2
Sistemas de Mineração de Dados

2.1.
Introdução

O processo de KDD (*Knowledge Discovery in Databases*) possui várias etapas. A Figura 1 apresenta as etapas do processo, demonstrando que antes da extração de informação (mineração de dados) é necessária uma grande manipulação dos dados para viabilizar a sua exploração.

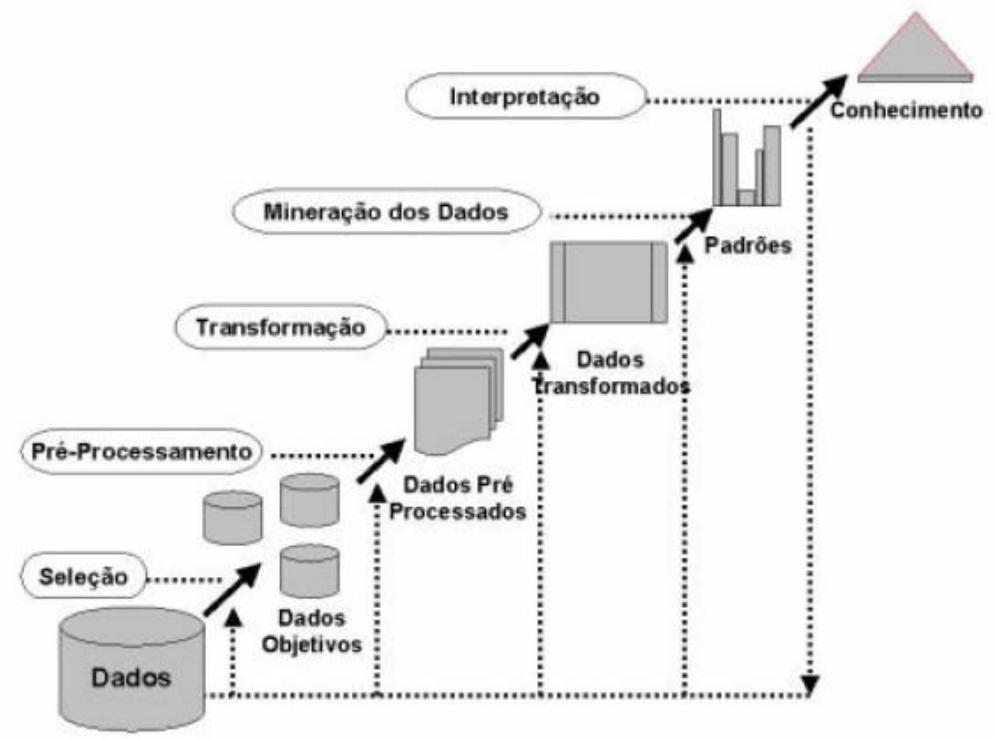

Figura 1: Fases do Processo de KDD

Antes de se realizar a mineração dos dados, é realizada a sua seleção. Nesta etapa, são definidos os dados realmente importantes para a extração de informações e posteriormente do conhecimento.

Em seguida, é realizado um pré-processamento dos dados, onde se deseja reduzir a complexidade do problema. Essa fase também é responsável pela limpeza dos dados. As inconsistências são identificadas, *outliers* são eliminados e os atributos mais significativos são selecionados, reduzindo a dimensionalidade do problema e evitando que variáveis correlatas sejam utilizadas em conjunto na análise.

A fase de transformação dos dados busca codificar e transformar os dados para facilitar o processo de mineração. As variáveis contínuas podem ser codificadas em variáveis categóricas. Por exemplo, uma variável representante da renda mensal de pessoas pode ser codificada, a partir de faixas de valores, em uma variável categórica com os seguintes valores: renda baixa, renda média e renda alta. A normalização dos valores de uma variável também pode ser considerada uma transformação, onde os valores vão passar a se encontrar no intervalo [0,1], por exemplo.

Para essas três primeiras fases do processo de KDD, normalmente é construído um *Data Warehouse* (DW). DWs são bancos de dados voltados para consulta, geralmente possuindo dados consolidados, de caráter histórico, não-operacionais e provenientes de diversas fontes. Com um DW montado, tem-se um repositório central com todos os dados necessários para a exploração, já selecionados, limpos e transformados.

A Mineração de Dados (*Data Mining*) é a principal etapa do processo de KDD, pois é a responsável por extrair informações implícitas e potencialmente úteis a partir dos dados. Essas informações serão utilizadas pelos usuários ou por outros sistemas para absorção de conhecimento e principalmente para tomada de decisões estratégicas. A mineração de dados é proveniente da técnica de aprendizado por máquina (*machine learning*) [LANA00] e atualmente, diversas técnicas são usadas para realizar essa tarefa, sendo que algumas delas serão mencionadas ainda nesta seção.

Alguns pesquisadores descrevem a Mineração de Dados como sinônima do processo de KDD ao invés de colocá-la como parte do mesmo. Nesta dissertação, a mineração de dados é encarada como a técnica ou o algoritmo que, de fato, extrai as informações implícitas dos bancos de dados. É igualmente importante não confundir mineração de dados com ferramentas de consulta OLAP (*On-Line Analytical Process*), as quais são responsáveis por facilitar e flexibilizar consultas a DW's multidimensionais, porém sem a característica de descobrir informações ocultas no mesmo.

2.2. Classificação de Padrões

A linha de aplicações mais estudada na Mineração de Dados é a Classificação de Padrões. Essa tarefa tem como principal objetivo identificar a quais classes pertencem os padrões sendo analisados. Um padrão pode ser definido como um vetor de valores que os atributos selecionados podem assumir.

Os padrões de classificação possuem atributos de dois tipos: os preditivos e os objetivos. Geralmente, um atributo objetivo corresponde a uma variável categórica, de preferência com poucas categorias, representando as classes previamente definidas; logo são conhecidos apenas na fase de treinamento (treinamento supervisionado). Os atributos preditivos são os utilizados pela técnica para inferir a que classe um novo padrão pertence. Dessa forma, o principal objetivo do algoritmo de classificação, em sua fase de treinamento, é identificar e aprender correlações entre atributos preditivos e objetivos.

O processo de classificação de padrões é realizado em duas fases. Inicialmente o algoritmo é treinado com padrões previamente classificados e, portanto, essa fase é chamada *in-sample*. Na segunda fase, denominada *out-of-sample*, o modelo utilizado deve ser capaz de receber novos padrões, com classificação desconhecida, e classificá-los corretamente após inferência sobre os atributos preditivos.

Em muitos casos, os padrões de treinamento não possuem as classes previamente definidas. Nesse caso, antes do processo de classificação é feito um agrupamento, onde um algoritmo descobre as classes inerentes aos padrões de treinamento. Algoritmos de agrupamento têm como principal objetivo maximizar a variância inter-classes e minimizar a variância intra-classes, agrupando em uma mesma classe padrões que possuem forte relação entre si. Os algoritmos de agrupamento mais utilizados são os hierárquicos [JOHN99], o K-Means [JOHN99], o Fuzzy C-means [DUDA73] e o Mapa de Kohonen [KOHO97].

2.3. Previsão de Séries Temporais

Esta tarefa de Mineração de Dados tem como principal objetivo prever, a partir de uma série de valores conhecidos, novos valores em um instante de tempo predefinido no futuro ($t+k$, onde $k \geq 1$).

A previsão de séries temporais é semelhante à classificação, pois da mesma forma deve inferir um atributo objetivo a partir de atributos preditivos. Porém, em previsão, normalmente o atributo objetivo é contínuo, diferentemente do processo de classificação. Outra diferença está na relação entre os padrões, pois neste caso eles compõem uma série temporal. Essas pequenas diferenças fazem com que algoritmos de classificação e previsão sejam muitas vezes diferentes.

Assim como na classificação de padrões, o modelo de previsão precisa selecionar os atributos preditivos que serão utilizados em seu treinamento. Um dos principais conjuntos de atributos diz respeito à janela de previsão. A janela define a quantidade de instantes, consecutivos ou não, de tempo na série que serão utilizados como atributos preditivos. Esse é um parâmetro específico de previsão de séries temporais, normalmente definido por um especialista ou através de análise estatística de correlação.

2.4. Técnicas de Mineração de Dados

2.4.1. Métodos Estatísticos

Dentre os métodos estatísticos mais utilizados em tarefas de classificação, destacam-se o método Linear de Fisher [DUDA01], Classificadores Bayesianos [CHEN96] e Redes Bayesianas [HECK96]. A seguir, cada um desses métodos será brevemente descrito.

2.4.1.1. Discriminante Linear de Fisher

O Discriminante Linear de Fisher, utilizado em tarefas de classificação, apresenta resultados promissores para casos em que as variáveis preditivas seguem uma distribuição normal multivariada e quando a variância inter-classes é bem próxima, apesar dessa configuração dificilmente ocorrer.

Esse método baseia-se na projeção do espaço onde se encontram as variáveis preditivas em um espaço de dimensão $q = \min\{p,(c-1)\}$, onde *p* é o número de variáveis e *c* o de classes. O algoritmo busca projetar as variáveis de

forma a aumentar a distância dos centróides das classes e minimizar a variância intra-classes. A partir dessa projeção são extraídas funções discriminantes que são aplicadas a novos padrões. A função que assumir maior valor irá determinar a classe do padrão em questão.

2.4.1.2. Classificadores Bayesianos

Este método baseia-se na Teoria de Probabilidade Bayesiana [SHEN93], representada pela seguinte fórmula básica:

$$P(A \mid B) = \frac{P(B \mid A) * P(A)}{P(B)} \qquad \text{Eq. 1}$$

Na Eq. 1, $P(A)$ representa a probabilidade do evento *A* ocorrer, independente de *B*. $P(B)$ é a probabilidade de *B* ocorrer, independente de *A*. Finalmente, $P(B/A)$ é a probabilidade do evento B ocorrer, dado que A ocorre. A teoria de probabilidade Bayesiana baseada nesses dados, através da Eq. 1, determina a probabilidade do evento *A* ocorrer, dado que *B* ocorre.

Na tarefa de classificação de padrões, a regra de Bayes pode ser utilizada para determinar as probabilidades dos padrões pertencerem a cada uma das classes em questão:

$$P(C = c \mid A_1 = a_1 \cap A_2 = a_2 \cap \ldots \cap A_k = a_k) \qquad \text{Eq. 2}$$

Um exemplo prático dessa expressão seria: “Qual a probabilidade de um indivíduo comprar um carro esportivo, dado que ele recebe um salário de cinco mil, mora em um bairro nobre, troca de carro de dois em dois anos e possui vinte anos?”, representado por $P(esporte \mid [5.000, nobre, 2, 20])$.

A grande vantagem desse método é a sua facilidade de interpretação porém são necessários muitos padrões de treinamento para se atingir resultados satisfatórios. Em [CHEN96] é mostrado um modelo detalhado utilizando métodos Bayesianos denominado *AutoClass*.

2.4.1.3. Redes Bayesianas

A Rede Bayesiana é um método que alia a experiência e o conhecimento de especialistas com as estatísticas dos dados. Esse método é realizado em duas etapas. Primeiro, um ou mais especialistas codificam seu conhecimento em Redes Bayesianas. Em seguida, utiliza-se o banco de dados para atualizar o conhecimento e criar uma ou mais novas Redes Bayesianas, melhorando as informações previamente fornecidas.

As redes bayesianas são parecidas com as Redes Neurais Artificiais, descritas mais a frente, porém com duas importantes diferenças: possuem maior facilidade de interpretação de seus resultados e fornecem a possibilidade de codificação de conhecimento por parte de um ou mais especialistas. O fato de existir a necessidade de um especialista para codificar o conhecimento torna este método difícil de ser automatizado, indo de encontro ao objetivo principal deste trabalho que é criar um modelo automático.

A referência [HECK96] apresenta maiores informações com relação a Redes Bayesianas, demonstrando em detalhes como aplicar esse método na descoberta de conhecimento a partir de bancos de dados.

2.4.2. Árvores de Decisão

As árvores de decisão representam uma técnica de classificação onde as regras de decisão estão dispostas na forma de uma árvore. A representação segue o seguinte padrão:

- Cada nó da árvore é rotulado por um dos atributos preditivos;
- Os arcos que interligam os nós são rotulados com os valores que a variável correspondente pode assumir;
- As folhas são nomeadas com as classes do problema.

No momento em que se deseja classificar um novo padrão, percorre-se os nós da árvore iniciando pela raiz e tomam-se decisões em cada nó a partir do valor assumido pelo atributo. Assim como a criação da árvore, a classificação é feita de forma *top-down*; logo a classificação termina quando a busca através da árvore atinge uma folha. O mesmo atributo pode aparecer mais de uma vez na árvore e podem existir caminhos na árvore onde alguns atributos não apareçam.

A Figura 2 mostra um exemplo de árvore de decisão onde são avaliados atributos que indicam se uma pessoa é casada ou empregada, com o objetivo de

avaliar o risco de crédito. A partir de uma árvore de decisão são geradas regras, onde cada caminho possível da árvore corresponde a uma regra do tipo SE-ENTÃO. Os antecedentes são formados pelos atributos preditivos que aparecem ao longo do caminho percorrido, testando os valores que os definem. Os conseqüentes são formados pelo atributo objetivo com a classe correspondente à folha. No exemplo da Figura 2, temos a seguinte regra: “Se empregado e casado então crédito aprovado.” Uma das desvantagens deste método é o frequente aparecimento de atributos irrelevantes nas regras (o atributo da raiz aparece em todas as regras), entretanto as árvores de decisão são eficientes computacionalmente.

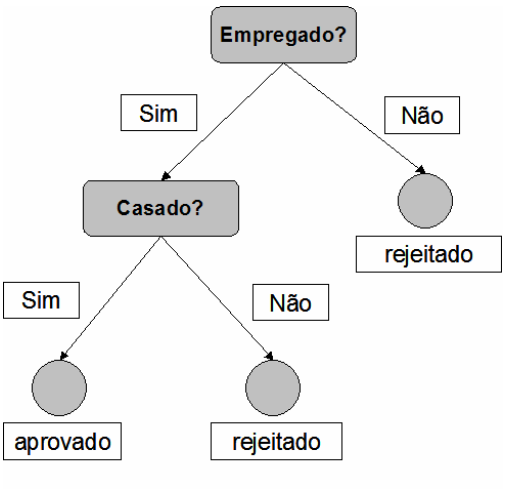

Figura 2: Exemplo de uma Árvore de Decisão

As características descritas acima são comuns a todas as técnicas baseadas em árvores de decisão; todavia o algoritmo de criação da árvore pode variar, além de outros detalhes como a forma de se realizar a decisão do melhor caminho em um nó e de como tratar atributos contínuos. Nas próximas seções serão brevemente descritos os algoritmos ID3, C4.5, CART e FID3.1.

2.4.2.1. ID3 e C4.5

O algoritmo ID3 foi desenvolvido por Ross Quinlan em 1983 [QUIN87]. Esse algoritmo avalia a informação contida nos atributos segundo a sua entropia. O atributo mais importante é colocado na raiz e, de forma *top-down*, a árvore é construída recursivamente.

O algoritmo ID3 não trabalha com atributos do tipo contínuo, limitando-o excessivamente. Sendo assim, sucessivas melhorias foram realizadas no algoritmo para que aceitasse atributos contínuos, nascendo o algoritmo C4.5 [QUIN93]. O algoritmo C4.5 também suporta valores desconhecidos para o caso de padrões incompletos.

2.4.2.2. CART (*Classification And Regression Trees*)

O algoritmo CART gera sua árvore de decisão realizando particionamentos binários no domínio dos atributos, gerando nós com apenas dois caminhos a seguir: sim ou não. Da mesma forma que o algoritmo C4.5, o CART é recursivo e pode lidar com dados ausentes e diversos tipos numéricos (contínuos, categóricos, booleanos, etc.).

Entretanto, por realizar particionamentos binários, esse algoritmo apresenta dificuldades de trabalhar com atributos que podem assumir mais de duas classes (ex.: baixo, médio e alto). Nesses casos, podem existir dois ou mais nós para representar o mesmo atributo, gerando árvores maiores e mais complexas.

2.4.2.3. FID3.1

O FID3.1 é uma árvore de decisão fuzzy, ou seja, as decisões tomadas nos nós da árvore seguem critérios fuzzy. Dessa forma, esse método apresenta uma maior generalização que as tradicionais árvores *crisp*. O FID3.1 foi proposto por Cezary Z. Janikow [JANI99] e é utilizado por modelos híbridos na extração de regras de classificação.

Esse algoritmo tem características semelhantes às árvores *crisp,* como o aprendizado *top-down* recursivo e a possibilidade de lidar com diversos tipos de atributos, porém tem alguns detalhes próprios, tais como: os padrões estão distribuídos nas partições através de funções de pertinência e os resultados variam em função das funções de pertinência e dos operadores utilizados.

Este algoritmo mostrou-se mais eficaz que as árvores de decisão *crisp* em aplicações *benchmark*, como demonstrado em [IRIA00].

2.4.3. Regras de Associação

Regras de Associação [KARU97] buscam, a partir de transações de bancos de dados, encontrar associações do tipo “ocorrem juntos” ou “implica em” entre itens. Por exemplo, pode-se gerar uma regra do tipo *X*->*Y*, onde *X* e *Y* são conjuntos de itens presentes na transação. O significado intuitivo dessa regra é que nas transações em que *X* ocorre, *Y* tende a ocorrer também.

Tender não significa que necessariamente ocorrerão juntos; para isso cada regra possui um suporte e uma confiança representados em porcentagem. O suporte de uma regra *X*->*Y* indica, dentre o total de transações, o número de vezes que os itens *X* e *Y* aparecem. A confiança de uma regra do mesmo tipo indica o número de vezes que, quando *X* ocorre, *Y* também ocorre. Um exemplo real de regra de associação seria: “50% das transações em que aparecem vinho tinto, queijo também aparece; 1% do total de transações contém esses dois itens”. Nesse caso, a regra possui 50% de confiança e 1% de suporte.

O especialista deve especificar os valores mínimos de suporte e confiança das regras para que assim essas possam ser filtradas com relação a sua qualidade. Esse método é interessante na descoberta de regras, porém não trata de tarefas de classificação e previsão, as quais são importantes em mineração de dados e serão exploradas nesta dissertação.

Os principais algoritmos de regras de associação são: A Priori e Paralelo\Distribuído. Esses algoritmos não serão explicados em detalhes, porém maiores informações podem ser encontradas em [AGRA94][CHEU96].

2.4.4. Algoritmos Genéticos

Os Algoritmos Genéticos são normalmente utilizados na mineração de dados para realizar tarefas de classificação de padrões, descrição de registros e seleção de atributos de bases de dados [GONÇ01].

Na classificação de padrões, os AGs podem buscar a evolução de regras que representem o domínio do problema. Essas regras podem ser, por exemplo, regras fuzzy, e de uma forma geral são de fácil interpretação, o que incentiva o uso dessa técnica.

Serão brevemente descritos a seguir dois dos mais conhecidos sistemas baseados em AGs para realizar tarefas de mineração de dados, o GA-Miner [RADC95] e o Rule Evolver [LOPE99]. Outros sistemas importantes são: LS-1

[SMIT80][SMIT84], GABIL [DEJO93], COGIN [GREE93] [GREE94], REGAL [GIOR94] e GA/Rule [PEI 95].

2.4.4.1. GA-Miner

O GA-Miner é um sistema que pode lidar com tarefas direcionadas de descoberta de conhecimento, não-direcionadas e para refinamento de hipóteses.

Quando a tarefa é não-direcionada, o mecanismo é mais genérico e não é especificado o tipo de regra desejado. A pergunta feita ao sistema pode ser, por exemplo: “Diga-me alguma coisa interessante sobre os meus dados”.

Já na mineração direcionada, o sistema recebe uma especificação que direciona a evolução. Uma pergunta ao sistema seria: “Caracterize os clientes que não são de confiança”.

Para refinamento de hipóteses, o usuário informa ao sistema uma hipótese para teste. O sistema avalia a veracidade da mesma e caso seja positiva, tenta refiná-la. Um exemplo de hipótese fornecida seria: “Acredito que pais de família trocam de carro todo ano. Estou certo?”.

2.4.4.2. Rule Evolver

O Rule Evolver, proposto em [LOPE99], é uma ferramenta que nasceu com o objetivo de diferenciar um grupo de padrões dos demais existentes no banco de dados, extraindo regras de associação.

A ferramenta está dividida em diversos módulos, cujas principais características são: seleção dos atributos do banco de dados; cálculo de informações estatísticas dos atributos; escolha dos operadores genéticos (4 para *crossover* e 2 para mutação); visualização do desempenho do sistema; e interpretação de regras de produção (Se - então).

2.4.5. Redes Neurais Artificiais

A Rede Neural Artificial (RNA) é uma técnica que busca simular a estrutura de um cérebro humano computacionalmente.

Da mesma forma que em um cérebro humano, as redes neurais apresentam uma estrutura altamente paralelizada, composta por processadores simples (neurônios artificiais) conectados entre si.

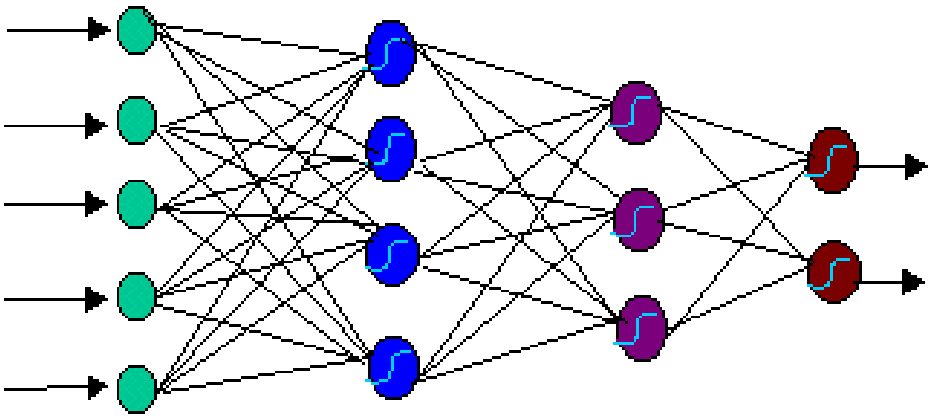

Figura 3: Exemplo de Topologia de Uma Rede Neural Artificial Perceptron Multicamadas

A Figura 3 apresenta um exemplo de topologia de uma rede neural. Percebe-se que existe uma primeira camada que recebe os valores de entrada, chamada camada de entrada. A última camada, denominada camada de saída, é a responsável por gerar as saídas da rede. Podem existir zero ou mais camadas intermediárias, chamadas camadas escondidas, cujo número de neurônios depende do problema.

As redes neurais artificiais são utilizadas na mineração de dados em tarefas como classificação, previsão e agrupamento.

Normalmente, quando se configura uma rede neural para realizar classificação de padrões, a camada de entrada tem tantos neurônios quanto atributos preditivos, e a camada de saída possui tantos neurônios quanto classes no problema. Dessa forma, a partir de um padrão apresentado à rede, a saída que for mais ativada determinará a classe correspondente.

Em previsão de séries temporais, normalmente a rede possui tantos neurônios quanto forem os valores dentro da "janela" de previsão escolhida. A camada de saída possui um único neurônio que irá gerar o valor da previsão para o instante futuro desejado.

Geralmente, tanto em classificação quanto em previsão as redes neurais utilizadas são do tipo *feed-forward* com aprendizado baseado no algoritmo *back-propagation*. O treinamento é realizado de forma supervisionada, ou seja, é preciso que existam padrões com atributos objetivos definidos.

Na tarefa de clusterização ou agrupamento, o treinamento é não-supervisionado, ou seja, não existem atributos objetivos definidos. Nesses casos, são utilizadas redes neurais auto-organizáveis, onde o Mapa de Kohonen

[KOHO97], baseado em aprendizado competitivo, vem apresentando resultados muito satisfatórios.

O problema indicado por especialistas na utilização de Redes Neurais Artificiais em tarefas de classificação e previsão é o fato dessas serem estruturas do tipo “caixa-preta”, ou seja, de difícil interpretação de seus resultados. Isso se deve ao fato de existirem camadas escondidas com possivelmente vários neurônios contendo vários pesos, o que se torna praticamente impossível de interpretar.

Recentemente, muitos trabalhos foram realizados para extrair regras a partir de redes neurais previamente treinadas. Com isso, o problema de interpretação dos resultados foi minimizado, encorajando a utilização desta técnica. Existem três vertentes de algoritmos de extração de regras:

- **Decomposição:** Utilizam informações dos pesos internos na geração das regras [THRU93] [NEAL96];
- ***Black-Box*:** Não utilizam informações internas, apenas fazem correlações das entradas com as saídas geradas [SETI96][SETI97];
- **Ecléticos:** Utilizam princípios dos dois anteriores [CRAV94], [TICK96].

2.4.6. Sistemas Neuro-Fuzzy

Os sistemas híbridos neuro-fuzzy nasceram com o objetivo de aliar o melhor de duas técnicas: as redes neurais artificiais e os sistemas de inferência fuzzy [MEND95][KLIR95]. Como foi visto no item anterior, as redes neurais possuem alta capacidade de aprendizado, porém a sua estrutura é de difícil interpretação. Já os sistemas fuzzy possuem regras de fácil interpretação. Fundamentos e conceitos básicos dessa técnica podem ser encontrados em [ZADE65]. A junção das duas modelagens gerou métodos muito mais adequados para a realização de tarefas de mineração de dados. Existem outros tipos de sistemas híbridos mas os sistemas neuro-fuzzy são os mais estudados na atualidade [SOUZ99].

A seguir, serão apresentados três sistemas neuro-fuzzy largamente utilizados da atualidade: ANFIS [JANG93], FSOM [VUOR94] e NEFCLASS [KRUS95].

2.4.6.1. Sistema ANFIS

O sistema ANFIS ou *Adaptive Network-based Fuzzy Inference System* foi proposto por Jang [JANG93] para a realização de tarefas de previsão e aproximação de funções.

A Figura 4 apresenta um exemplo de arquitetura deste método. A seguir, as camadas existentes serão descritas, em detalhe, para melhorar o entendimento do sistema.

- Camada 1: É a camada responsável por fuzzificar as entradas. Para cada valor do padrão de entrada, os graus de pertinência dos termos lingüísticos são calculados. Essas funções podem assumir diferentes formatos e os valores que os definem são considerados pesos fuzzy. No exemplo, existem apenas dois conjuntos fuzzy para cada variável, Alto (A) de Baixo (B).
- Camada 2: Esta é a camada de regras. Os antecedentes são provenientes da camada anterior e diferentes para cada regra. Nessa camada são realizadas operações t-norm (por isso a indicação com um **T**) que geram o nível de disparo da mesma. No exemplo dado, as saídas dos neurônios da camada 2 são calculados da seguinte forma:

$$\begin{aligned} S_1 &= A_1(x_1) * A_2(x_2) * A_3(x_3) \\ S_2 &= B_1(x_1) * B_2(x_2) * A_3(x_3) \\ S_3 &= B_1(x_1) * B_2(x_2) * B_3(x_3) \end{aligned} \qquad \text{Eq. 3}$$

- Camada 3: Esta camada é responsável por normalizar os valores dos níveis de disparo das regras, calculados na camada 2. Isso nem sempre é realizado em arquiteturas semelhantes, porém seu objetivo é realizar um pré-processamento para a defuzzificação. As saídas dos neurônios desta camada no exemplo da Figura 5 são dadas por:

$$\begin{aligned} S_1' &= S_1/(S_1 + S_2 + S_3) \\ S_2' &= S_2/(S_1 + S_2 + S_3) \\ S_3' &= S_3/(S_1 + S_2 + S_3) \end{aligned} \qquad \text{Eq. 4}$$

- Camada 4: É a camada responsável por realizar a inferência de fato. A saída dos neurônios é calculada multiplicando-se o nível de disparo das regras normalizado (saídas da camada 3) pelos conseqüentes correspondentes, indicados na Eq. 5 por C_i. Os conseqüentes podem ser *singletons* (constantes) ou combinação linear das entradas. Dessa forma, as saídas desta camada são calculadas da seguinte forma:

$$\begin{aligned} h_1 &= S_1^{'}.C_1 \\ h_2 &= S_2^{'}.C_2 \\ h_3 &= S_3^{'}.C_3 \end{aligned} \qquad \text{Eq. 5}$$

- Camada 5: O único neurônio presente nesta camada é responsável por realizar a defuzzificação e, conseqüentemente, calcular a saída do sistema. A saída é uma soma dos valores provenientes da camada anterior, representando uma defuzzificação por média ponderada, já que os níveis de disparo das regras foram normalizados. Sendo assim, a saída do sistema de exemplo é dada por:

$$Z = h_1 + h_2 + h_3 = \frac{S_1.C_1 + S_2.C_2 + S_3.C_3}{S_1 + S_2 + S_3} \qquad \text{Eq. 6}$$

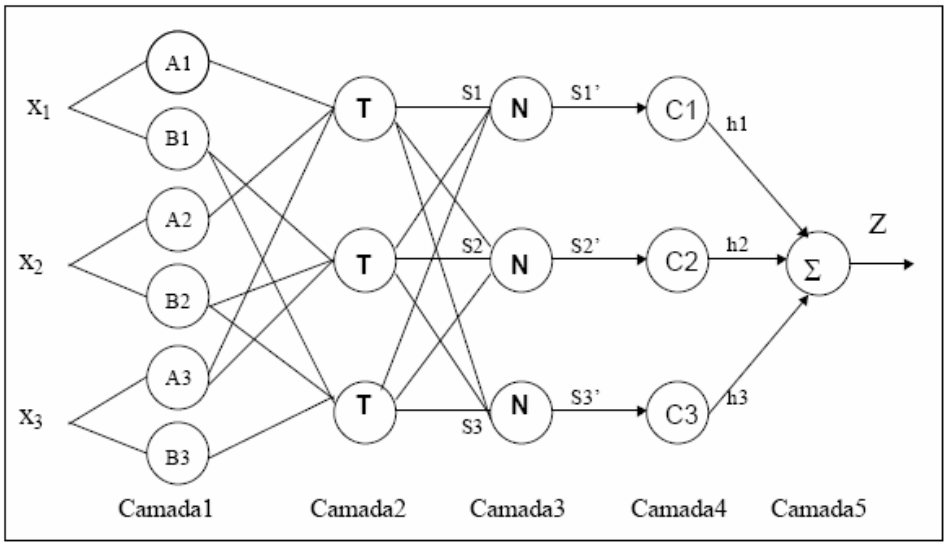

Figura 4: Exemplo de Arquitetura de um Sistema ANFIS

Esse sistema utiliza o particionamento *adaptive fuzzy-grid*. A estrutura da rede é gerada previamente e os pesos fuzzy são atualizados em dois passos: *backward* e *forward*. No passo *backward*, os conseqüentes ficam fixos e os antecedentes são atualizados por gradiente descendente. No passo *forward*, os conseqüentes são atualizados por Mínimos Quadrados Ordinários (MQO), enquanto os antecedentes permanecem inalterados.

2.4.6.2. FSOM

O *Fuzzy Self-Organized Map* ou FSOM, proposto por Vuorimaa [VUOR94], possui uma estrutura muito semelhante ao modelo ANFIS, como pode ser visto da Figura 5. As definições das camadas são as mesmas, com exceção de alguns detalhes que serão descritos.

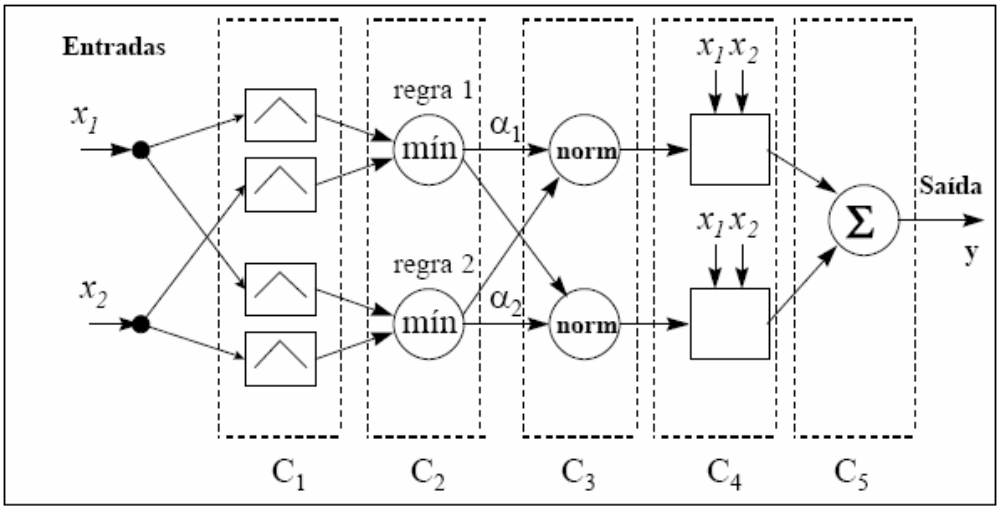

Figura 5: Exemplo de Arquitetura de um Sistema FSOM

Este método utiliza particionamento do tipo *Fuzzy-Box*. Além disso, as funções de pertinência utilizadas nos antecedentes são de formato triangular, caracterizadas por três valores: centro, largura esquerda e largura direita. Os conseqüentes, da mesma forma que o sistema ANFIS, podem ser *singletons* ou combinação linear das entradas. Outra diferença é o fato de a operação *t-norm* para cálculo do nível de disparo das regras ser o mínimo, como na expressão a seguir:

$$\alpha_i = mín\{\mu U_{i1}(x_1), \mu U_{i2}(x_2), ..., \mu U_{in}(x_n)\} \qquad \text{Eq. 7}$$

Onde:

- α_i é o nível de disparo sendo calculado para a regra *i*;
- $\mu U_{ij}(x_j)$ representa o grau de pertinência da valor x_j no conjunto U_{ij}.

A identificação da estrutura é feita através de um mapa auto-organizável que identifica *clusters*. A partir desses *clusters*, as regras são geradas automaticamente e as funções de pertinência são formadas seguindo o centro dos mesmos.

Neste método, os conseqüentes são atualizados através de *back-propagation* e os antecedentes através de um algoritmo supervisionado (LVQ).

Mais detalhes e aplicações deste método podem ser encontrados em [VUOR94][OJAL94].

2.4.6.3. NEFCLASS

O sistema NEFCLASS foi proposto por Nauck e Kruse [CARP91] com o objetivo principal de realizar classificação de padrões. Nauck chegou a desenvolver outros sistemas neuro-fuzzy para outras aplicações: o NEFCON [NAUC94] para controle e o NEFPROX [NAUC94b] para previsão.

A estrutura do NEFCLASS pode ser vista no exemplo da Figura 6 e as camadas existentes no sistema serão descritas a seguir:

- *Camada de Entrada*: responsável por repassar os valores dos atributos à camada de antecedentes.
- *Camada de Antecedentes*: responsável por calcular as pertinências dos valores repassados pela camada anterior. Este método utiliza três conjuntos fuzzy: alto, médio e baixo. Os três são computados para cada variável.
- *Camada de Regras*: realiza a operação *t-norm* com as pertinências correspondentes às regras que define.
- *Camada de Saída*: possui como número de neurônios a quantidade de classes do problema. Cada neurônio recebe os níveis de disparo das regras que contém a classe correspondente como conseqüente

e realiza uma operação *t-conorm* gerando a saída. Os pesos sinápticos que definem a importância de cada regra são iguais a 1 (todas regras possuem mesmo peso).

A saída mais ativada determinará a classe a que o padrão de entrada pertence. Este método utiliza particionamento *Adaptive Fuzzy-Grid* e suas regras são criadas a partir de uma heurística que mantém as melhores regras, como pode ser visto em [CARP91]. Seus antecedentes são atualizados a partir de um algoritmo de *backpropagation* do erro modificado e os conseqüentes não são atualizados por não fazer sentido, já que devem ser fixos.

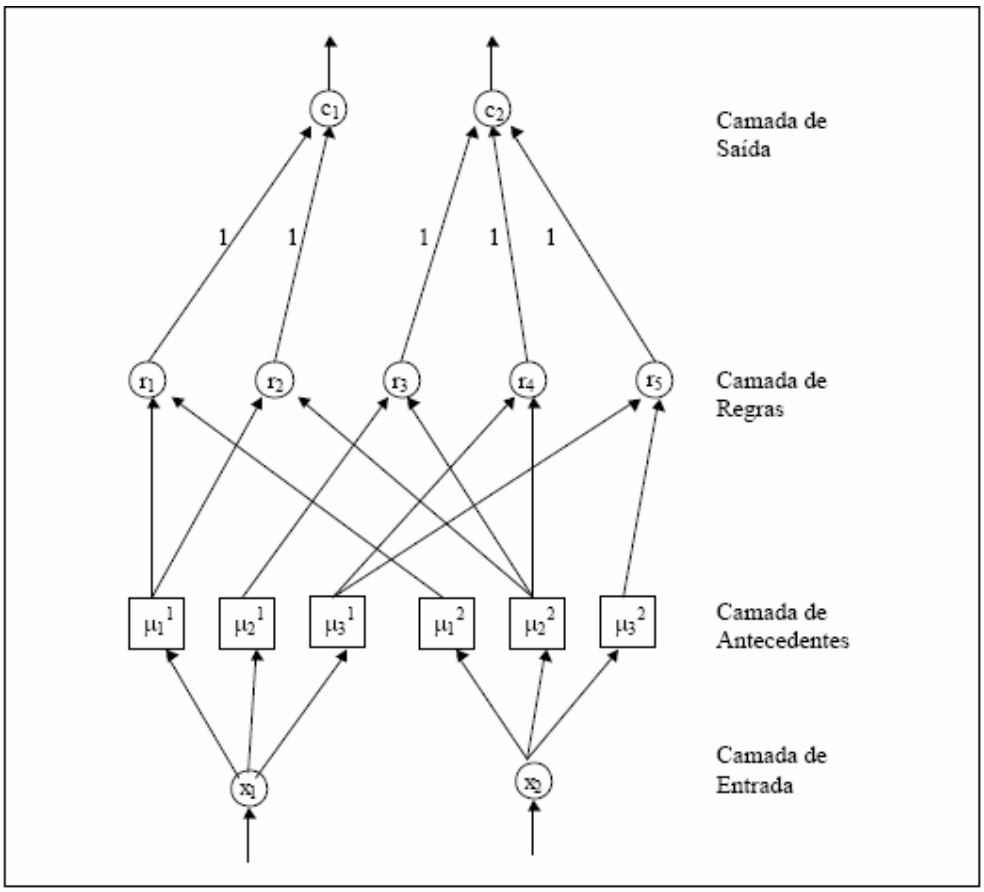

Figura 6: Exemplo de Arquitetura do Sistema NEFCLASS

No capítulo a seguir, será descrito em detalhes o Sistema Neuro-Fuzzy Hierárquico BSP e seus parâmetros. Esse sistema proposto por [SOUZ99], inspirado pelos modelos descritos nesta seção, foi utilizado como base para a modelagem do Sistema Híbrido Neuro-Fuzzy-Genético, objetivo principal desta dissertação.