# 3
# Técnicas e Modelos para Análise de Dados

## 3.1.
## Introdução

Este capítulo apresenta uma revisão sucinta dos mais importantes métodos de transformação e análise de dados, incluindo técnicas e formas de preparação de dados para um melhor desempenho dos modelos a serem utilizados e várias técnicas de modelagem distintas na busca pelo conhecimento.

## 3.2.
## Preparação dos Dados

A seguir serão discutidas algumas técnicas e práticas utilizadas na preparação de uma base de dados para a extração do conhecimento através de modelos de mineração de dados. Tais procedimentos são essenciais para garantir a consistência dos dados e uma representação dos mesmos que facilite a compreensão pelos modelos que serão aplicados. É importante salientar que as técnicas aqui apresentadas são somente alguns exemplos das ações que podem ser tomadas para uma melhoria da qualidade dos dados. Muitos outros procedimentos disponíveis na literatura [BALL99] [BERR00] [PYLE99] podem e devem ser aplicados, dependendo dos dados e da questão a ser solucionada.

Os métodos em maior detalhe nesta seção serão utilizados no sistema de retenção de clientes proposto por se encaixarem na natureza das bases de dados do problema do *churn*: dados empresariais coletados de fontes distintas e com grande chance de apresentar imperfeições ou necessitar de uma representação mais adequada para um melhor desempenho dos modelos [MOZE00] [YAN01] [YAN04].

### 3.2.1.
### Tratamento e Limpeza

Limpeza de dados visa detectar e remover anomalias presentes nos dados com o objetivo de aumentar e melhorar a sua qualidade. Tipicamente o processo de limpeza de dados não pode ser executado sem o envolvimento de um perito no negócio ao qual correspondem os dados, uma vez que a detecção e correção de anomalias requerem conhecimento especializado. A limpeza dos dados envolve uma verificação da consistência das informações, a correção de possíveis erros e o preenchimento ou a eliminação de valores nulos e redundantes. Nessa fase são identificados e removidos os dados duplicados e corrompidos. A execução dessa fase corrige a base de dados eliminando consultas desnecessárias que seriam executadas pelos modelos e que afetariam o seu desempenho.

Um exemplo comum na limpeza de dados é a procura por valores absurdos que não deveriam existir na base simplesmente por serem impossíveis na prática. Boas ilustrações disso são bases de dados que possuem idades ou tempos de contrato com clientes. Por vezes encontram-se clientes que possuem mais de 120 anos de vida, ou até mesmo clientes com menos de 2 anos de idade. Da mesma forma, encontrar consumidores que possuem um relacionamento de 400 anos com a empresa não é tão incomum assim. Esses valores são oriundos provavelmente de erros de digitação ou de preenchimento de cadastros. No esforço para limpeza e consistência dos dados, tais campos, mesmo sendo raros, devem ser preenchidos com valores possíveis, utilizando-se, por exemplo, médias ou medianas da variável. Outra opção seria a eliminação do registro que contém tal valor. A filosofia por trás dessas ações é evitar que tal valor atrapalhe a compreensão dos dados pelos modelos, levando-os a tomar conclusões errôneas.

Outro caso interessante de limpeza de dados é o tratamento de valores ausentes (*missing*). Se o número de observações ausentes for significativo, o desempenho de grande parte dos modelos de análise de dados pode ser sériamente comprometido. Para lidar com valores ausentes, em geral utiliza-se uma das seguintes abordagens:

- Ignorar a descrição do indivíduo ou mesmo eliminar o descritor;
- Preencher os valores ausentes manualmente;

- Usar uma constante global para representar os valores ausentes (não recomendado, pois o sistema pode identificar esse valor como alguma característica importante da variável se for muito frequente);
- Usar a média (ou a moda);
- Usar a média (ou a moda) por classe;
- Usar o valor mais provável segundo um modelo (regressão, regra de Bayes, árvores de decisão).

Cada um destes métodos possui vantagens e desvantagens ao ser aplicado. Simplesmente ignorar o padrão que possui valores ausentes é indicado para quando os dados são abundantes, mas pode ser impraticável se os dados são escassos ou contra-indicado se o padrão possui mais informação importante além das variáveis com valor ausente. Preencher o campo com algum valor (manualmente ou com uma constante, média, moda ou valor mais provável segundo algum modelo) salva o padrão da eliminação e aproveita todo o resto da sua informação, mas pode causar desempenho tendencioso na modelagem, principalmente se os valores ausentes forem muitos, ao criar uma realidade sobre os padrões que pode estar distante da verdade. Isso poderia levar algum modelo a considerar certas estruturas de comportamento nos dados que não deveriam existir.

[BALL99] e [PYLE99] apresentam em detalhes inúmeros outros exemplos e formas de tratamento e limpeza de dados.



### 3.2.2. Transformações

Em geral uma transformação nos dados envolve a aplicação de alguma fórmula matemática aos valores de um variável, buscando obter os dados em uma forma mais apropriada para a posterior modelagem, maximizando a informação, satisfazendo premissas de modelos ou simplesmente prevenindo erros.

Entre as transformações mais realizadas e importantes está a normalização ou padronização dos dados, feita com o objetivo de homogeneizar a variabilidade das variáveis de uma base de dados, criando um intervalo de amplitude similar onde todas as variáveis irão residir. Em geral a normalização é necessária no caso de variáveis com unidades diferentes ou dispersões muito heterogêneas. Entre as formas mais comuns de normalização estão:

- Normalização pelo desvio padrão: $y = \frac{x - \mu}{\sigma}$ *Equação 3.1*

- Normalização pela faixa de variação: $y = \frac{x - \min}{\max - \min}$ *Equação 3.2*

Nas Equações 3.1 e 3.2, *y* representa o novo valor normalizado; *x*, o valor atual; *μ* e *σ*, a média e o desvio padrão da variável; e *max* e *min*, os valores de máximo e mínimo, respectivamente.

Outra maneira muito utilizada para alteração dos dados é a codificação de variáveis categóricas em variáveis *dummy*, onde uma variável com *n* categorias é codificada em um vetor de tamanho *n* com um único valor não nulo, identificando a categoria da qual o registro fazia parte.

Além dessas transformações, a aplicação de logaritmos, da função inversa (1/*x*), a extração de diferenças temporais e outras também são bastante utilizadas na busca por uma representação mais adequada dos dados. Informação adicional sobre o assunto pode ser encontrada em [BALL99] e [PYLE99].

### 3.2.3. Oversampling

Muitos problemas que envolvem bases de dados tratam de variáveis categóricas altamente desequilibradas em termos da proporção de cada classe existente [BERR00] [MOZ00] [YAN01] [ARCH04] [AU03]. Por exemplo, uma base de telefonia celular pode possuir uma variável que denota se um cliente deixou ou não a empresa. Essa variável em geral possui algo em torno de 98% dos clientes como os que continuam na empresa e somente 2% dos clientes como os que terminaram sua relação com a operadora. O que acontece no momento da construção de qualquer modelo envolvendo uma variável deste tipo é que, dada essa distribuição desequilibrada entre as classes, o modelo terminará enxergando somente uma das classes, sendo incapaz de distinguir a classe de menor número de registros. Isso acontece porque o modelo reconhece que, se sua resposta for sempre dizer que todas as observações pertencem à classe com maior número de registros, ele acertará 98% dos padrões.

Para evitar esse problema e facilitar a distinção de classes, é realizado um procedimento conhecido como *oversampling*. Através do *oversampling* cria-se uma nova base de dados para a modelagem, selecionando-se aleatóriamente

um maior número de registros pertencentes à classe rara e um menor número de ocorrências da classe comum, desta forma ajustando a proporção entre as classes. No caso de variáveis binárias, por exemplo, em geral se deseja que a classe rara corresponda a algo entre 10% e 40% da base para a modelagem. Frequências entre 20% e 30% em geral dão bons resultados [BERR00].

Infelizmente, *oversampling* possui limitações. Dado que só existe um pequeno número de observações da classe rara na base de dados, não é possível criar uma base de qualquer tamanho para a análise, mesmo que a base de dados original seja imensa. Por exemplo, em uma base de 100.000 registros, se somente 2% pertencem a uma classe, isso significa que só estão disponíveis 2000 amostras desta certa classe. Sendo assim, é impossível construir uma base para modelagem com, por exemplo, 50.000 registros e uma freqüência maior do que 4% para a classe rara. É necessário, para atingir as proporções entre 10% e 40% mencionadas, que a base de dados seja bastante diminuída no que diz respeito às observações da classe comum.

O procedimento de *oversampling* descrito é bastante similar para variáveis com mais de duas classes.

## 3.3. Métodos de seleção de variáveis

A literatura possui um grande número de métodos de seleção de variáveis [BACK01] [BLUM97] [JANG93] [KWAK03] [YI97] [YAN01] [ZHEN04]. Essas várias técnicas dividem-se entre aquelas que dependem da modelagem completa de algum modelo para verificar a importância das entradas (*model based*), e aquelas totalmente independentes de modelos baseadas na execução de testes estatísticos entre os subconjuntos das variáveis de entrada e as saídas desejadas do modelo (*model free*).

O objetivo destas técnicas é otimizar a relação de informação entre as entradas e as saídas de algum modelo, reconhecendo quais entradas desempenham um papel importante na definição das saídas e eliminando entradas que porventura sejam irrelevantes ou contribuam para um melhor desempenho do modelo a ser utilizado. Desta forma existe uma redução da dimensão da base de dados devido ao corte de entradas de pouca importância, o que contribui para um melhor desempenho computacional. Além disso, ao verificar quais variáveis de entrada exercem maior influência na classificação ou previsão correta da variável de saída, também se obtém informação valiosa

sobre o problema estudado. Saber quais são as variáveis que influenciam o comportamento de um cliente, por exemplo, pode ser de grande ajuda para o departamento de marketing de uma empresa, na criação de uma campanha ou de alguma ação de fidelização.

Nesta seção serão apresentados alguns dos métodos de seleção de variáveis existentes, métodos esses que serão utilizados nos capítulos posteriores. Uma detalhada revisão sobre estes e muitos outros métodos pode ser encontrada em [CONT02].

### 3.3.1. Método do Estimador de Mínimo Quadrado (LSE)

Seja um sistema de *n* entradas e uma saída; o método LSE calcula a importância da i-ésima variável de entrada $x_i$ estimando o i-ésimo parâmetro $b_i$ da função *F* que descreve a variação da variável de saída $\Delta y$ em relação à variação de cada i–ésima variável de entrada $\Delta x_i$ sobre o conjunto completo de dados. A função *F* é dada pela Equação 3.3:

$$F = \Delta y = b_1[\Delta x_1] + b_2[\Delta x_2] + b_3[\Delta x_3] + ... + b_n[\Delta x_n] \qquad \textit{Equação 3.3}$$

Os componentes do vetor $\Delta y$ são obtidos subtraindo-se os valores da variável de saída nos padrões da base de dados em combinações de duas a duas, e os componentes do vetor $\Delta x_i$ são obtidos subtraindo-se os valores da variável de entrada $x_i$ nos padrões da base de dados em combinações de duas a duas.

Da Equação 3.3 pode-se dizer que cada parâmetro $b_i$ representa a importância da i-ésima variável de entrada com respeito à saída no sentido estatístico. O cálculo dos parâmetros $b_i$ é feito mediante um algoritmo de mínimos quadrados [CHUN00].

Uma descrição detalhada deste método encontra-se em [CONT02].

### 3.3.2. Método da Efetividade de uma Entrada Singular (SIE)

O objetivo deste método é calcular o *ranking* das entradas mediante a definição do SIE, que é o grau de efetividade de cada entrada na saída [CAO97] [CAO96-2].

Considerando o sistema linear $y = Gu$ de $m$ saídas e $r$ entradas, neste método a importância de cada variável de entrada é obtida expressando-se o vetor de entradas $u$ do sistema como a soma de duas projeções ortogonais $u_n$ e $u_c$, uma pertencente ao espaço nulo da matriz de transferência G do sistema e a outra pertencente ao complemento ortogonal do espaço nulo de G respetivamente.

Portanto, a efetividade de cada variável de entrada $x_i$ é calculada como a norma da projeção da entrada no canal $i$ sobre o complemento ortogonal do espaço nulo de G, já que a norma da projeção da entrada no canal i sobre o espaço nulo de G não tem efeito na variável de saída.

Maiores detalhes sobre este método podem ser encontrados em [CAO97] [CAO96-2] [CONT02].

### 3.3.3. Método baseado no modelo ANFIS

O método de seleção de variáveis baseado no modelo ANFIS [JANG93] [JANG94] realiza uma seleção de entradas segundo sua importância para o melhor desempenho de um modelo ANFIS, como o da Figura 3.1. A configuração do modelo ANFIS utilizada possui duas entradas divididas em 4 conjuntos fuzzy cada, fazendo uso de um particionamento *fuzzy grid* fixo. Assim, o espaço de entrada é dividido em 16 partições, segundo a Figura 3.2.

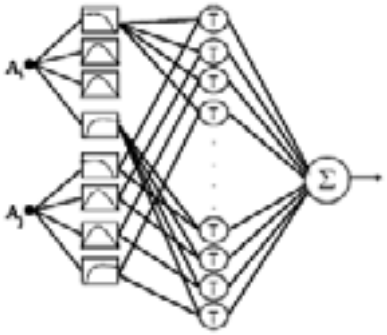

Figura 3.1 – Sistema ANFIS para Seleção de Variáveis

Utilizando como entradas atributos da base de dados escolhidos dois a dois, treina-se o sistema, durante um certo número de ciclos especificado e, em seguida, calcula-se o erro de classificação para esses dois atributos. Posteriormente, escolhe-se um novo par de atributos, treina-se o sistema pelo mesmo número de ciclos, até que todas as configurações de pares de entrada tenham sido testadas.

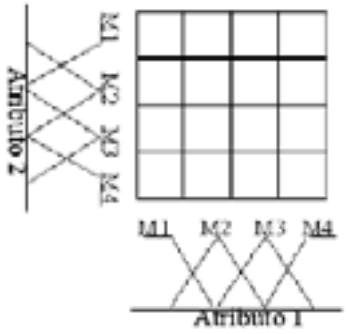


Figura 3.2 – Particionamento do sistema ANFIS utilizado

Posteriormente, as duplas de entradas são listadas em ordem crescente do valor do erro, selecionando-se as entradas de menor erro. Tais entradas são, supostamente, as que melhor contribuem com informação relevante para a caracterização das saídas.

## 3.4. Métodos de Classificação de Padrões

Um problema de classificação de padrões se resume a determinar, da forma mais correta possível, a classe de saída à qual o padrão de entrada (registro) pertence. O desenvolvimento de um modelo de classificação caracteriza-se pela definição do grupo de classes, das varáveis relevantes e de um conjunto de treinamento consistindo em exemplos (padrões) pré-classificados.

Grande parte dos problemas de classificação de padrões de interesse real tem duas fases: uma etapa *in sample* ou de treinamento (aprendizado), executada a partir do banco de dados existente, e uma etapa *out of sample* ou de generalização, onde são apresentados dados que não foram utilizados no treinamento. O que se deseja, fundamentalmente, é extrair do banco de dados as características que definem o mapeamento entre entrada e saída, para posteriormente utilizá-las na tomada de decisões em dados novos, ainda não avaliados pelo sistema.

Existem inúmeros métodos e técnicas [ARCH04] [AU03] [BURG98] [BERR00] [BISH96] [BOZD03] [CHUN00] [DUDA00] [FAYY96] [GONÇ01] [LOP99-1] [LOP99-2] [KLOS02] [MICH96] [MOZE00] [QUIN87] [SOUZ99] [ZHEN04] para utilização em aplicações de classificação de padrões, entre eles:

métodos estatísticos, algoritmos genéticos, árvores de decisão, redes neurais, sistemas híbridos neuro-fuzzy e máquinas de vetor de suporte.

Cada técnica possui características próprias, apresentando vantagens e desvantagens. Nas próximas seções serão apresentadas, de forma sucinta, cada uma dessas técnicas, as quais serão usadas no problema específico do *churn*. Todas as técnicas detalhadas nesta seção fizeram parte dos testes e da modelagem de *churn* através do sistema de retenção proposto. Essa escolha é feita, principalmente, porque tais técnicas apresentam grandes possibilidades de gerar bons resultados na classificação de dados desta natureza, como constata a literatura sobre modelagem de *churn* [ARCH04] [AU03] [BERR00] [MOZE00] [YAN04] [ZHEN04]. Alguns dos modelos testados ainda apresentam a vantagem de gerar regras interpretáveis (árvores de decisão, sistemas neuro-fuzzy e algoritmos genéticos) que podem vir a ser úteis para a compreensão do problema.



### 3.4.1. Métodos Estatísticos

Existem diversos métodos estatísticos para a classificação de padrões, alguns clássicos e outros mais recentes. Todos assumem a existência de uma variável (atributo) resposta, $y$, e uma coleção de variáveis de entrada, $x = (x_1, x_2, ..., x_j)$, além da disponibilidade de dados para treinamento. A meta é encontrar um modelo para prever $\hat{y} = f(x)$ que funcione bem quando aplicado a um novo dado. Um resumo dos principais modelos aplicados a todas as técnicas de indução pode ser encontrado em [FRIE95].

Entre as metodologias mais usadas estão os classificadores bayesianos [DUDA00]. O princípio básico de classificadores bayesianos está fundamentado na teoria da probabilidade bayesiana [DUDA00]. Os Classificadores Bayesianos são capazes de encontrar regras que respondem a perguntas do tipo;

- Qual a probabilidade de se jogar tênis dado que o dia está ensolarado, com temperatura quente, umidade alta e vento fraco? Em termos probabilísticos essa pergunta equivale a P(JOGAR TÊNIS = Sim | [Ensolarado, Quente, Alta, Fraco])
- Qual a probabilidade de NÃO se jogar tênis dado que o dia está ensolarado, com temperatura quente, umidade alta e vento fraco? Em termos probabilísticos essa pergunta equivale a P(JOGAR TÊNIS = Não | [Ensolarado, Quente, Alta, Fraco])

Esse tipo de método será utilizado por sua facilidade de interpretação e simplicidade, servindo de base de comparação para métodos mais complexos. Maiores detalhes sobre esses métodos são apresentados em [DUDA00].

### 3.4.2. Redes Neurais

Uma Rede Neural Artificial (RNA) é uma técnica computacional que constrói um modelo matemático, emulado por computador, de um sistema neural biológico simplificado, com capacidade de aprendizado, generalização, associação e abstração. As RNAs tentam aprender padrões diretamente dos dados através de um processo de repetidas apresentações dos dados à rede, ou seja, por experiência. Dessa forma, uma Rede Neural procura por relacionamentos, constrói modelos automaticamente, e os corrige de modo a diminuir seu próprio erro.

Semelhante ao sistema biológico, uma RNA possui, simplificadamente, um sistema de neurônios, ou nós, e conexões ponderadas (equivalentes às sinapses). Numa RNA os nós são arrumados em camadas, com conexões entre elas. A Figura 3.3 representa conceitualmente a arquitetura de uma RNA simples. Os círculos representam os nós e as linhas representam os pesos das conexões. Por convenção, a camada que recebe os dados é chamada camada de entrada e a camada que mostra o resultado é chamada camada de saída. A camada intermediária, onde se localiza o processamento interno, é tradicionalmente chamada de camada escondida. Uma RNA pode conter uma ou várias camadas escondidas, de acordo com a complexidade do problema [BISH96], [HAYK99].

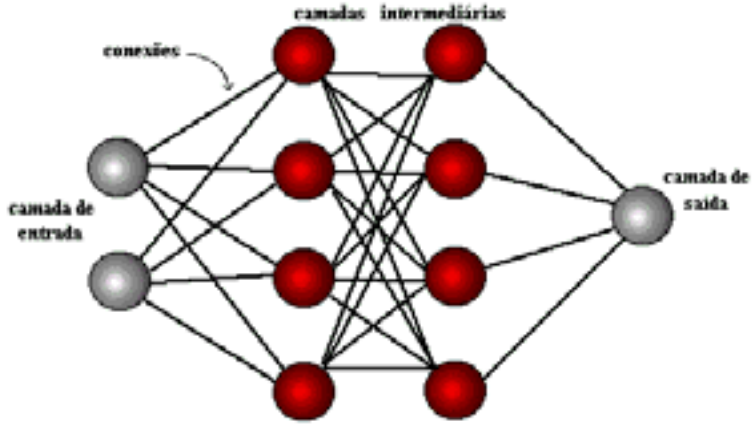


Figura 3.3 – Estrutura em camadas de uma Rede Neural

Para entender como uma RNA aprende é necessário saber como os pesos da rede afetam sua saída. O aprendizado de uma RNA envolve o ajuste dos pesos. A Figura 3.4 mostra o esquema de um neurônio artificial criado a partir do modelo simplificado do neurônio biológico. O neurônio artificial possui várias entradas, que podem ser estímulos do sistema ou saídas de outros neurônios.

O neurônio artificial é dividido em 2 seções funcionais. A primeira seção combina todas as entradas que alimenta o neurônio. Essa etapa indica como as entradas serão computadas (regra de propagação). A segunda seção recebe esse valor e faz um cálculo determinando o grau de importância da soma ponderada utilizando uma função de transferência, ou função de ativação. Essa função determina com que grau a soma causará uma excitação ou inibição do neurônio. Os tipos mais comuns de funções de ativação são sigmóide e tangente hiperbólica, pois fornecem a característica de não linearidade para uma RNA.

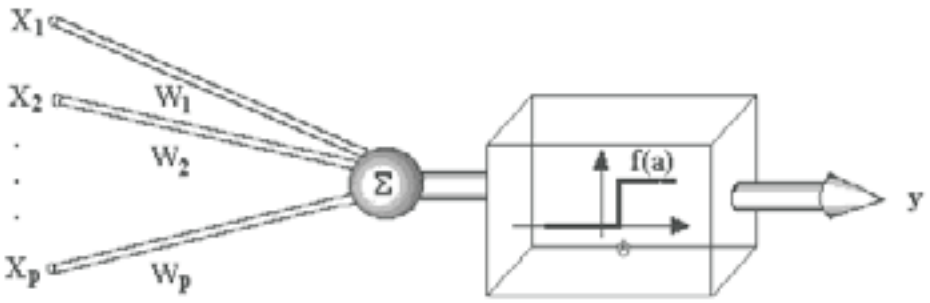


Figura 3.4 – Neurônio Artificial

Uma RNA ajusta seus pesos na fase de treinamento. É fornecido um dado de observação, o qual é processado, e uma resposta será produzida. O resultado fornecido é comparado com uma saída desejada. Se a rede acerta essa saída, então ela não faz nada; entretanto se o resultado não está correto, ocorre um ajuste dos pesos de modo que o erro seja minimizado.

As topologias mais comuns de RNAs são as de múltiplas camadas *feed-forward* e as redes recorrentes. O aprendizado de uma RNA pode ser dividido em 3 grupos: sem treinamento – os valores dos pesos sinápticos são estabelecidos a priori, ou seja, ajustados em um único passo, por exemplo, Redes de Hopfield [DAYH90]; treinamento supervisionado – a rede é treinada através do fornecimento dos valores de entrada e dos seus respectivos valores

de saída desejados; e treinamento não-supervisionado – o sistema extrai as características dos dados fornecidos, agrupando-os em classes (*clusters*).

Para mineração de dados, Redes Neurais são aplicadas em tarefas de classificação, *clustering*, aproximação de funções, previsão e verificação de tendências.

De um modo geral, a arquitetura de uma RNA recebe um padrão (atributos preditivos, por exemplo um registro de uma base de dados) como entrada através da primeira camada da rede (camada de entrada).

No caso de aprendizado não supervisionado, não existe uma variável alvo que possa ser utilizada para corrigir os pesos da rede. Esse tipo de aprendizado aplica-se em tarefas de *clustering*. As redes auto-organizáveis (por exemplo, Kohonen), baseadas em aprendizado competitivo, destacam-se como um bom algoritmo [FU94].

Em algoritmos supervisionados, as variáveis alvo são modeladas pela camada de saída da rede. Deste modo o algoritmo pode estimar o quanto a saída desejada está distante da saída real. O algoritmo mais comum em RNAs com aprendizado supervisionado é o *backpropagation*. Seu objetivo é minimizar a função de erro entre a saída real da rede e a saída desejada utilizando o método do gradiente descendente [HAYK99]. *Backpropagation* é utilizado para classificar, aproximar funções, prever e verificar tendências [BISH96].

Para a tarefa de classificação também são utilizadas as redes neurais probabilísticas, baseadas em classificadores bayesianos, e as redes RBF (*Radial-Basis Function*), baseadas em funções *gaussianas*. Esses algoritmos geram curvas de densidade de probabilidade, fornecendo resultados com bases estatísticas. Esses resultados indicam o grau de evidência sobre o qual se baseia a decisão. Entretanto, essa metodologia só funciona bem se existir um número suficiente de exemplos na base de dados. A Figura 3.5 ilustra a arquitetura desses algoritmos. A principal diferença está na interpretação do resultado, pois cada nó da camada de saída gera um valor que indica a probabilidade do padrão inserido na camada de entrada pertencer a uma determinada classe.

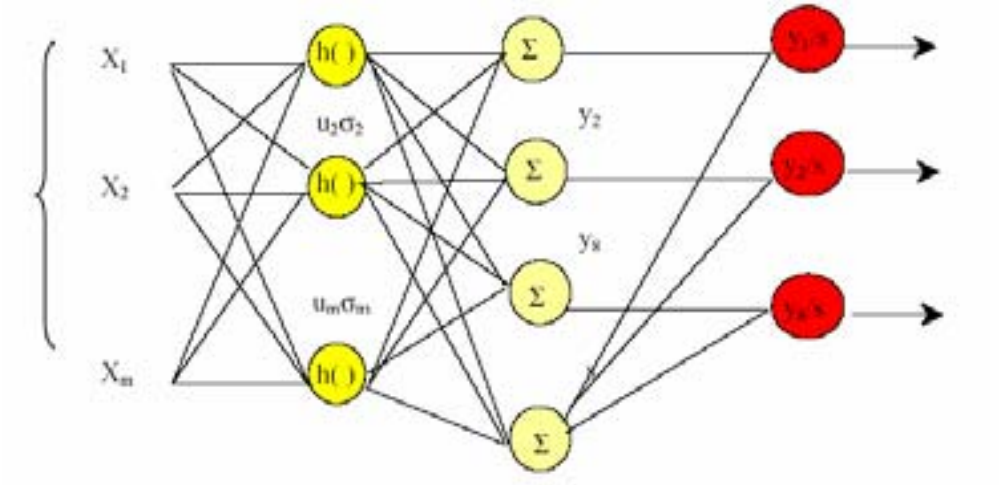


Figura 3.5 – Arquitetura de uma Rede Neural com bases estatísticas

Nesta topologia, RBF, (Figura 3.5) o número de neurônios da camada escondida é exatamente igual ao número de padrões apresentados para treinamento, onde: $X_i$ é um atributo preditivo do padrão apresentado, *h( )* é a função de ativação, $u_i$ é o centro da Gaussiana e σ é o desvio padrão.

As redes de Hopfield também são utilizadas para classificação. Nesse algoritmo as classes são consideradas estados. Fornecendo-se um padrão como entrada para a rede, os pesos são atualizados de modo a ocorrer uma convergência para um estado estável, que será sua classe.

### 3.4.3. Árvores de Decisão

Uma árvore de decisão é uma estrutura ramificada onde cada nó interno é rotulado com um dos atributos preditivos. Cada seta que sai do nó interno é rotulado com valores dos atributos de cada nó, e todo nó folha é rotulado com uma classe (valor da variável alvo). A Figura 3.6 apresenta a estrutura simplificada de uma árvore de decisão, com País e Idade sendo atributos preditivos e uma variável binária (S ou N) sendo o alvo.

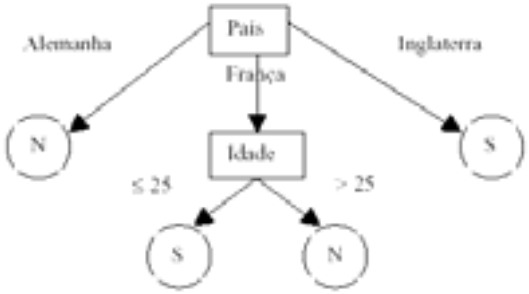


Figura 3.6 – Estrutura de uma árvore de decisão

Uma árvore de decisão é geralmente construída de maneira *top-down*, utilizando um algoritmo baseado na aproximação “dividir para conquistar” [QUIN93]. Inicialmente todas os padrões que estão sendo minerados são associados ao nó raiz da árvore. Então o algoritmo seleciona uma partição de atributos e divide o conjunto de padrões no nó raiz de acordo com o valor do atributo selecionado. O objetivo desse processo é separar as classes para que padõres de classes distintas tendam a ser associados a diferentes partições. Esse processo é recursivamente aplicado a subconjuntos de padrões criados pelas partições, produzindo subconjuntos de dados cada vez menores, até que um critério de parada seja satisfeito. Discussões sobre algoritmos de poda de uma árvore de decisão podem ser encontrados nas próximas seções e em [AGRA92], [ESPO95], através dos quais busca-se minimizar o tamanho da árvore, sem prejudicar a qualidade da solução.

As principais vantagens de algoritmos baseados em árvores de decisão são sua eficiência computacional e simplicidade. Devido ao uso da aproximação “dividir para conquistar”, entretanto, essa aproximação também possui desvantagem. Por exemplo, uma condição envolvendo um atributo que será incluído em todas as regras descobertas. Essa situação possivelmente produz regras com informações irrelevantes, além de desperdício de processamento. A seguir são apresentados alguns dos principais algoritmos baseados em árvores de decisão.

O algoritmo ID3, desenvolvido em 1983 por Ross Quinlan, da Universidade de Sydney, Austrália, deu origem às árvores de decisão. O ID3 e suas evoluções (ID4, ID6, C4.5, C5.0/See) [QUIN87] [QUIN93] converteram-se numa das metodologias mais comuns nas tarefas de descoberta de conhecimento. Existem também abordagens propostas para utilização de árvores de decisão fuzzy [JANI96], [JANI99] em tarefas de classificação e descoberta de conhecimento.

- **ID3 e C4.5** [QUIN87] [QUIN93]: são os algoritmos mais populares para tarefas de classificação. O algoritmo ID3 gera uma árvore de decisão a partir de informações contidas na base de dados, começando pelo atributo mais relevante do ponto de vista da informação nele contida, e continuando com os atributos seguintes segundo a avaliação da entropia de cada atributo. O algoritmo ID3 não trabalha com atributos de tipo contínuo, mas sucessivas atualizações do algoritmo adaptaram-se para lidar com este tipo de atributos, como é o caso do algoritmo C4.5. Este introduz algumas modificações para lidar com atributos contínuos e valores desconhecidos.
- **CART** [BREI84]: O algoritmo CART (Classification And Regression Trees), gera uma árvore de decisão realizando de forma recursiva subdivisões binárias na base de dados de treinamento. As subdivisões podem ser feitas sobre variáveis individuais (classificação) ou sobre uma combinação destas variáveis (regressão). A construção de uma árvore CART é realizada a partir dos exemplos contidos na base de treinamento. O algoritmo CART pode lidar com bases de dados incompletas, diferentes formatos numéricos (inteiro, flutuante,...) e atributos contínuos ou divididos em classes (a,b,c,...). O algoritmo CART gera uma estrutura de árvore binária da qual podem ser extraídas regras que são facilmente interpretáveis.
- **FID3.1** [JANI99]: O algoritmo FID3.1 constrói árvores de decisão fuzzy [JANI96]. As árvores de decisão fuzzy estão ultimamente sob intensa pesquisa devido aos bons resultados obtidos nas tarefas de classificação de padrões.

Em modelos de Árvore de Decisão, a estrutura da árvore é parametrizada através de definições como: o mínimo número de observações necessário para um *split* (partição da árvore) ser realizado; o número mínimo de observações em uma folha para que ela possa existir; e a profundidade máxima da árvore.

### 3.4.4.
### Sistemas Neuro-Fuzzy

Sistemas neuro-fuzzy (SNF) são sistemas híbridos que combinam as vantagens das redes neurais, no que se refere ao aprendizado, com o poder de interpretação lingüístico dos sistemas de inferência fuzzy.

Os sistemas neuro-fuzzy realizam, internamente, um mapeamento entre regiões do espaço de entrada em regiões fuzzy do espaço de saída, através de regras fuzzy do sistema. As regiões fuzzy do espaço de E/S são determinadas no processo de identificação da estrutura. Nesse processo, os espaços de entrada e/ou saída são divididos segundo um determinado método de partição. As variáveis de entrada e saída dos sistemas neuro-fuzzy são divididas em vários termos lingüísticos (por exemplo: baixo, alto) que são utilizados pelas regras fuzzy.

Em particular, entre os sistemas neuro-fuzzy existentes na literatura [JANG93] [GONÇ01] [KRUS95] [VUOR94] [SOUZ99], é de interesse para esta dissertação o modelo de sistema neuro-fuzzy hierárquico (NFHB) [SOUZ99], que utiliza o particionamento recursivo BSP (*Binary Space Partitioning*) [CHIN89] [CHRY92] para resolver duas grandes limitações dos sistemas neuro-fuzzy: a impraticabilidade de lidar com um grande número de entradas; e a forma limitada (em alguns casos inexistente) destes criarem sua própria estrutura e regras.

A primeira limitação ocorre em função da chamada explosão combinatorial das regras. Por exemplo, suponha-se que um determinado sistema neuro-fuzzy tenha cinco variáveis de entrada e cada uma delas tenha seu universo de discurso subdividido em quatro conjuntos fuzzy. Com esse sistema, pode-se chegar a um total de 1024 ($4^5$) regras. Suponha agora que se tenham 20 entradas. Usando-se a mesma divisão nos universos de discurso para cada variável de entrada, chega-se a um total impraticável de 1.099.511.627.776 ($4^{20}$) regras.

A segunda limitação está presente nos sistemas neuro-fuzzy atuais porque eles exibem (quando tal) uma capacidade limitada de criar sua estrutura. Alguns têm estrutura fixa, arbitrada a priori e não se permite que ela varie. Outros sistemas neuro-fuzzy, mais flexíveis, têm alguma capacidade de alterar sua estrutura, permitindo que se altere o número de divisões no universo de discurso de algumas de suas variáveis de entrada e, conseqüentemente, o número de regras. Entretanto, esta habilidade é reduzida e ainda mantém o primeiro tipo de limitação.

Uma solução para essas limitações é o uso de formas de particionamento recursivo, que permitem a criação de sistemas neuro-fuzzy com uma capacidade ilimitada de crescimento em sua estrutura.

A forma recursiva de particionamento BSP é utilizada [SOUZ99] para a criação de um modelo com hierarquia na estrutura e, por isso, hierarquia nas regras, mantendo ao mesmo tempo a interpretabilidade no mapeamento do espaço E/S. Esta forma de particionar o espaço é dita recursiva porque emprega um processo recursivo em sua geração. Maiores detalhes sobre o modelo podem ser encontrados em [SOUZ99] [GONÇ01] [CONT02].

### 3.4.5. Máquinas de Vetor de Suporte

Em um problema típico de classificação por máquinas de vetor de suporte (*Support Vector Machines*, SVM) [ARCH04] [BURG98] [DUDA00] [ZHEN04], existem $n$ observações $(x_1; y_1)$, ..., $(x_n; y_n)$ com $x_i \in R^n$ e $y_i \in \{-1, 1\}$ para todo $i$. O objetivo é então encontrar o hiperplano $x_i.w + b = 0$ que separe os pontos das classes positiva e negativa com uma margem máxima, penalizando linearmente pontos dentro da margem através de um parâmetro de regularização selecionado pelo usuário, $C > 0$. A Figura 3.5 ilustra o caso linearmente separável, com w sendo a normal ao plano e $|b|/||w||$ sendo a distância do plano à origem.

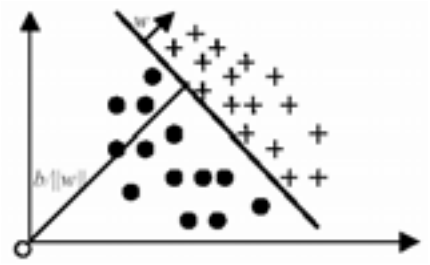

Figura 3.5 – Caso Linearmente Separável

A não-linearidade de uma SVM está presente através da escolha de uma função de *kernel* que pode ser utilizada como parâmetro do algoritmo de treinamento. A função dos kernel é mapear o espaço de entrada em um espaço de dimensão superior, onde o problema torna-se linearmente separável mesmo não o sendo no espaço original de menor dimensão. Entre os *kernel* mais utilizados [ARCH04] [BURG98] estão os polinômios de grau p, $K(x,y) = ((x.y) + 1)^p$, e o gaussiano, $K(x,y) = \exp[-|x-y|/(2\sigma^2)]$. Uma vez escolhido o *kernel* e seus

parâmetros, o único parâmetro que resta para ser definido é o parâmetro de penalização *C*.

Máquinas de vetor de suporte trazem uma nova opção para o problema de reconhecimento de padrões com claras conexões na teoria estatística de aprendizado. Elas diferem radicalmente de outros métodos como, por exemplo, redes neurais: o treinamento de uma SVM sempre encontra um mínimo global e a sua simples interpretação geométrica fornece muita margem para investigações mais profundas.

### 3.4.6. Algoritmos Genéticos

Os algoritmos genéticos têm sido empregados em mineração de dados para tarefas de classificação de padrões, descrição de registros, e seleção de atributos da base de dados [AU03] [KIRA92] [KOLL96] [LOPE00].

Na classificação de padrões, os modelos de algoritmos genéticos geram regras que exprimem a realidade do domínio da aplicação. Essas regras são de fácil interpretação, o que incentiva o uso dessa técnica.

Para exemplificar como os algoritmos genéticos trabalham com tarefas de classificação, será apresentado um método implementado em uma ferramenta nomeada Rule-Evolver [LOPE99].

**Rule-Evolver:** Baseado na teoria de algoritmos genéticos [MICH96], procura regras SE-ENTÃO para melhor descrever as classes de um problema. A parte SE de cada regra dita qual intervalo de cada atributo da base de dados melhor caracteriza a classe descrita após o ENTÃO. Regras compostas por diferentes atributos podem ser agregadas com a utilização do conectivo E. A Figura 3.6 detalha a representação do cromossomo no Rule Evolver:

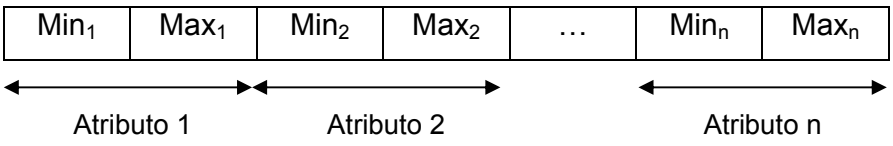


Figura 3.6 – Rule Evolver: Representação do Cromossoma

Como ilustrado na Figura 3.6, cada gene possui dois valores, correspondendo a um limite superior e um limite inferior. Desta forma, as regras serão evoluídas de forma a otimizar os valores destes máximos e mínimos de cada atributo, de maneira que os membros da classe prevista se encaixem entre

esses limites. Com essa definição, fica claro que nenhuma regra sozinha conseguiria classificar um banco de dados inteiro. Por isso, o método consiste na geração de um conjunto de regras capazes de classificar corretamente os padrões apresentados em cada uma das classes existentes.

A avaliação da adequabilidade de cada cromossomo em comparação aos demais é feita através da maximização de uma função de avaliação. Mais de 11 funções de avaliação estão disponíveis no Rule-Evolver. Uma delas, por exemplo, é a função chamada de Rule Interest, que se baseia em uma relação matemática entre a base de dados completa, o grupo de observações que satisfaz a condição de uma dada regra e o número de observações que satisfaz a previsão desta mesma regra. Muitos outros parâmetros podem ser variados no Rule-Evolver, entre eles diversos tipos de mutações e *crossovers*.

## 3.5. Resumo

Neste capítulo foram apresentados de forma resumida os conceitos de várias técnicas e modelos estatísticos e de inteligência computacional aplicados em análise de grandes bases de dados.

O capítulo seguinte ambienta o trabalho na indústria de telefonia celular, detalhando o problema do *churn*.