ANA CAROLINA MENDONÇA PILATTI DE PAULA

INTEGRAÇÃO DE MODELOS BASEADA EM SISTEMAS MULTI-AGENTE

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

CURITIBA

2005

ANA CAROLINA MENDONÇA PILATTI DE PAULA

INTEGRAÇÃO DE MODELOS BASEADA EM SISTEMAS MULTI-AGENTE

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

Área de Concentração: *Metodologias e Técnicas de Computação*

Orientador: Prof. Dr. Fabrício Enembreck
Co-orientador: Prof. Dr. Bráulio Coelho Ávila

CURITIBA
2005

Paula, Ana Carolina Mendonça Pilatti de
P324i Integração de modelos baseada em sistemas multi-agente /
2005 Ana Carolina Mendonça Pilatti de Paula ; orientador, Fabrício Emembreck ; co-orientador, Bráulio Coelho Ávila. – 2005.
xii, 81 f. : il. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2005
Inclui bibliografia

1. Exploração de dados. 2. Armazenamento de dados. 3. Agentes inteligentes (Software). I. Enembreck, Fabrício. II. Ávila, Bráulio Coelho. III. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática Aplicada. IV.Título.

CDD 21. ed. 005.74
006.3

Aos meus pais,
meu esposo Jefferson e
minha filha Ana Cristina.

Agradecimentos

À Pontifícia Universidade Católica do Paraná, aos professores do Curso de Pós-Graduação em Informática Aplicada, em especial aos professores Dr. Bráulio Coelho Ávila, Dr. Edson Emílio Scalabrin e ao orientador Prof. Dr. Fabrício Enembreck.

Aos colegas de laboratório Fabiano Mitsuo Hasegawa, Emerson Luiz dos Santos e Vera Lúcia Marchiori. Aos colegas de curso Igor de Souza Paiva, Marcelo Schuck e Simone Nasser.

Aos meus pais e ao meu esposo Jefferson de Paula pela compreensão e apoio constante.

Sumário

Capítulo 3

Capítulo 4

Capítulo 5

Capítulo 6

Lista de Figuras

Lista de Tabelas

Resumo

A Mineração de Dados permite produzir conhecimento a partir de dados armazenados. Com o crescimento das bases de dados e a necessidade de bases distribuídas, novas técnicas são pesquisadas visando acelerar o processo de extração do conhecimento nestas bases de dados. Este trabalho apresenta uma técnica de Mineração de Dados Distribuída baseada em um ambiente multi-agente, chamado *SMAMDD* (Sistema Multi-agente para Mineração de Dados Distribuídos), que utiliza dados particionados em subconjuntos. Em cada subconjunto agentes realizam a aprendizagem localmente, utilizando o ambiente *WEKA* e o algoritmo de aprendizagem *RIPPER*, e posteriormente os resultados geram um modelo global. Para tal, agentes realizam cooperação através de trocas de mensagens visando acelerar o processo de descoberta de conhecimentos. O processo de cooperação é hierárquico coordenado por um agente gestor. Basicamente, o processo envolve as seguintes fases: (1) preparação dos dados, (2) geração de modelos individuais, onde cada agente aplica o algoritmo de aprendizagem sobre um subconjunto de dados para a obtenção de regras, (3) cooperação entre os agentes, e (4) construção de um modelo baseado nos resultados obtidos a partir da cooperação entre os agentes. O modelo proposto tem como objetivo a seleção das melhores regras geradas que irão compor o modelo global sem, entretanto, produzir uma redução no desempenho da aprendizagem. A Mineração de Dados Distribuída permite oferecer melhor escalabilidade, aumento da segurança dos dados e melhor tempo de resposta se comparada a um modelo centralizado. O sistema Multi-agente para Mineração de Dados Distribuída proposto neste trabalho permite a descoberta de conhecimento em subconjuntos de bases oriundos de uma base maior. Os agentes responsáveis por analisar os dados locais e comunicar-se uns com os outros durante o etapa de mineração, trocaram conhecimentos locais até chegar a um conhecimento coerente global.

Palavras-Chave: 1.Mineração de Dados 2.Mineração de Dados Distribuído 3.Sistemas Multi-agente

Abstract

The Data Mining allows to produce knowledge from stored data. With the databases growth and the need to distributed, new techniques are searched aiming accelerate the knowledge extraction process in databases. This work introduces a Distributed Data Mining technique based in an ambient multi-agent, called *SMAMDD*, that uses data divided in subsets. In each subset agents accomplish the learning locally, using the environment *WEKA* and the learning algorithm *RIPPER*, and afterwards the results are combined in a global model. For such, it accomplishes the cooperation between agents through messages changes aiming accelerate the knowledge discovery process. The cooperation process is hierarchical coordinated for an agent manager. Basically, the process involves the following phases: (1) data preparation, (2) individual models generation, where each agent applies the learning algorithm on a data subset for the rules obtainment, (3) cooperation between agents, and (4) construction of an model based in the obtained results from the cooperation among agents. The proposed model has as objective the better generated rules selection that are going to compose the global model without, however, produce a reduction in the learning performance. The Distributed Data Mining allows to offer better scalability, data safety increase and better response time if compared to a centralized model. The Multi-agent system for proposed Distributed Data Mining in this work, allow the knowledge discovery in bases subsets, derived of a base larger. The responsible agents to for analyzing the local data and to communicate some with the others during the mining stage, they changed local knowledge until arrive to a global coherent knowledge.

Keywords: 1. Data Mining 2. Distributed Data Mining 3. Multi-agent Systems

Capítulo 1

Introdução

Recentemente, o avanço tecnológico tem permitido produzir conhecimento a partir de dados armazenados. O aumento no uso de códigos de barra em produtos, transações comerciais realizadas digitalmente e o advento da Internet, geram diariamente grandes quantidades de dados. Este crescimento tem gerado a necessidade de novas técnicas e ferramentas que possam inteligentemente e automaticamente transformar dados processados em informações e conhecimentos úteis. Conseqüentemente, a Mineração de Dados tornou-se uma área de pesquisa de extrema importância.

A Mineração de Dados (*Data Mining*) permite uma descoberta eficiente de informações válidas e não óbvias em uma grande coleção de dados, sendo utilizada para gerenciamento de informações e tomadas de decisões, permitindo o aumento das oportunidades de negócios.

O crescimento das bases de dados e a existência de bases distribuídas intensificou a pesquisa de novas técnicas para acelerar o processo de mineração de dados em grandes bases de dados, buscando soluções para o problema da escalabilidade (tratamento da dimensionalidade dos dados).

A Mineração de Dados Distribuída (DDM) é resultado da evolução da tecnologia de mineração de dados, sendo uma fusão desta com sistemas distribuídos. DDM considera o fato que dados estejam remotamente distribuídos entre diferentes locais interligados via rede.

DDM pode ser aplicada em conjuntos de dados públicos na Internet, bancos de dados corporativos em uma *Intranet*, ambientes de computação móvel, coleta de dados sensoriais distribuídos para monitoração, etc. A Mineração de Dados Distribuída oferece melhor

escalabilidade, possibilidade de aumento da segurança dos dados e melhor tempo de resposta se comparada a um modelo centralizado.

Assim sendo, existem várias aplicações de DDM onde o conhecimento obtido é importante, principalmente quando o conjunto de dados é volumoso. Este conhecimento é adquirido através de aplicações de Algoritmos de Aprendizado de Máquina (AM) sobre bases de dados, gerando conceitos que são representados de alguma forma.

De acordo com Chan, Stolfo e Prodomidis (2000), a maioria dos algoritmos de aprendizagem são computacionalmente complexos e requerem todos os dados na memória principal, entretanto, quando os dados excedem a memória principal, caracteriza-se o problema de escalabilidade.

Diversas técnicas têm sido propostas objetivando viabilizar a mineração em grandes bases de dados, entre elas: subdivisão dos dados, desenvolvimento de algoritmos eficientes e a utilização da representação relacional.

Segundo Chan (1998), a escalabilidade (tratamento da dimensionalidade dos dados) em mineração de dados distribuída está na capacidade em descobrir conhecimento sobre grandes quantidades de dados, sem redução no desempenho da aprendizagem. No projeto *JAM*, desenvolvido por Chan e Stofo (Stofo *et al.* 1997), são executados agentes de aprendizagem sobre bases locais, cada classificador pode importar classificadores e combinar com seu próprio classificador local, usando agentes de aprendizagem. Finalmente, os classificadores base e o meta-classificador são processados e o sistema *JAM* gerencia a execução de módulos para classificar conjuntos de dados. O sistema *JAM* pode ser visto como uma aplicação paralela em que dados locais funcionam de forma autônoma.

Este trabalho apresenta uma técnica de Mineração de Dados Distribuída baseada em um ambiente multi-agente, que utiliza integração de modelos. A integração de modelos consiste na união de modelos locais em um modelo global consistente. Em cada subconjunto agentes realizam a aprendizagem localmente e posteriormente os resultados são combinados em um modelo único. Para tal, realiza-se a cooperação entre agentes visando acelerar o processo de descoberta de conhecimento. O modelo proposto tem como objetivo a seleção das melhores regras geradas que irão compor o modelo global sem, entretanto, produzir uma redução no desempenho da descoberta de conhecimento.

1.1. Motivação

A maioria dos algoritmos de aprendizagem (regras de associação, *clustering*, classificação, pré-processamento, etc) são computacionalmente complexos e requerem que todos os dados estejam em um único local, o que inviabiliza a mineração em grandes bases de dados.

Para viabilizar a mineração em grandes bases, diversas técnicas têm sido propostas e implementadas por meio de algoritmos indutivos, entre elas, a subdivisão dos dados. A técnica implica em dividir um conjunto de dados em subconjuntos menores, sendo a descoberta de conhecimento realizada em cada subconjunto. A divisão dos dados reduz problemas de acesso à memória. Entretanto, avaliar a eficiência de técnicas de escalabilidade torna-se complexo se houver redução na qualidade da aprendizagem. Através da Mineração de Dados Distribuída busca-se utilizar métodos para resolver o problema do dimensionamento dos dados, o que motivou o estudo e desenvolvimento deste trabalho. O estudo da Inteligência Artificial Distribuída e da sub-área Sistemas Muli-agentes possibilitou a elaboração de um Sistema Multi-agente para Mineração de Dados Distribuídos (*SMAMDD*) que visa propor a integração de modelos individuais, gerados a partir de dados particionados, em um modelo global.

1.2.Objetivos Gerais

Este trabalho tem por objetivos: *i)* o estudo dos aspectos relacionados a Mineração de Dados Distribuídos e *ii)* o desenvolvimento de um Sistema Multi-Agente que permite descobrir conhecimento sobre bases de dados, as quais particionadas geram modelos individuais que irão compor um modelo global.

1.3 Organização

O presente trabalho está organizado em seis capítulos. O Capítulo 2 apresenta alguns conceitos sobre Mineração de Dados; posteriormente, são abordados os conceitos sobre Mineração de Dados Distribuídos no Capítulo 3. Conceitos sobre Inteligência Artificial Distribuída, Sistemas Multi-agente e Resolução Distribuída de Problemas são discutidos no Capítulo 4. O Capítulo 5 descreve um Sistema Multi-agente para Mineração de Dados Distribuídos proposto neste trabalho, bem como os resultados obtidos neste sistema. As conclusões finais e a proposta de trabalhos futuros são apresentadas no Capítulo 6.

Capítulo 2

Mineração de Dados

A grande quantidade de informações armazenadas por organizações em bancos de dados informatizados, pode esconder conhecimentos valiosos e úteis para a tomada de decisões. Uma área interdisciplinar, Descoberta de Conhecimento em Banco de Dados (KDD - *Knowledge Discovery from Data*), surgiu em resposta à necessidade de novas soluções para viabilizar a análise e extração de conhecimento em grandes bases de dados. Para extrair conhecimento, são necessárias ferramentas de exploração conhecidas como Mineração de Dados (MD). Uma das primeiras etapas no processo de KDD é a definição da tarefa de MD a ser realizada, que determina o tipo de conhecimento a ser descoberto. Neste capítulo é apresentado a técnica de mineração de dados, suas tarefas, o processo de descoberta de conhecimento em bancos de dados e alguns algoritmos conhecidos da literatura.

2.1. Definição

Mineração de dados (MD) (Han & Kamber, 2001; Hand, Mannila e Smyth, 2001) é uma técnica para descoberta de conhecimento em bases de dados, permitindo responder alguma pergunta específica de interesse do usuário.

"Não há um método de Mineração de Dados universal e a escolha de um algoritmo particular para uma aplicação particular é de certa forma uma arte" (Fayyad *et al.*, p. 86, 1996a).

Para obter respostas ou extrair conhecimentos interessantes de bases de dados, diversas técnicas de MD estão disponíveis na literatura, podendo as principais serem agrupadas em:

- Indução e/ou extração de regras;

- Redes Neurais;
- Algoritmos evolucionários;
- Técnicas estatísticas (redes bayesianas), etc;

Para a escolha da técnica mais adequada é necessário conhecer o domínio da aplicação de MD: quais são os atributos importantes, quais os relacionamentos possíveis, o que é uma função útil para o usuário, que padrões são conhecidos e assim por diante.

A MD pode ser utilizada em muitas tarefas diferentes, tais como: classificação, associação, regressão, sumarização, modelagem de dependência e identificação de mudanças e desvios. Abaixo descreve-se algumas destas tarefas.

2.1.1. Classificação

Classificação é uma das tarefas mais referenciadas na literatura de MD. Neste tipo de tarefa, o objetivo é descobrir um relacionamento entre um atributo meta (cujo valor ou classe será previsto) e um conjunto de atributos previsores. O sistema deve descobrir este relacionamento a partir de exemplos com classe conhecida. O relacionamento descoberto será usado para prever o valor do atributo meta (ou a classe) para exemplos cujas classes são desconhecidas.

A literatura apresenta diversas técnicas de classificação (King *et.al*. 1995; Hand, 1997). Segundo Michie *et. al*. (1994) sendo as principais propostas originárias de três campos de pesquisa: estatística, aprendizagem de máquina simbólica e redes neurais.

Existem diversas formas de representar o conhecimento em um sistema de aprendizagem. No contexto da tarefa de classificação, o conhecimento descoberto muitas vezes é expresso como um conjunto de regras do tipo SE-ENTÃO, uma vez que este tipo de representação de conhecimento é intuitivo para o usuário (Carvalho & Freitas, 2000).

Regras do tipo SE-ENTÃO são também chamadas de regras de produção, constituem uma forma de representação simbólica e possuem a seguinte forma:

SE <antecedente> ENTÃO <conseqüente>

O antecedente é formado por expressões condicionais envolvendo atributos do domínio da aplicação existentes no banco de dados.

O conseqüente é formado por expressões que indicam a previsão de algum valor para um atributo meta, obtido em função dos valores encontrados nos atributos que compõem o antecedente.

Portanto, a tarefa é descobrir regras de classificação capazes de prever o valor de um atributo meta a partir dos valores de atributos antecedentes, permitindo o planejamento de ações futuras.

Na tarefa de classificação, os dados são divididos em duas bases de dados, mutuamente exclusivas, chamadas base de treinamento e base de testes. O algoritmo de mineração de dados gera regras a partir da base de treinamento e as avalia sobre a base de testes.

A tarefa de classificação pode ser aplicada em diversas áreas, como:

- Bancos: determinar se uma aplicação de hipoteca é boa ou ruim, ou se uma transação de cartão de crédito é fraudulenta;
- Educação: determinar necessidades especiais para um novo estudante;
- Medicina: diagnosticar a presença de uma determinada doença;
- Justiça: determinar se a assinatura de uma pessoa é verdadeira ou fraudulenta.

2.1.2 Descoberta de Regras de Associação

Nesta tarefa, cada instância de dados (registro) corresponde a uma transação de um cliente com itens assumindo valores binários (verdadeiros ou falsos), indicando se o cliente comprou o item ou não. Uma regra de associação é uma relação na forma SE X ENTÃO Y, onde X e Y são conjuntos de itens e $X \cap Y = Ø$.

A cada regra de associação são atribuídos dois fatores: suporte e confiança. O suporte é igual ao número de registros com X e Y dividido pelo número total de registros. A confiança é igual ao número de registros com X e Y dividido pelo número de registros com X. Por exemplo, um supermercado pode descobrir que de 1000 clientes que efetuaram compras terça à noite, 200 compraram fraldas e destes 200 clientes, 50 compraram cervejas. Então a regra de associação seria "Se compra fralda então compra cerveja" com um Suporte de 200/1000 = 20% e Confiança de 50/200 = 25%.

A tarefa visa descobrir todas as regras de associação com suporte maior ou igual ao suporte mínimo e confiança maior ou igual a confiança mínima.

Embora ambas as regras de classificação e associação possuam a estrutura SE-ENTÃO, existem diferenças importantes entre elas. Algumas destas diferenças são: primeiramente, as regras de associação podem ter mais que um item no conseqüente enquanto que as regras de classificação sempre possuem um único atributo (meta) no conseqüente. Em

segundo, a tarefa de classificação, diferentemente da associação é assimétrica com respeito aos atributos antecedentes e o atributo meta. Atributos previsores podem ocorrer somente no antecedente da regra e o atributo meta ocorre somente no conseqüente da regra.

2.1.3 Agrupamento (*Clustering*)

Na tarefa de classificação a classe de um exemplo de treinamento é fornecida como entrada para o algoritmo de mineração de dados, caracterizando uma forma de aprendizagem supervisionada. De forma contrária, na tarefa de agrupamento o algoritmo de mineração de dados deve "descobrir" classes por si só, particionando os exemplos em *clusters*, o que é uma forma de aprendizagem não-supervisionada.

Exemplos similares (isto é, exemplos com valores de atributos similares) tendem a ser agrupados em um mesmo *cluster*, do contrário, exemplos diferentes são associados à *clusters* distintos. Após a definição dos *clusters*, cada *cluster* pode ser considerado como uma classe.

Finalizado o agrupamento, podem-se aplicar métodos de classificação para descobrir regras que descriminem registros de diferentes classes e métodos de sumarização que produzem descrições características de cada classe.

Há dois tipos básicos de agrupamentos: hierárquico e não-hierárquico. O agrupamento hierárquico descobre uma hierarquia de *clusters.* Neste caso, o usuário pode escolher, após o *clustering,* o número de *clusters* mais útil para ele. O agrupamento não-hierárquico requer que o usuário pré-defina antes do *clustering*, o número de *clusters* desejado. Entre os dois agrupamentos o hierárquico tende a ser computacionalmente mais caro que o não-hierárquico.

O agrupamento geralmente é utilizado para uma exploração ou entendimento inicial dos dados.

A tarefa de agrupamento pode ser aplicada em diversas áreas, como:

- Marketing: ajuda na descoberta de grupos distintos de clientes, e uso deste conhecimento para criar campanhas dirigidas;
- Uso de terras: identificação de áreas de uso similar a partir de uma base de observação via satélite;
- Seguros: identificação de grupos de assegurados com alto custo de sinistro;
- Planejamento urbano: identificação de grupos de casas de acordo com seu tipo, valor e localização geográfica;
- Estudos sobre terremotos: identificação de epicentros e seu agrupamento ao longo de

falhas geológicas.

2.1.4 Modelagem de Dependência (regras de indução)

Esta tarefa pode ser considerada como uma generalização da tarefa de classificação. Expresso como um conjunto de regras do tipo SE-ENTÃO, onde qualquer atributo pode ocorrer tanto no antecedente (SE) da regra como no conseqüente (ENTÃO) de uma outra regra, não podendo estar presente em ambos, antecedente e conseqüente, de uma mesma regra.

Assim como na tarefa de classificação, os dados podem ser divididos em duas bases distintas: base de treinamento, a qual será utilizada para gerar as regras, e base de testes para avaliá-las.

2.2 O Processo de Descoberta de Conhecimento

Segundo Fayyad *et al.* (1996a): "Descoberta de Conhecimento em Banco de Dados (KDD) é o processo não trivial de identificação de padrões, a partir de dados, que sejam válidos, novos, potencialmente úteis e compreensíveis".

Na definição de Fayyad, KDD é um processo geral de descoberta de conhecimento composto de quatro fases: preparação dos dados, busca de padrões, avaliação do conhecimento e refinamentos. Em geral, o resultado de uma fase é usado para dois propósitos: ser passado para a próxima fase e ser utilizado como um *feedback* das fases anteriores, possibilitando que os resultados produzidos em uma iteração possam ser usados para melhorar as próximas iterações do processo.

Os padrões devem ser novos, compreensíveis e úteis, trazendo algum benefício que possa ser facilmente compreendido pelo usuário auxiliando-o no processo de tomada de decisão.

Para que o conhecimento descoberto seja útil é importante que as metas estabelecidas estejam bem definidas. Segundo Fayyad *et al.*(1996a), no KDD as metas são definidas em função dos objetivos na utilização do sistema, podendo ser: verificação ou descoberta.

Quando a meta é do tipo verificação, o sistema está limitado a hipóteses definidas pelo usuário, enquanto que na descoberta o sistema encontra novos padrões de forma autônoma. A descoberta pode ser dividida em previsão e descrição, conforme Figura 2.1.

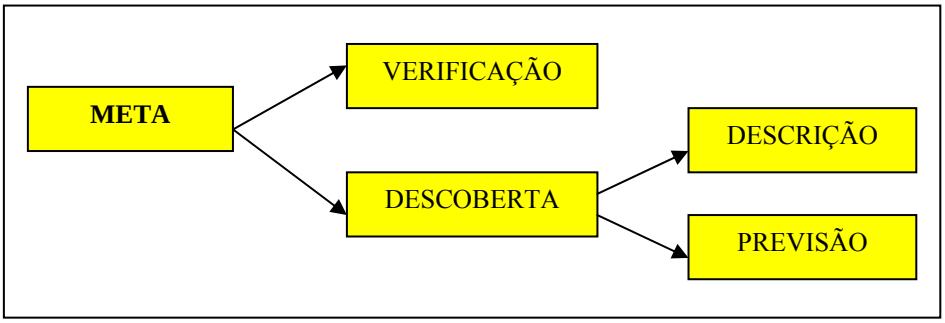

Figura 2.1. Tipos de metas do processo KDD

A descrição procura encontrar padrões interpretáveis pelos usuários, que descrevam os dados. A previsão parte de diversas variáveis para prever outras variáveis ou valores desconhecidos (Fayyad *et al.*, 1996b).

As metas de previsão e descrição são alcançadas através de alguma das seguintes tarefas de mineração de dados: classificação, regressão, agrupamento, sumarização, modelagem de dependência e identificação de mudanças e desvios, sendo a tarefa de classificação a mais empregada.

O processo de descoberta de conhecimento compreende as etapas de construção de uma *Data Warehouse (DW):* pré-processamento, mineração de dados (DM) e pós-processamento, conforme ilustra a Figura 2.2 (Freitas, 1998 p. 32).

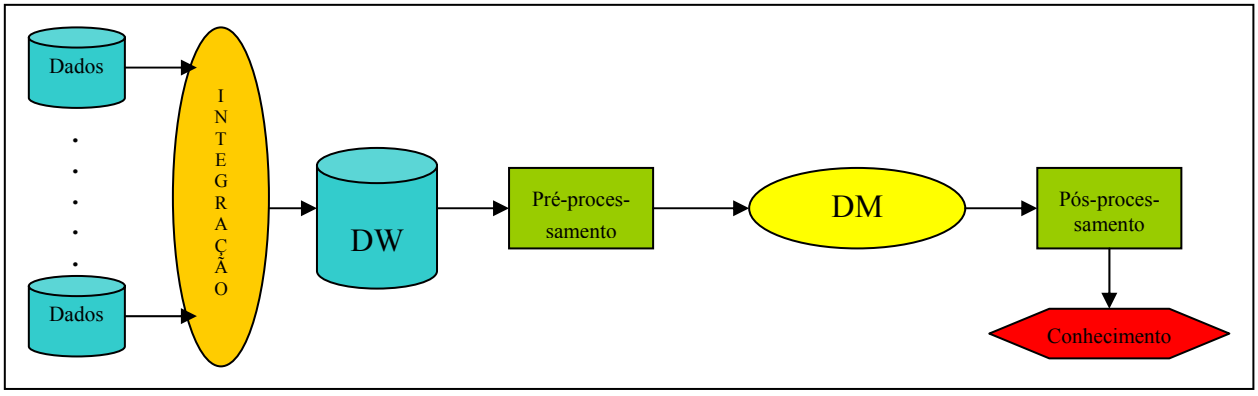

Figura 2.2. Etapas do processo de descoberta do conhecimento (Freitas, 1998 p.32)

A etapa de pré-processamento inclui três subfases: limpeza, seleção e transformação dos dados (não necessariamente nesta ordem). Algumas destas operações podem ser parcialmente realizadas durante a construção da DW.

A limpeza dos dados envolve a verificação de consistência, correção de erros, preenchimento ou eliminação de valores nulos, remoção de *outliers*, etc. Os métodos de

limpeza são dependentes do domínio sendo a participação do usuário importante para auxiliar na decisão de identificação de erros ou *outliers*.

A fase de seleção determina os atributos relevantes em uma base de dados, podendo ser realizada através do processo de *wrapper* ou do processo por filtro. Outro importante tipo de transformação de dados é a discretização de atributos contínuos.

O pré-processamento é seguido pela fase de Mineração de Dados, onde o algoritmo do processo de descoberta do conhecimento é aplicado sobre os dados pré-processados. Finalmente, o resultado do algoritmo de DM pode ser refinado na fase de pós-processamento, que envolve interpretação do conhecimento descoberto ou novos processamentos deste conhecimento. O objetivo desta fase é avaliar o processo de descoberta, melhorar a compreensão e/ou selecionar conhecimento descoberto que seja mais interessante. Quando são geradas muitas regras, é importante remover algumas regras e/ou condições para facilitar a compreensão do conhecimento extraído.

Existem diversas abordagens para avaliar o processo de descoberta de conhecimento, incluindo-se: precisão dos resultados (alguma medida da taxa de acerto), eficiência (tempo de processamento), facilidade de compreensão do conhecimento extraído, etc. A maior parte da literatura utiliza precisão (taxa de acerto) como principal meio para avaliar as técnicas de KDD (Freitas, 1998), principalmente no contexto da tarefa de classificação.

2.2.1 Data Warehouse

Data Warehouse (DW) é um repositório de dados projetado para ser utilizado como base para suporte à decisões e sistemas de descoberta do conhecimento. A integração dos dados é uma questão crucial, envolve padronização do formato e nome dos atributos, remoção das inconsistências nos dados, etc.

A atualização em um DW usualmente consiste da interseção de novos dados (e talvez a remoção de dados antigos na DW) em determinados períodos de tempo.

2.2.2 Seleção de atributos

A seleção de atributos consiste na seleção, entre todos os atributos avaliados no algoritmo de KDD, de um subconjunto de atributos relevantes que servirão de entrada para o algoritmo de *data mining*.

Entre as motivações para a seleção de atributos estão: maximizar a precisão da

descoberta de conhecimentos e reduzir o tempo de processamento do algoritmo de DM.

Há dois métodos principais de seleção de atributos: *wrapper* e filtro. No método *wrapper* os dados são divididos em dois subconjuntos: treinamento e avaliação. Uma pesquisa heurística é realizada sob os subconjuntos de atributos, sendo cada iteração desta pesquisa composta de três passos: selecionar um subconjunto de atributos de acordo com algum critério, aplicar o algoritmo de DM sobre o subconjunto de treinamento e mensurar a precisão da classificação das regras descobertas. Diferentemente do método *wrapper,* no método filtro o algoritmo de seleção dos atributos não utiliza o algoritmo de DM e avalia um atributo por vez.

Em geral, o método *wrapper* tende a ser mais efetivo, ou seja, resulta em uma menor taxa de erro de classificação se comparado com o método do tipo filtro, mas este último normalmente é mais eficiente, consumindo menor tempo de processamento.

2.2.3 Discretização

O algoritmo de discretização divide os valores contínuos (inteiros ou reais) de um atributo em pequenas listas de intervalos. Efetivamente, converte atributos contínuos em categóricos, isto é, cada intervalo resultante é considerado como um valor discreto do atributo. Na Figura 2.3 (Freitas, 1998) é ilustrada a discretização do atributo idade. Na parte inferior da figura há uma lista ordenada de valores contínuos do atributo idade que foram discretizados em cinco intervalos.

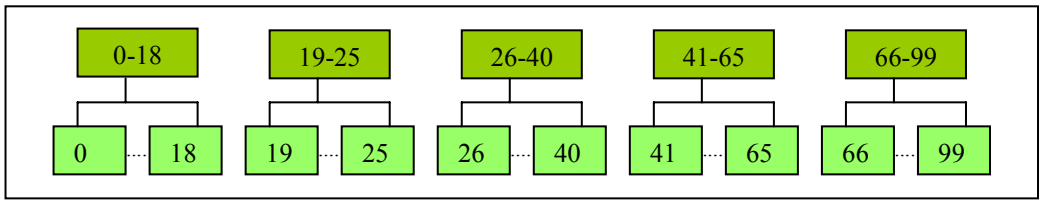

Figura 2.3. Exemplo de discretização do atributo Idade (Freitas, 1998)

Algoritmos de discretização podem ser classificados em dois grupos: cega para a classe e dirigida para a classe. Algoritmos cegos para a classe discretizam um dado atributo sem levar em consideração o valor do atributo meta. Algoritmos dirigidos para a classe consideram o valor do atributo meta quando discretizam um atributo. A entrada para este tipo de algoritmo é: o atributo a ser discretizado e o atributo meta. Os algoritmos de discretização de classe-dirigida trabalham melhor que os de classe-cega quando os intervalos resultantes

são utilizados em algoritmos de classificação.

Entre as vantagens da discretização estão: aumentar as possibilidades de algoritmos de DM serem aplicados a bases de dados, pois alguns algoritmos não tratam atributos contínuos; melhorar a compreensibilidade do conhecimento descoberto; reduzir o tempo de processamento de algoritmos de DM baseados em regras de indução.

A desvantagem da discretização é a possibilidade de perda de informações relevantes reduzindo a precisão da descoberta de conhecimento.

2.3 Algoritmos de Indução de Regras

Indução, em oposição a dedução, é o processo de se obter uma hipótese a partir dos dados e fatos já existentes. Indução pode ser explicada como sendo a conclusão de informações provenientes de dados; aprendizagem indutiva é o processo de construção de um modelo onde o ambiente, isto é, o banco de dados, é analisado para identificar padrões. Objetos semelhantes são agrupados em classes e regras formuladas por meio das quais é possível predizer a classe de objetos não vistos. Este processo de classificação identifica grupos nos quais cada qual tem um padrão único de valores que constitui a descrição da classe.

Aprendizagem indutiva de regras é o processo de observar uma série de dados e, a partir dela, gerar padrões. Pelo fato de explorar automaticamente a série de dados, o sistema indutivo cria hipóteses que conduzem a padrões. O processo é em sua essência semelhante àquilo que um analista humano faria em uma análise exploratória.

A indução de regras pode descobrir regras muito gerais, as quais lidam tanto com dados numéricos quanto não numéricos. Vários algoritmos de indução de regras têm sido propostos na literatura de aprendizagem de máquina, tipicamente distintos nos detalhes de como o processo de pesquisa é realizado. O algoritmo C4.5 (Quinlan - 1987, 1993) é o método mais conhecido para derivar regras em árvores de classificação. O algoritmo CN2 (Clark e Niblett, 1989) usa uma medida baseada em entropia para selecionar regras através de *beam search*. Outros recentes algoritmos de indução de regras, que fornecem precisão na classificação sobre grandes bases de dados, incluem: RL (Clearwater e Stern, 1991), Brute (Segal e Etzioni, 1994) e Ripper (Cohen, 1995). O algoritmo RISE (Domingos, 1996) é um interessante exemplo de algoritmo de indução de regras usando pesquisa heurística do *bottom-*

up. Holte (1993) descreve um interessante estudo no qual modelos de classificação de regras são comparados com outros classificadores.

2.4 Conclusão

Neste capítulo foram apresentadas técnicas de mineração de dados, suas tarefas, o processo de descoberta de conhecimento em banco de dados e alguns algoritmos de indução de regras propostos na literatura.

A seguir será apresentado a Mineração de Dados Distribuída, a qual considera o processo de MD no contexto de ambientes computacionais distribuídos.

Capítulo 3

Mineração de Dados Distribuída

Avanços na computação e comunicação aplicados a grandes bases de dados têm resultado em ambientes computacionais distribuídos, como a internet, *intranets* empresariais e redes *wireless* (sem fio). Muitos destes ambientes possuem diferentes recursos e volumosos dados distribuídos. A mineração de dados distribuída permite a extração de conhecimento destes ambientes, sendo aplicada em diferentes áreas como previsão do tempo, cotações financeiras, sensoriamento remoto, entre outros. Este capítulo apresenta a definição de mineração de dados distribuída, algoritmos, sistemas e algumas direções futuras de pesquisas nesta área.

3.1 Definição

A mineração de dados distribuída (DDM) é resultado da evolução da tecnologia de mineração de dados, sendo uma desta com sistemas distribuídos. DDM aceita o fato que dados estejam remotamente distribuídos entre diferentes locais interligados via rede.

Segundo Freitas (1998), mineração distribuída consiste em três fases:

- Dividir os dados para serem minerados em p subconjuntos de dados, onde p é o número de processadores disponíveis e enviar cada subconjunto para um processador distinto;
- Cada processador deve rodar um algoritmo de mineração sobre o subconjunto de dados local. Um processador pode rodar o mesmo algoritmo de mineração ou diferentes algoritmos;

- Combinar o conhecimento local descoberto por cada algoritmo de mineração em um conhecimento global consistente.

Tipicamente, cada algoritmo de mineração de dados trabalha independentemente um do outro. A Figura 3.1 (Freitas, 1998) abaixo ilustra todo o processo.

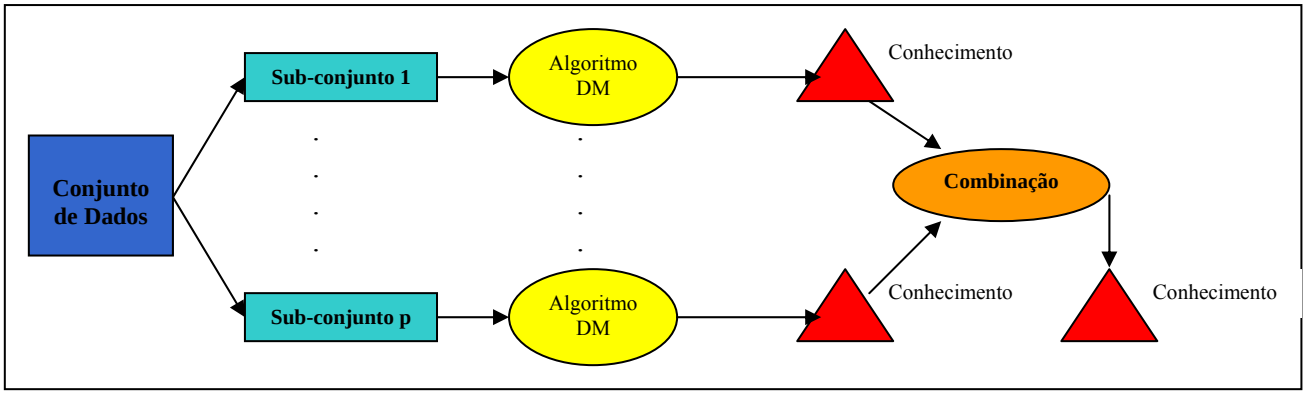

Figura 3.1. Estrutura básica de uma Mineração de Dados Distribuída (Freitas, 1998)

Na terceira fase do DDM, citada acima, o conhecimento local descoberto pelo algoritmo de mineração de dados, pode ser combinado de diferentes maneiras. Uma das maneiras mais fáceis é através da técnica de votação.

DDM pode ser utilizada para resolver diferentes tipos de problemas, como:

- *Extensibilidade*: suportam a inclusão de novas tecnologias de mineração;
- *Portabilidade*: capacidade de operar em diferentes ambientes ou plataformas;
- *Escalabilidade*: eficiência em minerar grandes volumes de dados sem perda de qualidade;
- *Eficiência*: determina a capacidade de utilizar corretamente os recursos disponíveis;
- *Compatibilidade*: integrar informações para bases de dados similares, mas com diferentes esquemas, para gerar modelos de dados mais precisos.

Muitos dos algoritmos e sistemas de DDM são projetados para modelos de dados relacionais podendo conter dados homogêneos ou heterogêneos.

As Tabelas 3.1 e 3.2 (Kargupta & Chan, 2000) ilustram um exemplo de dados homogêneos referentes à transação de cartões de crédito. Existem dois locais A e B, conectados por uma rede, sendo o objetivo da mineração de dados encontrar um padrão de transações fraudulentas.

Número da Conta	Valor	Localização	Registro Prévio	Transação Incomum
11992346	99,84	Curitiba	Bom	Não
12993339	29,33	Curitiba	Bom	Sim
45633341	34,98	Londrina	Ruim	Não
55564999	980,00	Maringá	Bom	Sim

Tabela 3.1 Caso Homogêneo: Local A - registro de transações de cartão de crédito

Número da Conta	Valor	Localização	Registro Prévio	Transação Incomum
87992364	20,00	Ponta Grossa	Bom	Não
67845921	447,34	Cascavel	Bom	Sim
85621341	19,78	Ponta Grossa	Ruim	Não
95345998	800,00	Foz do Iguaçu	Ruim	Sim

Tabela 3.2 Caso Homogêneo: Local B - registro de transações de cartão de crédito

Dados heterogêneos, ou seja, dados com características diferentes, também podem ser encontrados entre bases de dados. A Tabela 3.3 e 3.4 apresenta duas tabelas de dados de um determinado local X, sendo a primeira com dados de tempo e a segunda com dados demográficos. Na tabela 3.5 têm-se o conteúdo do local Y que contém dados de um guichê de cinema. O objetivo do processo de mineração de dados é detectar relações entre os ingressos de filmes vendidos e as características demográficas e de tempo de determinadas cidades.

Cidade	Temperatura	Umidade	Frio/Vento
Curitiba	20	55%	56
Londrina	27	40%	37
Maringá	29	50%	33
Ponta Grossa	22	60%	45
Florianópolis	30	94%	25

Tabela 3.3 Caso Heterogêneo: Local X - Dados do tempo

Cidade	Estado	Tamanho	Média Salarial	População Adolescente
Curitiba	PR	Grande	Alta	200
Londrina	PR	Média	Alta	250
Maringá	PR	Média	Média	125
Ponta Grossa	PR	Média	Média	100
Florianópolis	SC	Grande	Alta	270

Tabela 3.4 Caso Heterogêneo: Local X - Dados demográficos

Cidade	Filme	Avaliação	Vendas de Ingresso (em milhões)
Curitiba	Star Wars	A+	10
Londrina	Os Incríveis	B+	8
Maringá	O Refém	A -	5
Ponta Grossa	A Cruzada	A+	2
Florianópolis	Miss Simpatia	B-	12

Tabela 3.5 Caso Heterogêneo: Local Y - Dados vendas de ingressos de filmes

Muitas pesquisas têm sido feitas no desenvolvimento de algoritmos que utilizam as técnicas com o objetivo de se produzir técnicas escaláveis e computacionalmente mais eficientes de mineração de grandes conjuntos de dados. A seguir são apresentados alguns algoritmos de mineração de dados distribuída desenvolvidos com esse objetivo.

3.2 Algoritmos de Mineração de Dados Distribuída

Muitos algoritmos de DDM (Kargupta & Sivakumar, 2004 e Park & Kargupta, 2003) seguem uma estrutura funcional geral, onde é aplicada uma versão local do algoritmo para cada partição de dados, gerando um modelo local. Em seguida, todos os modelos locais são enviados para um único local para gerar um modelo global final. O processo de agregação dos modelos locais captura todas as características que podem ser detectadas em uma observação local e desenvolve modelos para comportamentos não conhecidos, através da transmissão de dados dos subconjuntos para o local central. Um dos objetivos dos algoritmos de DDM é minimizar o volume de dados transmitidos.

Algumas técnicas de mineração de dados distribuídos são orientadas a algoritmos, pois na busca do aumento de velocidade, modificam os algoritmos de mineração sem modificar os dados. Por outro lado, podem-se desenvolver métodos de mineração distribuída orientados a dados, que reduzem ou modificam os dados a serem minerados sem modificar o algoritmo.

Entre os algoritmos de DDM, classificadores distribuídos podem ser gerados a partir de dados homogêneos ou heterogêneos. Classificadores de aprendizagem para dados homogêneos são relatados por técnicas de aprendizagem conjunta (Bauer & Kohavi, 1999; Dietterich, 2000; Merz & Pazzani, 1999; Opitz & Maclin, 1999). A abordagem conjunta tem sido aplicada em vários domínios para melhorar a precisão da classificação. Nestas abordagens são produzidos múltiplos modelos (classificadores base) e combinados os resultados para atingir uma melhor precisão.

A estrutura de meta-aprendizagem (Chan & Stolfo, 1993b, 1993a, 1998) oferece outra possível abordagem de classificadores de aprendizagem para dados homogêneos distribuídos. A idéia básica de meta-aprendizagem é executar um número de processos de aprendizagem de máquina sobre um número de subconjuntos de dados em paralelo e combinar os resultados dos classificadores através de uma etapa adicional de aprendizagem. Inicialmente é realizada a tarefa de aprendizagem sobre cada subconjunto de dados, gerando para cada subconjunto um classificador base. Em seguida, uma tarefa de aprendizagem em separado, chamada meta-aprendizagem, integra todos os classificadores base processados em um meta classificador de alto nível, produzido a partir de um conjunto de treinamento meta-nível O conjunto de treinamento meta-nível é composto pelas predições dos classificadores bases individuais. Para todas as predições, o conhecimento das características e o desempenho dos classificadores bases são processados em um meta-classificador. A aprendizagem no meta-nível pode ser aplicada recursivamente, produzindo uma hierarquia de meta-classificadores. A meta-aprendizagem segue três passos principais:

- Gerar classificadores base para cada subconjunto de dados usando um algoritmo de aprendizagem;
- Unir os classificadores base em um único local e produzir meta-níveis de dados;
- Gerar um classificador final (meta-classificador) à partir dos meta-níveis de dados.

Duas técnicas de meta-aprendizagem (para combinar as predições locais geradas pelos algoritmos de mineração de dados locais) podem ser empregadas: abordagem do *árbitro* e do *combinador*.

A abordagem do *árbitro* faz uso de um classificador especial, chamado árbitro, para decidir a classe final de predições para uma dada característica. O árbitro gera um classificador executando um algoritmo de aprendizagem sobre instâncias difíceis de classificar com os classificadores base. A classificação é feita sobre a predição dos algoritmos de aprendizagem locais e a predição do árbitro. Estas predições são combinadas, retornando a maioria de ocorrências, com preferência dada às predições do árbitro, em caso de empate.

Na abordagem do *combinador*, a meta-aprendizagem aceita como entrada um conjunto de dados (treinamento) contendo as predições feitas por cada algoritmo de aprendizagem local e as predições corretas (contidas nos dados de treinamento e previstas pelo algoritmo de mineração local) para cada tupla correspondente de dados. Outras informações, tais como os valores dos atributos, podem também ser adicionados aos conjuntos de dados da meta-aprendizagem, dependendo da estratégia adotada para implementar a meta-aprendizagem. A meta-aprendizagem então usa estes dados para descobrir a relação entre as predições feitas pelos algoritmos locais e as predições corretas.

Meta-aprendizagem contribui para a redução do problema de escalabilidade em aprendizagem de máquina, pois melhora a eficiência dos processos de aprendizagem dos classificadores bases, os quais são executados em paralelo. A técnica pode ser considerada escalável, porque o meta-classificador é a combinação de classificadores base gerados a partir de pequenos subconjuntos de dados.

Segundo Chan (1998), o uso de métricas ou propriedades diferentes qualificam o melhor classificador. A combinação de classificadores considerados melhores em um meta-classificador podem, provavelmente, formar classificadores com maior precisão e eficiência, sem utilizar buscas exaustivas em um espaço inteiro de possibilidades. No entanto, Freitas (1998) afirma que a precisão da predição conseguida com técnicas de meta-aprendizagem tende a diminuir quando o número de subconjuntos de dados aumentar, a menos que ocorra um aumento na quantidade de dados contidos em cada subconjunto. Além disso, técnicas de meta-aprendizagem podem reduzir a compreensibilidade do conhecimento descoberto.

Em projetos de algoritmos de aprendizagem distribuídos considera-se que a precisão é a capacidade do algoritmo em classificar corretamente instâncias de diferentes subconjuntos de dados, e cobertura é a avaliação do desempenho do algoritmo sobre todos os conjuntos de dados, ou seja, a capacidade de alcance de uma regra sobre estes exemplos.

A Figura 3.2 (Freitas, 1998) ilustra um exemplo de meta-aprendizagem para *sites* distribuídos com dados homogêneos.

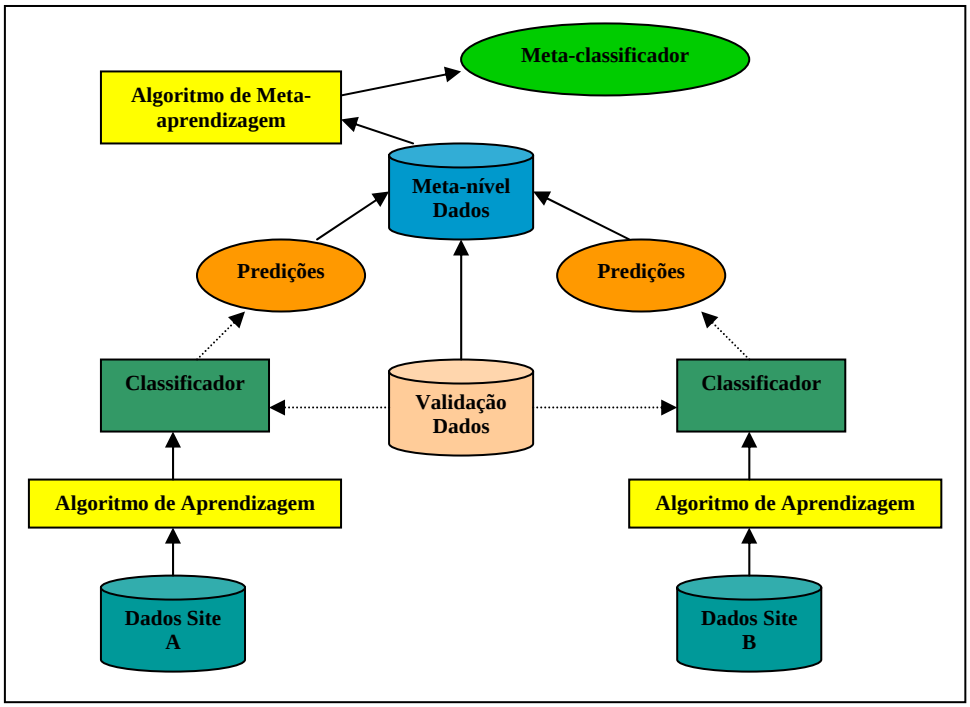

Figura 3.2 Meta-aprendizagem para dados homogêneos de *sites* distribuídos (Freitas, 1998)

A meta-aprendizagem se beneficia de duas características dos algoritmos de DDM: paralelismo e redução na comunicação. Todos os classificadores bases são gerados em paralelo e agrupados em um local central juntamente com o conjunto de validação; a comunicação passa a ser insignificante se comparada a taxa de transferência dos dados de cada *site*.

Algoritmos de DDM devem ser capazes de aprender a partir de diferentes características (dados heterogêneos) observadas de diferentes *sites* sem a necessidade de efetuar um *download* de todos os dados em um único local.

Em algumas aplicações de DDM heterogêneos não há a necessidade de detectar interações entre características de diferentes *sites*. Nestes casos pode-se utilizar uma abordagem baseada em conjunto. Nos casos gerais de DDM heterogêneos pode ser necessária a construção de classificadores usando interação não-linear das características dos diferentes *sites*.

Existem poucos algoritmos de DDM que utilizam um conjunto de classificadores para aprender sobre dados heterogêneos. Alguns desses algoritmos são baseados em agregação. A

técnica de agregação proposta em Tumer e Ghosh (2000) utiliza uma abordagem baseada em uma seqüência estatística para combinar modelos locais de alta variância gerados a partir de dados heterogêneos. A técnica trabalha ordenando as predições de diferentes classificadores utilizando-os de uma maneira apropriada.

Park e colaboradores (2002) desenvolveram uma técnica de aprendizagem em árvores de decisão para dados heterogêneos distribuídos. Esta abordagem pode ser classificada como uma abordagem baseada em conjuntos. Ela apresenta melhor performance que modelos de agregação simples mas sua performance é sensível a dimensionalidade dos exemplos.

Um algoritmo de aprendizagem Bayesiana cooperativa distribuída foi desenvolvida em Yamanishi (1997) a qual considera conjuntos de dados homogêneos. Uma "abordagem fragmentada" para minerar classificadores de recursos de dados distribuídos é sugerida por Cho e Wuthrich (1998). Neste método uma única regra boa é gerada em cada fonte de dados distribuídos. Estas regras são então classificadas usando algum critério e um número de regras no topo da classificação são selecionadas para formar o conjunto de regras. Lam e Segre (1997) propõem uma técnica para produzir automaticamente uma rede de crença Bayesiana para descobrir conhecimento usando uma abordagem distribuída. Outros trabalhos tratam da otimização de projetos de DDM (Turinsky e Grossman, 2000), poda do classificador (Prodromidis e Stolfo, 2000), avaliação da qualidade das fontes de dados distribuídos (Wuthrich, Cho, Pun e Zhang, 2000) e decomposição do problema e seleção de um modelo local em DDM (Pokrajac *et al.,* 1999).

3.3 Sistemas de Mineração Distribuída de Dados

Sistemas de DDM manuseiam diferentes componentes: algoritmos de mineração, subsistemas de comunicação, gerenciamento de recursos, planejamento de tarefas, interfaces com o usuário, entre outros. Isto permite um acesso eficiente aos dados e recursos computacionais distribuídos, monitoração do procedimento de mineração e apresentação dos resultados aos usuários de forma apropriada. Um sistema DDM de sucesso deve ser flexível o suficiente para adaptar-se a várias situações. Deve identificar dinamicamente a melhor estratégia de mineração de acordo com os recursos e fornecer uma maneira fácil de atualizar seus componentes.

Entre as arquiteturas propostas na literatura têm-se a cliente-servidor e a baseada em agentes. Todos os sistemas DDM baseados em agentes possuem um ou mais agentes em cada

base de dados. Estes agentes são responsáveis por analisar os dados locais e pró-ativamente comunicar-se uns com os outros durante o etapa de mineração, trocando conhecimentos locais até chegar-se a um conhecimento coerente global. Como é difícil um controle total sobre os recursos remotos, muitos sistemas baseados em agentes utilizam um agente supervisor, chamado de facilitador, que controla o comportamento dos agentes locais. Agentes Java para meta-aprendizagem (JAM) (Stolfo *et al.* 1997) e o sistema BODHI (Kargupta, Park, *et al.,* 2000) seguem esta abordagem.

3.4 Desafios em Mineração Distribuída de Dados

Em geral, técnicas de mineração de dados têm sido projetadas para aplicações de suporte a decisões off-line. A próxima geração desses sistemas deverá suportar aplicações de monitoramento em tempo real, o que irá requerer uma tecnologia de mineração de dados mais voltada à computação distribuída, comunicação e o gerenciamento dos recursos no ambiente, ou seja, DDM torna-se uma área cada vez mais promissora.

Muitas organizações estão utilizando aplicações de DDM em diferentes áreas, incluindo mineração de dados financeiros a partir de dispositivos móveis (Kargupta *et al.*, 2002), bases de dados distribuídas baseadas em redes com sensores (Bonnet, Gehrke e Seshadri, 2001), e análises de diagnósticos médicos (Wirth, Borth e Hipp, 2001).

Contudo, DDM ainda possui muitas questões a serem estudadas. Primeiramente, aplicações do mundo real tratam com cenários de dados distribuídos que podem ser vistos como diferentes do sentido tradicional de homogêneos e heterogêneos. Podem conter dados heterogêneos que compartilham mais de uma coluna ou não ter uma chave bem definida para múltiplas tuplas de dados. Portanto, maiores estudos sobre algoritmos para cenários heterogêneos e pré-processamento de dados sobre meta-dados são necessários.

DDM freqüentemente requer a troca de modelos de mineração de dados entre partições, requerendo um esquema de padronização para representar e efetuar a trocar modelos. A combinação de modelos de mineração de dados para *sites* de busca, como *Yahoo*! e *Google*, torna-se uma interessante aplicação de DDM a ser desenvolvida.

Finalmente, questões de interação homem-máquina em DDM requerem suporte a nível de sistema para interações em grupo, resolução de problemas de forma colaborativa, desenvolvimento de interfaces alternativas para equipamentos móveis e o tratamento de questões de segurança.

3.5 Conclusão

Neste capítulo foram apresentadas a definição de Mineração de Dados Distribuída, algoritmos, sistemas e as direções futuras de pesquisas nesta área.

Como discutido anteriormente, algoritmos de mineração de dados são complexos e técnicas de Mineração de Dados Distribuídas devem ser utilizadas. Nós acreditamos que algumas das técnicas propostas em Inteligência Artificial Distribuída podem ser aplicadas à Mineração de Dados Distribuída com o objetivo de reduzir a complexidade necessária para o treinamento e ao mesmo tempo garantir resultados de boa qualidade. No capítulo seguinte, discutimos algumas dessas abordagens introduzindo também alguns conceitos fundamentais sobre Sistemas Multi-agente e Resolução Distribuída de Problemas. Estes conceitos servirão como base para o sistema que será apresentado no capítulo seguinte.

Capítulo 4

Inteligência Artificial Distribuída

Inteligência Artificial pode ser definida como o ramo da ciência da computação que está interessada em projetar sistemas computacionais inteligentes. Um dos ramos da Inteligência Artificial (IA), a Inteligência Artificial Distribuída (IAD) está interessada em compreender e modelar ações e conhecimento nos quais vários sistemas interagem para o alcance de um objetivo. Duas áreas fundamentais são distinguidas na IAD: Resolução Distribuída de Problemas e Sistemas Multi-agente. A resolução distribuída de problemas está interessada em como solucionar um determinado problema através da divisão de tarefas em módulos que cooperam entre si, produzindo resultados que, através de uma coordenação centralizada, irão compor um resultado global. Sistemas multi-agente estão interessados no comportamento individual e coletivo de unidades computacionais resolvedoras de problemas (agentes), sem a existência de uma coordenação centralizada. Este capítulo apresenta a inteligência artificial distribuída, resolução distribuída de problemas e sistemas multi-agente.

4.1 Definição

A união da Inteligência Artificial (IA) com técnicas de Sistemas Distribuídos forma a Inteligência Artificial Distribuída (IAD). A IAD tornou-se um ramo de estudo recentemente. Em meados de 1980 foi realizado nos Estados Unidos, no *Massachusetts Institute of Technology (MIT),* o primeiro encontro de IAD para discutir questões relacionadas com a solução inteligente de problemas com sistemas consistindo de múltiplos solucionadores de problemas. Muitos artigos de grande repercussão resultaram das idéias apresentadas neste evento, como o artigo escrito por Nilsson em 1981 que descreveu a IAD como uma rede de sistemas livres (Wittig, 1992). Outro artigo da época foi escrito por Smith e Davis (Smith &

Davis, 1981), tratando da cooperação em Resolução de Problemas Distribuídos (RPD) e definindo a terminologia no contexto de tais sistemas (Wittig, 1992).

Em 1987, Sridharam produziu uma primeira taxonomia da IAD. Esta divisão baseia-se em oito dimensões diferentes: modelo, agentes, construção, uniformidade, recursos e interações. Definindo-se simplificadamente cada item, temos:

- *Modelo*: indica como o nível de distribuição é aplicado ao sistema, e como são classificados os agentes em termos de sua complexidade;
- *Granularidade*: mostra qual é o nível de interação entre os agentes;
- *Escala*: é a medida do número de agentes que compõe a comunidade;
- *Agentes*: indica qual é o grau de autonomia dos agentes em relação a comunidade;
- *Construção*: indica como o problema foi dividido entre os agentes;
- *Uniformidade*: define o grau de homogeneidade dos agentes; heterogeneidade significa ter agentes em diferentes formatos ou construídos sob diferentes metodologias.
- *Recursos*: define como os recursos são distribuídos pelo ambiente; podem ser considerados aqui desde equipamentos até tempo;
- *Interações*: aponta o nível de interação entre os agentes; depende do mecanismo de coordenação escolhido.

Bond e Gasser (1988) sugerem ainda alguns benefícios para utilização de IAD, de acordo com os seguintes critérios:

- *Adaptabilidade*: sistemas de IAD são mais apropriados para lidar com problemas distribuídos em termos espaciais, lógicos, temporais ou semânticos;
- *Custo*: um grande número de pequenas unidades computacionais pode ser mais interessante quando os custos com comunicação não são relevantes;
- *Desenvolvimento e Gerenciamento*: a inerente modularidade do sistema permite o desenvolvimento de partes do mesmo separadamente, garantindo a continuidade do ambiente;
- *Eficiência e Velocidade*: a concorrência e a distribuição de processos em diferentes computadores podem aumentar a velocidade de computação e raciocínio, desde que haja um nível de coordenação aceitável;
- *História*: a integração de recursos distribuídos, tais como redes de estações de trabalho ou diferentes domínios de especialistas;

- *Isolamento / Autonomia*: o controle de processos locais pode ser encarado como uma maneira de proteção ou de aumento da segurança do sistema;
- *Naturalidade*: alguns problemas são mais bem descritos em termos distribuídos;
- *Confiabilidade*: os sistemas distribuídos podem exibir um grau maior de confiabilidade e de segurança do que sistemas centralizados, pois podem prover redundância (um agente é capaz de realizar o que outro agente faz se necessário), múltiplas verificações, "triangulação de resultados", etc.
- *Limitação de recursos*: os agentes computacionais individuais ligados a recursos escassos podem cooperar em busca da resolução de problemas complexos e;
- *Especialização*: o conhecimento e as ações podem ser escolhidos de acordo com o domínio do agente.

Uma conceitualização de IAD mais compatível com a idéia de agente é dada por Jennings (1996) que diz que o objeto de investigação da IAD são os modelos de conhecimento, e as técnicas de comunicação e raciocínio necessárias para que agentes computacionais convivam em sociedades compostas de computadores e pessoas. Jennings ainda divide a IAD em duas áreas de pesquisa principais:

- *Resolução Distribuída de Problemas (RPD)* - que divide a solução de um problema em particular, entre um número de módulos que cooperam compartilhando conhecimento sobre o problema e sobre as soluções envolvidas.
- *Sistemas Multi-agente (SMA)* - estuda o comportamento de um conjunto de agentes autônomos (possivelmente pré-existentes) cujo objetivo comum é a solução de um dado problema.

Em SMA a atenção do projetista não está necessariamente voltada para um problema específico. Esta abordagem consiste na coordenação do comportamento inteligente de um conjunto de agentes autônomos, que podem ter sido criados antes do surgimento de um problema em particular.

É possível perceber que estas duas áreas diferem na forma de construção da solução do problema. A Resolução Distribuída de Problemas adota uma visão *top-down* dividindo o problema em partes que corresponderão a módulos computacionais, sendo o processo de coordenação das ações definido ainda em tempo de projeto. Os sistemas multi-agente são compostos por entidades computacionais, denominadas agentes, com capacidades e objetivos individuais que, uma vez agrupados em sociedade, trabalham juntos visando atingir o objetivo

do sistema, sendo que os agentes devem raciocinar a respeito das ações e do processo de coordenação em si.

4.2 Agentes

Uma variedade de definições de agentes tem sido apresentada por investigadores da área de IAD. Abaixo serão citadas algumas destas definições.

Em Russel (1995) define-se:

> "*Um agente é uma entidade que pode perceber o seu ambiente através de sensores e agir sobre este ambiente através de atuadores*."

Em Maes (1995) temos:

> "*Agentes autônomos são sistemas computacionais, os quais inseridos num ambiente dinâmico e complexo, percebem e atuam automaticamente neste ambiente, e fazendo-o, compreendem um conjunto de objetivos ou tarefas para as quais foram projetados*".

Em Hayes-Toth (1995) é apresentado:

> *"Agentes inteligentes executam de forma contínua três funções: (i) perceber as condições dinâmicas do ambiente, (ii) agir para afetar as condições do ambiente e, (iii) raciocinar para interpretar as percepções, resolver problemas, inferir e determinar ações".*

Wooldridge e Jennings (1995) visualizam um agente como sendo uma entidade com capacidade de resolução de problemas encapsulada. Inserido nesta visão, define o agente como tendo as seguintes propriedades:

- *autonomia* - executam a maior parte de suas ações sem interferência direta de agentes humanos ou de outros agentes computacionais, possuindo controle total sobre suas ações e estado interno;
- *habilidade social* - por impossibilidade de resolução de certos problemas ou por outro tipo de conveniência, interagem com outros agentes (humanos ou computacionais), para completarem a resolução de seus problemas, ou ainda para auxiliarem outros agentes. Disto surge a necessidade de que os agentes tenham capacidade para comunicar seus requisitos aos outros e um mecanismo decisório interno que defina quando e quais interações são apropriadas;
- *reatividade* - percebem e reagem à alterações no ambiente em que estiverem inseridos.

- *pró-atividade* - agentes, do tipo deliberativo, além de atuar em resposta às alterações ocorridas em seu ambiente, apresentam um comportamento orientado a objetivos, tomando iniciativas quando julgarem apropriado.

4.2.1 Autonomia

Entende-se autonomia como a capacidade do agente executar as suas atividades sem a necessidade de intervenção humana, possuindo certo grau de inteligência que capacita-o a sobreviver em um ambiente dinâmico.

Segundo Russel (1995), para definir um agente racional ideal é necessário considerar seu "conhecimento embutido". No caso das ações do agente basearem-se completamente no conhecimento embutido, ao ponto que este não considere suas percepções, podemos dizer que o agente tem falta de autonomia. O comportamento de um agente pode ser baseado tanto em sua própria experiência, quanto em seu conhecimento embutido (inicial) usado na construção do agente para um ambiente em particular. Assim, podemos concluir que o grau de autonomia de um agente está diretamente relacionado ao nível de independência que um agente tem em relação àquilo que lhe é nato (ou, "instintivo").

4.2.2 Categorias de agente

Segundo Jennings (1996), para que possa agir de maneira autônoma, agentes podem ter várias habilidades: percepção e interpretação de mensagens, raciocínio baseado em crenças, tomada de decisão, planejamento, e habilidade para executar planos incluindo passagem de mensagens. Jennings categoriza os agentes quanto ao nível de capacidade de resolução de problemas:

- *reativos* - reagem a alterações no ambiente ou a mensagens de outros agentes. Não têm capacidade de raciocínio sobre suas intenções, reagindo tão somente sobre regras e planos estereotipados. Suas ações podem ser: atualizar a base de fatos e enviar mensagens para outros agentes ou para o ambiente.
- *intencionais* - tem a habilidade de raciocínio sobre suas intenções e crenças, e criar e executar planos de ações. São considerados como sistemas de planejamento selecionando objetivos de acordo com suas motivações e raciocinar sobre estes, detecção e resolução de conflitos e coincidências de objetivos, selecionar e criar planos (agendamento de ações), detecção de conflitos entre planos (alocação de recursos), e, se necessário, executar e revisar planos.

- *sociais* - agentes intencionais são considerados sociais quando possuem modelos de outros agentes, sobre os quais raciocinam para tomar decisões e criar planos.

4.2.3 Arquiteturas de Agentes

Para definir arquiteturas internas dos agentes é necessário, primeiramente, saber qual o tipo de agente do qual se está tratando. Neste caso, um agente pode ser classificado como cognitivo ou reativo.

Um agente reativo não usa raciocínio simbólico complexo, estruturas de memória e uma representação interna explícita do conhecimento. Assim, não possui um histórico de suas ações passadas, nem pode fazer previsão de atos futuros. Com estas restrições este agente somente percebe o ambiente externo e, baseado nos estímulos do ambiente, reage de uma forma pré-determinada.

Agentes cognitivos são capazes de raciocinar a respeito de suas intenções e conhecimentos, criar planos de ação e executá-los. Possuem modelos explícitos do mundo externo, estruturas de memória que permitam manter um histórico de ações passadas e fazer previsões de ações futuras, e um sistema desenvolvido de cooperação e comunicação. O que diferencia do estímulo-resposta é que agentes cognitivos possuem controle deliberativo, na medida em que conseguem deliberar qual ação será executada, e posteriormente executá-la.

As arquiteturas dos agentes podem ser:

- *Arquiteturas deliberativas ou cognitivas* são aquelas que baseiam seu processo decisório em raciocínio lógico (explícito) que opera sobre uma representação simbólica (interna ao agente) do mundo. Não há consenso de que capacidades de planejamento sejam necessárias para que uma arquitetura seja considerada deliberativa, mas existem classificações que definem esta restrição. Neste modelo, encontram-se as noções dos modelos de si, dos outros, de crenças, de intenções, de atos da fala, etc. Estes modelos permitem implementar métodos complexos de cooperação e de negociação.
- *Arquiteturas reativas* definem que o processo de tomada de decisão ocorre em resposta a estímulos do ambiente ou por mensagens enviadas por outros agentes, tendo como base um conjunto de regras evento-ação. O comportamento inteligente do sistema resulta da coexistência e da combinação de comportamentos simples; os

agentes reativos utilizam constantemente o ambiente para se comunicar, dispondo de um protocolo de comunicação e de uma linguagem de comunicação reduzida.

- *Arquiteturas híbridas* adotam a idéia de combinar características reativas à uma arquitetura deliberativa, ou vice-versa. São provenientes das deficiências encontradas nas arquiteturas deliberativas e reativas, reunindo propriedades de ambas. As arquiteturas reativas têm dificuldades para modificar seus planos de ação a partir do momento em que a situação passa a divergir de seus objetivos iniciais. Por outro lado, arquiteturas deliberativas têm dificuldade de lidar com situações imprevistas ou que exigem decisões rápidas.

4.2.4 Cooperação entre Agentes

A cooperação entre agentes passa em geral pela troca de informações. Estas trocas se dão pelo compartilhamento de tarefas ou de resultados parciais (Smith & Davis, 1981). O compartilhamento de tarefas é realizado com o objetivo de balancear a carga computacional de um sistema, à medida que um problema global é dividido em subproblemas e cada um destes é alocado a um agente específico do sistema. No compartilhamento de tarefas, o controle é tipicamente guiado por objetivos, *i.e.*, o processamento efetuado por um agente visa realizar um objetivo, cujo resultado será utilizado para resolver um problema global. Enquanto que, no compartilhamento de resultados parciais, o controle é tipicamente guiado por dados, *i.e.,* o processamento efetuado em dado instante por um agente depende dos dados que ele tem disponível localmente ou externamente.

Segundo Durfee (1987), a cooperação entre agentes deve ser feita levando em conta os objetivos visados, por exemplo:

- acelerar a solução de um problema, privilegiando o trabalho paralelo dos agentes;
- obter várias soluções locais, utilizando as capacidades dos outros agentes para obter uma solução global;
- melhorar a confiabilidade dos resultados, utilizando o fato que os agentes podem verificar os resultados;
- reduzir a possibilidade de duplicação de processamento, atribuindo um subproblema a um número limitado de agentes, por exemplo a um único;
- reduzir o volume de comunicação, trocando apenas as informações necessárias.

Um ponto importante é que para implementar esses modos de cooperação, existe a necessidade de protocolos de comunicação relativamente elaborados. Davis e Smith (1983) utilizam a metáfora da negociação de contrato para propor um protocolo de alto nível, o *Contract-Net*. A negociação é uma discussão na qual os agentes interessados trocam informações a fim de chegar a um acordo sobre um dado serviço. Deve-se salientar que os conhecimentos para a atribuição de um contrato variam de uma aplicação para outra.

4.3 Resolução Distribuída de Problemas

Na Resolução Distribuída de Problemas (RDP) - (*Distributed Problem-Solving* - DPS) módulos computacionais (agentes) são designados para resolverem um problema em particular, dentro de uma concepção fechada de mundo. Desta maneira, o número de agentes será fixo, sendo que cada agente possui uma visão específica e incompleta do problema. Então, para a resolução de um problema, os agentes devem cooperar entre si, compartilhando conhecimento sobre o problema e sobre o processo de obter uma solução.

O projeto de um sistema RDP é realizado por um projetista que, primeiramente, realizará uma análise do problema a ser resolvido para, então, identificar os agentes necessários para a solução desse problema. Desta maneira, a tarefa de resolução será decomposta entre os vários agentes, buscando melhorar o processamento do sistema através da execução paralela.

Segundo Durfee (1987), o compartilhamento de tarefas entre agentes compreende as seguintes etapas: decomposição da tarefa - decompor a tarefa em sub-tarefas; alocação da tarefa - designar as sub-tarefas aos agentes competentes; realização da tarefa - realização da sub-tarefa pelo agente ou novamente a decomposição da mesma em sub-tarefas a serem realizadas por outros agentes; e síntese dos resultados - os resultados parciais das sub-tarefas irão compor o resultado global.

Na resolução distribuída de problemas, fica evidenciado um controle distribuído com respeito à construção do plano e não com respeito à execução. Da mesma forma, evidencia-se que o processo de coordenação das ações dos agentes é definido em tempo de projeto, dessa forma, os agentes não existem antecipadamente, ou seja: sua concepção, organização e interação são idealizadas face à existência de um problema a ser solucionado. Sendo assim, a resolução do problema é distribuída entre os agentes que cooperam entre si, dividindo e

compartilhando conhecimentos sobre o processo de obtenção da solução. Na Figura 4.1 (Sichman, 1995) pode ser vista a representação de um sistema RPD.

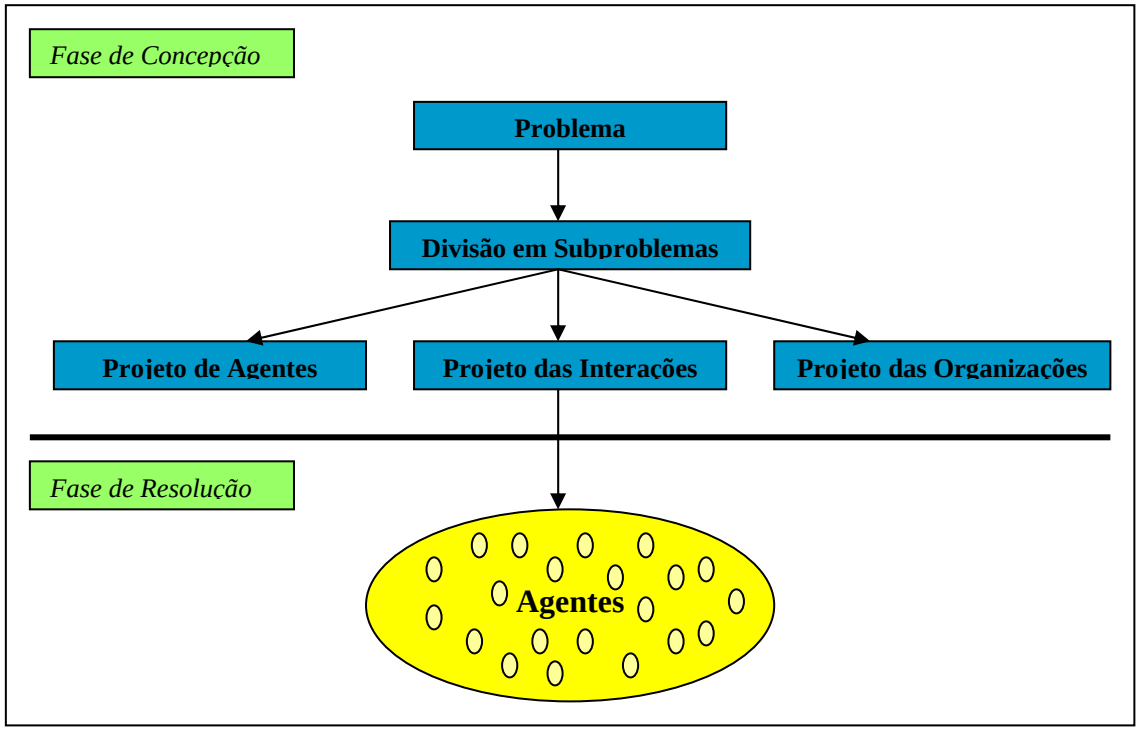

Figura 4.1 Estrutura de um sistema RDP (Sichman, 1995)

A estratégia de resolução, na RDP, apresenta as seguintes características:

- O problema é resolvido por um conjunto de agentes previamente desenvolvidos de forma a solucionar um determinado problema;
- É estipulada uma organização para restringir o comportamento de tais agentes. Esta organização é, na maioria dos casos, definida durante a concepção do sistema;
- A interação entre agentes ocorre por meio de intercambio de mensagens ou por compartilhamento de dados comuns. Esses procedimentos são quase sempre definidos durante a concepção do sistema e são intimamente ligados ao modelo algorítmico subjacente (como por exemplo, o quadro-negro) e ao problema que o sistema deve resolver;
- De forma a aumentar a velocidade de resolução, os agentes são executados de forma concorrente;

- A cooperação pode existir a nível de subproblemas, ou seja, os agentes dividem entre si as diversas partes do problema original (tarefas), ou podem aplicar diferentes estratégias para a resolução de uma mesma tarefa;
- Existe a noção de controle global, na maior parte dos casos, o qual garante um comportamento global coerente do sistema, conforme a organização inicialmente prevista.

Em alguns sistemas RDP parte-se de um problema bem definido e um controlador exerce uma distribuição das tarefas, atribuindo partes do problema aos agentes componentes do sistema. Nesse tipo de sistema, o conhecimento é completo apenas no agente central, enquanto que os demais agentes possuem apenas conhecimento completo sobre suas próprias tarefas e sobre os mecanismos com que foram dotados para executá-las com eficiência; ou seja, eles não possuem uma visão global do problema nem do sistema, visto que não participam do planejamento nem da síntese de resultados.

Os sistemas RDP apresentam um controle centralizado e guiado externamente (trabalha em função de requisições) em busca dos objetivos, controle este determinado pela transmissão de objetivos e tarefas de modo explícito. Ou seja, quando o agente central envia uma requisição a um outro agente, este último executará alguma ação como resposta, no sentido de cumprir a requisição. Sendo assim, os agentes possuem menos autonomia na hora de processar suas estratégias e, conseqüentemente, seu comportamento será pouco flexível.

Além de tudo, os sistemas RDP fazem uso de comunicação primitiva, ou seja, a comunicação é restrita a um conjunto pré-definido de sinais com uma interpretação fixa, o que, devido às restrições impostas pela quantidade e tipos de sinais limita a cooperação entre agentes.

Pode-se dar exemplos de alguns problemas resolvidos por RDP:

- Agentes móveis resolvem o problema da transmissão na rede cliente/servidor. Numa rede, a largura da banda é importante e às vezes os recursos são raros. A transação solicitada entre um cliente e um servidor pode requerer muitas voltas sobre a rede para ser completada, o que cria um tráfego muito grande e consome grande parte da banda, tornando a transmissão de dados muito ruim. O fluxo na rede torna-se reduzido na utilização de agentes para buscar as solicitações ou transações e enviar informações do cliente para o servidor. Desta maneira, somente

o que os agentes encontrarem será transmitido pela rede, tornando a velocidade de transmissão sensivelmente maior.

- A arquitetura de agentes também pode resolver problemas criados por intermitência ou má qualidade da conexão com a rede. Atualmente, algumas aplicações na rede são muito pesadas para completar a transação ou obter localização de informação. Se a conexão cair, o cliente deve freqüentemente reiniciar a transação do ponto de partida. Com a tecnologia de agentes o cliente poderá obter as informações, mesmo que a conexão não esteja ativa. O cliente pode trabalhar off-line. Os agentes podem completar as transações e retornar os resultados para o cliente quando for restabelecida a conexão.

Portanto, a Resolução Distribuída de Problemas parte de um problema bem definido dividido entre os agentes componentes do sistema, sendo o processo de controle centralizado e guiado em busca dos objetivos. Diferentemente, nos sistemas multi-agente estuda-se a coordenação da conduta inteligente entre um grupo de agentes inteligentes autônomos.

4.4 Sistemas Multi-agente

Um sistema multi-agente, segundo os trabalhos de Jennings, Sycara e Wooldridge (1998) e Wooldridge (2000), é um programa de computador com solucionadores de problemas localizados em ambientes interativos, que são capazes de ações flexíveis, autônomas e, ainda, socialmente organizadas que podem, mas não necessariamente, ser dirigidas para objetivos pré-determinados ou metas.

Existem diversos tipos de agentes, e eles podem ser: estacionários ou móveis, persistentes ou temporários, reativos ou cognitivos (Ferber, 1999).

Agentes estacionários são agentes de software que, uma vez lançados em um dado ambiente computacional, não tem a habilidade de se moverem pela rede para outros computadores. Agentes móveis são agentes de software que podem se mover para outros ambientes através da rede e, quando se movem, levam consigo seus estados internos, ou seja, sua representação e sua memória (Rabelo, 2002).

Agentes persistentes são agentes de software que, uma vez lançados em um dado ambiente computacional, não podem ser excluídos do sistema. Agentes temporários são agentes de software que tem uma vida finita, normalmente de duração igual ao tempo de uma

dada tarefa, ou seja, tão logo eles finalizam sua missão eles são excluídos do sistema, normalmente por eles próprios (Ferber, 1999).

O projeto de um sistema multi-agente deve considerar vários aspectos, como por exemplo, tipo ou classe do problema a ser tratado, níveis de autonomia dos agentes, representação do conhecimento, protocolos de comunicação, definição da organização do sistema, modelagem de dados, modelos de coordenação, cooperação e resolução de conflitos (Rabelo, 2002).

A organização de um sistema multi-agente pode ser classificada como: democrática, federada ou hierárquica. Na organização democrática os agentes não possuem organização, e atuam em graus similares de independência e autonomia, sem hierarquia alguma. Neste caso são necessários procedimentos de coordenação sofisticados para garantir convergência e evitar interações desnecessárias.

Na organização federada existe algum tipo de hierarquia. Há a presença de agentes facilitadores, agentes intermediários entre o cliente e o supervisor. Grupos de agentes da federação têm graus similares de independência e autonomia, e normalmente são bastante heterogêneos. Os procedimentos de coordenação são de média complexidade dado a própria existência do facilitador. Entretanto, as arquiteturas hierárquicas são as mais utilizadas em aplicações industriais (Rabelo, 2002).

Uma das propriedades essenciais de um agente é a sua capacidade de comunicar-se com outros agentes, usuários e sistemas visando atingir seus objetivos, a interação. Durante o planejamento da interação deve-se lembrar que a informação pode ser incompleta, imprecisa e/ou previsível, e a qualidade dela varia de acordo com o tipo de agente. A interação pode ser dividida em 4 camadas de complexidade: comunicação, coordenação, cooperação e colaboração (Wortmann e Szirbik, 2001)

A camada comunicação é básica de qualquer software que precisa interagir. A camada coordenação define as regras de interação considerando as agendas dos agentes de forma a se evitar comportamentos indesejados. A camada cooperação é uma camada encontrada apenas em sistemas onde a cooperação reflete uma estratégia de ação decidida pelo agente, permitindo a negociação. A camada mais refinada é a camada colaboração onde um agente tem capacidade de detectar possíveis objetivos comuns, e de planejar sua agenda com os outros de forma a atingir o objetivo da melhor forma possível, aproveitando ao máximo a partilha de informações.

Sistemas multi-agente são ideais para representar problemas que incluem muitos métodos de solução de problemas, múltiplos pontos de vista e múltiplas entidades. Nestes domínios, os sistemas multi-agente oferecem as vantagens da solução de problemas concorrente e distribuída, juntamente com as vantagens dos esquemas sofisticados de interação. Exemplos de interação incluem a cooperação no trabalho para alcançar um objetivo comum, a coordenação na organização das atividades da solução do problema, de modo que interações prejudiciais sejam evitadas e possibilidades benéficas sejam exploradas, e a negociação de restrições de subproblemas, de modo que se alcance um desempenho satisfatório. É a flexibilidade destas interações sociais que distingue os sistemas multi-agente dos softwares mais tradicionais e que confere poder e atratividade ao paradigma de agentes.

A Figura 4.2 mostra a estrutura de um SMA segundo a concepção de Sichman (1995).

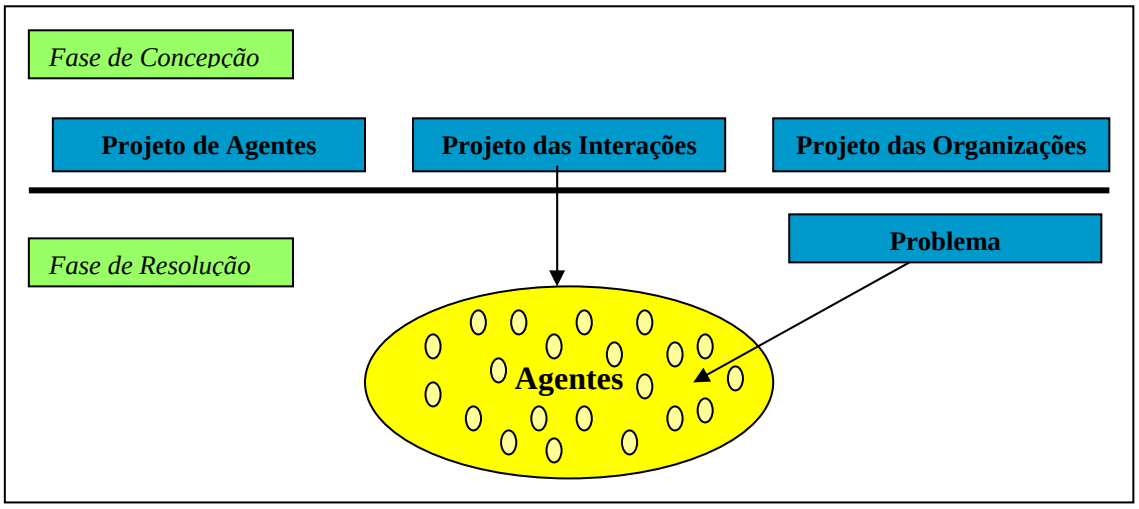

Figura 4.2 Estrutura de um SMA (Sichman, 1995)

Segundo Sichman (Sichman *et al.,* 1992) os principais problemas encontrados na abordagem SMA são os seguintes:

- *Descrição, decomposição e alocação de tarefas*: como as tarefas mais complexas poderão ser descritas e decompostas em tarefas mais específicas, e como essas tarefas serão alocadas e em que ordem deverão ser executadas.
- *Protocolo de Comunicação*: quais as primitivas que um protocolo de comunicação deveria utilizar num trabalho cooperativo.

- *Coerência, controle e coordenação*: como garantir um comportamento global coerente entre um conjunto de agentes, tendo cada um suas próprias habilidades e objetivos, e como deveria ser projetado o controle de tal sistema.
- *Conflitos e incertezas*: como podem ser resolvidos os conflitos que surgem, já que nenhum agente possui toda a informação do seu ambiente, e como dados incompletos podem ser distribuídos de forma a garantir resultados coerentes.
- *Linguagem de programação e ambiente*: sob o ponto de vista computacional, quais as linguagens de programação que poderiam ser utilizadas em tais sistemas.

Moulin e Chaib-Draa (1996) enfatizam as características que constituem vantagens significativas dos Sistemas Multi-agente sobre o solucionador de problemas monolítico, dentre elas:

- Geralmente, apresentam maior rapidez na resolução de problemas através do aproveitamento do paralelismo;
- Diminuição da comunicação, por transmitir somente soluções parciais em alto nível para outros agentes, ao invés de dados brutos para algum lugar central;
- Mais flexibilidade, por ter agentes de diferentes habilidades que são dinamicamente agrupados para resolver problemas;
- Aumento da tolerância a falhas, pela possibilidade de agentes assumirem responsabilidades daqueles que falham.

A aplicação da tecnologia de Sistemas Multi-agente é justificada quando o problema possuir as seguintes características:

- O domínio envolve distribuição intrínseca de dados, capacidades de resolução de problemas e responsabilidades;
- Necessidade de manter a autonomia de sub-partes, sem a perda da estrutura organizacional;
- Complexidade nas interações, incluindo negociação, compartilhamento de informação e coordenação;
- Impossibilidade de descrição da solução do problema *a priori*, devido à possibilidade de perturbações em tempo real no ambiente (por exemplo: falhas no equipamento) e processos de negócio de natureza dinâmica.

Existe uma série de domínios de aplicação onde a solução de problemas baseada em sistemas multi-agente é apropriada, como: manufatura, controle automatizado, telecomunicações, sistemas de transporte, gerenciamento de informação, *e-commerce* e jogos.
O domínio da manufatura pode ser modelado como uma hierarquia de áreas de trabalho. Pode haver áreas de trabalho em moagem, tornaria, pintura, montagem e assim por diante. Estas áreas de trabalho são agrupadas em subsistemas de manufatura, sendo cada subsistema uma função dentro do processo maior de manufatura. Algumas referências sobre manufatura baseada em agentes incluem trabalhos em seqüênciamento de produção (Chung & Wu, 1997), operações de manufatura (Oliveira *et al.,* 1997) e projeto colaborativo de produtos (Cutosky *et al.* 1993; Darr e Birmingham, 1996).

Controladores de processo são sistemas autônomos, reativos e freqüentemente distribuídos. Existem pesquisas em sistemas de controle de transporte (Corera *et al*., 1996), controle espacial (Schwuttke & Quan, 1993), aceleradores de feixe de partículas (Klein *et al.,* 2000), controle de tráfego aéreo (Ljunberg & Lucas, 1992) e outros.
Sistemas de telecomunicações são grandes redes distribuídas de componentes interativos que requerem monitoração e gerenciamento em tempo-real. Sistemas baseados em agentes têm sido usados para controle e gerenciamento de redes (Schoonderwoerd *et al.,* 1997), transmissão e comutação (Nishibe *et al.* 1993).

Sistemas de tráfego são quase que, por definição, distribuídos, localizados e autônomos. Entre as aplicações incluem-se a coordenação de comutadores e de carros compartilhados (Burmeister *et al.,* 1997) e o planejamento de transporte cooperativo (Fischer *et al.,* 1996).

Sistemas de agentes possibilitam o gerenciamento inteligente da informação, especialmente na Internet. Entre as aplicações incluem-se WEBMATE (Chen & Sycara, 1998), a filtragem de mensagens eletrônicas (Maes, 1994), assistente de navegação na Web (Lieberman, 1995) e um agente especialista locador (Kautz *et al.,* 1997).
Vários sistemas de agentes foram desenvolvidos para o comércio eletrônico, como para gerenciamento de portifólio (Sycara *et al.,* 1996), assistência de compras (Doorenbos *et al*., 1997; Krulwich, 1996) e catálogos interativos (Schrooten & van de Velde, 1997; Takahashi *et al.,* 1997).

Há muitos outros domínios onde abordagens baseadas em agentes são apropriadas e muito embora a tecnologia de agentes ofereça muitas vantagens potenciais para a solução

inteligente de problemas, existem muitos desafios a superar (Jennings *et al.,* 1998). Existem ainda muitos outros tópicos a serem tratados quando se estuda sistemas multi-agente, como protocolos de comunicação inter-agentes, ambientes de desenvolvimento, integração e interoperação de sistemas multi-agente, entre outros. Entretanto, nós nos limitamos neste trabalho a apresentar os conceitos básicos que permitam o entendimento de um sistema multi-agente.

4.4.1 Sistemas Multi-agente Reativos

A abordagem reativa foi introduzida por Brooks (1986), no domínio da robótica. Um exemplo de sistema multi-agente reativo é uma colônia de formigas, onde cada formiga é uma entidade simples. Mas uma colônia de formigas pode realizar trabalhos tais como: procura de alimentos, transporte do alimento até o formigueiro, defesa da colônia, etc. A execução destes trabalhos é complexa, muito embora a estrutura de cada formiga seja simples.

A seguir são destacadas algumas características de Sistemas Multi-agente reativos:

- Não há representação explícita de conhecimento. O conhecimento dos agentes é implícito e se manifesta através do seu comportamento;
- Não há representação do ambiente. O seu comportamento se baseia no que é percebido a cada instante do ambiente, mas sem uma representação explícita deste;
- Não há memória das ações. Os agentes reativos não mantém um histórico de suas ações, de forma que o resultado de uma ação passada não exerce nenhuma influência sobre suas ações futuras;
- Organização etológica. A forma de organização dos agentes reativos é similar a de animais como insetos e microorganismos, em oposição à organização social dos sistemas cognitivos;
- Grande número de membros. Os sistemas multi-agente reativos têm em geral, um grande número de agentes, de ordem de dezenas, centenas ou mesmo milhões de agentes.

Existem alguns ambientes de desenvolvimento para Sistemas Multi-agente reativos, como o sistema SEIEME - *Simulateur Evènementiel Multi-Entités* (Magnin, 1996) e o sistema *Swarm* (Minar, 1996).

4.4.2 Sistemas Multi-agente Cognitivos

Os SMA cognitivos são baseados em modelos organizacionais humanos. Segundo (Ferber, 1991) as principais características destes sistemas são:

- Mantém uma representação explícita de seu ambiente e de outros agentes da sociedade;
- Podem manter um histórico das interações e ações passadas;
- A comunicação entre agentes é feita através do envio e recebimento de mensagens;
- O mecanismo de controle é deliberativo, ou seja, os agentes raciocinam e decidem sobre quais objetivos devem alcançar;
- O modelo de organização é baseado em sistemas sociológicos, como as organizações humanas;
- Uma sociedade contém tipicamente poucos agentes.

4.5 Conclusão

Neste capítulo foram abordados conceitos sobre IAD, que concentra o seu foco no comportamento inteligente que emerge como produto da cooperação de diversas entidades as quais denominamos agentes e sua divisão em duas áreas maiores: Resolução Distribuída de Problemas (RDP) e Sistemas Multi-agente (SMA). RDP estuda como um conjunto de módulos coopera para dividir e compartilhar o conhecimento de um problema e no desenvolvimento da solução, enquanto os SMA estudam a coordenação da conduta inteligente entre um grupo de agentes inteligentes autônomos.

Baseado nos conceitos apresentados neste capítulo será proposto a seguir, a aplicação de um sistema multi-agente em mineração de dados distribuída que utiliza grandes quantidades de dados particionados em subconjuntos. Em cada subconjunto agentes realizam a aprendizagem localmente e posteriormente os resultados são combinados em um modelo global através de um processo de troca de mensagens e colaboração.

Capítulo 5

SMAMDD - Um Sistema Multi-agente para Mineração de Dados Distribuída

As técnicas de Mineração de Dados Distribuída têm sido usadas para descobrir conhecimento em grandes bases de dados. Com o crescimento das bases de dados e a necessidade de bases distribuídas, novas técnicas são pesquisadas visando acelerar o processo de extração do conhecimento em grandes bases de dados. Trabalhos recentes combinam as abordagens de Mineração de Dados com Sistemas Multi-agente (Schroeder & Bazzan, 2002; Viktor & Arndt, 2000; Klusch, Lodi & Moro, 2003). Este trabalho apresenta uma técnica de Mineração de Dados Distribuída baseada em um ambiente multi-agente, chamado *SMAMDD* (Sistema Multi-agente para Mineração de Dados Distribuídos), que utiliza dados particionados em subconjuntos objetivando a partir de modelos individuais gerar um modelo global consistente. Neste trabalho, um grupo de agentes é responsável por aplicar o algoritmo de aprendizagem de máquina *RIPPER* (Cohen, 1995) em subconjuntos de dados a serem minerados. A cooperação entre os agentes acontece pela troca de informações sobre as regras geradas em cada agente na busca das melhores regras para cada subconjunto de dados. A escolha das regras para cada agente baseia-se no fatores de qualidade da regra, cobertura e interseção entre regras. O modelo proposto tem como objetivo a seleção das melhores regras que irão compor o modelo global buscando-se aumento na qualidade e cobertura e diminuição na interseção entre as regras.

Basicamente, o processo envolve as seguintes fases: (1) preparação dos dados, (2) geração de modelos individuais, onde cada agente aplica o algoritmo *RIPPER* sobre um subconjunto de dados para a obtenção de regras, (3) cooperação entre os agentes através de

trocas de mensagens, e (4) construção de um modelo integrado baseado nos resultados obtidos a partir da cooperação entre os agentes.

A cooperação entre agentes visa acelerar o processo de descoberta de conhecimentos sobre as bases de dados, a partir de soluções locais que gerarão uma solução global. O modelo proposto busca a seleção das melhores regras geradas sem produzir uma redução no desempenho da descoberta de conhecimento.

5.1 Infra-estrutura do SMAMDD

No sistema de aprendizagem distribuído proposto, os agentes utilizam o mesmo algoritmo de aprendizagem de máquina em subconjuntos dos dados a serem minerados. Posteriormente, os modelos individuais são integrados para a geração de um modelo global. Nesta abordagem cada agente é responsável por um subconjunto de dados de tamanho reduzido, objetivando melhorar a performance do algoritmo.

Na fase inicial, de preparação dos dados, uma base de dados é dividida em *N* bases menores, uma para cada agente analisador, contendo 70% dos exemplos encontrados aleatoriamente da base de dados original, sendo os 30% restantes dos exemplos reservados para a base de teste (BTe). A partir de cada uma das sub-bases de 70% são geradas as bases de: 90% base de treinamento (BTr) e 10% base de validação (Bv) de cada agente, conforme Figura 5.1 abaixo.

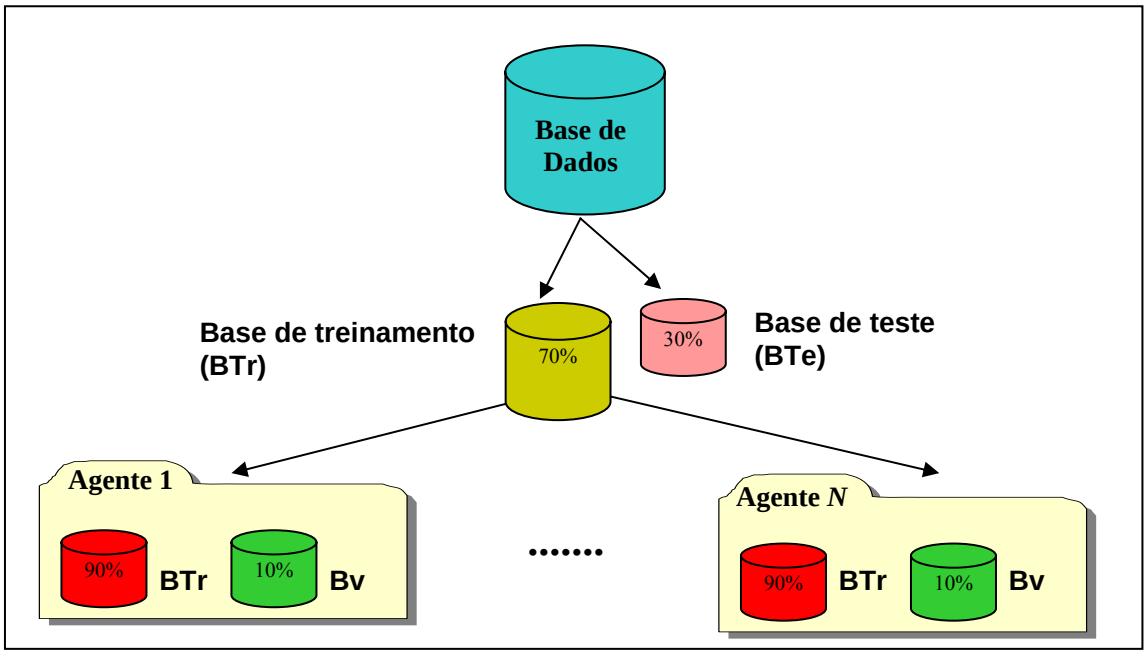

Figura 5.1: Fase inicial de preparação dos dados

Em uma segunda fase, utilizando-se da ferramenta *WEKA (Waikato Environment for Knowledge Analysis)* e do algoritmo de aprendizagem de máquina *RIPPER*, são geradas as regras do tipo SE-ENTÃO a partir das bases de treinamento de cada agente. Para cada regra gerada atribuiu-se um fator de *suporte*

$$Suporte(X \cup Y) = \frac{número\_total\_de\_registros\_com(X \cup Y)}{número\_total\_de\_transações\_no\_banco\_de\_dados}$$

e um fator de *precisão*

$$precisão(X \Rightarrow Y) = \frac{número\_total\_de\_transações\_com(X \cup Y)}{número\_transações\_com(X)}$$

os quais irão gerar o fator de *qualidade* da regra (qualidade = suporte X precisão), neste trabalho, sendo proposto como uma medida de avaliação entre regras.

Cada agente analisador utiliza também os fatores de *interseção* e *cobertura*, os quais representam o nível de interseção entre as regras e a quantidade de exemplos cobertos pelas regras do agente, respectivamente. Ao final do processo as regras geradas estão no formato: *rule (Premissas, Classe, [Suporte, Erro, Precisao, Cobertura, Intersecao, Qualidade, regra_propria]).* O parâmetro *regra_propria* representa se a regra foi gerada no agente ou incorporada de outra agente, passando a ser denotada como r*egra_externa.*

Após a geração das regras e determinação dos fatores mencionados, inicia-se o processo de cooperação entre os agentes para a descoberta das melhores regras. A cooperação ocorre através da troca de mensagens e possui uma coordenação hierárquica com a existência um agente gestor. Esta segunda fase está representada na Figura 5.2.

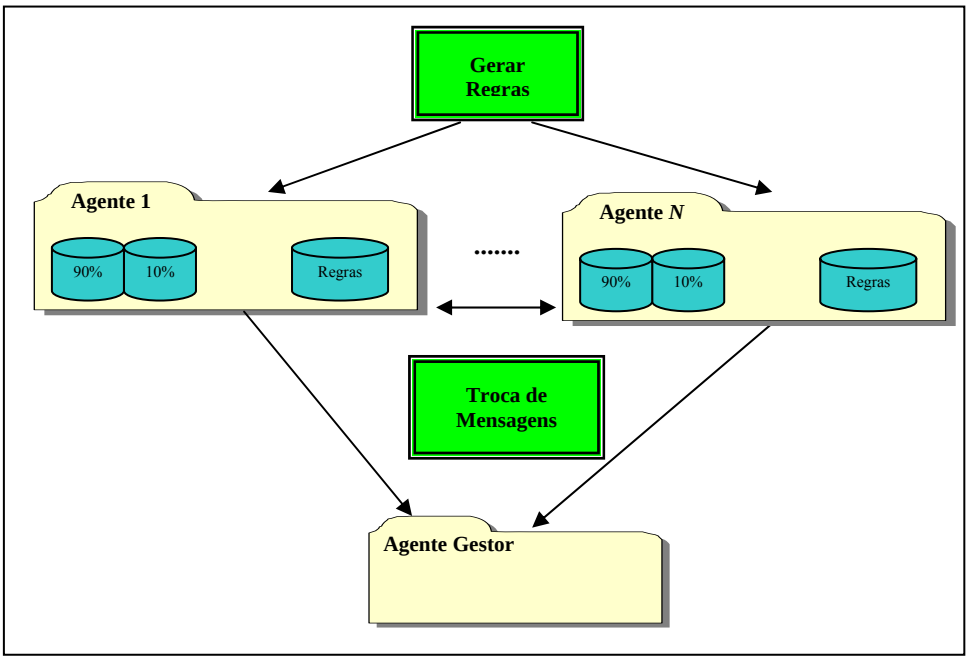

Figura 5.2: Segunda fase - geração das regras e troca de mensagens

Os agentes participantes são do tipo cognitivo possuindo conhecimentos prévios uns dos outros. Cada agente analisador possui como conhecimento individual o endereço de todos os outros agentes. O agente gestor, além de possuir o conhecimento dos endereços dos agentes analisadores, possui uma interface pela qual o operador humano é informado dos resultados obtidos.

A comunicação entre os agentes parte da necessidade de testar as regras geradas em um determinado agente sobre a base de validação dos outros agentes, verificando-se conjuntamente o fator de qualidade da regra sobre esta base de dados e os fatores de interseção e cobertura obtidos com a inserção desta regra em cada agente. Estes fatores irão gerar o *valor_estado* do agente, o qual é obtido através da fórmula:

$$valor\_estado = \frac{((fator\_qualidade \times 1{,}0) + (fator\_cobertura \times 0{,}5) + (fator\_\mathbf{int}erseção \times 0{,}1))}{3}$$

Conforme os pesos atribuídos a cada fator busca-se regras com fator de qualidade e cobertura altos e fator de interseção baixos. Os pesos utilizados neste trabalho foram determinados após algumas interações do *SMAMDD* visando melhores taxas de acerto.

Trabalhos futuros poderão abordar um estudo mais aprofundado sobre os valores a serem definidos para cada fator.

O objetivo principal será encontrar o agente que apresenta o maior *valor_estado* para uma determinada regra. O agente que apresentar o maior valor para uma determinada regra, deverá incorporá-la ao seu conjunto de regras e informar ao agente remetente e aos demais agentes que a possuem para excluí-la.

Após todas as regras de todos os agentes serem analisadas elas deverão ser testadas sobre a base de teste. As regras do agente que obtiverem maior taxa de acerto sobre a base de teste, irão compor o modelo global. Além das regras finais obtidas são apresentados os valores da taxa de acerto das regras, a quantidade final de regras geradas, a complexidade média das regras e desvio padrão. Os resultados obtidos são comparados aos resultados da Mineração de Dados realizada no ambiente *WEKA* sobre as mesmas bases de testes utilizando as técnicas *bagging* e *boosting* e o algoritmo *RIPPER* (Cohen, 1995) .

Bagging foi introduzido por Breiman (1996) com o objetivo de combinar vários classificadores, produzindo uma melhor taxa de acerto sobre os dados de teste. *Bagging* seleciona amostras de tamanhos iguais, a partir dos dados de treinamento e gera classificadores para cada amostra. Cada amostra de treinamento é formada por instâncias selecionadas aleatoriamente, mas com recobrimento, ou seja, uma instância pode aparecer repetidas vezes ou nenhuma em qualquer subconjunto de treinamento. A classificação de uma instância é realizada pelo critério de votação simples.

De acordo com Schapire (2001), *boosting* explora o problema da complementaridade nos classificadores, garantindo modelos mais complementares possíveis. Existem várias versões de *boosting*, mas a idéia geral é que todas as instâncias de treinamento são inicializadas com um peso ponderado, estes pesos são utilizados para estimar o erro dos classificadores. O erro de um classificador é medido pela soma dos pesos das instâncias mal classificadas dividido pela soma de todos os pesos das instâncias. O método força a atenção do algoritmo nas instâncias mal classificadas porque os pesos destas instâncias são aumentados e, das instâncias classificadas corretamente são diminuídos. O erro global é processado e o processo é repetido diversas vezes até que o erro global produzido pelo classificador seja pequeno.

5.1.1 Arquitetura

No contexto deste trabalho, um sistema multi-agente configura-se pela reunião de um conjunto de agentes autônomos que cooperam entre si para alcançar um objetivo comum de encontrar o melhor conjunto de regras sobre uma determinada base de dados. A cooperação ocorre através da troca de mensagens, sendo a comunicação entre os agentes fundamental para este processo.

No sistema multi-agente implementado existem N agentes analisadores e um agente gestor, os quais são compostos por duas camadas: base de conhecimento individual dos agentes que contêm suas especialidades e comunicação, que permite a troca de mensagens entre os agentes. O agente gestor, além das duas camadas funcionais possui a camada de coordenação, que controla a atividade de cooperação entre os agentes analisadores.

A arquitetura interna dos agentes acima definidos é apresentada nas Figuras 5.3 e 5.4:

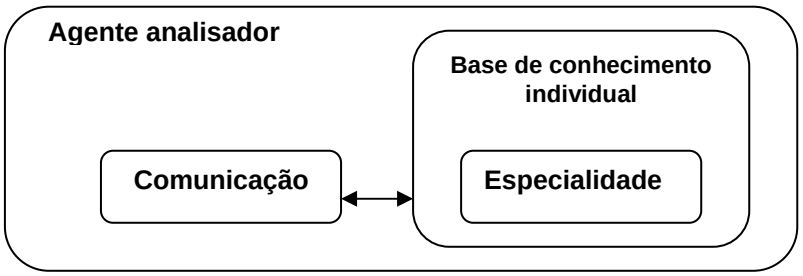

Figura 5.3: Arquitetura do agente analisador

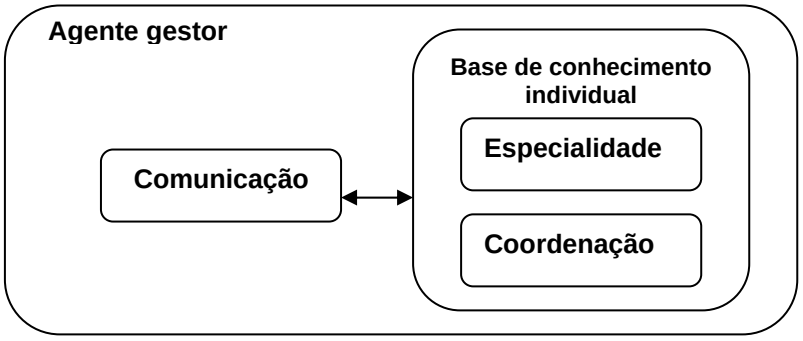

Figura 5.4: Arquitetura do agente gestor

O conhecimento individual dos agentes, tanto analisadores como gestor, compreende os endereços dos agentes na forma: *endereco_agente(nome, host, porta). Host* é o endereço do agente, *Port* é a identificação do número da porta de comunicação do agente e *Nome* é o nome de identificação do agente, que pode ser de *analisador1* até *analisador10* ou *gerente.* O agente gestor, além do endereço dos outros agentes, possui o endereço do operador humano

na forma: *endereco_operador(nome, host, porta).* Neste trabalho, todos os agentes encontram-se na máquina local, mas poderiam estar distribuídos em máquinas remotas.

A cooperação entre os agentes é suportada pelo módulo de comunicação. O módulo de comunicação disponibiliza os canais físicos e o protocolo necessário. A comunicação utiliza um mecanismo seletivo de comunicação assíncrona (troca de mensagens diretas) suportada por *sockets* do sistema operacional Linux.

Ao iniciar o sistema multi-agente, através do predicado *iniciar_sma*, para cada agente lançado é aberto um terminal que apresentará as mensagens trocadas durante a comunicação entre os agentes. Neste momento, também é especificado as capacidades dos agentes na forma: *capacidade(Nome_Agente, identificacao, implementacao, prioridade)*, onde identificação define a capacidade do agente e implementação define o arquivo da capacidade especificada. Os agentes analisadores possuem apenas a capacidade de comunicação e análise, já o agente gestor possui as capacidades de comunicação e gerência. A prioridade definida, estabelece a ordem de execução das capacidades num *looping* entre comunicação e análise/gerência, sendo a prioridade maior ("1") para a comunicação e a prioridade menor ("2") para a análise ou gerência.

Os agentes possuem um modelo o qual dá início ao ciclo vida do agente e ao processo de gerenciamento de suas capacidades obedecendo a prioridade das mesmas, conforme apresentado a seguir.

```
%inicia vida do agente
INICIO
      LEIA  Nome do agente
      EXECUTAR apagar arquivos de mensagens do agente
      EXECUTAR carregar capacidades do agente
      EXECUTAR looping de execução do agente
FIM.
%carrega capacidades do agente
INICIO
      PARA Agente= 1 ATÉ (N+1) FAÇA //Todos os agentes analisadores e o agente gestor
      INICIO
            CARREGAR capacidade do agente //capacidade será: comunicação, análise
ou gerência
      FIM-PARA
FIM.
```

```
%looping de execução do agente
INICIO
      ENQUANTO (houver mensagem) FAÇA
            EXECUTAR gerenciar capacidades do agente
            EXECUTAR limpar variáveis que já foram utilizadas
            EXECUTAR limpar pilha do programa
      FIM ENQUANTO
FIM.
%gerenciamento de capacidades
INICIO
      PARA Agente= 1 ATÉ (N+1) FAÇA //Todos os agentes analisadores e o agente gestor
      INICIO
            ORDENAR capacidades do agente por prioridade,
                  // prioridade "1" para comunicação e "2" para analise ou gerencia
            EXECUTAR capacidades do agente ordenadas,
      FIM-PARA
FIM.
```

Após o lançamento dos agentes, o operador humano, através da interface criada, pode solicitar o início do processo de geração das regras dos agentes analisadores que desencadeará a cooperação entre os agentes para encontrar o melhor conjunto de regras. A cooperação entre os agentes será coordenada pelo agente gestor.

5.1.2 Comunicação

A comunicação entre os agentes é realizada pelo módulo de comunicação, presente em todos os agentes, o qual disponibiliza os canais físicos e o protocolo necessário. Através do predicado *criar_canal_comunicacao(Nome, Host, Port, Socket)* instancia-se o *socket* e cria-se um canal de comunicação entre os agentes. *Host* é o endereço do agente o qual envia ou recebe uma mensagem, *Port* é a identificação do número da porta de comunicação do agente e *Nome* é o nome de identificação do agente, que pode ser de *analisador1* até *analisador10* ou *gerente*.

```
%cria ou instancia socket e gerencia a comunicação
INICIO
      LEIA nome do agente
      LEIA identificação do número da porta de comunicação do agente
      LEIA endereço do agente
      EXECUTAR criar canal de comunicação
                        //criar_canal_comunicacao(Nome, Host, Port, Socket)
```

```
      EXECUTAR gerenciamento da comunicação
FIM.
%criar canal de comunicação que cria ou instancia socket
INICIO
      CRIAR  comunicação       // create_communication(Host, Port, Socket)
      DEFINIR  canal de comunicação    // canal(Nome, Host, Port, Socket)
FIM.
%gerenciamento da comunicação do socket em questão
INICIO
      LER socket da mensagem do agente remetente
      SE remetente <> noagent ENTÃO
            LER mensagem
            ARMAZENAR mensagem
      SENÃO SE EXISTE mensagem a ser enviada ENTÃO
                  ENQUANTO houver mensagem FAÇA
                        ENVIA mensagem
                        APAGAR mensagem
                  FIM-ENQUANTO
            FIM-SE
      FIM-SE
FIM.
```

Toda mensagem criada será empacotada no seguinte formato: *[mensagem_para_ enviar, Mensagem, Host_Destino, Port_Destino]*, onde *Mensagem* possui o conteúdo da mensagem enviada, *Host_Destino* é o endereço do agente destinatário e *Port_Destino* é a identificação do número da porta de comunicação do agente destinatário.

Existem apenas dois tipos de mensagens a serem trocadas entre os agentes: *armazenar mensagem* e *enviar mensagem*. Quando um agente recebe uma mensagem do tipo *armazenar mensagem* ele deve armazená-la no seguinte formato: *armazenar_mensagem (Remetente, Host_Remetente, Port_Remetente, Mensagem_ Recebida)*. Uma mensagem do tipo *enviar mensagem* determina o formato da troca de mensagens entre os agentes da seguinte forma: *enviar_mensagem (mensagem_para_enviar(Mensagem, Host_Destinatario, Port_ Destinatario))*.

```
%armazena mensagem recebida
INICIO
      LER mensagem recebida
      ARMAZENAR mensagem recebida
FIM.
```

```
%enviar mensagem
INICIO
      LER canal de comunicação do agente
      CRIAR mensagem a ser enviada
      ESCREVER mensagem no canal de comunicação definido
      ESCREVER no terminal do remetente ('***** Mensagem Enviada *****')
('Para:' nome do destinatário)
FIM.
```

Todas as mensagens recebidas são processadas pelo predicado *processar_mensagens* e direcionadas para o módulo de tomada de ação.

```
%processa as mensagens recebidas e direciona para a tomada de ações
INICIO
      LER mensagem recebida
      GRAVAR mensagem recebida
      ESCREVER no terminal do destinatário ('***** Mensagem Recebida
*****') ('De:' nome do remetente) ('Mensagem:' mensagem recebida)
      INTERPRETAR mensagem
      EXECUTAR próxima ação baseado na mensagem recebida
FIM.
```

De acordo com as mensagens recebidas determinadas ações são realizadas. No agente gestor as ações possíveis são:

- verificar quais agentes estão ativos: entre os agentes analisadores quais já foram lançados.

```
%verifica quais agentes estão ativos
INICIO
      SE Agentes ativos = 0 ENTÃO
            ENVIAR mensagem ao operador "Estou sozinho"
      SENÃO
            PARA Agente=1 ATÉ N FAÇA
                  ENVIAR mensagem "você está ok?”
            FIM-PARA
            ENVIAR mensagem a si mesmo "checando agentes ativos"
      FIM-SE
FIM.
%retorno dos agentes ativos
INICIO
      LER mensagem
      Respondentes <- Respondentes + 1
```

```
      SE Respondentes = Agentes ativos ENTÃO
            IMPRIMIR Lista dos agentes que responderam
      SENÃO ENVIAR mensagem a si mesmo "checando agentes ativos"
      FIM-SE
FIM.
```

- desativar um ou mais agentes: entre os agentes analisadores ativos, desativar um especificamente ou todos.

```
%desativa todos os agentes
INICIO
      LER mensagem
      SE mensagem = "desativar todos agentes" ENTÃO
            SE Agentes ativos =1 ENTÃO    //somente agente gestor ativo
                  ENVIAR mensagem ao agente operador "Estou sozinho"
            SENÃO
                  PARA Agente =1 ATÉ N FAÇA
                        ENVIAR mensagem "se desativar"
                  FIM-PARA
                  Agentes ativos <- 1
                  posso ser desabilitado<- sim
                  ENVIAR mensagem a si mesmo "se desativar"
                  ENVIAR mensagem ao agente operador "Esta feito”

            FIM-SE
      FIM-SE
FIM.
%desativar um agente analisador
INICIO
      LER mensagem desativar(Agente)
      ENVIAR mensagem para Agente "se desativar"
FIM.
```

- auto-desativar: permite desativar o agente gestor quando não existem agentes analisadores ativos.

```
%desativar o agente gestor caso não exista mais agentes analisadores ativos
INICIO
      SE posso ser desabilitado = sim ENTÃO
            ENVIAR mensagem ao operador "Adeus"
            ENVIAR mensagem ao operador "Esta feito"
            Agentes ativos <- 0
```

```
      FIM-SE
FIM
```

- iniciar avaliação: ativa a geração das regras em cada agente analisador para iniciar o processo de gerenciamento das avaliações das regras dos agentes.

```
% inicializa as regras de cada agente
INICIO
      PARA Agente=1 ATÉ N FAÇA
            EXECUTAR inicializar regras do agente
      FIM-PARA
FIM.
```

- pronto para avaliação: aguarda a mensagem de término da geração das regras de todos os agentes ativos *(respondentes)* para iniciar o processo de gerenciar a avaliação das regras.

```
%pronto para avaliação
INICIO
      LER mensagem do agente respondente
      Respondentes = Respondentes +1
      SE Respondentes = número de agentes ativos ENTÃO
            ENVIAR mensagem para gerente "gerenciar avaliação"
      SENÃO espera o término da inicialização das regras dos agentes
      FIM-SE
FIM.
```

- gerenciar avaliação: inicia-se a troca de mensagens entre os agentes analisadores para encontrar o melhor agente para uma determinada regra.

```
%gerenciar avaliação
INICIO
      ENQUANTO Respondentes <> 0 FAÇA
            LER agente respondente
            ENVIAR mensagem ao agente respondente para "avaliar regras"
                  //ativa avaliar regra do agente analisador para testar suas regras
            PARA Agente=1 ATÉ (N-1) FAÇA   // demais agentes analisadores
                  ENVIAR mensagem "testar regras"
            FIM-PARA
            ENVIAR mensagem para si mesmo "esperando término avaliação"
            Respondentes <- Respondentes - 1
      FIM-ENQUANTO
      ENVIAR mensagem para si mesmo "unir regras"
FIM.
```

- avaliação terminada: retorna ao processo gerenciar avaliação para verificar se existe outra regra do agente a ser testada ou se outro agente pode iniciar o processo de avaliação de suas regras.

```
%avaliação terminada
INICIO
	LER mensagem
	SE mensagem = "avaliação terminada" ENTÃO
		ENVIAR mensagem a si mesmo "gerenciar avaliação"
				//retorna ao gerenciar avaliação para
				avaliar as regras de outro agente analisador
	SENÃO ENVIAR mensagem para si mesmo "esperando término avaliação"
	FIM-SE
FIM.
```

- ação para uma solicitação não conhecida: envia uma mensagem ao operador caso a solicitação não esteja prevista.

```
%pergunta não esperada
INICIO
	LER mensagem
	SE Agente = Operador E mensagem desconhecida ENTÃO
		ENVIAR mensagem ao operador "Nao sei se posso te ajudar"
	FIM-SE
FIM.
```

- preparar para finalizar: solicita a todos os agentes analisadores para calcular a taxa de acerto de suas regras sobre a base de validação.

```
INICIO
	PARA Agente=1 ATÉ N FAÇA
		ENVIAR mensagem "calcular taxa de acerto"
	FIM-PARA
FIM.
```

- respostas das taxas de acerto: recebe a taxa de acerto de todos os agentes sobre base de validação e solicita ao agente com maior taxa de acerto para enviar suas regras.

```
INICIO
	maior taxa acerto <-0
	LER taxa de acerto
	ENQUANTO Respondentes < Agentes Ativos FAÇA
		LER taxa acerto
		SE taxa acerto > maior taxa acerto ENTÃO
			ENVIAR mensagem ao agente "enviar regras"
```

```
            FIM-SE
      FIM-ENQUANTO
FIM.
```

- regras: recebe as regras do agente com maior taxa de acerto para posteriormente imprimir os resultados obtidos.

```
INICIO
      LER Regras
      GRAVAR Regras em arquivo
      CALCULAR Quantidade de Regras
      ENVIAR mensagem a si mesmo "Imprimir resultados"
FIM.
```

- imprimir resultados: imprime no terminal do agente gestor a taxa de acerto sobre a base de testes, a quantidade e complexidade média das regras finais e o desvio padrão.

```
INICIO
      IMPRIMIR taxa de acerto
      IMPRIMIR quantidade de regras
      IMPRIMIR complexidade média das regras
      IMPRIMIR desvio padrão
      IMPRIMIR regras
FIM.
```

Nos agentes analisadores as ações possíveis são:

- checar se o agente está ativo:

```
%resposta para pergunta voce_esta_ok
INICIO
      LER mensagem
      SE mensagem = "você está ok" ENTÃO
            ENVIAR mensagem "estou ok"
      FIM-SE
FIM.
```

- desativar o agente: desativa o agente analisador corrente.

```
% desativar agente
INICIO
      LER mensagem
      SE mensagem = "se desativar" ENTÃO
            Agentes ativos <- Agentes ativos -1
      FIM-SE
FIM.
```

- inicializar regras: ativa o predicado *inicializar_regras* que através da ferramenta *WEKA* e do algoritmo de aprendizagem de máquina *RIPPER*, gera a partir da base de treinamento de cada agente, as regras do tipo SE-ENTÃO. Para cada regra gerada atribuem-se os fatores de suporte, precisão e qualidade da regra e ainda os fatores de interseção e cobertura das regras. Após inicializar as regras o agente analisador envia uma mensagem ao agente gestor indicando estar pronto para avaliação das regras.

```
%inicializar regras para avaliação
INICIO
      EXECUTAR inicializar regras  // gera as regras e calcula os fatores
      ENVIAR mensagem ao gestor "pronto para avaliação"
FIM.
```

- avaliar regras: irá enviar, uma a uma, as regras de um determinado agente analisador aos demais agentes analisadores, os quais irão testar as regras. Após a avaliação de todas as regras, o agente analisador envia ao agente gestor a mensagem de avaliação terminada.

```
%avaliar regras
INICIO
      LER regras inicializadas em memória
      ENQUANTO  lista de regras <> vazio FAÇA
            LER Regra
            SE Regra <> regra_externa ENTÃO
                  PARA Agente=1 ATÉ (N-1) FAÇA   // outros agentes analisadores
                        ENVIAR mensagem "testar regra" [Regra]
      FIM-PARA
      FIM-ENQUANTO
      ENVIAR mensagem ao agente gestor  "avaliação terminada"
FIM.
```

- resposta teste: a partir do maior *valor_estado* obtido sobre os fatores de qualidade, cobertura e interseção, decide-se se uma determinada regra deve ser mantida no agente origem ou deve ser excluída do agente origem *(excluir_regra(Regra))* e incorporada pelo agente analisador com maior *valor_estado* através do predicado *incluir(Regra).* Após definir o melhor agente para uma determinada regra executa-se então *avaliar_regras* para testar as outras regras do agente.

```
%esperando resposta dos testes
INICIO
      ENQUANTO Agentes <> (N-1) FAÇA
```

```
        LER mensagem resposta do teste
                                //formato: resposta_teste([EstadoAgente:Regra])
        ARMAZENAR resposta em Estados agentes
    FIM-ENQUANTO
    ORDENAR Estados agentes do maior para menor
    LER primeiro Estado agente
    CALCULAR fator cobertura
    CALCULAR fator interseção
    CALCULAR valor estado novo
    SE Estado agente > valor estado novo ENTÃO
        ENVIAR mensagem "incluir regra"
        EXECUTAR excluir regra
        PARA Agente =1 ATÉ (N-2) FAÇA
            ENVIAR mensagem "excluir regra"
        FIM-PARA
        ENVIAR mensagem a si mesmo "avaliar regras"
    SENÃO ENVIAR mensagem a si mesmo "avaliar regras"
        PARA Agente =1 ATÉ (N-1) FAÇA
            ENVIAR mensagem "excluir regra"
        FIM-PARA
FIM.
```

- testar regras: para cada nova regra recebida, gera-se os fatores de suporte, precisão e qualidade da regra, juntamente com os fatores de interseção e cobertura das regras do agente. O *valor_estado* (gerado a partir da qualidade, cobertura e interseção) é enviado ao agente origem da regra.

```
%testando uma regra: 1. testar regra na base de teste; 2. calcular precisão, erro, suporte, qualidade; 3. testar regra com todas as outras para calcular interseção e cobertura entre regras.
INICIO
    LER mensagem e Regra_x a ser testada
    CARREGAR Base Teste
    CARREGAR Regras inicializadas
    SE não possuir a Regra ENTÃO
        CALCULAR Suporte e Erro da Regra_x
        Precisão <- (1 - (Erro/Suporte))
        Nova Qualidade <- Precisão * Suporte
        INSERIR Regra_x em Regras inicializadas
        CALCULAR Cobertura das regras
        CALCULAR Interseção das regras
```

```
            CALCULAR Valor Estado
            ENVIAR mensagem ao Destinatário a resposta teste
                                    //resposta_teste([EstadoNovo:Regra])
      SENÃO CALCULAR Cobertura das regras
            CALCULAR Interseção das regras
            CALCULAR Valor Estado
            ENVIAR mensagem ao Destinatário a resposta teste
FIM.
```

- incluir: inclui uma nova regra no agente e recalcula os fatores de suporte, erro, precisão, interseção e qualidade da regra; e recalcula os fatores de cobertura e interseção entre as regras.

```
INICIO
      SE não possui a Regra ENTÃO
            INSERIR Regra
            CALCULAR Suporte e Erro da Regra
            Precisão <- (1 - (Erro/Suporte))
            Nova Qualidade <- Precisão * Suporte
            CALCULAR Cobertura das regras
            CALCULAR Interseção das regras
      FIM-SE
FIM.
```

- excluir: exclui uma regra de um agente e recalcula para os demais a interseção e os fatores de cobertura e interseção entre regras.

```
INICIO
      SE possui Regra ENTÃO
            EXCLUIR Regra
            ENQUANTO Regras <> vazio FAÇA
                  CALCULAR Interseção
            FIM-ENQUANTO
            CALCULAR Cobertura das regras
            CALCULAR Interseção das regras
      FIM-SE
FIM.
```

- calcular taxa de acerto: a partir de todas as regras finais do agente analisador, calcula a taxa de acerto de cada regra do agente sobre a base de validação.

```
INICIO
      ENQUANTO Regras <> vazio FAÇA
            ENQUANTO Exemplo <> vazio FAÇA
```

```
                    SE Exemplo contêm Premissas e Classe ENTÃO
                          TAXA <- TAXA + 1,
                    FIM-SE
                    QTD EXEMPLOS <- QTD EXEMPLOS + 1
              FIM-ENQUANTO
        FIM-ENQUANTO
        TAXA ACERTO <- TAXA / QTD EXEMPLOS
        ENVIAR TAXA ACERTO para agente gestor
FIM.
```

- enviar regra: o agente analisador envia seu conjunto de regras para o agente gestor.

```
INICIO
        LER mensagem
        SE mensagem = "enviar regras" ENTÃO
              ENQUANTO Regras <> vazio FAÇA
                    ENVIAR Regra para agente gestor
              FIM-ENQUANTO
        FIM-SE
FIM.
```

5.2 Cooperação no SMAMDD

A interação entre os agentes é do tipo cooperativa e hierárquica sendo suportada pelas funcionalidades existentes no módulo de comunicação. Esta atividade essencial ao funcionamento de um sistema multi-agente é desencadeada pela necessidade de acelerar o processo de descoberta de conhecimentos sobre as bases de dados, a partir da seleção das melhores regras geradas localmente que irão compor um modelo global.

Inicialmente, o agente operador solicita ao agente gestor iniciar o processo de avaliação das regras dos agentes analisadores (Figura 5.5).

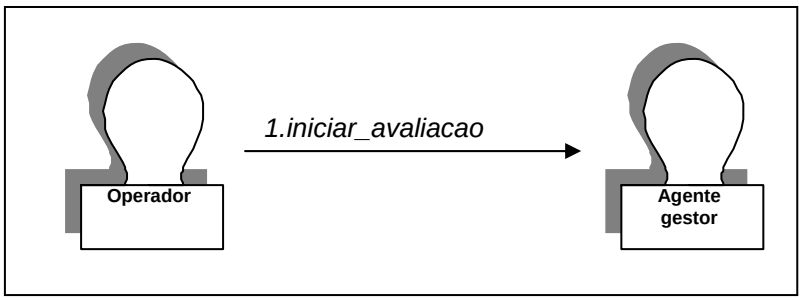

Figura 5.5: Agente operador solicita ao agente gestor *iniciar_avaliacao*

O agente gestor envia mensagem aos agentes analisadores para *inicializar_regras*. Após esta mensagem o agente gestor envia uma mensagem a si mesmo de *esperando_inicialização* (Figura 5.6).

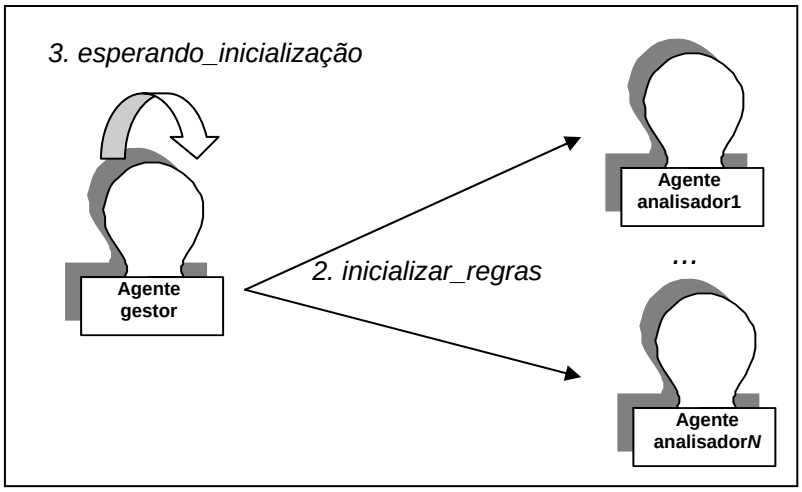

Figura 5.6: Agente gestor envia mensagens de *inicializar_regras* e *esperando_inicialização*

Quando os agentes analisadores terminam de gerar as regras enviam uma mensagem ao agente gestor de *pronto_para_avaliacao*. Após o agente gestor receber a mensagem de todos os agentes analisadores ativos, envia uma mensagem a si mesmo para *gerenciar_avaliacao* (Figura 5.7).

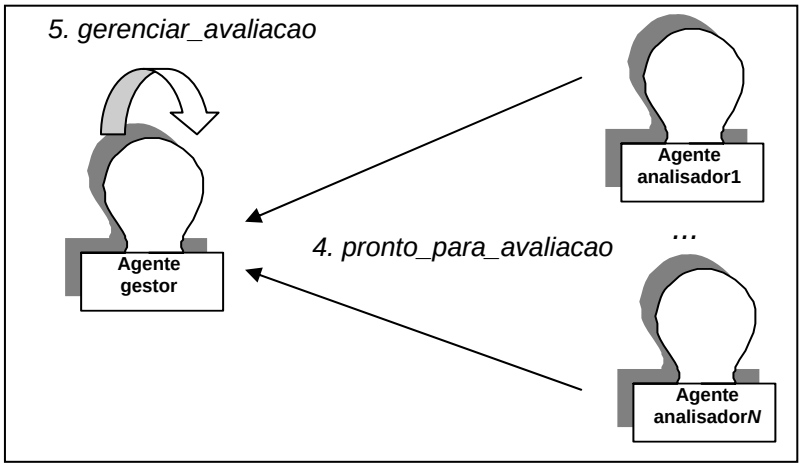

Figura 5.7: Mensagens dos agentes analisadores *pronto_para_avaliacao* e do agente gestor *gerenciar_avaliacao*

Ao iniciar o processo de gerenciamento da avaliação o agente gestor envia uma mensagem ao primeiro agente analisador respondente, para iniciar a avaliação de suas regras

(*avaliar_regras),* aos demais agentes envia a mensagem *testar_regras* e envia a si mesmo a mensagem *esperando_termino_avaliacao* (Figura 5.8).

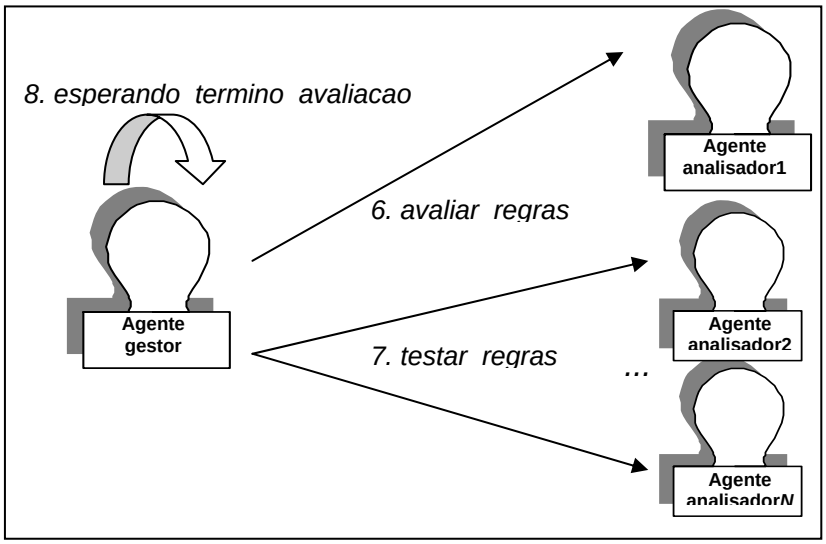

Figura 5.8: Agente gestor envia *avaliar_regras* e *testar_regras*

O agente analisador que recebe a mensagem *avaliar_regras* do agente gestor, envia a mensagem *testar_regra(Regra),* contendo sua primeira regra a ser testada, para todos os demais agentes analisadores. Quando um agente analisador recebe a mensagem *testar_regra(Regra)* do agente origem, calcula para esta nova regra os fatores de: suporte, precisão, qualidade, interseção e cobertura. Estes fatores permitem gerar o *valor_estado* da regra no agente. O *valor_estado* obtido é armazenado em *EstadoNovo* e enviado ao agente origem, com uma mensagem do tipo *resposta_teste([EstadoNovo:Regra])* (Figura 5.9). O procedimento de *testar_regras* continua até que todas as regras do agente sejam testadas. Quando a lista de regras estiver vazia, o agente envia a mensagem *avaliação_terminada* ao agente gestor (Figura 5.10).

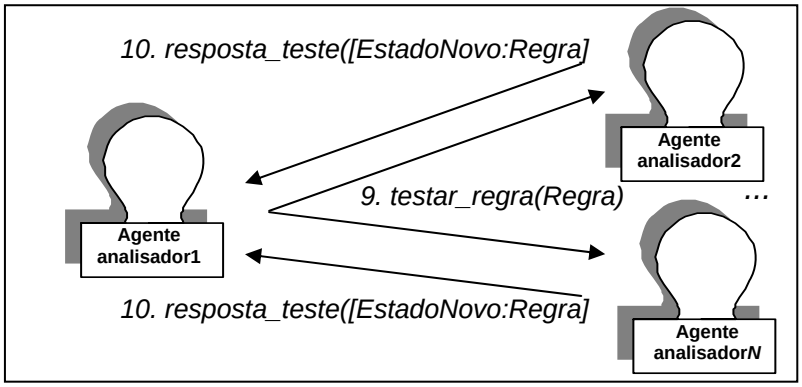

Figura 5.9: Agente *analisador1* envia mensagem aos demais agentes analisadores para testar sua regra

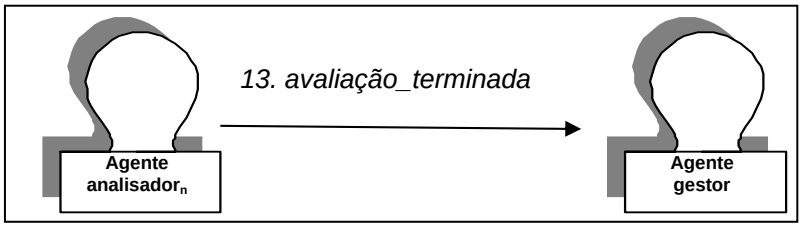

Figura 5.10: Agente *analisador*$_n$ envia mensagem ao agente gestor informando que terminou de avaliar suas regras

Após o agente analisador receber todas as respostas teste dos outros agentes analisadores, avalia qual agente possui o melhor *valor_estado* para a regra. Caso o melhor valor obtido seja o seu próprio, envia uma mensagem *avaliar_regras* para si mesmo para continuar a avaliação das regras restantes (Figura 5.11) e envia a mensagem *excluir(Regra)* aos demais agentes. Caso outro agente possua o melhor *valor_estado*, envia para este agente a mensagem *incluir(Regra)*, envia a mensagem *excluir(Regra)* aos demais agentes e a si mesmo e, posteriormente, envia uma mensagem *avaliar_regras* para si mesmo para continuar a avaliação das regras restantes. A mensagem *avaliar_regras* será enviada até que todas as suas regras sejam avaliadas (Figura 5.12). O procedimento de avaliação das regras irá ocorrer para todas as regras de todos os agentes analisadores.

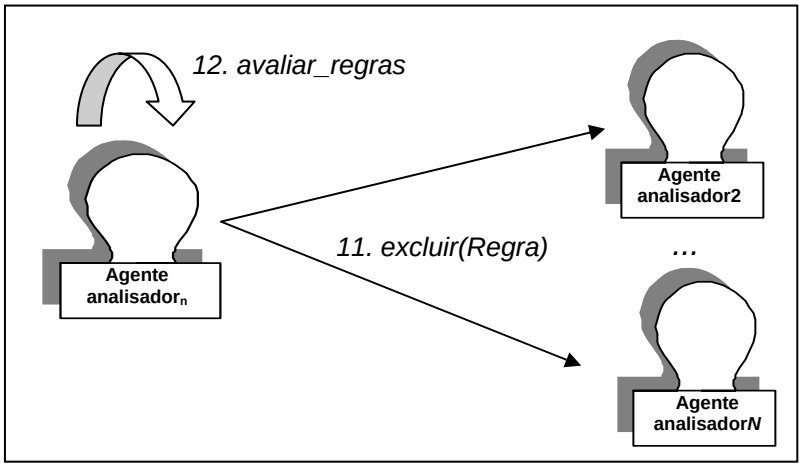

Figura 5.11: Agente *analisador*$_n$ envia mensagem aos demais agentes para *excluir(Regra)* e a si mesmo para avaliar suas outras regras

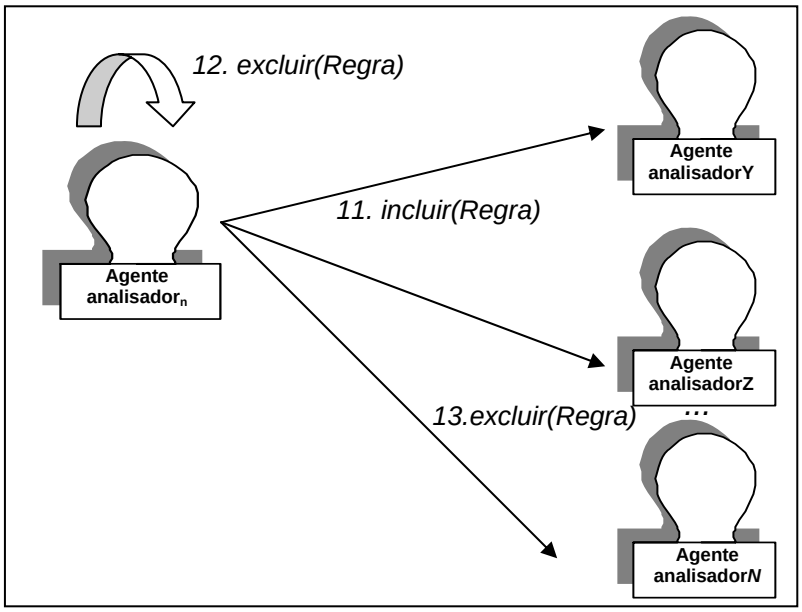

Figura 5.12: Agente *analisador*$_n$ envia mensagem para outro agente analisador *incluir(Regra)* e envia a si mesmo e aos demais agentes para *excluir(Regra)*

Quando todas as regras de todos os agentes analisadores já foram testadas o agente gestor envia uma mensagem a si mesmo para *preparar para finalizar* (Figura 5.13).

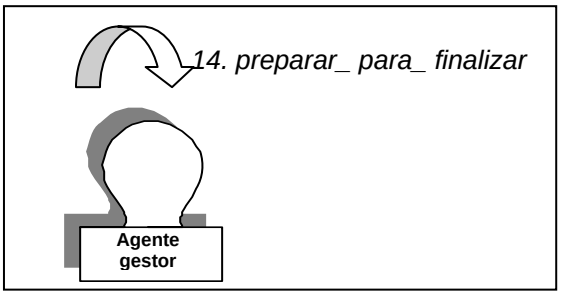

Figura 5.13: Agente gestor envia mensagem a si mesmo para *preparar_para_finalizar*

Ao receber a mensagem *preparar_para_finalizar* o agente gestor envia mensagens aos agentes analisadores solicitando calcular a taxa de acerto de suas regras. (Figura 5.14)

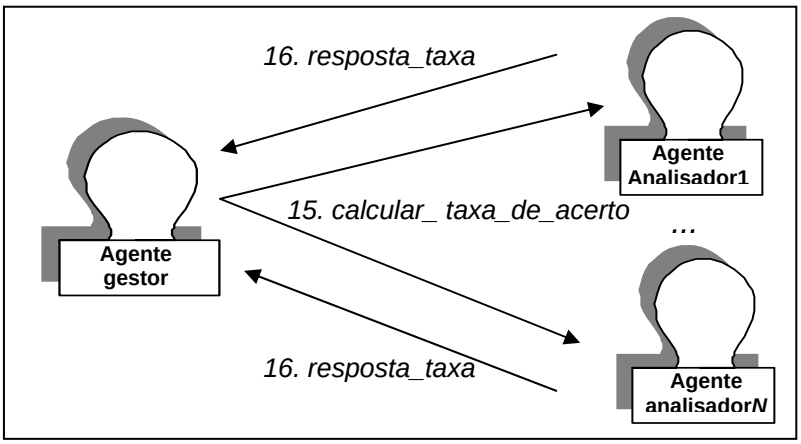

Figura 5.14: Agente gestor envia mensagem *calcular_taxa_de_acerto* aos agentes analisadores e recebe *resposta_taxa*

Após receber todos os valores de taxas de acerto de todos os agentes o agente gestor solicita ao agente com maior taxa de acerto que envie suas regras. (Figura 5.15).

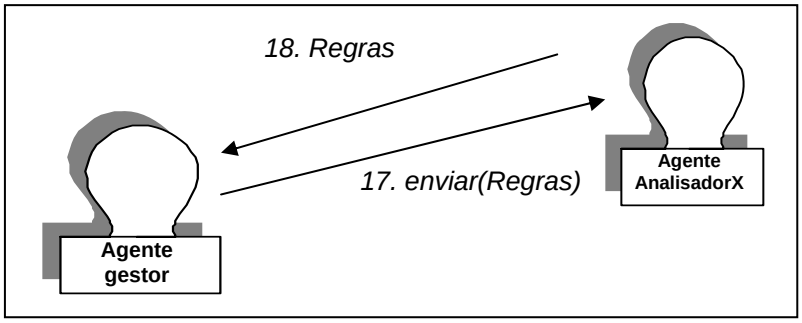

Figura 5.15: Agente gestor solicita enviar(Regras)

Finalizando o processo, após receber todas as regras e calcular a quantidade total de regras, a complexidade média das mesmas e o desvio padrão, o agente gestor envia a si mesmo a mensagem *imprimir_resultados.* (Figura 5.16)

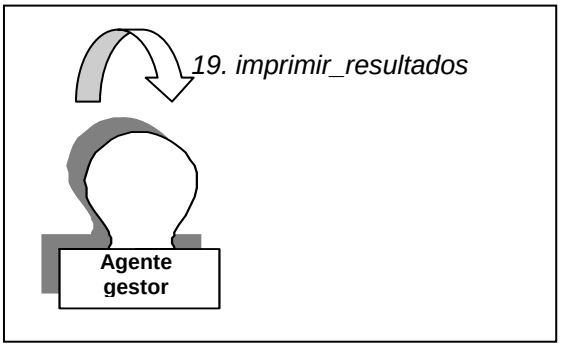

Figura 5.16: Agente gestor solicita a si mesmo *imprimir_resultados*

As mensagens de cooperação entre os agentes estão representadas no diagrama de seqüência apresentado na Figura 5.17 a seguir.

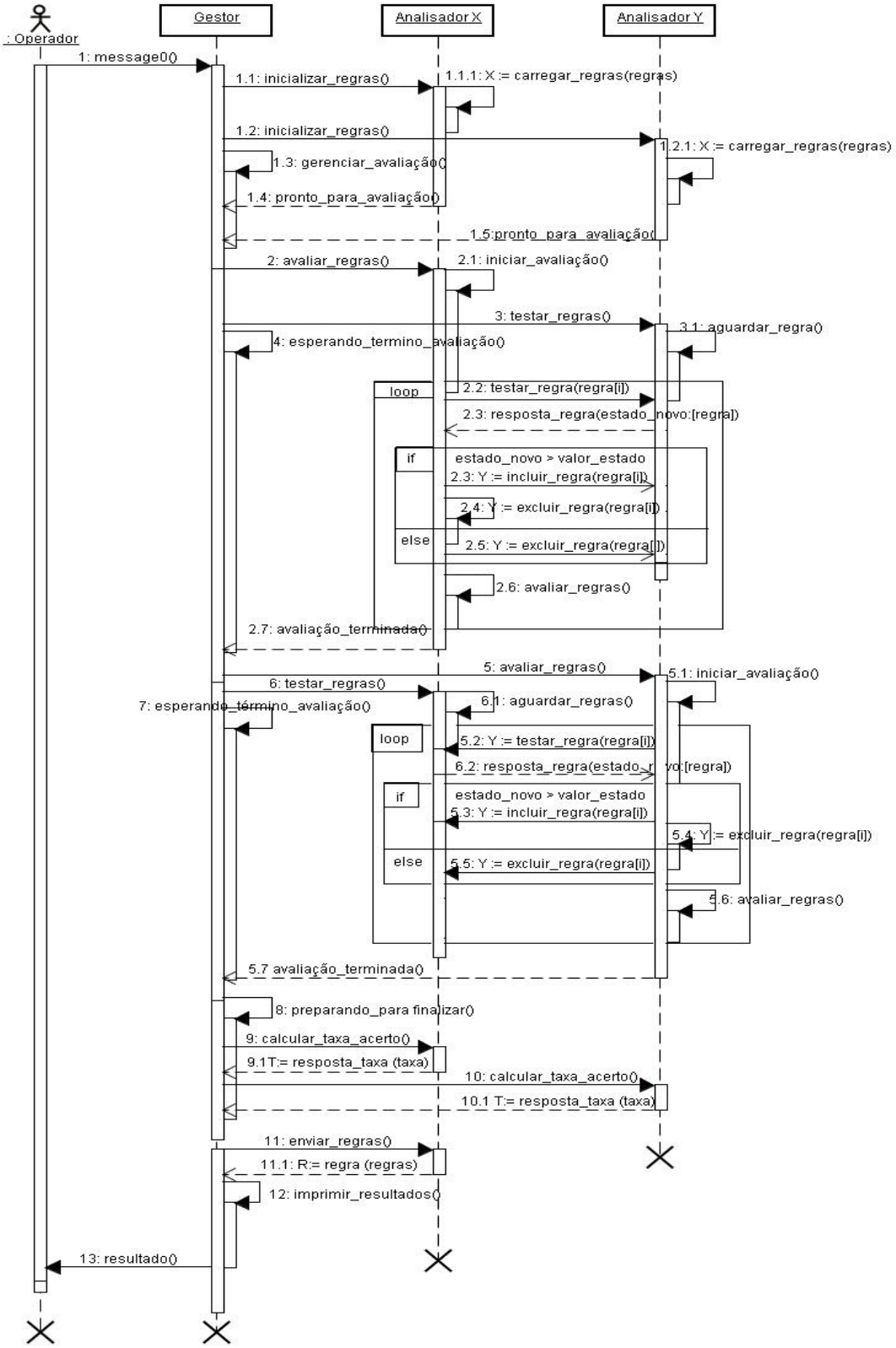

Figura 5.17: Diagrama de seqüência – mensagens de cooperação entre os agentes

5.3 Resultados

Com o objetivo de avaliar o trabalho proposto foram utilizadas as seguintes bases de dados públicas obtidas do Repositório UCI[1]: Breast Cancer, Vote, Zoo, Lymph, Soybean, Balance-scale, Audiology, Splice, Kr-vs-kp e Mushroom. Foram comparados os resultados do sistema *SMAMDD* com os resultados apresentados pelas técnicas *bagging*, *boosting* e o algoritmo *RIPPER* (Cohen, 1995).

Para cada base de dados utilizada foram realizadas dez interações, sendo em cada interação a base dividida em *n* sub-bases através do processo de *cross validation*, onde os exemplos são selecionados randomicamente. Ao final de cada interação das bases no sistema *SMAMDD* são computados os valores da taxa de acerto, do número de regras e da complexidade das regras.

A Tabela 5.1 apresenta algumas informações sobre as bases de dados como: número de atributos, número de classes e número de exemplos.

Número	Base de Dados	N. Atributos	N. Classes	N. Exemplos
1	Breast Cancer	9	2	286
2	Vote	16	2	435
3	Zoo	17	7	101
4	Lymph	18	4	148
5	Soybean	35	19	683
6	Balance-scale	4	3	625
7	Audiology	69	24	226
8	Splice	61	3	3190
9	Kr-vs-kp	36	2	3196
10	Mushroom	22	2	8124

Tabela 5.1: Informações sobre as bases de dados

[1] [1] *www.ics.uci.edu/mlearn/MLRepository.html*

A Tabela 5.2 compara os resultados das taxas de acerto do sistema *SMAMDD* com os técnicas *bagging*, *boosting* e o algoritmo *RIPPER*.

BASES	SMAMDD	RIPPER	BAGGING	BOOSTING
Breast Cancer	**80,23 ±4,4**	71,74 ±3,8	71,39 ±3,3	72,55 ±4,8
Vote	**97,56 ±1,83**	94,88 ±2,50	95,41 ±1,12	93,99 ±1,39
Zôo	91,29 ±6,28	88,71 ±4,87	89,03 ±5,09	**94,84 ±4,36**
Lymph	**84,22 ±11,01**	74,67 ±5,36	80,44 ±7,16	83,78 ±2,78
Soybean	**95,36 ±2,14**	91,12 ±2,17	92,19 ±1,89	92,24 ±1,72
Balance-scale	**85,37 ±4,53**	73,62 ±4,45	79,41 ±3,56	79,15 ±4,15
Audiology	**81,32 ±6,44**	73,97 ±5,53	73,23 ±7,03	75,59 ±7,54
Splice	**98,12 ±1,85**	93,31 ±0,74	95,08 ±0,66	94,67 ±0,23
Kr-vs-kp	97,04 ±2,51	98,99 ±0,32	99,21 ±0,24	**99,37 ±0,41**
Mushroom	**100,00 ±0,00**	99,93 ±0,16	100,00 ±0,00	99,93 ±0,16

Tabela 5.2: Taxas médias de acerto

Os resultados da Tabela 5.2 são muito encorajadores, pois na maioria das bases de dados o sistema *SMAMDD* apresentou taxas de acerto razoáveis. A base que apresentou os melhores resultados foi a base Balance-scale, a qual possui um número reduzido de atributos que levaram a uma melhor da taxa de acerto.

Somente na base de dados Zoo o sistema *SMAMDD* apresentou uma taxa de acerto consideravelmente menor em relação a técnica *boosting.* A base Zoo possui um número reduzido de exemplos em relação a um número proporcionalmente elevado de atributos o que provavelmente ocasionou a perda de performance na taxa de acerto.

A base de dados Lymph, mesmo apresentando uma taxa de acerto no sistema *SMAMDD* maior em relação aos demais, possui um desvio padrão elevado causado novamente pela proporção do número de atributos (elevado) em relação ao número de exemplos.

Observou-se que algumas bases apresentaram desvio padrão elevado, demonstrando a instabilidade do algoritmo diante da proporção do número de atributos em relação ao número de exemplos. A partir destes resultados percebe-se que o sistema *SMAMDD* apresenta

melhores performances nas taxas de acerto quando o número de atributos é pequeno em relação ao número de exemplos.

A Tabela 5.3 compara a quantidade de regras geradas pelo sistema *SMAMDD* em relação as técnicas *bagging*, *boosting* e o algoritmo *RIPPER*.

BASES	**SMAMDD**	**RIPPER**	**BAGGING**	**BOOSTING**
Breast Cancer	3,9 ±1,20	2,7 ±1,03	4,3 ±0,90	**2,3 ±1,40**
Vote	3,2 ±1,03	3,2 ±1,03	**2,7 ±0,48**	4,0 ±2,36
Zoo	**5,7 ±0,48**	**5,7 ±0,48**	6,4 ±0,84	6,2 ±1,40
Lymph	5,2 ±2,30	5,2 ±1,87	5,4 ±1,26	**5,0 ±2,11**
Soybean	49,3 ±18,86	25,1 ±1,29	26 ±2,54	**22,1 ±7,11**
Balance-scale	17,2 ±7,61	11,6 ±3,24	12,4 ±3,75	**10,1 ±3,97**
Audiology	17,7 ±5,19	**13,3 ±1,64**	13,6 ±1,17	16,10 ±3,25
Splice	29,67 ±10,07	**13,67 ±5,51**	17,67 ±4,51	29,0 ±3,46
Kr-vs-kp	38,5 ±7,94	14,5 ±1,38	14,5 ±2,26	**10,0 ±4,82**
Mushroom	12,8 ±2,68	**8,6 ±0,55**	8,8 ±0,84	**8,6 ±0,55**

Tabela 5.3: Número médio de regras

A partir dos valores apresentados na tabela 5.3 percebe-se que o sistema *SMAMDD* tende a produzir um modelo mais complexo em relação ao número de regras se comparado as demais técnicas. É perceptível que o aumento no tamanho das bases de dados, em geral, gerou um aumento considerável no número de regras e um desvio padrão elevado.

A Tabela 5.4 apresenta a complexidade das regras geradas pelo sistema *SMAMDD* e pelas técnicas *bagging*, *boosting* e o algoritmo *RIPPER*.

BASES	SMAMDD	RIPPER	BAGGING	BOOSTING
Breast Cancer	1,84 ±0,14	**1,10 ±0,16**	1,49 ±0,12	1,41 ±0,16
Vote	1,31 ±0,60	1,31 ±0,60	**1,22 ±0,56**	1,36 ±0,65
Zôo	1,08 ±0,17	1,08 ±0,17	**1,00 ±0,16**	1,03 ±0,14
Lymph	1,24 ±0,24	**1,18 ±0,23**	1,44 ±0,23	1,27 ±0,55
Soybean	1,86 ±0,21	**1,59 ±0,06**	1,72 ±0,07	1,89 ±0,32
Balance-scale	1,76 ±0,16	**1,71 ±0,20**	1,85 ±0,19	1,84 ±0,42
Audiology	1,61 ±0,11	**1,50 ±0,12**	1,57 ±0,14	1,66 ±0,21
Splice	3,95 ±0,16	3,52 ±0,44	**3,36 ±0,37**	3,75 ±0,31
Kr-vs-kp	3,39 ±0,24	**2,78 ±0,10**	2,89 ±0,24	3,53 ±1,86
Mushroom	1,51 ±0,11	1,37 ±0,10	**1,36 ±0,10**	1,41 ±0,11

Tabela 5.4: Complexidade média das regras

Conforme os valores apresentados na Tabela 5.4 a complexidade das regras geradas no sistema *SMAMDD* é razoável e comparável as demais técnicas. Mesmo as bases de dados maiores não resultaram em valores elevados, demonstrando que o número de atributos e exemplos não afetou significativamente na complexidade das regras.

A partir das tabelas acima relacionadas, leva-se a crer que o sistema *SMAMDD,* o qual utiliza a integração de modelos, resulta em valores comparáveis as demais técnicas de aprendizagem de máquina. A possibilidade da descoberta de conhecimento em subconjuntos de bases, oriundos de uma base maior, apresentou resultados para as taxas de acerto durante as interações, de boa qualidade comprovando a eficiência da proposta.

Conclusão

A utilização de técnicas de Inteligência Artificial Distribuída (IAD), uma sub-área de Inteligência Artificial que estuda Sistemas Multi-agente (SMA) podem contribuir para aplicações em Mineração de Dados Distribuída (MDD) visando melhorar a aprendizagem através da integração de conhecimento. Os Sistemas de MDD manuseiam diferentes componentes: algoritmos de mineração, subsistemas de comunicação, gerenciamento de recursos, planejamento de tarefas, interfaces com o usuário, entre outros.

O presente trabalho apresentou um Sistema Multi-agente para Mineração de Dados Distribuída em dados particionados em subconjuntos, simulando um ambiente distribuído. Em cada subconjunto agentes realizaram a aprendizagem localmente sendo os resultados avaliados para se chegar a um modelo global.

O sistema proposto objetivou contribuir para a Mineração de Dados em ambientes onde os dados estão distribuídos em diferentes bancos de dados. Modelos individuais foram gerados por agentes utilizando diferentes dados de treinamento obtidos de uma base única. Esta abordagem empregou cooperação entre os agentes através de trocas de mensagens em busca da seleção das melhores regras geradas.

Para a avaliação dos resultados obtidos foram utilizadas as base de dados: *Breast Cancer, Vote, Zoo, Lymph, Soybean, Balance-scale, Audiology, Splice, Kr-vs-kp e Mushroom.* A possibilidade de comparar diferentes técnicas de aprendizagem como *bagging* e *boosting* com o sistema proposto, permitiu avaliar a relação entre o número de regras geradas, complexidade das mesmas e taxas de acerto obtidas.

Como trabalhos futuros novas interações poderão ser realizadas em diferentes domínios de bases de dados, diferentes algoritmos de classificação e estudos sobre diferentes medidas de avaliação de regras sobre o sistema proposto. Poderá ser realizado ainda, o estudo de novas estratégias que permitam reduzir o número de avaliações das regras, como escolher somente alguns agentes para avaliar, diminuindo desta forma o tempo de processamento do sistema.

Referências Bibliográficas

[ALV97] ÁLVARES, L. O.; SICHMAN, J. S. Introdução aos Sistemas Multiagentes. In: JAI, XVII Congresso da SBC, Brasília (DF), 1997.

[BAU99] BAUER, E.; KOHAVI, R. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning, 36,* p. 105–139, 1999.

[BIC98] BICA, F.; SILVEIRA, R.; VICCARI, R. Eletrotutor III: Uma Abordagem Multiagentes*. IX Simpósio Brasileiro de Informática na Educação/SBIE*, 1998.

[BIT98a] BITTENCOURT, G. Inteligência Artificial Distribuída, Anais: *I Workshop de Computação do ITA,* 1998.

[BIT98b] BITTENCOURT, G. *Inteligência Artificial: ferramentas e teorias*, Editora da UFSC, 1998.

[BON88] BOND A. H.; GASSER L. *Reading in Distribuited Artificial Inteligence*, Morgan Kaufmann, 1988.

[BON01] BONNET, P.; GEHRKE, J.; SESHADRI, P. Towards sensor database systems. In *Proceedings of the Second International Conference on Mobile Data Management*, p. 3–14. Hong Kong: Springer, 2001.

[BRE96] BREIMAN, L. . Bagging Predictors. *Machine Learning*, n. 2, vol. 24, p.p. 123-140, 1996.

[BRO86] BROOKS, R. *A robust layered control system for a mobile robot.* IEEE J. Rob. & Aut., RA2 (1):14-22, Mar. 1986.

[BUR97] BURMEISTER, B.; HADDADI, A.; MATYLIS, G. Applications of multi-agent systems in traffic and transportation. *IEE Transactions on Software Engineering*, 144(1):51-60, February 1997.

[CAR00] CARVALHO, D. R., FREITAS, A. A hybrid decision tree/genetic algorithm for coping with the problem of small disjuncts in Data Mining. *Proc 2000 Genetic and Evolutionary Computation Conf. (GECCO-2000)*, 1061-1068. Las Vegas, NV, USA. July 2000.

[CHA93a] CHAN, P.; STOLFO, S. Experiments on multistrategy learning by meta-learning. In *Proceedings of the Second International Conference on Information Knowledge Management* , p. 314–323, 1993.

[CHA93b] CHAN, P.; STOLFO, S. Toward parallel and distributed learning by meta-learning. In *Working Notes AAAI Workshop in Knowledge Discovery in Databases,* p. 227–240. Menlo Park, CA: AAAI, 1993.

[CHA98] CHAN, P.; STOLFO, S. Toward scalable learning with non-uniform class and cost distribution: A case study in credit card fraud detection. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, p. 164–168. Menlo Park, CA: AAAI Press, 1998.

[CHE98] CHEN, L.; SYCARA, K. WebMate: A Personal Agent for Browsing and Searching. *Proceedings of the Second International Conference on Autonomous Agents*, Minneapolis, MN, 1998.

[CHO98] CHO, V.; W¨UTHRICH, B. Toward real time discovery from distributed information sources. In *Second Pacific-Asia Conference, PAKKD '98* , p. 376–377. Melbourne, Australia. Springer-Verlag, 1998.

[CHU97] CHUNG, J. K.; WU, J. S., An ATM-based CDMA cellular network for voice communications. *In Proc. VTC'97,* Phoenix, Az, USA, 1997.

[CLA89] CLARK, P.; NIBLETT, T. The CN2 induction algorithm. *Machine Learning*, 3, 261-283, 1989.

[CLE90] CLEARWATER, S.; PROVOST, F. RL4: A tool for knowledge-based induction. In *Proceedings of the Second International IEEE Conference on Tools for Artificial Intelligence*, 24-30. IEEE CS Press, 1990.

[COH95] COHEN, W. W. Fast effective rule induction. In: *Proc. Of the Twelfth Intl. Conf. on Machine Learning*, pp: 115-123. 1995.

[COR96] CORERA J. *et. al. An architecture for real-time reasoning and system control.* IEEE Expert 7(6); 1996.

[CUT93] CUTOSKY *et. al. PACT: An Experiment in integrating concurrent engineering systems, Computer*, Vol. 26, No. 1, 1993.

[DAR96] DARR, T. P.; BIRMINGHAM, W. P. *An Attribute-Space Representation and Algorithm for Concurrent Engineering*. AIEDAM, 10(1), pp 21- 35, 1996.

[DIE00] DIETTERICH, T. G. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization. *Machine Learning, 40,* p. 139–158, 2000.

[DOM96] DOMINGOS, P. Efficient Specific-to-General Rule Induction. In *Proc. Of the Second Intl. Conf. on Knowledge Discovery and Data Mining (KDD´96),* Menlo Park, CA: AAAI Press, pp: 319-322, 1996.

[DOO97] DOORENBOS, R.B.; ETZIONI, O.; WELD, D. S. A scalable comparison-shopping agent for the World-Wide Web. In *Proceedings of the 1st International Conference on Autonomous Agents*. pp. 39-48, NewYork, ACM Press, 1997.

[DUR87] DURFEE, E. H. *A Unified Approach to Dynamic Coordination: Planning Actions and Interactions in a Distributed Problem Solving Network*, COINS Technical Report 87-84, Univ. of Massachusetts at Amherst, 1987.

[FAY96a] FAYYAD, U.M.; PIATETSKY-SHAPIRO, G.; SMYTH, P. Knowledge Discovery and Data Mining: Towards a Unifying Framework. *Second International Conference on KD & DM.* Portland, Oregon, 1996a.

[FAY96b] FAYYAD, U.M.; PIATETSKY-SHAPIRO, G.; SMYTH, P. *Advances in knowledge discovery & data mining.* Chapter1: From data mining to knowledge discovery: an overview. AAAI/MIT, 1996b.

[FER99] FERBER, J. *Multi-Agent System, An Introduction to Distributed Artificial Intelligence* Addison-Wesley Publishers, 1999.

[FIS96] FISCHER, K.; MULLER, J.P.; PISCHEL, M. Cooperative Transportation Scheduling: an Application Domain for DAI. *Journal of Applied Artificial Intelligence. Special issue on Intelligent Agents*, 10(1), 1996.

[FRA96] FRANKLIN, S.; GRAESSE, A. Is it an agent, or just a program? A taxonomy for autonomous agents *Proceedings of the Third International Workshop on Agent Theories*, Architecture and Languages, 1996, Springer-Verlag.

[FRE98] FREITAS, A.; LAVINGTON, S. H. *Mining very largedatabases with parallel processing* Kluwer Academic Publishers The Netherlands, 1998.

[FRI97] FRIEDMAN-HILL, E. *Jess, the Java Expert System Shell*. Disponível em: http://herzberg.ca.sandia.gov/jess, 1997, acessado em janeiro de 2005.

[FRI02] FRIGO, L.B.; BITTENCOURT, G. MathTutor: Uma ferramenta de apoio a aprendizagem, *Anais do XXII Congresso da Sociedade Brasileira de Computação – X WEI*, 2002.

[GIR98] GIRAFFA, L. M.; VICCARI, R.M.; SELF, J. *Multi-agent based pedagogical games*. Inteligent Tutoring Systems – ITS, 1998.

[HAN01a] HAND, D.; MANNILA, H.; SMYTH, P. *Principals of Data Mining.* MIT Press, Cambridge, Mass, 2001.

[HAN01b] HAN J.; KAMBER, M. *Data Mining: Concepts and Techniques.* Morgan Kaufman Publishers, San Francisco, CA, 2001.

[HAN97] HAND, D. J. *Construction and Assessment of Classification Rules*. John Wiley and Sons, England. 1997 (Capítulos 6, 7 e 8).

[HOL93] HOLTE, R. C. Very simple classification rules perform well on most commonly used datasets. *Machine Learning* 11:63-90, 1993.

[JAT97] JATLite "Java Agent Template Lite" Stanford University, 1997, Disponível em: http://java.stanford.edu/java_agent/html/index2.html, acessado em junho de 2005.

[JEN99] JENNINGS, N. R.; WOOLDRIDGE, M.; KINNY, D. *A Methodology for Agent-Oriented Analysis and Design* Proc. 3rd Int Conference on Autonomous Agents (Agents-99) Seattle, WA. 28 ,1999.

[JEN98] JENNINGS, N.; SYCARA, K.; WOOLDRIDGE, M. *A Roadmap of Agent Research and Development, Autonomous Agents and Multi-Agent Systems*, 1:7-38, 1998.

[JEN96] JENNINGS, N.R. *Coordination tecniques for distributed artificial intelligence*. In: O'HARE, G.M.P.; JENNINGS, N.R. (Eds.). Foundations of distributed artificial intelligence. New York: John Wiley & Sons, p.187- 210, 1996.

[JEN94] JENNINGS, N. R. *Cooperation in Industrial Multi-agent Systems*, World Scientific, 1994.

[KAU97] KAUTZ, H.; SELMAN, B.; SHAH, M. The hidden Web. *AI Magazine*, 18(2):27--36, 1997.

[KAR04] KARGUPTA ,H.; SIVAKUMAR, K. Existential pleasures of distributed data minig. In *Data Mining: Next Generation Challenges and Future Directions*, edited by H. Kargupta, A. Joshi, K. Sivakumar, e Y. Yesha, MIT/ AAAI Press, 2004.

[KAR00] KARGUPTA ,H.; CHAN, P. *Advances in Distributed and Parallel Knowledge Discovery*. AAAI/ MIT, 2000.

[KAR00] KARGUPTA, H. *et. al.* Collective data mining: A new perspective towards distributed data mining. In *Advances in Distributed and Parallel Knowledge Discovery* p. 133–184. Cambridge, MA: AAAI/MIT Press, 2000.

[KAR02] KARGUPTA, H. *et. al*. Mobimine: Monitoring the stock market from a PDA. *ACM SigKDD Explorations, 3,* p. 31–46, 2002.

[KIN95] KING, R.D. *et. al. STATLOG: Comparison of classification algorithms on large real-world problems.* Applied Artificial Intelligence, 9(3). May/June 1995.

[KLE00] KLEIN, M. *et. al. Airborne Remote Sensing of the Super-Cooled Water and Temperature Environment*. NAS3-00051, 2000

[KRU96] KRULWICH, B. *The bargainfinder agent: Comparison price shopping on the internet.* In Agents, Bots, and other Internet Beasties, pages 257--263. Macmillan Publishing, May 1996.

[LAM97] LAM, W.; SEGRE, A. M. Distributed data mining of probabilistic knowledge. In *Proceedings of the 17th International Conference on Distributed Computing Systems* p. 178–185. Los Alamitos, CA: IEEE Computer Society Press, 1997.

[LIE95] LIEBERMAN, H. Letizia: An agent that assists web browsing. In *Proocedings of the International Joint Conference on Artificial Intelligence (IJCAI95),* pages 924-929, 1995.

[LJU92] LJUNBERG, M.; LUCAS, A. The OASIS air traffic management system. In *Proceedings of the Second Pacific Rim International Conference on AI* (PRICAI-92), Seoul, Korea, 1992.

[MAE94] MAES, P. *Agents that Reduce Work and Information Overload*. Commun. ACM 37,7, 31-40, 1994.

[MER99] MERZ, C. J.; PAZZANI, M. J. A principal components approach to combining regression estimates. *MachineLearning,* 36, p. 9–32, 1999.

[MIC94] MICHIE, D. *et. al. Machine learning, neural and statistical classification,* New York: Ellis Horwood, 1994 (Capítulos 1, 2 e 11).

[MIN96] MINAR,N.; LANGTON, C.; BURKHART, R. *The Swarm simulation System: a toolkit for building Multi-agent simulations*, Santa Fe Institute (U.S.A.), 1996.

[MOU96] MOULIN, B; CHAIB-DRAA, B. An Overview of Distributed Artificial Intelligence. In: O'Hare, Greg; Jennings, Nicholas R. (Eds.). *Foundations of distributed artificial intelligence*. [S.l.]: John Wiley and Sons, 1996. cap.1.

[NIS93] NISHIBE, Y. *et. al.* Distributed channel allocation in atm networks. In *Proceedings of the IEEE Globecom Conference*, pages 12.2.1-12.2.7, Houston, TX., 1993.

[OLI97] OLIVEIRA E., FONSECA J.M., GARO A., Maciv: a DAI based resource management system, In *International Journal on Applied Artificial Intelligence*, V.11, N.6, pgs. 525-550, Taylor & Francis, 1997.

[OLI96] OLIVEIRA, F.M. Inteligência Artificial Distribuída. In: IV ESCOLA REGIONAL DE INFORMÁTICA, Canoas, 3, 1996. *Anais: Sociedade Brasileira de Computação*, 239p., 1996.

[OPI99] OPITZ, D.; MACLIN, R. Popular ensemble methods: An empirical study. In *Journal of Artificial Intelligence Research, 11*, p. 169–198, 1999.

[PAR03] PARK, B.; KARGUPTA, H., Distributed Data Mining: Algorithms, Systems, and Applications, In *The Handbook of Data Mining*, edited by N. Ye, Lawrence Erlbaum Associates, pp: 341-358, 2003.

[PAR02] PARK, B. *et. al.* Distributed, collaborative data analysis from heterogeneous sites using a scalable evolutionary technique. *Applied Intelligence, 16,* p.19–42, 2002.

[PAR94] PARUNAK, H. *Applications of Distributed Artificial Intelligence in Industry* Foundations of Distributed Artificial Intelligence. Wiley Inter-Science, cap 4, 1994.

[POK99] POKRAJAC, D. *et. al.* Distribution Comparison for Site-Specific Regression Modeling in Agriculture, *Proc. IEEE/INNS Int'l Conf. on Neural Neural Networks*, Washington, D.C., July, 1999, in press.

[PRO00] PRODROMIDIS, A.; STOLFO, S. Cost complexity-based pruning of ensemble classifiers. In *Workshop on Distributed and Parallel Knowledge Discovery at KDD 2000,* p.30–40, Boston, 2000.

[QUI93] QUINLAN, J. R. *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.

[RAB02] RABELO, R. J. *Sistemas Multi-agente* Disponível em : http://www.das.ufsc.br /~rabelo,2002, acessado em maio de 2005.

[SCH02] SCHROEDER, L. F.; BAZZAN, A. L. C. A multi-agent system to facilitate knowledge discovery: an application to bioinformatics. In *Proceedings of the Workshop on Bioinformatics and Multi-Agent Systems*, Bologna, Italy, 2002.

[SCH97a] SCHROOTEN, R.; VAN DE VELDE, W. *Software agent foundation for dynamic interactive electronic catalogs*. Applied Artificial Intelligence, 11:459--481, 1997.

[SCH97b] SCHOONDERWOERD, R. *et. al.* A*nt-based load balancing in telecommunications networks*. Preprint, 1997.

[SCH93] SCHWUTTKE, U. M.; QUAN, A. G. Enhancing performance of cooperating agents in real time diagnosis systems. *Proc. 13th Int. Joint Conference on Artificial Intelligence,* Chamberry, France, pp 332-337, 1993.

[SEG94] SEGAL, R.; ETZIONI, O. Learning decision lists using homogenous rules. In: *Proc. Of the Twelfth National Conf. on Artificial Intelligence,* 619-625. Seattle, WA: AAAI Press, 1994.

[SCH01] SCHAPIRE, R. E.; FREUND, Y.. The Boosting Approach to Machine Learning: An Overview. In *MSRI Workshop on Nonlinear Estimation and Classification*, Berkeley, CA. 2001

[SIC95] SICHMAN, J S. *Du Raisonnement social chas les Agents: Une Approche Fondée sur la Théorie de la Dépendance*. Tese de doutorado. Laborateire d'Informatique Fondamentale et d'Intelligence Artificielle, Institut National Polytechnique de Grenoble, Set., 1995.

[SIC92] SICHMAN, J.; DEMAZEAU, Y.; BOISSIER, O. When can knowledge-based systems be called agents? In *XII Congresso da Sociedade Brasileira de Computação, IX Simpósio Brasileiro de Inteligência Artificial*, p. 172-185, 1992.

[SIC92] SICHMAN, J S; DEMAZEU, Y; BOISSIER, O. How can knowledge-based systems be called agents? In: *Simpósio Brasileiro de Inteligência Artificial, 9,* 1992, Rio de Janeiro. Anais: Rio de Janeiro: Pontifícia Universidade Católica do Rio de Janeiro, 1992. p.173-185.

[SMI83] SMITH, R. G., DAVIS, R. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20: 63-109, 1983.

[SMI81] SMITH, R. G., DAVIS, R. Frameworks for cooperation in distributed problem solving. *IEEE Trans. On Systems, Man, and Cybernetics* 11(1): 61-70.

[SRI87] SRIDHARAM, N.S. Workshop on Distribuited IA *AI Magazine*, 1986

[STO95] STOLFO, S. J. *et. al.* A parallel and distributed environment for database rule processing: open problems and future directions. In *M. Abdelguerfi & S. Lavington (Ed) Emerging Trends in Database and Knowledge-Base Machine*, 225-253. IEEE, 1995.

[STO97] STOLFO, S. *et. al.* Jam: Java agents for meta-learning over distributed databases. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining,* 1997, p. 74– 81. Menlo Park, CA: AAAI Press.

[SYC96] SYCARA, K. *et. al.* Distributed Intelligent Agents, *IEEE Expert,* December 1996.

[TAK97] TAKAHASHI, K *et. al. Intelligent pages: Collecting shop and service information with software agents.* Applied Artificial Intelligence, 11(6):489-500, 1997.

[TUM00] TUMER, K.; GHOSH, J. Robust order statistics based ensemble for distributed data mining. In H. Kargupta & P. Chan (Eds.), *Advances in distributed and parallel knowledge discovery*, 2000, p. 185–210. Cambridge, MA: MIT Press.

[TUR00] TURINSKY, A. L.; GROSSMAN, R. L. A framework for finding distributed data mining strategies that are intermediate between centralized strategies and in-place strategies. In *Workshop on Distributed and Parallel Knowledge Discovery* , 2000 Boston.

[VIK00] VIKTOR, H.; ARNDT, H. Combining data mining and human expertise for making decisions, sense and policies. *Journal of Systems and Information Technology*, 2000.

[WIR01] WIRTH, R.; BORTH, M.; HIPP, J. When distribution is part of the semantics: A new problem class for distributed knowledge discovery. In *Proceedings of the PKDD-*

2001 Workshop on Ubiquitous Data Mining for Mobile and Distributed Environments. 2001, Freiburg, Germany.

[WIT92] WITTIG, T. *ARCHON : On Architecture for Multi-Agent Systems*, Ellis Horwood, 1992.

[WOO00] WOOLDRIDGE, M. J. *Reasoning about Rational Agents.* MIT Press, 2000.

[WOR01] WORTMANN, H.; SZIRBIK, N. ICT Issues among Collaborative Enterprises: from Rigid to Adaptive Agent-Based Technologies *International Journal of Production Planning and Control 12 (5)* p. 452-465, 2001.

[WUT00] WUTHRICH, B. *et. al.* Data quality in distributed environments, In H. Kargupta and P. Chan, editors, *Advances in Distributed and Parallel Knowledge Discovery*, pp 278-294. AAAI/MIT Press, 2000.

[YAM97] YAMANISHI, K. Distributed cooperative bayesian learning strategies. In *Proceedings of COLT'97* , 1997, p. 250–262. New York: ACM.