Anáse de desempenho dos algoritmos Apriori e *Fuzzy* Apriori na extração de regras de associação aplicado a um Sistema de Detecção de Intrusos. Ricardo Ferreira Vieira de Castro. – 2014.

100 f.

Orientadora : Prof<sup>a</sup>. Dr<sup>a</sup> Maria Luiza Fernandes Velloso

Co-orientador: Prof. Dr. Nival Nunes de Almeida

Dissertação (Mestrado) – Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia.

Bibliografia.

1. Engenharia Eletrônica. 2. *Extração de regras Fuzzy* - Dissertações. I. Velloso, Maria Luiza Fernandes II. Universidade do Estado do Rio de Janeiro. III. Título.

CDU 004.032.6:37

Autorizo, apenas para fins acadêmicos e científicos, a reprodução total ou parcial desta dissertação, desde que citada a fonte.

---

Assinatura

---

Data

Ricardo Ferreira Vieira de Castro

**Análise de desempenho dos algoritmos Apriori e *Fuzzy Apriori* na extração de regras de associação aplicado a um Sistema de Detecção de Intrusos**

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Engenharia Eletrônica, da Universidade do Estado do Rio de Janeiro. Área de concentração: Sistemas Inteligentes e Automação.

Aprovado em 20 de fevereiro de 2014.  
Banca Examinadora:

---

Prof<sup>a</sup> Dra. Maria Luiza Fernandes Velloso (Orientadora)  
Faculdade de Engenharia, UERJ

---

Prof. Dr. Nival Nunes de Almeida (Coorientador)  
Faculdade de Engenharia, UERJ

---

Prof. Dr. Orlando Bernardo Filho  
Faculdade de Engenharia, UERJ

---

Profa Dra. Karla Tereza Figueiredo Leite  
Pontifícia Universidade Católica do Rio de Janeiro

Rio de Janeiro

2014

## DEDICATÓRIA

Aos meus pais, saudades eternas, por todo amor e dedicação que me dispensaram ao longo de suas vidas, e onde sempre me orientaram a ter fé e nunca desistir dos meus sonhos por mais distante que estejam, e a minha esposa Rebecca Galera.

## **AGRADECIMENTOS**

A todos aqueles que de alguma forma me incentivaram, ajudaram e compartilharam das minhas dificuldades, das minhas ausências, dos meus momentos mau-humorados, dos momentos de alegria por ter galgado mais um degrau no trabalho, e por todos os sentimentos e experiências por mim vividas. Entre todos, não poderia deixar de tecer agradecimentos especiais por aqueles que diretamente contribuíram por mais esta etapa em minha vida.

A minha orientadora, professora Dra. Maria Luiza Fernandes Velloso, pelo apoio, paciência e conhecimentos transmitidos ao longo do curso.

Ao meu coorientador, professor Dr. Nival Nunes de Almeida por toda ajuda, atenção e todo apoio dispensado.

À Universidade do Estado do Rio de Janeiro - UERJ por ter me recebido e apoiado ao longo do curso.

Ao Instituto Nacional de Tecnologia, através da Comissão de Pós-Graduação, pela concessão de participação no Programa de Pós-Graduação Stricto-Sensu da UERJ.

A Dra. Maria Cristina Zamberland, do Instituto Nacional de Tecnologia, por ser uma fiel incentivadora dos meus estudos.

Ao amigo Me. Joaquim Augusto, da Divisão de Energia do Instituto Nacional de Tecnologia, que me trilhou e incentivou aos estudos na UERJ.

## RESUMO

CASTRO, Ricardo Ferreira Vieira. Análise de desempenho dos algoritmos Apriori e *Fuzzy Apriori* na extração de regras de associação aplicado a um Sistema de Detecção de Intrusos. 2014. 141f. Dissertação (Mestrado em Engenharia Eletrônica) - Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2014.

A extração de regras de associação (ARM - *Association Rule Mining*) de dados quantitativos tem sido pesquisa de grande interesse na área de *data mining*. Existe ainda uma busca permanente de algoritmos para melhorar o desempenho relacionado a quantidade de regras, sua relevância e a performance computacional. O algoritmo APRIORI, muito encontrado na literatura científica para extração de regras de associação, transforma os atributos com valores quantitativos em partições de valores binários. No entanto, esse método proposto remete ao problema conhecido como *sharp boundaries* que consiste em poder subestimar ou superestimar elementos próximos dos limites da partição, e portanto levar a uma representação imprecisa de semântica. Para tratar o problema, o algoritmo *FUZZY APRIORI*, baseando-se na teoria dos conjuntos fuzzy, transforma os atributos com valores quantitativos em partições de termos linguísticos (*FARM - Fuzzy Association Rule Mining*). As regras extraídas são expressas em termos linguísticos, que são mais natural e interpretável pelo raciocínio humano.

Nesta dissertação, temos a proposta de analisar o desempenho dos dois algoritmos numa base de registros de conexões TCP/IP de um Sistema de Detecção de Intruso, e através de algoritmos de classificação associativa avaliar a precisão das regras extraídas.

Palavras-chave: Apriori. *Fuzzy Apriori*. Extração de regras de associação. Regras de classificação associativa. Detecção de intrusos.

## ABSTRACT

The mining of association rules ( ARM ) of quantitative data has been of great research interest in the area of data mining . There is an ongoing search algorithms to improve performance related to the amount of rules, its relevance and computational performance. The APRIORI algorithm, often found scientific literature for association rule mining, transforms attributes with quantitative values in partitions with binary values. However, this proposed method leads to the problem known as sharp boundaries that consist of underestimate or overestimate near the boundaries of the partition elements and therefore lead to an inaccurate representation of the semantics. To resolve the problem , the *FUZZY* APRIORI algorithm, based on fuzzy set theory, transforms attributes with quantitative values in linguistic terms partitions (*FARM*).

In this work , we proposed to compare the performance of both algorithms apply on records of TCP / IP connections of an Intrusion Detection System, and through associative classification algorithms to evaluate the accuracy of the rules mining.

Keywords: Apriori. Fuzzy Apriori. Fuzzy Association Rule Mining. Associative classification rule. Intruder Detection.

## LISTA DE ILUSTRAÇÕES

Figura 1 - Sistema de Detecção de Intrusão por uso indevido. ....	27
Figura 2 - Probabilidade de Detecção entre Atividades Normais e Anormais. ....	28
Figura 3 - Sistema de Detecção de Intrusão por anomalia. ....	29
Figura 4 - Fases operacionais do processo de KDD .....	29
Figura 5 – Etapas do processo de KDD .....	30
Figura 6 – Análise de cesta de mercado ajuda a entender consumidores. ....	36
Figura 7 – Precisão e significância no mundo real. ....	39
Figura 8 - Exemplo de mapeamento de espaço de entrada e saída. ....	39
Figura 9 - Roteiro básico de inferência <i>fuzzy</i> . ....	40
Figura 10. Representação básica de conjunto <i>crisp</i> e <i>fuzzy</i> . ....	42
Figura 11 – Conjunto de altura das pessoas. ....	42
Figura 12 – Conjunto <i>crisp</i> de altura das pessoas. ....	42
Figura 13 – Conjunto <i>fuzzy</i> de altura das pessoas. ....	43
Figura 14 - Variáveis linguísticas dos conjuntos de altura das pessoas. ....	44
Figura 15 - Funções de pertinência para a variável altura. ....	45
Figura 16 - Funções de pertinências de linha reta. ....	47
Figura 17 - Funções de pertinências <i>Gaussianas</i> . ....	47
Figura 18 - Funções de pertinências <i>Sigmoidal</i> . ....	48
Figura 19 - Funções de pertinências <i>Polinomial</i> . ....	48
Figura 20 - Regras <i>fuzzy</i> com dois antecedentes. ....	50
Figura 21 - Sistema de Inferência <i>Fuzzy</i> .....	51
Figura 22 - Diagrama do problema da gorjeta. ....	53
Figura 23 - Diagrama de inferência <i>fuzz</i> . ....	53
Figura 24 - Detalhamento do processo de inferência <i>fuzzy</i> . ....	53
Figura 25 – Transações de compra no supermercado. ....	55
Figura 26 - Fluxograma do algoritmo Apriori. ....	58
Figura 27 - Pseudocódigo principal do algoritmo Apriori. ....	59
Figura 28 - Pseudocódigo da função <i>apriori-gen</i> do algoritmo Apriori. ....	60
Figura 29 – Rede de conjunto de itens. ....	61
Figura 30 – Princípio Apriori aplicado na rede de itemsets .....	61
Figura 31 - Pseudocódigo da função <i>genrules</i> do algoritmo Apriori. ....	62

Figura 32 - Exemplo de aplicação do Algoritmo Apriori.....	64
Figura 33 – Exemplo de extração de regra quantitativa - <i>Crisp</i> .....	67
Figura 34 – Fluxograma Algoritmo Fuzzy Apriori Clássico .....	70
Figura 35 – Fluxograma Algoritmo Fuzzy Apriori-A.....	71
Figura 36 – Função de pertinência usada no exemplo.....	72
Figura 37 - Modelo básico dos registros de conexões. ....	87
Figura 38 - Matriz conjunto de dados 10%KDD x 10%KDD/DoS.....	106
Figura 39 - Comparação dos métodos de discretização supervisionado e não supervisionado.....	111
Figura 40 - Relação entre o número de regras de associação e valor mínimo de suporte para vários valores de confiança mínimo.....	116
Figura 41 - Relação entre o número de regras de classificação e valor mínimo de suporte para vários valores de confiança mínimo.....	117
Figura 42 - Relação entre o percentual de precisão alcançado e valor mínimo de suporte para vários valores de confiança mínimo.....	118
Figura 43 - Matriz confusão suporte 30% e confiança 60%. ....	119
Figura 44 - Matriz confusão suporte 80% e confiança 90%. ....	119
Figura 43 - Relação entre o número de regras de associação e valor mínimo de suporte para vários valores de confiança mínimo.....	120
Figura 44 - Relação entre o número de regras de classificação e valor mínimo de suporte para vários valores de confiança mínimo.....	121
Figura 45 - Relação entre o percentual de precisão alcançado e valor mínimo de suporte para vários valores de confiança mínimo.....	122
Figura 46 - Relação entre o número de regras de associação e valor mínimo de suporte para valor de confiança mínimo de 40%.....	124
Figura 47 - Relação entre o número de regras de classificação e valor mínimo de suporte para valor de confiança mínimo de 40% e 90%.....	125
Figura 48 - Relação entre o percentual de precisão alcançado e valor mínimo de suporte para valor de confiança mínimo de 40% e 90%.....	126

## LISTA DE GRÁFICOS

Gráfico 1 - Distribuição das características básicas do conjunto de dados KDD99 por tipo de categoria de ataque .....	92
Gráfico 2 - Comparação da distribuição da amostragem dos tipos de ataques DoS (10%KDD/DoS x WANG).....	97
Gráfico 3 - Distribuição das amostras dos tipos de ataques do conjunto de dados de treinamento KDD99. ....	102

## LISTA DE TABELAS

Tabela 1 – Transações de compras armazenadas no banco de dados. ....	55
Tabela 2 – Associações de regras x suporte e confiança. ....	55
Tabela 3 – Base de dados $\mathcal{D}$ e respectivos <i>itemsets</i> . ....	63
Tabela 4 – Ocorrência e suporte dos atributos.....	63
Tabela 5 – Exemplo do conjunto de dados. ....	72
Tabela 6 – Conjuntos <i>fuzzy</i> transformado à partir da Tabela 5.....	73
Tabela 7 – Conjuntos temporário $C_1$ .....	73
Tabela 8 – Conjuntos de 1-itemsets com seus valores <i>count</i> . ....	74
Tabela 9 – Conjuntos de 1-itemsets $L_1$ .....	75
Tabela 10 – O valor de pertinência para $(ST\_Middle, DB\_High)$ .....	76
Tabela 11 – Valor de pertinência dos demais 2-itemset.....	76
Tabela 12 – Conjunto temporário $C_2$ . ....	78
Tabela 13 – Counts dos itemsets <i>fuzzy</i> em $C_2$ . ....	78
Tabela 14 – <i>Itemset</i> e seus respectivos <i>count</i> em $L_2$ . ....	79
Tabela 15 – <i>Itemset</i> e seus respectivos <i>count</i> em $L_2$ . ....	79
Tabela 16 – Atributos das conexões de rede. ....	87
Tabela 17 – Categoria dos atributos - básicas do TCP/IP.....	88
Tabela 18 – Categoria dos atributos - por conhecimento especialista. ....	89
Tabela 19 – Categoria dos atributos - teoria da janela de dois segundos.....	89
Tabela 20 - Características básicas da base de dados KDD99 em termos de número de amostras e percentual por tipo de categoria de ataque.....	92
Tabela 21 - Tipos de ataque, quantidade de amostras e classe da categoria da base de dados de treinamento (10% KDD). ....	92
Tabela 22 - Tipos de ataque, quantidade de amostras e classe da categoria da base de dados de teste ( <i>Corrected</i> ). ....	93
Tabela 23 - Tipos de ataque, quantidade de amostras e classe da categoria da base de dados original (Whole KDD). ....	94
Tabela 24 - Conjunto de dados 10%KDD x 10%KDD/DoS.....	96
Tabela 25 - Conjunto de dados 10%KDD/DoS.....	96
Tabela 26 - Reamostragem segundo WANG. ....	97
Tabela 27 - Seleção de atributos relevantes da base de dados KDD99. ....	100

Tabela 28 - Seleção de atributos relevantes em base de dados KDD99 .....	103
Tabela 29 - Avaliação da efetividade dos 10 atributos selecionados. ....	104
Tabela 30 – Distribuição dos valores dos atributos em 10%KDD. ....	104
Tabela 31 – Distribuição dos valores dos atributos em 10%KDD/DoS. ....	105
Tabela 32 - Ferramentas e tarefas utilizadas no desenvolvimento. ....	107
Tabela 33 - Arquivos gerados para serem filtrados. ....	108
Tabela 34 - Atributos relevantes selecionados em 10%KDD_DoS. ....	109
Tabela 35 - Dados comparativos dos Algoritmos Apriori e Fuzzy Apriori para valores de confiança mínimo de 40%. ....	123
Tabela 36 - Dados comparativos dos Algoritmos Apriori e Fuzzy Apriori para valores de confiança mínimo de 90%. ....	124

## LISTA DE SIGLAS E ABREVIATURAS

AI - Artificial Intelligence

AMD - Advanced Micro Devices

ARFF - Attribute-Relation File Format

ARM - Association Rule Mining

CBA - Classification Based on Association

CFA -R - Classifier Fuzzy Association Rule

CFAR - Classification with Fuzzy Association Rules

CSV - Comma Separated Values

FARM - *Fuzzy* Association Rule Mining

FPR - False Positive Rate

GB - Giga Byte

GUI - Graphical User Interface

HIDS - Host Intrusion Detection System

IDS - Intruder Detection System

KDD - Knowledge Discovery in Databases

KEEL - Knowledge Extraction based on Evolutionary Learning

NIDS - Network Intrusion Detection System

RAM - Random Access Memory Random

SBP - Sharp Boundary Problem

UCI - University of California, Irvine

WEKA - Waikato Environment for Knowledge Analysis

## SUMÁRIO

	<b>INTRODUÇÃO</b> .....	16
<b>1</b>	<b>PROPOSTA DE TRABALHO</b> .....	16
<b>1.1</b>	<b>Motivação</b> .....	17
<b>1.2</b>	<b>Objetivos</b> .....	20
<b>1.3</b>	<b>Estrutura da dissertação</b> .....	21
<b>2</b>	<b>FUNDAMENTAÇÃO TEÓRICA</b> .....	22
<b>2.1</b>	<b>Antecedentes e Estado da arte</b> .....	22
<b>2.2</b>	<b>Sistema de Detecção de Intrusos</b> .....	23
2.2.1	<u>Introdução</u> .....	23
2.2.2	<u>Arquitetura dos SDIs</u> .....	25
2.2.3	<u>Detecção por uso indevido</u> .....	26
2.2.4	<u>Detecção por anomalia</u> .....	28
2.2.5	<u>As etapas do processo de KDD</u> .....	29
2.2.6	<u>Mineração de Dados (Data Mining)</u> .....	33
2.2.7	<u>Regras de associação na Mineração de Dados</u> .....	36
<b>2.3</b>	<b>Sistemas <i>Fuzzy</i></b> .....	38
2.3.1	<u>Conjuntos <i>Fuzzy</i></u> .....	40
2.3.2	<u>Variável linguística</u> .....	43
2.3.3	<u>Função de pertinência</u> .....	45
2.3.4	<u>Regras <i>Fuzzy</i> SE - ENTÃO</u> .....	49
2.3.5	<u>Sistema de Inferência <i>Fuzzy</i></u> .....	50
<b>2.4</b>	<b>Extração de Regras de Associação (Qualitativo)</b> .....	54
2.4.1	<u>O Algoritmo APRIORI</u> .....	57
2.4.1.1	Geração e poda de candidatos e cálculo do suporte.....	58
2.4.1.2	Geração de regras de associação .....	61
2.4.2	<u>Exemplo de aplicação - Qualitativo / Booleanos</u> .....	62
<b>2.5</b>	<b>Extração de Regras de Associação (Quantitativo - <i>CRIPS</i>)</b> .....	65
2.5.1	<u>Mapeando atributos quantitativos em Booleanos</u> .....	65
2.5.2	<u>Exemplo de aplicação - Quantitativo / Booleano</u> .....	66
2.5.3	<u>Classificação associativa - Algoritmo CBA</u> .....	67
<b>2.6</b>	<b>Extração de Regras de Associação (Quantitativo - <i>FUZZY</i>)</b> .....	68

2.6.1	<u>Algoritmo Fuzzy Apriori</u> .....	69
2.6.2	<u>Exemplo de aplicação - Quantitativo / Fuzzy</u> .....	72
2.6.3	<u>Classificação associativa - Algoritmo CFAR</u> .....	81
<b>3</b>	<b>EXPERIMENTOS</b> .....	<b>82</b>
<b>3.1</b>	<b>Ferramentas de mineração de dados</b> .....	<b>82</b>
3.1.1	<u>WEKA (Waikato Environment for Knowledge Analysis)</u> .....	83
3.1.2	<u>KEEL (Knowledge Extraction based on Evolutionary Learning)</u> .....	84
<b>3.2</b>	<b>Descrição dos dados</b> .....	<b>86</b>
3.2.1	<u>Características da base de dados KDD99</u> .....	86
3.2.2	<u>Distribuição dos registro de conexões e ataques</u> .....	91
3.2.3	<u>Reamostragem do conjunto de dados</u> .....	95
3.2.4	<u>Seleção de atributos relevantes do conjunto de dados</u> .....	97
3.2.4.1	Trabalhos selecionados para seleção de atributos num IDS .....	99
3.2.4.2	Atributos relevantes selecionados para o trabalho .....	101
3.2.5	<u>Conjunto de dados KDD99/DoS pré-processado</u> .....	105
<b>3.3</b>	<b>Metodologia</b> .....	<b>106</b>
3.3.1	<u>Pré-processamento</u> .....	107
3.3.1.1	Reamostragem .....	108
3.3.1.2	Seleção de atributos .....	109
3.3.2	<u>Aplicação do Algoritmo APRIORI</u> .....	110
3.3.2.1	Transformação dos dados - Discretização .....	110
3.3.2.2	Mineração dos Dados - Extração de regras de associação.....	111
3.3.2.3	Interpretação e Avaliação - Classificador de regras .....	112
3.3.3	<u>Aplicação do Algoritmo FUZZY APRIORI</u> .....	113
3.3.3.1	Mineração dos Dados - Extração de regras de associação.....	113
3.3.3.2	Interpretação e Avaliação - Classificador de regras .....	114
<b>4</b>	<b>RESULTADOS e ANÁLISE DOS EXPERIMENTOS</b> .....	<b>115</b>
<b>4.1</b>	<b>Introdução</b> .....	<b>115</b>
<b>4.2</b>	<b>Aplicação do Algoritmo Apriori</b> .....	<b>115</b>
4.2.1	<u>Quantidade de regras de associação extraídas</u> .....	116
4.2.2	<u>Conjunto de regras de classificação</u> .....	117
4.2.3	<u>Precisão das regras de associação</u> .....	118
<b>4.3</b>	<b>Aplicação do Algoritmo Fuzzy Apriori</b> .....	<b>120</b>
4.3.1	<u>Quantidade de regras de associação extraídas</u> .....	120

4.3.2	<u>Conjunto de regras de classificação</u> .....	121
4.3.3	<u>Precisão das regras de associação</u> .....	122
<b>4.4</b>	<b>Algoritmo Apriori x Fuzzy Apriori</b> .....	<b>123</b>
4.4.1	<u>Análise comparativa de quantidade de regras de associação</u> .....	124
4.4.2	<u>Análise comparativa de quantidade de regras de classificação</u> .....	125
4.4.3	<u>Análise comparativa de precisão das regras de associação</u> .....	125
	<b>CONCLUSÃO E FUTUROS TRABALHOS</b> .....	<b>127</b>
	<b>REFERÊNCIAS</b> .....	<b>131</b>

## INTRODUÇÃO

*"O saber agente aprende com os mestres e com os livros.  
A sabedoria se aprende é com a vida e com os humildes"  
Cora Coralina*

Este trabalho realiza uma análise de desempenho entre algoritmos para extração de regras de associação aplicados em base de dados que possuem atributos de valores quantitativos, através de uma avaliação da quantidade de regras de associação extraídas e pela respectiva precisão de acerto dessas regras.

### 1 PROPOSTA DE TRABALHO

A extração de regras de associação de uma base de dados, conhecida como *Association Rule Mining (ARM)*, é normalmente aplicada para o problema de descoberta de conjuntos de itens freqüentes e posterior extração de regras de associação destes conjuntos. Entretanto, uma base de dados, em muitos dos casos, possui um conjunto de atributos que podem ser quantitativos<sup>1</sup> ou qualitativos<sup>2</sup> e. Por exemplo, uma das técnicas que são aplicadas para extração de regras de associação nas quais possuem atributos quantitativos utiliza o conceito *FUZZY*, mais conhecida como *Fuzzy Association Rule Mining (FARM)*. A base de dados utilizada neste trabalho, denominada de *KDD Cup 1999 Data*, foi empregada na Terceira Competição Internacional de descoberta de conhecimento de mineração de dados. Esta base é composta por um padrão de dados a serem auditados que inclui uma grande variedade de intrusões simuladas em um ambiente de rede militar. Inicialmente, esses dados inicialmente passaram por algumas etapas de pré-processamento até o seu resultado final, tais como: seleção de atributos relevantes reamostragem dos dados.

---

<sup>1</sup> Variáveis quantitativas: são as características que podem ser medidas em uma escala quantitativa, ou seja, apresentam valores numéricos que fazem sentido. Podem ser contínuas ou discretas.

<sup>2</sup> Variáveis qualitativas (ou categóricas): são as características que não possuem valores quantitativos, mas, ao contrário, são definidas por várias categorias, ou seja, representam uma classificação dos indivíduos. Podem ser nominais ou ordinais.

A abordagem proposta neste trabalho é apresentar a comparação dos algoritmos Apriori e *Fuzzy Aprior*, aplicados a extração de regras de associação, no contexto de atributos que possuem valores quantitativos, onde basicamente no primeiro os valores quantitativos são particionados em valores binários, e no segundo, utilizando conceitos fuzzy, particionados em termos de pertinência e suas respectivas funções de pertinência.

Essa abordagem ajudará a fazer uma análise dos algoritmos Apriori e *Fuzzy Apriori*, onde através da classificação das regras de associação geradas, será possível avaliar em que condições qual o algoritmo que apresenta melhor precisão de acerto no reconhecimento de ataques de um sistema de detecção de intrusos.

## 1.1 Motivação

As últimas duas décadas vieram acompanhadas de um aumento exponencial na quantidade de informações que são armazenadas em Data Center. A Internet e seus serviços de e-commerce ampliaram o acúmulo de dados que acontecem a uma taxa surpreendente. Bancos de Dados estão cada vez mais volumosos de um modo que nossa capacidade de analisar e compreender essa massa de dados está bem aquém de nossa habilidade de coletar e armazenar dados. O valor destes dados armazenados está tipicamente ligado à capacidade de extrair informações de mais alto nível que se encontra subjacente a estes dados, ou seja, informação útil que sirva para dar suporte a decisões, e para exploração e melhor entendimento do fenômeno gerador dos dados. Podem existir padrões ou tendências úteis interessantes que, se descobertos, podem ser utilizados, por exemplo, para otimizar um processo de negócio em uma empresa, ajudar no entendimento dos resultados de um experimento científico, ajudar médicos a entender efeitos de um tratamento entre outros. Neste contexto está situado o KDD (*Knowledge Discovery in Databases*) (FAYY, 1996), que busca extrair de grandes bases de dados as informações que não podem ser facilmente encontradas.

Inserido neste processo de descoberta, existe a etapa de *Data Mining* (Mineração de Dados) onde são aplicadas técnicas para descoberta de novos conhecimentos. Juntamente com as etapas de Seleção, Processamento,

Transformação e Interpretação, a Mineração de Dados completa as fases do processo de KDD. Diversas definições de Mineração de Dados podem ser encontradas na literatura. Entre elas podemos destacar que: *Mineração de dados, de forma simples, é o processo de extração ou mineração de conhecimento em grandes quantidades de dados* (HAN, KAMBER, 2001).

O processo de mineração pode ser utilizado em diversos tipos de banco de dados, e em função deste se pode definir que tipo de padrões ou relacionamentos se deseja minerar. A aplicação ou funcionalidade da mineração de dados vai definir o tipo de relacionamento entre as instâncias e seus atributos utilizados na mineração. Dentro do processo de mineração essa funcionalidade é conhecida por alguns autores, como resultados (*outcomes*) ou tarefas (*tasks*) (CORTEZ, PORCARO, LIFSCHITZ, 2002). Diversos autores tratam essas tarefas de formas diferente, mas as mais regularmente utilizadas são: Classificação, Regressão, Clusterização e Associação que é a tarefa tema desta dissertação.

Regras de associação são usadas para descobrir elementos que ocorrem em comum dentro de um determinado conjunto de dados (MENZIES, HU, 2003) . Ou seja, buscar correlação entre atributos, onde o objetivo é encontrar itens que são mais frequentes e que impliquem na presença de outros na mesma transação. Após a identificação desses itens frequentes destina-se a identificar as regras fortes descobertas em bancos de dados usando diferentes medidas de interesse (suporte e confiança) (SHAPIRO, FRAWLEY, 1991) . Baseado no conceito de regras fortes, AGRAWAL et al. (1993), introduziu regras de associação para descobrir regularidades entre os produtos vendidos em transações de grande porte registradas por ponto-de-venda (POS) nos sistemas em supermercados. Uma regra de associação é uma expressão  $X \rightarrow Y$ , onde  $X$  é um conjunto de itens e  $Y$  é um item único. Isto significa num conjunto de transações, se todos os itens contidos em  $X$  existem em uma transação, então  $Y$  também ocorre na transação com alta probabilidade. Por exemplo, assume-se que consumidores que compraram pão e manteiga também compraram leite. Então a regra de associação "Pão e Manteiga  $\rightarrow$  Leite" será minerada. Essa análise associativa ficou conhecida como Market Basket Analysis.

Prosseguindo com esse detalhamento do ambiente KDD, Mineração de Dados e Regras de Associação, encontramos vários estudos de algoritmos para a implementação desta tarefa de Associação no contexto da Mineração de Dados. Um

grande número de algoritmos para extração de regras de associação tem sido desenvolvido com diferentes eficiências. Qualquer algoritmo pode encontrar o mesmo grupo de regras mas a eficiência computacional e o consumo de memória podem ser diferentes. Muitos dos esforços na pesquisa de regras de associação tem sido orientado para simplificar o conjunto de regras gerados e melhorar a performance do algoritmo. Mas esses não são apenas os únicos problemas que podem ser encontrados quando as regras são geradas e aplicadas em diferentes domínios. Outros problemas precisam ser levados em consideração na proposta de modelos de associação, e uma das maiores maiores desvantagens de algoritmos de regras de associação são: i) obter grande número de regras, ii) obter regras não interessantes, iii) baixa performance do algoritmo (MORENO, et al., 2005). Alguns dos principais algoritmos utilizados na implementação da mineração de itens frequentes e extração de regras baseado no limite de corte mínimo das medidas de suporte e confiança são: Apriori (AGRAWAL, SRIKANT, 1994), FP-Growth (HAN et al., 2004), ECLAT(ZAKI, 2000), dentre outros.

Os primeiros estudos de algoritmos na extração de regras de associação identificava a relação entre transações de um mesmo conjunto de dados usando valores binários, no entanto, transações com valores quantitativos são comumente vistos nas aplicações do mundo real. AGRAWAL et al. (1996), dando continuidade a suas pesquisas propôs um método para extração de regras de associação de atributos com valores quantitativos. Seu método proposto transforma os itens quantitativos em itens binários através da partição dos atributos com domínio quantitativo. Por exemplo, se o item Idade numa base de dados D possui valores de (0, 100], então ele pode ser particionado em três novos itens como Idade (0,30], Idade (30,60] e Idade (60,100] respectivamente. No entanto, esse método proposto não resolve o problema conhecido como *sharp boundaries* que consiste em poder subestimar ou superestimar elementos próximos dos limites da partição, e portanto levar a uma representação imprecisa de semântica.

Para tratar o problema de *sharp boundaries*, uma nova técnica utilizando *fuzzy sets* e *fuzzy* itens, normalmente em forma de rótulos ou termos linguísticos, são usados e definidos no domínio. KUOK et al. (1998), descreve *fuzzy association rules mining* (FARM) como sendo: "*Extração de regras de associação fuzzy é o descobrimento das regras de associação usando conceito fuzzy set de tal modo que atributos quantitativos podem ser manipulados*". Por exemplo, o atributo Idade pode

parecer tal como:  $F_{idade} = \{\text{criança, jovem, adulto, idoso}\}$ . O conceito de conjuntos *fuzzy* tem se mostrado melhor que o método de partição em itens binários porque conjuntos *fuzzy* proporcionam uma transição suave entre os membros e não membros de um conjunto, conseqüentemente trata melhor o problema de *sharp boundaries*. As regras extraídas são expressas em termos linguísticos, que são mais natural e interpretável para os seres humanos.

Tradicionalmente, a análise de regras de associação tem sido considerada como uma técnica não-supervisionada, por isso tem sido aplicado para tarefas de descoberta de conhecimento. Estudos têm mostrado que os algoritmos de descoberta de conhecimento, como a mineração de regras de associação pode ser aplicado com sucesso para a previsão de problemas de classificação. Nesses casos, os algoritmos usados para a geração de regras de associação deve ser adaptado às peculiaridades de previsões, a fim de construir classificadores eficazes. Alguns trabalhos foram realizados, onde algoritmos de mineração de associação foram modificados de modo que eles podem ser utilizados para a classificação / previsão. A proposta desta categoria de classificadores denominado de *class association rules* (CARs) é a classificação baseada em associação - *CBA (Classification Based on Association)* algoritmo. O algoritmo consiste de duas partes, um gerador de regra que é baseado no algoritmo Apriori para encontrar regras de associação em AGRAWAL, SRIKANT (1994), e um construtor de classificador com base nestas regras (LIU et al., 1998).

## 1.2 Objetivos

Visando estabelecer parâmetros para comparação de desempenho entre técnicas aplicadas na tarefa de extração de regras de associação da etapa de mineração de dados com atributos compostos de alores quantitativos, os objetivos estão assim definidos:

- analisar o desempenho dos algoritmos para extração de regras de associação Apriori e *Fuzzy Apriori*.
- preparar a base de dados, extraída de um sistema de detecção de intruso, para os experimentos.

- estabelecer métricas para medir o desempenho: i) quantitativo de regras de associação extraídas e ii) precisão das regras de associação extraídas.
- utilizar ferramentas GUI (*Graphical User Interface*) para os experimentos:
  - WEKA (*Waikato Environment for Knowledge Analysis*) (HALL et al., 2009) para o algoritmo Apriori e classificador.
  - KEEL (*Knowledge Extraction based on Evolutionary Learning*) (ALCALA et al., 2011) para o algoritmo FUZZY APRIORI e classificador.

### **1.3 Estrutura da dissertação**

A dissertação está estruturada em quatro capítulos. O capítulo 1 é apresenta a proposta de trabalho com a motivação, o objetivo e a estrutura os da dissertação. O capítulo 2 aborda a Fundamentação Teórica levando em consideração toda a teoria aplicada no trabalho. O capítulo 3 apresenta o Experimento com as ferramentas utilizadas, a descrição dos dados e toda a metodologia desenvolvida. O capítulo 4 aborda todos os resultados e análise dos experimentos.

## 2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo trata da fundamentação teórica necessária para a evolução do trabalho. A base teórica de temas relacionados ao estado da arte, Descoberta de Conhecimento em Base de Dados, mineração de regras de associação CRISP e FUZZY, abordagem simplificada de sistema de detecção de intruso e ferramentas de mineração de dados utilizadas no trabalho serão tratadas.

### 2.1 Antecedentes e Estado da arte

Várias técnicas de mineração de dados tem sido aplicada na detecção de intrusão, pois tem a vantagem de descobrir conhecimento útil que descreve o comportamento do usuário à partir de grandes conjuntos de dados. Indução de regras de associação é uma das abordagens mais conhecidas em técnicas de mineração de dados. A grande maioria dos trabalhos relacionados utilizam o conjunto de dados KDD99<sup>3</sup>. No entanto, diferentes *frameworks*<sup>4</sup>, categorias de ataques escolhido, métodos de avaliação, algoritmos e etc, se diferenciam entre os estudos deste tema.

Em WU et al., (2011), regras de associação fuzzy e classificador foram implementadas na linguagem standard C e para as regras de associação Booleana foi utilizada a implementação segundo Borgelt's <sup>5</sup>. Os resultados apontam taxa de detecção de 92.5% utilizando todos os registros.

EL-SEMARY et al., (2006) usando um sistema de mineração de dados que integra o algoritmo Apriori e Fuzzy Apriori (Kuok) para gerar regras fuzzy, capta a características de interesse do tráfego de rede. Usando um motor de inferência, implementado usando FuzzyJess, o sistema de detecção de intrusos avalia essas

---

<sup>3</sup> Conjunto de dados usado em *The Third International Knowledge Discovery and Data Mining Tools Competition* - <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>

<sup>4</sup> *Framework* é um conjunto de classes que colaboram para realizar uma responsabilidade para um domínio de um subsistema da aplicação (FAYAD e SCHMIDT).

<sup>5</sup> Christian Borgelt em <http://www.borgelt.net/index.html>.

regras e sinaliza para o administrador da rede indicações para se disparar o conjunto de regras.

IDRIS et al., (2009), também utilizou o algoritmo Fuzzy Apriori para propor um sistema híbrido capaz de detectar ambos os tipos de ataque, quer seja por uso indevido ou por anomalia. As regras inseridas num motor de inferência tiveram um percentual de detecção de 90% para o método de *uso indevido* e 85% para *anomalias*.

No estudo realizado por CHAPKE et al., (2012), foi proposto a criação de um sistema baseado em lógica fuzzy para identificar de forma eficaz as atividades de intrusão. Seu modelo híbrido incorporando técnicas melhoradas da lógica fuzzy e de mineração de dados, tinha como objetivo detectar ataques do tipo *uso indevido* e do tipo *anomalia*.

## **2.2 Sistema de Detecção de Intrusos**

### **2.2.1 Introdução**

Os dados que trafegam nas redes de computadores são cada vez mais importantes e valiosos, e essas devem ser seguras para que todos as pessoas possam utilizar sem risco e medo. O objetivo da segurança é fazer com que as pessoas se sintam livres e seguras para desfrutar as redes de computadores sem medo ou preocupação de comprometer suas informações garantindo assim seus direitos, interesses e privacidade (WANG, 2009).

A Segurança da Informação se refere à proteção existente sobre as informações tanto no ambiente corporativo (empresas) quanto às pessoais. Entende-se por informação todo e qualquer conteúdo ou dado que tenha valor para alguma organização ou pessoa.

A norma ABNT NBR ISO/IEC 27002:2005 define segurança da informação como a proteção da informação contra os vários tipos de ameaças com o objetivo de garantir a continuidade do negócio, minimizar seus riscos, e maximizar o retorno sobre os investimentos e as oportunidades do negócio.

Existem três aspectos chave da segurança da informação comumente referidos como CIA<sup>6</sup> Triad: Confidencialidade, Integridade e Disponibilidade.

- A confidencialidade: nenhum acesso à informação deverá ser garantido a pessoas ou sistemas não autorizados, isto é, apenas aqueles com os direitos e privilégios necessários e concedidos serão capazes de acessar a informação, independente do seu local de armazenamento, ou ainda, que esteja em processamento ou em trânsito. A violação da confidencialidade pode ocorrer por modo intencional através de ataques, captura de tráfego, entre outros, bem como de forma não deliberada por imperícia ou negligência da pessoa.
- A integridade: se refere à confiabilidade dos recursos de informação, e isto inclui o conceito de integridade de dados. Ou seja, que os dados não foram alterados de forma inadequada, seja por acidente ou deliberadamente por atividade nociva entre os momentos de transmissão e recepção.
- A disponibilidade: é o aspecto da segurança da informação que diz que esta deve estar disponível para ser acessada quando solicitada por uma pessoa ou sistema legítimo. Isto requer que toda estrutura que permite acesso e transporte da informação deve ser protegida para que não seja degradada ou se torne indisponível. Com relação à disponibilidade, os tipos de ataque que representam maior ameaça são DoS (Denial of Service) e DDoS (Distributed Denial of Service) , que visam danificar ou sobrecarregar sistemas.

Segurança é um processo e não um estado. Envolve quatro etapas, sendo estas o de avaliação, proteção, de detecção e de resposta ao ocorrido. A etapa de detecção é o tema principal deste trabalho, onde a aplicação dos algoritmos de extração de regras de associação se aplicam numa base de dados de um Sistema de Detecção de Intrusos.

Segundo BACE (2000), detecção de intrusão é definido como o processo de monitorar eventos que ocorrem em um sistema computacional ou redes de computadores, de forma a encontrar evidências de possíveis ataques a segurança.

---

<sup>6</sup> CIA (Central Intelligence Agency) é uma referência amplamente utilizada para a avaliação de segurança de sistemas de informação.

Existe uma lista extensa de tecnologias que protegem o perímetro das redes de computadores contra invasões e ameaças externas. No entanto, esses sistemas não podem interromper os invasores que obtêm acesso autenticado e se utilizem do sistema como se fossem usuários legítimos. Assim, torna-se importante monitorar as informações embutidas nos pacotes que atravessam o *firewall* para analisar como os usuários, legítimos ou não, utilizam seus ambientes computacionais, monitorando suas atividades e permitindo a detecção das ações de invasão (WANG, 2009).

Os primeiros conceitos sobre sistemas com características para detectar intrusos foram propostos por James P. Anderson em 1980 (ANDERSON, 1980). Segundo WANG (2009), os sistemas para detectar invasões tiveram seu início Dorothy Denning e Peter Neumann também em meados de 1980. Na época, eles observaram que os invasores tinham uma forma de agir diferente dos usuários normais do sistema, sendo assim possível a análise quantitativa do ataque. Para identificar os eventos de ataques nos sistemas, é necessário a criação de ferramentas inteligentes que analisam os protocolos de rede, análise de estatísticas computacionais e mineração de dados.

Essas ferramentas chamadas de Sistemas de Detecção de Intrusão (SDI), são sistemas automatizados que procuram por evidências de intrusão direta ou indireta, incluindo eventos que já aconteceram ou que estão atualmente em curso. Quanto mais rápida for a detecção do evento, menores serão os prejuízos para o ambiente computacional causados pelo ataque.

O funcionamento de um SDI é baseado na premissa de ser possível detectar o comportamento anormal, pois este será notoriamente diferente de um comportamento considerado padrão, usualmente praticado por um usuário legítimo. Normalmente, os SDI implementam modelos estatísticos ou se baseiam em regras que detectam o uso indevido.

DENNING (1987) propõe um modelo de sistema especialista para detecção de intrusão em tempo real capaz de detectar interrupções, penetrações e outras formas de ataque. Esse modelo se baseia na hipótese de que um evento pode ser detectado a partir do monitoramento dos registros de auditoria dos sistemas em busca de padrões de uso anormal.

### 2.2.2 Arquitetura dos SDIs

Os SDIs são classificados em dois modelos de monitoração: Host (HIDS - Host Intrusion Detection System) ou em Rede (NIDS - Network Intrusion Detection System). HIDS refere-se à detecção de intrusão que ocorre em um sistema host único. HIDS atualmente envolve a instalação de um agente na máquina local que monitora e informa sobre a atividade de configuração do sistema e da aplicação. Algumas habilidades comuns de sistemas HIDS incluem a análise de *log*, correlação de eventos, verificação de integridade, a aplicação de políticas, detecção de *rootkit*, e alertas. NIDS tem seu funcionamento baseado na análise do tráfego da rede, pacote a pacote em busca de atividades suspeitas, o que poderia ser um ataque ou atividade não autorizada.

Os SDIs podem detectar a intrusão para poucos ataques em tempo real ou a posteriori, onde neste último os dados são analisados de forma seqüencial *off-line* (batch). Geralmente os SDIs adotam uma configuração de “quase tempo real” para alguns ataques, ou seja, necessitam agregar informações suficientes durante algum tempo para poder fazer um julgamento. Ao detectar um ataque o SDI pode agir passivamente ou ativamente, onde passivamente consiste em enviar alertas sobre o evento, e ativamente atuar sobre os dispositivos de proteção fazendo com que os mesmos bloqueiem o ataque. Este modo deve ser utilizado com muito cuidado, pois pode se tornar uma nova vulnerabilidade.

As abordagens de detecção da intrusão consistem em três categorias. A primeira por uso indevido (*misuse detection*) que a partir de um banco de dados de assinaturas de eventos anteriormente caracterizados como maliciosos, procurar pela presença destas particularidades no tráfego analisado; a segunda por detecção por anomalia (*anomaly detection*) onde a partir de uma referência de características de comportamento considerados normais, pesquisa-se qualquer desvio que seja significativo dentro de limites pré-estabelecidos pelo especialista; e a terceira, os sistemas híbridos que basicamente utilizam as técnicas citadas anteriormente.

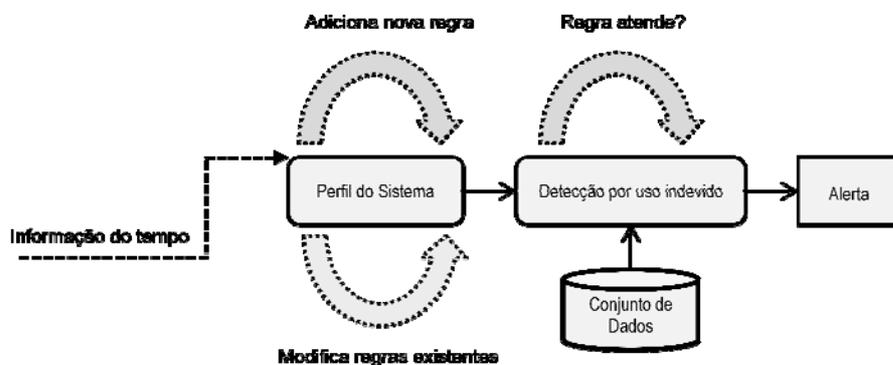
### 2.2.3 Detecção por uso indevido

A detecção por uso indevido se baseia na representação dos eventos na forma de padrões ou assinaturas os quais permitirão detectar pequenas variações nas características do ataque. Embora seja totalmente eficaz na descoberta de ataques conhecidos, é totalmente ineficaz quando confrontado com padrões de ataques novos ou desconhecidos já que as novas assinaturas ainda não estão disponíveis.

O desafio dos sistemas de detecção por uso indevido é definir regras que atendam todas as possibilidades de ataques. Quaisquer erros na definição destas assinaturas irá aumentar a taxa de falsos alarmes e comprometerá a eficácia da técnica de detecção.

A Figura 1 ilustra um modelo típico de detecção por uso indevido. O modelo consiste de quatro componentes, sendo: coletor de dados, perfil do sistema, detecção de uso indevido e saída / resposta em forma de alerta. Os dados são coletados a partir de uma ou várias fontes de dados, tráfego de rede, sistema de rastreamento, etc.

Figura 1 - Sistema de Detecção de Intrusão por uso indevido.



Os sistemas baseados em assinatura podem ser:

- Sistemas especialistas: capazes de analisar através de regras SE/ENTÃO, reconhecendo ataques e formulando uma solução ao problema;
- Por modelamento: compara informações de registros de auditoria com modelos de comportamentos de ataque para gerar uma suspeita ou para descartá-la.

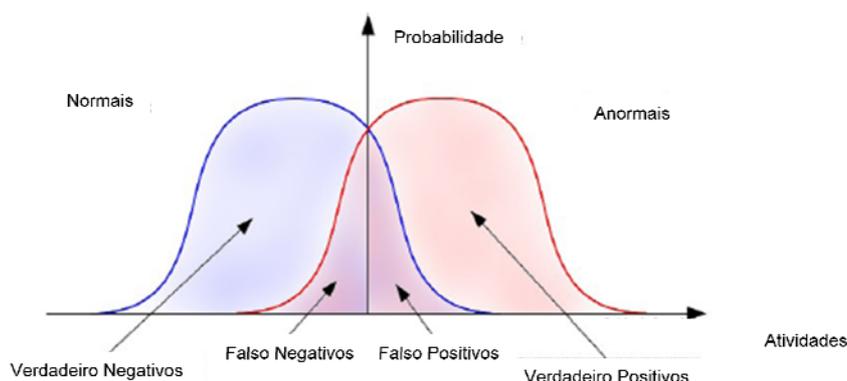
### 2.2.4 Detecção por anomalia

Diferente da técnica de detecção de uso indevido, detecção de anomalias é dedicado a estabelecer perfis normais de atividade para o sistema. Baseia-se no pressuposto de que todas as atividades de um intruso são necessariamente anômalas. Essa técnica estrutura primeiro um perfil de utilização normal do sistema, e qualquer variação ou desvio deste perfil que tenha significado estatístico e ultrapasse os limites previamente estabelecidos, pode vir a ser um ataque.

Os dados de tráfego tidos como normais são obtidos e salvos pelo componente coleção. O componente de detecção de anomalias decide quão longe as atividades atuais desviam-se dos perfis normais do sistema e qual a percentagem dessas atividades deve ser marcada como anormal. Finalmente o componente de resposta relata a intrusão e, possivelmente, correspondente informação de tempo.

A principal vantagem da detecção de anomalias é a sua capacidade para encontrar novos ataques assim que ocorrem, porém requerem uma fase de treinamento e uma configuração mais específica e cuidadosa do seu limiar de detecção. O seu maior desafio é estabelecer um limiar no qual não se deixe de detectar tentativas reais, chamadas de falso negativo (Figura 2).

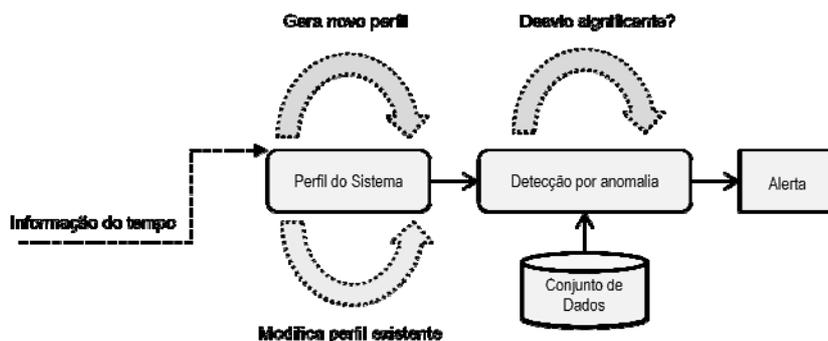
Figura 2 - Probabilidade de Detecção entre Atividades Normais e Anormais.



A Figura 3 ilustra um modelo típico de detecção por anomalia. O modelo consiste de quatro componentes, sendo: coletor de dados, perfil normal do sistema ,

detecção de anomalias e saída/resposta em forma de alerta. Os dados são coletados a partir de uma ou várias fontes de dados, tráfego de rede, sistema de rastreamento, etc.

Figura 3 - Sistema de Detecção de Intrusão por anomalia.



#### 2.2.5 As etapas do processo de KDD

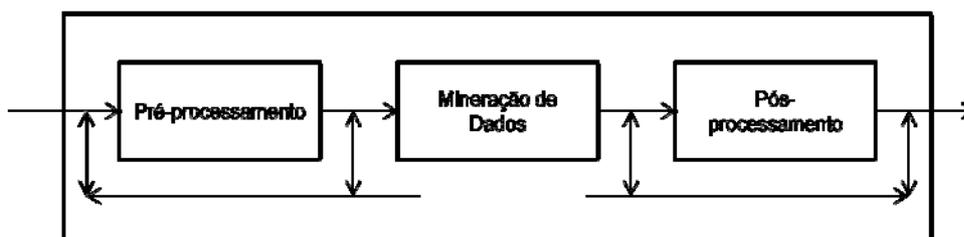
O processo de KDD é composto por três fases operacionais (Figura 4):

1ª fase: pré-processamento que compreende as funções que se relacionam a captação, à organização e ao tratamento de dados, cujo objetivo é preparar os dados para os algoritmos da etapa seguinte;

2ª fase: mineração de dados que realiza a busca efetiva de conhecimentos úteis no contexto proposto para a aplicação do KDD;

3ª fase: pós-processamento que abrange o tratamento do conhecimento obtido na etapa anterior (GOLDSCHIMIDT e PASSOS, 2005).

Figura 4 - Fases operacionais do processo de KDD

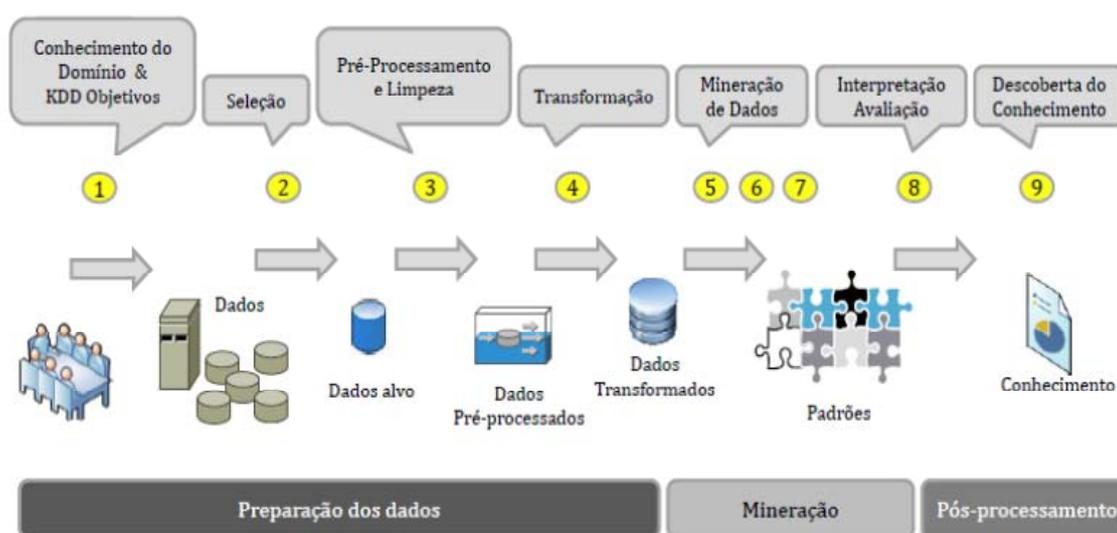


Fonte: GOLDSCHIMIDT e PASSOS. 2005

Segundo FAYYAD et al. (1996), o processo de KDD possui várias etapas interligadas e inseridas nas fases operacionais, sendo elas: seleção, pré-processamento e limpeza, transformação, mineração de dados, interpretação e avaliação, e que também pode ocorrer loops entre quaisquer de duas etapas.

De acordo com MAIMON e ROKACH (2005), as etapas da KDD possuem quatro etapas adicionais ao modelo de FAYYAD (1996), sendo duas delas conceituais (uma no início e outra no final do processo), e a etapa de Mineração de Dados é desdobrada em duas outras etapas internas (Figura 5).

Figura 5 – Etapas do processo de KDD



Fonte: O autor, 2013. Modificado - MAIMON e ROKACH (2005),

A seguir, tem-se uma breve descrição das nove etapas do processo de KDD, demonstrado na Figura 5, começando pela etapa de conhecimento do domínio e definição dos objetivos do KDD.

- i. **O desenvolvimento de uma compreensão do domínio da aplicação:** Esta é a etapa inicial. Ela prepara o cenário para entender o que deve ser feito com as muitas decisões que deverão de ser tomadas. As pessoas que estão na gerência de um projeto de KDD precisam entender e definir os objetivos do usuário final e do ambiente em que o processo de descoberta de conhecimento terá lugar. Conforme o processo de KDD avança, pode haver até mesmo uma revisão desta etapa.

- a. Tendo entendido os objetivos do KDD, o pré-processamento de dados se inicia, conforme definido nas próximas três etapas (note aqui que alguns dos métodos são similares ao algoritmo de Mineração de Dados (Data Mining) mas aqui são usados no contexto de pré-processamento).
- ii. **Selecionando e criando um conjunto de dados sobre o qual vai ser realizada a descoberta de conhecimento:** uma vez definidos os objetivos, os dados que vão ser utilizados para a descoberta de conhecimento devem ser determinados. Isso inclui descobrir que dado está disponível, obtendo dados adicionais se necessário, e, em seguida, integrando todos os dados para a descoberta de conhecimento em um conjunto de dados, incluindo os atributos que serão considerados para o processo. Este processo é muito importante porque o *Data Mining* aprende e descobre a partir dos dados disponíveis. Esta é a base de evidências para a construção dos modelos. Se alguns atributos importantes estão faltando, então todo o estudo pode falhar. Por outro lado, se para coletar, organizar e operar repositórios de dados complexos é caro por outro lado ganha-se com a oportunidade de melhor entendimento do fenômeno. Essa compensação representa um aspecto onde o interativo e iterativo de KDD está ocorrendo. Isso começa com o melhor conjunto de dados disponíveis e, posteriormente, se expande e observa o efeito em termos de descoberta de conhecimento e modelagem.
- iii. **Pré-processamento e limpeza:** Nesta etapa, a confiabilidade dos dados é melhorada. Isto inclui limpeza de dados, como a manipulação de valores faltosos e remoção de ruído ou discrepantes. Existem muitos métodos descritos na literatura, de não se fazer nenhum tipo de pré-processamento num projeto de KDD. Isto pode envolver métodos estatísticos complexos ou utilizar algoritmo de *Data Mining* neste contexto. Por exemplo, se existe uma suspeita que um determinado atributo é de confiabilidade insuficiente ou tem muitos dados faltosos, então este atributo poderia tornar-se o objetivo de um algoritmo supervisionado de *Data Mining*. Para este atributo será desenvolvido um modelo de predição e, em seguida, os dados faltosos, se for o caso, podem ser previstos.
- iv. **Transformação de dados.** Nesta fase, a geração de melhores dados para a próxima etapa de extração de dados (*Data Mining*), é preparada e

desenvolvida. Alguns métodos aqui incluem redução de dimensão e transformação de atributo (como discretização de atributos numéricos e transformação funcional). Esta etapa pode ser crucial para o sucesso de todo o projeto KDD, e é geralmente muito específico. Por exemplo, na realização de exames médicos, a combinação de atributos pode muitas vezes, ser o fator mais importante, e não cada um por si. No entanto, mesmo quando não usarmos a transformação correta dos dados no início, podemos obter um resultado inesperado que nos indique a uma transformação necessária (na próxima iteração).

Tendo completado as quatro etapas acima, as quatro etapas seguintes estão relacionados com a parte de *Data Mining*, onde o foco é sobre os aspectos algorítmicos empregados para cada projeto.

- v. **Escolhendo a tarefa apropriada de Mineração de Dados (*Data Mining*):** Nesta etapa é preciso definir que tipo de Mineração de Dados será usado, por exemplo, classificação, regressão, ou clustering. Isso depende principalmente dos objetivos de KDD, e também nas etapas anteriores. Há dois objetivos principais em *Data Mining*: predição e descrição. Predição é muitas vezes referido como *Data Mining* supervisionado, enquanto *Data Mining* descritivo inclui os aspectos não supervisionados e visualização de *Data Mining*. A maioria das técnicas de *Data Mining* são baseadas em aprendizagem indutiva, onde um modelo é construído explicitamente ou implicitamente pela generalização a partir de um número suficiente de exemplos de treinamento. A premissa fundamental da abordagem indutiva é que o modelo treinado é aplicável a casos futuros. A estratégia também leva em conta o nível de meta-aprendizagem para o conjunto particular de dados disponíveis
- vi. **Escolhendo o algoritmo de Mineração de Dados (*Data Mining*):** Definindo a estratégia agora é decidir sobre as táticas. Esta etapa inclui selecionar o método específico para ser utilizado para a busca de padrões. Por exemplo, em considerando a precisão versus a interpretabilidade, a primeira é melhor com redes neurais, enquanto que o último é melhor com árvore de decisão. Para cada uma das estratégias de meta-aprendizagem, existem diversas possibilidades de como ela pode ser realizada. Meta-aprendizagem centra-se em explicar o que faz com que um algoritmo de *Data Mining* seja bem sucedido ou não em um problema particular. Assim, esta abordagem tenta

compreender as condições em que um algoritmo de *Data Mining* é mais adequado.

- vii. **Empregando o algoritmo de Mineração de Dados (Data Mining):** Finalmente, a execução do algoritmo de mineração de dados é alcançada. Nesta etapa pode ser preciso aplicar o algoritmo várias vezes até que um resultado satisfatório seja obtido, por exemplo, por meio do ajuste de parâmetros de controle do algoritmo.
- viii. **Avaliação:** Nesta etapa, é avaliado e interpretado os padrões extraídos (regras, confiabilidade, etc), com relação às metas definidas na primeira etapa, centrando-se na compreensão e utilidade do modelo induzido. O última etapa é o uso e *feedback* geral sobre os padrões e os resultados descobertos obtidos pelo de *Data Mining*.
- ix. **Usando o conhecimento descoberto:** Nesta última etapa, o conhecimento torna-se ativo no sentido de que podemos fazer alterações no sistema e medir os efeitos. Na verdade, o sucesso deste passo determina a eficácia global do processo de KDD. Há muitos desafios nesta etapa, como perder as "condições de laboratório" em que operamos. Por exemplo, o conhecimento foi descoberto a partir de uma determinada imagem estática (normalmente amostra) dos dados, mas torna-se agora que os dados dinâmicos. Estruturas de dados podem mudar (certos atributos tornam-se indisponível), e o domínio de dados pode ser alterada (tal como, um atributo pode ter um valor que não foi considerada anteriormente).

#### 2.2.6 Mineração de Dados (Data Mining)

Uma das definições de mineração de dados foi dada anteriormente no capítulo 1 - Proposta de Trabalho, sendo: *Mineração de dados, de forma simplificadas, é o processo de extração ou mineração de conhecimento em grandes quantidades de dados (HAM, KAMBER, 2001)*. Assim, a etapa de mineração de dados localizada entre o Pré-processamento e o Pós-processamento, se torna muito importante de todo no processo KDD, pois é nela que são aplicadas as técnicas (algoritmos) de busca de conhecimentos a serem utilizados no problema em

questão, que podem ser feitas através de Redes Neurais, Algoritmos Genéticos, Lógica Difusa, modelos estatísticos e probabilísticos (GOLDSCHIMIDT e PASSOS, 2005). Neste caso, têm-se os seguintes passos:

- a. Seleção da tarefa de mineração de dados: uma ou uma combinação de tarefas deve ser selecionada dentre os diversos tipos de tarefas possíveis, dentre as mais utilizadas: classificação, regressão, agrupamento e associação.
- b. Escolha da técnica de mineração de dados: em função da tarefa selecionada, uma técnica específica, também denominado de algoritmo, será aplicado nos dados.
- c. Aplicação de mineração de dados: busca por padrões de interesse em uma forma representacional particular ou em um conjunto de aplicações.

A maioria dos autores consideram as principais tarefas da etapa de mineração de dados como sendo: classificação, regressão, agrupamento e associação, as quais são descritas a seguir segundo BERRY e LINOFF (1997).

- *Classificação* consiste em analisar previamente as características dos objetos e atribuir a cada um deles um conjunto predefinido de classes. Os objetos a serem classificados são geralmente representados por registros em banco de dados e o ato de classificação consiste em adicionar uma nova coluna com algum tipo de código de classe. Nesta tarefa, o modelo analisa o conjunto de registros fornecidos, com cada registro já contendo a indicação da classe que pertence, a fim de 'aprender' como classificar um novo registro. Por exemplo, categorizar um empregado pelo seu salário anual, identificando faixas de seus salários mensais em baixa, média e alta.
- *Regressão* é similar à classificação, porém a principal diferença é quanto à natureza do atributo a ser predito, que é quantitativo ao invés de ser qualitativo (discreto) (TAN et al., 2006). Assim, é possível estimar o valor de uma determinada variável analisando-se os valores das demais. Na prática, a regressão é muitas vezes usada para executar uma tarefa de classificação. Por exemplo, um conjunto de registros contendo os valores mensais de vendas de uma loja durante um período de tempo. Após ter analisado os dados, o modelo é capaz de dizer qual será o valor total de

venda em uma determinada época futura (não está sendo considerado outros fatores externos).

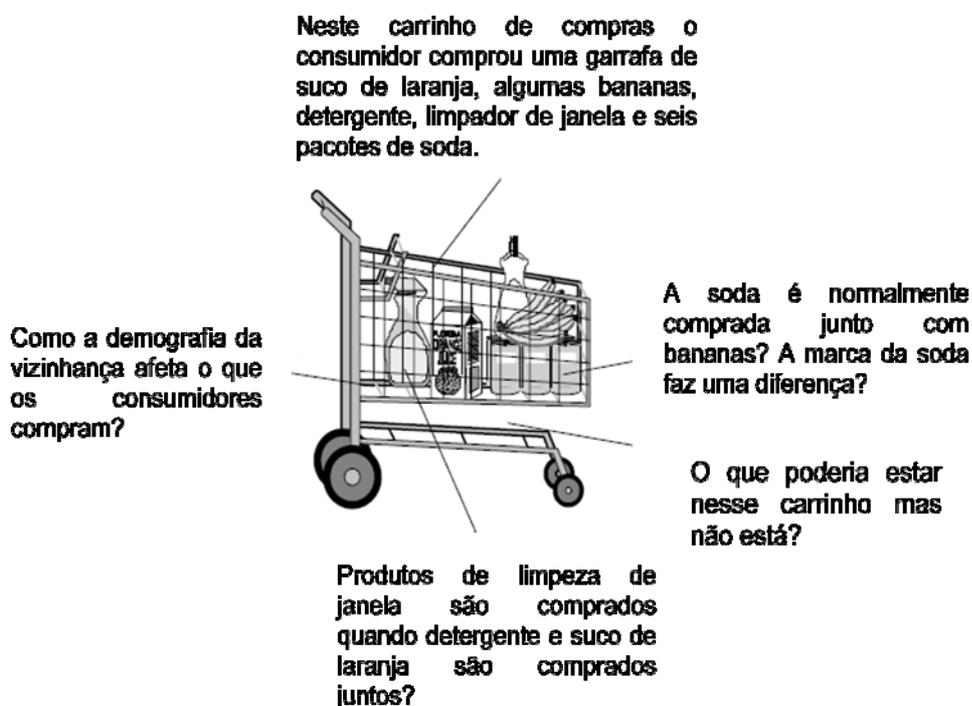
- *Agrupamento* ou *Clustering* visa segmentar um conjunto de dados heterogêneos num número de subgrupos mais homogêneos. Seu objetivo é formar grupos de registros o mais homogêneos em si, porém o mais heterogêneo dos outros grupos de dados. A diferença fundamental entre a formação de agrupamento e a classificação é que no agrupamento não existem classes predefinidas (aprendizado não-supervisionado) para classificar os registros. Os registros são agrupados em função de suas similaridades básicas, ou seja, quando se deseja formar agrupamentos, seleciona-se um conjunto de atributos (variáveis) e em função da similaridade desses atributos são formados os grupos. Por exemplo, podemos utilizar dados de um recenseamento nacional para formar grupos de domicílios, utilizando os atributos escolaridade, profissão, faixa etária, sexo, número de filhos. Observa-se que não existem classes pré definidas e poderemos ter num mesmo grupo domicílios de estados geograficamente opostos, porém, semelhantes nestes atributos (variáveis). Além disso, ela não tem a pretensão de classificar, estimar ou prever o valor de uma variável, ela apenas identifica os grupos de dados similares (CORTES et al., 2002).
- *Associação* também conhecida na área de marketing como grupos de afinidade ou *basket market analysis*, esta tarefa determina que “coisas” estão relacionadas, ou seja, descobrir as regras de associação condicionadas a valores de atributos que ocorrem juntos em um conjunto de dados. Se aplica nos casos em que deseja-se estudar preferências, afinidades entre itens, visando principalmente criar agrupamento de produtos formando pacotes de oportunidades para consumidores. Por exemplo, uma vez observado que dois itens são freqüentemente adquiridos juntos num supermercado, pode-se organizar as gôndolas de forma a oferecer estes produtos juntos, pois existe grande afinidade na preferência de seus compradores. Como esta tarefa é parte integrante desta dissertação será analisada em detalhes nos próximos tópicos.

### 2.2.7 Regras de associação na Mineração de Dados

Segundo LAROSE (2005), regras de associação ou análise de afinidades é o estudo dos atributos ou categorias de itens que estão relacionados, ou seja, aqueles que possuem alguma associação entre si. A representação do problema de extração de regras de associação, foi proposto inicialmente por AGRAWAL et al. (1993), onde introduziu regras de associação para descobrir regularidades entre os produtos vendidos em transações de grande porte registradas por ponto-de-venda (POS) nos sistemas em supermercados.

Métodos de regras de associação, tal como *market basket analysis*, procuram descobrir associações entre os atributos de forma a quantificar a relação entre dois ou mais atributos. Para transmitir a idéia de análise da cesta de mercado, observa-se a figura do carrinho de compras na Figura 6 contendo vários produtos adquiridos por um consumidor em um determinado supermercado.

Figura 6 – Análise de cesta de mercado ajuda a entender consumidores.



Fonte: BERRY e LINOFF (1997).

Um carrinho de compras nos informa sobre o que um cliente comprou em uma ida ao supermercado. Logo, podemos concluir que uma lista completa de compras feitas por todos os clientes em um determinado período fornece muito mais informação importante, tais como que produtos e quando. Cada cliente compra produtos diferenciados um dos outros, em quantidades diferentes e em épocas diferentes. A análise de cesta de mercado sugere informações relevantes sobre os produtos, como por exemplo, que produtos tendem a ser adquiridos em conjunto e que são mais favoráveis à promoção. Isto também impacta em revisão do layout das lojas fazendo com que os clientes sejam induzidos a adquirir produtos que normalmente não seriam adquiridos se estivessem em outra posição. A tarefa de mineração de dados afinada com a análise de cesta de mercado remete a geração de regras de associação que representam padrões nos dados e padrões de comportamento sem um alvo específico (BERRY e LINOFF, 1997).

Regras de associação tem a forma "SE" (antecedente), "ENTÃO" (consequente), juntamente com medidas de suporte e confiança associadas à regra.

A medida de suporte  $s$  para uma regra de associação em particular  $A \Rightarrow B$ , onde  $A$  é o *antecedente* e  $B$  o *consequente*, é a proporção de transações no conjunto de dados  $D$  que contêm ambos  $A$  e  $B$ , definido por:

$$\text{suporte} = P(A \cap B) = \frac{\text{número de transações que contêm ambos } A \text{ e } B}{\text{total do número de transações}} \quad (1)$$

A medida de confiança  $c$  para uma associação de regra  $A \Rightarrow B$  é a medida de precisão da regra, como determinado pelo percentual de transações em  $D$  contendo  $A$  que também contêm  $B$ , definido por:

$$\text{confiança} = P(B|A) = \frac{P(A \cap B)}{P(A)} = \frac{\text{número de transações que contêm ambos } A \text{ e } B}{\text{número de transações que contêm } A} \quad (2)$$

Por exemplo, um supermercado tem registrado em um dia da semana que de 1000 consumidores 200 compraram leite, e desses 200 que compraram leite, 50 compraram manteiga. Então, a associação de regra será: Se comprou leite (antecedente) *então* comprou manteiga (consequente), com medida de *suporte* de  $50/100 = 5\%$  e a medida de *confiança* de  $50/200 = 25\%$ .

## 2.3 Sistemas *Fuzzy*

A lógica nebulosa, difusa ou *fuzzy*, foi desenvolvida nos anos 60 por Lofti A. Zadeh, professor no departamento de engenharia elétrica e ciências da computação da Universidade da Califórnia, em Berkeley nos Estados Unidos, com o objetivo de fornecer um ferramental matemático para o tratamento de informações de caráter impreciso ou vago e auxiliar no controle e tomada de decisão.

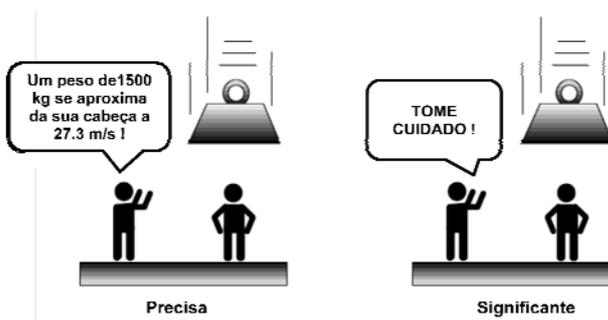
Sistemas *Fuzzy* é composto por um conjunto de fundamentos que vão desde a teoria de conjuntos *fuzzy*, funções de pertinência, lógica *fuzzy*, passando pelas regras SE/ENTÃO e fechando nos Sistemas de Inferência *Fuzzy*.

A Teoria de Conjuntos *Fuzzy* e os Conceitos de Lógica *Fuzzy* tem vasta aplicação em diversos setores, e basicamente sua utilização é aplicada para traduzir em termos matemáticos a informação imprecisa expressa por um conjunto de regras linguísticas. Se um operador de um equipamento qualquer tiver a capacidade de traduzir sua forma de operar este equipamento em um conjunto de regras da forma SE / ENTÃO, será possível desenvolver um algoritmo passível de ser implementado em computador. O resultado é um sistema de inferência baseado em regras, no qual a Teoria de Conjuntos *Fuzzy* e Lógica *Fuzzy* fornecem o ferramental matemático para se lidar com as tais regras linguísticas (TANSCHKEIT, 2007).

Lógica *Fuzzy* pode ser definida como sendo os princípios formais de raciocínio aproximado, com raciocínio preciso visto como um caso específico (ZADEH, 1988). Em geral a lógica *Fuzzy* é descrita como uma lógica booleana convencional sem a restrição de que todas as variáveis devem ser 0 ou 1. Lógica *Fuzzy* é tudo sobre a importância relativa da precisão. Respostas imprecisas as vezes são mais eficientes (Figura 7).

Segundo ROSS (1995), a linguagem natural é talvez a forma mais poderosa de transmitir informação de que os seres humanos possuem para qualquer tipo de situação que exija uma solução ou raciocínio. Por essas razões é que a Lógica *Fuzzy* é implementada na linguagem natural. Combinando a linguagem natural e informação numérica em Sistemas *Fuzzy* ou Nebulosos é possível fornecer uma estrutura que represente o conhecimento, restrições e procedimentos de inferência.

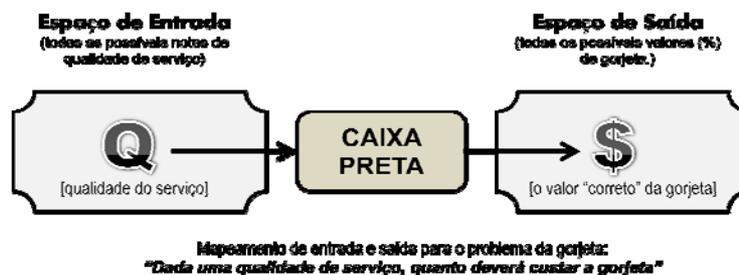
Figura 7 – Precisão e significância no mundo real.



Fonte: JANG et al. 1995.

Segundo JANG et al., (1995), a lógica *fuzzy* é uma forma conveniente para mapear um espaço de entrada para um espaço de saída. Semelhante a uma caixa preta que faz 'algo' para calcular a solução, ou seja, a saída. Este é o ponto de partida de tudo que será abordado nesta seção. Para representar o que seria dizer em mapear o espaço de entrada para o espaço de saída, temos o seguinte exemplo: Um cliente de um restaurante diz quanto bom foi o serviço de atendimento que será sugerido a ele um valor para a gorjeta (Figura 8).

Figura 8 - Exemplo de mapeamento de espaço de entrada e saída.



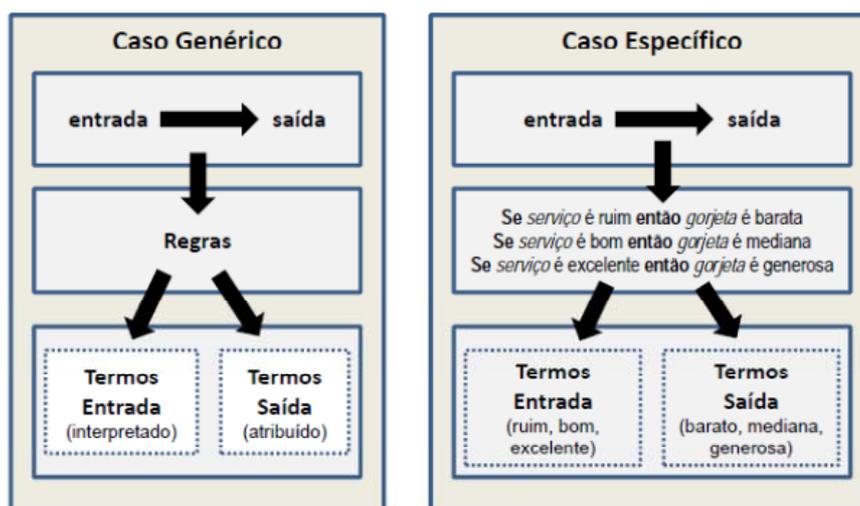
Fonte: Autor, 2014. Modificado - JANG et al. 1995.

Entre o espaço de entrada e o de saída existe uma caixa preta que vai executar "algo" para fazer este trabalho. Uma lista extensa poderia ser mencionada, tais como: sistemas *fuzzy*, sistemas lineares, sistemas especialista, redes neurais, equações diferenciais, e tantas outras mais recentes quando da publicação deste trabalho, tais como redes *neuro-fuzzy*, sistemas evolucionários agregados a outras soluções e etc. Então por que usar a lógica *fuzzy*? Segundo ZADEH (1965), a resposta é porque normalmente é mais interpretável, rápido e mais barato que as outras soluções (JANG et al., 1995).

O principal mecanismo para se fazer essa interligação entre o espaço de entrada com o de saída, através da caixa preta, é uma lista de instruções da forma SE - ENTÃO, chamadas de regras. A construção de um interpretador de regras implica em definir todos os termos que se planeja usar e as características que os descrevem. No exemplo da gorjeta é preciso definir as faixas de qualidade de atendimento para se dizer o quanto foi bom o atendimento.

O diagrama da Figura 9 é um roteiro básico do processo de inferência *fuzzy*, onde do lado esquerdo se tem uma visão genérica de um sistema difuso e do lado direito um sistema *Fuzzy* específico (exemplo da gorjeta). Em resumo, o conceito de inferência *fuzzy* representado nesta figura é um método que interpreta os valores do vetor de entrada e, baseado em um conjunto de regras, atribui valores ao vetor de saída.

Figura 9 - Roteiro básico de inferência *fuzzy*.



Fonte: Autor, 2014. Modificado - JANG et al. 1995.

### 2.3.1 Conjuntos *Fuzzy*

A principal limitação para a eficácia para a classe de sistemas especialistas<sup>7</sup> é o problema conhecido como *sharp boundary* (SBP), que leva à subestimação ou

<sup>7</sup> São Sistemas que têm como objetivo simular o raciocínio de um profissional especialista em alguma área de conhecimento bem específica

superestimação de elementos próximos dos limites da partição, como resultado da estratégia de particionamento de atributos quantitativos, o que, conseqüentemente, afeta a precisão do sistema especialista (VERLINDE, COOK, e BOUTE, 2006).

Na teoria clássica a pertinência de elementos em relação a um conjunto é assumido como termos binários de acordo com a sua condição *crisp*. Um elemento pertencer ou não a um conjunto, os limites desse conjunto são *crisp*. Ou seja, o conceito de pertinência de um elemento a um conjunto fica bem definido. Dado um conjunto  $A$  em um universo  $X$ , os elementos deste universo simplesmente pertencem ou não pertencem àquele conjunto. A função característica  $f_A(x)$  expressa esse conceito.

$$f_A(x) = \begin{cases} 1 & \text{se e somente se } x \in A \\ 0 & \text{se e somente se } x \notin A \end{cases} \quad (3)$$

ZADEH (1965) propôs uma abordagem mais ampla de tal forma que a função característica pudesse assumir um valor infinito de valores no intervalo  $[0,1]$ . Um conjunto *fuzzy*  $A$  em um universo  $X$  é definido por uma função de pertinência  $\mu_A(x): X \rightarrow [0,1]$ , e representado por um conjunto de pares ordenados

$$A = \left\{ \frac{\mu_A(x)}{x} \right\} \quad x \in X \quad (4)$$

onde  $\mu_A(x)$  indica o quanto  $x$  é compatível com o conjunto  $A$ . O que dessa forma, um determinado elemento pode vir a pertencer a mais de um conjunto *fuzzy* com diferentes graus de pertinência (TANSCHKEIT, 2007).

Segundo KUOK et. al, 1998, o conceito de conjuntos *fuzzy* é melhor do que métodos de partição binárias porque conjuntos *fuzzy* permitem uma suave transição entre membro e não membro em relação ao seu conjunto. Conseqüentemente o tratamento do problema de SBP é contornado minimizando a exclusão dos número de elementos próximo ao limite (*boundary*) (Figura 10).

Para representar a diferença da lógica convencional da lógica *fuzzy*, o conjunto de altura das pessoas será usado como exemplo (Figura 11).

Figura 10. Representação básica de conjunto *crisp* e *fuzzy*.

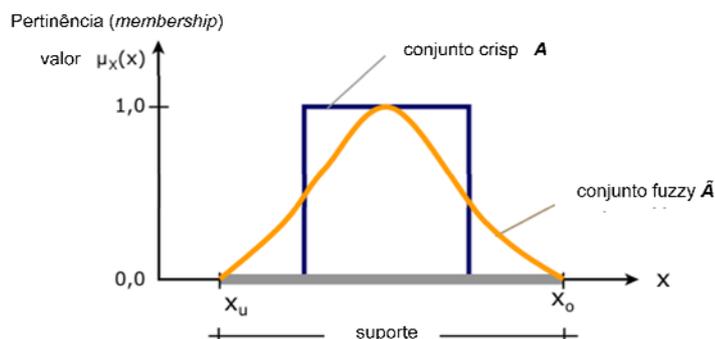
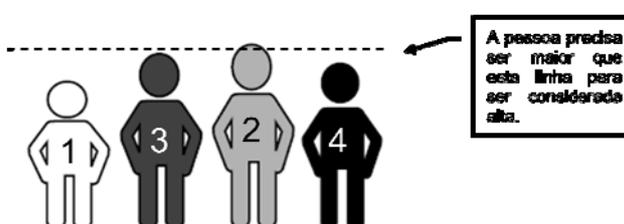


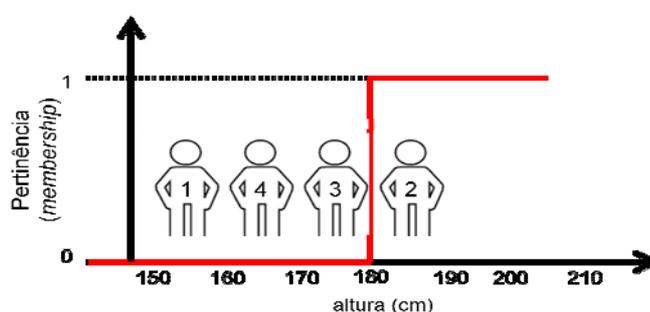
Figura 11 – Conjunto de altura das pessoas.



Fonte: Autor, 2014. Modificado - JANG et al. 1995.

A lógica convencional (*sharp boundary*) vai considerar a altura da pessoa como "alta", por exemplo, 180cm, e vai considerar uma pessoa como "baixa" sendo menor que 180cm (Figura 12). No entanto, se temos 2 pessoas, uma com 179.9 cm de altura e outra com 180.1 cm de altura, será considerado a primeira como sendo uma pessoa baixa e a segunda como uma pessoa alta. Não tem senso considerar uma pessoa baixa e outra alta se a diferença entre elas é somente 0.2 cm.

Figura 12 – Conjunto *crisp* de altura das pessoas.



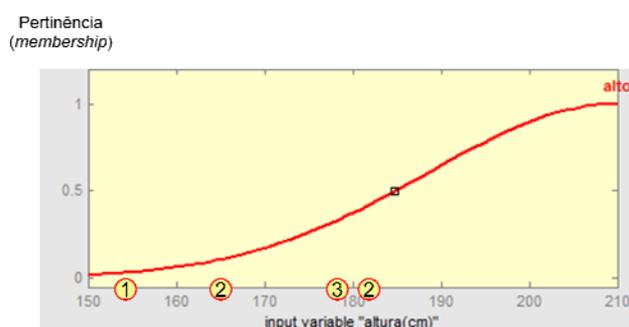
Fonte: Autor, 2014. Modificado - JANG et al., 1995.

O pensamento humano não trabalha desta forma. Não faz sentido dizer que a pessoa número 2 é alta e a número 3 baixa. Neste caso também temos graduação

entre alto e baixo. Também não podemos considerar o número 1 baixo e o número 4 também baixo, porque o número 4 realmente não é considerado alto, mas também não baixo como o número 1.

No contexto da lógica *fuzzy*, temos duas pessoas com certo grau de pertinência aos dois conjuntos, variando entre 0 e 1, ou seja, para dizer se uma pessoa é alta ou baixa temos fatores do raciocínio humano que vão ajudar nessa decisão. Dessa forma, pode-se concluir que os conjuntos *fuzzy* que classificam os elementos de um dado universo de discurso são menos rígidos do que aqueles utilizados na lógica convencional, pois eles admitem graus parciais de pertinência (entre 0 e 1). A Figura 13 mostra uma representação do conjunto *fuzzy* de altura das pessoas em termos de função de pertinência. Observa-se que não existe o problema de *sharp boundary* em 180 cm, mas uma transição suave.

Figura 13 – Conjunto *fuzzy* de altura das pessoas.



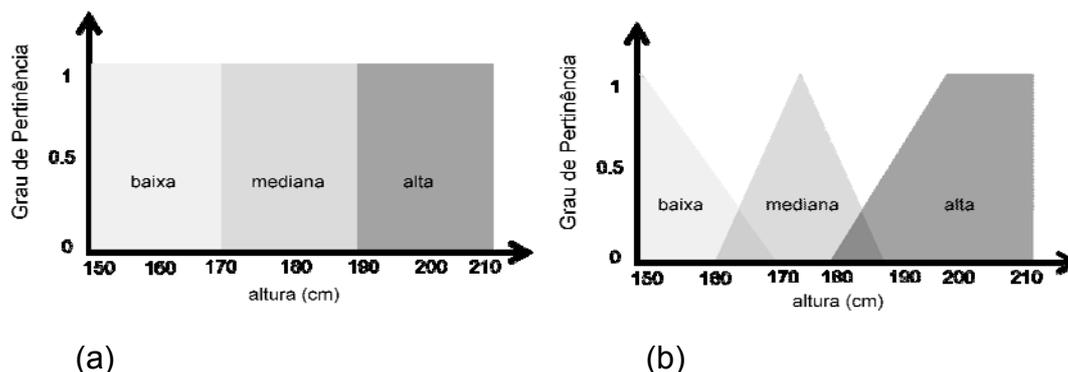
Fonte: Autor, 2014.

### 2.3.2 Variável linguística

Uma variável linguística é aquela cujos valores são palavras em lugar de números (TANSCHKEIT, 2007). Esses nomes são geralmente relacionadas a conceitos do cotidiano da linguagem humana, tais como: quente, morno, perto, muito perto, e no exemplo que está sendo abordado do conjunto de altura das pessoas, são as palavras "baixa", "mediana" e "alta". Essas variáveis linguísticas são descritas tanto por intermédio de conjuntos *crisp* como *fuzzy*, representados por funções de pertinência. No mundo *crisp*, os valores de grau de pertinência podem ser 0 e 1, e no mundo *fuzzy* esses valores podem variar entre 0 e 1.

No exemplo do conjunto das alturas, a faixa de alturas (universo de discurso) que vai de 150 a 210 cm foi dividida em três classes: baixa, mediana e alta. A representação Booleana<sup>8</sup> referente aos conjuntos crisp, pode ser representada conforme a Figura 14 (a). A Figura 14 (b), é representação *fuzzy* do conjuntos *fuzzy*.

Figura 14 - Variáveis linguísticas dos conjuntos de altura das pessoas.



Fonte: Autor, 2014. Modificado - JANG et al., 1995.

Essa representação das variáveis linguísticas é um ponto muito importante para essa dissertação, pois basicamente o trabalho se resume em fazer uma análise do modelo de particionamento de valores quantitativos que podem ser crisp através da discretização ou *fuzzy* através das variáveis linguísticas, em ambos para desses valores quantitativos.

Uma variável linguística nos permite descrever em valores qualitativos e quantitativos, caracterizada por uma quántupla  $(N, T(N), X, G, M)$ , onde:

$N$ : nome da variável.

$T(N)$ : conjunto de nomes dos valores linguísticos de  $N$ .

$X$ : universo de discurso.

$G$ : regra sintática para gerar os valores de  $N$  como uma composição de termos de  $T(N)$  conectivos lógicos, modificadores e delimitadores.

$M$ : regra semântica, para associar a cada valor gerado por  $G$  um conjunto *fuzzy* em  $X$ .

Em função do exemplo das variável altura do conjunto de altura das pessoas temos:

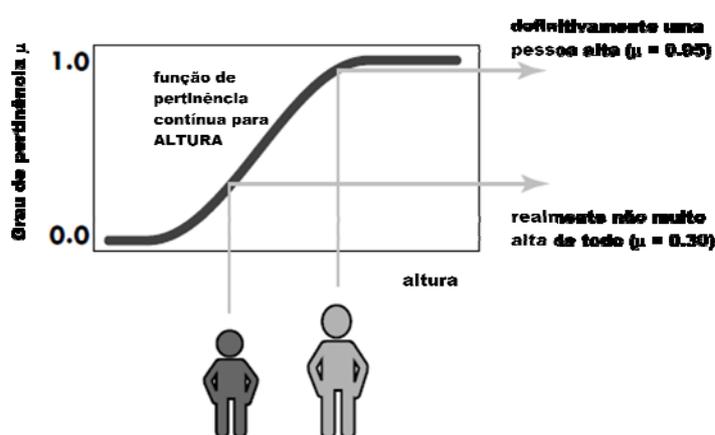
<sup>8</sup> São estruturas algébricas que "captam as propriedades essenciais" dos operadores lógicos e de conjuntos, ou ainda oferece um estrutura para se lidar com "afirmações" (SCHEINERMAN, 2003),

- $N$ : altura.
- $T(N)$ : {baixa, mediana, alta}
- $X$ : 150 a 210 cm
- $G$ : altura não baixa e não muito alta, por exemplo
- $M$ : associa o valor acima a um conjunto *fuzzy* cuja função de pertinência exprime o seu significado.

### 2.3.3 Função de pertinência

Uma função de pertinência (*membership function* - MF) é uma curva que define como cada ponto do espaço de entrada é mapeado para um valor de pertinência, ou grau de pertinência, entre 0 e 1, designado de  $\mu$ . Voltando ao exemplo do conjunto da altura das pessoas observou-se que a distinção entre a altura das pessoas de número 3 e 2 é inviável. Essa curva é que vai definir essa transição entre não tão alto para alto. Ou seja, ambas as pessoas são altas em algum grau de pertinência  $\mu$ , mas uma é significativamente menor do que a outra (Figura 15).

Figura 15 - Funções de pertinência para a variável altura.



Fonte: JANG et al., 1995.

Na seção de Variáveis Linguísticas a Figura 14 apresentou as funções de pertinência para a variável altura. Observa-se que estaturas de até 1,5 metros

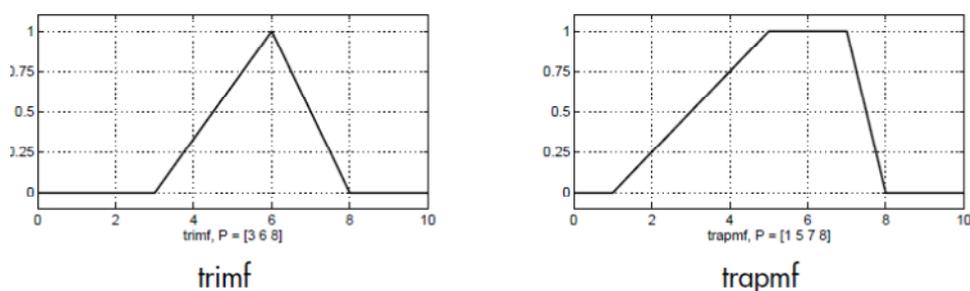
apresentam grau de pertinência igual a 1 no conjunto A (baixa). O grau de pertinência neste conjunto decresce à medida que a estatura aumenta. Considerando que uma altura de 1,75 metros é totalmente compatível com o conjunto B (mediana), enquanto que alturas acima de 1,80 metros (aproximadamente) apresentam grau de pertinência diferente de zero em C (alta). Pessoas com altura acima de 2 metros são definitivamente altas. Com relação a alturas entorno de 1,75 metros tem grau de pertinência diferente de zero somente no conjunto B, o que poderia parecer inadequado para alguns observadores. Estes talvez fosse preferir que as funções de pertinência de A e B se interceptassem em 1,75 metros (com grau de pertinência nulos), por exemplo.

Percebe-se que a construção das funções de pertinência podem variar em diferentes grupo de pessoas, pois depende da percepção de cada um e de suas experiências. Um jogador de basquete russo provavelmente teria percepção diferente para a criação das funções de pertinência de um jóquei de corrida de cavalo.

Segundo JANG et al, 1995, são diversos os tipos de funções de pertinência geralmente utilizados para a modelagem de uma problema *Fuzzy*. A única condição é que realmente precisam satisfazer a variação entre 0 e 1, podendo assim assumir diferentes tipos de curva. A escolha de qual função de pertinência a ser usada vai depender da aplicação sob o ponto de vista de simplicidade, conveniência, velocidade e eficiência.

É muito importante definir qual o tipo de função de pertinência que será usado porque isto depende em quanto tempo será preciso para desenhar e fazer os ajustes do sistema *fuzzy*. Usualmente é preferível utilizar funções de pertinência parametrizável, e não definir uma função de pertinência ponto por ponto. As funções de pertinências parametrizáveis mais comum e mais simples são aquelas formadas por linhas retas. Uma das mais simples é a *triangular (trimf)*. Nada mais é que um conjunto de três pontos formando um triângulo. A *trapezoidal (trapmf)* tem no topo uma linha reta formando que é na verdade uma curva triangular truncada. A função de pertinência *triangular* não apresenta descontinuidade entre a mudança do crescimento da função. Diferentemente da *trapezoidal* que apresenta descontinuidade. Essas funções de pertinências tem a vantagem da simplicidade (Figura 16).

Figura 16 - Funções de pertinências de linha reta.



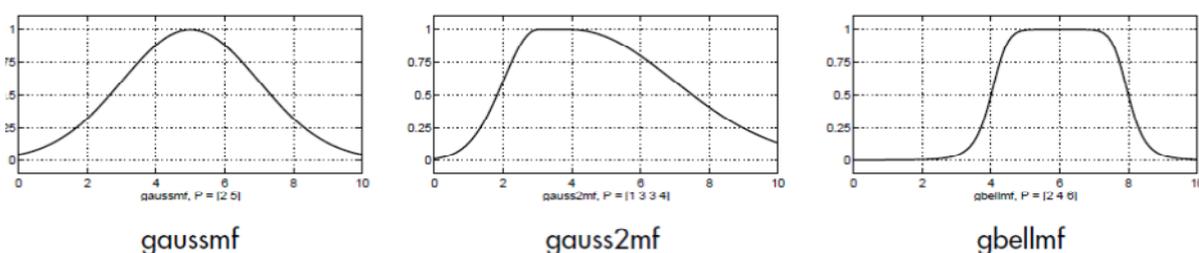
Fonte: JANG et al, 1995.

É possível utilizar outros tipos de funções de pertinência, mas a menos que haja um motivo muito bom para usar outras é preferível utilizar as funções de pertinência triangular ou trapezoidal. Elas tem se mostrado boas bastante nos resultados alcançados na maioria das aplicações (JANG et al, 1995).

Outras opções de funções de pertinência existentes, são:

i) as construídas em curvas de distribuição *Gaussiana*: Uma simples curva *Gaussiana* e um composto de dois lados com diferentes curvas *Gaussianas* (*gaussmf* e *gauss2mf*). Uma terceira função de pertinência *Gaussiana* nomeada de *generalized bell* (*gbellmf*) é especificada por três parâmetros, sendo um a mais que a função de pertinência *Gaussiana*. Por causa da suavidade e da sua firme conotação, *Gaussiana* e *generalized bell* são métodos populares para específicos conjuntos *fuzzy*. Ambas as curvas tem a vantagem de serem de transição suave e "não zero" em todos os pontos (Figura 17).

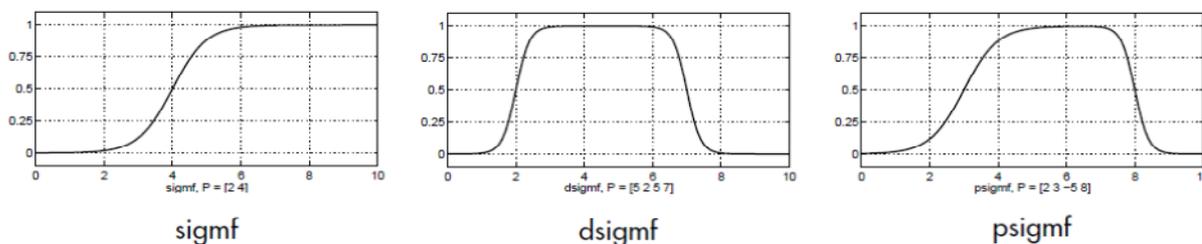
Figura 17 - Funções de pertinências *Gaussianas*.



ii) *Sigmoidal* que pode ser aberta para esquerda ou direita. Funções de pertinência assimétrica e fechada (sem ser aberta na esquerda ou direita) pode ser sintetizada usando duas funções *sigmoidal*, portanto além do básico

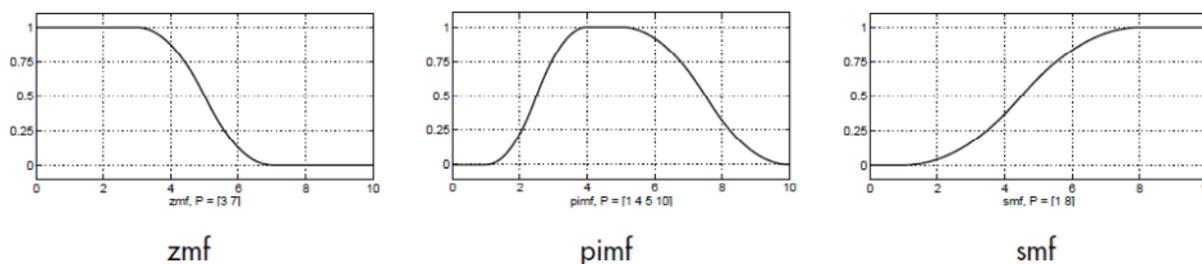
*sigmf* também tem-se a diferença entre funções *sigmoidal* (*dsigmoid*), e o produto de duas funções *sigmoidal* (*psigmoid*) (Figura 18).

Figura 18 - Funções de pertinências *Sigmoidal*.



iii) Curvas de base polinomial são responsáveis por várias funções de pertinência, dentre elas, nomeadas de: Z, S, e Pi. A função *zmf* é uma curva polinomial assimétrica aberta para a esquerda, *smf* é uma função imagem-espelho que se abre para a direita e *pimf* é zero em ambos os extremos com um aumento no meio (Figura 19).

Figura 19 - Funções de pertinências *Polinomial*.



Para o exemplo do conjunto da altura das pessoas, a variável altura foi representada por 3 funções de pertinência para cada um dos conjuntos *fuzzy*

A escolha de qual das funções de pertinência disponíveis deve ser adotada para representar o conjunto *fuzzy* depende: i) do contexto do problema estudado e ii) da experiência e perspectiva do usuário. Do ponto de vista apenas da Teoria *Fuzzy*, qualquer uma das funções de pertinência é tem potencial para representar um conjunto *fuzzy*, porém, o que deve ser notado é que cada uma das funções produz conjuntos *fuzzy* distintos. Importante destacar, que não existe uma regra única para o uso das funções de pertinência.

### 2.3.4 Regras Fuzzy SE - ENTÃO

As regras *fuzzy* são regras normais utilizadas para operar, da maneira correta, conjuntos *fuzzy*, com o objetivo de obter consequentes. Para criar tais regras é preciso de um raciocínio coerente com o que se deseja e obter. Para isso, este raciocínio pode ser dividido em duas etapas: (1) avaliar o antecedente da regra e (2) aplicar o resultado no consequente. , OU, SENÃO, E e uma combinação delas

Regras *Fuzzy* SE/ENTÃO, OU, SENÃO, E e uma combinação delas, são regras cujo antecedentes, consequentes ou ambos são *fuzzy* ao invés de *crisp*, e o consequente (resultado de saída) pode ser numérico ou lingüístico. A interpretação mais comum e amplamente utilizado considera uma regra *fuzzy* do tipo: "se  $x$  é  $A$  então  $y$  é  $B$ " como um ponto *fuzzy*  $A \times B$  e uma coleção de regras *fuzzy* "se  $x$  é  $A_i$  então  $y$  é  $B_i$ "  $i = 1, \dots, n$ . (ZADEH, 1992, 1994 ). Nesta estrutura de regra se/então, temos que:

- i) onde,  $A$  e  $B$  são termos lingüísticos de um conjunto *fuzzy* (por exemplo "alta" do exemplo anterior);
- ii) onde,  $x$  é o valor de pertinência de uma entrada ( $\mu_{ALTA}(x)$ );
- iii) e, ' $x$  é  $A$ ' é chamado de antecedente e descreve uma condição;
- iv) e, ' $y$  é  $B$ ' é chamado de consequente e descreve uma conclusão;

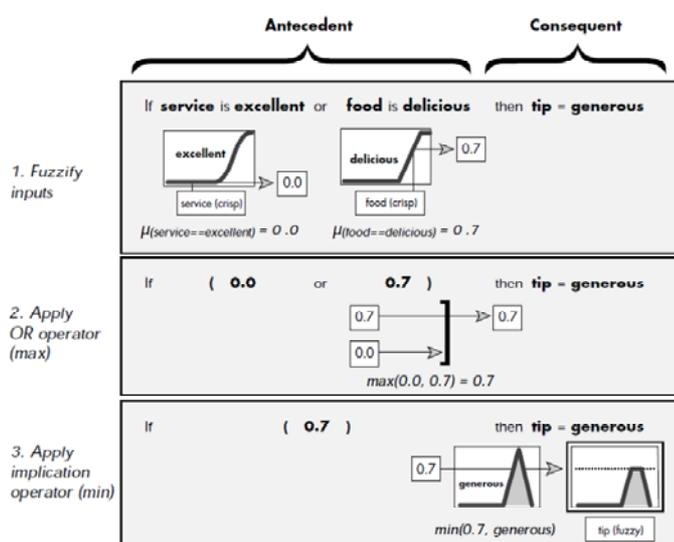
Referindo-se ao exemplo da gorjeta é possível ter uma regra do tipo: **se** (*serviço* é bom) **então** (*gorjeta* é média). Onde "bom" é representado como um número entre 0 e 1, e então, o antecedente é uma interpretação que retorna um número único entre 0 e 1. Por outro lado, "média" é representado como um conjunto *fuzzy*, e então, o consequente é uma atribuição que assume o conjunto *fuzzy*  $B$  inteiro para a variável de saída  $y$  (JANG et al, 1995).

Podemos citar outro exemplo de uma possível regra *fuzzy* de um sistema de detecção de intruso, e que poderia ocorrer neste trabalho: **se** (*service* é private) **e** (*count* é count\_low) **e** (*serv\_count* é serv\_count\_low) **então** (*atack\_type* é DoS). No caso tem-se os antecedentes *service* como valor lingüístico, *count* e *serv\_count* como valores quantitativos e o consequente *atack\_type* como valor lingüístico.

A palavra *is* é usada de diferentes maneiras dependendo se ela aparece como antecedente ou consequente. A primeira parte da regra ou antecedente é uma *condição* ou *premissa* e a segunda parte ou consequente é uma *atribuição* ou

*conclusão*. Para casos em que existam mais de um antecedente, é preciso encontrar um grau de pertinência resultante de todos os dos antecedentes (Figura 20). Nos casos em que o conectivo entre os antecedentes seja *e*, deve-se utilizar métodos de combinação, contanto que o resultado não ultrapasse o valor de menor pertinência entre os antecedentes; um exemplo de método seria o mínimo das pertinências. Já nos casos em que o conectivo entre os antecedentes for *ou*, deve-se utilizar métodos de combinação, contanto que o resultado não seja menor que o maior grau de pertinência (JANG et al, 1995).

Figura 20 - Regras *fuzzy* com dois antecedentes.



Fonte: JANG et al., 1995.

### 2.3.5 Sistema de Inferência *Fuzzy*

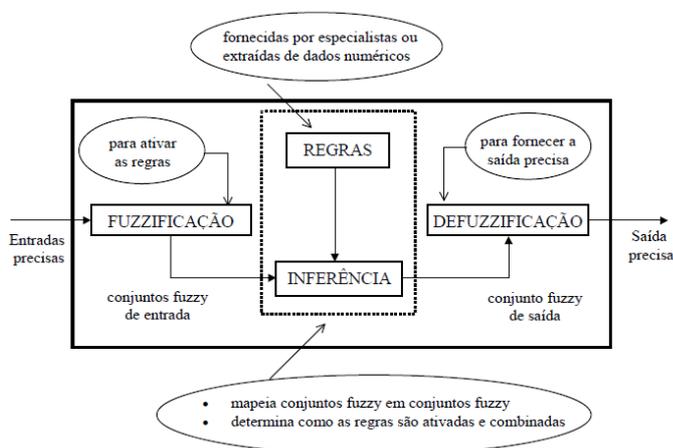
O Sistema de Inferência *Fuzzy* é uma estrutura computacional popular baseada nos conceitos de teoria de conjuntos *fuzzy*, regras *fuzzy* SE... ENTÃO e raciocínio *fuzzy* (JANG, 1995). Como também, através de um processo de avaliação do espaço de entrada com o objetivo de, através das regras previamente definidas e das entradas, obter resultados para uma tomada de decisão utilizando-se de todos os elementos que foram vistos neste tópico de Sistema *Fuzzy*: conjuntos *fuzzy*,

variável linguística, função de pertinência, operadores de lógica *fuzzy* e regras se/então.

Existem dois tipos de sistemas de inferência que variam conforme o tipo de saída determinado: Mamdani e Sugeno. O método de Mamdani foi um dos primeiros sistemas de controle construídos usando a teoria dos conjuntos *fuzzy*. De um modo geral, o método Sugeno pode ser utilizado para modelar qualquer sistema de inferência em que as funções de pertinência saída são linear ou constante.

Um Sistema de Inferência *Fuzzy* é mostrado na Figura 21, onde estão identificadas as funções de cada bloco.

Figura 21 - Sistema de Inferência *Fuzzy*



Fonte: Tanscheit (1995)

Segundo JANG et al., (1995), podemos fazer uma breve descrição dos módulos, sendo:

**Fuzzificação:** Nessa etapa para cada entrada, através das funções de pertinência, obtém-se o grau de pertinência com que cada entrada pertence a cada conjunto *fuzzy*. Cada uma dessas entradas foi previamente definida no seu respectivo no universo de discurso e associada a um grau de pertinência em cada conjunto *fuzzy* através do conhecimento do especialista. A entrada é um valor numérico e a saída um valor entre 0 e 1.

**Inferência:** Uma vez que as entradas foram *fuzzyficadas*, é sabido que o grau em cada parte do antecedente foi satisfeita para cada regra. Nessa etapa ocorrem as operações com conjuntos *fuzzy* propriamente ditas. Ou seja, combinação dos

*antecedentes* das regras, *implicação* e *modus ponens generalizado*<sup>9</sup>. Os conjuntos *fuzzy* de entrada, relativos aos antecedentes das regras, e o de saída, referente ao conseqüente, podem ser definidos previamente ou, alternativamente, gerados automaticamente a partir dos dados. Dois operadores são os principais na construção de funções de pertinência: E e OU. E são bons bastante para construir sistemas bem definidos.

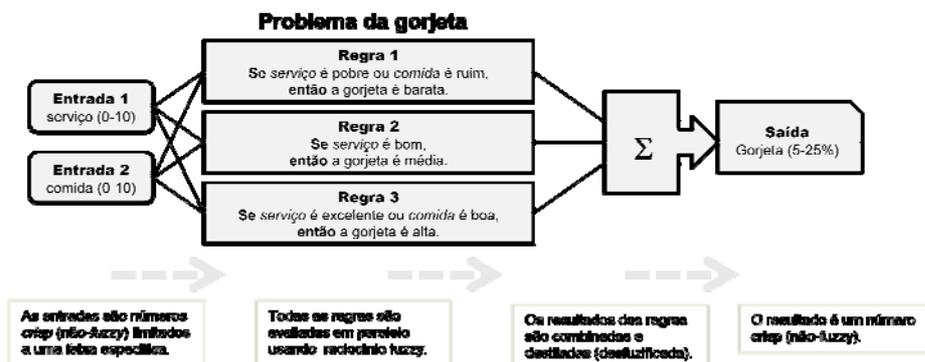
**Regras:** Podem ser fornecidas por especialistas, em forma de sentenças linguísticas, e são fundamentais no desempenho de um sistema de inferência *fuzzy*. Tomando como exemplo um sistema de detecção de intruso, este só terá uma taxa de precisão de acerto dos tipos de ataque se as regras que definem a estratégia de avaliação dos ataques forem consistentes. Extrair regras de especialistas na forma de sentenças do tipo SE/ENTÃO não é uma tarefa tão fácil, por mais conhecedores que eles sejam do problema em questão. Um método em substituição ao uso de especialistas para a definição da base de regras, existem métodos de extração de regras de dados numéricos que são os métodos que serão abordados neste trabalho.

**Defuzzificação:** Para se obter um valor de saída é necessário *defuzzificar* o conjunto *fuzzy* de saída obtido no processo de inferência. Nesta etapa é feita uma interpretação dessas informações. Existem vários métodos de defuzzificação, mas dois dos mais comuns são o *centro de gravidade (centroid)* e a *média dos máximos (mom)*. Dos dois, talvez o mais popular método seja o *centro de gravidade, que retorna o centro de uma área de uma curva*. Existem ainda outros três métodos: *bisector*, *maior dos máximos (largest of maximum)* e o *menor dos máximos (smallest of maximum)*. No caso do exemplo do problema da gorjeta, temos os seguintes diagramas que representam bem todo o processo de inferência *fuzzy*: Na Figura 22, tem-se o diagrama da estrutura básica do problema. Na Figura 23, o diagrama mostra de uma forma macro as etapas do fluxo das informações do processo do sistema de inferência *fuzzy*. E na Figura 24, tem-se todo processo real em detalhes.

---

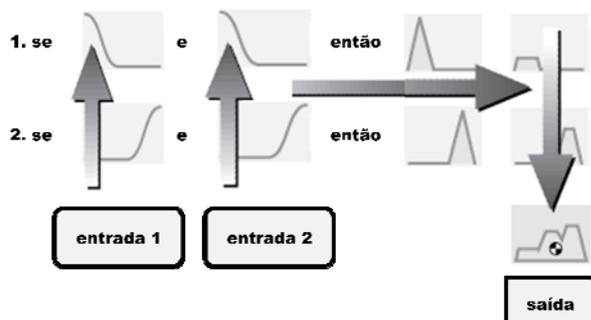
<sup>9</sup> Modus ponens: Ou MP é uma forma de argumento válido em lógica. Ex: Se o bolo é feito com açúcar, então o bolo é doce

Figura 22 - Diagrama do problema da gorjeta.



Fonte: JANG et al., 1995.

Figura 23 - Diagrama de inferência *fuzz*.



Fonte: JANG et al., 1995.

Figura 24 - Detalhamento do processo de inferência *fuzzy* .

Fonte: JANG et al., 1995.

## 2.4 Extração de Regras de Associação (Qualitativo)

O objetivo da mineração de dados é descobrir associações importantes entre os itens de tal forma que a presença de alguns itens em uma transação implicará a presença de alguns outros itens. Para atingir este objetivo, AGRAWL et al. (1994) propôs o conhecido algoritmo Apriori que, com se baseia no conceito de grandes conjuntos de itens para encontrar regras de associação entre os dados de uma transação (AGRAWL et al, 1995) (MANNILA. H. 1997). O processo de mineração foi dividido em duas fases. Na primeira fase, durante o scanamento das transações dos dados, os conjuntos de itens candidatos foram gerados e totalizados. Se o valor do suporte de um conjunto de itens nas transações dos dados for maior que o limite do valor mínimo de suporte previamente definido, o conjunto de itens será considerado um conjunto de itens grande (*large itemsets*).

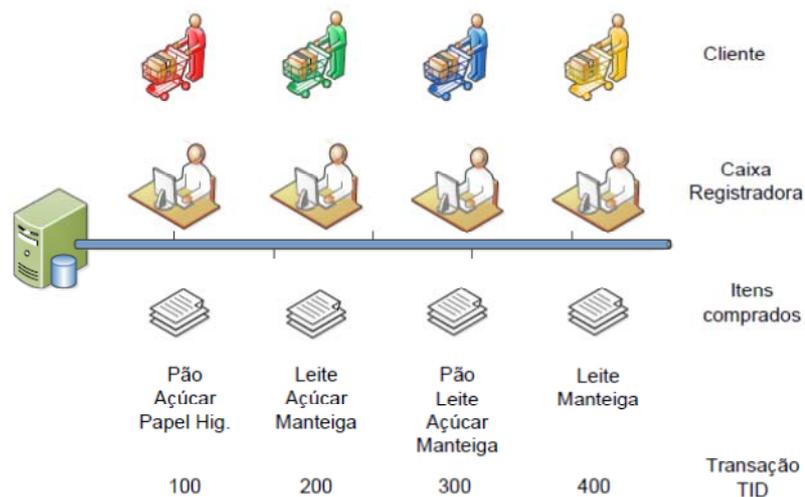
Conjuntos de itens grandes que contêm apenas os um único item são então combinados para formar conjuntos de itens candidatos que contenham dois itens. Este processo será repetido até que todos os grandes conjuntos de itens tenham sido encontrados. Na segunda fase, as regras de associação foram induzidas a partir de grandes conjuntos de itens encontrados na primeira fase

O método de extração de regras de associação, conhecido como *market basket analysis*, se baseia em descobrir regras de associações entre atributos de valores qualitativos.

Para melhor ilustrar a regras de associação consideremos outro exemplo de uma base de dados, representado pela Fig. 25, adaptado de CHEN et al., (1996) e de AGRAWAL e SRIKANT (1994), que é o banco de dados de transações efetuadas pelos clientes de um supermercado. A cada compra de um cliente, são registrados neste banco de dados todos os itens (produtos) comprados naquele momento. Se este mesmo cliente ao final da compra, isto é, após o pagamento na caixa registradora, retornar ao supermercado e efetuar uma nova compra, com um ou mais produtos, é considerado uma nova transação.

Podemos então dizer que a Tabela 1 representa as transações ocorridas na Figura 21. No exemplo em questão, consideramos  $I = \{i_1, i_2, \dots, i_n\}$  o conjunto de todos os itens da cesta de compras (pão, açúcar, leite, papel e manteiga), e  $T = \{t_1, t_2, \dots, t_n\}$  o conjunto de todas as transações (100, 200, 300 e 400).

Figura 25 – Transações de compra no supermercado.



Fonte: O autor, 2013. Modificado - CHEN et al (1996).

Cada transação  $t_n$  é um *itemset* que contém  $k$ -*itemset* onde  $k$  é o número de itens da transação. A transação TID=100 é um *itemset* com 3-*itemset*.

Tabela 1 – Transações de compras armazenadas no banco de dados.

TID	ITENS	ITEMSET
100	Pão, Açúcar e Papel	3- <i>itemset</i>
200	Leite, Açúcar e Manteiga	3- <i>itemset</i>
300	Pão, Leite, Açúcar e Manteiga	4- <i>itemset</i>
400	Leite e Manteiga	2- <i>itemsets</i>

Ainda no exemplo da Figura 5 podemos dizer que com relação aos itens leite, açúcar, papel e manteiga, temos as seguintes possíveis associações de regras e respectivas medidas de *suporte* e *confiança* apresentado na Tabela 2.

Tabela 2 – Associações de regras x suporte e confiança.

(se) antecedente (então) consequente	Suporte (%)	Confiança (%)
leite (3) => manteiga (3)	75	100
leite (3) => açúcar (2)	75	66,67
leite e açúcar (2) => manteiga (2)	50	100
pão e leite (1) => papel (0)	25	0
papel (1) => pão (1)	25	100

Neste caso cabe uma questão: Será que o fato de um possível associação de regra ter um grau de confiança alto é suficiente para a considerarmos uma regra forte? No exemplo anterior (Tabela 2), os itens pão e papel, aparecem juntos somente na primeira transação (TID 100), ou seja, poucos consumidores compram estes dois itens juntos. Entretanto, apenas pelo fato de que em 100% das transações onde o papel aparece o item pão também aparece, temos que o grau de confiança da regra {papel} > {pão} é de 100%. Nesse caso não podemos afirmar que pelo fato do grau de confiança ser alto a regra seja boa, pois esta confiança diz respeito somente a uma única transação suportada {pão, açúcar, papel}.

Neste sentido, as regras são consideradas fortes quando possuem os valores de suporte e confiança acima dos valores mínimos previamente definidos, e este critério vai depender do tipo de aplicação. Por exemplo, um analista de *marketing* interessado em encontrar qual supermercado compra juntos determinado itens, pode definir um nível mínimo de suporte e confiança como sendo 20% e 70% respectivamente. Por outro lado, um analista de segurança de TI teria necessidade de reduzir o nível de suporte mínimo para 1% ou menos, desde que comparativamente quanto menos transações fraudulentas são interessantes (LAROSE, 2005).

O grande problema que os algoritmos enfrentam ao extrair essas associações de regras é quanto a dimensionalidade. O número de possíveis associações aumenta exponencialmente com o número de atributos na ordem de  $k \times 2^{k-1}$  possíveis regras de associação. No exemplo anterior temos 4 itens, então seguindo essa relação teremos 32 possíveis associações de regras. Suponha-se que agora uma loja de conveniência tenha 100 diferentes itens onde um consumidor pode comprar ou não qualquer combinação desses 100 itens. Então um total de  $100 \times 2^{99} \approx 6.4 \times 10^{31}$  possíveis associações de regras aguardam para serem varridas pelo algoritmo.

Para contornar esse problema a maioria dos algoritmos possui uma estratégia em comum que é separar o problema de gerar regras de associação em duas subtarefas principais (TAN et al., 2006), a geração de itemsets frequentes e a geração das regras de associação propriamente dita.

**Geração de *itemsets* frequentes:** fazer uma busca no conjunto de dados e selecionar todos os *itemsets* que satisfaçam o limiar do valor mínimo de suporte (*minsup*);

**Geração de Regras de associação:** extrair todas as regras com confiança superior a *minconf* a partir dos *itemsets* frequentes encontrados na etapa anterior.

Nesse sentido, o algoritmo denominado Apriori foi um dos primeiros algoritmos que utilizaram a técnica de geração de itemset frequentes e posterior geração de regras de associação, e será base do desenvolvimento dessa dissertação.

#### 2.4.1 O Algoritmo APRIORI

Quando se trata do assunto de extração de regras de associação em mineração de dados o algoritmo Apriori é um dos mais conhecidos e considerado um clássico. Concebido pela equipe de pesquisa do Projeto QUEST da IBM que originou o software *Intelligent Miner*, (AGRAWAL, 1994) e (SRIKANT, 1994), trata-se de um algoritmo que resolve o problema de mineração de *itemsets* frequentes.

Dado um banco de dados com conjunto de transações  $\mathcal{D}$  e um nível mínimo de suporte  $\beta$ , o algoritmo encontra todos os *itemsets* frequentes com relação a  $\mathcal{D}$  e  $\beta$ . De posse de todos os *itemsets* frequentes com relação a  $\mathcal{D}$  e  $\beta$ , para obter-se as regras de associação interessantes, basta considerar, para cada *itemset* frequente  $L$ , todas as regras candidatas  $A \rightarrow (L - A)$ , onde  $A \subset L$  e testar para cada uma destas regras candidatas se o seu grau de confiança atende o nível mínimo de confiança  $\alpha$ .

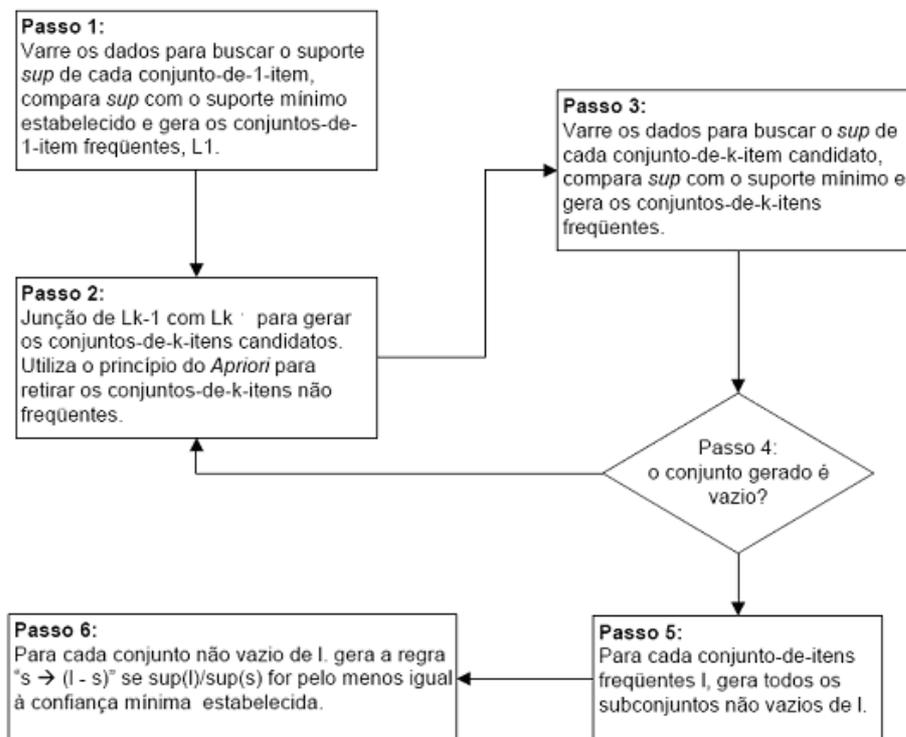
A estratégia do algoritmo Apriori é basicamente realizada em duas etapas, sendo elas: i) geração de itens frequentes e ii) geração de regras. A primeira etapa de geração de itens frequentes é realizada em duas outras subetapas, sendo elas: i) a *geração e poda de conjuntos de itens frequentes* e ii) o *cálculo de suporte dos candidatos*.

O diferencial deste algoritmo está na criação dos itemsets candidatos usando a propriedade *anti-monotonicity* que garante que, se um conjunto de itens não for

frequente, então todos os seus superconjuntos também não são frequentes. Assim, o algoritmo aumenta seu desempenho, já que não perde tempo varrendo e analisando esses superconjuntos. Essa otimização é possível de ocorrer porque a busca em largura garante que os valores dos suportes de todos os subconjuntos de um itemset candidato são previamente conhecidos.

Segundo PITONI (2002) apresenta, de forma simplificada, o fluxograma do processo descrito anteriormente (Figura 26).

Figura 26 - Fluxograma do algoritmo Apriori.



Fonte: PITONI,2002.

#### 2.4.1.1 Geração e poda de candidatos e cálculo do suporte

A Figura 27 mostra o pseudo-código principal do Algoritmo Apriori. O primeiro passo consiste numa varredura na base de dados para cada conjunto de *itemsets* candidatos de tamanho *k*, e que posteriormente, para cada próximo passo *k*, ocorrem as seguintes operações:

- i) Os conjuntos de itens candidatos de tamanho  $K$  são gerados, tendo como base o conjunto de itens freqüentes encontrados anteriormente (passo anterior  $k - 1$ ) através da função *apriori-gen*;
- ii) O suporte de cada candidato “ $c$ ” é calculado através de uma contagem “ $c.count++$ ”;
- iii) o algoritmo prossegue iterativamente até que o conjunto dos itens freqüentes  $L_k$  (que atingirem o suporte mínimo determinado) é gerado, cuja união constitui a “saída” do algoritmo.

Figura 27 - Pseudocódigo principal do algoritmo Apriori.

```

1. Entrada: Uma base de dados  $D$  e o valor de suporte mínimo  $min\_sup$ .
2. Saída: O conjunto  $L$  com todos os itemsets frequentes.
3. Função apriori-main( $D, min\_sup$ )
4.  $L_1 = \{\text{conjunto dos } itemsets \text{ frequentes de tamanho } 1 \text{ contidos em } D\}$ ;
5. para ( $k = 2; L_{k-1} \neq \emptyset; k++$ )
6.      $C_k = \text{apriori-gen}(L_{k-1})$ ;
7.     para todas transações  $t \in D$  fazer
8.          $C_t = \text{subset}(C_k, t)$ ;
9.         para todos candidatos  $c \in C_t$  fazer
10.             $c.count++$ ;
11.         fim para
12.     fim para

```

Fonte: MARIANO, 2011.

Observa-se que o algoritmo faz uso de duas funções, *apriori-gen* e *subset*, onde cada uma realiza uma operação específica (MARIANO, 2011).

- A função *apriori-gen* (linha 6), recebe como argumento o conjunto  $L_{k-1}$  (composto por todos os  $(k-1)$ -*itemsets* frequentes e retorna o conjunto  $C_k$  (composto por todos  $k$ -*itemsets* candidatos). A tarefa realizada por essa função é dividida em dois passos: *join* (junção) e *prune* (ajuste). No passo *join* são geradas as combinações através da junção de  $L_{k-1}$  com  $L_{k-1}$ . Já no passo *prune* são eliminados os *itemsets* candidatos que possuem algum subconjunto não frequente. Em resumo, é feito um ajuste onde são removidos todos os  $k$ -*itemsets*  $c \in C_k$  se existe um  $(k-1)$ -*itemset*  $s$ , em que  $s \subset c$  e  $s \notin L_{k-1}$ . Esses dois passos podem ser visualizados na Figura 28.
- A função *subset* (linha 8 da Figura 7) recebe como argumento o conjunto  $C_k$  com os *itemsets* já podados e uma transação  $t$  da base de dados, e

retorna o conjunto  $C_t$  composto pelos *itemsets* presentes em  $t$ . Essa função utiliza a *hash-tree* para encontrar os *itemsets* de  $C_k$  em uma transação  $t$ . Com isso, o suporte de cada *itemset*  $c$  em  $C_k$  é determinado no propósito de eliminar os *itemsets* que não são frequentes na base de dados de entrada.

Figura 28 - Pseudocódigo da função *apriori-gen* do algoritmo Apriori.

```

1.  Função apriori-gen( $L_k$ )
// Passo 1 (join)
2.  para cada itemset  $l_1 \in L_{k-1}$  fazer
3.      para cada itemset  $l_2 \in L_{k-1}$  fazer
4.          se ( $l_1[1] = l_2[1] \wedge l_1[2] = l_2[2] \wedge \dots \wedge l_1[k-1] < l_2[k-1]$ ) então
5.               $c = l_1[1] . l_1[2] \dots l_1[k-2] . l_1[k-1] . l_2[k-1]$ ;
6.              adicione  $c$  em  $C_k$ ;
7.          fim se
8.      fim para
9.  fim para
// Passo 2 (prune)
10. para todos candidatos  $c \in C_k$  fazer
11.     para todos ( $k-1$ )-subsets  $s \subset c$  fazer
12.         se ( $s \notin L_{k-1}$ ) então
13.             delete  $c$  de  $C_k$ ;
14.         fim se
15.     fim para
16. fim para
17. retorne  $C_k$ ;

```

Fonte: MARIANO, 2011.

O algoritmo Apriori, bem como outros algoritmos, na fase de *prune* (passo 2) exploram uma propriedade que otimiza seu desempenho na busca de conjuntos de itens frequentes. Conhecida como “*downward closure property*” de suporte (AGRAWAL, 1993) e (TAN, 2005), também conhecida como “*anti-monotonicity*” (PEI, HAN e LAKSHMANAN, 2001). Essa propriedade diz que se um conjunto de itens é frequente, então todos os seus subconjuntos também devem ser frequentes, além de utilizar recursos da memória principal e estrutura *hash*. Isso significa se  $X$  está contido em  $Y$  e  $X$  não é freqüente, logo  $Y$  também não é freqüente. Isto implica uma diminuição do tempo de execução, pois se  $X$  não é freqüente, então não será necessário calcular o suporte de  $Y$ , e o algoritmo não precisará ler novamente todo o bando de dados. Para ilustrar a ideia por trás dessa propriedade, considere a rede

de *itemset* da Figura 29 onde mostra uma rede de conjuntos de itens possíveis de combinação a partir da lista de itens {a, b, c, d, e}.

Suponha que {c, d, e} é um *itemset* frequente. Obviamente qualquer transação que contém {c,d,e} precisa também conter seus subsets, {c,d}, {c,e}, {d,e}, {c}, {d} e {e}. Como resultado, se {c,d,e} é frequente, então todos subsets de {c,d,e} (sombreados na figura) deve também ser frequentes (Figura 30).

Figura 29 – Rede de conjunto de itens.

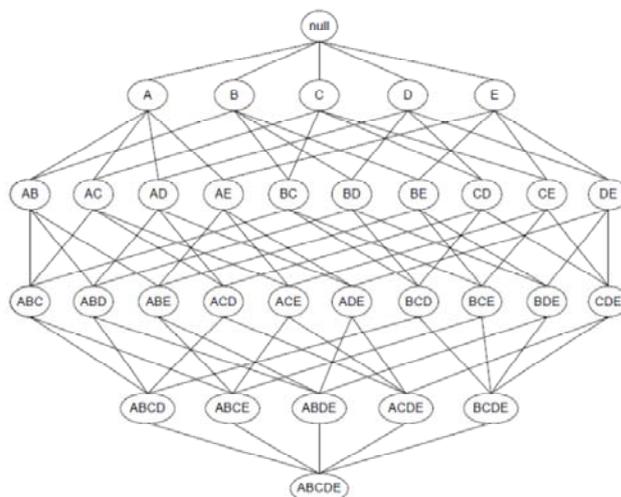
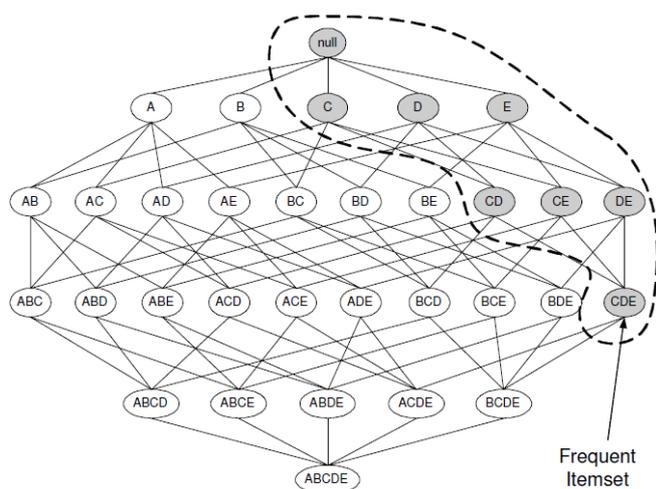


Figura 30 – Princípio Apriori aplicado na rede de itemsets



#### 2.4.1.2 Geração de regras de associação

Após gerar o conjunto  $L$  contendo todos os *itemsets* frequentes, é possível iniciar a fase de extração de regras de associação através do uso da função *ap-genrules* (Figura 31). Nessa fase o conjunto  $L$  é recebido como argumento juntamente com o valor *min\_conf* (valor de confiança mínima definida para as regras). A partir de cada *itemset* presente em  $L$ , são extraídos todos os subconjuntos de itens possíveis, onde em cada subconjunto, calcula-se a confiança da regra a ser gerada, definida por:  $confiança = \text{suporte}(X)/\text{suporte}(Y)$ , com  $X = \{k\text{-itemset} \mid k\text{-itemset} \in L\}$  e  $Y = \{1 \leq i < k \mid i\text{-itemset} \subset k\text{-itemset}\}$ . Caso o valor da confiança satisfaça o valor mínimo definido previamente, a regra poderá ser extraída. Sendo assim, todas as possíveis associações de regras são geradas a partir do conjunto de *itemsets* frequentes, e exibidas ao final da execução do algoritmo (MARIANO, 2011).

Figura 31 - Pseudocódigo da função *genrules* do algoritmo Apriori.

```

1. Entrada: Um conjunto de itemsets  $L$  e a confiança mínima da regra min_conf.
2. Saída: O conjunto de regras  $R$ .
3. Função ap-genrules( $L$ , min_conf)
4. para todos  $k\text{-itemsets} \in L$  fazer
5.     para ( $i = k-1$ ;  $i \geq 1$ ;  $i--$ )
6.         para todos  $i\text{-itemsets} \subset k\text{-itemset}$  fazer
7.              $conf = \text{suporte}(k\text{-itemset}) / \text{suporte}(i\text{-itemset})$ ;
8.             se ( $conf \geq \text{min\_conf}$ ) então
9.                 adicione  $i\text{-itemset} \rightarrow (k\text{-itemset} - i\text{-itemset})$  em  $R$ ;
10.            fim se
11.        fim para
12.    fim para
13. fim para
14. retorne  $R$ ;

```

Fonte: MARIANO, 2011.

#### 2.4.2 Exemplo de aplicação - Qualitatito / Booleanos

De maneira a exemplificar o funcionamento do Algoritmo Apriori vamos utilizar a mesma base de dados da Figura 5. A base de dados  $\mathcal{D}$  formada por cinco atributos {pão, leite, açúcar, papel, manteiga} e quatro transações (TID) (Tabela 3), e tendo como medida de suporte mínimo 50%.

Para ilustrar o problema, na Tabela 4 é exibida as ocorrências e o suporte de cada atributo, mas essa fase inicial de ordenar o *Conjunto de Itens*, a contagem de cada item e sua respectiva incidência dentro da base é feita pelo algoritmo gerando  $C_1$ .

Tabela 3 – Base de dados  $\mathcal{D}$  e respectivos *itemsets*.

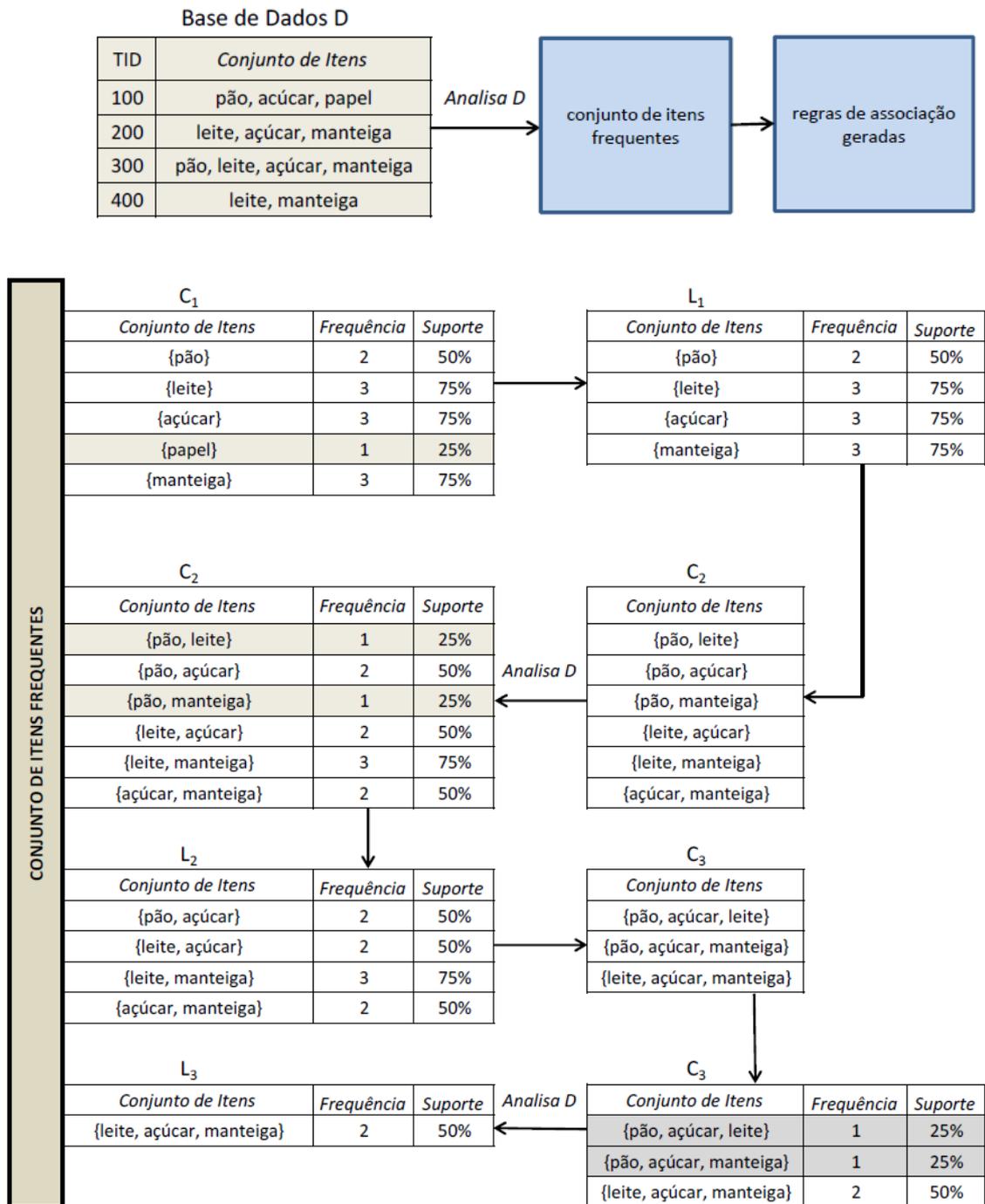
TID	PÃO	LEITE	AÇUCAR	PAPEL	MANTEIGA
100	SIM	não	SIM	SIM	não
200	não	SIM	SIM	não	SIM
300	SIM	SIM	SIM	não	SIM
400	não	SIM	não	não	SIM

Tabela 4 – Ocorrência e suporte dos atributos.

	PÃO	LEITE	AÇUCAR	PAPEL	MANTEIGA
Ocorrências	2	3	3	1	3
Suporte	50%	75%	75%	25%	75%

Todas as fases subsequentes descritas a seguir estão representadas na Figura 32. O próximo passo é efetuar a “poda” do item que não obtiver uma frequência na base de dados. Para o exemplo estará sendo considerado como medida de suporte mínimo o valor de 50% (frequência 2). Nesse caso, o item que sofrerá a “poda” é o {papel}, pois contém apenas uma incidência na base, gerando então o conjunto de itens  $L_1$ . Em seguida, o algoritmo efetua mais uma verificação das incidências de itens compostos e os ordena, gerando assim  $C_2$ . Mais uma vez o algoritmo efetua a análise da base  $\mathcal{D}$  e novamente exclui os itens que não possuem frequência mínima na base, que nesse caso são os itens {pão, leite} e {pão, manteiga}, gerando o  $L_2$ . Em seguida, é feita uma nova checagem das incidências para gerar um novo conjunto de itens  $C_3$  que neste caso gerou {leite, açúcar, manteiga}. O algoritmo retorna as análises da base  $\mathcal{D}$  para gerar o  $L_3$  e finaliza o processo, pois não existe mais conjunto a ser gerado além do conjunto de itens  $L_3$ .

Figura 32 - Exemplo de aplicação do Algoritmo Apriori.



REGRAS DE ASSOCIAÇÃO GERADAS	REGRAS CANDIDATAS COM 2 ITENS			REGRAS CANDIDATAS COM 3 ITENS		
	<i>Conjunto de Itens</i>	{leite e açúcar}	<i>Confiança</i>	<i>Conjunto de Itens</i>	{leite, açúcar, manteiga}	<i>Confiança</i>
	SE	ENTÃO		SE	ENTÃO	
	{leite}	{açúcar}	66.67%	{leite e açúcar}	{manteiga}	100%
	{açúcar}	{leite}	66.67%	{leite e manteiga}	{açúcar}	66.67%
	<i>Conjunto de Itens</i>	{leite e manteiga}	<i>Confiança</i>	{açúcar e manteiga}	{leite}	100%
	SE	ENTÃO		{leite}	{açúcar e manteiga}	66.67%
	{leite}	{manteiga}	100%	{açúcar}	{leite e manteiga}	66.67%
	{manteiga}	{leite}	100%	{manteiga}	{leite e açúcar}	66.67%
	<i>Conjunto de Itens</i>	{açúcar e manteiga}	<i>Confiança</i>			
SE	ENTÃO					
{açúcar}	{manteiga}	66.67%				
{manteiga}	{açúcar}	66.67%				

Fonte: Autor, 2014.

## 2.5 Extração de Regras de Associação (Quantitativo - CRIPS)

Conforme visto na seção anterior, em AGRAWAL et al. (1994), foi proposto método para extração de regras de associação de valores binários. Entretanto, atributos podem ser quantitativos ou qualitativos. Atributos booleanos podem ser considerados um caso especial de atributos qualitativos. Como não seja é possível aplicar diretamente algoritmos binários é preciso transformar o problema de atributos quantitativos em binários.

AGRAWAL et al. (1996), propôs um método para extração de regras de associação, mas dessa vez para atributos com valores quantitativos. Seu método propõe primeiro determinar o número de partições para cada atributo quantitativo, e então é mapeado todos os valores possíveis de cada atributo em um conjunto de inteiros consecutivos (discretizar). Em seguida é encontrado os *itemsets frequentes* cujo valor de suporte seja maior que o valor mínimo de suporte previamente definido pelo usuário. Esses *itemset* frequentes são então processados para gerar as regras de associação.

### 2.5.1 Mapeando atributos quantitativos em Booleanos

Um conjunto de transações é formado por linhas (instâncias) e colunas (atributos), onde para cada atributo tem-se um item e um valor correspondente para cada instância. O valor de um atributo para um dado registro é "1" se o item correspondente ao atributo é presente na transação correspondente ao registro, e "0" se não. Atributos podem ser quantitativos (idade, salário) ou qualitativo (cor, marca de carro). Atributos booleanos pode ser considerados como um caso especial de atributo qualitativo.

Se o domínio de valores quantitativos é grande, primeiro será feito uma partição dos valores em intervalos e então mapear cada par (atributo, intervalo) em atributo booleano. Uma vez mapeado todos os atributos é possível usar qualquer algoritmo aplicado para extração de regras de associação *fuzzy*, como por exemplo o algoritmo Apriori (AGRAWAL et al., 1994).

### 2.5.2 Exemplo de aplicação - Quantitativo / Booleano

Considerando a tabela "Pessoa" da Figura 33 (a), tem-se dois atributos quantitativos, *Idade* e *NumCars*. O atributo *Idade* é particionado em dois intervalos e o atributo qualitativo (categórico), *Casado*, tem dois atributos booleanos "Casado: sim" e "Casado: não". Como o número de valores para *NumCars* é pequeno, *NumCars* não é particionado em intervalos; cada valor é mapeado para um campo booleano (Figura 33 (b)). Assumindo um valor mínimo de suporte de 40% e de confiança mínima de 50%, Figura 33 (c) mostra algumas algumas regras. (AGRAWAL et al., 1996).

Figura 33 – Exemplo de extração de regra quantitativa - *Crisp*.

Pessoas			
ID	Idade	Casado	NumCars
100	23	não	1
200	25	sim	1
300	29	não	0
400	34	sim	2
500	38	sim	2

(a)

Mapeando para Regras de Associação Booleana							
ID	Idade {20..29}	Idade {30..39}	Casado (sim)	Casado (não)	NumCars {0}	NumCars {1}	NumCars {2}
100	1	0	0	1	0	1	0
200	1	0	1	0	0	1	0
300	1	0	0	1	1	0	0
400	0	1	1	0	0	0	1
500	0	1	1	0	0	0	1

(b)

Amostra: Regras		
Regra	Suporte	Confiança
{idade: 30..39} e {casado: sim} $\Rightarrow$ {NumCars: 2}	40%	100%
{idade: 20..29} $\Rightarrow$ {NumCars: 1}	60%	66.6%

(c)

### 2.5.3 Classificação associativa - Algoritmo CBA

O algoritmo aplicado para classificação das regras de associação foi o classificador JCBA (implementação JAVA do algoritmo CBA) que trabalha com regras de associação de classe

Extração de regras de classificação destina-se a descobrir um conjunto pequeno de regras que vai formar um classificador preciso. Enquanto em regras de associação a classe (target) pode ou não estar presente, que é o caso do conjunto de dados utilizado neste trabalho onde a classe é o tipo de ataque, para regras de classificação existe um e somente um pre-determinado atributo classe.

A proposta do algoritmo introduzida por LIU et al. (1998), chamada de algoritmo CBA (Classification Based on Association), é integrar essas duas técnicas

de extração (associação e classificação). A sua construção é dividida em duas etapas. A primeira é feita com a extração de um subconjunto especial de regras de associação, cujo lado direito da regra (consequente) são restritos a classificação do atributo classe. Este subconjunto de regras é chamada de *class association rules* (CARs), que é uma adaptação do algoritmo de extração de regras de associação introduzido por AGRAWAL e SRIKANT (1994), e a segunda, é construir um classificador baseando-se nas regras CARs geradas.

Para construir o melhor classificador de todo o conjunto de regras gerados, o algoritmo envolve a avaliação de todos os possíveis subconjuntos de regras, onde aplicados no conjunto de dados treinamento, é selecionado o subconjunto de regras que possui a seqüência de regras que vai fornecer o menor número de erros durante a classificação. Ou seja, seja  $R$  um conjunto de regras gerados (CARs), e  $D$  o conjunto treinamento, o algoritmo vai aplicar filtros para eliminar as regras de  $R$  que geram mais erros (p.ex. regras conflitantes), selecionando assim, um conjunto de regras de alta confiança em  $R$  que satisfaça  $D$  para criar um classificador.

## 2.6 Extração de Regras de Associação (Quantitativo - FUZZY)

Conforme visto na seção anterior, em AGRAWAL et al. (1996), foi proposto extração de regras de associação de atributos com valores quantitativos. Este algoritmo encontrava as regras de associação particionando os domínios dos atributos e combinando partições adjacentes (discretização), então transformando o os atributos em binários. Embora este método possa resolver o problema ocasionado por domínio infinito, este causa o problema de *sharp boundary*. Ou seja, ele sub-valoriza ou super-valoriza os elementos próximos aos limites das partições. Entretanto, com a introdução da Teoria *Fuzzy* proposta por ZADEH (1965), foi possível resolver o problema de *sharp boundary*, e vem sendo amplamente utilizada nos domínios de extração de regras de associação.

Em KUOK et al, (1998), foi introduzido o algoritmo para extração de regras de associação de atributos quantitativos utilizando a teoria dos conjuntos Fuzzy.

Nomeado como *Fuzzy Apriori*, a proposta de extração de regras de associação de atributos com valores quantitativos se dividem em duas etapas,

sendo: i) transformar os valores dos atributos quantitativos em termos linguísticos com sua respectiva variável linguística e função de pertinência associada (partições *fuzzy*), e em seguida é feita seleção de um único termo linguístico que possui a máxima cardinalidade escalar<sup>10</sup> mantendo assim o número de regiões fuzzy igual ao número original de atributos; ii) encontrar os *item sets* frequentes à partir de um valor de suporte mínimo, e através de um valor mínimo de confiança, extrair as regras de associação.

Pode-se dizer então que dado um conjunto de dados  $D = \{t_1, t_2, \dots, t_n\}$  com atributos  $I$  e conjuntos fuzzy associado com atributos em  $I$ , deseja-se encontrar relações de interesse potencialmente úteis. A expressão para uma regra de associação fuzzy é dada por:

If  $X = \{x_1, x_2, \dots, x_i\}$  é  $A = \{f_1, f_2, \dots, f_i\}$  então  $Y = \{y_1, y_2, \dots, y_j\}$  é  $B = \{g_1, g_2, \dots, g_j\}$  (5)  
onde

$$f_i \in \{\text{conjuntos fuzzy relacionados ao atributo } x_i\}$$

$$g_j \in \{\text{conjuntos fuzzy relacionados ao atributo } y_j\}$$

e  $X, Y$  são itemsets.  $X$  e  $Y$  são subconjuntos de  $I$  and eles são dissociados o que significa que não compartilham os mesmos atributos.  $A$  e  $B$  contem conjuntos associados com atributos correspondentes em  $X$  e  $Y$ . Como na associação binária, " $X$  é  $A$ " é chamado de antecedente da regra enquanto que " $Y$  é  $B$ " é chamado de consequente da regra. Se a regra é interessante, deve ter valor de suporte suficiente e valor alto confiança.

### 2.6.1 Algoritmo Fuzzy Apriori

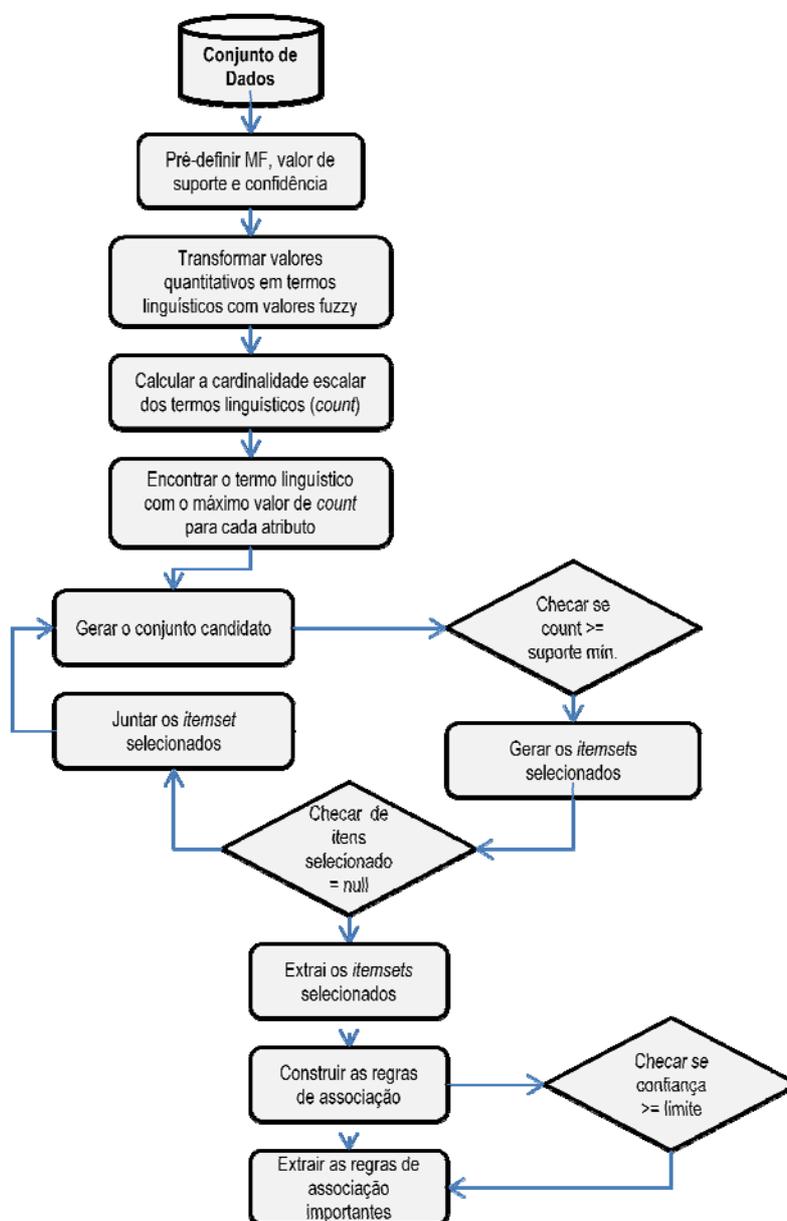
Muitos estudos tem desenvolvido algoritmos para extrair regras de associação fuzzy, e muitos deles são extensão do algoritmo apriori introduzido por KUOK et al, (1998). Como visto na seção anterior a extração de regras de associação de

---

<sup>10</sup> Da um conjunto fuzzy  $\tilde{A}$ , definido em  $X$ , é o somatório dos graus de pertinência de todos os elementos de  $X$  em  $\tilde{A}$ .

atributos com valores quantitativos se dividem em duas etapas, sendo: i) no primeiro passo, *fuzzy* frequente 1 - conjunto de itens  $L_1$  são gerados através do *scaneamento* da base de dados. Juntando-se  $L_1$  a ele mesmo, é possível formar novos candidatos de conjunto de itens fuzzy  $C_2$ . De acordo com o princípio Apriori  $C_2$  é reduzido para formar  $L_2$ , conseqüentemente gerando  $L_3, \dots, L_k$ . O algoritmo para extrair regras de associação fuzzy a partir de conjunto de itens fuzzy é similar a extração de regras de associação Booleana. O fluxograma da Figura 34. mostra o fluxo básico do algoritmo clássico Fuzzy Apriori.

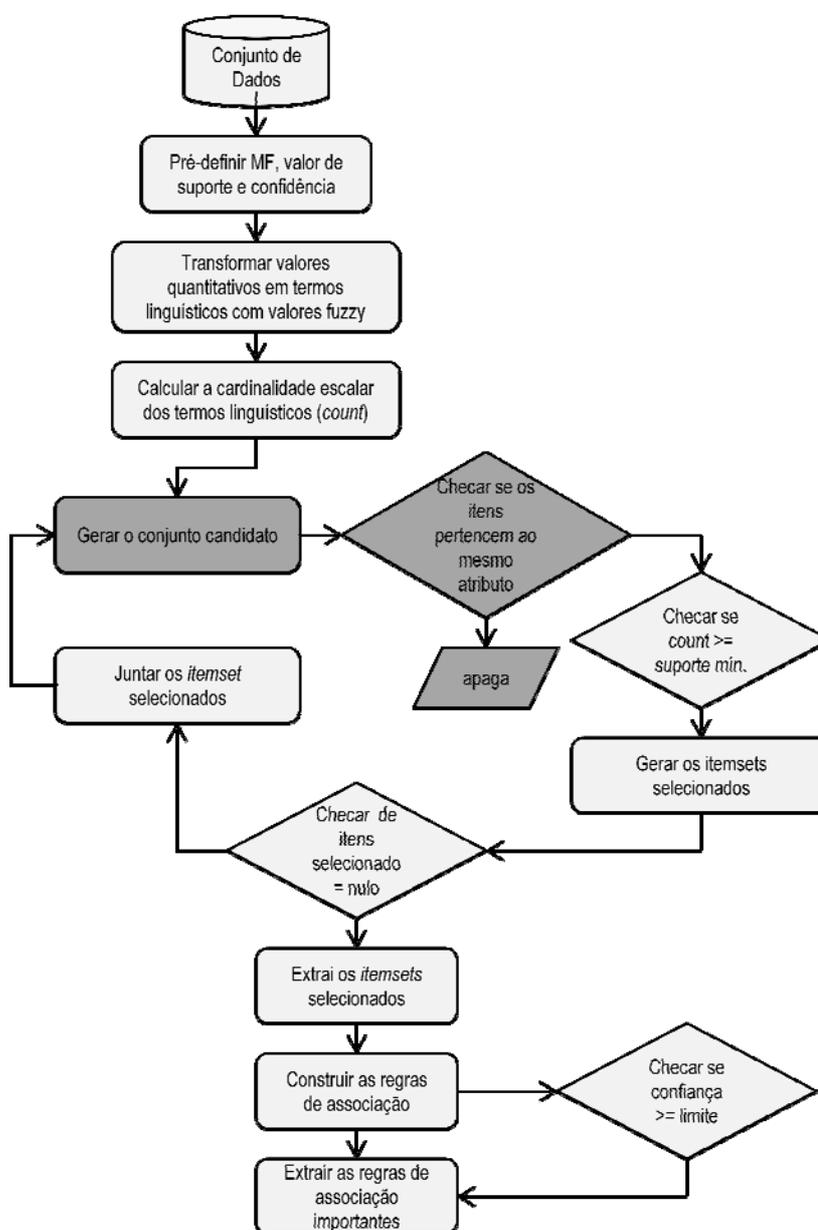
Figura 34 – Fluxograma Algoritmo Fuzzy Apriori Clássico



Fonte: HONG et al., (2001),

O algoritmo fuzzy apriori utilizado nos experimentos (FuzzyApriori-A) foi introduzido por HONG et al, (2001). O autor baseou-se numa pequena variação do fuzzy apriori clássico, que segundo sua proposta todos os termos linguístico são usados, mas com a condição de que termos linguísticos pertencentes ao mesmo atributo não podem pertencer ao mesmo conjunto de itens. A diferença entre eles é observada na área mais escura da Figura 35.

Figura 35 – Fluxograma Algoritmo Fuzzy Apriori-A.



Fonte: HONG et al., (2001),

### 2.6.2 Exemplo de aplicação - Quantitativo / Fuzzy

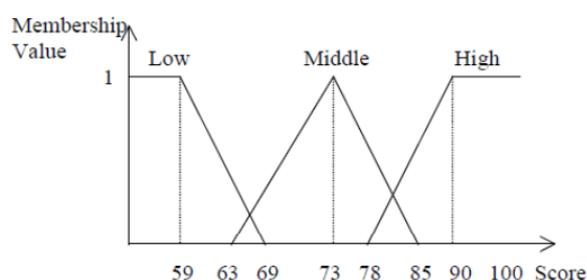
Nesta sessão será apresentado um exemplo clássico para ilustrar a aplicação do algoritmo *Fuzzy Apriori*, onde será utilizado um conjunto de dados com atributos de valores quantitativos. O exemplo segundo HONG et. al, (2004) aplica o algoritmo *Fuzzy Apriori* proposto por KUOK et al, (1998), com o objetivo de reduzir o tempo computacional. O algoritmo é aplicado para extrair regras de associação para uma grade de cursos de acordo com dados históricos relativos às notas dos estudantes nos cursos. O conjunto de dados inclui 10 instâncias, como é mostrado na Tabela 5.

Tabela 5 – Exemplo do conjunto de dados.

Caso #	ST	DB	OOP	DS	MIS
1	86.0	77.0	86.0	71.0	68.0
2	61.0	79.0	89.0	77.0	80.0
3	84.0	89.0	86.0	79.0	89.0
4	73.0	86.0	79.0	84.0	62.0
5	70.0	89.0	87.0	72.0	79.0
6	65.0	77.0	86.0	61.0	87.0
7	67.0	87.0	75.0	71.0	80.0
8	86.0	63.0	64.0	84.0	86.0
9	75.0	65.0	79.0	87.0	88.0
10	79.0	63.0	63.0	85.0	89.0

Cada caso consiste de cinco notas: Estatística (ST), Banco de Dados (DB), Programação orientada a objeto (OOP), Estrutura de Dados (DS), e Sistema de Gerenciamento de Informação (MIS). As funções de pertinência triangular são usadas para representar o conjunto *fuzzy* para as notas dos cursos (Figura 36).

Figura 36 – Função de pertinência usada no exemplo.



Fonte: HONG, et. al., 2004.

A partir da Tabela 6, cada atributo tem 3 regiões *fuzzy*: *Low*, *Middle* e *High*. Então, os 3 valores de pertinência *fuzzy* são atribuídos para cada nota dos cursos de acordo com a função de pertinência predefinida. Para as transações de dados da Tabela xx, a proposta do algoritmo para extração de regras de associação segue os passos abaixo.

**Passo 1:** Transforma os valores quantitativos de cada transação em *fuzzy set* (Tabela xx). O valor que está em negrito é o valor (diferente de zero) que está na região da partição *fuzzy* que possui a cardinalidade escalar de maior o maior valor, e que será visto no Passo x.

Tabela 6 – Conjuntos *fuzzy* transformado à partir da Tabela 5.

Case	ST			DB			OOP			DS			MIS		
	Low	Middle	High	Low	Middle	High	Low	Middle	High	Low	Middle	High	Low	Middle	High
1	0.0	0.0	0.7	0.0	0.7	0.0	0.0	0.0	<b>0.7</b>	0.0	<b>0.8</b>	0.0	0.1	0.5	0.0
2	0.8	0.0	0.0	0.0	0.5	<b>0.1</b>	0.0	0.0	<b>0.9</b>	0.0	<b>0.7</b>	0.0	0.0	0.4	<b>0.2</b>
3	0.0	<b>0.1</b>	0.5	0.0	0.0	<b>0.9</b>	0.0	0.0	<b>0.7</b>	0.0	<b>0.5</b>	0.1	0.0	0.0	<b>0.9</b>
4	0.0	<b>1.0</b>	0.0	0.0	0.0	<b>0.7</b>	0.0	0.5	<b>0.1</b>	0.0	<b>0.1</b>	0.5	0.7	0.0	0.0
5	0.0	<b>0.7</b>	0.0	0.0	0.0	<b>0.9</b>	0.0	0.0	<b>0.8</b>	0.0	<b>0.9</b>	0.0	0.0	0.5	<b>0.1</b>
6	0.4	<b>0.2</b>	0.0	0.0	0.7	0.0	0.0	0.0	<b>0.7</b>	0.8	0.0	0.0	0.0	0.0	<b>0.8</b>
7	0.2	<b>0.4</b>	0.0	0.0	0.0	<b>0.8</b>	0.0	0.8	0.0	0.0	<b>0.8</b>	0.0	0.0	0.4	<b>0.2</b>
8	0.0	0.0	0.7	0.6	0.0	0.0	0.5	0.1	0.0	0.0	<b>0.1</b>	0.5	0.0	0.0	<b>0.7</b>
9	0.0	<b>0.8</b>	0.0	0.4	0.2	0.0	0.0	0.5	<b>0.1</b>	0.0	0.0	0.8	0.0	0.0	<b>0.8</b>
10	0.0	<b>0.5</b>	0.1	0.6	0.0	0.0	0.6	0.0	0.0	0.0	0.0	0.6	0.0	0.0	<b>0.9</b>
Count	1.4	3.7	2.0	1.6	2.1	3.4	1.1	1.9	4.0	0.8	3.9	2.5	0.8	1.8	4.6

**Passo 2:** Construir um conjunto de dados temporário incluindo todos os pares de cada dado (Tabela 7).

Tabela 7 – Conjuntos temporário C1.

Case	Conjunto de <i>Itemsets</i>
1	{(ST_Hight,0.7) , (DB_Middle,0.7) , (OOP_High,0.7) , (DS_Middle,0.8) , (MIS_Low,0.1) , (MIS_Middle,0.5)}
2	{(ST_Low,0.8) , (DB_Middle,0.5) , (DB_High,0.1) , (OOP_High,0.9) , (DS_Middle,0.7) , (MIS_Middle,0.4) , (MIS_High,0.2)}
3	{(ST_Middle,0.1) , (ST_High,0.5) , (DB_High,0.9) , (OOP_High,0.7) , (DS_Middle,0.5) , (DS_High,0.1) , (MIS_High,0.9)}
4	{(ST_Middle,1.0) , (DB_High,0.7) , (OOP_Middle,0.5) , (OOP_High,0.1) , (DS_Middle,0.1) , (DS_High,0.5) , (MIS_Low,0.7)}

5	{(ST_Middle,0.7) , (DB_High,0.9) , (OOP_High,0.8) , (DS_Middle,0.9) , (MIS_Middle,0.5) , (MIS_High,0.1)}
6	{(ST_Low,0.4) , (ST_Middle,0.2) , (DB_Middle,0.7) , (OOP_High,0.7) , (DS_Low,0.8) , (MIS_High,0.8)}
7	{(ST_Low,0.2) , (ST_Middle,0.4) , (DB_High,0.8) , (OOP_Middle,0.8) , (DS_Middle,0.8) , (MIS_Middle,0.4) , (MIS_High,0.2)}
8	{(ST_Hight,0.7) , (DB_Low,0.6) , (OOP_Low,0.5) , (OOP_Middle,0.1) , (DS_Middle,0.1) , (DS_High,0.5) , (MIS_High,0.7)}
9	{(ST_Middle,0.8) , (DB_Low,0.4) , (DB_Middle,0.2) , (OOP_Middle,0.5) , (OOP_High,0.1) , (DS_High,0.8) , (MIS_High,0.8)}
10	{(ST_Middle,0.5) , (ST_High,0.1) , (DB_Low,0.6) , (OOP_Low,0.6) , (DS_High,0.6) , (MIS_High,0.9)}

**Passo 3:** Para cada região (*Low*, *Middle*, *High*) calcular a cardinalidade escalar de todas as transações. Ver *count* na Tabela xx. Tendo ST\_Low como exemplo, a cardinalidade escalar =  $(0.8 + 0.4 + 0.2) = 1.4$ . Repetir esse cálculo para todas as demais regiões (Tabela 8).

Tabela 8 – Conjuntos de 1-itemsets com seus valores *count*.

<i>Itemset</i>	<i>Count</i>
ST_Low	1.4
ST_Middle	3.7
ST_High	2
DB_Low	1.6
DB_Middle	2.1
DB_High	3.4
OOP_Low	1.1
OOP_Middle	1.9
OOP_High	4
DS_Low	0.8
DS_Middle	3.9
DS_High	2.5
MIS_Low	0.8
MIS_Middle	1.8
MIS_High	4.6

**Passo 4:** Para cada região é encontrado o maior valor *count* entre as 3 possíveis regiões. Tendo o curso *ST* como exemplo, o *count* é 1.4 para região *Low*, 3.7 para região *Middle*, e 2.0 para região *High*. Desde que o valor *count* para região *Middle* é a maior entre as 3, esta região então será usada para representar o curso *ST* no processo de mineração posterior. Estes passos são repetidos para as outras regiões. Ficando então para o curso *DB*, *OOP* e *MIS* a região *High* como escolhida., e para os cursos *ST* e *DS* a região *Middle*.

**Passo 5:** Para cada região selecionada no Passo 4, é verificado se o valor *count* se maior ou igual ao valor de suporte mínimo  $\alpha$  predefinido. Este valor de suporte é usualmente atribuído pelo usuário de acordo com a distribuição dos itens frequentes. Para o exemplo será assumido que o valor de suporte mínimo  $\alpha = 2.0$ . Como os valores de *ST\_Middle*, *DB\_Middle*, *OOP\_High*, *DS\_Middle* e *MIS\_High* são maiores que 2.0, esses itens são colocados em  $L_1$  (Tabela 9).

Tabela 9 – Conjuntos de 1-itemsets  $L_1$ .

<i>Itemset</i>	<i>Count</i>
<i>ST_Middle</i>	3.7
<i>DB_High</i>	3.4
<i>OOP_High</i>	4
<i>DS_Middle</i>	3.9
<i>MIS_High</i>	4.6

**Passo 6:** Novo ciclo para gerar novos candidatos. Setar  $r=1$ .

**Passo 7:** Gerar o conjunto candidato  $C_{r+1}$  a partir de  $L_1$ .  $C_2$  é primeiro gerado a partir de  $L_1$ . Ou seja, é gerado todas as possíveis combinações entre os itemsets de  $L_1$  como segue: (*ST\_Middle*, *DB\_High*), (*ST\_Middle*, *OOP\_High*), (*ST\_Middle*, *DS\_Middle*), (*ST\_Middle*, *MIS\_High*), (*DB\_High*, *OOP\_High*), (*DB\_High*, *DS\_Middle*), (*DB\_High*, *MIS\_High*), (*OOP\_High*, *DS\_Middle*), (*OOP\_High*, *MIS\_High*), (*DS\_Middle*, *MIS\_High*).

**Passo 8:** Construir um conjunto vazio temporário  $\bar{C}_{r+1}$ .  $\bar{C}_2$  é então construído.

**Passo 9:** Para cada novo *itemset* candidato formado em  $C_2$  os seguintes sub-passos são seguidos:

- a) Para cada transação, é calculado o valor de pertinência *fuzzy* para o *itemset* à partir de  $\overline{C_1}$ . O operador mínimo é usado para a intersecção (em negrito). Pegando o *itemset* (ST\_Middle, DB\_High) como exemplo, somente os casos 3, 4, 5 e 7 contem ambos os itens *ST\_Middle* e *DB\_High* em  $\overline{C_1}$ . As funções de pertinência *fuzzy* derivadas são mostradas na Tabela 10.

Tabela 10 – O valor de pertinência para (ST\_Middle, DB\_High).

Case	ST_Middle	DB_High	(ST_Middle, DB_High). operador mínimo
1	0.0	0.0	0.0
2	<b>0.0</b>	0.1	0.0
3	<b>0.1</b>	0.9	0.1
4	1.0	<b>0.7</b>	0.7
5	<b>0.7</b>	0.9	0.7
6	0.2	<b>0.0</b>	0.0
7	<b>0.4</b>	0.8	0.4
8	0.0	0.0	0.0
9	0.8	<b>0.0</b>	0.0
10	0.5	<b>0.0</b>	0.0

Os resultados para os demais 2-itemsets pode ser visto a seguir (Tabela 11):

Tabela 11 – Valor de pertinência dos demais 2-itemset.

Case	ST_Middle	OOP_High	Op. Min.	ST_Middle	DS_Middle	Op. Min.	ST_Middle	MIS_High	Op. Min..
1	<b>0.0</b>	0.7	0.0	<b>0.0</b>	0.8	0.0	0.0	0.0	0.0
2	<b>0.0</b>	0.9	0.0	<b>0.0</b>	0.7	0.0	<b>0.0</b>	0.2	0.0
3	<b>0.1</b>	0.7	0.1	<b>0.1</b>	0.5	0.1	<b>0.1</b>	0.9	0.1
4	1.0	<b>0.1</b>	0.1	1.0	<b>0.1</b>	0.1	1.0	<b>0.0</b>	0.0
5	<b>0.7</b>	0.8	0.7	<b>0.7</b>	0.9	0.7	0.7	<b>0.1</b>	0.1
6	<b>0.2</b>	0.7	0.2	0.2	<b>0.0</b>	0.0	<b>0.2</b>	0.8	0.2
7	0.4	<b>0.0</b>	0.0	<b>0.4</b>	0.8	0.4	<b>0.4</b>	0.2	0.2
8	0.0	0.0	0.0	<b>0.0</b>	0.1	0.0	<b>0.0</b>	0.7	0.0
9	0.8	<b>0.1</b>	0.1	0.8	<b>0.0</b>	0.0	0.8	0.8	0.8
10	0.5	<b>0.0</b>	0.0	0.5	<b>0.0</b>	0.0	<b>0.5</b>	0.9	0.5

Case	DB_High	OOP_High	Op. Min.	DB_High	DS_Middle	Op. Min.	DB_High	MIS_High	Op. Min.
1	<b>0.0</b>	0.7	0.0	<b>0.0</b>	0.8	0.0	0.0	0.0	0.0
2	<b>0.1</b>	0.9	0.1	<b>0.1</b>	0.7	0.1	<b>0.1</b>	0.2	0.1
3	0.9	<b>0.7</b>	0.7	0.9	<b>0.5</b>	0.5	0.9	0.9	0.9
4	0.7	<b>0.1</b>	0.1	0.7	<b>0.1</b>	0.1	0.7	<b>0.0</b>	0.0
5	0.9	<b>0.8</b>	0.8	0.9	0.9	0.9	0.9	<b>0.1</b>	0.1
6	<b>0.0</b>	0.7	0.0	0.0	0.0	0.0	<b>0.0</b>	0.8	0.0
7	0.8	<b>0.0</b>	0.0	0.8	0.8	0.8	0.8	<b>0.2</b>	0.2
8	0.0	0.0	0.0	<b>0.0</b>	0.1	0.0	<b>0.0</b>	0.7	0.0
9	<b>0.0</b>	0.1	0.0	0.0	0.0	0.0	<b>0.0</b>	0.8	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0	<b>0.0</b>	0.9	0.0

Case	OOP_High	DS_Middle	Op. Min.	OOP_High	MIS_High	Op. Min.
1	<b>0.7</b>	0.8	0.7	0.7	<b>0.0</b>	0.0
2	0.9	<b>0.7</b>	0.7	0.9	<b>0.2</b>	0.2
3	0.7	<b>0.5</b>	0.5	<b>0.7</b>	0.9	0.7
4	0.1	0.1	0.1	0.1	<b>0.0</b>	0.0
5	<b>0.8</b>	0.9	0.8	0.8	<b>0.1</b>	0.1
6	0.7	<b>0.0</b>	0.0	<b>0.7</b>	0.8	0.7
7	<b>0.0</b>	0.8	0.0	<b>0.0</b>	0.2	0.0
8	<b>0.0</b>	0.1	0.0	<b>0.0</b>	0.7	0.0
9	0.1	<b>0.0</b>	0.0	<b>0.1</b>	0.8	0.1
10	0.0	0.0	0.0	<b>0.0</b>	0.9	0.0

Case	DS_Middle	MIS_High	Op. Min.
1	0.8	0.0	0.0
2	0.7	0.2	0.2
3	0.5	0.9	0.5
4	0.1	0.0	0.0
5	0.9	0.1	0.1
6	0.0	0.8	0.0
7	0.8	0.2	0.2
8	0.1	0.7	0.1
9	0.0	0.8	0.0
10	0.0	0.9	0.0

b) Armazenar os pares de cada case em  $\overline{C2}$  (Tabela 12).

Tabela 12 – Conjunto temporário  $\overline{C_2}$ .

Case	Conjunto de <i>Itemsets</i>
1	{{(OOP_High,DS_Middle,0.8)}
2	{{(DB_High,OOP_High,0.1),(DB_High,DS_Middle,0.1),(DB_High,MIS_High,0.1),(OOP_High,DS_Middle,0.7),(OOP_High,MIS_High,0.2),(DS_Middle,MIS_High,0.2)}
3	{{(ST_Middle,DB_High,0.1),(ST_Middle,OOP_High,0.1),(ST_Middle,DS_Middle,0.1),(ST_Middle,MIS_High,0.1),(DB_High,OOP_High,0.7),(DB_High,DS_Middle,0.5),(DB_High,MIS_High,0.9),(OOP_High,DS_Middle,0.5),(OOP_High,MIS_High,0.7),(DS_Middle,MIS_High,0.5)}
4	{{(ST_Middle,DB_High,0.7),(ST_Middle,OOP_High,0.1),(ST_Middle,DS_Middle,0.1),(DB_High,OOP_High,0.1),(DB_High,DS_Middle,0.1),(OOP_High,DS_Middle,0.1)}
5	{{(ST_Middle,DB_High,0.7),(ST_Middle,OOP_High,0.7),(ST_Middle,DS_Middle,0.7),(ST_Middle,MIS_High,0.1),(DB_High,OOP_High,0.8),(DB_High,DS_Middle,0.9),(DB_High,MIS_High,0.1),(OOP_High,DS_Middle,0.8),(OOP_High,MIS_High,0.1),(DS_Middle,MIS_High,0.1)}
6	{{(ST_Middle,OOP_High,0.2),(ST_Middle,MIS_High,0.2),(OOP_High,MIS_High,0.7)}
7	{{(ST_Middle,DB_High,0.4),(ST_Middle,DS_Middle,0.4),(ST_Middle,MIS_High,0.2),(DB_High,DS_Middle,0.8),(DB_High,MIS_High,0.2),(DS_Middle,MIS_High,0.2)}
8	{{(DS_Middle,MIS_High,0.1)}
9	{{(ST_Middle,OOP_High,0.1),(ST_Middle,MIS_High,0.8),(OOP_High,MIS_High,0.1)}
10	{{(ST_Middle,MIS_High,0.5)}

- c) Habilitar *counts* usando  $\overline{C_2}$ . A cardinalidade escalar (*count*) de cada itemset candidato em  $C_2$  é então calculado (Tabela 13) O valor em negrito será visto no sub-passo seguinte.

Tabela 13 – Counts dos itemsets *fuzzy* em  $C_2$ .

<i>Itemset</i>	<b>Count</b>
(ST_Middle, DB_High)	1.9
(ST_Middle, OOP_High)	1.2
(ST_Middle, DS_Middle)	1.3
(ST_Middle, MIS_High)	1.9

(DB_High, OOP_High)	1.7
<b>(DB_High,DS_Middle)</b>	<b>2.4</b>
(DB_High,MIS_High)	1.3
<b>(OOP_High,DS_Middle)</b>	<b>2.8</b>
(OOP_High, MIS_High)	1.8
(DS_Middle, MIS_High)	1.1

- d) Verificar quais *counts* são maiores ou iguais ao valor de suporte mínimo do suporte predefinido, no exemplo  $\alpha = 2.0$ . (Tabela x.) Dois *itemsets*, (DB\_High, DS\_Middle) e (OOP\_High, DS\_Middle), são mantidos em  $L_2$  (Tabela 14).

Tabela 14 – *Itemset* e seus respectivos *count* em  $L_2$ .

<i>Itemset</i>	<b>Count</b>
(DB_High,DS_Middle)	2.4
(OOP_High,DS_Middle)	2.8

**Passo 10:** Se  $L_{r+1}$  é nulo, então o passo seguinte é executado; caso contrário, habilita-se  $r = r+1 = 2$ . Os passos 7 a 10 são repetidos para encontrar  $L_3$ .  $C_3$  é primeiro gerado à partir de  $L_2$ , e somente o itemset (DB\_High, OOP\_High, DS\_Middle) é formado, e possui seu valor count = 1.5 (Tabela 15). Ou seja, menor que o valor de suporte mínimo predefinido ( $\alpha = 2$ ). Então, este valor não é colocado em  $L_3$ . Desde que  $L_3$  é um conjunto vazio, o passo 11 começa.

Tabela 15 – *Itemset* e seus respectivos *count* em  $L_2$ .

<b>Case</b>	<b>DB_High</b>	<b>OOP_High</b>	<b>DS_Middle</b>	<b>Op. Min.</b>
1	<b>0.0</b>	0.7	0.8	0.0
2	<b>0.1</b>	0.9	0.7	0.1
3	0.9	0.7	<b>0.5</b>	0.5
4	0.7	<b>0.1</b>	<b>0.1</b>	0.1
5	0.9	<b>0.8</b>	0.9	0.8
6	<b>0.0</b>	0.7	<b>0.0</b>	0.0
7	0.8	<b>0.0</b>	0.8	0.0

8	0.0	0.0	0.1	0.0
9	0.0	0.1	0.0	0.0
10	0.0	0.0	0.0	0.0
count				1.5

**Passo 11:** Construção das regras de associação para cada *itemset* que teve o valor *count* superior ou igual ao valor mínimo de suporte predefinido.

- a) As 4 possíveis regras de associação são geradas à partir de 2-itemsets (DB\_High,DS\_Middle) e (OOP\_High,DS\_Middle).

SE *DB = High*, ENTÃO *DS = Middle*  
 SE *DS = Middle*, ENTÃO *DB = High*  
 SE *OOP = High*, ENTÃO *DS = Middle*  
 SE *DS = Middle*, ENTÃO *OOP = High*

- b) Calcular os valores de confiança das regra de associação acima. Assume-se que para o exemplo o valor mínimo de confiança  $\lambda$  é 0.7.

SE *DB = High*, ENTÃO *DS = Middle*

$\frac{DB\_High \cap DS\_Middle}{DB\_High}$	$\frac{2.4}{3.4}$
---	-------------------

0.71

SE *DS = Middle*, ENTÃO *DB = High*

$\frac{DS\_Middle \cap DB\_High}{DS\_Middle}$	$\frac{2.4}{3.9}$
---	-------------------

0.62

SE *OOP = High*, ENTÃO *DS = Middle*

$\frac{OOP\_High \cap DS\_Middle}{OOP\_High}$	$\frac{2.8}{4.0}$
---	-------------------

0.70

SE *DS = Middle*, ENTÃO *OOP = High*

$\frac{DS\_Middle \cap OOP\_High}{DS\_Middle}$	$\frac{2.8}{3.9}$
--	-------------------

0.72

**Passo 12:** Verificar quais os valores de confiança das regras de associação acima são maiores ou iguais ao valor mínimo de confiança  $\lambda$  predefinido. Desde que o valor de confiança é definido como 0.7, as seguintes 3 regras são dadas como resultado de saída ao usuário.

1) Se a nota de *Database* é *high*, então a nota de *Data Structure* é *middle* com valor de confiança de 0.71;

2) Se a nota de *Programação orientada a objeto* é *high*, então a nota de *Data Structure* é *middle* com valor de confiança de 0.70;

3) Se a nota de *Data Structure* é *middle*, então a nota de *Programação orientada a objeto* é *high* com valor de confiança de 0.72;

**Depois do Passo 12:** As 3 regras acima são então as saídas como meta-conhecimento atribuído as dadas transações.

### 2.6.3 Classificação associativa - Algoritmo CFAR

O algoritmo aplicado para classificação das regras de associação foi o classificador CFAR (Classification with Fuzzy Association Rules) (CHEN, 2008).

O problema de *sharp boundary* que ocorre com as regras de associação de atributos com valores quantitativos foi resolvido com a proposta de regras de associação baseada em conjuntos fuzzy. O mesmo problema de *sharp boundary* ocorre com a classificação associativa onde a partição dos atributos quantitativos, na etapa de geração de regras, também é baseada em conjuntos fuzzy.

Como no classificador CBA, a classificação baseada em regras de associação proposta no classificador CFAR, tem o objetivo de construir um classificador através da eliminação de regras de baixa qualidade e assim selecionar um subconjunto de regras de alta confiança para criar um classificador associativo.

### 3 EXPERIMENTOS

Este capítulo aborda a modelagem dos dados utilizado para os experimentos dessa dissertação. Os dados utilizados foram extraídos de uma base de dados utilizada no KDD Cup 1999 Data - Terceira Competição Internacional de Descoberta de Conhecimento e ferramentas de Mineração de Dados que ocorreu em conjunto com a Quinta Conferência Internacional de Descoberta de Conhecimento e Mineração (KDD99). Nesta competição foi escolhido o problema de detecção de intruso e a base de dados serviu como *benchmarking* para a construção de uma rede de detecção de intrusos, onde um modelo preditivo seria capaz de distinguir entre conexões “ruins” invasões ou ataques, e “boas” como conexões normais'.

#### 3.1 Ferramentas de mineração de dados

Importante destacar nesse trabalho a dificuldade de se encontrar *software* com interface GUI (*Graphical User Interface*) para aplicação do algoritmo *Fuzzy Apriori* para a extração de regras de associação. Alguns softwares voltados para mineração de dados aplicados em conjunto de dados com atributos de valor quantitativos, mas com pré-processamento na forma de discretização para aplicação do algoritmo Apriori, foram encontrados. Dentre todos os pesquisados, o WEKA além de ser o mais utilizado e possuir vasta literatura, foi compatível com os experimentos deste trabalho.

Entretanto, para a aplicação do algoritmo *Fuzzy Apriori*, nenhum software havia sido encontrado. Todos os trabalhos relacionados apontavam para utilização de pseudos-códigos, menção de que os experimentos foram feitos na linguagem C+, Java ou no ambiente Matlab, mas efetivamente nenhuma ferramenta GUI que pudesse ser utilizada para a análise de desempenho dos dois algoritmos.

Depois de uma extensa pesquisa, primeiramente foi encontrado o *software Fuzzy Apriori-T (LUCS/KDD)*<sup>11</sup> desenvolvido pela equipe do Dr. Coenen<sup>12</sup> da

---

<sup>11</sup> *Fuzzy Apriori-T*: <http://cgi.csc.liv.ac.uk/~frans/KDD/Software/FuzzyAprioriT/fuzzyAprioriT.html>

<sup>12</sup> Professor e Vice-Chefe do Departamento de Ciência da Computação da Universidade de Liverpool.

Universidade de Liverpool. Entretanto a ferramenta não possui uma interface amigável e os dados para serem lidos possuem formatação própria que exigiu um esforço computacional e de preparação dos dados excessivo. Disponível para download em 7 arquivos do tipo .java, foi necessário fazer a compilação dos arquivos através da ferramenta Eclipse Standard / SDK<sup>13</sup>. Para complementar, o software não realiza a extração de regras de associação tendo como consequente apenas a classe *Ataque* (ataque ou normal). Ou seja, um elevado número de regras de associação é extraído, e para fazer a filtragem apenas das regras de associação com o consequente *Ataque*, não seria uma tarefa fácil diante do elevado número de registros (instâncias) do conjunto de dados KDD99.

Esta opção de *software* LUCS/KDD ficou em suspenso até que após pesquisas em ferramentas de busca da internet, o *software* KEEL surgiu como uma opção que poderia ser utilizada no trabalho. Como o WEKA, o KEEL é um software que agrega algoritmos de pesquisadores elevando assim a abrangência das aplicações. Como o software LUCS/KDD, o software KEEL também não disponibiliza o algoritmo Apriori gerando regras de associação com apenas o consequente classe *Ataque*. Tendo que optar entre o LUCS/KDD e o KEEL, apesar do KEEL também não apresentar usabilidade muito favorável, mas muito mais amigável que o LUCS/KDD, optou-se por este tendo como justificativa sua similaridade com o software WEKA em termos de formato de arquivo, a opção de desenvolvimento de experimentos, e principalmente a disponibilidade de outros algoritmos agregados a ferramenta, tais como os de classificação de regras.

Para o desenvolvimento dos experimentos dessa dissertação, foram então escolhidas as duas ferramentas baseadas em software do tipo *open source*, sendo elas: WEKA (HALL et. al., 2009) e KEEL (ALCALA et. al., 2011).

### 3.1.1 WEKA (Waikato Environment for Knowledge Analysis)

WEKA é um programa open sourced desenvolvido pela Universidade de Waikato na Nova Zelândia no ano de 1999. É uma ferramenta para aplicações em

---

<sup>13</sup> Eclipse: <http://www.eclipse.org/>

mineração de dados desenvolvida em JAVA e que tem agregado estados da arte em máquinas de aprendizagem e algoritmos de mineração de dados que possibilita a extração de conhecimento. Por ser uma ferramenta que possui código aberto e tem fins acadêmicos, estando disponível no sítio do próprio desenvolvedor<sup>14</sup> (WEKA) que pode ser utilizado por qualquer usuário. A equipe de desenvolvimento mantém periodicamente atualizada a versão através de correções e releases, além de manter uma lista de discussões sobre a ferramenta. Grande parte de seus componentes de software são resultados de teses e dissertações de grupos de pesquisa desta universidade. Segundo (Silva, 2004), inicialmente o desenvolvimento do software visava à investigação de técnicas de aprendizagem de máquina, enquanto sua aplicação inicial foi direcionada para a agricultura, uma área chave na economia da Nova Zelândia.

A tela principal da ferramenta é o WEKA GUI Chooser e dispõe dos botões, Simple CLI, Explorer, Experimenter e KnowledgeFlow.

- *Explorer*: é usado para explorar os dados com WEKA dando acesso a todas as facilidade para uso das tarefas de mineração de dados.
- *Experimenter*: permite o usuário criar, analisar, modificar e rodar em larga escala experimentos. Pode ser usado para responder questões do tipo qual o melhor esquema, se houver.
- *Knowledge flow*: tem a mesma função que o *Explorer*. Suporta aprendizagem incremental e utiliza algoritmos incremental para processar dados.
- *Simple CLI*: apenas fornece todas as funcionalidades através de linhas de comando.

A versão utilizada n experimentos foi a versão 3.7.10.

### 3.1.2 KEEL (Knowledge Extraction based on Evolutionary Learning)

A exemplo do software WEKA é um programa *open sourced* (GPLv3) JAVA desenvolvido avaliar diferentes algoritmos para mineração de dados e também foi projetado com objetivo de pesquisa e ensino. É o primeiro software deste tipo para

---

<sup>14</sup> <http://www.cs.waikato.ac.nz/ml/weka/downloading.html>

mineração de dados que contêm código livre baseado em bibliotecas JAVA de Algoritmos de Aprendizagem Evolucionária.

A versão de 03/07/2013 possui uma lista de algoritmos de extração de conhecimentos que foram incluídas e agrupadas em diferentes famílias e que está disponível no sítio do desenvolvedor<sup>15</sup>, tais como: técnicas de seleção (conjunto de treinamento, seleção de atributos, discretização, métodos de imputação de valores ausentes, etc) pré-processamento, algoritmos de aprendizado baseados Inteligência Computacional, incluindo algoritmos de aprendizagem regras evolutivas com base em diferentes abordagens (Pittsburgh , Michigan e IRL, ...), e os modelos híbridos, tais como sistemas *fuzzy* genéticos, redes neurais evolutivas, etc.

A versão utilizada do software foi a *KEEL (Open source version, 2013-04-01)*, pois a última versão *Latest version: KEEL (Open source version, 2013-07-03)*, apresentou problemas de conversão dos arquivos de resultados de extensão *.tra* e *.tst* para *.xml*.

A tela do software KEEL dispõe das opções, *Data Management*, *Experiments*, *Educacional*, *Help* e *Modules*. O ambiente para realização das análises desta dissertação é o *Data Management* e *Experiments*.

A descrição básica de cada uma das opções segue abaixo:

- *Data Management*: permite que os usuários criem novos conjuntos de dados ou criar partições de um arquivo já existente. Além disso é possível visualizar e modificar as informações de conjuntos de dados, com a única restrição de que eles devem estar em formato de arquivo disponível no software.
- *Experiments*: tem o objetivo de projetar experimentos usando uma interface gráfica. O objetivo é a utilização de conjuntos de dados e algoritmos disponíveis para gerar uma estrutura de diretório com todas os arquivos necessários para executar os experimentos planejados.
- *Educacional*: esta opção tem o objetivo de experimentos desejados usando uma interface gráfica e uma execução on-line desses experimentos, sendo ainda possível parar e retornar posteriormente. É possível também visualizar os resultados diretamente no ambiente gráfico. Entretanto, esse módulo do software possui ainda número reduzido de métodos disponíveis.

---

<sup>15</sup> <http://keel.es>

- *Help*: apenas fornece todas as funcionalidades através de linhas de comando.
- *Modules*: inclui novos módulos extensivos de funcionalidades do software KEEL, tais como: *Imbalanced Learning*, *Non-Parametric Statistical Analysis* e *Multiple Instance Learning*.

## 3.2 Descrição dos dados

O arquivo fonte foi extraído do repositório UC Irvine Machine Learning Repository (UCI)<sup>16</sup>, para a realização dos experimentos. Os dados são formados por um conjunto de registros de conexões de pacotes TCP começando e terminando em um tempo definido, fluxo de dados tendo como origem e destino um endereço IP origem e alvo respectivamente e um protocolo bem definido. Os dados incluem um conjunto de dados de treinamento e outro de teste. O de treinamento foi formado após 7 semanas de monitoramento do ambiente de rede e o de teste foi formado após 2 semanas de monitoramento, e onde o mesmo não possui a mesma probabilidade de distribuição do conjunto de treinamento. Desde que a distribuição de probabilidade não é a mesma, em nosso experimento, iremos usar apenas o conjunto de treinamento. O conjunto de dados foi pré-processado em aproximadamente 5.000.000 de registros de conexões como KDD Cup 1999.

### 3.2.1 Características da base de dados KDD99

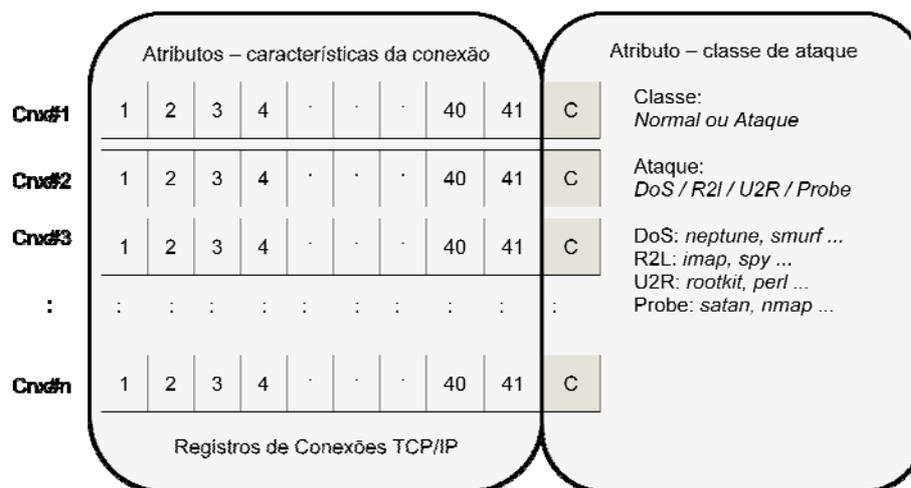
Cada conexão de rede é representada por 41 atributos (KDD99) que descrevem as diferentes características de cada conexão, e ainda, uma classe nomeada por um rótulo, assumido por cada uma delas, como um tipo de ataque específico ou normal. Cada atributo que caracteriza cada uma das conexões é classificado como qualitativo ou quantitativo, onde os qualitativos assumem valores

---

<sup>16</sup> University of California - Irvine - <http://archive.ics.uci.edu/ml/index.html>

0 ou 1, normal ou erro, e os quantitativos variam dentro de uma faixa de valores. A Figura 37 representa o modelo básico dos registros de conexões e a Tabela 16 representa os 41 atributos do conjunto de dados KDD99 com a respectiva categoria.

Figura 37 - Modelo básico dos registros de conexões.



Fonte: Autor, 2013

Tabela 16 – Atributos das conexões de rede.

#	Atributos	Categoria	#	Atributos	Categoria
1	duration	Contínuo	1	is_guest_login	Qualitativo
2	protocol_type	Qualitativo	2	count	Quantitativo
3	service	Qualitativo	3	srv_count	Quantitativo
4	src_byte	Quantitativo	4	serror_rate	Quantitativo
5	dst_byte	Quantitativo	5	srv_serror_rate	Quantitativo
6	flag	Qualitativo	6	rerror_rate	Quantitativo
7	land	Qualitativo	7	srv_rerror_rate	Quantitativo
8	wrong_fragment	Quantitativo	8	same_srv_rate	Quantitativo
9	urgent	Quantitativo	9	diff_srv_rate	Quantitativo
10	hot	Quantitativo	10	srv_diff_host_rate	Quantitativo
11	num_failed_logins	Quantitativo	11	dst_host_count	Quantitativo
12	logged_in	Qualitativo	12	dst_host_srv_count	Quantitativo
13	num_comprised	Quantitativo	13	dst_host_same_srv_count	Quantitativo
14	root_shell	Qualitativo	14	dst_host_diff_srv_rate	Quantitativo
15	su_attempted	Qualitativo	15	dst_host_same_src_port_rate	Quantitativo
16	num_root	Quantitativo	16	dst_host_srv_diff_host_rate	Quantitativo
17	num_file_creations	Quantitativo	17	dst_host_serror_rate	Quantitativo
18	num_shells	Quantitativo	18	dst_host_srv_serror_rate	Quantitativo

19	num_access_files	Quantitativo	19	dst_host_error_rate	Quantitativo
20	num_outbund_cmds	Quantitativo	20	dst_host_srv_error_rate	Quantitativo
21	is_hot_login	Qualitativo			

Esses atributos da conexão são agrupados em três categorias. Na primeira categoria, segundo STOLFO et al. (1999), define como sendo atributos de nível superior que ajudam a distinguir as conexões normais de ataques. São conhecidas como atributos básicos do protocolo TCP/IP. Para a segunda categoria STOLFO et al. (1999) utiliza o domínio do conhecimento para adicionar atributos que parecem ser um comportamento suspeito, como por exemplo, o número de tentativas de login. Esses atributos são chamados de atributos de conteúdo. Fechando a terceira categoria temos os atributos de tráfego calculado usando a teoria da janela de dois segundos. Uma breve descrição de cada uma desses três agrupamentos e dos respectivos atributos o respectivo e feito a seguir:

- a) **Características básicas do TCP/IP:** atributos de nível superior que ajudam a distinguir as conexões normais de ataques. São conhecidas como atributos básicos ao se analisar um fluxo de pacotes de protocolo TCP/IP (Tabela 17)

Tabela 17 – Categoria dos atributos - básicas do TCP/IP.

Atributo	Descrição
duration	Tempo em segundos de conexão
protocol_type	Tipo de protocolo usado (TCP, UDP, etc.)
service	Serviço de rede utilizado (HTTP, Telnet, etc.)
src_byte	Número de bytes enviados da origem ao destino
dst_byte	Número de bytes enviados do destino à origem
flag	Estado da conexão (normal ou erro)
land	"1" se o host e a porta da origem e destino são os mesmos, "0" caso contrário
wrong_fragment	Número de fragmentos com erro
urgent	Número de pacotes com flag urgente habilitado

- b) **Características sugeridas através do conhecimento da área:** utiliza o domínio do conhecimento para adicionar atributos que parecem ser um comportamento suspeito. Ou seja, apenas com o auxílio de especialistas para se chegar a conclusões de que padrões associados a conexões, podem representar um dado tipo de ataque, como por exemplo, o número

de tentativas de login como usuário privilegiado em determinada conexão. Esses atributos são chamados de atributos de conteúdo (Tabela 18).

Tabela 18 – Categoria dos atributos - por conhecimento especialista.

Atributo	Descrição
hot	Número de indicadores “chaves”
num_failed_logins	Número de tentativa de login com falha
logged_in	“1” se o login obteve sucesso, e “0” caso contrário
num_comprised	Número de condições de “comprometimento”
root_shell	“1” se o shell root é obtido; “0” caso contrário
su_attempted	“1” se comando “su root” é atendido; “0” caso contrário
num_root	Número de acessos como root
num_file_creations	Número de operações de criação de arquivos
num_shells	Números de Shell prompts abertos
num_access_files	Número de operações a arquivos de controle de acesso
num_outbund_cmds	Número de comandos externos em uma sessão FTP
is_hot_login	“1” se o login pertence à lista “hot”, “0” caso contrário
is_guest_login	“1” se login é do tipo “guest”, “0” caso contrário

- c) **Características obtidas através de uma janela de 2 segundos:** são atributos de tráfego calculado usando a teoria da janela de dois segundos. Informações importantes referentes a certos ataques somente podem ser obtidas levando-se o tempo em consideração (Tabela 19).

Tabela 19 – Categoria dos atributos - teoria da janela de dois segundos.

Atributo	Descrição
count	Número de conexões para o mesmo host como conexão atual nos últimos 2 segundos.
srv_count	Número de conexões para o mesmo serviço que o usado nesta conexão nos últimos 2 segundos.
serror_rate	% de conexões que tiveram erros do tipo “SYN”.
srv_serror_rate	% de conexões que tiveram erros do tipo “SYN” para este serviço.
rerror_rate	% de conexões que tiveram erros do tipo “REJ”.
srv_rerror_rate	% de conexões que tiveram erros do tipo “REJ” para este serviço.
same_srv_rate	% de conexões ao mesmo serviço.

diff_srv_rate	% de conexões a diferentes serviços
srv_diff_host_rate	% de conexões a diferentes hosts.
dst_host_count	Número de conexões com mesmo host de destino que esta.
dst_host_srv_count	Número de conexões com mesmo host de destino e mesmo serviço que esta.
dst_host_same_srv_count	% de conexões com o mesmo host de destino e mesmo serviço que esta.
dst_host_diff_srv_rate	% de conexões com o mesmo host de destino e serviços diferentes que esta.
dst_host_same_src_port_rate	% de conexões com o mesmo host de destino e a mesma porta de origem que a conexão atual.
dst_host_srv_diff_host_rate	% de conexões para o mesmo serviço vindo de diferentes hosts.
dst_host_serror_rate	% de conexões para o mesmo host que o da conexão atual e que possui um erro S0
dst_host_srv_serror_rate	% de conexões para o mesmo host e serviço que o da conexão atual e que possui um erro S0.
dst_host_rerror_rate	% de conexões para o mesmo host que apresentem flag RST.
dst_host_srv_rerror_rate	% de conexões para o mesmo host e serviço da conexão atual que apresentem flag RST.

Conforme mencionado anteriormente cada conexão de rede é representada por 41 atributos e mais um atributo de classe nomeada por um rótulo, assumido por cada uma delas, como um tipo de ataque específico ou normal. A taxonomia propõe a classificação dos tipos de ataque e intrusões em quatro categorias distintas, sendo elas:

- (DOS) - negação de serviço *denial of service*.
- (Remoto para usuário - R2L) - acesso não autorizado de uma máquina remota, isto é, adivinhar senha *guessing password*.
- (Usuário para usuário - U2R) - acesso não autorizado de usuário para privilégios de super usuário (*root*).
- (PROBING) – reconhecimento.

DoS: é uma classe de ataques contra a disponibilidade de sistemas computacionais. O objetivo do atacante é causar a indisponibilidade do serviço alvo, seja através do consumo excessivo de recursos computacionais como memória, processador ou rede para atendimento a solicitações falsas ou seja através de indisponibilidade total do serviço por exploração de uma falha grave existente no mesmo.

R2L: Esta categoria de ataques e intrusões engloba todos os ataques em que o intruso possui acesso ao sistema como um usuário normal e consegue elevar seu

nível de privilégio para o de um usuário especial (como o usuário root em sistemas Unix ou administrador em outras plataformas).

U2R: respondem a situações em que o intruso possui conectividade com a máquina vítima – sem, entretanto, possuir uma conta de usuário – e explorando alguma vulnerabilidade existente consegue obter acesso local a máquina.

PROBING: esta técnica normalmente é empregada em uma etapa que antecede o ataque ou intrusão. Nesta categoria estão ferramentas e mecanismos automatizados que permitem ao invasor encontrar novas vítimas e conhecer melhor sua vítima. O invasor pode descobrir, por exemplo, quais sistemas estão ativos em determinado momento, quais serviços estão ativos em determinado computador, o fabricante e versão do software que fornece cada um destes serviços dentre outras informações.

### 3.2.2 Distribuição dos registros de conexões e ataques

O conjunto de dados utilizado na competição KDD 99 foi dividido em três bases de dados diferentes denominadas de 10% KDD, *Corrected KDD* e *Whole KDD*. Estas bases referem-se a base de dados utilizada para treinamento, teste e o conjunto original respectivamente. A base de dados 10% KDD, empregada como proposta da fase de treinamento, leva esse nome por representar apenas 10% dos dados originais (*Whole KDD*) e contém 22 tipos de ataques dos 39 presentes na fase de teste. Ou seja, os ataques conhecidos são aqueles presentes na fase de treinamento enquanto que os novos ataques são os ataques adicionais na fase de teste não disponíveis na fase de treinamento. Esta base de dados 10% KDD contém mais exemplos de ataque do que conexão normal e os tipos de ataques não são representados igualmente. Por causa da sua natureza, ataques do tipo DoS contabilizam a maioria dos ataques.

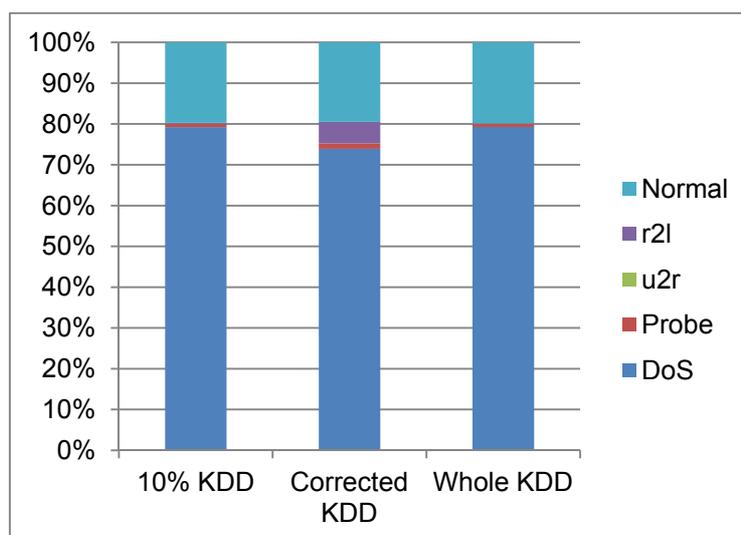
A base de dados KDD10% utilizada inicialmente nesta dissertação consiste de 494.020 registros de conexões dentre os quais 97.277 (19,69%) são normais, 391.458 (79,24%) sofreram o tipo de ataque DOS, 4.107 (0,83%), do tipo Probe, 1.126 (0,23%) do tipo R2L, e 52 (0,01%) do tipo U2R (Tabela 20). A respectiva distribuição dos tipos de ataque na base de dados 10%KDD (treinamento),

*Corrected KDD* (teste) e *Whole KDD* (base original) podem ser visualizados nas Tabela 19 e sua respectiva representação gráfica em Gráfico 1.

Tabela 20 - Características básicas da base de dados KDD99 em termos de número de amostras e percentual por tipo de categoria de ataque.

Categoria Ataque	10% KDD		<i>Corrected</i> KDD		<i>Whole</i> KDD	
	Treinamento	%	Teste	%	Original	%
DoS	391458	79,24%	229853	73,90%	3883370	79,28%
Probe	4107	0,83%	4166	1,34%	41102	0,84%
u2r	52	0,01%	70	0,02%	52	0,001%
r2l	1126	0,23%	16347	5,26%	1126	0,02%
Normal	97277	19,69%	60593	19,48%	972780	19,86%
Total	494020	100%	311029	100%	4898430	100%

Gráfico 1 - Distribuição das características básicas do conjunto de dados KDD99 por tipo de categoria de ataque



Fonte: Autor, 2014.

A lista dos rótulos de classe de ataque (Ataque), com a quantidade de amostras (Qtd amostra), o percentual da quantidade de amostras (%) e das respectivas categoria de tipo de ataque (Categoria) para cada base de dados do KDD99, sendo 10% KDD (treinamento), *Corrected* (teste) e *Whole* KDD (original), podem ser vistas na Tabela 21, Tabela 22 e Tabela 23 respectivamente.

Tabela 21 - Tipos de ataque, quantidade de amostras e classe da categoria da base de dados de treinamento (10% KDD).

Ataque	Qtd amostra	%	Categoria
Smurf	280790	56.84%	dos
Neptune	107201	21.70%	dos
Normal	97277	19.69%	normal
Back	2203	0.45%	dos
Satan	1589	0.32%	probe
Ipsweep	1247	0.25%	probe
portsweep	1040	0.21%	probe
warezclient	1020	0.21%	r2l
Teardrop	979	0.20%	dos
Pod	264	0.05%	dos
Nmap	231	0.05%	probe
guess_passwd	53	0.01%	r2l
buffer_overflow	30	0.01%	u2r
Land	21	0.00%	dos
warezmaster	20	0.00%	r2l
imap	12	0.00%	r2l
rootkit	10	0.00%	u2r
loadmodule	9	0.00%	u2r
ftp_write	8	0.00%	r2l
multihop	7	0.00%	r2l
phf	4	0.00%	r2l
perl	3	0.00%	u2r
spy	2	0.00%	r2l
TOTAL	494020	100%	

Tabela 22 - Tipos de ataque, quantidade de amostras e classe da categoria da base de dados de teste (*Corrected*).

Ataque	Qtd. amostra	%	Categoria
smurf	164091	52,75746%	DOS
normal	60593	19,48146%	Normal
neptune	58001	18,64810%	DOS
snmpgetattack	7741	2,48884%	r2l
mailbomb	5000	1,60757%	DOS
guess_passwd	4367	1,40405%	r2l
snmpguess	2406	0,77356%	r2l
satan	1633	0,52503%	Probing
warezmaster	1602	0,51506%	r2l
back	1098	0,35302%	DOS
mscan	1053	0,33855%	Probing
apache2	794	0,25528%	DOS
processatable	759	0,24403%	DOS

saint	736	0,23663%	Probing
portsweep	354	0,11382%	Probing
ipsweep	306	0,09838%	Probing
httptunnel	158	0,05080%	u2r
pod	87	0,02797%	DOS
nmap	84	0,02701%	Probing
buffer_overflow	22	0,00707%	u2r
multihop	18	0,00579%	r2l
named	17	0,00547%	r2l
sendmail	17	0,00547%	r2l
ps	16	0,00514%	u2r
rootkit	13	0,00418%	u2r
xterm	13	0,00418%	u2r
teardrop	12	0,00386%	DOS
land	9	0,00289%	DOS
xlock	9	0,00289%	r2l
xsnoop	4	0,00129%	r2l
ftp_write	3	0,00096%	r2l
udpstorm	2	0,00064%	DOS
phf	2	0,00064%	r2l
loadmodule	2	0,00064%	u2r
perl	2	0,00064%	u2r
sqlattack	2	0,00064%	u2r
worm	2	0,00064%	r2l
imap	1	0,00032%	r2l
TOTAL	311029	100%	

Tabela 23 - Tipos de ataque, quantidade de amostras e classe da categoria da base de dados original (Whole KDD).

Ataque	Qtd amostra	%	Categoria
smurf	2807886	57,32219%	DOS
neptune	1072017	21,88492%	DOS
normal	972780	19,85902%	Normal
satan	15892	0,32443%	Probing
ipsweep	12481	0,25480%	Probing
portsweep	10413	0,21258%	Probing
nmap	2316	0,04728%	Probing
back	2203	0,04497%	DOS
warezclient	1020	0,02082%	r2l
teardrop	979	0,01999%	DOS
pod	264	0,00539%	DOS
guess_passwd	53	0,00108%	r2l
buffer_overflow	30	0,00061%	u2r

land	21	0,00043%	DOS
warezmaster	20	0,00041%	r2l
imap	12	0,00024%	r2l
rootkit	10	0,00020%	u2r
loadmodule	9	0,00018%	u2r
ftp_write	8	0,00016%	r2l
multihop	5	0,00010%	r2l
phf	4	0,00008%	r2l
perl	3	0,00006%	u2r
spy	2	0,00004%	r2l
TOTAL	4898428	100%	

### 3.2.3 Reamostragem do conjunto de dados

Um aspecto que tem grande impacto em problemas de mineração de regras de associação e classificação é a desigualdade na distribuição dos padrões entre os grupos, que tende a aparecer quando as informações associadas a determinadas classes são mais difíceis de se obter. Pode-se observar esse comportamento, por exemplo, em um estudo sobre uma doença rara em uma certa população. A proporção de pessoas doentes encontradas é muito menor que a proporção de pessoas saudáveis. Em situação com problemas dessa natureza, onde os números de exemplos entre as classes no conjunto de treinamento variam significativamente, algoritmos de aprendizagem tradicionais têm apresentado dificuldade em distinguir entre os vários grupos. Em geral, a tendência é favorecer as classes com maior probabilidade de ocorrência, resultando em baixas taxas de reconhecimento para os grupos minoritários. O problema de classes desbalanceadas tendem a aparecer por que principalmente os algoritmos tradicionais assumem diferentes erros como igualmente importantes, supondo que as distribuições são relativamente equilibradas

No caso do conjunto de dados KDD99, pode ser visto anteriormente na Tabela 9 este desbalanceamento de dados. A categoria de ataque DoS possui no conjunto de dados de treinamento mais de 79,24% dos tipos de ataque, 1,07% os demais ataques e 19,69% são os registros sem ataque ditos normais. Como o objetivo do trabalho é fazer uma análise comparativa de algoritmo aplicado na mineração de regras de associação e posterior classificação, buscou-se reduzir a

amostra de registros (instâncias) desconsiderando aqueles inexpressivos. Para o desenvolvimento do trabalho foi então escolhido um novo conjunto de dados denominado 10%KDD/DoS que se comparado com o conjunto de dados de treinamento 10%KDD original, a redução de amostras é menor que 1 % (Tabela 24). Ou seja, é uma redução insignificante e que para a proposta dessa dissertação ainda torna os experimentos com alto custo computacional.

Tabela 24 - Conjunto de dados 10%KDD x 10%KDD/DoS.

Categoria Ataque	Qtd amostra - 10% KDD		Qtd amostra - 10% KDD - DoS	
DoS	391458	79.24%	391458	80.10%
Probe	4107	0.83%	0	0.00%
u2r	52	0.01%	0	0.00%
r2l	1126	0.23%	0	0.00%
Normal	97277	19.69%	97277	19.90%
Total	494020	100%	488735	100.0%

Outro ponto a ser observado neste conjunto de dados KDD/DoS é o mesmo problema de desbalanceamento de dados quanto a distribuição individual dos tipos de ataques da categoria DoS. Os ataques *back*, *teardrop*, *pod* e *land* juntos representam 0.7% do total dos ataques (Tabela 25). Percentual bem inferior aos 79.38% dos demais ataques smurf e neptune, o que apontou para uma reamostragem do conjunto DoS de forma a tratar este desbalanceamento de dados.

Tabela 25 - Conjunto de dados 10%KDD/DoS.

Ataque	Qtd amostra	%	Categoria
Smurf	280790	57.45%	dos
Neptune	107201	21.93%	dos
Normal	97277	19.90%	normal
Back	2203	0.45%	dos
Teardrop	979	0.20%	dos
Pod	264	0.05%	dos
Land	21	0.004%	dos
TOTAL	488735	100%	

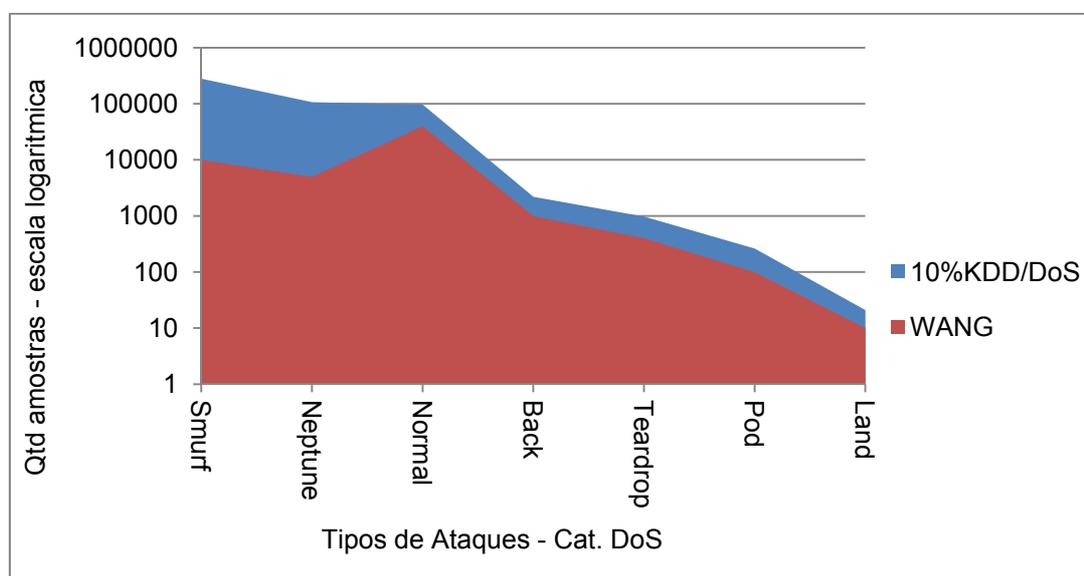
Apesar da tarefa de reamostragem se dar anterior a etapa de seleção de atributos na etapa de pré-processamento de mineração de dados, a mesma se deu após a escolha do trabalho segundo WANG et. al. (2008) para seleção de atributos

relevantes que será visto no tópico seguinte. De maneira a tornar os dados coerentes desta dissertação, a exemplo da seleção dos atributos relevantes, a reamostragem do conjunto de dados DoS, segundo WANG et. al. (2008), também foi utilizada (Tabela 26) (Gráfico 2).

Tabela 26 - Reamostragem segundo WANG.

Ataque	10%KDD/DoS	%	WANG	%
Smurf	280790	57.45%	10000	17.70%
Neptune	107201	21.93%	5000	8.85%
Normal	97277	19.90%	40000	70.78%
Back	2203	0.45%	1000	1.77%
Teardrop	979	0.20%	400	0.71%
Pod	264	0.05%	100	0.18%
Land	21	0.004%	10	0.02%
TOTAL	488735	100%	56510	100%

Gráfico 2 - Comparação da distribuição da amostragem dos tipos de ataques DoS (10%KDD/DoS x WANG).



Fonte: Autor, 2014.

### 3.2.4 Seleção de atributos relevantes do conjunto de dados

Segundo LIU e MOTODA (1998) o processo de seleção de atributo é definido como a determinação de um subconjunto de atributos que será responsável por generalizar a informação contida na base de dados, isto de acordo com a resposta que se espera obter de tal base de dados.

A Seleção de atributos juntamente com a reamostragem são tarefas da etapa da fase de preparação do processo de Descoberta de Conhecimento em Banco de Dados. Já implícito no nome, o objetivo é escolher um subconjunto de atributos (na literatura também chamada de variáveis), ditos relevantes, a fim de reduzir a dimensão do banco de dados. Assim, reduz-se a complexidade do banco de dados, bem como o tempo de processamento para extrair dele algum conhecimento. Além disso, atributos irrelevantes (que não possuem informação útil para a tarefa), tais como aqueles que possuem valor constante para todos os registros, podem causar ruído no resultado final. É usual também de se encontrar em grandes bases de dados atributos redundantes, que são aqueles que possuem valores iguais ou proporcionais, e isto pode ser evitado com a aplicação de técnicas de Seleção de Atributos.

Segundo KAYACIK et. al. (2005), um dos maiores desafios na detecção de intrusão baseado em rede é a extensa quantidade de dados coletados a partir da rede. Portanto, antes de alimentar com dados um algoritmo de aprendizado de máquina, o tráfego da rede original, sem tratamento, precisa ser reduzido em registros de conexão de alta relevância de maneira que não haja comprometimento da qualidade do processo de mineração dos dados. Cada registro de conexão de alto nível possui uma característica composta por conjunto de atributos. A seleção desses atributos relevantes requer amplo conhecimento de domínio que é fundamental para a eficácia dos algoritmos empregados levando a simplificação do problema, aumento do desempenho e taxas de detecção mais precisas (OLUSOLA et al, 2010).

Pesquisadores aplicam vários procedimentos de análise dos dados a fim de selecionar o conjunto de atributos que maximizam a eficácia das técnicas de mineração de dados. Cada um desses atributos oferece um pedaço valioso de informação do sistema. A extração de atributos pode ser classificada em relação à sua contribuição e de acordo com sua utilização. Ou seja, determinar a característica dos atributos mais relevante que melhor discrimina o comportamento da rede de maneira a determinar a classe de ataque ou normal.

Métodos de seleção de atributo pode ser dividido em filtros e abordagens *wrapper*. Para esta dissertação não será objeto de estudo métodos para seleção de atributos em máquina de aprendizagem aplicado em sistema de detecção de intruso. Primeiramente será apresentado trabalhos realizados na área de seleção de atributos em sistemas de detecção de intruso. Em seguida, selecionar um trabalho de tal forma que a seleção de atributos proposto seja aplicado técnicas de regras de associação em valores quantitativos.

#### 3.2.4.1 Trabalhos selecionados para seleção de atributos num IDS

A pesquisa de trabalhos realizados na área de seleção de atributos em Sistemas de Detecção de Intruso (IDS) foi baseada em relevância e trabalhos correlacionados com a proposta apresentada nesta dissertação que é a extração de regras de associação em valores quantitativos, que no caso utiliza a base de dados de treinamento do KDD99 (10%KDD) como referência. Dentre os trabalhos selecionados, tem-se:

(OLUSOLA et. al., 2010) - *Analysis of KDD'99 Detection Dataset for Selection of Relevance Features.*

O conjunto de grau dependência e índice de dependência de cada classe foram empregados para determinar os atributos mais relevantes para cada classe. Os resultados empíricos mostraram que sete características não foram relevantes na detecção de qualquer classe.

(KAYACIK et. al., 2005) - *Selecting Features for Intrusion Detection: A Feature Relevance Analysis on KDD 99 Intrusion Detection Datasets.*

O índice de ganho de informação (*information gain*) foi empregado para determinar os atributos relevantes para cada classe. Os resultados indicam que as classes de ataque normal, neptune e smurf são altamente relacionados com algum atributo que faz sua classificação se tornar fácil. Desde que essas três classes consomem até 98% dos dados de treinamento, é muito fácil para um algoritmo de máquina de aprendizagem atingir bons resultados.

(YOGENDRA et. al., 2011) - *Intrusion Detection using Supervised Learning with Feature Set Reduction.*

O índice de ganho de informação (*information gain*) foi empregado para determinar os atributos relevantes para cada classe. Dois algoritmos de aprendizagem (NB e BayesNet) foram usados para a tarefa de detecção de intruso e comparado seus desempenhos relativos. Os resultados mostraram que 11 atributos foram selecionados com algo ganho de informação.

(WANG et. al., 2008) - *Towards fast detecting intrusions - using key attributes of network traffic.*

O índice de ganho de informação (*information gain*), wrapper com *Bayesian Network* (BN) e árvore de decisão (C.45) respectivamente foram empregados para determinar os atributos relevantes para classe de ataque DoS. O resultado empírico baseado no conjunto de dados KDD99, bem como a classe de ataque DoS, mostraram que usando somente 10 atributos, a precisão de detecção se mantém a mesma ou as vezes se torna melhor comparada com o uso dos 41 atributos, ambos utilizando classificadores BN e C4.5.

(CHOU et. al., 2008) - *Network Intrusion Detection Design Using Feature Selection of Soft Computing Paradigms.*

Neste trabalho dois algoritmos baseados para seleção de atributos, Correlation Based Feature Selection (CFS) e Fast Correlation-Based Filter (FCBF), e 2 dois algoritmos de aprendizagem de máquina, NB e C4.5, para avaliar a precisão de detecção são usados. Os resultados empíricos indicam que o algoritmo para seleção de atributos, para cada classe de ataque, é capaz de reduzir o tamanho do banco de dados.

O resultado dos trabalhos de WANG e CHOU são segmentados por categoria de ataque DoS, Probe, U2R e R2L. Já os trabalhos de KAYACIK, OLUSOLA e YOGENDRA abrangem todos os tipos de ataque independente de sua categoria. A consolidação dessa pesquisa de trabalhos relacionados a seleção de atributos relevantes pode ser visualizada na Tabela 27.

Tabela 27 - Seleção de atributos relevantes da base de dados KDD99.

#	Atributo	WEI	Chou	Kaya cik	Oluso la	Yogendra
---	----------	-----	------	----------	----------	----------

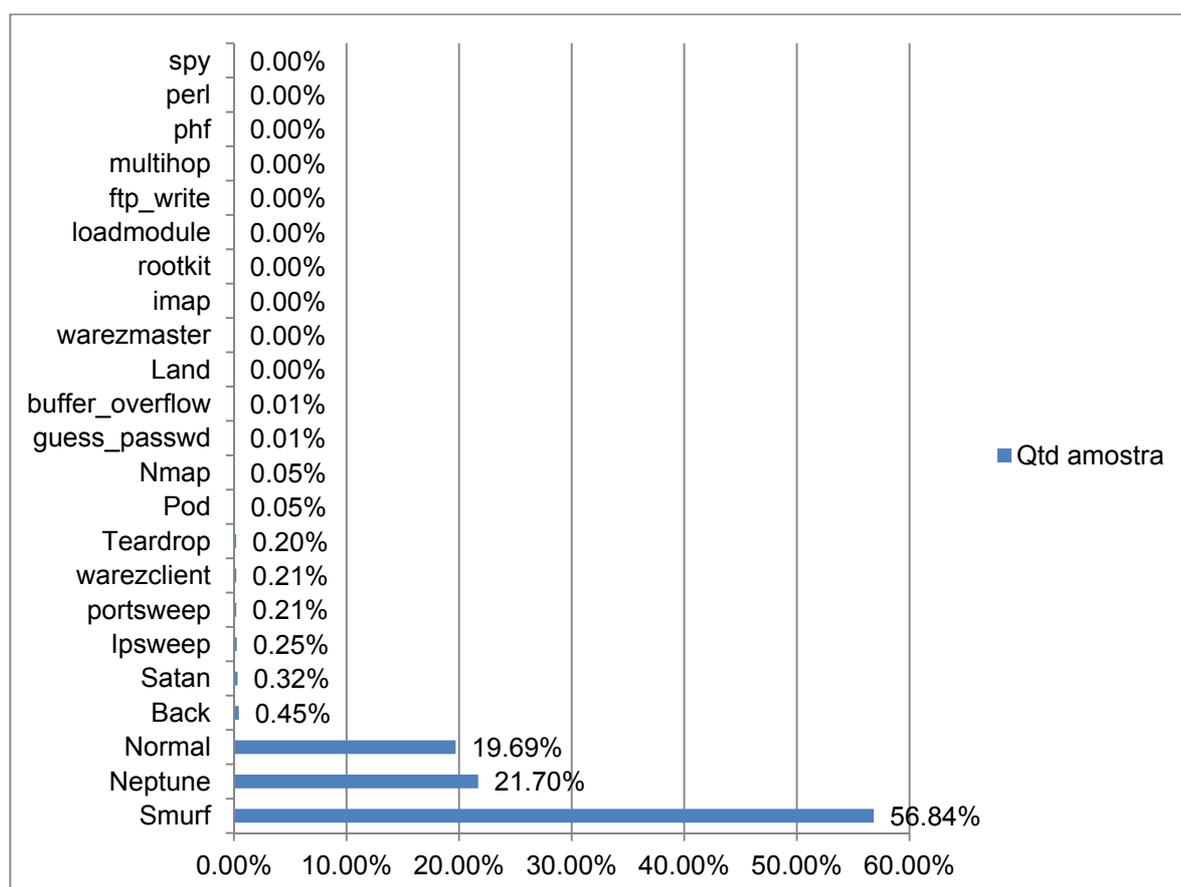
		DoS	Probe	U2R	R2L	DoS	Probe	U2R	R2L			
1	Duration			x	x	x	x	x	x			
2	Protocol Type				x	x	x	x	x			
3	Service	x	x	x	x	x	x	x	x	x	x	x
4	Flag	x	x			x	x		x	x		x
5	SrcBytes	x	x	x	x	x			x	x	x	x
6	DstBytes	x	x	x		x				x	x	x
7	land									x	x	
8	wrong_fragment	x									x	
9	urgente											
10	hot	x			x			x	x			
11	FailedLogins										x	
12	loggd_in			x		x	x					x
13	num_compromised	x			x							
14	root_shell				x						x	
15	su_attempted											
16	NumRoot						x	x		x		
17	num_file_creations											
18	num_shells											
19	num_access_files											
20	num_outbound_cmds											
21	is_host_login											
22	is_guest_login			x					x			
23	Count	x		x		x					x	x
24	SrvCount	x				x					x	
25	serror_rate						x					
26	srv_serror_rate											
27	rerror_rate						x			x		
28	srv_rerror_rate						x				x	
29	same_srv_rate		x				x				x	
30	diff_srv_rate		x				x			x	x	
31	srv_diff_host_rate			x		x						
32	dst_host_count		x	x	x	x						x
33	dst_host_srv_count			x	x					x		
34	dst_hot_same_srv_rate											
35	dst_host_diff_srv_rate		x									x
36	dst_host_same_src_port_rate				x						x	
37	dst_host_srv_diff_host_rate	x				x				x		x
38	dst_host_serror_rate											x
39	dst_host_srv_serror_rate		x								x	x
40	dst_host_rerror_rate		x				x					
41	dst_hot_srv_rerror_rate											
	<b>Atributos selecionados</b>	<b>10</b>	<b>10</b>	<b>10</b>	<b>10</b>	<b>12</b>	<b>12</b>	<b>5</b>	<b>7</b>	<b>10</b>	<b>14</b>	<b>11</b>

Fonte: Autor, 2014.

### 3.2.4.2 Atributos relevantes selecionados para o trabalho

Conforme pode ser observado anteriormente na Tabela 20, existe um grande desequilíbrio entre o quantitativo de amostras para cada tipo de ataque e consequentemente suas respectivas categorias. Observa-se no Gráfico 3 que mais de 98% das amostras estão distribuídas entre as classes normal e os ataques smurf e neptune, ambos da categoria de ataque DoS, e isso indica que essas classes são altamente relacionadas com algum atributo tornando assim de fácil classificação para os algoritmos (KAYACIK et. al., 2005).

Gráfico 3 - Distribuição das amostras dos tipos de ataques do conjunto de dados de treinamento KDD99.



Fonte: O Autor, 2014.

Como o objetivo da dissertação está direcionada na análise de desempenho de algoritmos de extração de regras de associação de valores quantitativos e não em análise de algoritmos para seleção de atributos, buscou-se o melhor modelo de

conjunto de dados para aplicar os experimentos envolvendo os algoritmos Apriori e Fuzzy Apriori. Nesse sentido, optou-se por fazer a seleção de atributos relevantes relacionados aos ataques do tipo DoS já que representam mais de 98% das classes na amostra de treinamento 10%KDD.

A seleção de atributos por tipo de categoria de ataque (DoS, Probe, U2R e R2L) ficaram resumidos aos trabalhos de WANG e CHOU que apresentaram 7 atributos relevantes em comum: *Service*, *Flag*, *Src\_Bytes*, *Dst\_Bytes*, *Count*, *Srv\_Count* e *dst\_host\_srv\_diff\_host\_rate*. Quanto aos demais três atributos relevantes selecionados em cada um dos trabalhos, tiveram ao menos uma indicação em outro trabalho, conforme sinalizado com "x" em negrito na Tabela 28. O atributo *Wrong\_fragment* do trabalho de WANG teve também uma indicação no trabalho segundo OLUSOLA et. al. (2010), e *logged\_ind* e *st\_host\_count* do trabalho de CHOU tiveram também indicação no trabalho segundo YOGENDRA et. al. (2011).

Tabela 28 - Seleção de atributos relevantes em base de dados KDD99

#	Atributo	WEI	Chou	Kayacik	Olusola	Yogendra
		DoS	DoS			
3	Service	x	x	x	x	x
4	Flag	x	x	x		x
5	SrcBytes	x	x	x	x	x
6	DstBytes	x	x	x	x	x
8	wrong_fragment	<b>X</b>			<b>X</b>	
10	hot	<b>X</b>				
12	logged_in		<b>X</b>			<b>X</b>
13	num_compromised	<b>X</b>				
23	Count	x	x		x	x
24	SrvCount	x	x		x	
31	srv_diff_host_rate		<b>X</b>			
32	dst_host_count		<b>X</b>			<b>X</b>
37	dst_host_srv_diff_host_rate	x	x	x		x
<b>Atributos selecionados</b>		<b>10</b>	<b>10</b>			

Fonte: O Autor, 2014.

A avaliação da efetividade dos 10 atributos selecionados, em ambos os trabalhos, comparando-se com os resultados usando todos os 41 atributos pode ser vista na Tabela 29. A taxa de detecção (*Detection Rate*) calculado como

percentagem de intrusões detectadas, e taxa de falso positivo (FPR), calculado como percentagem de conexões normais falsamente classificadas como intrusão, foram os critérios utilizados pelos autores para avaliação.

Tabela 29 - Avaliação da efetividade dos 10 atributos selecionados.

Método	Usando 41 atributos				Usando 10 atributos			
	DR(%)		FPR(%)		DR(%)		FPR(%)	
	WANG	CHOU	WANG	CHOU	WANG	CHOU	WANG	CHOU
BN	98.73	99.12	0.08	0.01	<b>99.88</b>	99.16	<b>0</b>	0.01
C4.5	99.96	99.97	0.15	0.04	99.87	<b>99.97</b>	0.14	<b>0.03</b>

Fonte: O Autor, 2014.

Em geral, resultados de detecção são ideais quando as taxas de detecção são muito altas enquanto que as taxas falso positivo são muito baixas (WANG et. al., 2008). Os resultados mostram que usando os 10 atributos selecionados o método de classificação BN apresentou melhores resultados, tanto para DR quanto para FPR, no modelo segundo WANG. Já para o método de classificação C4.5, tanto para DR quanto para FPR, o modelo segundo CHOU apresentou melhores resultados.

Apesar dos resultados apresentados pelas taxas DR e FPR terem sido favoráveis para ambos os trabalhos, WANG et. al., (2008) utilizou o tradicional método de ranqueamento de ganho de informação (*gain information*) muito utilizado por diversos autores. E ainda, em WANG os testes conduzidos foram separados por tipo de categoria de classe de ataque. Ou seja, 4 experimentos individualmente por tipo de ataque foram realizados. Para finalizar, WANG sugere em seu trabalho uma reamostragem da distribuição dos tipos de ataque por categoria, e que será a mesma base de dados utilizada nesta dissertação.

Baseando-se em todas as informações coletadas para definição de quais atributos relevantes serão selecionados, optou-se pelo conjunto de atributos sugerido segundo WANG et. al. (2008). A distribuição dos valores dos atributos selecionados no conjunto de dados 10%KDD é apresentado na Tabela 30.

Tabela 30 – Distribuição dos valores dos atributos em 10%KDD.

#	ATRIBUTO	VARIAÇÃO	QTDS
---	----------	----------	------

3	Service	nominal	66
4	Flag	nominal	11
5	SrcBytes	0 - 693375640	3300
6	DstBytes	0 - 5155468	10725
8	wrong_fragment	0 - 3	3
10	hot	0 -30	22
13	num_compromised	0 - 884	23
23	Count	0 - 511	490
24	SrvCount	0 - 511	470
37	dst_host_srv_diff_host_rate	0 - 1	65

### 3.2.5 Conjunto de dados KDD99/DoS pré-processado

A distribuição final do conjunto de dados 10%KDD/DoS após as tarefas de reamostragem e seleção de atributos relevantes é apresentado na Tabela 31.

Tabela 31 – Distribuição dos valores dos atributos em 10%KDD/DoS.

#	ATRIBUTO	CATEGORIA	VARIAÇÃO	QTDS
3	Service	nominal	-	16
4	Flag	nominal	-	9
5	SrcBytes	quantitativo	0 - 2194619	2386
6	DstBytes	quantitativo	0 - 2432884	7120
8	wrong_fragment	quantitativo	0 - 3	3
10	hot	quantitativo	0 -30	17
13	num_compromised	quantitativo	0 - 884	10
23	Count	quantitativo	0 - 511	397
24	SrvCount	quantitativo	0 - 511	346
37	dst_host_srv_diff_host_rate	quantitativo	0 - 1	58

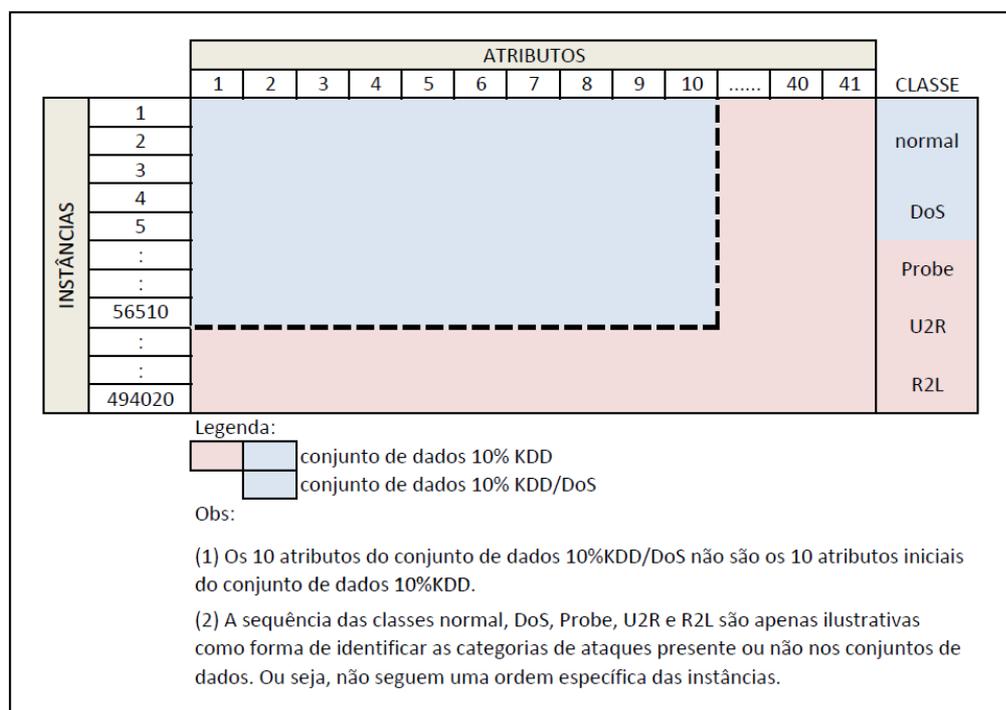
Consolidando as informações referente ao conjunto de dados KDD99/DoS que será utilizado no desenvolvimento dos experimentos, temos:

- Base de Dados de treinamento 10% KDD
  - # registros de conexões: 494.021 registros
  - # atributos: 41

- # classes: categorias de ataques (4) + normal / tipos de ataques (22)
- Pós Pré-processamento - Base de Dados de treinamento 10% KDD/DoS
  - # registros de conexões: 56.510 registros
  - # atributos: 10
  - # classes: os 6 tipos de ataque da categoria DoS foram consolidadas em um único tipo de ataque nomeado de "ataque". Total de classe: 2 (ataque e normal)
  - # *resample* realizado individualmente em cada tipo de ataque (WEKA).
  - # o atributo SERVICE foi analisado no sentido de agrupar os valores que apontavam para a mesma classe de ataque. Dos 66 tipos de SERVICE 42 foram agrupados como *service\_neptune* e 8 como *service\_normal*.

Concluída as tarefas de reamostragem e seleção de atributos relevantes, é possível representar uma comparação entre o conjunto de dados original 10%KDD e o que será utilizado nos experimentos (10%KDD/DoS) (Figura 40).

Figura 38 - Matriz conjunto de dados 10%KDD x 10%KDD/DoS.



Fonte: O Autor, 2014.

### 3.3 Metodologia

O capítulo a seguir tem o objetivo de detalhar todas as etapas realizadas em nossa proposta de trabalho. As etapas aqui consideradas seguirão os tópicos: Pré-processamento (reamostragem e seleção de atributos), transformação dos dados (discretização e conjuntos fuzzy), mineração de dados (extração de regras de associação) e Interpretação e avaliação (classificação) e resultados.

Todos os experimentos foram desenvolvidos utilizando as ferramentas de mineração de dados baseado software *open source*, sendo elas: *Weka* e *Keel* já abordados na seção 3.1. O parâmetro de configuração de memória no *Weka* foi *maxheap=2048M*.

Na Tabela 32 é detalhado etapas do processo de descoberta de conhecimento - KDD (pré-processamento e mineração de dados) e as respectivas tarefas e técnicas aplicadas que serão abordadas neste capítulo.

Tabela 32 - Ferramentas e tarefas utilizadas no desenvolvimento.

Ferramenta	Etapas	#	Tarefa	Técnica
WEKA (3.7.10) 	Pré-processamento	4.1	Reamostragem do conjunto de dados 10%KDD/DoS	Filter/Unsupervised/Instance/Resample
		4.2	Seleção de Atributos	Não se aplica. Tarefa segundo WANG
	Transformação dos Dados	4.3	Discretização dos atributos com valores quantitativos	Filter/Supervised/Attribute/Discretize
	Mineração dos Dados	4.4	Extração de Regras de Associação	Algoritmo Apriori
	Interpretação e Avaliação	4.5	Classificadores de regras	Classifiers/Rules/CAR/JCBA
KEEL (2.0) (2013-04-01) 	Transformação dos Dados	4.6	Particionamento <i>Fuzzy</i> dos atributos quantitativos	Algoritmo <i>Fuzzy</i> Apriori
	Mineração de Dados		Extração de Regras de Associação	Algoritmo <i>Fuzzy</i> Apriori
	Interpretação e Avaliação	4.7	Classificadores de regras de associação	Algoritmo CFAR

### 3.3.1 Pré-processamento

Esta seção aborda dois tópicos da etapa de pré-processamento do processo KDD antes dos experimentos com o algoritmo Apriori, sendo eles: reamostragem e seleção de atributos relevantes.

### 3.3.1.1 Reamostragem

Na proposta de reamostragem do conjunto de dados de treinamento 10%KDD\_DoS segundo WANG et. al. (2008), um percentual de redutor servirá de parâmetro para aplicação no filtro não supervisionado de instância no WEKA. A Tabela 33 mostra para cada um dos tipos de ataque, a quantidade de amostras do conjunto de dados 10%KDD/DoS, a quantidade de amostras segundo a proposta de reamostragem de WANG et. al., (2008) e o respectivo percentual do redutor (% redutor).

Tabela 33 - Arquivos gerados para serem filtrados.

Ataque	Qtd amostras 10%KDD/DoS	Qtd amostras WANG	% redutor
smurf	280790	10000	3.562
neptune	107201	5000	4.665
normal	97277	40000	41.12
back	2203	1000	45.393
teardrop	979	400	40.86
pod	264	100	37.88
land	21	10	47.62
TOTAL	488735	56510	

O filtro de reamostragem selecionado foi Choose / filters / unsupervised / instance / resample. Antes de aplicar o filtro é necessário ajustar parâmetros, sendo eles: noReplacement (*False*), ou seja, com substituição, e em *sampleSizePercent* esse valor é alterado de 100.0 para o % *redutor* respectivo de cada tipo de ataque visto na anteriormente Tabela xx. Os demais parâmetros permanecem inalterados.

Com objetivo de melhorar o desempenho dos algoritmos, reduzindo o número de valores nominais das variáveis categóricas que não acrescentam informação para descoberta de conhecimento, essas variáveis tiveram seu conteúdo modificado. Nesse sentido, verificou-se que o atributo *service*, composto por 65 valores nominais diferentes, possui 42 valores com um único tipo de classe *attack\_type* - *neptune*, e 8 variáveis com o único tipo de classe *attack\_type* - *normal*. Então, os 42 valores nominais foram renomeados para *service\_neptune* e os 8 valores nominais para *service\_normal*.

### 3.3.1.2 Seleção de atributos

Esta tarefa foi amplamente abordada na seção 3.2.4. Os 10 atributos relevantes selecionados para o trabalho, com a respectiva característica de cada atributo, é representado na Tabela 34. A *Categoria* indica a característica de cada atributo (nominal ou quantitativo) onde os atributos quantitativos serão particionados em segmentos discretizados ou em conjuntos *fuzzy*. *Variação* indica a faixa de valores de cada atributo e *QTDS* é a quantidade de valores diferentes. Por exemplo, o atributo FLAG possuiu 9 tipos de valores, sendo eles: *private*, *ecr\_i*, *tim\_i*, *finger*, *telnet*, *http*, *domain*, *service\_neptune*, *time*, *pop\_3*, *other*, *auth*, *ftp*, *ftp\_data*, *smtp*.

Tabela 34 - Atributos relevantes selecionados em 10%KDD\_DoS.

#	ATRIBUTO	CATEGORIA	VARIAÇÃO	QTDS
3	Service	nominal	-	16
4	Flag	nominal	-	9
5	SrcBytes	quantitativo	0 - 2194619	2386
6	DstBytes	quantitativo	0 - 2432884	7120
8	wrong_fragment	quantitativo	0 - 3	3
10	hot	quantitativo	0 -30	17
13	num_compromised	quantitativo	0 - 884	10
23	Count	quantitativo	0 - 511	397
24	SrvCount	quantitativo	0 - 511	346
37	dst_host_srv_diff_host_rate	quantitativo	0 - 1	58

### 3.3.2 Aplicação do Algoritmo APRIORI

As três etapas do processo KDD, transformação dos dados, mineração dos dados e interpretação e avaliação durante o desenvolvimento do algoritmo Apriori, serão abordados à seguir.

#### 3.3.2.1 Transformação dos dados - Discretização

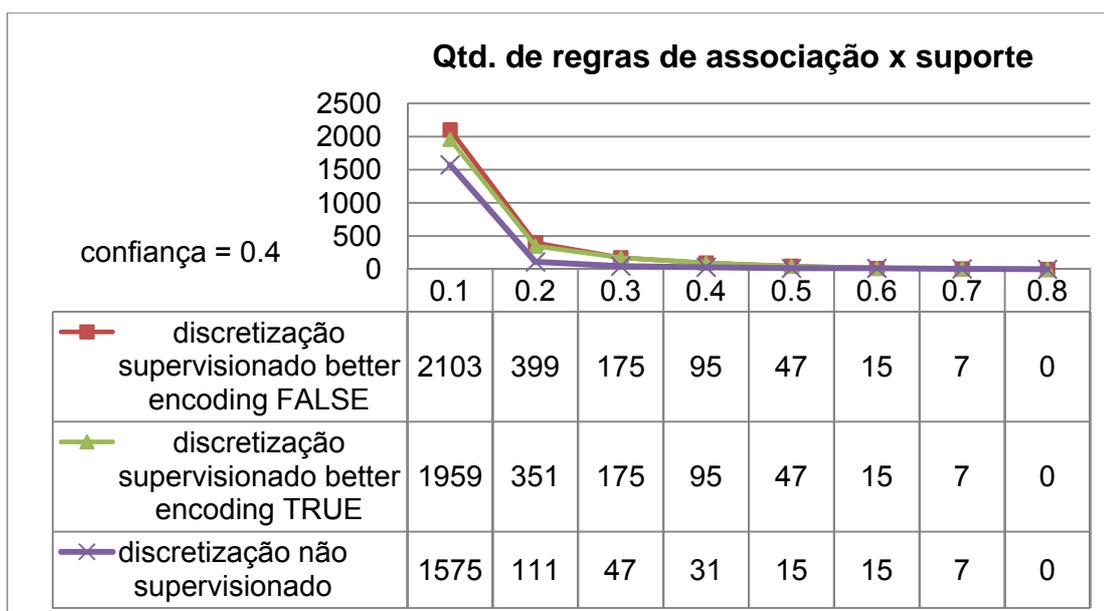
Antes de iniciar a tarefa de extração das regras de associação do conjunto de dados 10\$KDD/DoS é necessário particionar os atributos quantitativos em intervalos e definir em quantas partições cada uma deve ter. O problema potencial em combinar intervalos para atributos quantitativos é o número de regras encontradas que pode ser muito grande (AGRAWAL, et. al., 2006).

O software WEKA possui dois filtros de discretização, sendo um supervisionado e outro não supervisionado. Basicamente o modo não supervisionado possibilita definir o número de partições dos atributos quantitativos e também possui o parâmetro *useEqualFrequency* que define que as partições sejam de mesma frequência. O modo supervisionado não define um valor fixo de partições, mas tem a opção de usar uma codificação mais eficiente, ou não, do ponto de divisão das partições (*useBetterEncoding*).

Antes de definir qual o método de filtragem para discretização (supervisionado ou não supervisionado), um teste de comparação de quantitativo de regras de associação extraídas foi realizado. Utilizou-se como valor mínimo do parâmetro confiança 40% e a faixa de 10% a 80% para o parâmetro suporte. Os parâmetros de configuração considerados foram: i) supervisionado: *useBetterEncoding* (True), ii) supervisionado: *useBetterEncoding* (False) e iii) não supervisionado: *bins* (5) e *useEqualFrequency* (True). Os resultados encontrados mostram que o modo não

supervisionado apresentou quantitativo de regras de associação menor que ambos os modos supervisionados (Figura 39), e será adotado nos experimentos deste capítulo.

Figura 39 - Comparação dos métodos de discretização supervisionado e não supervisionado.



O filtro selecionado para discretização foi *Choose / filters / unsupervised / attribute / discretize*. Antes de aplicar o filtro é necessário ajustar os parâmetros, sendo eles: *bins* (5) e *useEqualFrequency* (True). Os atributos quantitativos foram discretizados em 5 partições de iguais frequências para posterior aplicação do algoritmo Apriori.

### 3.3.2.2 Mineração dos Dados - Extração de regras de associação

Esta seção abordará a extração de regras de associação do conjunto de dados 10%KDD\_DoS utilizando o algoritmo Apriori. Utilizando o arquivo 10%KDD/DoS (discretizado) é selecionado no WEKA o algoritmo Apriori no caminho: *Associate/Associator (Choose) /associations/Apriori*. Os parâmetros utilizados na configuração, são: *car* (TRUE) (classe de regras de associação é habilitada ao invés

generalizar todas as regras de associação), *classindex* (11) (é o index da classe *attack\_type*), *lowerBoundMinSupport* (valor mínimo de suporte predefinido); *metricType* (*confidence* - confiança), *minMetric* (valor mínimo de confiança predefinido) são previamente definidos.

Para os experimentos, será feita uma variação dos parâmetros de suporte e confiança para avaliar o quantitativo de regras de associação geradas em função desses 2 parâmetros. O valor de suporte variou de 5% a 70% (7 valores) e o valor de confiança 40%, 60%, 80% e 90% (4 valores).

### 3.3.2.3 Interpretação e Avaliação - Classificador de regras

Esta seção abordará a classificação das regras de associação geradas no tópico anterior. O algoritmo aplicado para classificação foi o classificador JCBA (implementação JAVA do algoritmo CBA) que trabalha com regras de associação de classe.

Na tela de configuração do classificador JCBA, o parâmetro *carminer* permite a seleção do algoritmo para a extração das regras de associação, e que em nosso trabalho será o *Apriori*. Isso significa que o classificador JCBA possui na verdade 2 tarefas embutidas, sendo a primeira a geração dos itens frequentes e extração das regras de associação (clássico Apriori), e a segunda, a classificação das regras de associação propriamente ditas.

No WEKA, o classificador CBA é selecionado no caminho: *Classify/Classifiers* (Choose) */rules/car/JCBA*. O parâmetro *carminer* (Choose)/*Apriori* é previamente definido e para os experimentos será feita uma variação dos parâmetros de suporte e confiança para classificar as regras de associação geradas em função desses 2 parâmetros. O valor de suporte variou de 5% até 70% (7 valores) e o valor de confiança 40%, 60%, 80% e 90%.

O parâmetro de teste em *Test Option* a opção *Use Training Set* foi selecionada. As demais opções implicaram em uma demora muito significativa com uma diferença no resultado de precisão da regra classificada sem relevância. Por exemplo, selecionando *Cross Validation* (10), em suporte 20% e confiança 50%, o

tempo de espera foi maior quase 800% (15 s contra 119 s) e o percentual de precisão uma diferença de 0.02% (98.53% contra 98.55%).

### 3.3.3 Aplicação do Algoritmo FUZZY APRIORI

Nesta sessão será abordado a aplicação do algoritmo *Fuzzy Apriori* para extração de regras de associação *fuzzy* e interpretação e avaliação do conjunto de dados KDD99. A etapa de transformação dos dados (discretização) que foi realizada anteriormente durante o desenvolvimento do algoritmo Apriori, agora no algoritmo *Fuzzy Apriori* é feita na mesma etapa de mineração dos dados. Ou seja, antes da mineração das regras de associação, inicialmente é realizado o particionamento dos atributos de entrada em conjuntos *fuzzy*.

#### 3.3.3.1 Mineração dos Dados - Extração de regras de associação

O algoritmo *FUZZY APRIORI-A* (HONG, et, al, 2001), disponível no software KEEL, será utilizado para a tarefa de extração de regras de associação *fuzzy*.

O software KEEL não tem a mesma usabilidade<sup>17</sup> que o software WEKA, pois necessita de interações consecutivas do usuário até que se tenha o resultado esperado. Como as regras de associação geradas no KEEL incluem consequentes diferentes da Classe *Ataque*, os arquivos de resultados *.tra* e *.tst* tiveram que ser convertidos para *.xml*, e posteriormente tratados no Excel para que fosse obtido apenas os consequentes de Classe *Ataque*.

No modo *Experiments* do KEEL, o tipo de experimento Unsupervised Learning é o método a ser selecionado e o tipo de partição foi without validation. O algoritmo selecionado na árvore de diretório *Algorithms/Association\_Rules* a opção *FuzzyApriori-A* é aplicado ao conjunto de dados do experimento. O mesmo número

---

<sup>17</sup> Termo usado para definir a facilidade com que as pessoas podem empregar uma ferramenta ou objeto a fim de realizar uma tarefa específica e importante.

de partições escolhida nos experimentos do algoritmo Apriori foram escolhidas para o *Fuzzy Apriori*, ou seja, cinco partições. Para os experimentos, foi feita uma variação dos parâmetros de suporte e confiança, sendo: valor de suporte variou de 5% a 70% (7 valores) e o valor de confiança 40%, 60%, 80% e 90%.

### 3.3.3.2 Interpretação e Avaliação - Classificador de regras

O algoritmo aplicado para classificação foi o classificador CFA-R que trabalha com regras de associação de classe (CHEN, 1998). A exemplo do classificador (LIU, et. al., 1998) (LI, et. al., 2001). visto anteriormente, são regras de associação que possuem exclusivamente uma classe de atributo que é o único atributo consequente na relação *Se / Então*.

No modo *Experiments* da ferramenta, o tipo de experimento selecionado é *Classification* e o tipo de partição *without validation* e o algoritmo de classificação CFAR-C é selecionado dentre as opções de *Associative Classification*. Os parâmetros de configuração do classificador são: o número de partições (5) e os parâmetros de suporte e confiança, sendo: valor de suporte variou de 5% a 70% (7 valores) e o valor de confiança 40%, 60%, 80% e 90%. Ou seja, foram os mesmos parâmetros utilizados no algoritmo apriori. A proposta de manter os mesmos parâmetros de quantidade de partições (discretas ou *fuzzy*) e de suporte e confiança, tem como objetivo igualar as condições previamente determinadas dos experimentos. Assim, os resultados alcançados poderão ser analisados de forma mais homogênea.

## **4 RESULTADOS E ANÁLISE DOS EXPERIMENTOS**

Neste capítulo são apresentados os resultados obtidos e a avaliação dos experimentos de acordo com a metodologia proposta no capítulo anterior. Apresentando os resultados pertinentes ao desempenho de cada algoritmo nos experimentos realizados, permitindo assim uma comparação dos resultados de forma a apontar qual a melhor indicação para o uso de cada um dos algoritmos no ambiente proposto.

### **4.1 Introdução**

Para cada um dos algoritmos aplicados durante a realização dos experimentos, teve como objetivo extrair regras de associação de uma base de dados contendo registros de conexões TCP/IP. Essa base de dados foi utilizada numa competição internacional de descoberta de conhecimento e ferramentas de mineração de dados., e serviram como benchmarking para a construção de um Sistema de Detecção de Intrusos. A proposta foi de criar modelo preditivo capaz de distinguir entre conexões “ruins” invasões ou ataques, e “boas” como conexões normais'

Uma vez gerados os resultados relativos a extração das regras de associação, foi feita uma análise de desempenho das regras geradas através de algoritmos de classificação.

### **4.2 Aplicação do Algoritmo Apriori**

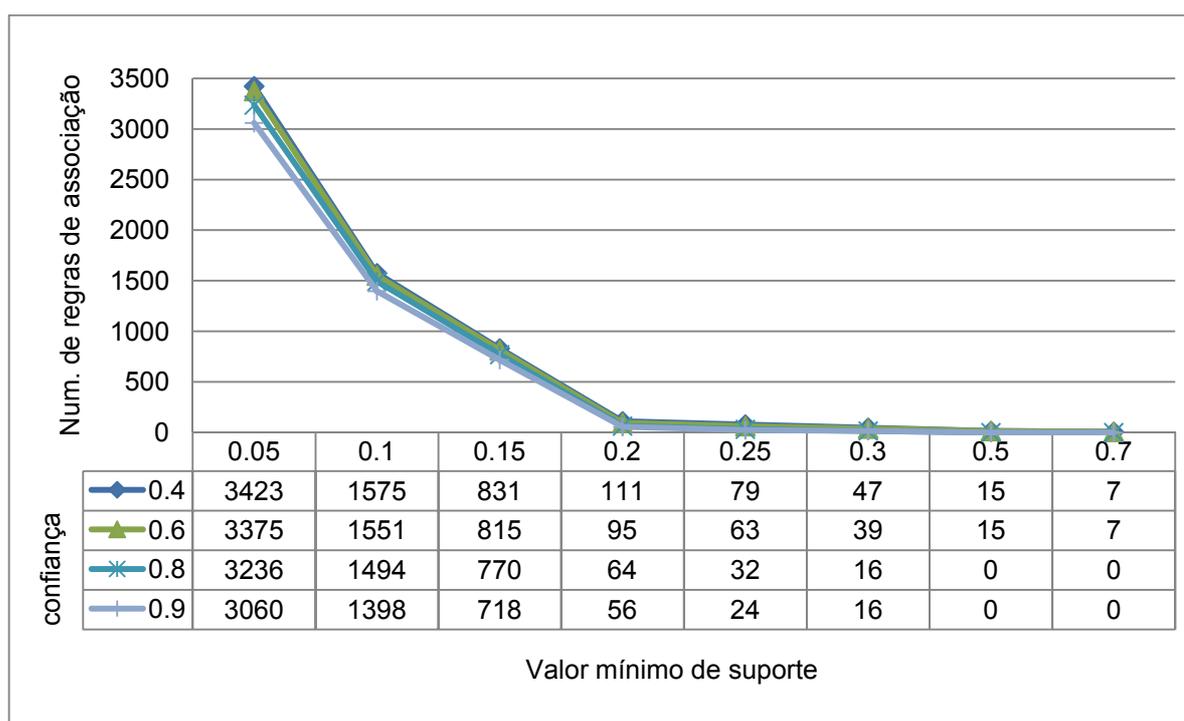
Conforme visto no capítulo dos Experimentos, foi feita uma variação dos parâmetros de suporte mínimo e de confiança mínimo para avaliar o quantitativo de regras de associação geradas e para cálculo da precisão em função desses 2

parâmetros. O valor de suporte variou de 5% a 70% (8 valores) e o valor de confiança 40%, 60%, 80% e 90% (4 valores).

#### 4.2.1 Quantidade de regras de associação extraídas

Os resultados da distribuição do número de regras de associação extraídas variando os valores de suporte mínimo e de confiança mínimo são mostrados na Figura 40.

Figura 40 - Relação entre o número de regras de associação e valor mínimo de suporte para vários valores de confiança mínimo.



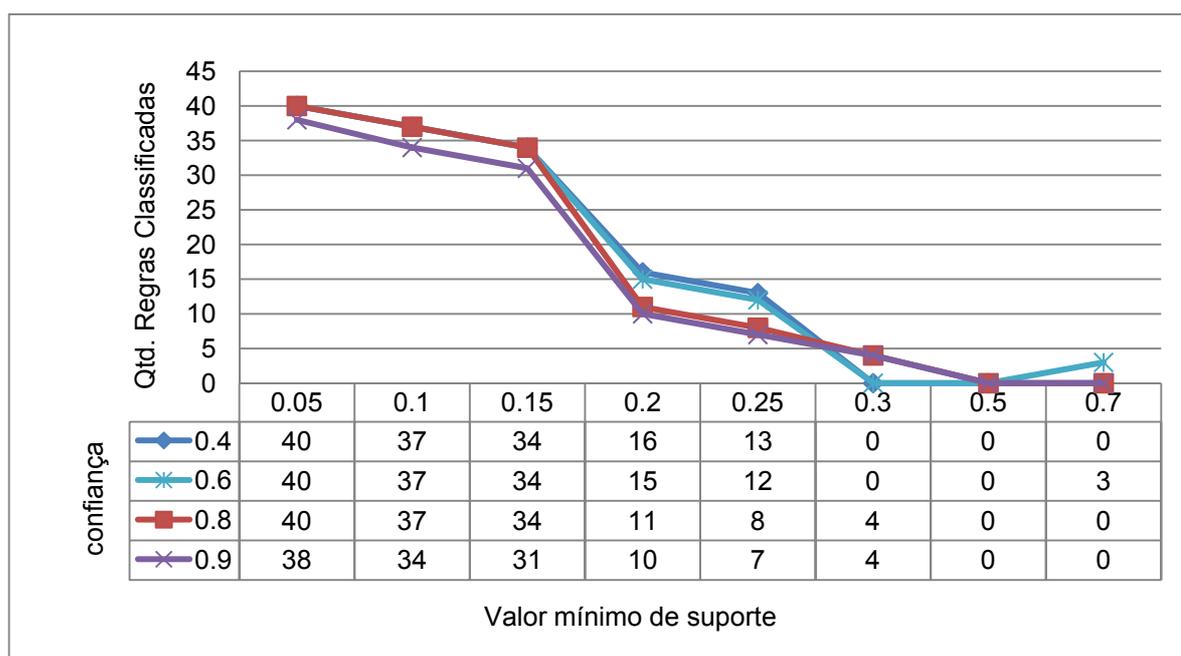
Pela Figura x. pode ser visto que o número de regras de associação decresce conforme o valor de suporte mínimo aumenta. Esta observação é consistente, pois devido a redução do número de *large itemsets* o quantitativo de regras diminui. Além disso, observa-se também que a curva do número de regras de associação com valores mínimos de confiança maiores foi praticamente imperceptível do que o daqueles com valores mínimos de confiança menores, o que significa que o valor

mínimo de suporte não teve um grande efeito sobre o número de regras de associação derivada de valores menores ou maiores de confiança mínimo.

#### 4.2.2 Conjunto de regras de classificação

Experimentos foram feitos para descobrir um pequeno conjunto de regras de associação, denominadas de regras de classificação, para formar um classificador associativo (CBA). O percentual de precisão de acerto deste classificador é mostrado na Figura 41.

Figura 41 - Relação entre o número de regras de classificação e valor mínimo de suporte para vários valores de confiança mínimo.



Pela Figura x. pode ser visto que o número de regras de classificação decresce conforme o valor de suporte mínimo aumenta. Esta observação também é consistente, pois as regras de classificação são pequenos conjuntos de regras gerados à partir do conjunto de regras de associação extraídas anteriormente.

Observa-se também que a curva do número de regras de associação para valores de suporte mínimo entre 5% e 15% não teve nenhum efeito sobre o número

de regras de associação derivada de valores menores ou maiores de confiança mínimo entre 40% e 80%. Apresentando apenas uma redução do número de regras de associação para o valor de confiança mínimo de 90%.

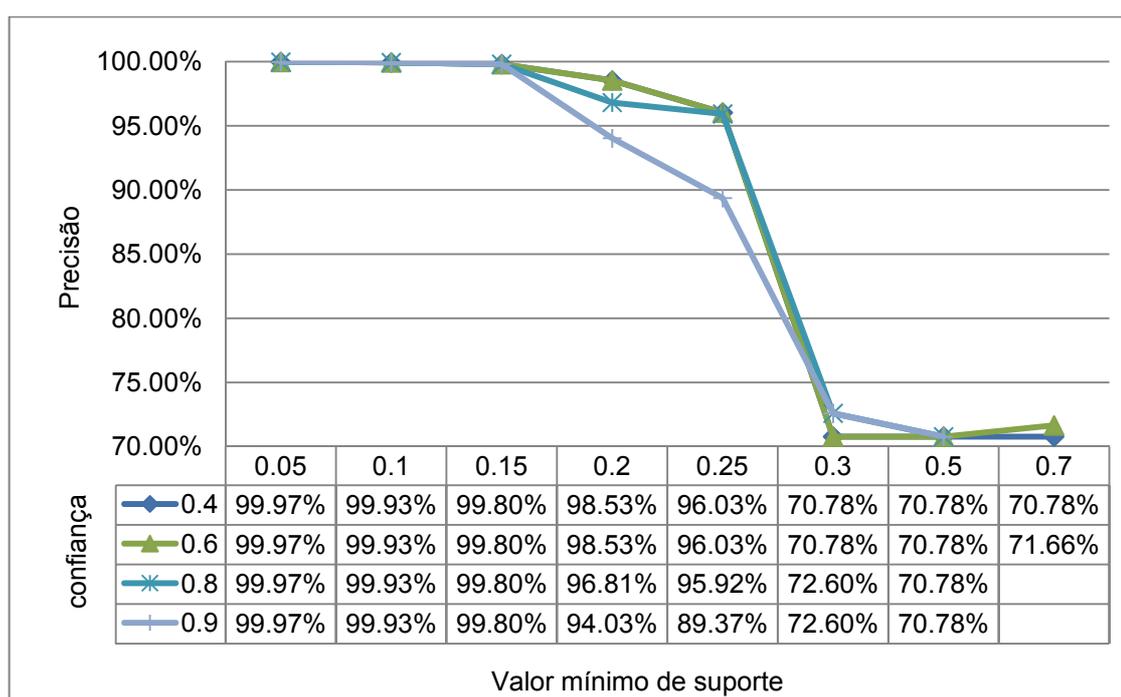
Já nos valores de suporte mínimo de 20% e 25% é que tiveram efeito sobre o número de regras de associação derivada de valores mínimos de confiança

À partir do valor de suporte mínimo de 30% não foram mais geradas regras de classificação, a menos do valor de suporte mínimo de 30% e valor de confiança mínimo de 80% e 90%, que registrou a ocorrência de 4 regras de classificação.

#### 4.2.3 Precisão das regras de associação

Experimentos foram feitos para medir a precisão das regras de associação extraídas, tendo como parâmetro, o conjunto de regras de classificação geradas no tópico anterior. Os resultados seguindo a mesma variação dos valores mínimo de suporte e confiança são mostrados na Figura 42.

Figura 42 - Relação entre o percentual de precisão alcançado e valor mínimo de suporte para vários valores de confiança mínimo.



Na Figura x observa-se que os valores menores de suporte mínimo (5% a 15%) mantêm a precisão em valores altos, próximos de 100%, e permaneceu inalterado independente dos valores mínimo de confiança. Ou seja, o valor mínimo de confiança não teve impacto na precisão das regras extraídas num determinado valor de suporte mínimo (5% a 15%). Esse resultado era esperado, pois a quantidade de regras de associação geradas também se manteve inalterado, com exceção do valor mínimo de confiança de 90%, mas que não teve impacto no resultado da precisão.

Para os valores de suporte mínimo entre 20% e 25% é que apresentaram variação de precisão significativa e decrescente conforme a variação do valor de confiança mínimo.

Os valores de suporte mínimo, à partir de 30%, apresentam percentual de precisão de 70.78%. Esse valor, através da matriz confusão, percebe-se que dos 56510 registros de conexões TCP/IP de possíveis ataques, apenas os ataques da categoria *Normal* são reconhecidos como *Normal* na sua totalidade (40.000 registros). Os ataques da categoria *Ataque* (16510 registros) não são reconhecidos como *Ataque* (Fig. 43). Com exceção apenas dos valores de confiança mínimo de 80% e 90%, é que ainda apresentaram algum percentual de acerto (6.5% - *ataque* e 56.7% - *normal*) (Figura 44).

Figura 43 - Matriz confusão suporte 30% e confiança 60%.

a	b	<-- classified as
0	16510	a = ATAQUE
0	40000	b = NORMAL

Figura 44 - Matriz confusão suporte 80% e confiança 90%.

a	b	<-- classified as
15500	1010	a = ATAQUE
14476	25524	b = NORMAL

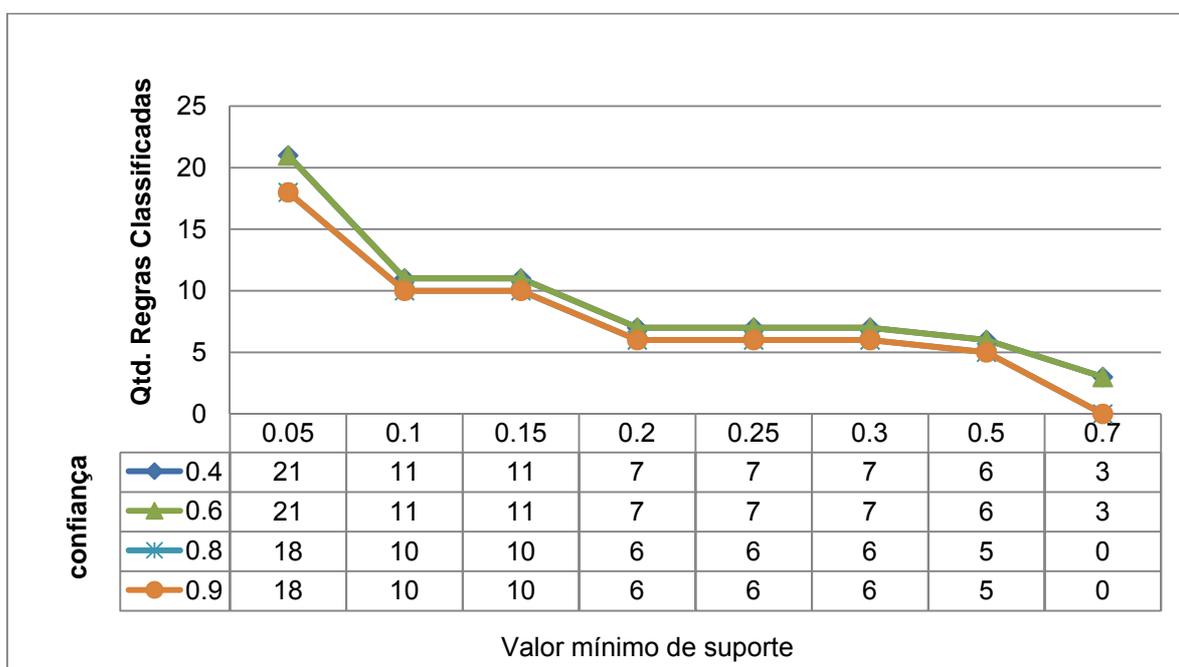


Pela Figura x. pode ser visto que o número de regras de associação decresce suavemente conforme o valor de confiança mínimo aumenta. No entanto, a variação do valor de suporte mínimo entre 5% e 30%, em um determinado valor de confiança mínimo, não teve impacto no quantitativo de regras de associação extraídas. Valores de suporte mínimo superior a 30% indicam redução do número de regras de associação extraídas.

#### 4.3.2 Conjunto de regras de classificação

Experimentos foram feitos para descobrir um pequeno conjunto de regras de associação, denominadas de regras de classificação, para formar um classificador associativo (CFAR). O percentual de precisão de acerto deste classificador é mostrado na Figura 44.

Figura 44 - Relação entre o número de regras de classificação e valor mínimo de suporte para vários valores de confiança mínimo.



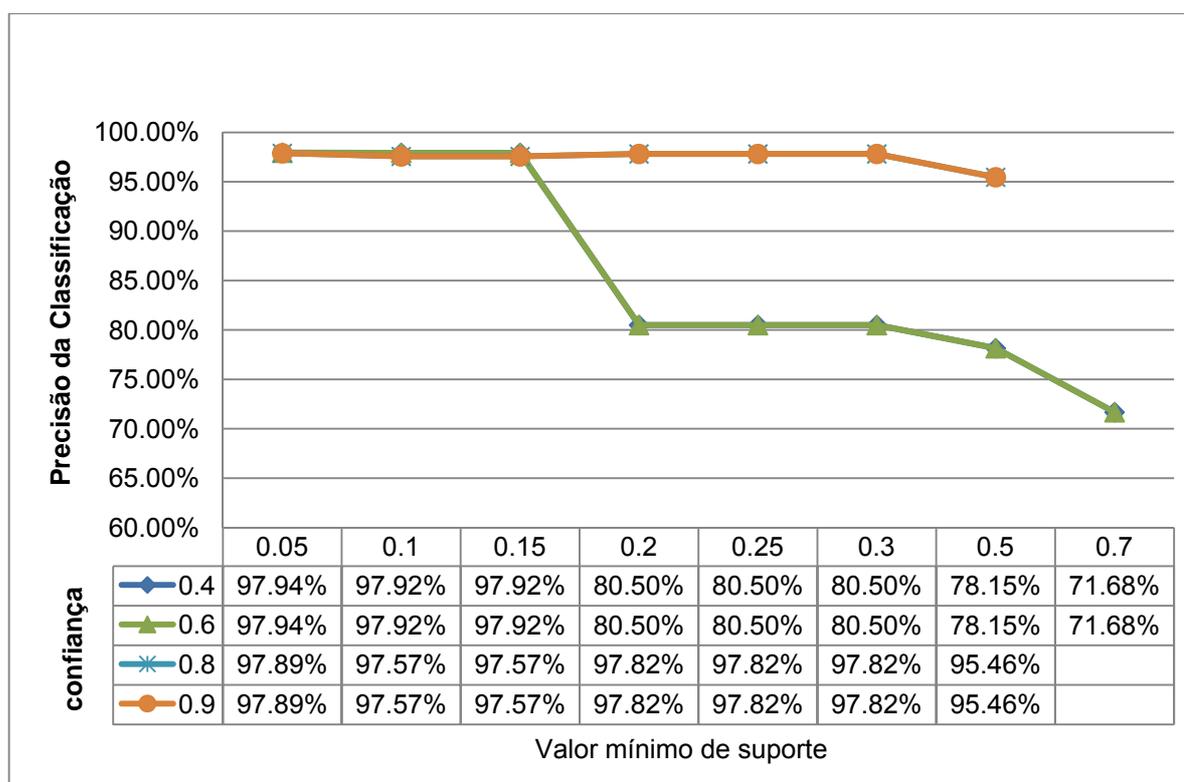
Observa-se que o número de regras de classificação decresce muito suavemente conforme o valor de suporte mínimo aumenta.

Observa-se também que a curva do número de regras de classificação para valores de suporte mínimo de 40% e 60% permaneceu inalterado, porém, diferenciado dos valores de suporte mínimo de 80% e 90% que também permaneceram inalterado.

#### 4.3.3 Precisão das regras de associação

Experimentos foram feitos para medir a precisão das regras de associação extraídas, tendo como parâmetro, o conjunto de regras de classificação geradas no tópico anterior. Os resultados seguindo a mesma variação dos valores mínimo de suporte e confiança são mostrados na Figura 45.

Figura 45 - Relação entre o percentual de precisão alcançado e valor mínimo de suporte para vários valores de confiança mínimo.



Observa-se que os valores menores de suporte mínimo (5% a 15%) mantêm a precisão em valores altos, próximos de 100%. Para valores de suporte mínimo a partir de 20% o percentual de precisão para valores mínimo de suporte de 40% e 60% começam a reduzir. Para valores de suporte mínimo de 80% e 90% o

percentual de precisão se mantêm em valores altos, tendo uma pequena redução à partir do valor mínimo de suporte de 50%.

#### 4.4 Algoritmo Apriori x Fuzzy Apriori

Para fazer uma análise de desempenho dos Algoritmos Apriori e Fuzzy Apriori, foram selecionados valores mínimos de suporte e confiança que apresentassem melhores resultados para essa análise. Na análise comparativa de quantidade de regras de associação foi utilizado, por exemplo, o valor de confiança mínimo de 40%. As curvas dos demais valores de confiança apresentaram distribuição semelhantes (Tabela 35.)

Para análise comparativa de quantidade de regras de classificação foi utilizado, por exemplo, os valores de confiança mínimo de 40% e 90% (Tabelas 35 e 36). As curvas dos demais valores de confiança apresentaram distribuição semelhantes.

Tabela 35 - Dados comparativos dos Algoritmos Apriori e Fuzzy Apriori para valores de confiança mínimo de 40%.

Sup. Min.	Conf. Min.	APRIORI			FUZZY APRIORI		
		Qtd. de Regras de Associação extraídas	% acerto	Qtd. de Regras de Classificação	Qtd de Regras de Associação extraídas	% acerto	Qtd de Regras de Classificação
0.05	0.4	3423	99.97%	40	1023	97.94%	21
0.1	0.4	1575	99.93%	37	1023	97.92%	11
0.15	0.4	831	99.80%	34	1023	97.92%	11
0.2	0.4	111	98.53%	16	1023	80.50%	7
0.25	0.4	79	96.03%	13	1023	80.50%	7
0.3	0.4	47	70.78%	0	1023	80.50%	7
0.5	0.4	15	70.78%	0	511	78.15%	6
0.7	0.4	7	70.78%	0	31	71.68%	3

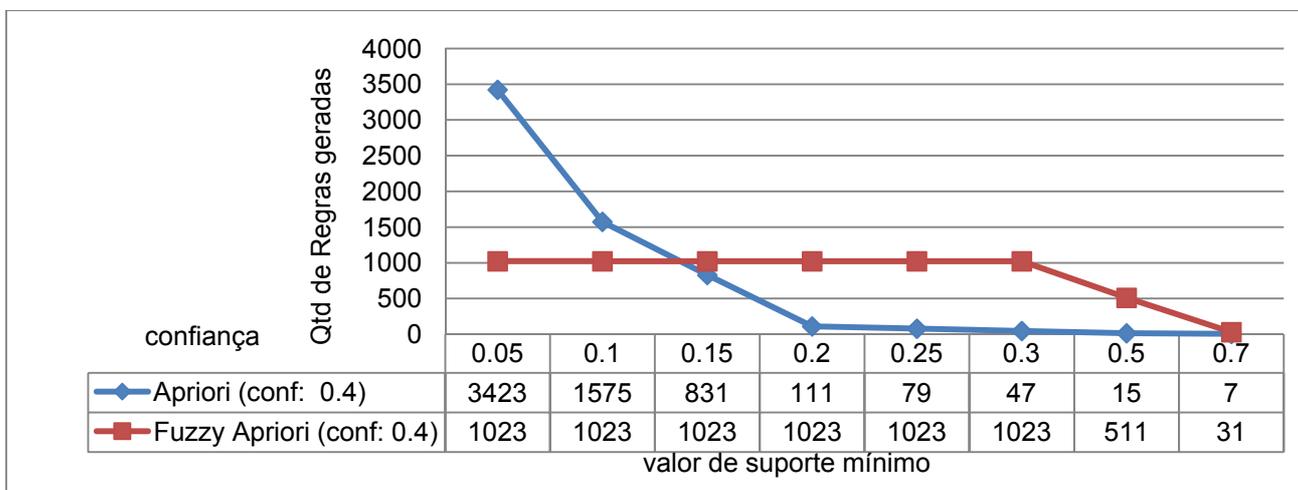
Tabela 36 - Dados comparativos dos Algoritmos Apriori e Fuzzy Apriori para valores de confiança mínimo de 90%.

Sup. Min.	Conf. Min.	APRIORI		FUZZY APRIORI			
		Qtd de Regras de Associação extraídas	% acerto	Qtd de Regras de Classificação	Qtd de Regras de Associação extraídas	% acerto	Qtd de Regras de Classificação
0.05	0.9	3060	99.97%	38	832	97.89%	18
0.1	0.9	1398	99.93%	34	832	97.57%	10
0.15	0.9	718	99.80%	31	832	97.57%	10
0.2	0.9	56	94.03%	10	832	97.82%	6
0.25	0.9	24	89.37%	7	832	97.82%	6
0.3	0.9	16	72.60%	4	832	97.82%	6
0.5	0.9	0	70.78%	0	320	95.46%	5
0.7	0.9	0		0	0		0

#### 4.4.1 Análise comparativa de quantidade de regras de associação

O valor de confiança mínimo de 40% serviu de parâmetro para representar a distribuição da quantidade de regras de associação extraídas dos dois algoritmos (Figura 46).

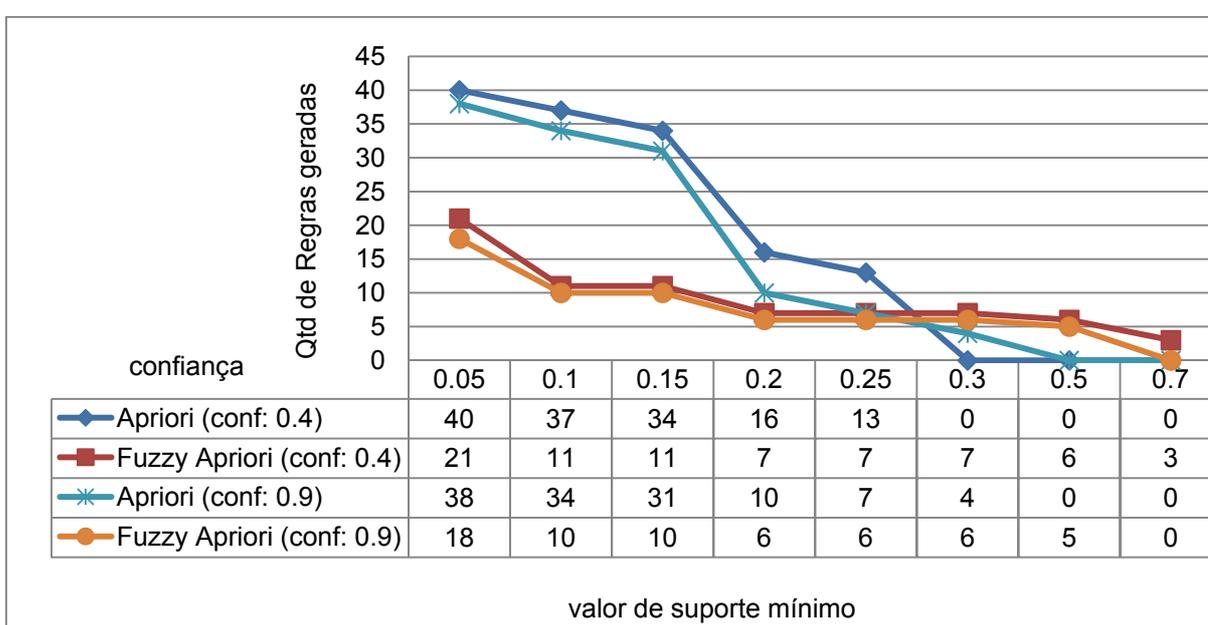
Figura 46 - Relação entre o número de regras de associação e valor mínimo de suporte para valor de confiança mínimo de 40%.



#### 4.4.2 Análise comparativa de quantidade de regras de classificação

O valor de confiança mínimo de 40% e 90% serviram de parâmetros para representar a distribuição da quantidade de regras de classificação extraídas dos dois algoritmos (Figura 47).

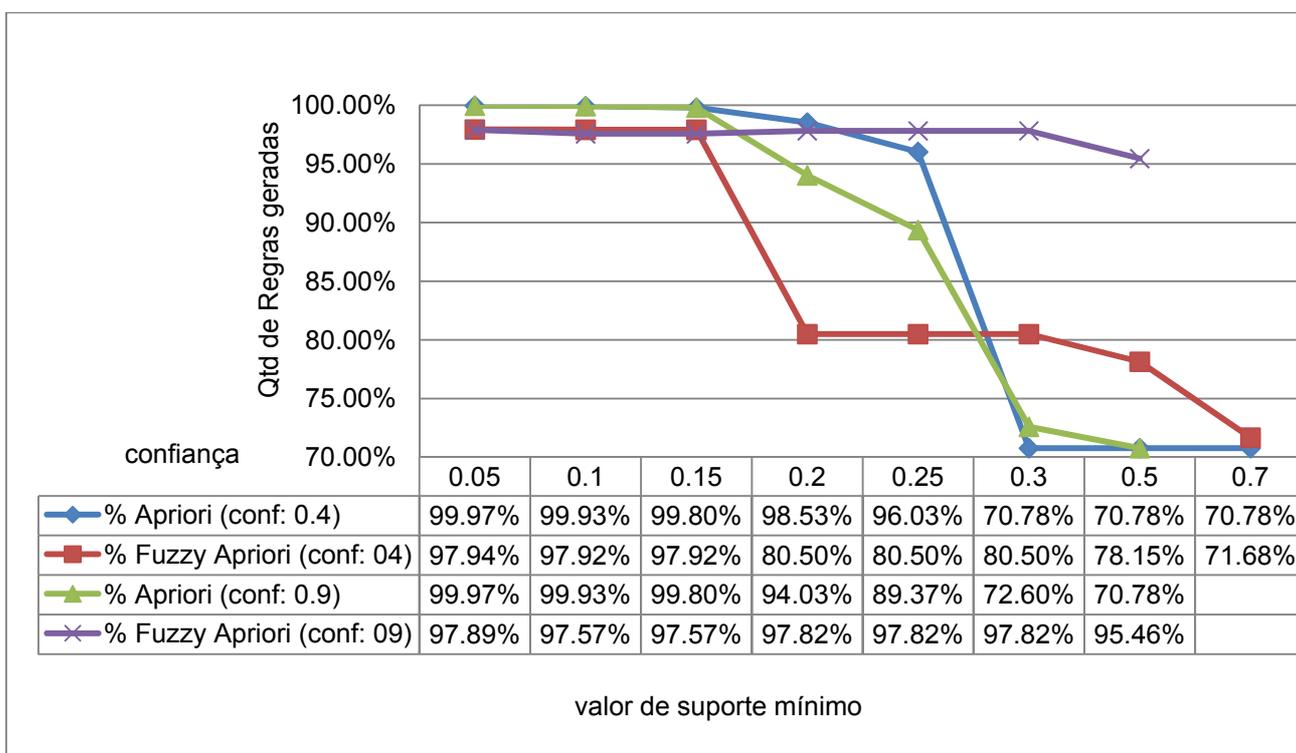
Figura 47 - Relação entre o número de regras de classificação e valor mínimo de suporte para valor de confiança mínimo de 40% e 90%.



#### 4.4.3 Análise comparativa de precisão das regras de associação

O valor de confiança mínimo de 40% e 90% serviram de parâmetros para representar a precisão das regras de associação extraídas dos dois algoritmos (Figura 48).

Figura 48 - Relação entre o percentual de precisão alcançado e valor mínimo de suporte para valor de confiança mínimo de 40% e 90%.



## CONCLUSÃO E FUTUROS TRABALHOS

A pesquisa apresentada neste trabalho abordou uma análise comparativa de desempenho dos algoritmos de regras de associação Apriori (AGRAWAL et al., 1996) e Fuzzy Apriori (HONG et al., 2001) aplicados num conjunto de dados de Sistema de Detecção de Intrusos.

Uma das contribuições de relevância abordada nesta pesquisa refere-se a comparação de algoritmos de aprendizado de máquina na extração de conhecimento de dados aplicados em cenário real.

Este trabalho teve como objetivo comparar técnicas de extração de regras de associação de atributos relevantes de um conjunto de dados de um Sistema de Detecção de Intrusos onde utilizou-se uma abordagem de detecção de intrusos por uso indevido. A proposta era de se analisar as taxas de erros alcançadas no processo de detecção de padrões de ataques.

O modelo apresentado possui uma arquitetura de dados composta por dez atributos relevantes e um atributo classe, e onde os resultados gerados pelo processo de extração de regras de associação e de extração de regras de classificação associativa, foi possível analisar os eventos em padrões considerados normais ou como ataque.

Ambos os algoritmos utilizados, Apriori e Fuzzy Apriori, extrairam regras de associação tidas como fortes capaz de classificar os ataques presentes no conjunto de dados das conexões KDD 99.

O conjunto de dados de detecção de intrusos já tinha presente uma saída classe previamente definida, de modo que o objetivo era de se extrair as regras de associação deste conjunto de dados e através de um subconjunto de dados fazer a classificação associativa, de modo a determinar se os registros de entrada correspondiam a uma conexão normal ou de ataque.

Devido ao conjunto de dados se apresentar desbalanceada foi preciso fazer considerações de pré-processamento de maneira que o conjunto de dados pudesse ser analisado. Como o percentual de ataques do tipo DoS representavam quase que 80% dos ataques, os ataques U2R e R2L foram desconsiderado. Devido a baixa percentagem de amostras no conjunto de dados, os ataques U2R e R2I

apresentariam baixa percentagem de acerto de detecção, o que prejudicaria na comparação dos algoritmos.

Para ambos os experimentos foram utilizados cinco partições para os atributos de valores quantitativos, sendo no Apriori na forma discretizados e no Fuzzy Apriori na forma de conjuntos fuzzy.

Os resultados obtidos com a utilização somente dos atributos relevantes para a extração de regras de associação relevantes e a respectiva classificação, foram muito satisfatório, pois em ambos os algoritmos se aproximaram e mantiveram em taxas de acertos elevadas.

Com relação os experimentos realizados de quantidade de regras de associação extraídas, quantidade de regras de classificação associativa extraídas e o respectivo percentual de acerto, temos a considerar que:

- O número de regras de associação utilizando o algoritmo Apriori se comportou de forma esperada. Ou seja, conforme o valor de suporte mínimo aumentava o número de regras decrescia. Em valores de suporte mínimo inferior, o algoritmo Apriori apresentou um quantitativo de regras de associação superior ao Fuzzy Apriori. No entanto, à partir do valor de suporte mínimo de 15%, e considerando em todos os valores mínimo de confiança, o quantitativo de regras de associação no Fuzzy Apriori foi superior e sua redução foi mais gradual. Em ambos a variação do valor mínimo de confiança acompanhou a mesma tendência, ou seja, conforme o valor mínimo de confiança aumentava o número de regras diminuía.
- Com relação a quantidade do número de regras de classificação associativa, que nada mais é que um subconjunto das regras de associação, teve um comportamento similar e esperado. Ou seja, conforme o valor de suporte mínimo aumentava o número de regras de associação classificativa diminuía. O número de regras de associação classificativa no algoritmo Fuzzy Apriori foi menor que no algoritmo Apriori em valores mínimo de suporte inferior, mas à partir do valor de suporte mínimo de 30% esse número foi superior. Observa-se no algoritmo Fuzzy Apriori, apesar de ser gerada em menor quantidade para valores de suporte mínimo inferior, elas continuaram a ser geradas para valores de suporte mínimo superior. Enquanto que no algoritmo Apriori para esses valores de suporte mínimo superior não foram mais geradas. Ou seja, no algoritmo Apriori não houve um subconjunto de

regras de associação suficiente que pudesse identificar a taxa de acerto das regras de associação extraídas.

- Quanto as taxas de precisão de acertos, observa-se que para valores de suporte inferior o algoritmo Apriori apresentou uma taxa de acerto superior, tendo como exemplos:
  - Com valor de suporte mínimo de confiança de 40% e 60%, o percentual de acerto no Fuzzy Apriori foi superior somente após que o subconjunto de regras de classificação associativa do algoritmo Apriori deixou de ser gerado.
  - Já com valor de suporte mínimo de confiança de 80% e 90%, o percentual de acerto no Fuzzy Apriori foi superior à partir do valor de suporte mínimo de 15%, mesmo tendo um subconjunto de regras de classificação associativa inferior que o subconjunto do algoritmo Apriori.

Diante dos números apresentados podemos considerar que o número de regras de associação extraídas no algoritmo Apriori apresentou uma tendência de curva mais regular, enquanto que no Fuzzy Apriori o número de regras de associação extraídas se manteve elevado e estável em valores de suporte mínimo entre 5% e 30%.

Quanto a taxa de precisão de acerto dos registros de ataque do conjunto de dados do Sistema de Detecção de Intrusos, o Fuzzy Apriori apresentou taxas de acerto superiores para um subconjunto de regras inferior ao Apriori. Ou seja, com um número de regras de classificação associativa menor a taxa de acerto foi superior. Observa-se no entanto que para este fato o Fuzzy Apriori teve um quantitativo de regras de associação gerada superior.

Embora o método Fuzzy Apriori tenha se apresentado como uma possível solução para um Sistema de Detecção de Intruso, observa-se que as características desse conjunto de dados não é uma das melhores opções. Segundo TAVALLAEE (2009), observou que após realizar uma análise estatística do conjunto de dados KDD99, questões importantes observadas afetam o desempenho dos algoritmos avaliados, e os resultados podem não ser tão satisfatórios para o sistema de detecção de intrusos. Uma delas foi grande número de registros redundantes. No conjuntos de treinamento e teste, verificou-se que cerca de 78% e 75% dos registros são duplicados, respectivamente. Esta grande quantidade de registros redundantes no

conjunto de treinamento causará que os algoritmos terão pouca quantidade de registros para extrair regras de associação fortes e consistentes, o que no trabalho, implicou em não considerar os ataques U2R e R2L.

A partir das análises e comparações realizadas, uma contribuição significativa deste trabalho foi de demonstrar a qualidade do algoritmo Fuzzy Apriori em relação ao algoritmo Apriori. Dessa maneira, uma proposta de trabalho futuro seria avançar na qualidade das regras de associação extraídas de um sistema de detecção de intrusos, visando assim melhorar a precisão de acerto dos ataques. Uma possível sugestão seria automatizar o dimensionamento das partições fuzzy de forma que um algoritmo possa prover variáveis linguísticas com características relevantes.

## REFERÊNCIAS

- AGRAWAL, Rakesh; IMIELINSKI, T.; SWAMI, A. *Mining Association Rules between Sets of Items in Large Databases*, SIGMOD 5/93, 207-216, Washington, USA, 1993.
- AGRAWAL, Rakesh; SRIKANT, Ramakrishnan . *Mining generalized association rules*. In: Proceedings of the 21st International Conference on Very Large Data Bases, pp. 407–419, 1995.
- AGRAWAL, Rakesh; SRIKANT, Ramakrishnan. *Fast Algorithms for Mining Association Rules*. In: 20th VLDB Conference, 487-498, Santiago, Chile, 1994.
- AGRAWAL, Rakesh; SRIKANT, Ramakrishnan. *Mining quantitative association rules in large relational tables*. In SIGMOD 96, 1996
- ALCALA, J. ; FERNANDEZ, A.; LUENGO, J.; DERRAC, J.; GARCIA, S.; SANCHEZ, L.; HERRERA, F. *KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework*. Journal of Multiple-Valued Logic and Soft Computing 17:2-3 (2011) 255-287
- ANDERSON, James P. *Computer Security Threat Monitoring and Surveillance, Technical Report*. James P. Anderson Co. Fort Washington, PA, abr. 1980.
- ASSOCIAÇÃO BRASILEIRA DE NORMAS E TÉCNICAS – ABNT. NBR ISO/IEC 17799:2005 – Tecnologia da informação – Técnicas de segurança – Código de prática para a gestão da segurança da Informação. Rio de Janeiro: ABNT, 2005.
- BERRY, J. A. M.; LINOFF, Gordon. *Data Mining Techniques for Marketing, Sales, and Customer Suppor*. John Wiley & Sons, Inc., 1997.
- BIANCO, D. *The Tao of Network Security Monitoring: Beyond Intrusion Detection*. In Information Security; Dec2004, Vol. 7 Issue 12, p16
- BIONDI, L. N.; COELHO, P. H. G.; AMARAL, J. L. M.; MELLO, M. H. C. S. *Minicurso de Sistemas Especialistas Nebuloso*. XXXVIII Simpósio Brasileiro de Pesquisa Operacional, 2006.
- CHAPKE, P. P.; RAUT, A. B. *Intrusion Detection System using fuzzy logic and Data Mining Technique*. International Journal of Advanced Research ni Computer Science and Software Engineering. IJARCSSE - Volume 2, Issue 12, December 2012.
- CHEN, G.; LIUI, H.; YU, L; WEI, Q.; ZHANG, X. *A New Approach to Classification Based on Association Rule Mining*. Decision Support System 42 (2006) 674-689.
- CHEN, Z; CHEN, G. *Bulding an associative classifier based on fuzzy association rules*. International Journal o Computational Intelligence Systems 1:3 (2008) 262-273.
- CORTES, S. C.; PORCARO, R. M.; LIFSCHITZ, Sérgio. *Mineração de Dados - Funcionalidades, Técnicas e Abordagens*. PUC-RioInf.MCC10/02, Maio 2002.

DAVENPORT, T. H., PRUSAK, L. *Conhecimento Empresarial: como as organizações gerenciam o seu capital intelectual*. 1998. Campus. Rio de Janeiro.

DENNING, D. E. *An intrusion detection model*. *IEEE Transaction on Software Eng* 13 (1987), 222–232.

EI-SEMARAY, A.; EDMONDS, J.; GONZALES-PINO, J.; PAPA, M. *Applying data mining of fuzzy association rules to network intrusion detection*. In Proc. of the 2006 IEEE Workshop on Information Assurance, West Point, NY, pp. 100-107, 200.

FAYYAD, U. et al. *Advances in Knowledge Discovery and Data Mining*. MIT Press, Cambridge: 1996.

FAYYAD, U; PIATETSKY-SHAPIRO, G.; SMYTH, P. *From data mining to knowledge discovery in databases*. *AI Magazine* 17 (1996) 37-54.

GODINEZ, F.; HUTTER, D; MOROY, R. *Attribute Reduction for Effective Intrusion Detection*. In Proceedings of the Advances in Web Intelligence Conference (AWIC'2004), Cancun, Mexico, pp. 74-83, 2004.

GOLDSCHIMIDT, R.; PASSOS, E. *Data mining: Um guia prático*. Rio de Janeiro: Campus, 2005.

HALL, Mark; FRANK, Eibe; HOLMES, Geoffrey; PFAHRINGR, Bernhard; REUTEMAN, Peter; WITTEN, Ian H. *The WEKA Data Mining Software: An Update*. SIGKDD Explorations, 2009, Volume 11, Issue 1.

HAN Jiawei; KAMBER Micheline. *Data Mining - Concepts and Techniques*. Morgan Kaufmann Publishers, Inc, 2001.

HAN, J; PEI J; YIWEN Y; MAO, R. *Mining frequent patterns without candidate generation: A frequent-pattern tree approach*. *Data Min Knowl Discov* 8(1):53–87, 2004.

HONG, T. P.; KUO, C.S.; CHI, S.C. *Trade-off between computation time and number of rules for fuzzy mining from quantitative data*. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 9:5 (2001) 587-604.

HONG, T.P.; KUO, C.S.; WANG, S.L. *A fuzzy AprioriTid mining algorithm with reduced computational time*. *Applied Soft Computing* , 2004, 5(1): 1-10

HULLERMEIER, E; YI, Y. *In Defense of Fuzzy Association Analysis*. *IEEE Transactions on Systems, Man, and Cybernetic - Part B*, 37(4): 1039-1043, 2007.

IDRIS, N. B. *Improved Intrusion Detection System Using Fuzzy Logic For Detecting Anomaly And Misuse Type Of Attacks*. The proceedings of the International Conference on soft computing and pattern recognition, (SoCPaR), Malacca, Malaysia, 2009.

JANG, J.-S. R.; GULLEY, N. *The Fuzzy Logic Toolbox for use with MATLAB*. The MathWorks, Inc., Natick, Massachusetts, 1995.

JANG, J.S.R.; C.T.S. & MIZUTANI, i, E. (1997), *Neuro-Fuzzy and Soft Computing*, Prentice Hall, New York-USA.

KACPRZYK, J. *Multistage fuzzy control*. Chichester, UK: John Wiley Sons, 1997.

KAYACIK, H. G.; ZINCIR-HEYWOOD, A. N.; HEYWOOD, M. I. *Selecting Features for Intrusion Detection: A Feature Relevance Analysis on KDD 99 Intrusion Detection Datasets*. In Third Annual Conference on Privacy, Security and Trust, St. Andrews, New Brunswick, Canada, October 2005.

KDD Cup 1999 data (Computer network intrusion detection). Disponível em <<http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>> Último Acesso em: 26/01/2014.

KUOK, C. M.; FU, A.; WONG, M. H. *Mining Fuzzy Association Rules in Databases*, SIGMOD Record, pp. 41-46, Vol. 27, No. 1, 1998.

LAROSE, D. T. *Discovering Knowledge in Data: An Introduction to Data Mining*. ISBN 0-471-66657-2, John Wiley & Sons, Inc, 2005.

LEE, W.; STOLFO, S. J.; MOKD, K. W. *A data mining framework for building intrusion detection models*. In Proceedings of the 1999 IEEE Symposium on Security and Privacy, May 1999.

LI, Wenmin, HAN, Jiawei; PEI, Jian. *CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules*. In: ICDM'01, 369-376, 2001.

LIU, B.; HSU, W.; MA, Y. *Integrating Classification and Association Rule Mining*. In: Fourth International Conference on Knowledge Discovery and Data Mining, 80-86, 1998.

LIU, H.; MOTODA, H. *Feature Selection for Knowledge Discovery and Data Mining*, July 1998, ISBN 0-7923-8198-X, by Kluwer Academic Publishers.

MAIMON, Oded; ROKACH, Lior. *Chapter 1: Introduction to Knowledge Discovery in Databases*. Data Mining and Knowledge Discovery Handbook. 2005.

MANNILA. H. (1997) *Methods and problems in data mining*. In: Proceedings of the 6th International Conference on Database Theory (ICDT'97), LNCS, Springer-Verlag, 1186: 41–55.

MARIANO, A. M. *Comparação de Algoritmos Paralelos para a Extração de Regras de Associação no Modelo de Memória Distribuída*. Dissertação - Universidade Federal do Mato Grosso do Sul, 2011.

MENZIES, Tim; HU, Ying. *Data Mining For Busy People*. IEEE Computer, Outubro de 2003, pgs. 18-25.

MORENO, N. M.; SEGRERA, Saddys; LOPEZ, F. V. *Association Rules: Problems, solutions and new applications*. Actas del III Taller Nacional de Minería de Datos y Aprendizaje, TAMIDA 2005, pp.317-323.

OLUSOLA, A. A.; OLADELE, S. A.; ABOSEDE, O. D. *Analysis of KDD'99 Intrusion Detection Dataset for Selection of Relevance Features*. In Proceeding of the Word Confress on Engineering and Computer Science, San Francisco, USA, october 2010.

PEI, J; HAN, J; LAKSHMANAN, LVS; Data Engineering, 2001. Proceedings. 17th International Conference on, 433-442.

PITONI, R. M. *Mineração de Regras de Associação nos Canais de Informação do Direto*. Trabalho de Conclusão de Curso – UFRS, 2002.

ROOS, T. J. *Fuzzy Logic with Engineering Applications*. McGraw-Hill, Inc., New. York, 1995.

SCHEINERMAN, E. R. *Matemática Discreta - Uma Introdução*. Cengage Learning Editores; 2003. ISBN 978-85-221-0291-4. p. 27.

SHAPIRO, Gregory; FRAWLEY, W. J. *Discovery, analysis, and presentation of strong rules*. In, eds., Knowledge Discovery in Databases, AAAI/MIT Press, Cambridge, MA. (1991).

SILVA, M. P. S. *Mineração de Dados - Conceitos, Aplicações e Experimentos com WEKA*. Minicurso de Mineração de Dados. UERN & INPE, 2004.

TAN, P.-N.; STEINBACH, M.; KUMAR, V. *Introduction to Data Mining*. Pearson, 2006.

TANSCHIEIT, R. *Sistemas fuzzy*, In: Inteligência computacional: aplicada à administração, economia e engenharia em Matlab, pp. 229–264, São Paulo, Thomson Learning, 2007.

TANSCHIEIT, R. (2007). *Sistemas fuzzy*. Thomson Learning, São Paulo, pp. 229–264.

TAVALLAEE, M.; BAGHERI, E.; LU, W.; GHORBANI, A.A. *Detailed Analysis of the KDD CUP 99 Data Set In IEEE Symposium on Computational Intelligence in Security and Defense Applications* . (CISDA) (2009), pp. 1-6.

UCI - UCIrvine Machine Learning Repository. Disponível em <<http://archive.ics.uci.edu/ml/index.html>> Acesso em: 26/01/2014.

VERLINDE, H., De COCK, M., BOUTE, R. *Fuzzy Versus Quantitative Association Rules: A Fair Data-Driven Comparison*. IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, 36, 679-683 (2006).

WANG, Jie. *Computer Network Security: Theory and Practice*. Higher Education Press, Beijing and Springer-Verlag GmbH Berlin Heidelberg, 2009.

WANG, Wei; GOMBAULT, S.; GUYET, T. *Towards Fast Detecting Intrusions: Using Key Attributes of Network Traffic*. Internet Monitoring and Protection, 2008. ICIMP '08. The Third International Conference on, janeiro/2008.

WU, K. X.; HAO J.; WANG, C. *Application of Fuzzy Association Rules in Intrusion Detection*, International Conference on Internet Computing & Information Services (ICICIS), 2011, pp. 269 – 272.

YOGENDRA, K. J.; UPENDRA. Article: *Intrusion Detection using Supervised Learning with Feature Set Reduction*. International Journal of Computer Applications 33(6):22-31, November 2011.

ZADEH, L. (1965). *Fuzzy sets*, *Information Control* 8, 338-353. fonte: BAE.

ZAKE, M. J. *Scalable algorithms for association mining*. IEEE Transactions on Knowledge and Data Engineering, 12(3):372–390, 2000