Universidade
Estadual de Londrina

ANDRÉ RICARDO GONÇALVES

FUNDAMENTOS E APLICAÇÕES DE TÉCNICAS DE APRENDIZAGEM DE MÁQUINA

Londrina

2008

ANDRÉ RICARDO GONÇALVES

FUNDAMENTOS E APLICAÇÕES DE TÉCNICAS DE APRENDIZAGEM DE MÁQUINA

Trabalho apresentado ao Departamento de Computação da Universidade Estadual de Londrina, como parte do requisito para obtenção do título de Bacharel em Ciência da Computação, sob orientação da Profª. Drª. Maria Angélica de Oliveira Camargo Brunetto.

Londrina
2008

ANDRÉ RICARDO GONÇALVES

FUNDAMENTOS E APLICAÇÕES DE TÉCNICAS DE APRENDIZAGEM DE MÁQUINA

COMISSÃO EXAMINADORA

Profª. Drª. Maria Angélica de Oliveira Camargo Brunetto
Universidade Estadual de Londrina

Profº. Drº.
Universidade Estadual de Londrina

Profº. Drº.
Universidade Estadual de Londrina

Londrina 05 de Novembro de 2008

Aos meus pais ...

Vera Lúcia Borges,

Lourival dos Santos Gonçalves.

Agradecimentos

Primeiramente agradeço a Deus por me dar a vida e por estar nela.

Aos meus pais, Vera Lúcia e Lourival Gonçalves, pelo total incentivo, servindo como uma base firme onde eu posso me sustentar. Agradeço a toda minha família, meus irmãos Evandro Luis e Lourival Jr., que também tiveram suas parcelas de contribuição.

Agradeço a minha namorada, Vânia, por sua companhia e pelo carinho tido por mim. Mesmo quando um pouco ausente, sempre compreensível e me apoiando pra prosseguir.

Agradeço a minha orientadora, professora Drª Maria Angélica de Oliveira Camargo Brunetto, por me dar a oportunidade de trabalhar no Laboratório Protem e participar dos projetos de pesquisa. Por sempre ter um tempo, no meio de suas inúmeras tarefas, para discutir sobre os projetos e trabalhos, contribuindo para o meu aprendizado.

Agradeço ao professor Pedro Paulo, por sua ajuda em várias partes deste trabalho.

Agradeço aos professores do Departamento de Computação, pelos conhecimentos transmitidos, pela paciência e amizade.

Aos vários amigos conquistados nestes quatro anos, tanto na UEL, quando em Londrina. Pessoas de diferentes lugares e pensamentos que contribuíram para o meu crescimento como pessoa.

"Nenhum trabalho de qualidade pode ser feito
sem concentração e auto-sacrifício, esforço e dúvida."
(Max Beerbohm)

GONÇALVES, André Ricardo. **Fundamentos e aplicações de técnicas de aprendizagem de máquina**. 2008. 133f. Trabalho de Conclusão de Curso (Ciência da Computação) - Universidade Estadual de Londrina, Londrina. 2008.

Resumo

As técnicas de aprendizagem de máquina vêm despontando como importantes ferramentas para resolução de problemas complexos, os quais não são solucionáveis por métodos exatos, devido a dinâmica complexa do problema, a alta dimensionalidade dos dados, a interdependência com outros problemas. Uma subárea da inteligência artificial, a aprendizagem de máquina tem por objetivo a extração e armazenamento do conhecimento sobre um ambiente, por meio de heurísticas de aprendizagem, e com isso ser capaz de predizer informações e/ou identificar regras e padrões do ambiente inserido. As heurísticas de aprendizagem são chamadas de técnicas de aprendizagem de máquina, que são encarregadas de retirar informações de um conjunto de dados. O objetivo deste trabalho é investigar as principais técnicas de aprendizagem existentes na literatura, analisando seu funcionamento e suas aplicações. A fim de verificar suas reais habilidades e limites, serão modelados e implementados *softwares* capazes de simular estas diferentes metodologias.

Palavras-chaves: Aprendizagem de Máquina, Computação Evolucionária, Aprendizagem Estatística, Modelo Probabilístico, Redes Neurais.

GONÇALVES, André Ricardo. **Fundamentals and aplications of machine learning techniques**. 2008. 133f. Senior Research Project (Computer Science) - State University of Londrina, Londrina. 2008.

Abstract

Techniques of machine learning to come own as important tools for solving problems complexes, which are not remedied by exact methods, because the complex dynamics of the problem, the high dimensionality of the data, the interdependence with other problems. A sub-field of artificial intelligence, the machine learning has the goal of extraction and storage of knowledge about an environment, through Learning from heuristic, and thereby be able to predict information and/or identify rules and patterns of environment inserted. The heuristics of learning are called machine-learning techniques, which are responsible for removing information from a data set. The purpose of this study is to investigate the main techniques for learning in the literature, examining its operation and its applications. In order to verify their real abilities and limitations, will be shaped and implemented software capable of simulating these different methodologies.

Key-words: Machine Learning, Evolutionary Computing, Statistic Learning, Probabilistic Models, Neural Networks.

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de siglas e abreviaturas

IA Inteligência Artificial

AM Aprendizagem de Máquina

RNA Redes Neurais Artificiais

MB Máquina de Boltzman

MLP *Multilayer Perceptron*

SOM *Self-Organizing Maps*

RBF *Radial Basis Function*

CE Computação Evolucionária

AG Algoritmo Genético

ACO *Ant Colony Optimization*

AS *Ant System*

ACS *Ant Colony System*

PSO *Particle Swarm Optimization*

RB Rede Bayesiana

TPC Tabela de Probabilidade Condicional

SVM *Support Vector Machine*

TAE Teoria da Aprendizagem Estatística

VC Vapnik-Chervonenkis

DAG *Directed Acyclic Graph*

UCT Um-contra-Todos

TCT Todos-contra-Todos

Sumário

Introdução

A capacidade intelectual humana tem sido um tema discutido há mais de 2000 anos, inicialmente com os filósofos gregos Platão, Aristóteles e Sócrates. Mesmo com um grande avanço tanto no campo psico-filosófico quanto na medicina, ainda há muita discussão sobre alguns processos mentais dos seres humanos. Dentre este processos destaca-se a capacidade humana em aprender e armazenar conhecimento.

A aprendizagem é definida como a forma que indivíduos adquirem conhecimento, desenvolvem habilidades e alteram seu comportamento. Algumas teorias de aprendizagem, de cunho pedagógico, foram propostas por vários pesquisadores, sendo a teoria de Jean Piaget uma das mais utilizadas atualmente.

Piaget afirmava que basicamente a aprendizagem constitui de três conceitos fundamentais: *assimilação*, *acomodação* levando a um estado de *equilibração*. Onde o indivíduo, a partir de experiências anteriores, realiza uma assimilação com a situação atual (desconhecida), modificando os esquemas mentais em função das experiências e das relações com o meio, levando a uma auto-regulação interna do organismo, ou seja, o total conhecimento do domínio (equilibração).

Modelos computacionais foram propostos com o intuito de desenvolver uma máquina inteligente, capaz de aprender, armazenar o que aprendeu e posteriormente aplicar o conhecimento obtido em situações desconhecidas. Os primeiros estudos sobre a criação de uma máquina pensante surgiram nos anos 40, mais especificamente no trabalho de (MCCULLOCH; PITTS, 1943), o qual propunha a estrutura de um neurônio artificial. A partir disto vários estudos sobre aprendizagem e representação do conhecimento foram desenvolvidos.

Vários conceitos definidos nas teorias e paradigmas de aprendizagem, principalmente o conceito de aprendizagem utilizando-se de observações anteriores desenvolvidos por Piaget, foram pontos chave para elaboração de teorias e modelos computacionais inteligentes, originando então uma área de estudo conhecida como *aprendizagem de máquina*.

Aprendizagem de Máquina estuda como construir programas de computador que podem aprender ou melhorarem seu desempenho em alguma tarefa por meio de alguma experiência (SOUTO et al., 2003).

Em aprendizagem de máquina o conhecimento é armazenado de diversas maneiras, tais como na forma de peso nas conexões entre neurônios; posição física em um determinado ambiente; em tabelas de probabilidades, além de outras. Já o processo de aprendizagem ou extração do conhecimento é realizado por meio de cálculos estatísticos, cálculos probabilísticos, comparações por similaridade, entre outros.

Este trabalho tem como objetivo apresentar um conteúdo introdutório dos diversos paradigmas de aprendizagem de máquina, e uma descrição mais detalhada de algumas das principais técnicas para representação e extração do conhecimento utilizadas atualmente, identificando as vantagens e fragilidades de cada uma e suas aplicações em problemas reais. As técnicas apresentadas são: Redes Neurais Artificiais, Algoritmo Genético, Inteligência de Enxames, Redes *Bayesianas* e Máquina de Vetor Suporte.

O trabalho é estruturado como segue: no capítulo 1 são apresentados os paradigmas de aprendizagem de máquina; no capítulo 2 os conceitos de Redes Neurais Artificiais são abordados; o capítulo 3 descreve a técnica baseada na teoria Darwiniana da evolução das espécies, o Algoritmo Genético; já no capítulo 4 são discutidas as técnicas que utilizam-se da inteligência coletiva, denominada de Inteligência de Enxames, discutindo as técnicas de Otimização por Colônia de Formigas e Otimização por Enxame de Partículas; o capítulo 5 apresenta a técnica que usa teoria das probabilidades e teoria dos grafos, as Redes *Bayesianas*; o capítulo 6 discute a teoria da aprendizagem estatística para apresentar as Máquinas de Vetor Suporte; no capítulo 7 são realizados dois estudos de casos, sobre problemas da área da saúde, para demonstrar o poder destas técnicas; e por fim as conclusões deste trabalho são realizadas.

1 Aprendizagem de Máquina

Aprendizagem de Máquina (AM) é uma subárea da inteligência artificial, que tem por objetivo desenvolver sistemas capazes de "aprender". Por aprender entende-se a obtenção de conhecimento, extração de regras que regem um domínio. Diversos métodos foram desenvolvidos para extrair conhecimento, utilizando diversos paradigmas distintos, os quais serão apresentados neste capítulo.

Para que fosse possível construir esses programas "inteligentes", diversos processos do mundo real foram fontes de inspiração, como o funcionamento do cérebro humano, a teoria da evolução Darwiniana e genética, o comportamento de agentes simples como colônias de formigas, além de teoremas e aprendizagens estatísticas.

Os algoritmos de aprendizagem de máquina vêm ganhando reconhecimento, devido aos resultados atrativos que vêm sendo obtidos na aplicação em diversas áreas como: reconhecimento de padrões, processamento de sinais, diagnósticos médicos, reconhecimento de fala e escrita, detecção de fraude em cartão de crédito (MITCHELL, 1997) e mais recentemente em bioinformática, motores de busca, entre outros.

1.1 Tipos de Aprendizagem

Basicamente existem dois tipos aprendizagem: uma utilizando conceitos de lógica dedutiva, denominada de *aprendizagem dedutiva*; e outra que extrai conhecimento a partir de uma base de exemplos observados previamente, denominado de *aprendizagem indutiva*.

1.1.1 Aprendizagem Dedutiva

A aprendizagem dedutiva utiliza-se da lógica dedutiva para extração de conhecimento. A partir de um conjunto de declarações (afirmações), conhecidas como *premissas*

e usando de argumentos dedutivos é possível obter algumas conclusões, sendo a conclusão verídica dada pela veracidade das premissas.

Um exemplo clássico atribuído a Aristóteles pode ser utilizado para explicar o raciocínio dedutivo:

Todo Homem é mortal. *(premissa)*
Sócrates é um homem. *(premissa)*
Sócrates é mortal. *(conclusão)*

Generalizando a idéia, temos:

Se A é verdade então B é verdade.

O raciocínio lógico dedutivo consiste em determinar a veracidade de uma declaração ainda não conhecida, exclusivamente baseando-se na veracidade de outras declarações pré-estabelecidas (premissas) e utilizando de regras de inferência. A partir disto podemos concluir:

Se A então B. Se B então C. Se C então D.

Sempre considerando a veracidade das premissas A, B e C, podemos por dedução concluir que D é verdadeiro.

1.1.2 Aprendizagem Indutiva

Outro importante tipo de aprendizagem é o modelo indutivo, no qual o processo de obtenção de conhecimento é realizado sobre experiências anteriores. Este modelo é caracterizado como aprendizagem que parte do específico para o geral.

De acordo com (FERRO, 2004) a indução é uma forma de inferência lógica que permite obter conclusão genéricas sobre um conjunto particular de exemplos, ou casos observados.

O conjunto de exemplos utilizados na aprendizagem indutiva consiste em um conjunto de características de um domínio que juntas determinam o estado do mesmo (rótulo). O estado pode ser ou não informado no processo de aprendizagem determinando

assim o modo de aprendizagem. Os modos de aprendizagem são melhor discutidos na seção 1.2.1.

A tabela 1 mostra o formato comumente utilizado para representar um conjunto de dados d_i com n exemplos e m características do domínio em questão. Os vetores $x_i, i = 1, 2, ..., m$ descrevem o vetor de características e $y_i, i = 1, 2, ..., n$ sendo o respectivo rótulo.

	x_1	x_2	...	x_m	y
d_1	x_{11}	x_{12}	...	x_{1m}	y_1
d_2	x_{21}	x_{22}	...	x_{2m}	y_2
...	...	...	...	...	...
d_n	x_{n1}	x_{n2}	...	x_{nm}	y_n

Tabela 1: Formato dos exemplos rotulados

O processo de aprendizagem indutiva é utilizado na maioria das técnicas de aprendizagem de máquina (MITCHELL, 1997).

1.2 Classificação de processos de aprendizagem

Diversos sistemas de AM podem ser classificados com base em suas características, quanto aos seguintes conceitos: **modo**, **paradigma** e **forma de aprendizagem**.

1.2.1 Modos de aprendizagem

Os modos dizem como os exemplos serão apresentados ao algoritmo de aprendizagem, denominado de *indutor*, especificamente em relação a presença ou não do conjunto de atributos de saída (rótulos). Destacam-se três modos: *supervisionado*, *não-supervisionado* e *semi-supervisionado*.

Supervisionado Também conhecido como aprendizagem com professor, neste modelo há um conhecimento sobre as saídas do ambiente, dada determinadas características. Em outras palavras podemos dizer que o indutor recebe o conjunto de dados de entrada (características) e o conjunto de dados de saída, com isso podendo saber se o sistema está ou não aprendendo. Dentre as técnicas apresentadas neste trabalho, as redes neurais, algoritmos genéticos, máquina de vetor suporte, *swarm intelligence* e alguns tipos de redes *Bayesianas* se enquadram neste modo de aprendizagem.

Não-supervisionado Quando não há conhecimento dos rótulos dos exemplos, uma aprendizagem não-supervisionada pode ser realizada. Neste modelo geralmente são dadas condições para realizar uma *medida independente da tarefa*, da qualidade da representação que o modelo deve aprender (HAYKIN, 1999), como a competição entre os indivíduos, comparações por similaridade, dentre outras, sendo o melhor indivíduo a solução do sistema. Técnicas que utilizam este modo de treinamento são: redes neurais do tipo mapas auto-organizáveis; métodos de clusterização como agrupamento hierárquico e k-médias (JAIN; MURTY; FLYNN, 1999).

Semi-supervisionado Surgido recentemente, consiste em um modelo híbrido composto pelos dois modos anteriores. Possibilita a aprendizagem mesmo quando um pequeno conjunto de dados rotulados é disponível. Boas referências sobre aprendizagem semi-supervisionada são (BLUM; MITCHELL, 1998) e (MATSUBARA, 2004).

1.2.2 Paradigmas de Aprendizagem

Com o objetivo de extrair conhecimento a partir de exemplos, utilizou-se de várias ferramentas de outras áreas para realização de tal tarefa, como: estatística, manipulação simbólica, técnicas de similaridade, entre outras. Com isso vários paradigmas foram propostos, os quais são: **simbólico**, **Estatístico**, **Baseado em Instâncias**, **Conexionista** e **Evolutivo**.

Simbólico Sistemas de aprendizagem simbólica, utilizam-se de representações simbólicas, de modo a aprender construindo conceitos através de análise de exemplos e contra-exemplos deste conceito. Representações simbólicas estão tipicamente na forma de expressões lógica, árvores de decisão, regras de produção ou redes semânticas.

Estatístico Diversas pesquisas na área da estatística geraram como frutos vários métodos de classificação, muitos deles semelhantes aos métodos empregados em aprendizagem de máquina. De modo geral estes métodos utilizam modelos estatísticos para encontrar uma boa aproximação do conceito induzido. Dentre os métodos destaca-se a aprendizagem *Bayesiana* que a partir de um conhecimento prévio do domínio e combinado com exemplos de treinamento é capaz de obter a probabilidade final de uma hipótese (probabilidade *a posteriori*).

Baseado em instâncias Este paradigma utiliza casos similares já rotulados e a partir destes podem assumir que um novo caso terá o mesmo rótulo, de acordo com alguma medida de similaridade. Algumas medidas de similaridades utilizadas são distância euclidiana,

distância de Manhattan, produto interno, entre outras. Dentre as técnicas que utilizam este paradigma pode-se destacar: Raciocínio Baseado em Casos e Vizinhos mais próximos *(Nearest Neighbors)*.

Conexionista A principal técnica deste paradigma são as redes neurais, que é um modelo simplista de cérebro humano, no qual o conhecimento é armazenado na forma de pesos das *conexões* entre unidades de processamento (nós). O paradigma conexionista é utilizado para descrever a área que estuda as potencialidades e limitações desta técnica.

Evolutivo Baseando-se na teoria da evolução das espécies, desenvolvida por Charles Darwin, este paradigma utiliza um conjunto de indivíduos que competem entre si, para determinar quais serão extintos (menos aptos) e quais sobreviveram e geraram descendentes (mais aptos). Um classificador evolutivo consiste de uma população de indivíduos de classificação que competem entre si para fazer predição. Técnicas que utilizam aprendizagem evolutiva são Algoritmo Genético, Programação genética e *Swarm Intelligence*.

1.2.3 Formas de Aprendizagem

Dentro da aprendizagem indutiva é possível distinguir entre duas *formas de aprendizagem*, que diz respeito á forma como os exemplos serão apresentados ao indutor, aprendizagem *incremental* e *não-incremental*.

Incremental A definição do conhecimento (hipótese) é atualizada ou não a cada novo exemplo observado. Esta forma possui a característica de ser flexível, porém a ordem dos exemplos pode ser importante. Resumidamente o algoritmo na forma incremental tenta atualizar a hipótese antiga sempre que novos exemplos são adicionados ao conjunto de treinamento. Uma das vantagens de um algoritmo incremental é agilidade com que o conhecimento é atualizado a cada nova observação (FERRO, 2004).

Não-Incremental Também conhecida como modo em lote ou *batch*, esta forma de apresentação possui como principal característica a exigência de que todos os exemplos estejam simultaneamente disponíveis para algoritmo de aprendizagem, sendo a hipótese gerada a partir de todo o conjunto de exemplos.

1.3 Critérios de avaliação

Após a execução do algoritmo de aprendizagem, geralmente é utilizada alguma forma de avaliar se o sistema obteve sucesso na extração do conhecimento. Há várias maneiras para realizar esta tarefa, os critérios mais utilizados são:

1. **Precisão na classificação** Definida pela porcentagem de predições corretas, dado um conjunto de exemplos ainda não conhecido. Esta técnica, conhecida como *validação cruzada* (KOHAVI, 1995), consiste de dividir o conjunto total de exemplos em dois subconjuntos, um para treinamento e outro para teste, os exemplos para teste são apresentados e a porcentagem de predições corretas é analisada.

2. **Complexidade Computacional** Analisa os recursos computacionais necessários, tempo e espaço, para medir a aprendizagem. Podemos realizar uma análise sob dois aspectos diferentes:

    1. Complexidade de Geração: custo necessário para extração das regras através dos exemplos;
    2. Complexidade de Execução: custo computacional necessário para classificar um novo exemplo utilizando as regras aprendidas.

3. **Legibilidade na descrição das regras** É desejável que as regras geradas sejam de fácil compreensão para o ser humano, mostrando ao usuário algo interessante sobre o domínio de aplicação. As regras então podem ser usadas diretamente pelo usuário como uma intensificação de seu próprio conhecimento (FERRO, 2004). Esse critério é interessante em casos que as regras devem estar transparentes ao entendimento humano, como é o caso da mineração de dados.

1.4 Considerações

Foram apresentados os principais conceitos de aprendizagem de máquina, destacando algumas características que possibilitam a classificação dos diversos tipos de métodos para extração e armazenamento de conhecimento.

Também foram descritos dois principais tipos de aprendizagem, uma que utiliza a lógica dedutiva e outra que utiliza um conjunto de exemplos previamente observados.

O aprendizagem indutiva será o foco deste trabalho, pois é o modelo de aprendizagem mais utilizado em inteligência artificial e tem produzido sólidos resultados (BARANAUSKAS; MONARD, 2000). Mais especificamente os métodos aqui abordados serão destinados à tarefa de classificação, utilizando um conjunto de dados rotulados, obtidos em um banco de dados para aprendizagem de máquina.

Uma lição básica, porém muito importante é a não existência de um único método que domina todos os outros em todos os problemas. O importante é conhecer as capacidades e limitações dos diferentes algoritmos e então determinar qual é o melhor para um problema específico.

Com uma grande variedade de algoritmos de aprendizagem, uma abordagem comumente utilizada na prática é aplicar o problema em diferentes algoritmos, estimando sua acurácia e então escolhe o algoritmo que obteve a melhor acurácia para o domínio dado.

2 Redes Neurais Artificiais

O cérebro humano é uma máquina altamente poderosa e complexa capaz de processar uma grande quantidade de informações em tempo mínimo. As unidades principais do cérebro são os neurônios e é por meio deles que as informações são transmitidas e processadas.

As tarefas realizadas pelo cérebro intrigam os pesquisadores, como por exemplo, o cérebro é apto a reconhecer um rosto familiar dentre uma multidão, em tempo da ordem de milésimos de segundo. As respostas sobre alguns enigmas do funcionamento do cérebro se perpetuam até os dias de hoje.

O que é conhecido sobre o funcionamento do cérebro é que o mesmo desenvolve suas regras através da "experiência" adquirida em situações vividas anteriormente. Segundo (HAYKIN, 1999) o desenvolvimento do cérebro humano dá-se principalmente nos dois primeiros anos de vida, mas se arrasta por toda a vida.

Inspirando-se neste modelo, diversos pesquisadores tentaram simular o funcionamento do cérebro, principalmente o processo de aprendizagem por experiência, a fim de criar sistemas inteligentes capazes de realizar tarefas como classificação, reconhecimento de padrões, processamento de imagens entre outras.

Como resultados destas pesquisas surgiram o modelo do neurônio artificial e posteriormente um sistema com vários neurônios interconectados, a chamada Rede Neural.

2.1 Inspiração Biológica

Como dito anteriormente, o neurônio é a unidade básica do cérebro humano. Uma célula especializada na transmissão de informações, pois nelas estão introduzidas propriedades de excitabilidade e condução de mensagens nervosas.

De acordo com (ARBIB, 2002) o neurônio é constituído por três partes principais: a *soma* ou corpo celular, do qual emanam algumas ramificações denominadas de

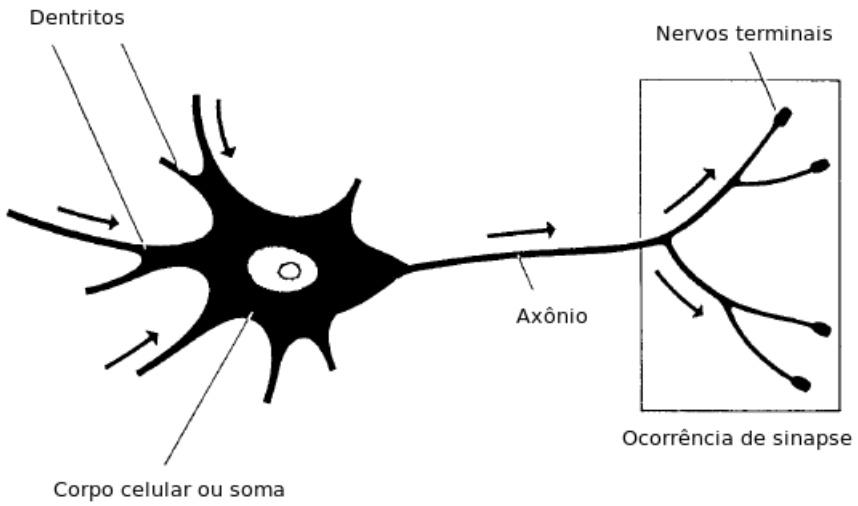

Figura 1: Modelo abstrato do neurônio biológico. [Adaptado de (ARBIB, 2002)]

dendritos, e por fim outra ramificação descendente da soma, porém mais extensa chamada de *axônio*. Nas extremidades dos axônios estão os nervos terminais, que pelos quais é realizada a transmissão das informações para outros neurônios, esta transmissão é conhecida como *sinapse*.

Arbib (2002) observa que a *soma* e os dendritos formam a superfície de entrada do neurônio e o axônio a superfície de saída do fluxo de informação. A figura 1 identifica as partes principais do neurônio e as setas mostram o fluxo da informação

A informação transmitida pelos neurônios na realidade são impulsos elétricos, que segundo (REIS, 2008) o impulso elétrico é a mensagem que os neurônios transmitem uns aos outros, ou seja, é a propagação de um estímulo ao longo dos neurônios que pode ser qualquer sinal captado pelos receptores nervosos. O estímulo resulta na capacidade dos neurônios se excitarem através da ação de um estímulo.

2.2 Modelo Matemático do Neurônio Biológico

A partir da estrutura e funcionamento do neurônio biológico, pesquisadores tentaram simular este sistema em computador.

O modelo mais bem aceito foi proposto por McCulloch e Pitts (1943), conhecido como *Perceptron*, o qual implementa de maneira simplificada os componentes e o funcionamento de um neurônio biológico.

Neste modelo, os impulsos elétricos provenientes de outros neurônios são representados pelos chamados *sinais de entrada*, (x_j), dentre os vários estímulos recebidos, alguns excitarão mais e outros menos o neurônio receptor, essa medida de quão excitatório é o estímulo é representada no modelo de McCulloch e Pitts através dos *pesos sinápticos*, quanto maior o valor do peso, mais excitatório é o estímulo. Os pesos sinápticos são representados por w_{kj}, onde k representa o índice do neurônio em questão e j se refere ao terminal de entrada da sinapse à qual o peso sináptico se refere.

A *soma* é representada por uma composição de dois módulos, o primeiro é uma *junção aditiva*, somatório dos estímulos (sinais de entrada) multiplicado pelo seu fator excitatório (pesos sinápticos), e posteriormente uma *função de ativação*, que definirá com base nas entradas e pesos sinápticos, qual será a saída do neurônio. O axônio é aqui representado pela saída (y_k) obtida pela aplicação da função de ativação.

Assim como no modelo biológico, o estímulo pode ser excitatório ou inibitório, representado pelo peso sináptico positivo ou negativo respectivamente. A figura 2 apresenta o modelo de neurônio artificial proposto por McCulloch e Pitts (1943).

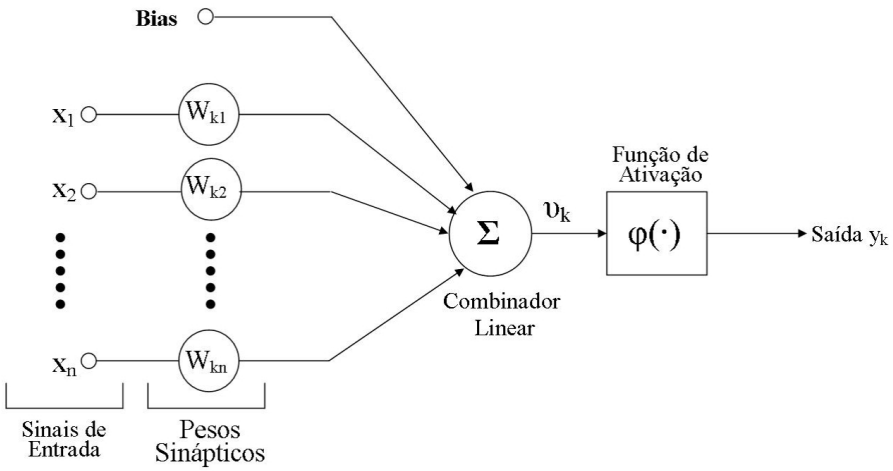

Figura 2: Modelo de neurônio artificial proposto por McCulloch e Pitts

O modelo proposto possui uma natureza binária, tanto os sinais de entrada quanto a saída são valores binários. McCulloch acreditava que o funcionamento do sistema nervoso central possuía um caráter binário, ou seja, um neurônio influencia ou não outro neurônio, mas posteriormente mostrou-se que não era dessa forma. Esta característica do modelo de McCulloch e Pitts foi referenciada como propriedade do "tudo ou nada" (HAYKIN, 1999).

Em termos matemáticos o neurônio artificial i pode ser representado como:

$$u_k = \sum_{j=1}^{m} w_{kj} \cdot x_j \tag{2.1}$$

onde m é o número de sinais de entrada incidentes no neurônio k e posteriormente a aplicação da função de ativação

$$y_k = \varphi(u_k) \tag{2.2}$$

Basicamente a função de ativação recebe o valor provido pela junção aditiva, denominado por (HAYKIN, 1999) como "campo local induzido", atribuindo 1, se o campo local induzido daquele neurônio for não-negativo e 0 caso contrário. A função de ativação adotada no modelo é uma função degrau definida como segue:

$$\varphi(u) = \begin{cases} 1 & \text{, se } u \geq 0 \\ 0 & \text{, se } u < 0 \end{cases} \tag{2.3}$$

Outras funções de ativação foram apresentadas na literatura, dentre elas a função *sigmóide*, a qual segundo (HAYKIN, 1999) é de longe a forma mais utilizada na construção de redes neurais artificiais, tem um comportamento estritamente crescente que exibe um balanceamento entre comportamento linear e não-linear.

Um exemplo de função sigmóide é a *função logística* definida por

$$\varphi(u) = \frac{1}{1 + exp(-au)} \tag{2.4}$$

onde a é o parâmetro de inclinação da função sigmóide. Outro fator relevante desta função, como observado por (HAYKIN, 1999) é que esta função é diferenciável, que é uma característica importante da teoria das redes neurais.

McCulloch e Pitts mostraram que o *Perceptron* era capaz de computar funções lógicas e aritméticas, como funções lógicas do tipo "or" e "and" (BASHEER; HAJMEER, 2000).

Rosenblatt (1958) propôs o Perceptron como o primeiro modelo para aprendizagem supervisionada, chamada de Rede Perceptron ou Perceptron de Camada Única. Esta é a forma mais simples de rede neural, usada para classificação de padrões linearmente separáveis (HAYKIN, 1999).

2.3 Redes Neurais Artificiais

Uma rede neural típica é constituída por um conjunto de neurônios interligados, influenciando uns aos outros formando um sistema maior capaz de armazenar conhecimento adquirido por meio de exemplos apresentados e assim podendo realizar inferências sobre novos exemplos (novas situações) desconhecidos.

As redes neurais são comumente apresentadas como um grafo orientado, onde os vértices são os neurônios e as arestas as sinapses, a direção das arestas informa o tipo de alimentação, ou seja, como os neurônios são alimentados (recebem sinais de entrada), Alimentação de redes neurais é discutida na seção 2.3.1. Um exemplo de uma rede neural como um grafo orientado é mostrado na figura 3.

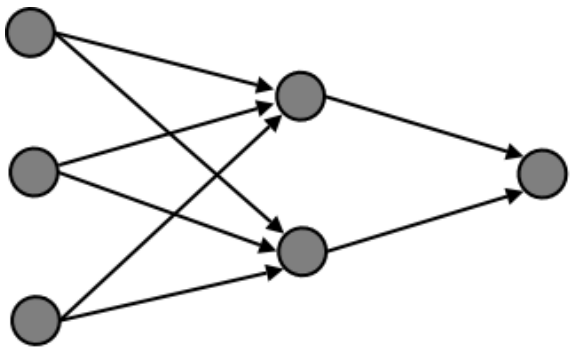

Figura 3: Rede Neural representada como um grafo orientado

De acordo com (ACHARYA et al., 2003) as redes neurais derivam seu poder devido a sua estrutura massiva e paralela e a habilidade de aprender por experiência. Essa experiência é transmitida por meio de exemplos obtidos do mundo real, definidos como um conjunto de características que levam a uma determinada situação. Se a situação gerada pela combinação de características for informada a rede, a aprendizagem é dita ser supervisionada e não-supervisionada caso contrário.

O conhecimento obtido pela rede através dos exemplos é armazenado na forma de pesos das conexões, os quais serão ajustados a fim de tomar decisões corretas a partir de novas entradas, ou seja, novas situações do mundo real não conhecidas pela rede.

O processo de ajuste dos pesos sinápticos é realizado pelo algoritmo de aprendizagem, responsável em armazenar na rede o conhecimento do mundo real obtido através de exemplos. Na literatura são relatados vários algoritmos de aprendizagem, dentre eles o *backpropagation* que é o algoritmo mais utilizado (BASHEER; HAJMEER, 2000).

Acharya et al. (2003) identifica três decisões importantes no processo de

construção de uma rede neural: (i) a topologia da rede, a topologia de redes neurais serão discutidas na seção 2.3.1; (ii) algoritmo de aprendizagem e (iii) função de ativação.

2.3.1 Topologias das Redes Neurais

A Topologia de uma rede neural diz respeito à disposição dos neurônios na rede, como são estruturados. A topologia da rede está diretamente ligada ao tipo de algoritmo de aprendizagem utilizado.

Em geral é possível identificar três classes de topologias de rede:

1. **Redes alimentadas adiante (*Feed-forward networks*)**

   Alguns tipos de redes são estruturados em forma de *camadas*, os neurônios são dispostos em conjuntos distintos e ordenados seqüencialmente, conjuntos esses denominados de *camadas*. Nas redes alimentadas adiante, o fluxo de informação é sempre da camada de entrada para a camada de saída. Haykin (1999) distingue as redes alimentadas adiante em Redes de Camada Única e Redes de Múltiplas Camadas, apenas diferenciando do número de camadas, mas o conceito de alimentação adiante ou direta é o mesmo. A figura 4 apresenta um modelo de rede alimentada adiante.

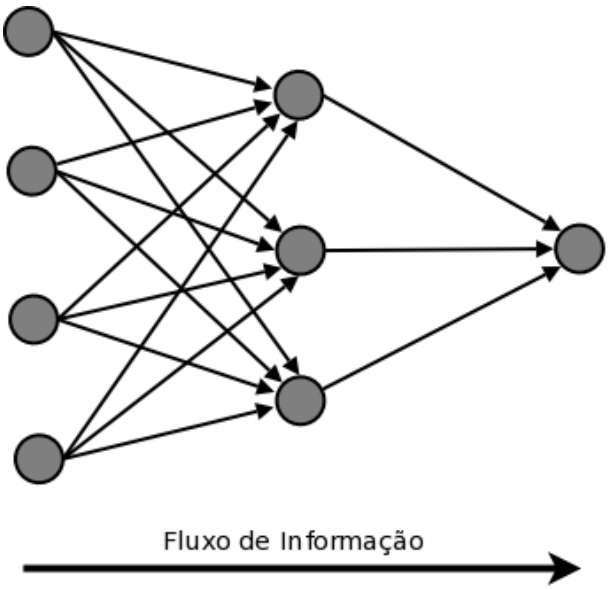

Figura 4: Rede neural alimentada adiante

Algumas características importantes de uma rede alimentada adiante são apresentadas abaixo:

(a) Os neurônios são arranjados em camadas, a camada inicial recebe os sinais de entrada e a camada final obtém as saídas. As camadas intermediárias são chamadas de *camadas ocultas*.

(b) Cada neurônio de uma camada é conectado com todos os neurônios da camada seguinte.

(c) Não há conexões entre neurônios de uma mesma camada.

A informação (alimento) provém da camada de entrada ou neurônios (nós) fonte e posteriormente é transmitida para as camadas seguintes até a camada de saída. Este tipo de rede é também conhecido como *rede acíclica*, pois na representação de grafos não possui ciclos.

2. **2. Redes Recorrentes (***Feed-backward networks***)**

Nas redes recorrentes há a ocorrência de *realimentação*, na qual a saída de um neurônio é aplicada como entrada no próprio neurônio e/ou em outros neurônios de camadas anteriores, ou seja, há a ocorrência de um ciclo no grafo. A figura 5 mostra uma rede neural, na qual alguns neurônios são realimentados.

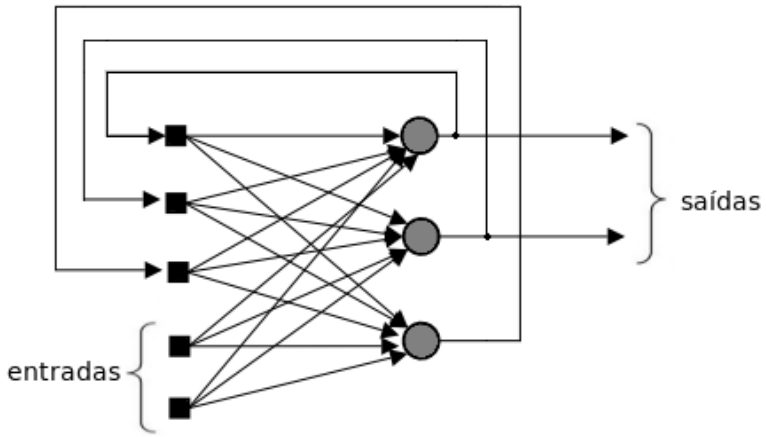

Figura 5: Exemplo de uma rede neural com realimentação

3. **Redes Competitivas**

Nesta classe de redes neurais os neurônios estão divididos em duas camadas, a camada de entrada ou nós fontes e a camada de saída, conhecida como *"grade"*. Os neurônios da grade são forçados a competir entre si, com base no nível de similaridade entre o padrão de entrada e a grade de neurônios e somente o neurônio vencedor será disparado (ativado) a cada iteração (BASHEER; HAJMEER, 2000). Redes desta classe utilizam

algoritmo de aprendizagem competitivo. A rede mais conhecida desta classe é a rede de Kohonen, também conhecida como Mapa Auto-Organizável.

2.3.2 Processo de Aprendizagem

A principal característica de uma rede neural é a sua capacidade de aprender a partir de um ambiente e de melhorar seu desempenho por meio da aprendizagem. Esta aprendizagem, como observa (BASHEER; HAJMEER, 2000), é um processo de atualização da representação interna do sistema em resposta a um estimulo externo, podendo desempenhar uma tarefa específica.

A atualização da representação é realizada sob a forma de modificação da arquitetura, ajuste dos pesos das conexões entre os neurônios e ativando regras de neurônios individuais.

As regras de aprendizagem definem como a rede deve ajustar os pesos sinápticos. Haykin (1999) identifica quatro tipos de regras de aprendizagem:

1. **Aprendizagem por Correção de Erro**
   Utilizado em treinamento supervisionado, esta técnica ajusta os pesos sinápticos por meio do erro, que é obtido através da diferença entre o valor de saída da rede e o valor esperado em um ciclo de treinamento. Com isso gradualmente vai diminuindo o erro geral da rede.

2. **Aprendizagem Hebbiana**
   Baseado no postulado de aprendizagem de Hebb (HEBB, 1949), que afirma: "se dois neurônios em ambos os lados de uma sinapse são ativados sincronamente e simultaneamente, então a força daquela sinapse é seletivamente aumentada". Este processo de treinamento é feito localmente, ajustando o peso das conexões baseado nas atividades dos neurônios.

3. **Aprendizagem de Boltzmann**
   Um método de aprendizagem estocástico derivado das idéias enraizadas na mecânica estatística (HAYKIN, 1999). Uma rede que implementa o método de aprendizagem de Boltzmann é dita ser uma Máquina de Boltzman (MB). Neste modelo os neurônios são estocásticos, podendo residir em dois estados possíveis, ligado (+1) e desligado (-1), e ainda são divididos em dois grupos funcionais, *presos* e *livres*, sendo responsáveis pela interação com o ambiente e pela explicação das restrições subjacentes dos padrões de en-

trada do ambiente, respectivamente. Um ponto importante na MB é que seus neurônios possuem conexões bidirecionais (HINTON; ACKLEY; SEJNOWSKI, 1985). De acordo com (HAYKIN, 1999) este modelo pode ser visto como um procedimento de aprendizagem não-supervisionado para modelar uma distribuição de probabilidade, especificada pelos padrões presos aos neurônios visíveis com probabilidades apropriadas.

4. **Aprendizagem Competitiva**
   Neste modelo de aprendizagem os neurônios são forçados a competir entre si e somente um será ativo, em uma dada iteração, o vencedor, ou seja, o que tiver maior similaridade com o padrão de entrada. Todos os pesos dos neurônios próximos ao neurônio vencedor terão seus valores ajustados.

As regras que definem como serão atualizados os pesos sinápticos constituem o algoritmo de aprendizagem. O algoritmo mais utilizado na literatura é o algoritmo *backpropagation*, baseado na propagação do erro.

2.3.3 Redes Multilayer Perceptron

Redes *Multilayer Perceptron* (MLP) são redes alimentadas adiante que possuem uma ou mais camadas de neurônios entre as camadas de entrada e saída, chamada de *camada oculta* (LIPPMANN, 1987). Esta camada adiciona um poder maior em relação às redes *Perceptron* de camada única, que classifica apenas padrões linearmente separáveis, sendo os neurônios ocultos responsáveis por capturar a não-linearidade dos dados.

Neste modelo todos os neurônios são ligados aos neurônios da camada subseqüente, não havendo ligação com os neurônios laterais (da mesma camada) e também não ocorre realimentação.

A aprendizagem de uma rede neural MLP é um processo iterativo, conhecido como aprendizagem por experiência, no qual padrões de treinamento (exemplos) são apresentados a rede e com base nos erros obtidos, são realizados ajustes nos pesos sinápticos, com o intuito de diminuir o erro nas próximas iterações.

O principal algoritmo de treinamento é o algoritmo de retropropagação de erro (*error backpropagation*), baseado na regra de aprendizagem por correção de erro, que consiste basicamente de dois passos, um para frente e outro para trás.

O passo para frente é chamado de propagação, os valores provindos dos neurônios de entrada (nós fontes) são aplicados aos neurônios ocultos e posteriormente suas

saídas são aplicadas como entradas aos neurônios da camada final, obtendo a resposta da rede. Durante este passo os pesos sinápticos da rede são todos fixos (HAYKIN, 1999).

Já o passo para trás é incumbido de ajustar os pesos sinápticos, por meio do cálculo do erro realizado na camada de saída, os pesos sinápticos entre as camadas antecessoras são ajustados de acordo com uma regra de correção de erro. A figura 6 mostra uma rede MLP.

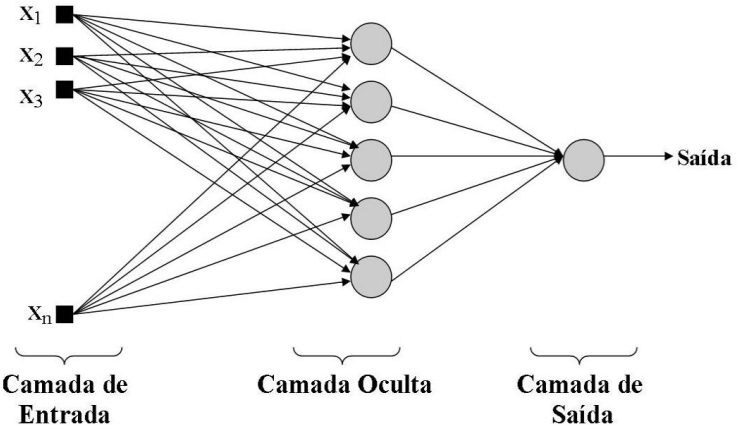

Figura 6: Arquitetura da rede MLP

O algoritmo *backpropagation* utiliza a técnica de busca por gradiente para minimizar a função custo. O algoritmo 1 mostra as etapas do *backpropagation*, baseado em (LIPPMANN, 1987).

Algoritmo 1: Algoritmo de treinamento *backpropagation*

```
1  begin
2      Atribuição de valores iniciais aos pesos sinápticos;
3      repeat
4          Apresentação à rede dos padrões de entrada e as saídas desejadas;
5          Cálculo dos valores de saída dos neurônios ocultos;
6          Cálculo dos valores de saída dos neurônios de saída (resposta real da rede);
7          Cálculo do erro (diferença entre resposta da rede e valor esperado);
8          Ajuste dos pesos sinápticos;
9      until Condição de parada não satisfeita ;
10 end
```

As etapas do *backpropagation* são melhores discutidas e detalhadas abaixo.

Linha 2 O processo de atribuição de valores iniciais dos pesos sinápticos é comumente realizado de maneira aleatória, atribuindo pequenos valores, geralmente no intervalo $\{0,1\}$;

Linha 4 Sendo o treinamento supervisionado é necessária a apresentação dos padrões de entrada juntamente com a saída conhecida previamente;

Linhas 5 e 6 O cálculo dos valores de saída são realizados pela aplicação do campo local induzido (v) (HAYKIN, 1999) à uma função de ativação. De uma maneira matemática, o campo local induzido para o neurônio j na camada l na iteração n é obtido por

$$v_j^{(l)}(n) = \sum_{i=1}^{r+1} w_{ji}^{(l)}(n) y_i^{(l-1)}(n) \tag{2.5}$$

onde $y_i^{(l-1)}(n)$ é o sinal de saída do neurônio i na camada anterior $l-1$, na iteração n, $w_{ji}^{(l)}$ é o peso sináptico do neurônio j da camada l, que é alimentado pelo neurônio i da camada $l-1$ e r o número de neurônios na camada anterior $(l-1)$. Observe que ao invés de r é utilizado $r+1$, isto se deve ao fato da inclusão de um *bias*, um neurônio adicional com valor $+1$. O bias equivale a dizer: $y_{r+1}^{(l-1)}(n) = +1$.

Denotaremos por $\varphi_j(v_j)$, sendo $\varphi(\cdot)$ a função de ativação e v_j o campo local induzido do neurônio j, o valor de saída de um neurônio j da camada l. Assim o valor de saída do neurônio

$$y_j^{(l)} = \varphi(v_j(n)) \tag{2.6}$$

Valores de saída estarão dentro do intervalo definido pela função de ativação. As funções de ativação geralmente utilizadas podem ser encontradas em (HAYKIN, 1999)

Linha 7 O sinal de erro da saída da rede, na iteração n é calculada por $e_j(n) = d_j(n) - y_j(n)$, onde d_j é a j-ésima resposta desejada e y_j é a j-ésima resposta da rede;

Linha 8 O ajuste dos pesos é o núcleo deste algoritmo, sendo realizado da camada oculta para a camada de entrada. Em qualquer camada l os valores dos pesos $w_{ji}^{(l)}$ na iteração n serão ajustados da iteração anterior $(n-1)$ de acordo com:

$$w_{ji}^{(l)}(n) = w_{ji}^{(l)}(n-1) + \Delta w_{ji}^{(l)}(n) \tag{2.7}$$

onde $\Delta w_{ji}^{l}(n)$ é correção aplicada ao peso da conexão. A correção é determinada pela *regra delta modificada*, definida como segue:

$$\Delta w_{ji}^{(l)}(n+1) = \eta \delta_j^{(l)} y_i^{(l-1)} + \mu \Delta w_{ji}^{(l)}(n) \tag{2.8}$$

onde η é a taxa de aprendizagem, $\delta_j^{(l)}$ é o gradiente local, μ é a constante de momento e $y_i^{(l-1)}$ é o sinal de saída do neurônio i na camada anterior $l-1$.

A taxa de aprendizagem define o tamanho do passo de atualização e a constante de momento é utilizado para que o método possa fugir do mínimo local na superfície de erro, objetivando o mínimo global.

O gradiente local é calculado de maneira distinta entre neurônios de saída e ocultos. Sendo L a camada de saída, o gradiente do neurônio j, na iteração n da camada de saída é calculado por:

$$\delta_j^{(L)}(n) = e_j^{(L)}(n)\varphi'(v_j^{(L)}) \tag{2.9}$$

onde $\varphi'(\cdot)$ é derivada da função de ativação. Para os neurônios ocultos a correção é obtida por

$$\delta_j^{(l)}(n) = \varphi'(v_j^{(l)}) \left(\sum_{i=1}^{r+1} \delta_i^{(l+1)} w_{ij}^{(l+1)} \right) \tag{2.10}$$

sendo l uma camada oculta qualquer.

Linha 9 Basheer e Hajmeer (2000) identificam alguns critérios de parada, dentre eles (i) erro de treinamento $(e \leq \epsilon)$, (ii) gradiente do erro menor que um σ ou (iii) utilizando técnica de *validação cruzada*.

O critério de parada (iii) é geralmente utilizado através da análise gráfica do comportamento do erro, com base na técnica de validação cruzada (KOHAVI, 1995) e (AMARI et al., 1997).

2.3.4 Dificuldades de implementação da rede MLP

Para que uma rede *Multilayer Perceptron* possa obter bons resultados na aplicação em problemas reais, uma boa configuração da mesma deve ser feita. Esta configuração é realizada de maneira distinta dependendo de várias características do problema, como o tipo dos dados de entrada (inteiro, *booleanos*, híbrido), número de padrões disponíveis para treinamento e teste, a dimensionalidade dos dados de entrada, entre outros. Para isso, valores ideais dos vários parâmetros da rede MLP devem ser utilizados, porém estes valores não são de fácil obtenção.

De acordo com (BASHEER; HAJMEER, 2000) alguns parâmetros são determinados por tentativa e erro, ou seja, são atribuídos vários valores distintos aos parâmetros e analisando os resultados obtidos, a melhor configuração é escolhida. Dentre esses parâmetros, a taxa de aprendizagem, a constante de momento, o número de camadas ocultas e o número de neurônios nas camadas ocultas, possuem um maior grau de dificuldade para o ajuste dos valores.

Algumas soluções para contornar este problema foram sugeridas, como a aplicação de outras técnicas de aprendizagem de máquina, as quais são conhecidas pela alta aplicabilidade em problemas de otimização, como Algoritmos Genéticos (HAN; MAY, 1996), Otimização por Enxame de Partículas (ZHANG; SHAO; LI, 2000) e Otimização por Colônias de Formigas (GONÇALVES; CAMARGO-BRUNETO, 2008c).

Outra dificuldade é a determinação do número ideal de ciclos de treinamento da rede, que de acordo com (BASHEER; HAJMEER, 2000) é determinado por tentativa e erro. Se um número muito grande de ciclos de treinamento for aplicado, a rede entra em um processo de "memorização" dos padrões a ela apresentados, chamado de super-treinamento (*overtraining*), perdendo assim a capacidade de generalização. E se um número muito pequeno for aplicado, a rede torna-se incapaz de representar os dados. Este fenômeno é mostrado pela figura 7.

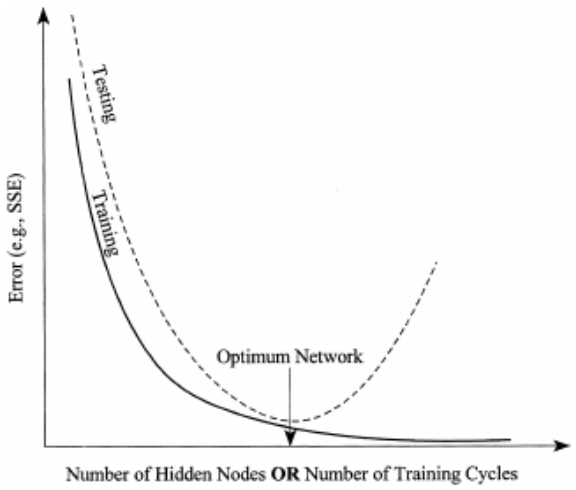

Figura 7: Número de ciclos de treinamento ótimo. [Fonte: (BASHEER; HAJMEER, 2000)]

O super-treinamento é identificado quando o erro de teste, obtido pela validação cruzada, começa a aumentar depois de ter diminuído. Para evitar o super-treinamento a aprendizagem deve ser interrompida quando os erros não variam mais, indicando que a rede está treinada, ou quando o erro de teste aumentar, depois de ter diminuído. A técnica de visualização gráfica dos erros obtidos pela utilização da técnica de validação cruzada do tipo

Holdout é comumente utilizada (KOHAVI, 1995) e (AMARI et al., 1997). Porém nesta abordagem há a necessidade de executar um número grande de ciclos de treinamento antes da geração do gráfico. Uma maneira de automatizar este procedimento de identificação gráfica utilizando regressão linear foi proposta por (GONÇALVES; CAMARGO-BRUNETO, 2008b).

2.4 Outros Modelos

Diferentes modelos de redes neurais foram propostos na literatura, com diferentes algoritmos de treinamento, de topologias, de paradigmas de aprendizagem e com aplicações diversas. As redes popularmente utilizadas são: Redes de Função de Base Radial, a Rede de Kohonen ou Mapas Auto-Organizáveis e as Redes de Hopfield, que são apresentadas a seguir.

2.4.1 Redes de Função de Base Radial

Diferentemente das Redes Perceptron Multicamadas, as redes de Função de Base Radial, do inglês *Radial Basis Function* (RBF), trabalham o projeto de uma rede neural como um problema de ajuste de curvas (aproximação) em um espaço de alta dimensionalidade (HAYKIN, 1999).

Como observado por (HAYKIN, 1999), a aprendizagem de uma rede RBF é um processo de busca em uma superfície multidimensional, que forneça melhores ajustes para os dados de treinamento, sendo os "melhores ajustes" uma medida no sentido estocástico e a generalização equivale ao uso da superfície multidimensional para interpolação dos dados de teste.

As redes RBF diferem das redes MLP por três características principais, como observa (VON ZUBEN; ATTUX, 2008):

1. Sempre apresenta uma única camada intermediária (oculta);
2. Neurônios da saída são sempre lineares;
3. Os neurônios da camada intermediária têm funções de base radial como função de ativação, ao invés de funções sigmoidais ou outras.

Uma rede RBF em sua forma mais simples é uma rede constituída por três camadas, com papéis distintos. A camada inicial ou nós fonte são encarregados de prover

informações do ambiente (dados de entrada), a segunda e única camada oculta da rede é responsável pela transformação não-linear do espaço de entrada para o espaço oculto, sendo o último um espaço de alta dimensionalidade (na maioria das aplicações). E por fim a camada de saída, que é linear, fornece uma resposta ao estímulo gerado pela aplicação de um padrão (dados de entrada), pela camada de entrada.

As redes RBF podem ser aplicadas principalmente em classificação de padrões e aproximação de função (GUPTA; JIN; HOMMA, 2003).

2.4.2 Mapas Auto-Organizáveis

Os Mapas Auto-Organizáveis, do inglês *Self-Organizing Maps* (SOM), constituem outra classe de redes neurais, as redes com aprendizagem não-supervisionada. Esta rede é também conhecida como um tipo de Rede de Kohonen, devido ao fato deste modelo ter sido proposto por Kohonen (1982).

A base biológica do SOM está na forma que os neurônios se organizam, muitas vezes refletem algumas características físicas sentidas pelos estímulos externos (LIPPMANN, 1987). Ou seja, o cérebro humano é organizado em várias áreas, de modo que entradas sensoriais diferentes são representadas por mapas computacionais ordenados topologicamente (HAYKIN, 1999).

Utilizando este conhecimento biológico, a rede de Kohonen utiliza-se de duas camadas, a camada de entrada e a grade pós-sináptica ou mapa de características, sendo a última um arranjo bi-dimensional de neurônios. A figura 8 mostra um mapa auto-organizável.

Como pode ser observado pela figura 8, na grade pós-sináptica os neurônios são interligados aos neurônios mais próximos e os neurônios de entrada são ligados com todos os neurônios da grade.

O mapa auto-organizável, como descrito por (LIPPMANN, 1987) é treinado utilizando uma aprendizagem híbrida composta por aprendizagem Hebbiana e competitiva, onde os neurônios da grade pós-sináptica competem entre si, com base em uma medida de similaridade com o sinal de entrada, o neurônio mais similar é dito ser o vencedor. O neurônio vencedor por sua vez excita os neurônios próximos a ele. A distância euclidiana é geralmente utilizada como medida de similaridade.

O algoritmo responsável pela formação do mapa começa pela atribuição de valores iniciais dos pesos sinápticos da grade, que segundo (HAYKIN, 1999) deve ser feito

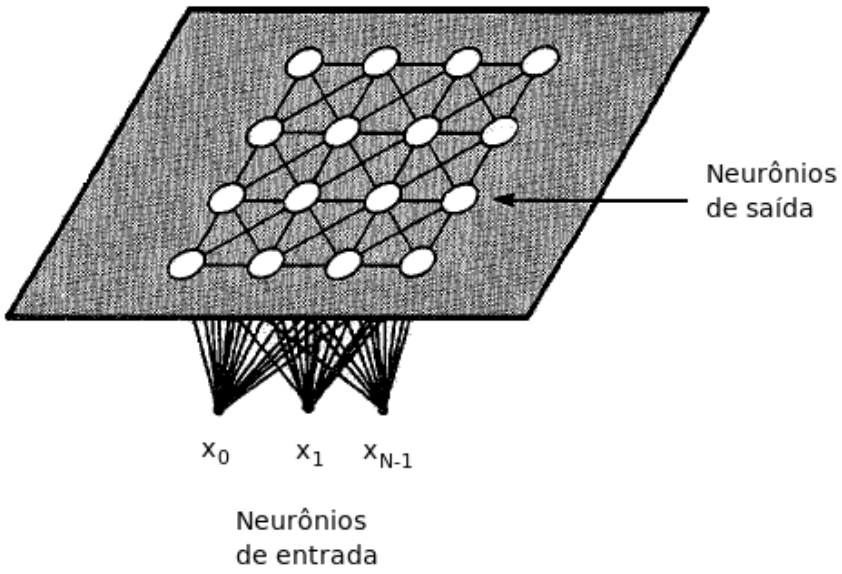

Figura 8: Disposição dos neurônios em uma rede de Kohonen. [Adaptado de (LIPPMANN, 1987)]

atribuindo valores pequenos gerados aleatoriamente.

Após esta etapa, Haykin (1999) identifica três processos essenciais envolvidos na formação do mapa auto-organizável:

Competição Para cada padrão de entrada, os neurônios da grade competem entre si, calculando a distância euclidiana entre seus pesos sinápticos e os valores do padrão de entrada, sendo o neurônio com a menor distância o vencedor (Aprendizagem Competitiva);

Cooperação O neurônio vencedor determina uma área, vizinhança topológica, na qual os neurônios cooperarão entre si (Aprendizagem Hebbiana);

Adaptação Sináptica Determinada a vizinhança, os neurônios cooperam de modo a atualizarem seus pesos sinápticos, sendo que os mais próximos do neurônio vencedor sofrem modificações mais significativas do que os neurônios mais distantes.

A função Gaussiana é utilizada para definir como será feita a atualização dos pesos dos neurônios na vizinhança, neste caso o neurônio vencedor está localizado no centro da vizinhança. A Eq. (2.11) mostra o cálculo da excitação de um neurônio j, sendo d a distância entre o neurônio j e o neurônio vencedor i. A distância euclidiana, definida por $D(x, y) = \sqrt{x^2 - y^2}$, é comumente utilizada.

$$h_{ji} = exp\left(-\frac{d_{ji}^2}{2\sigma^2}\right) \tag{2.11}$$

onde σ é a "largura efetiva" da vizinhança topológica. Ela mede o grau com o qual neurônios excitados na vizinhança participam do processo de aprendizagem (HAYKIN, 1999).

O autor supracitado ainda afirma que em um sentido qualitativo a vizinhança topológica gaussiana é mais biologicamente apropriada do que uma vizinhança retangular.

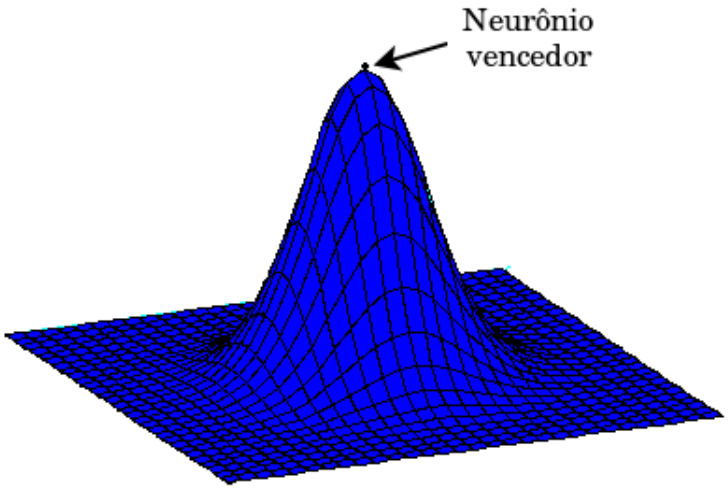

Figura 9: Vizinhança gaussiana

Com o decorrer da execução do algoritmo, os padrões de entrada similares serão mapeados em neurônios topologicamente próximos, formando áreas ou *clusters*.

O SOM é aplicado à clusterização de dados, ou seja, agrupamento de dados intrinsecamente semelhantes, utilizado quando nada é conhecido sobre os dados. Os mapas de Kohonen podem ser aplicados em compressão de dados, uma vez que dados de alta dimensionalidade são mapeados em um espaço de baixa dimensão, preservando seu conteúdo (BASHEER; HAJMEER, 2000).

2.4.3 Rede de Hopfield

A rede de Hopfield pode ser vista como uma rede totalmente conectada agindo como uma memória associativa, capaz de armazenar padrões (HOPFIELD, 1984). De acordo com (KOVáCS, 1996) uma memória associativa serve para armazenar um conjunto de vetores, de tal forma que se for endereçada com um vetor arbitrário *y*, retornará como saída aquele vetor mais próximo em algum sentido pré-definido. Geralmente é utilizada a distância de

Hamming, como forma de mensurar a proximidade entre os vetores. A distância de Hamming entre dois vetores (*strings*) de mesmo tamanho é definido pelo número de posições que os símbolos correspondentes são diferentes (LIPPMANN, 1987).

Observando a definição é possível concluir que uma memória associativa pode ser interpretada como um classificador de padrões, onde as classes são representadas pelos vetores armazenados.

O modelo proposto por (HOPFIELD, 1982) é a implementação de uma memória associativa por uma rede recorrente, porém não há auto-relimentação. Como identificado na figura 10.

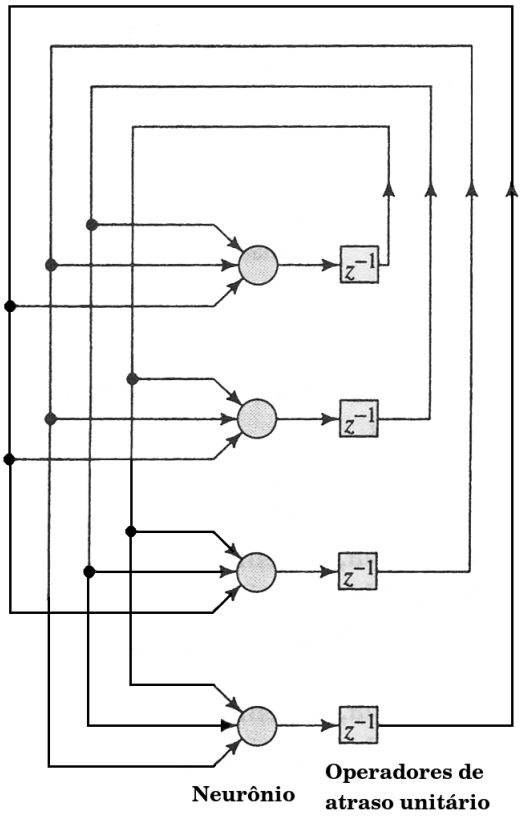

Figura 10: Memória Associativa recorrente de Hopfield. [Fonte: (HAYKIN, 1999)]

Neste modelo os vetores de entrada são normalmente binários. A rede é iniciada por um vetor $x(0)$ e gera $y(0)$ como saída, sendo $y(0)$ reaplicado como entrada da rede no ciclo posterior. Esse processo é realizado até que a rede se estabilize, ou seja, $y(k+1) \cong y(k)$, este processo é referido como *dinâmica da rede* (KOVáCS, 1996). O estado no qual a rede se estabilizou é dito ser um mínimo local do problema.

Com o intuito de análise o processo de evolução da rede, uma função de

energia ou de *Lyapunov* é utilizada. De acordo com (KOVáCS, 1996) a função de energia da rede submetida a sua própria dinâmica, só pode decrescer ou permanecer constante, ou seja, no decorrer da execução a rede tenta ir para estados melhores ou no mínimo iguais ao inicial.

Como observado por (LIPPMANN, 1987), a rede de Hopfield é mais apropriada quando uma exata representação binária é possível e menos apropriada quando os valores de entrada são contínuos.

Os principais focos de aplicação são: classificação e reconhecimento de padrões, quando usada como uma memória associativa; e também em otimização (HOPFIELD; TANK, 1986), quando aplicada considerando a função de energia.

2.5 Aplicações

Com o grande número distinto de redes neurais presentes na literatura e considerando a diversidade de tipos de aprendizagem e topologias, as redes neurais de um modo geral são aplicadas em grande parte das áreas do conhecimento. Basheer e Hajmeer (2000) destacaram sete categorias, as quais as RNA's vem sendo aplicadas com sucesso.

1. Classificação de Padrões

Dentre as categorias aqui apresentadas, a classificação de padrões talvez seja a categoria com maior aplicação das RNA's. Basicamente o problema consiste em classificar padrões de entrada entre classes previamente conhecidas, tomando como base de conhecimento algumas características do problema aplicado. As redes MLP e RBF são comumente aplicadas a esta classe de problemas.

2. Clusterização

A clusterização pode ser vista como um problema de classificação de padrões, quando é desconhecido previamente quantas e quais são as classes a ser classificadas. A este tipo de problema aplicam-se redes que utilizem aprendizagem não-supervisionada, podendo destacar a Rede de Kohonen.

3. Aproximação de Função

Quando é desejada a aproximação de funções complexas e com alta não-

linearidade, algoritmos exatos nem sempre obtém boas aproximações, com isso as redes neurais surgem como uma alternativa para esse problema. As redes MLP são conhecidas como aproximadores universais, podendo aproximar qualquer função em qualquer nível de precisão. Destaca-se também as redes RBF em que seu processo de aprendizagem nada mais é do que a interpolação dos dados de entrada em uma superfície multidimensional.

4. Previsão

Com base em um histórico de cenários e as devidas ações tomadas é possível prever qual será a ação a ser executada em um novo cenário. As redes MLP e RBF são aplicadas a esta categoria de problemas.

5. Otimização

Problemas de otimização consistem em encontrar o valor máximo ou mínimo a partir de uma função objetivo. Mesmo sendo um campo bem estabelecido na matemática, as redes neurais principalmente as redes de Hopfield foram aplicadas com sucesso em problemas de otimização complexos e não-lineares.

6. Associação

A associação consiste no desenvolvimento de uma rede associadora de padrões, a qual foi treinada utilizando dados com ruídos (*noise*) e posteriormente é aplicada a classificação de padrões sem ruídos. Também pode ser aplicado na reconstrução de dados corrompidos ou ausentes. Redes de Hopfield são geralmente utilizadas nesta aplicação (LIPPMANN, 1987).

7. Controle

Aplicação de redes neurais para auxiliar um sistema de controle adaptativo. O objetivo é gerar uma entrada de controle, tal que o sistema siga a trajetória determinada pelo modelo de referência (JAIN; MAO; MOHIUDDIN, 1996).

2.6 Conclusão

As redes neurais são modelos pertencentes ao paradigma de aprendizagem conexionista indutivo, na qual um conjunto de unidades simples de processamento (neurônios), são fortemente conectados, formando uma complexa e altamente robusta ferramenta de aquisição do conhecimento.

O conhecimento é armazenado na forma de pesos das conexões entre os neurônios, por pesos entende-se como nível de influência de um neurônio no neurônio seguinte, nível este que pode ser negativo (inibitório) ou positivo (excitatório).

A grande aplicabilidade das redes neurais é dada pelo fato da diversidade de modelos presentes na literatura, com diferentes modos, paradigmas e formas de aprendizagem. Destaca-se a rede *Multilayer Perceptron*, denominada por (BASHEER; HAJMEER, 2000) como o *"workhorse"*das redes neurais, sendo este um dos modelos mais largamente utilizado (BASHEER; HAJMEER, 2000).

Algumas de suas características principais são: (i) a habilidade de reconhecer e aprender as relações fundamentais entre as entradas e a saída, sem levar em consideração a dimensionalidade dos dados e a não-linearidade do sistema; (ii) a alta tolerância para dados ruidosos (*noise data*).

Redes neurais possuem suas limitações, destacando: (i) sucesso depende da qualidade e quantidade de dados disponível para treinamento; (ii) ausência de regras claras e procedimentos efetivos para determinação da arquitetura ótima; (iii) a incapacidade de explicar de forma compreensível como são obtidas as repostas, dado um conjunto novo de entrada, característica esta que originou o termo *black boxes*. Termo este utilizado pelo não conhecimento do funcionamento interno de uma rede neural.

A fim de suprimir estas deficiências, várias abordagens foram propostas na literatura. Em conjunto com suas características positivas, pode-se concluir que as redes neurais constituem um dos mais importantes e eficazes métodos de aprendizagem de máquina presentes na literatura até então.

3 Algoritmo Genético

Charles Darwin afirmava que o mecanismo de evolução é uma competição que seleciona os indivíduos mais bem adaptados em seu ambiente podendo assegurar descendentes, transmitindo as características que permitiram sua sobrevivência (DRÉO et al., 2005).

Com base na teoria Darwiniana, pesquisadores tentaram imitar tal mecanismo de evolução. Pesquisas essas que levaram a criação de vários métodos de busca e otimização em problemas complexos, os quais são intratáveis por métodos clássicos. O Algoritmo Genético (AG) pertencente a essa classe de métodos embasados na teoria de evolução das espécies é o método mais difundido na comunidade científica e também como ferramenta para resolução de diversos problemas da engenharia, matemática, computação, entre outros.

Este capítulo apresenta os principais conceitos de Algoritmos Genéticos, especificando suas etapas de execução, detalhes de implementação além da descrição de aspectos teóricos.

3.1 Computação Evolucionária

A Computação Evolucionária (CE) é uma sub-área da Inteligência Artificial, na qual diversos "algoritmos evolucionários" estão presentes. Algoritmos que possuem como inspiração a teoria da evolução das espécies descrita por Charles Darwin em 1859 no famoso livro "A Origem das Espécies", teoria essa que introduziu o conceito de *seleção natural*.

Os algoritmos evolucionários são constituídos de quatro elementos básicos:

1. Uma população de indivíduos, onde cada indivíduo corresponde a uma solução candidata do sistema;
2. Uma maneira de criar novos indivíduos, a partir de indivíduos já existentes;
3. Uma forma de medir a qualidade da solução que cada indivíduo corresponde;

4. Um método de selecionar os melhores indivíduos, aplicando assim o princípio da seleção natural.

Os algoritmos evolucionários que pertencem a CE são listados abaixo.

- Algoritmo Genético;
- Programação Genética;
- Sistemas Classificadores;
- Estratégias Evolutivas;
- Programação Evolucionária.

Mesmo com algumas diferenças, estes algoritmos possuem a mesma idéia geral, trabalham com uma população de indivíduos, os quais representam possíveis soluções do problema. Esses indivíduos passam por algumas modificações genéticas e os mais aptos são selecionados para gerar descendentes, criando uma nova população de indivíduos provavelmente mais aptos. Este processo se repete até que encontre um indivíduo que represente a solução ótima ou próxima dela.

3.2 Algoritmo Genético

O Algoritmo Genético (AG) constitui um modelo matemático que simula a teoria da evolução Darwiniana, no qual existe inicialmente um conjunto de indivíduos (população inicial), que representam possíveis soluções para um determinado problema e a cada iteração, chamada de *geração* em AG. Os indivíduos são avaliados em relação ao seu nível de adaptabilidade com o meio externo e os mais aptos são selecionados para gerar descendentes.

Uma nova população é gerada através da aplicação de operadores genéticos, como recombinação e mutação de genes. Este processo é realizado até um determinado número de gerações e o indivíduo mais apto encontrado é dito ser a solução do problema (LOPES, 2006).

O algoritmo genético foi apresentado inicialmente por John Holland em seu trabalho intitulado de *"Adaptation in Natural and Artificial Systems"* em 1975, com o objetivo de formalizar matematicamente e explicar os processos de adaptação de processos naturais

e desenvolver sistemas artificiais que mantenham os mecanismos originais encontrados em sistemas naturais (IYODA, 2000).

Em AG os indivíduos são representados por cromossomos e cada símbolo do cromossomo é chamado de *gene*. Os genes por sua vez armazenam informações, os *alelos*.

Os cromossomos são geralmente implementados como uma cadeia de *bits*, comumente utiliza-se um vetor como estrutura de armazenamento. O nível de adaptabilidade do indivíduo em relação ao meio é calculado com base na função objetivo, que nos casos mais simples é própria função que se quer maximizar. Em AG esta função é denominada função de *fitness*.

Seja $P(t)$ a população de indivíduos na geração t, o algoritmo 2 apresenta o pseudo-código do AG canônico descrito em (MICHALEWICZ, 1996).

Algoritmo 2: Algoritmo Genético Canônico

```
1  begin
2      t ← 0;
3      inicia P(t);
4      avaliar P(t);
5      while não condição de parada do
6          t ← t + 1;
7          selecione P(t) a partir de P(t − 1);
8          altere P(t);
9          avalie P(t);
10     end
11 end
```

O primeiro passo de um AG é a geração da população inicial, que é formada por um conjunto de cromossomos que representam possíveis soluções de um determinado problema, chamadas de *soluções-candidatas* (LACERDA; CARVALHO, 1999). Este processo é realizado de maneira aleatória na maioria das aplicações.

Durante o processo evolutivo os melhores indivíduos são selecionados com base em seu valor de *fitness*, ou seja, quanto maior o nível de adaptabilidade de um indivíduo em relação ao ambiente, maiores serão suas chances de sobreviver e gerar descendentes.

No decorrer da execução do AG, os cromossomos podem sofrer trocas de genes (*crossover*) e mutações, com o intuito de explorar regiões desconhecidas no espaço de

busca. Este processo é repetido até que uma solução satisfatória seja encontrada (LACERDA; CARVALHO, 1999).

De acordo com (LACERDA; CARVALHO, 1999) alguns dos principais critérios de parada são:

1. utilização de um número máximo de gerações;
2. obtenção do valor ótimo da função objetivo, se for conhecido;
3. quando não houver mais melhorias significativas no cromossomo de maior aptidão, ou seja, o sistema convergiu.

Segundo (TANOMARU, 1995) o algoritmo genético não afirma que as soluções sejam ótimas, mas os processos naturais, principalmente relacionados aos seres vivos são soberbamente adequados ao nosso mundo.

3.2.1 Codificação de Indivíduos

O processo de codificação de indivíduos é onde ocorre a representação de cada possível solução, de um problema qualquer, como uma seqüência de símbolos gerados a partir de um alfabeto finito (TANOMARU, 1995).

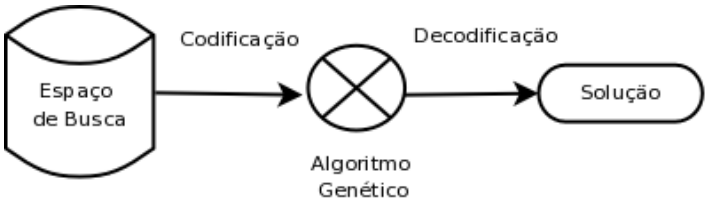

Figura 11: Processo de codificação e apresentação dos dados ao AG

Holland (1975) inicialmente utilizou a codificação binária e afirma que qualquer problema pode ser satisfatoriamente representado pela codificação binária.

Além da codificação binária, existem outras codificações que dependendo do problema, obtêm resultados melhores utilizando tais codificações. Como a codificação inteira, no qual os genes são representados por número inteiros, codificação de *string*, onde os cromossomos são letras do alfabeto, entre outras.

Para cada tipo de codificação existem operadores de *crossover* e mutação específicos, os quais serão apresentados na seção 3.2.3.1 e 3.2.3.2.

Codificação Binária

Em grande parte dos problemas práticos tratados com AG é utilizada a codificação binária (LOPES, 2006). Isto se deve ao fato de que geralmente as variáveis do problema trabalhado podem ser representadas em números binários.

De acordo com (TANOMARU, 1995) um indivíduo x, em uma codificação binária utilizando a notação binária posicional, é representado por uma seqüência $s = [b_n....b_2b_1]$, onde n é o número de bits necessários para representar x e cada $b_i \in$ 0,1. Por exemplo, se o espaço de busca é o intervalo [0,100], um vetor binário de 7 posições é necessário para representar esta variável.

Para a utilização do alfabeto binário é necessária uma discretização das variáveis, ou seja, um mapeamento entre o intervalo real da variável, [x_{min},x_{max}] em um intervalo binário [0,2^n], onde n é o número de bits necessários para representar o maior número no intervalo real (LOPES, 2006).

Segundo Michalewicz (1996) quando a codificação binária é utilizada para representar valores reais, o tamanho do cromossomo dependerá da precisão requerida. Sendo t o tamanho do intervalo real da variável e p a precisão requerida. O tamanho do cromossomo necessário para representar esta variável é calculado conforme a Eq. (3.1).

$$n = \lfloor log_2(t * 10^p) \rfloor + 1 \tag{3.1}$$

Por exemplo, se o intervalo de busca é [1,3], temos que o tamanho do intervalo é igual a 2 ($t = 3 - 1 = 2$), se utilizarmos uma precisão de 5 casas decimais ($p = 5$), seria necessário $\lfloor log_2(2 * 10^5) \rfloor + 1 = 18$ bits. Valores inteiros podem ser mapeados da mesma maneira, utilizando $p = 0$.

Com o valor do tamanho do cromossomo obtido, é necessário mapear o intervalo real da variável no intervalo binário. Este processo é realizado em duas etapas: inicialmente aplicamos a função de mapeamento sobre o valor real da variável e posteriormente convertemos o valor obtido para base binária. Sendo x o valor real da variável e $\hat{x}$ o valor mapeado, a função de mapeamento é descrita pela Eq. (3.2).

$$\hat{x} = \frac{(x - x_{min}) \cdot (2^n - 1)}{t} \tag{3.2}$$

Assim o cromossomo é representado pela cadeia de bits obtida com a conversão de $\hat{x}$ para base binária.

Já o processo de decodificação é realizado em duas etapas: inicialmente converte-se o cromossomo da base 2 para base 10 de acordo com 3.3, na qual o resultado é um valor discreto $\hat{x} \in \mathbf{N} \mid 0 \leq x \leq 2^n - 1$.

$$s = [b_n b_{n-1} .... b_2 b_1 b_0] = \sum_{i=0}^{n} b_i \cdot 2^i = \hat{x} \tag{3.3}$$

Na etapa seguinte $\hat{x}$ é mapeado de volta no espaço de busca, de acordo com 3.4, onde x_{min} e x_{max} indicam os limites do intervalo de busca.

$$x = x_{min} + \left(\frac{x_{max} - x_{min}}{2^n - 1} \right) \cdot \hat{x} \tag{3.4}$$

Codificação Real

Segundo (MICHALEWICZ, 1996), a codificação binária possui algumas desvantagens quando é aplicada à problemas de alta dimensionalidade e com alta precisão numérica. Isto ocorre pelo fato de quando há um grande número de variáveis, obtêm-se longas cadeias de bits que podem fazer o algoritmo convergir vagarosamente (LACERDA; CARVALHO, 1999). Por exemplo, para 100 variáveis com domínio no intervalo [-500,500], com precisão de 6 casas decimais, a cadeia de bits possuirá um tamanho igual a 30, gerando um espaço de busca igual a 10^{1000}, o que é intratável na prática. Um AG utilizando codificação binária obtém um pobre desempenho para este tipo de problema (MICHALEWICZ, 1996).

Na codificação real, o cromossomo é representado por um vetor de números reais, no qual cada posição do vetor (gene) contém o valor de uma variável do problema. A figura 12 mostra um cromossomo utilizando codificação real.

1.029	0.215	0.984	0.128	1.854	1.024

Figura 12: Exemplo de cromossomo na codificação real

Os cromossomos gerados são menores, em relação a codificação binária e também esta codificação é de melhor compreensão para o ser humano (LACERDA; CARVALHO,

1999).

3.2.2 Seleção

De modo a escolher os melhores indivíduos da população, emulando o processo de seleção natural, um método de seleção é utilizado.

Existem vários métodos de seleção presentes na literatura, porém não há um senso comum em relação a qual método é melhor para determinado tipo de problema, isto ainda é uma questão aberta em AG (MITCHELL, 1996).

Holland (1975) inicialmente utilizou um método de seleção baseado na proporção do valor de *fitness* do indivíduo em relação a soma do *fitness* de toda da população (MITCHELL, 1996). Sendo assim a probabilidade de um indivíduo i, ser escolhido é dado por

$$p_i = \frac{f_i}{\sum_{k=1}^{N} f_k} \tag{3.5}$$

onde f_i é o valor do *fitness* do indivíduo i e N o número de indivíduos na população.

De acordo com (TANOMARU, 1995) se não houvesse o processo de seleção, além do AG perder a grande parte do caráter evolutivo, o mesmo seria um processo ineficiente similar a uma busca aleatória.

Método da Roleta

O método mais utilizado e também o mais simples, é conhecido como método da roleta (*roulette wheel*). Nesta abordagem utiliza-se uma roleta fictícia, na qual cada cavidade ("casa") da roleta representa um indivíduo, sendo a área da cavidade proporcional ao valor do *fitness* do indivíduo (TANOMARU, 1995). Dessa forma indivíduos com maior *fitness* têm maior chance de ser escolhido para gerar descendentes. Exemplo de uma roleta fictícia é mostrada na figura 13, onde os indivíduos com maior valor de *fitness* possui uma cavidade maior.

O método da roleta apresentado por (LACERDA; CARVALHO, 1999) pode ser

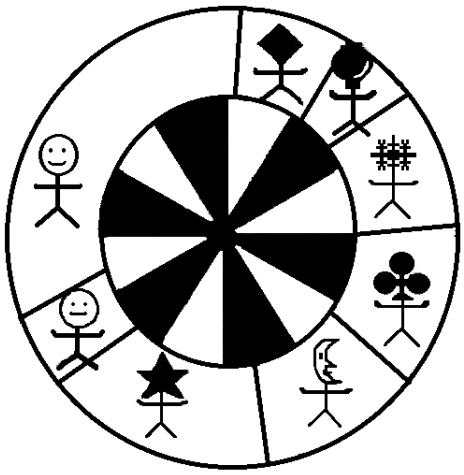

Figura 13: Exemplo de roleta de seleção

visto pelo algoritmo 3.

Algoritmo 3: Algoritmo da Roleta

```
1  begin
2      total ← ∑_{i=1}^{N} f_i;
3      rand = randomico(0,total);
4      total parcial ← 0;
5      i ← 0;
6      repeat
7          i ← i + 1;
8          total parcial ← total parcial + f_i;
9      until total parcial ≥ rand ;
10     return indivíduo s_i
11 end
```

Segundo (LOPES, 2006) o método da roleta é muito "agressivo", pois discrimina indivíduos de melhor *fitness* em detrimento de indivíduos com menor valor, ocasionando uma convergência em poucas gerações e provavelmente para um máximo local. Métodos de ordenamento, em especial o linear reduzem este tipo de problema, o método baseado em *rank* é um exemplo desta classe.

Método baseado em Rank

Inicialmente os indivíduos são ordenados por seus valores de *fitness*, objeti-

vando determinar as probabilidades de seleção, as quais podem ser determinadas por mapeamentos lineares ou não-lineares. Mapeamentos estes que modificam os valores dos *fitness* antes da seleção afim de aumentar a pressão seletiva, a pressão seletiva é discutida na seção 3.2.5.

Seleção por torneio

Neste método são escolhidos n cromossomos aleatoriamente, com probabilidade de escolha iguais e cromossomo com maior *fitness* é selecionado para gerar descendentes. Segundo (LOPES, 2006; LACERDA; CARVALHO, 1999) valores de $n = 2$ ou $n = 3$ são comumente utilizados.

Elitismo

Operadores de *crossover* e mutação são essenciais para aperfeiçoamento dos cromossomos e a exploração de outras regiões no espaço de busca. Mas vale ressaltar que os melhores cromossomos podem ser perdidos de uma geração para outra com a aplicação destes operadores. Para que seja possível a preservação dos melhores indivíduos, uma estratégia de mantê-los na população foi criada, a qual é denominada de *elitismo*. Nesta abordagem os melhores cromossomos são transferidos para a próxima geração sem alterações.

O desempenho do AG com elitismo e sem elitismo é mostrada na figura 14.

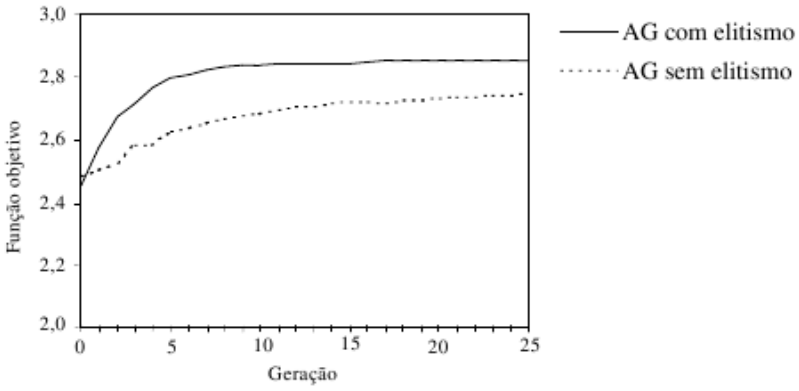

Figura 14: Desempenho do AG com e sem elitismo. [Fonte: (LACERDA; CARVALHO, 1999)]

De acordo com (TANOMARU, 1995) geralmente supervisiona e preserva apenas um indivíduos, mas em grandes populações pode ser interessante garantir uma certa percentagem dos melhores indivíduos.

3.2.3 Operadores Genéticos

Operadores genéticos são responsáveis pela geração de novos indivíduos (nova população) a partir de uma população inicial. A cada geração novos indivíduos vão emergindo, os quais, em teoria, são melhores (mais aptos) do que seus genitores. A figura 15 mostra onde os operadores são aplicados no AG.

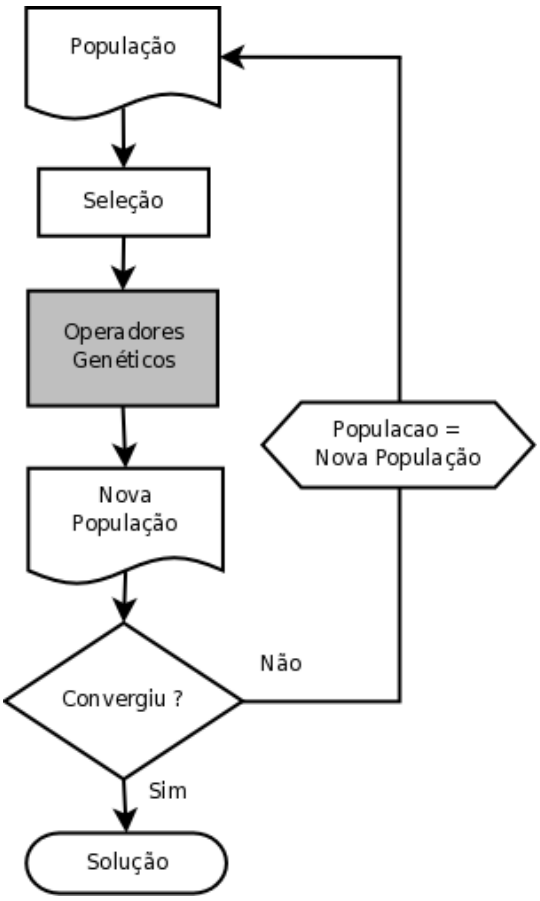

Figura 15: Aplicação dos operadores genéticos

Outro ponto a destacar é que os operadores são inseparavelmente ligados ao tipo de codificação utilizada (LOPES, 2006). Neste trabalho serão apresentados os operadores mais difundidos na literatura do AG, o *crossover* (ou recombinação) e a mutação.

Assim nos processos biológicos os fenômenos da recombinação e da mutação, podem ou não ocorrer, havendo certa probabilidade de ocorrência. Assim sendo, em AG é aplicada uma probabilidade de ocorrência a cada operador, geralmente um valor fixo, mas de acordo com (LOPES, 2006) vários autores reportam que esta probabilidade deve ser modificada ao longo da execução do AG. Valores dos parâmetros do AG são discutidos na seção 3.2.4.

3.2.3.1 Crossover

O operador genético do *crossover* cria novos indivíduos para a população através da recombinação de partes diferentes de dois cromossomos-pai escolhidos através do método de seleção (LOPES, 2006). No *crossover* a partir de dois cromossomos-pai dois novos indivíduos são gerados, os quais são chamados de *descendentes*.

Este operador é aplicado com uma determinada probabilidade de ocorrência, para cada par de cromossomos selecionados, denominada taxa de *crossover*, a qual é definida pelo usuário. Uma discussão sobre valores ideais é feita na seção 3.2.4. Caso não ocorra o *crossover* os descendentes serão iguais aos pais, permitindo a preservação de algumas soluções (LACERDA; CARVALHO, 1999).

Diferentes operadores de *crossover* tem sido apresentados na literatura, a escolha de qual aplicar é dependente do tipo de codificação escolhida. Neste trabalho são apresentados *crossover* para codificação binária e real.

Crossover para representação binária

Os operadores de *crossover* binários são simples e de fácil implementação, os mais utilizados são: *crossover* de um ponto e *crossover* de dois ou vários pontos.

Dados dois cromossomos-pai P_1 e P_2, no operador de *crossover* de um ponto, um "corte" é realizado em uma posição randomicamente escolhida de ambos os cromossomos P_1 e P_2, esta posição de corte é denominada ponto de *crossover*, dois novos indivíduos F_1 e F_2 são gerados a partir da concatenação das partes alternadas dos cromossomos-pai (LOPES, 2006), esse processo é mostrado pela figura 16.

A abordagem de dois ou vários pontos é uma extensão do *crossover* de um ponto, na qual dois ou mais pontos de *crossover* são escolhidos e os cortes são então realizados. A figura 17 mostra o operador *crossover* de 2 pontos.

De acordo com (MITCHELL, 1996) a escolha de qual operador de *crossover* utilizar depende da função de *fitness*, codificação utilizada e outros detalhes do AG, não havendo uma definição, o autor supracitado ainda afirma que normalmente a escolha é realizada de maneira empírica.

Crossover para representação real

Para (OBITKO, 1998) todos os operadores da representação binária podem ser aqui aplicados. (MICHALEWICZ, 1996) destaca três operadores, os quais são definidos a

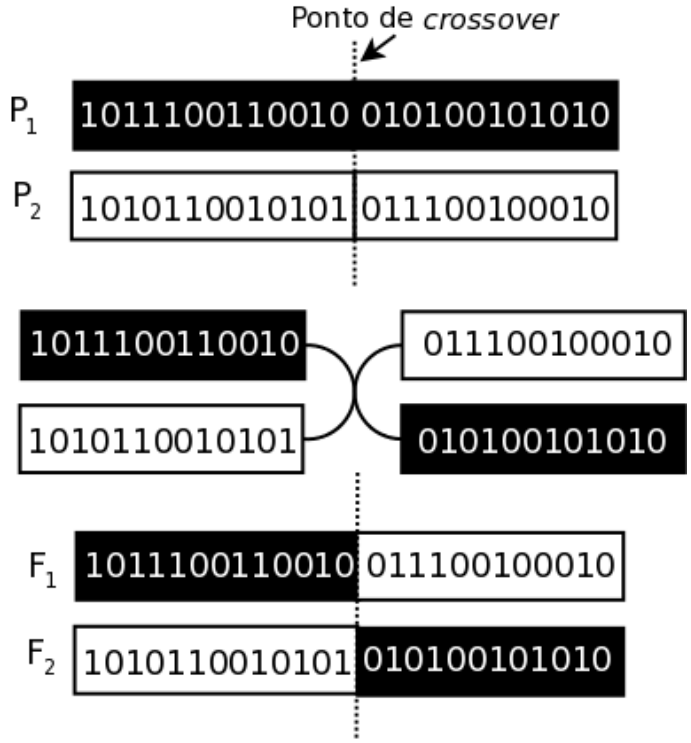

Figura 16: Operador de *crossover* de 1 ponto para codificação binária

seguir.

- **Crossover Simples** Este operador é a extensão da versão binária do *crossover* de um ponto, para a representação real.

- **Crossover Aritmético** Neste operador os cromossomos descendentes são gerados a partir de uma combinação linear dos cromossomos-pai (P_1 e P_2). A combinação linear que gera os cromossomos filhos F_1 e F_2 é dada pelas Eqs. 3.6 e 3.7 respectivamente.

$$F_1 = \alpha \cdot P_1 + (1 - \alpha) \cdot P_2 \tag{3.6}$$

$$F_2 = \alpha \cdot P_2 + (1 - \alpha) \cdot P_1 \tag{3.7}$$

  Neste operador o valor de α é aleatoriamente obtido dentro do intervalo [0,1].

- **Crossover Heurístico**

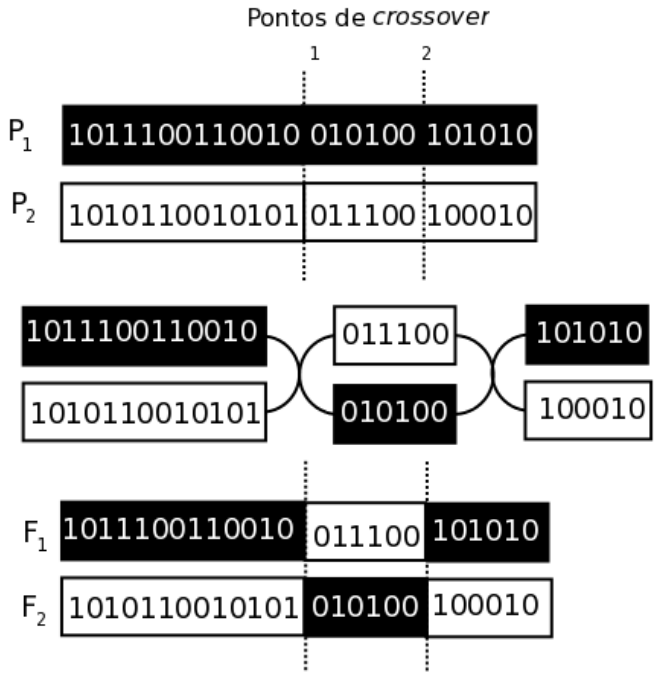

Figura 17: Operador de *crossover* de 2 pontos para codificação binária

Neste operador, a partir de dois cromossomos-pai P_1 e P_2 um ou nenhum cromossomo filho é gerado. O cromossomo-filho (F_1) é gerado de acordo com a seguinte regra:

$$F_1 = r \cdot (P_2 - P_1) + P_2 \tag{3.8}$$

desde que P_2 tenha um melhor valor de *fitness* do que P_1 e r é um número aleatório dentro do intervalo [0,1]. (MICHALEWICZ, 1996) afirma que este operador contribui para um melhor refinamento local e busca em direções mais promissoras.

Muitos outros operadores podem ser encontrados na literatura, vários destes podem ser encontrados em (LACERDA; CARVALHO, 1999).

3.2.3.2 Mutação

O operador de mutação é responsável pela exploração global do espaço de busca introduzindo novo material genético em indivíduos já existentes (LOPES, 2006). (IYODA, 2000) observa que a idéia intuitiva por trás do operador de mutação é criar uma variabilidade extra na população, mas sem destruir o progresso já obtido com a busca. Para que a mutação não prejudique os resultados até então obtidos, uma pequena taxa de mutação é geralmente aplicada, valores dos parâmetros são discutidos na seção 3.2.4.

Mutação para representação binária

O operador de mutação mais simples e mais comumente utilizado apenas seleciona randomicamente um gene do cromossomo e inverte seu valor, como mostrado na figura 18.

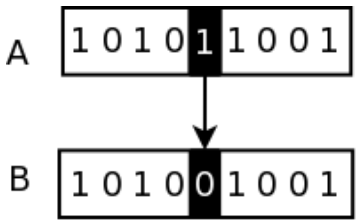

Figura 18: Operador de mutação para codificação binária

Uma variação é a seleção e inversão dos valores de vários genes do cromossomo (OBITKO, 1998).

Mutação para representação Real

De acordo com (IYODA, 2000) os operadores mais utilizados são o uniforme e o gaussiano, mas havendo uma gama de outros operadores na literatura.

- **Mutação Uniforme**

  Apresentado inicialmente por (MICHALEWICZ, 1996), este operador aplica uma simples substituição de um gene por um número aleatório, dentro do intervalo [v_{min},v_{max}], onde v_{min} e v_{max} representam o limite do intervalo permitido para o gene descendente. O número aleatório é gerado com base na distribuição uniforme.

- **Mutação Gaussiana**

  É uma variação da mutação uniforme, onde substitui um gene por um número aleatório de uma distribuição normal.

$$c_i = \begin{cases} N(p_i, \sigma) & \text{, se } i = j \\ p_i & \text{, caso contrário} \end{cases} \tag{3.9}$$

  onde $N(p_i, \sigma)$ é a distribuição normal com média p_i e desvio padrão σ.

- **Mutação Limite**

  Nesta abordagem um gene aleatório é selecionado e substituído por um dos limites do intervalo, [v_{min},v_{max}], com igual probabilidade de escolha. Este operador é utilizado para

evitar a perda de diversidade dos filhos gerados pelo *crossover* aritmético, que tende a trazer os genes para o centro do intervalo permitido (LACERDA; CARVALHO, 1999).

3.2.4 Parâmetros do Algoritmo Genético

A escolha de bons parâmetros para o AG é de extrema importância para um bom desempenho do mesmo. Parâmetros esses como o número de indivíduos na população, as probabilidades de execução dos operadores genéticos como o *crossover* e mutação.

De acordo com (LOPES, 2006) não existe uma unanimidade entre os autores em relação à valores ideais de probabilidade e alguns processos de auto-ajuste dos parâmetros estão sendo propostos na literatura recente, objetivando não apenas uma melhor eficiência do AG, mas também reduzindo a responsabilidade do usuário em determinar os parâmetros corretos.

Mesmo sem a existência de um padrão universal, alguns autores têm sugerido algumas abordagens para adaptação dos parâmetros. Segundo (MITCHELL, 1996) as pessoas usualmente utilizam valores, os quais são conhecidos por trabalharem bem em casos semelhantes reportados na literatura.

Alguns autores realizaram uma bateria de testes a fim de identificar como a variação dos parâmetros afetam o desempenho do AG. Configurações largamente utilizadas na comunidade de AG foram obtidas por (DEJONG, 1975 apud MITCHELL, 1996), os testes foram realizados com diversas funções objetivo. Com base nos testes (DEJONG, 1975) indica que entre 50-100 indivíduos em uma população são capazes de cobrir satisfatoriamente o espaço de busca, para a probabilidade de ocorrência de *crossover* (de um ponto) a melhor taxa foi em torno de 60% já a mutação a melhor taxa foi de 0.1% por bit.

Outros autores também sugerem bons parâmetros para AG, para (LACERDA; CARVALHO, 1999) a taxa de *crossover* varia entre 60% e 90% e taxa de mutação entre 0.1% e 5%. Para (TANOMARU, 1995) uma população entre 50-200 indivíduos são suficientes para resolver a maioria dos problemas e ainda afirma que populações maiores podem ser necessárias para problemas mais complexos.

3.2.5 Pressão seletiva

Entende-se por pressão seletiva a razão entre o maior *fitness* da população e o *fitness* médio (LACERDA; CARVALHO, 1999), a qual segundo (LOPES, 2006) é uma das causas

da perda de diversidade genética.

Quando a pressão seletiva é muito baixa, ou seja, o *fitness* é praticamente igual para toda a população, o algoritmo genético apresenta um comportamento aleatório, pois no processo de seleção os indivíduos possuem praticamente a mesma probabilidade de serem escolhidos. Quando há uma alta pressão seletiva, alguns indivíduos com valor de *fitness* muito acima da média, denominados de *super-indivíduos*, terão vários descendentes nas próximas gerações, que em casos extremos, indivíduos próximos do ótimo global mas com baixo valor de *fitness* poderão ser extintos. Como conseqüência disto o AG tenderá a convergir rapidamente e provavelmente para um máximo local (TANOMARU, 1995).

Lopes (2006) observa que é desejável em gerações iniciais que a pressão seletiva seja mínima, permitindo que indivíduos com baixo valor de *fitness* tenham chances de serem escolhidos do processo seletivo. Já na fase final do AG os indivíduos tendem a ter valores de *fitness* muito próximos devido à convergência do AG, e nesta etapa é preferível que a pressão seletiva seja máxima, de modo a induzir a evolução no sentido do ótimo global.

Métodos de seleção como o Método da Roleta que utiliza o valor de *fitness* dos indivíduos como base para o processo de escolha, criam uma pressão seletiva de maneira inversa, fornecendo uma alta pressão seletiva no início e uma baixa na etapa final.

Para solucionar este tipo de problema é possível a utilização da abordagem de escalonamento, o qual realiza um remapeamento do valor de *fitness* dos indivíduos antes da seleção. Uma das diferentes técnicas de escalonamento mais utilizada é a linear proposta por (GOLDBERG, 1989), que consiste em manter o *fitness* médio, atribuir ao *fitness* mínimo zero e o valor máximo seja a média multiplicada por uma constante (valor entre 1,20 e 2,00). Com isso os valores de *fitness* da população sempre estarão dentro de um intervalo predeterminado, independente do momento evolutivo (LOPES, 2006).

3.2.6 Teorias sobre AG

Algoritmos Genéticos são aplicados em inúmeros problemas de diversas áreas do conhecimento, pelos quais comprovou-se que o AG realmente funciona. Mas porque ele funciona? Em qual embasamento teórico ele se sustenta? Algumas pesquisas foram realizadas e em 1975 John Holland apresentou a Teoria dos Esquemas.

A Teoria dos Esquemas foi elaborada sobre a codificação binária, utilizando a *string* de bits e o alfabeto binário $\{0,1\}$.

Um esquema é um modelo (*template*) de um cromossomo que representa um conjunto de cromossomos. Segundo (MICHALEWICZ, 1996) um esquema (*schema*) é construído pela introdução do símbolo *don't care* (*) no alfabeto de genes, representando que naquela posição do esquema pode ser atribuído qualquer símbolo do alfabeto. O autor supracitado afirma que um esquema representa um hiperplano ou um subconjunto do espaço de busca.

Por exemplo, o esquema [01*1] representa os cromossomos [0101] e [0111]. Fica claro que o esquema [0001] representa um único esquema, [0001], já o esquema [***] representa todos os cromossomos de tamanho 3. Em termos gerais um esquema pode representar 2^r cromossomos, onde r é o número de símbolos *don't care* presentes no esquema (IYODA, 2000).

Alguns outros conceitos foram definidos por Holland (HOLLAND, 1975), como o conceito de ordem, o comprimento e *fitness* de um esquema. A ordem $o(H)$ é a quantidade de 0's e 1's contidos no esquema. O comprimento δ(H), é a diferença entre a última posição que contém 1 ou 0 e a primeira posição que contém um 1 ou 0. Por exemplo, o esquema H = [01**1*1*] possui um comprimento δ(H) = 7-1 = 6 e ordem igual a o(H) = 4.

O *fitness* de um esquema H em uma geração t, $eval(H,t)$ é a média dos valores de *fitness* dos cromossomos representados pelo esquema H (MICHALEWICZ, 1996).

A partir destas definições é possível prever a variação do número de esquemas H entre duas gerações consecutivas. Seja $m(H,t)$ e número de cromossomos representados pelo esquema H na geração t, α a média do valor de *fitness* para toda a população e $eval(H,t)$ o *fitness* do esquema H na geração t, a Eq. (3.10) é conhecida segundo (MICHALEWICZ, 1996) como equação do crescimento reprodutivo do esquema (*reproductive schema growth equation*), que define o número esperado de cópias do esquema H para a próxima geração.

$$m(H,t+1) = \frac{\alpha}{eval(H,t)} \cdot m(H,t) \tag{3.10}$$

Com base na Eq. (3.10) é possível concluir que haverá um aumento na amostragem de esquemas H se o *fitness* de H ($eval(H,t)$) for maior do que a média dos valores de *fitness* da população (α), ou seja, haverá um crescimento se H representar bons cromossomos (LACERDA; CARVALHO, 1999). E ainda, os operadores de *crossover* e mutação não afetaram, de maneira destrutiva, o esquemas curtos e de baixa ordem (IYODA, 2000). Como conseqüência desta Eq. (3.10) foi formulado o Teorema dos Esquemas.

Teorema 3.2.1 *(Teorema dos Esquemas) Esquemas curtos, com baixa ordem e com fitness*

acima da média da população, aumentam exponencialmente sua participação em gerações subseqüentes.

Os esquemas bons e curtos recebem o nome de *blocos construtivos*. As informações destes blocos serão combinadas com outros blocos bons e curtos no decorrer das gerações e esta combinação obterá indivíduos de alta aptidão (LACERDA; CARVALHO, 1999). Esta hipótese é conhecida como *Hipótese dos Blocos Construtivos*.

Hipótese 3.2.1 *(Hipótese dos Blocos Construtivos) Um algoritmo genético apresenta um desempenho quase-ótimo através da justaposição de esquemas curtos, de baixa ordem e de alto desempenho chamados de blocos construtivos.*

Em alguns problemas a Hipótese dos Blocos Construtivos não é confirmada, nestes casos dois blocos construtivos combinados resultam em um cromossomo ruim. Este fenômeno é conhecido como decepção (*deception*) (MICHALEWICZ, 1996).

De acordo com (LACERDA; CARVALHO, 1999) este tipo de fenômeno ocorre com cromossomos com alta *epistasia*. Em AG entende-se por epistasia como o nível de interação entre genes de um cromossomo, quando o valor de um gene influencia o valor de outros genes. Problemas com esta característica são raramente encontrados na prática.

Iyoda (2000) afirma que a hipótese dos blocos construtivos não fornece uma explicação do motivo pelo qual o AG funciona, apenas indica os motivos pelos quais ele funciona para uma determinada classe de problemas.

Com o intuito de contornar este problema algumas alternativas foram sugeridas, como apresenta (MICHALEWICZ, 1996). Entre elas, sugere-se, caso haja um conhecimento prévio sobre a função objetivo realizar uma codificação apropriada, de modo a estimular a criação de blocos construtivos.

3.3 Aplicações

Algoritmo Genético é basicamente aplicado em problemas de otimização, estendendo a outros problemas apenas modelando o mesmo como um problema de otimização.

Aplicações em problemas de Processamento de Imagem foram investigados por (CENTENO et al., 2005) e (RAHNAMAYAN; TIZHOOSH; SALAMA, 2005). AG's foram aplicados à problemas de Engenharia de software por (DAI et al., 2003) e (WEGENER et al., 1997). Já

(NODA; FREITAS; LOPES, 2000) e (FIDELIS; LOPES; FREITAS, 2000) utilizaram AG para mineração de dados.

Outras aplicações de AG foram propostas na literatura, algumas com sucesso menor comparado a outras técnicas, mas em outras obteve sucesso, mostrando que AG é fortemente indicado em problemas reais. AG são também combinados com outras técnicas como redes neurais, formando sistemas híbridos altamente robustos.

3.4 Conclusão

Neste capítulo foram apresentados vários conceitos da Computação Evolucionária, área essa que tem despertado grande interesse na comunidade científica devido sua grande aplicabilidade em problemas reais. Um foco maior foi dado nos Algoritmos Genéticos, descrevendo sua estrutura básica, suas etapas intermediárias, foram identificadas suas fragilidades e alternativas de contornar estas dificuldades, além da explanação de alguns aspectos teóricos do AG.

Mesmo com o grande avanço dos algoritmos genéticos nos últimos anos, algumas questões ainda permanecem em aberto, como identifica (MICHALEWICZ, 1996). O autor apresenta algumas direções que segundo ele terão uma grande atividade e significantes resultados em um futuro breve, são elas: Fundamentação Teórica, Otimização de Funções, Representação de Indivíduos e Operadores, Sistemas Auto-Adaptativos e AG Paralelos. Essas pesquisas têm por objetivo explicar e melhorar cada vez mais esta técnica.

4 Inteligência de Enxames

Com a busca incessante por modelos inteligentes inspirados em processos do mundo real, foi observado que agentes simples os quais sozinhos são incapazes de realizar tarefas simples, quando se interagem emergem um sistema auto-organizado capaz de desenvolver tarefas de grande complexidade (LOPES, 2006), como as abelhas na construção e estruturação de uma colméia, os cupins que constroem complexos sistemas de túneis, as formigas que podem encontrar caminhos diretos quando buscam por alimentos.

Inspirados nestes processos naturais foram desenvolvidos modelos computacionais interessantes, baseados no conceito de inteligência coletiva. Estudos que levaram a criação de uma nova área na inteligência computacional, a inteligência de enxames (*swarm intelligence*).

Dentro do contexto da inteligência de enxames, uma técnica que vêm sendo aplicada com bastante sucesso em problemas de otimização, é a chamada Otimização por Colônias de Formigas, do inglês *Ant Colony Optimization* (ACO) criada por Colorni, Dorigo e Maniezzo (1992), que simula o comportamento de uma colônia de formigas em relação a certos problemas por ela encontrado, como encontrar o menor caminho entre o formigueiro e o local onde se encontra o alimento. Outra técnica bastante difundida é a Otimização por Enxame de Partículas, do inglês *Particle Swarm Optimization* (PSO), que utiliza do comportamento social de algumas espécies para resolução de problemas complexos. Ambas técnicas são abordadas neste trabalho.

4.1 Ant Colony Optimization

Em meados dos anos quarenta o entomologista francês Pierre-Paul Grassé, descreveu como "estímulo significante" a reação de algumas espécies de térmites (inseto da família do cupim). Esta reação podia agir como um novo estímulo significante para outros insetos da colônia. Grassé foi o primeiro a utilizar o termo *estigmergia* para identificar o particular

tipo de comunicação entre os agentes (térmites)(DORIGO; BIRATTARI; STUTZLE, 2006).

Dorigo, Birattari e Stutzle (2006) observa duas principais características que diferenciam a *estigmergia* de outros tipos de comunicação:

- Estigmergia é uma forma de comunicação indireta e não-simbólica, intermediada pelo ambiente: insetos trocam informações modificando seus ambientes;
- A informação estigmérgica é local, podendo ser acessada somente pelo inseto que visitar o local (ou imediatamente sua vizinhança).

Algumas espécies de formigas utilizam desta forma de comunicação, sendo feita por meio de depósito de substância no local utilizado pela formiga, substância denominada de *ferormônio*. Ao passar por uma trilha, uma formiga deposita certa quantidade de ferormônio, posteriormente outras formigas que passem pelo mesmo local, serão estimuladas a seguirem as trilhas com a maior quantidade de ferormônio.

Um experimento realizado por Deneubourg et al. (1990), visava demonstrar o processo de comunicação entre as formigas via depósito de ferormônio. Na fase inicial colocaram-se duas trilhas de tamanhos iguais entre o ninho e a fonte de alimento, como mostra a figura 19. No começo as formigas escolhiam trilhas randomicamente, algum tempo depois todas as formigas seguiam sempre a mesma trilha, devido a uma das trilhas conter uma maior quantidade de ferormônio. Deneubourg e colaboradores repetiram o experimento várias vezes e identificaram que cada trilha era escolhida em 50% dos casos.

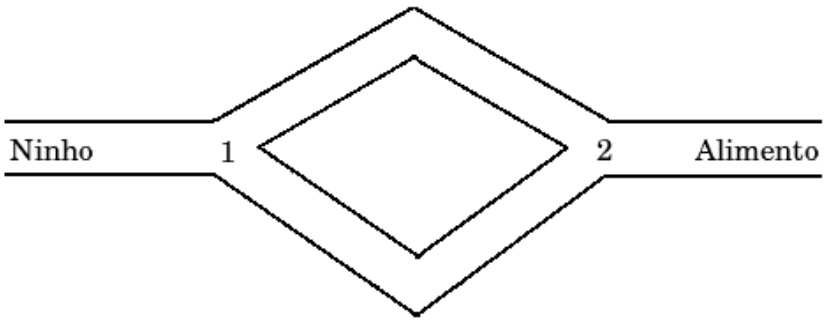

Figura 19: Experimento 1: trilhas com mesmo tamanho

Na fase final do experimento Deneubourg e colaboradores, modificaram o tamanho de uma das trilhas, fazendo-a significativamente mais longa, como mostra a figura 20.

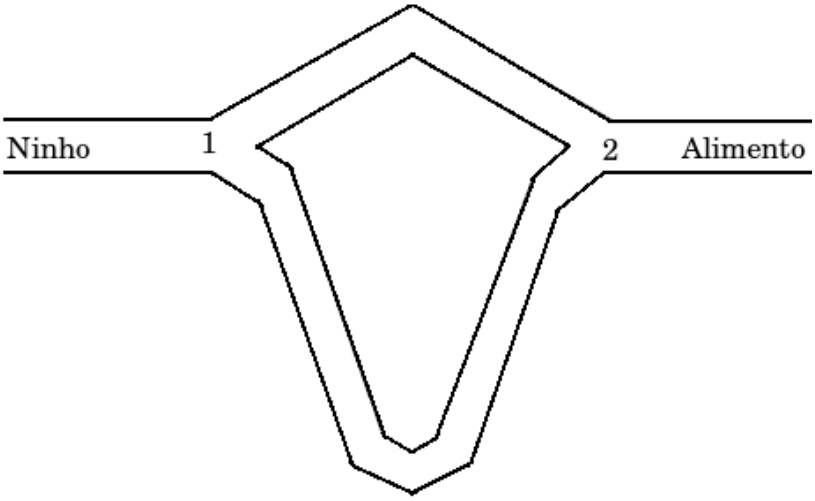

Figura 20: Experimento 2: trilhas com tamanhos diferentes

Inicialmente as formigas seguiam trilhas aleatoriamente, mas com o passar do tempo, as formigas tendiam a trilhas mais curtas. Com isso os autores identificaram uma importante regra: "as formigas escolhem por acaso a trilha que primeiro alcança a fonte de alimento", devido ao fato da trilha mais curta receber mais rapidamente o ferormônio do que trilha maior, fazendo com que as formigas que fazem o caminho de volta utilizem a trilha mais curta, essas por sua vez depositam ferormônio, aumentando a quantidade presente na trilha menor, convergindo para a situação onde todas as formigas utilizem o caminho mais curto.

Baseado nesta característica, Colorni, Dorigo e Maniezzo (1992) propuseram uma técnica denominada *Ant System*, que utiliza todos estes conceitos, para solução de problemas de otimização de rotas, abrindo caminho para um grande número de outras pesquisas nesta área, sendo este campo denominado de Otimização por Colônia de Formigas.

4.1.1 Ant System

O *Ant System* (AS) foi o primeiro algoritmo ACO proposto na literatura (COLORNI; DORIGO; MANIEZZO, 1992). É um modelo inspirado no comportamento de algumas espécies de formigas durante o processo de busca por alimento, no qual as formigas ao passar por trilhas vão deixando certa quantidade de ferormônio pelo caminho, assim as formigas que venham posteriormente, são atraídas por esta substância tendo alta probabilidade de seguir este mesmo caminho (DORIGO; BIRATTARI; STUTZLE, 2006).

É um modelo baseado em população, cada indivíduo é a representação de uma possível solução do sistema, isso permite trabalhar com diversas soluções ao mesmo

tempo. Inicialmente as formigas estão em seu ninho e com o passar do tempo elas vão percorrendo as trilhas e depositando seu ferormônio até que cheguem ao seu destino (fonte de alimento). A representação do sistema é feita utilizando um grafo direcionado, como mostra a figura 21.

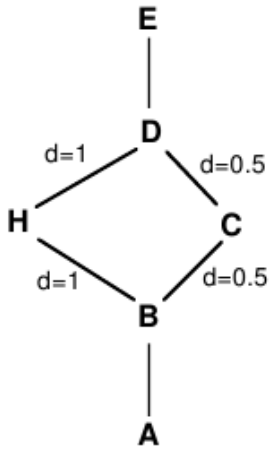

Figura 21: Representação do ambiente

Segundo (DORIGO; BIRATTARI; STUTZLE, 2006) a quantidade de ferormônio em cada trilha é atualizada em relação ao número m de formigas que utilizaram esta trilha, a fórmula de atualização da quantidade de ferormônio de uma trilha entre os pontos i e j é dada pela Eq. (4.1).

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{4.1}$$

onde ρ é a taxa de evaporação, m o número de formigas que utilizaram esta trilha e $\Delta\tau_{ij}^{k}$ é a quantidade de ferormônio deixado pela formiga k na trilha entre os pontos i e j, o qual é calculado conforme a Eq. (4.2).

$$\Delta_{ij}^{k} = \begin{cases} Q/L_k & \textit{, se a formiga k utilizou esta trilha} \\ 0 & \textit{, caso contrário} \end{cases} \tag{4.2}$$

caso a formiga k tenha utilizado esta trilha e $\Delta_{ij}^{k} = 0$ caso contrário, onde Q é uma constante e L_k é o comprimento da rota utilizado pela formiga k (DORIGO; BIRATTARI; STUTZLE, 2006).

Segundo (DORIGO; MANIEZZO; COLORNI, 1996), em uma visão mais prática, a taxa de evaporação é utilizada para evitar que o sistema tenha uma convergência rápida

para uma solução sub-ótima. A quantidade inicial de ferormônio em cada trilha ($\tau_{ij}(0)$) é obtida arbitrariamente, comumente utilizam-se valores pequenos (COLORNI; DORIGO; MANIEZZO, 1992).

Após as formigas terem passado por uma trilha, elas necessitam escolher uma nova trilha a ser seguida, a fim de atingir o objetivo final, chegar à fonte de alimento. A quantidade de ferormônio acumulado em cada trilha será decisiva para sua escolha, quanto mais ferormônio a trilha contiver maior será a probabilidade de esta trilha ser seguida, a probabilidade de uma formiga k, escolher uma trilha partindo de i para o ponto j, é dada pela Eq. (4.3)

$$p_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum\limits_{c_{il} \in N(s^p)}} & \text{, se } c_{il} \in N(s^p) \\ 0 & \text{, caso contrário} \end{cases} \tag{4.3}$$

onde N(s^p) é o conjunto de trilhas ainda não utilizadas pela formiga. Segundo (DORIGO; BIRATTARI; STUTZLE, 2006), os parâmetros α e β, controlam a relativa importância do ferormônio em relação à informação heurística, η_{ij}^{β}, a qual de acordo com (COLORNI; DORIGO; MANIEZZO, 1992) é chamado de *visibilidade*, que diz que pontos mais próximos devem ter uma maior probabilidade de serem escolhidos, a mesma é definida conforme a fórmula 4.4

$$\eta_{ij}^{\beta} = \frac{1}{d_{ij}} \tag{4.4}$$

onde d_{ij} é a distância entre os pontos i e j.

Este processo continua até que todas as formigas cheguem ao destino. Com o passar do tempo, as formigas tenderão a seguir caminhos mais curtos, devido a um maior trânsito de formigas, e conseqüentemente um maior despejo de ferormônio, por unidade de tempo. Isso causa uma maior concentração de ferormônios em caminhos mais curtos, até que todas as formigas utilizem um único caminho, o mais curto (o sistema convirja para solução ótima, ou próxima dela).

4.1.2 Outros algoritmos ACO

Diversos outros algoritmos baseados no Ant System foram propostos na literatura. Estes métodos são melhorias e extensões do *Ant System* original.

Dorigo, Birattari e Stutzle (2006) apresentam os principais algoritmos ACO propostos, destacando-se duas das variantes mais bem sucedidas do *Ant System*, o $\mathcal{MAX}$-$\mathcal{MIN}$ *Ant System* e o *Ant Colony System*.

4.1.2.1 $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System

Proposto por (UTZLE; HOOS, 1997) este algoritmo é um melhoramento sobre AS original, no qual o processo de atualização da quantidade de ferormônio presente na trilha, é realizado pela melhor formiga, não dentre todas como o AS original. Além disso, foi imposto um limite, inferior e superior, na quantidade de ferormônio em cada trilha.

Com isso a atualização do ferormônio da trilha i para j, é dado pela Eq. (4.5)

$$\tau_{ij} = \left[(1-\rho)\cdot\tau_{ij} + \Delta\tau_{ij}^{melhor}\right]_{\tau_{min}}^{\tau_{max}} \tag{4.5}$$

onde τ_{min} e τ_{max} representam o limite inferior e superior imposto para o ferormônio. O operador $[x]_b^a$ é definido pela Eq. (4.6).

$$[x]_b^a = \begin{cases} a & \text{, se } x > a, \\ b & \text{, se } x < b, \\ x & \text{, caso contrário} \end{cases} \tag{4.6}$$

A quantidade de ferormônio deixada pela formiga que utilizou o percurso com menor custo (melhor formiga), é calculado segundo a Eq. (4.7), onde L_{melhor} é o custo do percurso realizado pela melhor formiga.

$$\Delta\tau_{ij}^{melhor} = \begin{cases} \frac{1}{L_{melhor}} & \text{, se (i,j) pertence ao melhor percurso} \\ 0 & \text{, caso contrário} \end{cases} \tag{4.7}$$

De acordo com (DORIGO; BIRATTARI; STUTZLE, 2006), o valor de L_{melhor} pode ser o melhor percurso da iteração corrente (L_{mi}), o melhor percurso desde o início do algoritmo (L_{mt}) ou ainda uma combinação dos dois.

O autor supracitado ainda observa que os valores de τ_{min} e τ_{max} são obtidos de maneira empírica, e até o momento não há direção de como obter esses valores.

4.1.2.2 Ant Colony System

O algoritmo *Ant Colony System* (ACS) utiliza-se do conceito de atualização local do ferormônio (*local pheromone update*), em adição a atualização do ferormônio no final de percurso.

A atualização local do ferormônio é realizada por todas as formigas em cada passo da construção, cada formiga aplica esta atualização somente na última trilha percorrida (DORIGO; BIRATTARI; STUTZLE, 2006). A atualização local é realizada de acordo com a Eq. (4.8), onde $\varphi \in (0,1]$ é o coeficiente de depósito de ferormônio e τ_0 é a quantidade inicial de ferormônio.

$$\tau_{ij} = (1 - \varphi) \cdot \tau_{ij} + \varphi \cdot \tau_0 \tag{4.8}$$

O objetivo principal da atualização local é a produção de diferentes soluções, dada o aumento das alterações na quantidade de ferormônio nas trilhas, criando uma maior diversidade de trilhas, encorajando novas formigas a escolherem outras soluções (DORIGO; BIRATTARI; STUTZLE, 2006).

Assim como no $\mathcal{MAX}$-$\mathcal{MIN}$ *Ant System*, a atualização no final de cada percurso é realizada considerando apenas a formiga que obteve o percurso de menor custo, assim a fórmula de atualização é dada pela Eq. (4.9)

$$\tau_{ij} = \begin{cases} (1 - \rho) \cdot \tau_{ij} + \rho \cdot \Delta\tau_{ij} & \text{, se (i,j) pertence ao melhor percurso} \\ \tau_{ij} & \text{, caso contrário} \end{cases} \tag{4.9}$$

O valor de $\Delta\tau_{ij}$ é calculado da mesma forma do $\mathcal{MAX}$-$\mathcal{MIN}$ *Ant System*, conforme a Eq. (4.7), com L_{melhor} podendo ser melhor da iteração (L_{mi}) ou melhor dentre todos (L_{mt}).

Como observa (DORIGO; BIRATTARI; STUTZLE, 2006) outra importante diferença entre ACS e AS é o modo como é feito o cálculo da probabilidade de escolha dentre as trilhas possíveis de uma formiga. No ACS é utilizada uma regra denominada de *regra pseudo-aleatória proporcional*, na qual a probabilidade de escolha de uma trilha entre i e j depende de uma variável aleatória $q \in [0,1]$ uniformemente distribuída e ainda um outro parâmetro q_0. Assim uma trilha t é escolhida segundo a Eq. (4.10).

$$t = \begin{cases} \arg\max_{c_{il} \in N(s^p)} \left\{ \tau_{il} \eta_{il}^{\beta} \right\} & \text{, se } q \leq q_0 \\ \textit{Equação 4.3} & \text{, caso contrário} \end{cases} \tag{4.10}$$

4.2 Particle Swarm Optimization

A técnica denominada Otimização por Enxames de Partículas, do inglês *Particle Swarm Optimization (PSO)*, é uma abordagem estocástica, baseada em população que simula o processo comportamental de interação entre os indivíduos de um grupo (MEDEIROS, 2005).

Sua teoria é fundamentada pela observação do comportamento de grupos de animais como pássaros, peixes, entre outros, que realizam interessantes tarefas de otimização na execução de atividades simples, como a busca por alimentos (KENNEDY; EBERHART, 1995). Com isso o PSO busca a simulação destas atividades naturalmente simples, porém matematicamente complexas.

Com base nas observações comportamentais dos grupos, pode-se concluir que o comportamento do mesmo é influenciado pelos resultados de experiências obtidos por cada indivíduo e também com a experiência obtida pelo grupo (KENNEDY; EBERHART, 1995).

Nesta abordagem cada indivíduo é dito ser uma possível solução para o problema investigado, sendo atribuído a cada indivíduo um valor que está relacionado a adequação da partícula com a solução do problema, denominada de *fitness*, e também uma variável velocidade que representa a direção do movimento do indivíduo (partícula). Com o passar do tempo os indivíduos vão ajustando suas velocidades em relação a melhor solução (melhor *fitness*) encontrada por ele próprio e também pela melhor solução do grupo, realizando este processo até que os indivíduos (partículas) encontrem o alimento (melhor solução global ou próxima dela).

O valor de *fitness* é calculado com base na função objetivo que se deseja maximizar (minimizar). Segundo (LOPES, 2006) o valor de *fitness* é definido pela natureza do problema de otimização e é computada por uma função objetivo que avalia um vetor solução.

4.2.1 Simulando o Comportamento Social

Objetivando a simulação dos movimentos dos bandos de pássaros, diversos pesquisadores buscam entender como eles se mantêem aninhados e alteram suas direções repentinamente, dispersando, e novamente reagrupando sem haver nenhum tipo de colisão.

A partir destas pesquisas, alguns modelos de simulação foram surgindo, estes modelos são fundamentados na distância individual entre as aves, e que a sincronia no comportamento do bando está no esforço dos pássaros em manter uma distância ótima entre eles e seus vizinhos (KENNEDY; EBERHART, 1995).

Kennedy e Eberhart (1995) afirmam que a hipótese fundamental para o desenvolvimento do PSO, surgiu a partir de um relato do sociobiologista Edward O. Wilson, que descrevia sobre a busca por comida de um cardume de peixes. Wilson afirmava que em teoria os membros individuais do cardume poderiam obter vantagem nas descobertas e pela experiência prévia de todos os outros membros do cardume durante o processo de busca por alimento, sendo que esta vantagem poderia ser decisiva, mesmo com a desvantagem da competição pelos itens de alimento, não importando o quão imprevisível estão distribuídos no ambiente (WILSON, 1975 apud KENNEDY; EBERHART, 1995). Este relato sugere que o compartilhamento social da informação entre os indivíduos oferece uma vantagem evolucionária.

4.2.2 Modelo Matemático do Comportamento Social

Com base nas teorias sociobiológicas, Kennedy e Eberhart (1995) apresentaram um modelo matemático básico do comportamento social de indivíduos em grupo. Neste modelo os indivíduos guardam consigo informações para orientá-los durante o processo de busca, os quais são elas: a posição do indivíduo no ambiente (X_{id}^t), a qual é um vetor *n*-dimensional, dependendo da dimensão dos dados do problema; a velocidade (V_{id}^t) e a melhor posição encontrada por ele até o momento (P_{id}). Para (LOPES, 2006) o termo velocidade não tem a dimensão de espaço/tempo, mas sim a variação da posição do indivíduo no espaço de busca (ΔX_{id}^t), que por motivo de simplificação os autores definiram como *velocidade*. Além disso, guarda a melhor solução global que é definida por P_{gd}, que é utilizada no processo de atualização das velocidades dos indivíduos.

Inicialmente o vetor posição e a velocidade de cada indivíduo são iniciados de maneira randômica. Para o vetor posição os valores estão no intervalo no qual se deseja obter o valor máximo (mínimo) da função objetivo, para um boa dispersão dos indivíduos no espaço de busca, um bom gerador de números randômicos é necessário.

Segundo (KENNEDY, 1997) para que o sistema não exploda e implemente realisticamente a simulação das alterações incrementais na aprendizagem humana, um valor V_{max} é utilizado como um limite de velocidade, tanto para a geração inicial das velocidades de cada indivíduo, quanto no processo de atualização das mesmas, com isso os valores iniciais das velocidades estão no intervalo [-V_{max} , V_{max}].

De posse destes valores iniciais, o processo de avaliação e atualização das posições dos indivíduos é iniciada. Este processo é realizado de forma iterativa, até que uma condição de parada seja satisfeita. Alguns dos critérios de parada mais utilizados é a definição de um número máximo de iterações, ou quando os indivíduos não mais alterarem suas posições (LOPES, 2006).

Os valores de *fitness* são calculados para cada indivíduo, e a posição do indivíduo com melhor *fitness* é atribuída a uma variável P_{gb}, que guarda a posição da melhor solução obtida por todos os indivíduos até o momento. Com isso todos os outros indivíduos atualizam suas velocidades pela equação definida por (KENNEDY; EBERHART, 1995):

$$V_{id}^{t+1} = V_{id}^{t} + \varphi_{id}(P_{id} - X_{id}) + \varphi_{id}(P_{gd} - X_{id}) \tag{4.11}$$

O segundo termo do lado direito da equação, é descrito por (KENNEDY; EBERHART, 1995), como sendo o fator cognitivo do indivíduo e o terceiro termo o fator social. Na visão matemática o segundo termo define a distância entre a posição atual do indivíduo e a melhor posição que ele obteve até o momento, e o terceiro termo a distância entre a sua posição atual e a posição do indivíduo que obteve a melhor solução global (melhor valor de *fitness*).

Os valores de φ_{id} estão dentro do intervalo no qual se busca a solução ótima, e segundo (KENNEDY, 1997) pode ser um valor randômico, dentro do intervalo de busca.

De acordo com (LOPES, 2006) para que se possa cobrir satisfatoriamente o espaço de busca, um número de indivíduos entre 20 e 50 é suficiente.

Com as velocidades atualizadas, agora pode ser realizado o processo de atualização da posição de todos os indivíduos, conforme a Eq. (4.12) descrita por (KENNEDY; EBERHART, 1995).

$$X_{id}^{t+1} = X_{id}^{t} + V_{id}^{t+1} \tag{4.12}$$

Este processo é realizado até que algum critério de parada pré-definido seja

satisfeito. O algoritmo 4 apresenta o pseudo-código do algoritmo PSO.

Algoritmo 4: Particle Swarm Optimization

```
1  begin
2      for todos os indivíduos do
3          Inicialize a posição e velocidade randomicamente (dentro do espaço de busca);
4      end
5      while não condição de parada do
6          Calcule o valor de fitness para todos os indivíduos;
7          Obtenha a posição do indivíduo com melhor solução global;
8          Atualize a velocidade de cada indivíduo;
9          Atualize a posição de cada indivíduo;
10     end
11     Melhor solução da última iteração;
12 end
```

4.2.3 Variações do Algoritmo PSO

Existem algumas variações do algoritmo PSO canônico. Um modelo interessante utiliza o conceito de que um indivíduo inicialmente tende a imitar os indivíduos que estão mais próximos a ele. Com isso surgiu o termo "melhor vizinho", que consiste em utilizar a posição do melhor vizinho, ao invés de utilizar a melhor posição global obtida até o momento.

Este modelo foi inicialmente proposto por (KENNEDY, 1997), o qual apresentava que a vizinhança é descrita pelo equivalente à um ciclo de comunicação, ou seja, para cada indivíduo i, a vizinhança era composta pelos indivíduos, $i-1$, i e $i+1$. A figura 22 ilustra o conceito de ciclo de comunicação.

Utilizando este conceito de melhor vizinho, chamamos de P_{vd} a posição do melhor vizinho, a Eq. (4.13) apresenta como são atualizadas as velocidades dos indivíduos.

$$V_{id}^{t+1} = V_{id}^{t} + \varphi_{id}(P_{id} - X_{id}) + \varphi_{id}(P_{vd} - X_{id}) \tag{4.13}$$

O processo de atualização das posições é a mesmo apresentado pela Eq. (4.12).

Outros modelos foram apresentados por Kennedy (1997), os quais são denominados Modelo somente Cognitivo (*"Cognition-only" Model*), Modelo somente Social (*"Social-*

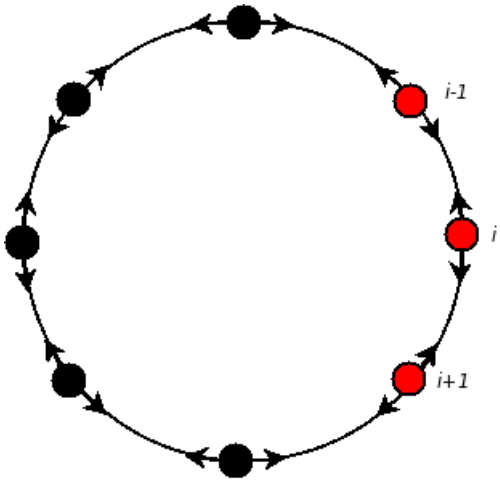

Figura 22: Ciclo de comunicação entre os indivíduos

only" Model) e o Modelo *"Selfless"* (*"Selfless" Model*).

4.2.3.1 Modelo Somente Cognitivo

A ciência cognitiva vem tentando tratar indivíduos como se estivessem isolados, como se a cognição ocorresse dentro da cabeça do indivíduo. Face a isto, Kennedy (1997) apresentou um modelo no qual se baseia apenas no fator cognitivo, com isso a fórmula da atualização da velocidade é descrita por:

$$V_{id}^{t+1} = V_{id}^{t} + \varphi_{id}(P_{id} - X_{id}) \tag{4.14}$$

A equação que define a atualização da posição dos indivíduos é a mesma apresentada pela Eq. (4.12). Em teste comparativo com o modelo PSO canônico, aplicando ambos os modelos na otimização dos pesos de uma rede neural, Kennedy (1997) mostra que o modelo somente cognitivo obteve um desempenho proporcionalmente satisfatório, mas é suscetível a falhas dependendo dos parâmetros utilizados em V_{max} e φ.

4.2.3.2 Modelo Somente Social

Este modelo trabalha utilizando somente o fator social do algoritmo canônico, e também utilizando o conceito de melhor vizinho. Este modelo indica um processo sociofisiológico sem a tendência de indivíduos voltarem a crer que o sucesso provêm de experiências passadas (KENNEDY, 1997). A Eq. (4.15) identifica como são atualizadas as velocidades dos indivíduos.

$$V_{id}^{t+1} = V_{id}^{t} + \varphi_{id}(P_{vd} - X_{id}) \tag{4.15}$$

Em comparação com o modelo canônico e o modelo somente cognitivo, aplicados na otimização dos pesos de uma rede neural, o modelo somente social mostrou uma convergência mais rápida, até mesmo sobre o modelo canônico. Mas isto não pode ser afirmado, pois o mesmo pode não exibir um mesmo desempenho em outros tipos de problema.

4.2.3.3 Modelo "Selfless"

No modelo somente social, quando a posição do próprio indivíduo for a melhor em relação a vizinhança, pode ocorrer uma confusão do indivíduo em relação a auto-influência e a influência dos outros. Com o objetivo de tentar eliminar esta confusão, Kennedy (1997) apresentou o modelo *"Selfless"*, no qual o cálculo do melhor *fitness* na vizinhança exclui o próprio indivíduo do processo.

Este modelo apresentou uma convergência mais rápida, aplicados à otimização de pesos de uma rede neural, em relação ao modelo somente cognitivo, mas obteve um desempenho pior em relação aos modelos canônico e o modelo somente social.

A escolha do melhor modelo apresentado depende diretamente do problema no qual será aplicado, a dimensão dos dados de entrada, entre outras características.

4.3 Aplicações

Pesquisas na área de Inteligência de Enxames tiveram grande avanço nos últimos anos, fazendo com que as abordagens ACO e PSO fossem aplicadas em um grande número de problemas reais. Dorigo, Birattari e Stutzle (2006) destaca que ACO foi aplicado, na maioria, em problemas $\mathcal{NP}$-difícil. Problemas estes, onde os melhores algoritmos conhecidos garantem encontrar a solução ótima em tempo exponencial, sendo muitas vezes de impossível aplicação na prática.

Face a isto, algoritmos de inteligência de enxame surgem como uma alternativa promissora, podendo encontrar soluções de alta qualidade rapidamente.

Algoritmos ACO foram aplicados a problemas redes de telecomunicação, especificamente em roteamento (SIM; SUN, 2003); problemas de agendamento (SOCHA; SAMPELS; MANFRIN, 2003); bioinformática (SHMYGELSKA; HOOS, 2005); otimização de parâmetros de

redes neurais (GONÇALVES; CAMARGO-BRUNETO, 2008c), entre outros.

O algoritmo PSO foi aplicado em diversos problemas complexos de várias áreas como problemas de engenharia nuclear (MEDEIROS, 2005); otimização de parâmetros de redes neurais (GONÇALVES; CAMARGO-BRUNETO, 2008a); otimização de funções (LIANG et al., 2006); mineração de dados (SOUSA; SILVA; NEVES, 2004).

Ambas abordagens podem ser combinadas entre si e/ou com outras técnicas formando sistemas híbridos eficientes.

4.4 Conclusão

Alguns sistemas encontrados na natureza impressionam pela capacidade de resolução de problemas complexos, mesmo tendo como atores agentes simples. Este comportamento, denominado de *inteligência coletiva*, é provido pelas interações sociais entre os próprios indivíduos e com o ambiente, e em conjunto formam um sistema auto-organizado complexo, apto a realizar tarefas complexas. O segredo destes sistemas está na capacidade de comunicação e auto-organização que os permite saber quais são suas atividades por meio do conhecimento do que os outros estão fazendo.

Ferramentas computacionais foram desenvolvidas inspirando-se na interação social entre indivíduos, considerando os princípios da inteligência coletiva, sendo eles: *proximidade entre indivíduos*, *capacidade de avaliar seu comportamento*, *reagir a situações inesperadas*, *nem todas as variações ambientais afetam o comportamento do agente* e *capacidade de adequar ao ambiente*. Ferramentas essas que se mostraram bastante atrativas, através de bem sucedidas aplicações em problemas reais, embora ainda seja necessário um aprofundamento no seu embasamento teórico (LOPES, 2006).

5 Redes Bayesianas

Em muitos problemas reais não há informações completas sobre o ambiente, seja por falha na coleta dos dados, imprecisão do aparelho de coleta ou até mesmo sendo a informação de impossível obtenção. Nestes casos técnicas que trabalham com o raciocínio probabilístico podem ser interessantes.

Métodos de raciocínio probabilístico podem trabalhar bem em ambientes onde existem informações parciais (incompletas) ou informações aproximadas (não exatas), ou seja, tais métodos podem ser aplicados sobre incertezas. Em ambientes de incerteza é possível utilizar-se de ferramentas como a Teoria da Probabilidade com enfoque *Bayesiano*, que considera a probabilidade como o grau de certeza da ocorrência de um evento.

Estes modelos ainda podem ser estendidos a casos onde um banco de exemplos está disponível, e até mesmo onde há falta de informação nos bancos de exemplos, nestes casos os modelos estimarão tais informações, por meio de um processo de *imputação*.

5.1 Cálculo de Probabilidades

A Probabilidade é um campo da matemática que estuda e analisa a ocorrência de fenômenos aleatórios. Fenômenos aleatórios são experimentos repetidos sob as mesmas condições produzem resultados que não se pode prever com certeza (MORGADO et al., 2001). Outros conceitos importantes dentro da probabilidade são definidos a seguir.

Definição 5.1.1 *Espaço amostral é o conjunto de todos os resultados possíveis de um experimento aleatório.*

Definição 5.1.2 *Evento é qualquer subconjunto do espaço amostral.*

Meyer (2000) apresenta uma definição formal do conceito de probabilidade.

Definição 5.1.3 *Dado um experimento ϵ e S o espaço amostral associado a ϵ. A cada evento A associaremos um número real representado por $P(A)$, denominado de probabilidade de A e que satisfaça as seguintes propriedades:*

1. *0 $\leq P(A) \leq$ 1.*
2. *$P(S) = 1$.*
3. *Se A e B forem mutuamente exclusivos , então $P(A \vee B) = P(A) + P(B)$*

Da propriedade (1) é possível identificar que os valores das probabilidades estarão no intervalo [0,1]. Pela propriedade (2) concluímos que a soma de todos os eventos do espaço amostral é igual a 1, e a propriedade (3) diz que, sendo dois eventos mutuamente exclusivos, ou seja, se um está presente então o outro estará ausente, a união das probabilidades é igual à soma das mesmas isoladas. Esta probabilidade é também chamada de **probabilidade incondicional**, pois não depende de nenhuma condição anterior.

5.1.1 Probabilidade Condicional e Independência Condicional

Ao contrário da probabilidade incondicional, a probabilidade condicional depende de uma condição anterior. Representada por $P(B|A)$, a probabilidade condicional pode ser interpretada como: "A probabilidade da ocorrência do evento B, dada a ocorrência do evento A". Se calcularmos $P(B|A)$, estaremos essencialmente calculando $P(B)$ em relação ao espaço amostral reduzido de A (MEYER, 2000).

A definição formal da probabilidade condicional, como observa (HAZZAN; IEZZI, 2004), utiliza-se do conceito de freqüência relativa. Seja um experimento repetido *n* vezes e seja n_A, n_B e $n_{A \wedge B}$, o número de vezes que ocorreram os eventos A, B e A $\wedge$ B. Sendo assim o termo $n_{A \wedge B}/n_A$ representa a freqüência relativa de B condicionada a ocorrência do evento A.

A partir disso é possível afirmar que

$$P(B|A) = \frac{P(A \wedge B)}{P(A)} \tag{5.1}$$

desde que $P(A) > 0$. Sendo assim existem duas maneiras de calcular a probabilidade condicionada $P(B|A)$ (MEYER, 2000):

1. Diretamente, considerando a probabilidade de B em relação ao espaço amostral reduzido de A;

2. Aplicando a definição acima, onde $P(A \wedge B)$ e $P(A)$ são calculados em relação ao espaço amostral original.

Uma importante conseqüênciada probabilidade condicional é Teorema da multiplicação (HAZZAN; IEZZI, 2004):

Teorema 5.1.1 *(Teorema da Multiplicação) A probabilidade de dois eventos ocorrerem simultaneamente é o produto da probabilidade de um deles pela probabilidade do outro dado o primeiro.*

O teorema da multiplicação é representado pela Eq. (5.2)

$$P(A \wedge B) = P(B|A) \cdot P(A) \tag{5.2}$$

Outro conceito importante é a *independência de eventos*. Um evento A independe de B se:

$$P(A|B) = P(A) \tag{5.3}$$

ou seja, A independe de B se a ocorrência de B não afeta a probabilidade de A. Observando o evento B é possível concluir que B também independe de A, pois

$$P(B|A) = \frac{P(A \wedge B)}{P(A)} = \frac{P(B) \cdot P(A|B)}{P(A)} = \frac{P(B) \cdot P(A)}{P(A)} = P(B) \tag{5.4}$$

utilizando o teorema da multiplicação é possível identificar ainda

$$P(A \wedge B) = P(A) \cdot P(B|A) = P(A) \cdot P(B) \tag{5.5}$$

A partir disso é possível definir formalmente a independência de dois eventos:

Definição 5.1.4 *Dois eventos A e B são chamados de independentes se:*

$$P(A \wedge B) = P(A) \cdot P(B)$$

Outro conceito importante é a independência condicional, uma extensão da independência entre dois eventos. A independência condicional pode ser definida como:

Definição 5.1.5 *Um evento X é condicionalmente independente de Y dado Z se a distribuição de probabilidade que rege Z é independente de Y dado o valor de Z, que pode ser representado*

$$P(X|Y \wedge Z) = P(X|Z)$$

5.1.2 Teorema de Bayes

Considere uma partição de um espaço amostral S um conjunto de eventos A_1,A_2,A_3, ..., A_n, os eventos A_i são mutuamente exclusivos e sua união é S. Agora dado outro evento B com probabilidade $P(B) > 0$ então:

$$B = S \wedge B = (A_1 \vee A_2 \vee .... \vee A_n) \wedge B$$

onde $A_i \wedge B$ são mutuamente exclusivos. Conseqüentemente a probabilidade da ocorrência de B é dada por:

$$P(B) = P(A_1 \wedge B) + P(A_2 \wedge B) + ... + P(A_n \wedge B) = \sum_i P(A_i \wedge B)$$

Utilizando-se do teorema da multiplicação 5.1.1, temos que:

$$P(B) = \Sigma_i P(A_i \wedge B) = \Sigma_i P(B|A_i) \cdot P(A_i) \tag{5.6}$$

Além do mais é possível notar que

$$P(A_i \wedge B) = P(B|A_i) \cdot P(A_i) = P(A_i) \cdot P(B)$$

resolvendo em ordem a $P(A_i|B)$, chega-se o Teorema de Bayes (PAULINO; TURKMAN; MURTEIRA, 2003)

$$P(A_i|B) = \frac{P(B|A_i) \cdot P(A_i)}{P(B)} = \frac{P(B|A_i) \cdot P(A_i)}{\sum_i P(B|A_i) \cdot P(A_i)} \tag{5.7}$$

A definição formal do Teorema de Bayes apresentada por (LIPSCHUTZ, 1993) é mostrada pelo teorema 5.1.2.

Teorema 5.1.2 *Suponha A_1,A_2,A_3, ..., A_n ser uma partição de S e B, um evento qualquer. Então para qualquer i*

$$P(A_i|B) = \frac{P(B|A_i) \cdot P(A_i)}{P(B)} = \frac{P(B|A_i) \cdot P(A_i)}{\sum_i P(B|A_i) \cdot P(A_i)} \tag{5.8}$$

Uma interpretação do teorema de Bayes consiste em considerar os eventos A_i como "causas" do evento B, sendo atribuído probabilidades deste evento atuar na ocorrência de B. Esta probabilidade é calculada antes da realização do experimento, sendo designada como a probabilidade *a priori* de A_i. Após a realização do experimento, é conhecido que o evento B ocorreu, então a probabilidade *a priori* é revista por meio da fórmula de Bayes e então passa a atribuir aos eventos A_i, *i*= 1,2,...n as probabilidades *a posteriori* $P(A_i|B)$, *i = 1,2,...n* (CRAMÉR, 1955) (PAULINO; TURKMAN; MURTEIRA, 2003).

Como observado por (PAULINO; TURKMAN; MURTEIRA, 2003) o Teorema de Bayes é para muitos, um dos poucos resultados da matemática que se propõe a caracterizar a aprendizagem com a experiência, ou seja, a modificação de atitude inicial em relação as "causas" depois de ter a informação adicional de que certo acontecimento ou acontecimentos se realizaram.

5.1.3 Variáveis aleatórias e Distribuição de Probabilidade Conjunta

De acordo com (MEYER, 2000) uma variável aleatória é uma função que associa a cada elemento um valor real. O conjunto de valores que uma variável aleatória X pode assumir é chamado de espaço de X. Uma variável aleatória é dita ser discreta se o espaço é finito e contável (NEAPOLITAN, 2003).

De acordo com (CHARNIAK, 1991), a distribuição de probabilidade conjunta (*joint probability distribution*) de um conjunto de variáveis aleatórias X_1, X_2, .., X_n é definida como $P(X_1 \wedge X_2 \wedge .. \wedge X_n)$, para todos os valores de X_1, X_2, .., X_n. A distribuição conjunta de um grupo de variáveis aleatórias fornece toda a informação sobre a distribuição.

A distribuição de probabilidade pode ser representada em uma tabela, como mostra o exemplo abaixo.

Exemplo 5.1.1 *Para um conjunto de variáveis aleatórias binárias {a,b} a distribuição de probabilidade conjunta pode ser representada como mostra a tabela 2.*

	a	¬ a
b	0.04	0.06
¬ b	0.01	0.89

Tabela 2: Distribuição de probabilidade conjunta de duas variáveis binárias

Para *n* variáveis booleanas a distribuição conjunta terá 2^n valores. De qualquer forma a soma de toda a distribuição conjunta é igual a 1, pois a probabilidade de todas as possíveis respostas deve ser 1 (CHARNIAK, 1991).

5.2 Inferência Bayesiana

O processo de obtenção da probabilidade *a posteriori* a partir da probabilidade *a priori* é chamado de Inferência Bayesiana (NEAPOLITAN, 2003).

As inferências Bayesianas sobre uma variável aleatória Y, são baseadas em probabilidades subjetivas ou credibilidades *a posteriori* associadas aos valores do espaço de Y e condicionadas pelo valor particular de um evento X (PAULINO; TURKMAN; MURTEIRA, 2003). Probabilidades subjetivas diferentemente das probabilidades relativas não podem ser obtidas por simples repetição de um experimento, ela é a medida do nível de "confiança" que se tem sobre a verdade de uma determinada proposição. Por exemplo, a probabilidade de uma pessoa ter uma doença A não pode ser obtida como em um experimento de lançamento de dados.

Neapolitan (2003) apresenta as etapas realizadas no processo de modelagem de uma situação a fim de obter informações adicionais sobre ela e para isso utiliza-se da inferência bayesiana:

1. Identificação das variáveis aleatórias do modelo, que representaram as características ou causas e efeitos dentro da situação;

2. Determinação do conjunto mutuamente exclusivo de valores para cada uma das variáveis. Esses valores podem ser obtidos considerando os diferentes estados que a característica pode estar;

3. Decidir as probabilidades de uma variável aleatória ter seu valor, ou seja, calcular a distribuição das probabilidades, o que nem sempre pode ser obtido diretamente;

4. Utilizando dos relacionamentos entre variáveis, identificando as dependências e posteriormente calculando as probabilidades condicionais é possível a obtenção da distribuição das probabilidades.

Neapolitan (2003) observa ainda que a especificação das variáveis e seus valores devem ser precisos o suficiente para satisfazer os requerimentos da situação modelada. Com a situação modelada e com as probabilidades calculadas é possível inferir qualquer indagação sobre a situação.

5.3 Redes Bayesianas

A aplicação da inferência bayesiana sobre um número pequeno de variáveis relacionadas é um processo relativamente simples. Mas em situações reais onde um grande número de variáveis e estados é encontrado a inferência pode não ser trivial.

Uma rede Bayesiana, também chamada de rede de crença, rede probabilística ou rede causal, pode ser vista como um modelo que utiliza teoria dos grafos, condições de Markov e distribuição de probabilidades para representar uma situação, suas variáveis e estados e a partir disto realizar inferências.

Quando uma situação possui um grande número de características (variáveis) surgem alguns problemas, como relatado por (NEAPOLITAN, 2003), considerando que a distribuição de probabilidade conjunta não é prontamente acessível o número exponencial de cálculos necessário na aplicação do teorema de Bayes 5.1.2 torna a inferência impraticável.

Mitchell (1997) define que as redes Bayesianas descrevem a distribuição de probabilidade sobre um conjunto de variáveis. Já (MARQUES; DUTRA, 2008) afirma que matematicamente uma rede bayesiana é uma representação compacta de uma tabela de probabilidades conjunta do universo do problema e que pelo ponto de vista de um especialista esta técnica constitui em um modelo gráfico que representa de forma simples as relações de causalidade das variáveis de um sistema.

Em redes Bayesianas a representação das variáveis e relações é feita utilizando Teoria dos Grafos. As variáveis são os nós e os arcos identificam as relações entre as variáveis, formando um grafo dirigido e sem ciclos, *Directed Acyclic Graph* (DAG), como mostra a figura 23. Neste exemplo a variável Z é condicionada as variáveis X e Y.

Uma Rede Bayesiana consiste do seguinte (MARQUES; DUTRA, 2008):

- Um conjunto de variáveis e um conjunto de arcos ligando as variáveis;
- Cada variável possui um número limitado de estados mutuamente exclusivos;
- As variáveis e arcos formam um grafo dirigido e sem ciclos DAG;

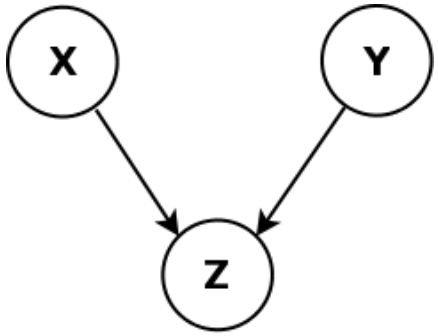

Figura 23: Grafo construído a partir de variáveis e suas relações

- Para cada variável A que possui como pais B_1, ..., B_n existe uma tabela de probabilidade condicional (TPC) $P(A|B_1 \wedge ... \wedge B_m)$.

Caso a variável A for não possua um pai, a tabela de probabilidade é reduzida a probabilidade incondicional $P(A)$.

Uma rede Bayesiana é a representação correta de um domínio caso a **condição de Markov** seja satisfeita. A condição de Markov é definida por (NEAPOLITAN, 2003) como:

Definição 5.3.1 *(Condição de Markov) Suponha a distribuição de probabilidade conjunta das variáveis aleatórias em um conjunto de nós V em um DAG $\mathbb{G} = (V, E)$. Então dizemos que (G, P) satisfazem a condição de Markov se cada variável $X \in V$, X é condicionalmente independente dos nós não descendentes dados seus pais.*

A condição de Markov afirma que as variáveis não-descendentes não fornecem informações adicionais sobre a variável em questão.

De acordo com (PEARL, 1988), considerando F_X e Pa_X o conjunto de filhos e dos pais do nó X respectivamente, e ainda Pa_{F_x} como o conjunto dos pais dos descendentes diretos de X. O conjunto de nós formados pela união destes três conjuntos é denominado de *Markov Blanket*. Os nós pertencentes ao *Markov Blanket* são os únicos nós da rede necessários para predizer o comportamento do nó.

De acordo com (MARQUES; DUTRA, 2008), uma vez definida topologia da rede (distribuição dos nós e os relacionamentos entre as variáveis), é preciso determinar as probabilidades dos nós que participam em dependências diretas e utilizar estas para computar as demais probabilidades desejadas.

O exemplo abaixo, extraído de (RUSSELL; NORVIG, 1995), mostra as etapas de

identificação das características (variáveis), seus conjunto de valores e a construção topológica da rede (mapa causal).

Exemplo 5.3.1 *"Um novo alarme contra assaltos é instalado, mesmo sendo muito confiável na detecção de assaltos ele pode disparar caso ocorra um terremoto. Os dois vizinhos João e Maria se disponibilizaram a telefonar caso o alarme dispare. João sempre liga quando ouve o alarme, entretanto algumas vezes ele confunde o alarme com o telefone e também liga nestes casos. Já a Maria gosta de ouvir música alta e às vezes não houve o alarme disparar, não ligando nestes casos.*"

A modelagem do domínio pode ser representada como segue:

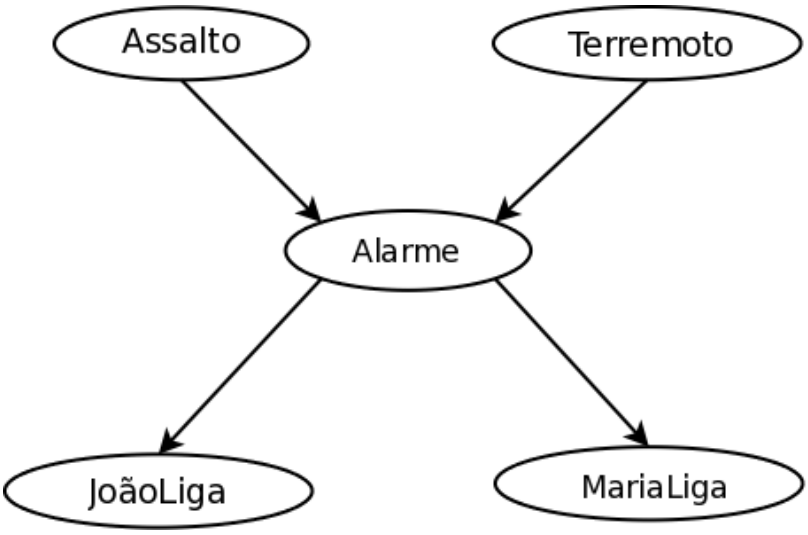

Figura 24: Representação de uma Rede Bayesiana do domínio

É possível notar que as condições da Maria estar ouvindo música e do telefone estar tocando, consequentemente confundindo João, não estão sendo expressas na representação. Essas condições estão implícitas, associados à incerteza relacionada pelos arcos Alarme → JoãoLig e Alarme → MariaLig, pois calcular estas probabilidades seria muito dispendioso ou até impossível. Sendo assim o sistema pode manipular um grande número de probabilidades, mesmo de forma aproximada (MARQUES; DUTRA, 2008).

Após a definição da topologia da rede é necessário calcular a tabela de probabilidade condicional, a qual expressará as probabilidades condicionais de cada variável (nó) dado seus pais (predecessores imediatos). A tabela 5.3 mostra a tabela da variável representada na rede pelo nó *Alarme*, dado seus pais *Assalto* e *Terremoto*.

Assalto	Terremoto	$P(Alarme\|Assalto, Terremoto)$	
		V	F
V	V	0.95	0.05
V	F	0.95	0.05
F	V	0.29	0.71
F	F	0.001	0.999

Tabela 3: Tabela de probabilidade condicional do nó *Alarme*

Os nós que não possuem pai *Assalto* e *Terremoto*, as probabilidades incondicionais são atribuídas por um especialista ou de modo freqüêncista, utilizando a freqüência relativa da ocorrência destes eventos. Para isso um banco de exemplos satisfatoriamente grande deve ser considerado, a fim de obter valores fidedignos da proporção.

Com as tabelas de probabilidade condicional de cada nó calculada, é possível obter a distribuição de probabilidade conjunta e conseqüentemente inferir qualquer evidência sobre o domínio. A figura 25 mostra as tabelas probabilidade condicional de cada nó da Rede Bayesiana da figura 24.

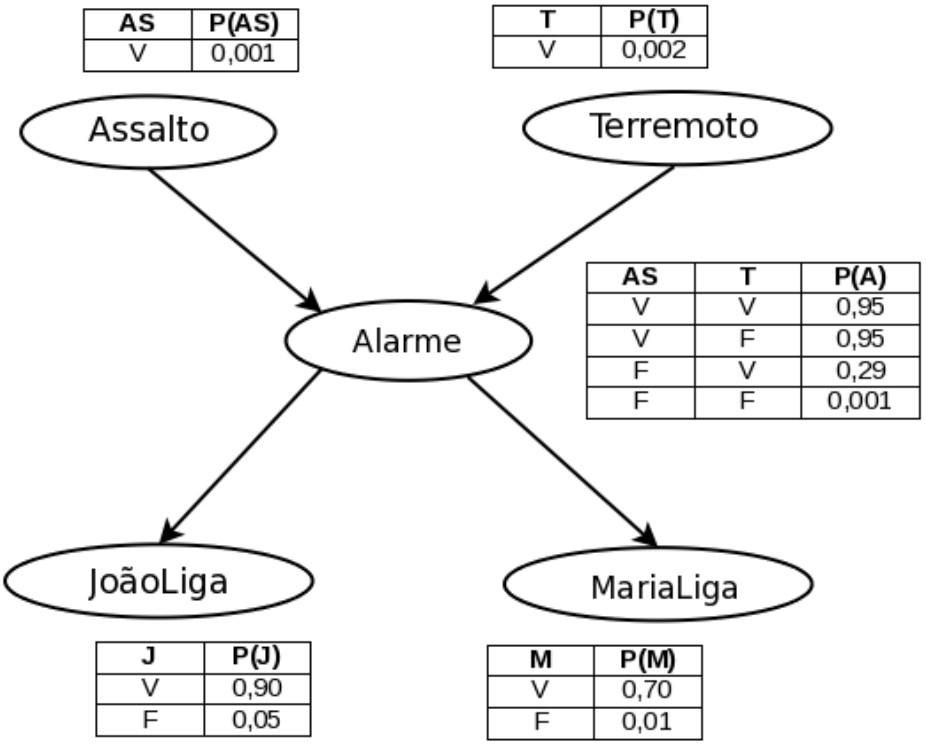

Figura 25: Representação de uma Rede Bayesiana do domínio

5.3.1 Cálculo da distribuição de probabilidade conjunta

Com as tabelas de probabilidade condicional calculadas podemos obter a distribuição de probabilidade conjunta de todo o domínio.

Sendo X_i um nó da rede e *pa(X_i)* representando os pais de X_i. Dessa maneira $X_1, X_2, ..., X_n$ identifica todos os nós do domínio e denotaremos por $P(X_1, X_2, ..., X_n)$ como a distribuição de probabilidade conjunta da rede.

O teorema a seguir define o cálculo da distribuição de probabilidade conjunta de todos os nós, como sendo o produto da probabilidade condicional de todos os nós dados seus pais.

Teorema 5.3.1 *Se uma rede bayesiana satisfaz a condição de Markov, então sua distribuição de probabilidade conjunta é igual ao produto das probabilidades condicionais de todos os nós dado os valores de seus pais.*

A prova do teorema 5.3.1 pode ser encontrada em (NEAPOLITAN, 2003). De uma maneira matemática podemos definir a distribuição de probabilidade conjunta como

$$P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i|pa(X_i)) \tag{5.9}$$

Com isso podemos concluir que as tabelas de probabilidade condicional constituem uma representação distribuída da tabela de probabilidade conjunta do domínio em questão.

Do exemplo 5.3.1, poderíamos querer obter a probabilidade do alarme disparar sem que um assalto e nem um terremoto tenha ocorrido, além de ambos, João e Maria, ligarem. Podemos representar esta indagação por:

$$\begin{aligned} &P(A \wedge \neg AS \wedge \neg T \wedge J \wedge M) \\ &\quad = P(J|A) \text{ x } P(M|A) \text{ x } P(A|\neg AS \wedge \neg T) \text{ x } P(\neg AS) \text{ x } P(\neg T) \\ &\quad = 0.9 \text{ x } 0.70 \text{ x } 0.001 \text{ x } 0.999 \text{ x } 0.998 = 0.00062 \end{aligned}$$

Como observado por (RUSSELL; NORVIG, 1995), o processo geral para cons-

trução de uma rede Bayesiana é dado pelo algoritmo 5..

Algoritmo 5: Algoritmo para construção de uma Rede Bayesiana

```
1 begin
2     Escolher um conjunto de variáveis relevantes X_i que descrevam o domínio;
3     Escolher uma ordem para as variáveis;
4     while Existir variáveis do
5         Selecione uma variável X_i e adicione um nó na rede;
6         Determine os nós pais pa(X_i), dentre os nós que já estejam na rede, de modo
          que a condição de Markov seja satisfeita;
7         Determine a tabela de probabilidade condicional para X_i;
8     end
9 end
```

A condição de que os novos nós devem ser conectados aos nós antigos, garantem que o grafo seja sempre acíclico.

5.3.2 Inferência em redes Bayesianas

Com a rede Bayesiana definida, pode-se extrair conhecimento nela representado através de um processo de inferência. De acordo com (HRUSCHKA JR., 2003) existem vários métodos para realização de inferência, dentre os métodos tradicionais destacam-se o de propagação em poliárvores (PEARL, 1988) e o de eliminação de variáveis (COZMAN, 2000) com suas variações.

Como destacado por (RUSSELL; NORVIG, 1995), inferências podem ser realizadas sobre redes Bayesianas, em quatro maneiras distintas:

1. **Diagnósticos**: partindo dos efeitos para as causas;
2. **Causa**: partindo das causas para os efeitos;
3. **Intercausal**: entre causas de um efeito comum;
4. **Mistas**: combinação de dois ou mais tipos descritos acima.

O autor supracitado ainda afirma que as redes Bayesianas, podem ser utilizadas para outros fins, como:

- Tomar decisões baseadas em probabilidades;
- Decidir quais evidências adicionais devem ser observadas, a fim de obter total conhecimento do domínio;
- Realizar uma *análise sensitiva* para entender quais aspectos do modelo tem maior impacto sobre determinadas variáveis;
- Explicar os resultados de uma inferência probabilística ao usuário.

5.3.3 Aprendizagem Bayesiana

A aprendizagem Bayesiana pode ser visto como uma forma de obter a representação interna da rede que define um dado domínio de modo a facilitar a extração do conhecimento.

Dentro do processo de aprendizagem é necessário calcular as distribuições de probabilidade (parâmetros numéricos) e identificar a estrutura da rede, ou seja, identificar variáveis e as relações de interdependência dadas pelos arcos (HRUSCHKA JR., 2003).

O processo de aprendizagem é dividido em duas partes: aprendizagem da estrutura (e relações entre as variáveis); e a aprendizagem dos parâmetros numéricos (distribuição de probabilidade).

Ambas as partes, estrutura e parâmetros, podem ser aprendidas por meio de um especialista e indutivamente.

Por aprendizagem com especialista entende-se que o conhecimento será transmitido por meio de um especialista, que será responsável por definir e/ou supervisionar a construção da rede baseando-se em seu conhecimento. Já a aprendizagem indutiva utiliza-se de um banco de dados de exemplos, e partindo deste a rede é construída automaticamente.

Diversos algoritmos foram propostos na literatura de redes Bayesianas, com objetivo de encontrar a estrutura que represente fielmente o domínio modelado; e algoritmos que determinam as distribuições de probabilidade, considerando aprendizagem indutiva.

De acordo com (HRUSCHKA JR., 2003), o processo de obtenção dos parâmetros numéricos é geralmente mais simples do que a construção da estrutura da rede.

A aprendizagem dos parâmetros numéricos, considerando que a rede já está estruturada, pode ser estimados através das freqüências relativas, caso exista uma quantidade de dados significativa de uma amostra aleatória.

Para a aprendizagem de estrutura, (HRUSCHKA JR., 2003) observa que existem várias metodologias na literatura sendo que cada uma aplica-se melhor em um tipo de aplicação. Por serem bastante específicas não é possível definir qual é a melhor.

Dentre os métodos existentes destacam-se:

- Métodos de Verossimilhança Máxima;
- Métodos de Teste de Hipóteses;
- Métodos de Verossimilhança Extendidos;
- Métodos *"Minimum Information Complexity"*;
- Métodos *"Resampling"*;
- Métodos Bayesianos, destacando o clássico algoritmo $K2$ (COOPER; HERSKOVITS, 1992).

5.4 Classificador Naive Bayes

Uma rede bayesiana pode ser modelada como um classificador, calculando a probabilidade de $P(C|V)$, onde C representa a classe analisada e V o conjunto de variáveis que descrevem os padrões.

O classificador mais importante dentre os classificadores Bayesianos é o *Naive Bayes*, descrito em (DUDA; HART, 1973). É um modelo simples que se destaca pelos sucessos obtidos na aplicação em diversos problemas, mesmo comparado à classificadores mais complexos (FRIEDMAN; GEIGER; GOLDSZMIDT, 1997).

Este modelo descreve um caso particular de uma rede Bayesiana, o qual considera que as variáveis do domínio são condicionalmente independentes, ou seja, uma característica não é relacionada com a outra. Em decorrência desta restrição utiliza-se o termo *"naive"*. A figura 26 mostra a estrutura da rede *Naive Bayes*, considerando sua restrição.

A classificação é então feita aplicando o teorema de Bayes para calcular a probabilidade de C dado uma particular instância de A_1, A_2, ..., A_n e então predizendo a classe com a maior probabilidade *a posteriori* (FRIEDMAN; GEIGER; GOLDSZMIDT, 1997). De outra maneira:

$$classificador(A_1, A_2, ..., A_n) = \arg\max_c P(c) \prod_i P(A_i|c) \tag{5.10}$$

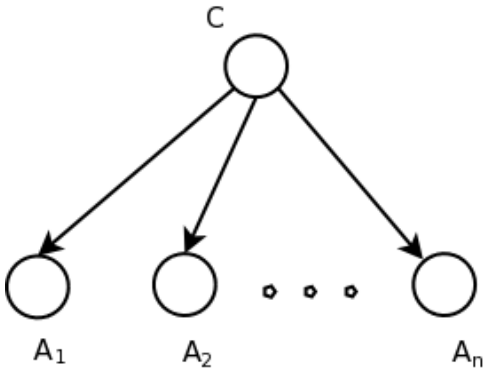

Figura 26: Estrutura de uma rede *Naive Bayes*

O processo de aprendizagem do *Naive Bayes* é feito de maneira indutiva, apresentando um conjunto de dados de treinamento e calculando a probabilidade condicional de cada atributo A_i, dado a classe C (FRIEDMAN; GEIGER; GOLDSZMIDT, 1997).

O algoritmo 6, baseado em (CARVALHO,), identifica as etapas do treinamento do *Naive Bayes*.

Algoritmo 6: Algoritmo de aprendizagem do *Naive Bayes*

Input: Exemplos para treinamento
begin
  for *cada classe* C_j **do**
    Obtenha probabilidade incondicional $P(C_j)$;
    for *cada atributo* A_i *de um exemplo* **do**
      Obtenha a probabilidade estimada $P(A_i|C_j)$;
    end
  end
end

Com a rede treinada é possível realizar a classificação de novos padrões, utilizando a definição 5.10.

A probabilidade incondicional das classes C_j pode ser obtida por meio do conhecimento de um especialista ou atribuindo probabilidades iguais para todas as classes.

Vários outros algoritmos foram propostos como melhorias do *Naive Bayes*, como o *Tree Augmented Naive Bayes (TAN)* apresentado por (FRIEDMAN; GEIGER; GOLDSZMIDT, 1997), o *BN Augmented Naive Bayes (BAN)*, o *General Bayesian Network (GBN)* (CHENG; GREINER, 1999), entre outros.

5.5 Dificuldades na aplicação

Mitchell (1997) identifica algumas dificuldades práticas na aplicação de redes *Bayesianas*, como a necessidade de conhecimento inicial de muitas probabilidades incondicionais. Quando estas probabilidades não são conhecidas, elas são muitas vezes estimadas, com base em conhecimento de especialistas, dados disponíveis previamente e hipóteses sobre a forma das distribuições de probabilidades.

O autor supracitado ainda observa outro empecilho, o significativo custo computacional necessário para determinar a hipótese Bayesiana ótima em casos mais gerais, porém em casos mais restritos o custo pode ser reduzido.

5.6 Aplicações

Diversas aplicações em várias áreas do conhecimento obtiveram ótimos resultados comparados à outras técnicas. Dentre as áreas aplicadas destacam-se, diagnóstico médico ((HECKERMAN, 1990), (LONG; FRASER; NAIMI, 1997)), aprendizagem de mapas (BASYE; VITTER, 1997), interpretação de linguagem (GOLDMAN, 1990), visão (LEVITT; AGOSTA; BINFORD, 1990) entre outros.

Além das aplicações acima descritas, uma rede Bayesiana pode ser utilizada como um classificador, como o *Naive Bayes*. Este classificador vem sendo utilizado em várias áreas, mesmo sendo um modelo simples, ele tem obtido sucesso comparado à outros classificadores mais sofisticados. Áreas essas como classificação textual ((PENG; SCHUURMANS, 2003), (MCCALLUM; NIGAM, 1998)), filtros *anti-spam* (ANDROUTSOPOULOS et al., 2000) (uma aplicação particular das classificações textuais), identificação de genes (bioinformática) (YOUSEF et al., 2006), entre outros.

5.7 Conclusão

As redes Bayesianas utilizam dos conceitos de mapas causais, para modelar domínios. Mapas causais estes que descrevem as variáveis (nós) e as relações de causa e efeito entre elas, na forma de um grafo acíclico. A intensidade das relações é dada pelas tabelas de probabilidade condicional de cada variável, que quantifica as probabilidades de ocorrência de um evento dado seus pais.

O cálculo das probabilidades é obtido com a aplicação do teorema de Bayes,

a partir das probabilidades *a priori*, adquiridas com o auxílio de um especialista ou através de um banco de dados.

Com isso podemos concluir que uma rede Bayesiana que represente corretamente um domínio, pode ser considerada um método bastante atrativo para armazenamento e extração de conhecimento. E ainda podemos destacar o não menos relevante método de classificação *Naive Bayes*, o qual foi provado por inúmeros trabalhos que mesmo possuindo fortes restrições, é incrivelmente eficiente.

6 Máquina de Vetor Suporte

Fundamentada na Teoria da Aprendizagem Estatística, a Máquina de Vetor Suporte, do inglês *Support Vector Machine* (SVM), foi desenvolvida por (VAPNIK, 1995), com o intuito de resolver problemas de classificação de padrões.

Segundo(HAYKIN, 1999) a máquina de vetor suporte é uma outra categoria das redes neurais alimentadas adiante, ou seja, redes cujas saídas dos neurônios de uma camada alimentam os neurônios da camada posterior, não ocorrendo a realimentação.

Esta técnica originalmente desenvolvida para classificação binária, busca a construção de um hiperplano como superfície de decisão, de tal forma que a separação entre exemplos seja máxima. Isso considerando padrões linearmente separáveis[1].

Já para padrões não-linearmente separáveis, busca-se uma função de mapeamento Φ apropriada para tornar o conjunto mapeado linearmente separável.

Devido a sua eficiência em trabalhar com dados de alta dimensionalidade é reportada na literatura como uma técnica altamente robusta, muitas vezes comparada as Redes Neurais (SUNG; MUKKAMALA, 2003) e (DING; DUBCHAK, 2001).

6.1 Teoria da Aprendizagem Estatística

A Teoria da Aprendizagem Estatística (TAE) visa estabelecer condições matemáticas que permitem escolher um classificador, com bom desempenho, para o conjunto de dados disponíveis para treinamento e teste (LORENA; CARVALHO, 2003a). Em outras palavras esta teoria busca encontrar um bom classificador levando em consideração todo o conjunto de dados, porém se abstendo de casos particulares. Na próxima seção serão apresentados alguns conceitos básicos da TAE.

[1] Os padrões devem estar suficientemente separados entre si para assegurar que a superfície de decisão consista de um hiperplano (HAYKIN, 1999).

6.1.1 Conceitos Básicos

O desempenho desejado de um classificador f é que o mesmo obtenha o menor erro durante o treinamento, sendo o erro mensurado pelo número de predições incorretas de f. Sendo assim definimos como risco empírico $R_{emp}(f)$, como sendo a medida de perda entre a resposta desejada e a resposta real. A Eq. (6.1) mostra a definição do risco empírico.

$$R_{emp}(f) = \frac{1}{n}\sum_{i=1}^{n} c(f(\mathbf{x_i}), y_i)) \tag{6.1}$$

onde $c(\cdot)$ é a função de custo relacionada a previsão $f(\mathbf{x_i})$ com a saída desejada y_i (LORENA; CARVALHO, 2003b), onde um tipo de função de custo é a "perda 0/1" definida pela Eq. (6.2). O processo de busca por uma função f' que represente um menor valor de R_{emp} é denominado de *Minimização do Risco Empírico*.

$$c(f(\mathbf{x_i}), y_i) = \begin{cases} 1 & \text{, se } y_i \text{f}(x_i) < 0 \\ 0 & \text{, caso contrário} \end{cases} \tag{6.2}$$

Sobre a hipótese de que os padrões de treinamento ($\mathbf{x_i}$,y_i) são gerados por uma distribuição de probabilidade $P(x, y)$ em $\mathbb{R}^N$ x {-1,+1} sendo P desconhecida. A probabilidade de classificação incorreta do classificador f é denominada de Risco Funcional, que quantifica a capacidade de generalização, conforme é mostrado pela Eq. (6.3) (SMOLA et al., 1999a) (SMOLA et al., 1999b).

$$R(f) = \int c(f(\mathbf{x_i}), y_i) dP(\mathbf{x_i}, y_i) \tag{6.3}$$

Durante processo de treinamento, $R_{emp}(f)$, pode ser facilmente obtido, ao contrário de $R(f)$, pois em geral a distribuição de probabilidades P é desconhecida (LORENA; CARVALHO, 2003a).

A partir disto, dado um conjunto de dados de treinamento ($\mathbf{x_i}$,y_i) com $\mathbf{x_i} \in \mathbb{R}^N$ e $y_i \in \{-1, +1\}$, $i = \{1, 2, ..., n\}$, sendo $\mathbf{x_i}$ o vetor de entrada e y_i o rótulo da classe.

O objetivo então é estimar uma função f: $\mathbb{R}^N \rightarrow \{-1, +1\}$. Caso nenhuma restrição seja imposta na classe de funções em que se escolhe a estimativa f, pode ocorrer que a função obtenha um bom desempenho no conjunto de treinamento, porém não tendo o mesmo desempenho em padrões desconhecidos, sendo este fenômeno denominado de *"'overfitting"*. Em outras palavras, a minimização apenas do risco empírico $R_{emp}(f)$ não garante uma boa

capacidade de generalização, sendo desejado um classificador f^* tal que $R(f^*) = \min_{f \in F} R(f)$, onde F é o conjunto de funções f possíveis.

A figura 27 mostra um exemplo onde uma classe de funções pode ser utilizada para separar padrões linearmente separáveis. É necessário determinar uma função que minimize o R_{emp}, representado na figura como a reta mais escura.

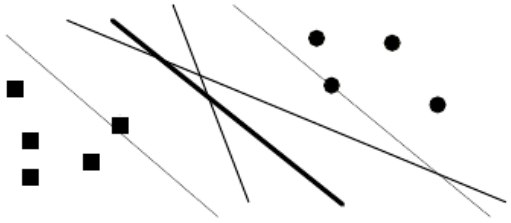

Figura 27: Classe de hiperplanos com um hiperplano ótimo

A TAE provê formas de limitar a classe de funções (hiperplanos), com o intuito de prevenir modelos ruins, ou seja, que levem ao *"overfitting"*, implementando uma função com a capacidade adequada para o conjunto de dados de treinamento (HEARST et al., 1998). Estas limitações são impostas ao risco funcional da função. Os limites utilizam o conceito de dimensão Vapnik-Chervonenkis (VC).

6.1.2 Dimensão VC

De acordo com (SMOLA et al., 1999b), dado um conjunto de funções sinal G, sua dimensão VC é definida como o tamanho do maior conjunto de pontos que pode ser particionado arbitrariamente pelas funções contidas em G.

Em outras palavras a dimensão VC do conjunto de funções de classificação G é o número máximo de exemplos de treinamento que pode ser aprendido pela máquina sem erro, para todas as rotulações possíveis das funções de classificação (HAYKIN, 1999).

De forma genérica, para funções lineares no $\mathbb{R}^N$ para $n \geq 2$ a dimensão VC é dada pela expressão 6.4.

$$VC(n) = n + 1. \tag{6.4}$$

6.1.3 Conceito de margem e vetores suporte

Sendo $f(x) = (\mathbf{w} \cdot \mathbf{x}) + b$ um hiperplano, podemos definir como *margem* como a menor distância entre os exemplos do conjunto de treinamento e o hiperplano utilizado para separação destas classes (LORENA; CARVALHO, 2003b). A margem determina quão bem duas classes podem ser separadas (SMOLA et al., 1999b).

A margem máxima é obtida com a utilização do hiperplano ótimo, a definição de hiperplano ótimo é apresentada na seção 6.2.1. Podemos definir formalmente a margem conforme 6.1.1.

Definição 6.1.1 *(Margem) A margem ρ de um classificador f é definida por*

$$\rho = \min_i y_i f(\mathbf{x_i}). \tag{6.5}$$

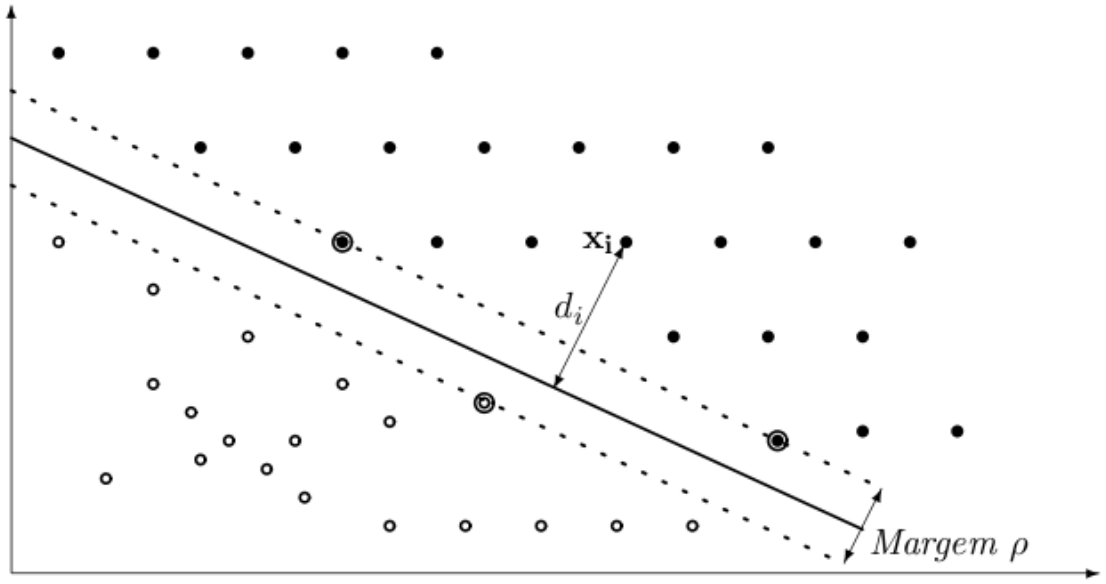

Figura 28: Identificação da margem ρ e dos vetores suporte sobre a linha pontilhada. [Fonte: (LIMA, 2002)]

A margem é obtida pela distância entre o hiperplano e os vetores que estão mais próximos a ele, sendo estes vetores denominados de *vetores suporte*. De acordo com (SMOLA et al., 1999b) os vetores suporte são padrões críticos, que sozinhos determinam o hiperplano ótimo, sendo os outros padrões (não-críticos) irrelevantes, podendo ser removidos do conjunto de treinamento sem afetar os resultados. Na figura 28 os vetores suporte são destacados por círculos externos nos padrões.

6.2 Classificação de Padrões Linearmente Separáveis

Uma classificação linear consiste em determinar uma função $f : X \subseteq \mathbb{R}^N \rightarrow \mathbb{R}^N$, que atribui um rótulo (+1) se $f(x) \geqslant 0$ e (-1) caso contrário. Considerando uma função linear, podemos representá-la pela Eq. (6.7).

$$f(x) = \langle \mathbf{w} \cdot \mathbf{x} \rangle + b \tag{6.6}$$

$$= \sum_{i=1}^{n} w_i \mathbf{x_i} + b \tag{6.7}$$

onde $\mathbf{w}$ e $b \in \mathbb{R}^N$ x $\mathbb{R}^N$, são conhecidos como *vetor peso* e *bias*, sendo estes parâmetros responsáveis por controlar a função e a regra de decisão (LIMA, 2002). Os valores de $\mathbf{w}$ e b são obtidos pelo processo de aprendizagem a partir dos dados de entrada.

O vetor peso $(\mathbf{w})$ e o *bias* (b) podem ser interpretados geometricamente sobre um hiperplano. Um hiperplano é um subespaço afim, que divide um espaço em duas partes, correspondendo a dados de duas classes distintas (LIMA, 2002). O vetor peso $(\mathbf{w})$ define uma direção perpendicular ao hiperplano, como mostra a figura 29, e com a variação do *bias* o hiperplano é movido paralelamente a ele mesmo.

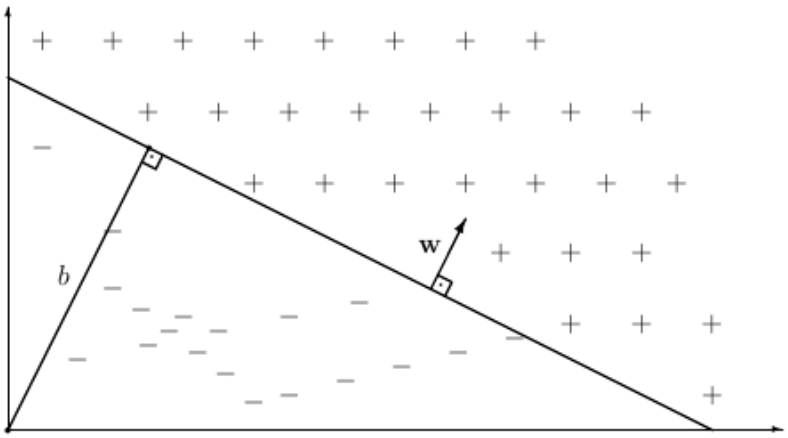

Figura 29: Interpretação geométrica de **w** e **b** sobre um hiperplano. [Fonte: (LIMA, 2002)]

Sendo assim um SVM linear busca encontrar um hiperplano que separe perfeitamente os dados de cada classe e cuja margem de separação seja máxima, sendo este hiperplano denominado de *hiperplano ótimo*.

6.2.1 Hiperplano Ótimo

Assumindo-se que o conjunto de treinamento é linearmente separável, o hiperplano ótimo é o hiperplano de separação com maior margem. O hiperplano ótimo é definido como:

$$\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0 \tag{6.8}$$

sendo $\mathbf{w}$ e b, o vetor peso e o *bias* respectivamente.

Considerando a restrição imposta pela Eq. (6.9), os classificadores lineares que separam um conjunto de treinamento possuem margem positiva. Ou seja, esta restrição afirma que não há nenhum dado entre $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$ e $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = \pm 1$, sendo a margem sempre maior que a distância entre os hiperplanos $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$ e $|\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 1|$. Devido a estas suposições as SVMs obtidas são normalmente chamadas de SVMs com margens rígidas (ou largas).

$$\begin{aligned} \langle \mathbf{w} \cdot \mathbf{x_i} \rangle + b \geq +1 \quad &, \textit{para } y_i = \textit{+1} \\ \langle \mathbf{w} \cdot \mathbf{x_i} \rangle + b \leq -1 \quad &, \textit{para } y_i = \textit{-1} \end{aligned} \tag{6.9}$$

Estas equações podem ser combinadas em

$$y_i(\langle \mathbf{w} \cdot \mathbf{x_i} \rangle + b) \geq 1, \;\; i = \{1, 2, ..., n\} \tag{6.10}$$

Seja d_+ (d_-) a distância euclidiana entre os vetores suporte positivos (negativos) e o hiperplano, definimos como *margem* ρ de um hiperplano de separação como sendo a maior margem geométrica entre todos os hiperplanos, podemos representar por $\rho = (d_+ + d_-)$. Denotaremos por $d_i(\mathbf{w}, b; \boldsymbol{x_i})$, como a distância de um dado $\boldsymbol{x_i}$ ao hiperplano (**w**,b), sendo calculado pela Eq. (6.11) (LIMA, 2002)

$$d_i(\mathbf{w}, b; \mathbf{x_i}) = \frac{|\langle \mathbf{w} \cdot \mathbf{x_i} \rangle + b|}{\|\mathbf{w}\|} = \frac{y_i(\langle \mathbf{w} \cdot \mathbf{x_i} \rangle + b)}{\|\mathbf{w}\|} \tag{6.11}$$

levando em consideração a restrição imposta pela Eq. (6.10), podemos escrever

$$d_i(\mathbf{w}, b; \mathbf{x_i}) \geq \frac{1}{\|\mathbf{w}\|}. \tag{6.12}$$

Com isso podemos identificar $\frac{1}{\|\mathbf{w}\|}$ como o limite inferior da distância entre os vetores suporte $\boldsymbol{x_i}$ e o hiperplano de separação ($\mathbf{w}$,b), as distâncias d_+ e d_- ficam

$$d_+ = d_- = \frac{1}{\|\mathbf{w}\|}. \tag{6.13}$$

Como suposto anteriormente que a margem é sempre maior que a última instância, a minimização de $\|\mathbf{w}\|$ leva a maximização da margem. A partir disto podemos definir a margem ρ através da Eq. (6.14)

$$\rho = (d_+ + d_-) = \frac{\mathbf{2}}{\|\mathbf{w}\|}. \tag{6.14}$$

A figura 30 mostra a distância entre hiperplanos e os vetores suporte.

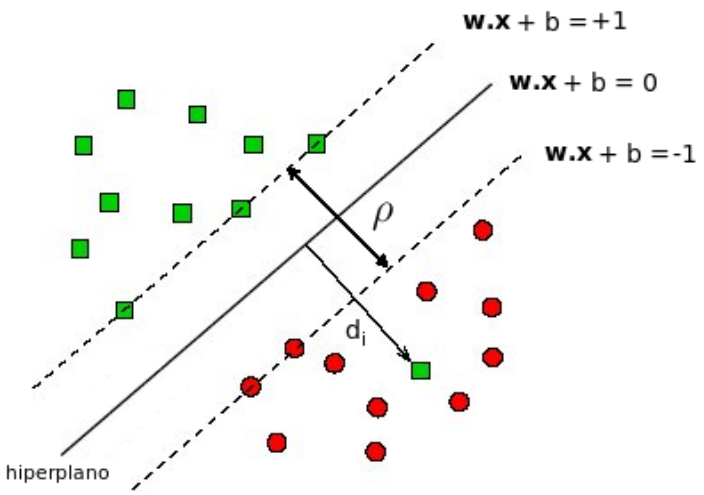

Figura 30: Distância entre hiperplanos e vetores suporte.

O hiperplano ótimo é dado pela minimização da norma $\|\mathbf{w}\|$, considerando a restrição da Eq. (6.10). Formalmente podemos reescrever (LORENA; CARVALHO, 2003b)

$$\begin{aligned} &\textbf{Problema} && P1 \\ &\textbf{Minimizar} && \|\mathbf{w}\| \\ &\textbf{Sujeito a} && y_i(\langle \mathbf{w} \cdot \mathbf{x_i} \rangle + b) \geq 1, \text{ para i=\{1,2,...,n\}} \end{aligned} \tag{6.15}$$

Como pode ser observado, o problema acima trata-se de um problema clássico de otimização, denominado *programação quadrática* (HEARST et al., 1998). Este problema

pode ser resolvido com o método clássico de *multiplicadores de Lagrange* (LIMA, 2002).

Utilizando a teoria dos multiplicadores de Lagrange, podemos representar o problema 6.15, como:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i (y_i(\langle \mathbf{w} \cdot \mathbf{x_i} \rangle + b) - 1) \tag{6.16}$$

onde α_i são os *multiplicadores de Lagrange*. O problema então passa a ser a minimização de $L(\mathbf{w}, b, \alpha)$, em relação a $\mathbf{w}$ e b e a maximização dos α_i (LORENA; CARVALHO, 2003b). O método dos multiplicadores encontra os pontos ótimos igualando as derivadas parciais a zero. Sendo assim os pontos ótimos da Eq. (6.16) são obtidos por meio da resolução das igualdades:

$$\frac{\partial L}{\partial b} = 0 \tag{6.17}$$

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \tag{6.18}$$

sendo

$$\frac{\partial L}{\partial \mathbf{w}} = \left(\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, ..., \frac{\partial L}{\partial w_n} \right). \tag{6.19}$$

A partir das Equações 6.17 e 6.18 obtém-se

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{6.20}$$

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x_i}. \tag{6.21}$$

Substituindo as equações 6.20 e 6.21 no lado direito da Eq. (6.16), chegamos ao seguinte problema de otimização:

$$\begin{array}{ll} \textbf{Problema} & \textbf{P2} \\ \textbf{Maximizar} & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \mathbf{x_i} \cdot x_j \rangle \\ \textbf{Sujeito a} & \begin{cases} \alpha_i \geq 0, \ \ i = \{1, ..., n\} \\ \sum_{i=1}^{n} \alpha_i y_i = 0 \end{cases} \end{array} \tag{6.22}$$

O objetivo então é determinar os valores ótimos de (**w**,b), que representaremos por ($\mathbf{w}^*$,b^*). Através da Eq. (6.21) podemos calcular $\mathbf{w}^*$ e b^* como segue:

$$\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x_i} \tag{6.23}$$

o valor de b^* pode ser obtido utilizando as equações de Karush-Kuhn-Tucker (KKT)

$$\alpha_i^* (y_i (\langle \mathbf{w}^* \cdot \mathbf{x_i} \rangle + b^*) - 1) = 0, \ \ i = \{1, ..., n\} \tag{6.24}$$

É possível observar que os α_i^*'s assumem valores positivos para os exemplos de treinamento que estão a uma distância do hiperplano ótimo igual a largura da margem, ou seja, os vetores suporte. Para o restante dos exemplos, α_i^* é nulo. Portanto, conclui-se que o hiperplano ótimo é obtido unicamente pelos vetores suporte. Logo, se apenas o subconjunto de dados de treinamento formado pelos vetores suporte fossem utilizados, o hiperplano obtido seria o mesmo gerado pela utilização de todo o conjunto (LORENA; CARVALHO, 2003b).

Dado um vetor suporte $\mathbf{x_j}$, podemos obter b^* por meio da condição de KKT

$$b^* = y_j - \langle \mathbf{w}^* \cdot \mathbf{x_j} \rangle. \tag{6.25}$$

Com os valores dos parâmetros $\mathbf{w}^*$ e b^* calculados, podemos classificar de um novo padrão **z** apenas calculando

$$sgn(\langle \mathbf{w}^* \cdot \mathbf{z} \rangle) + b^*) \tag{6.26}$$

a classificação é dada apenas pelo cálculo do produto interno entre o novo padrão e todos os vetores suporte.

6.3 Classificação de Padrões Não-Linearmente Separáveis

As SVMs apresentadas até agora, trabalham apenas quando os padrões são linearmente separáveis. Em problemas reais esta característica é dificilmente encontrada, sendo a maioria deles complexos e não-lineares. Para estender a SVM linear a resolução de problemas não-lineares foram introduzidas funções reais, que mapeiam o conjunto de treinamento em um espaço linearmente separável, o *espaço de características*.

Um conjunto de dados é dito ser não-linearmente separável, caso não seja possível separar os dados com um hiperplano. A figura 31 mostra um conjunto linearmente e outro não-linearmente separável.

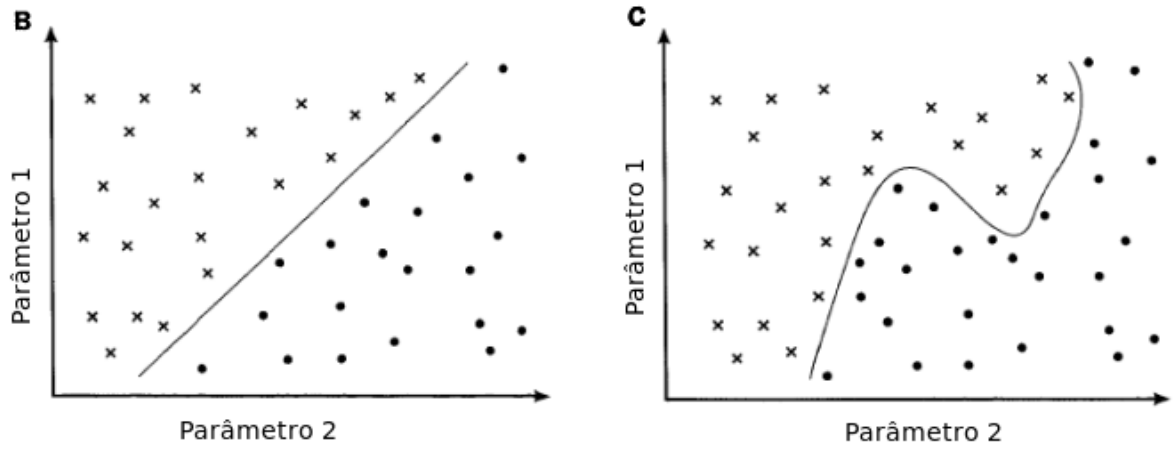

Figura 31: Exemplos de padrões linearmente e não-linearmente separável respectivamente.

O teorema de Cover afirma que um problema não-linear tem maior probabilidade de ser linearmente separável, em um espaço de mais alta dimensionalidade (SMOLA et al., 1999b). A partir disso, a SVM não-linear realiza uma mudança de dimensionalidade, por meio das funções *Kernel*, caindo então em um problema de classificação linear, podendo fazer uso do hiperplano ótimo.

6.3.1 Mapeamento no espaço de características

Seja o conjunto de entrada S representado pelos pares $\{(\mathbf{x_1}, y_1), ..., (\mathbf{x_n}, y_n)\}$, com $y_i, i = 1, 2, .., n$ o rótulo de cada padrão i, o conjunto de dados de treinamento.

O espaço de característica é um espaço de mais alta dimensionalidade no qual

serão mapeados o conjunto de entrada S, por meio de uma função Φ, a fim de obter um novo conjunto de dados S' linearmente separável, representado por $\{(\mathbf{\Phi}(\mathbf{x_1}), y_1), ..., (\mathbf{\Phi}(\mathbf{x_n}), y_n)\}$. A figura 32 mostra o processo de mapeamento.

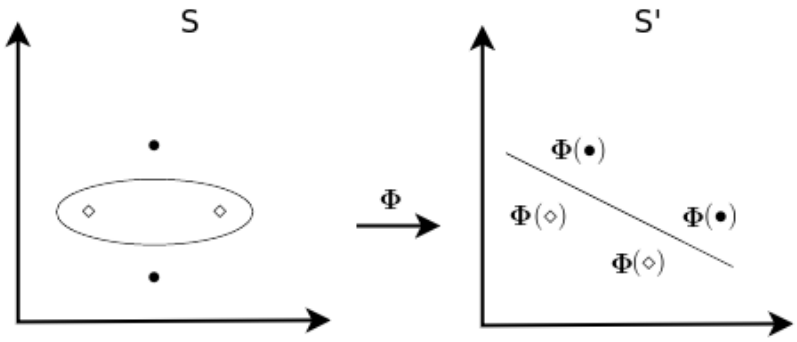

Figura 32: Mapeamento do conjunto de treinamento para o espaço de característica

Muitas vezes é interessante diminuir o número de características do conjunto de entrada, em um subconjunto menor com apenas os atributos que contém as informações essenciais, este procedimento é denominado de *redução da dimensionalidade* (LIMA, 2002).

Esta modificação de conjunto de atributos apresenta ganho no sistema, diminuindo o tempo computacional e aumentando a acurácia, pois geralmente estes parâmetros degradam o sistema com aumento do número de características, sendo este fenômeno referenciado de *maldição da dimensionalidade* (LIMA, 2002).

6.3.2 SVMs lineares no espaço de características

Com os dados de treinamento mapeados para o espaço de características, a única modificação a ser feita na SVM linear descrita na Eq. (6.22) é a utilização dos valores mapeados $\Phi(\mathbf{x})$ no lugar de $\mathbf{x}$, sendo assim o problema consiste em

$$\begin{aligned} &\textbf{Problema} \quad \textbf{P3} \\ &\textbf{Maximizar} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \mathbf{\Phi}(\mathbf{x_i}) \cdot \mathbf{\Phi}(\mathbf{x_j}) \rangle \\ &\textbf{Sujeito a} \quad \begin{cases} \alpha_i \geq 0, \;\; i = \{1, ..., n\} \\ \displaystyle\sum_{i=1}^{n} \alpha_i y_i = 0 \end{cases} \end{aligned} \tag{6.27}$$

valendo para classificação não-linear as mesmas considerações do Karush-Kuhn-Tucker, des-

crito no classificador linear. O hiperplano de decisão ótimo, agora é definido como

$$(\mathbf{w} \cdot \Phi(\mathbf{x})) + b = 0 \tag{6.28}$$

O problema de classificação não-linear de um novo padrão **z** é solucionado calculando

$$sgn(\langle \mathbf{w}^* \cdot \mathbf{\Phi}(\mathbf{z}) \rangle + b^*) \tag{6.29}$$

6.3.3 Funções "Kernel"

Uma função *Kernel* recebe dois dados de entrada $\mathbf{x_i}$ e $\mathbf{x_j}$ e calcula o produto interno destes dados no espaço de características

$$\mathcal{K}(\mathbf{x_i}, \mathbf{x_j}) = \langle \Phi(\mathbf{x_i}) \cdot \Phi(\mathbf{x_j}) \rangle \tag{6.30}$$

sendo necessário que a função $\Phi(\cdot)$ pertença a um domínio, onde seja possível o cálculo do produto interno. Funções estas que satisfazem as condições do *Teorema de Mercer*.

Uma função é dita ser uma função *Kernel* se ela satisfaz as condições estabelecidas pelo *Teorema de Mercer*.

Teorema 6.3.1 *(Teorema de Mercer) Uma função é dita ser uma função Kernel, se a matriz K é positivamente definida, onde K é obtida por*

$$K = K_{ij} = \mathcal{K}(\mathbf{x_i}, \mathbf{x_j}). \tag{6.31}$$

Uma matriz é positivamente definida, se seus autovalores são maiores que zero. As funções *Kernel* que satisfazem as condições do Teorema de Mercer, são chamadas de *Kernels de Mercer* (SMOLA et al., 1999b).

Algumas das funções *Kernels* mais utilizadas estão descritas na tabela 4.

Algumas considerações sobre as funções *Kernels* descritas na tabela 4, foram apresentadas por (HAYKIN, 1999):

- Na Máquina de aprendizagem polinomial a potência p é especificada *a priori* pelo usuário;

Tipo de Kernel	**Função $\mathcal{K}(\mathbf{x_i}, \mathbf{x_j})$**	**Tipo do Classificador**
Polinomial	$(\langle \mathbf{x_i} \cdot \mathbf{x_j} \rangle + 1)^p$	Máquina de aprendizagem polinomial
Gaussiano (ou RBF)	$\exp\left(-\frac{\|\mathbf{x_i} - \mathbf{x_j}\|^2}{2\sigma^2}\right)$	Rede RBF
Sigmoidal	$\tanh(\beta_0 \langle \mathbf{x_i} \cdot \mathbf{x_j} \rangle) + \beta_1$	Perceptron de duas camadas

Tabela 4: Resumo dos *Kernels* mais populares

- Na Rede RBF a amplitude σ^2, comum a todos os *Kernels*, é especificada pelo usuário;
- No Perceptron de duas camadas o teorema de Mercer é satisfeito apenas para alguns valores de β_0 e β_1.

Algumas escolhas devem ser feitas para obtenção de uma SVM, como a função *Kernel*, os parâmetros da função, além da definição do algoritmo para determinação do hiperplano ótimo (LORENA; CARVALHO, 2003b).

6.4 Classificação Multiclasses

As máquinas de vetores suporte foram propostas inicialmente como ferramenta de classificação binária. Porém muitos dos problemas reais possuem características multiclasses. Para que fosse possível a utilização da SVM neste tipo de aplicação, foram propostos alguns procedimentos ara estender a SVM binária.

Em termos formais, em um sistema multiclasses o conjunto de treinamento é composto por pares $(\mathbf{x_i}, y_i)$, tal que $y_i \in \{1, ..., k\}$, com $k > 2$, onde k é o número de classes.

As principais abordagens utilizam como base a decomposição de um problema multiclasse com k classes, $k > 2$, em k problemas binários, destacando os métodos: decomposição "Um-Contra-Todos" e decomposição "Todos-Contra-Todos". Estes métodos são apresentados na seção 6.4.1 e 6.4.2 respectivamente.

6.4.1 Decomposição "Um-Contra-Todos"

O método "Um-Contra-Todos" (UCT) baseia-se na construção de k classificadores binários, sendo k o número de classes. Cada classificador f_i é responsável por classificar uma classe i das demais.

Dado um novo padrão $\mathbf{x}$, a classe a qual este novo padrão pertence é a classe representada pelo classificador que obteve o valor máximo entre o k classificadores. Formalmente é definido por

$$f(x) = \arg \max_{1 \leq i \leq k} (f_i(\mathbf{x})). \tag{6.32}$$

6.4.2 Decomposição "Todos-Contra-Todos"

Proposto por (FRIEDMAN, 1996) a abordagem "Todos-Contra-Todos" (TCT) consiste em comparar as classes duas a duas, sendo necessárias $k \cdot (k-1)/2$ SVMs, onde k é o número de classes. Para decidir a qual classe pertence um novo padrão $\mathbf{x}$, utiliza-se um esquema de votação por maioria, onde cada uma das SVMs fornecem um resultado. A solução final é dada pela classe com maior quantidade de votos.

Com um grande número de SVMs executadas, um alto tempo de processamento é necessário, comparado ao método de decomposição UCT. Por exemplo, sendo $k = 10$, no método UCT são necessárias 10 SVMs enquanto seriam necessárias 45 SVMs no método TCT. O método TCT não provê limites de generalização (PLATT; CRISTIANINI; SHAWE-TAYLOR, 2000).

Com o intuito de solucionar estes problemas, (PLATT; CRISTIANINI; SHAWE-TAYLOR, 2000) propôs um modelo, denominado DAGSVM, que utiliza grafos direcionados acíclicos (DAG), sendo cada nó do grafo encarregado de classificar entre duas classes distintas. A figura 33 mostra o grafo de SVMs que representa a arquitetura do modelo DAGSVM.

Segundo (PLATT; CRISTIANINI; SHAWE-TAYLOR, 2000) o algoritmo trabalho equivalentemente a operação sobre um lista, onde cada nó da árvore elimina uma classe da lista.

Inicialmente todas as classes estão na lista, sendo que cada nó do grafo decide, entre a primeira e última classe da lista, a classe com maior chance do padrão a pertencer. A classe não escolhida é eliminada da lista e continua o processo com a primeira e última classe da nova lista. O algoritmo termina quando há apenas uma classe na lista. Sendo

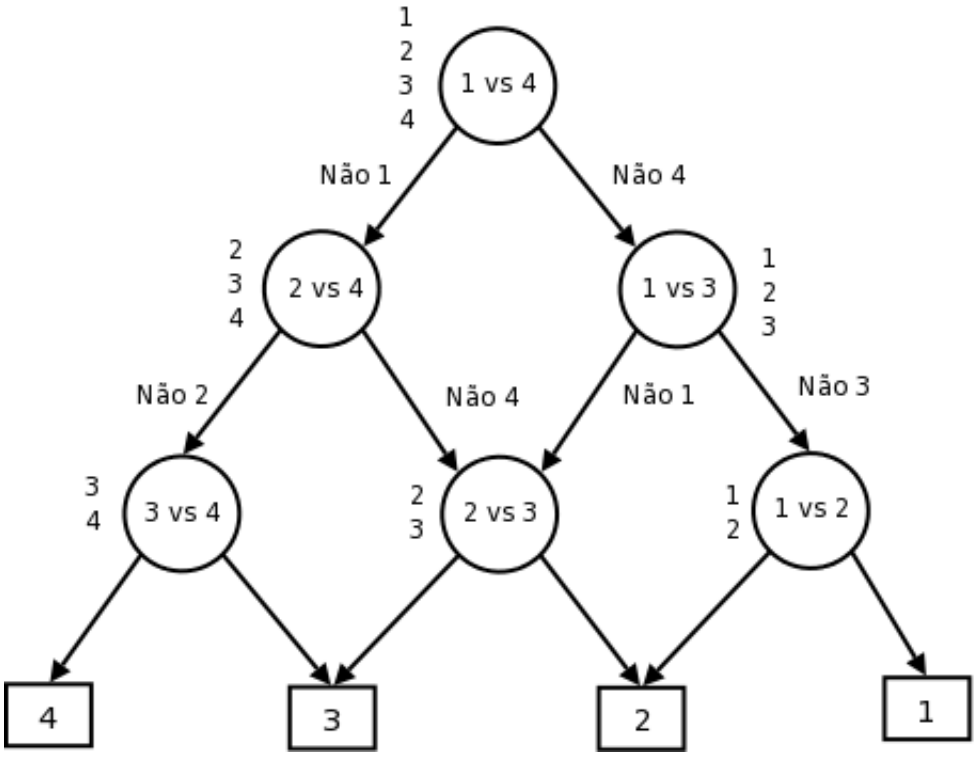

Figura 33: Arquitetura do método DAGSVM.

assim o número de avaliações requeridas pelo DAGSVM é $k-1$, conseqüentemente possui um tempo de processamento bem menor do que o método TCT originalmente proposto.

(PLATT; CRISTIANINI; SHAWE-TAYLOR, 2000) mostrou que o limite do erro de generalização é determinado pela margem máxima da SVM de cada nó do grafo.

6.5 Aplicações

A proposta inicial da SVM foi a aplicação em problemas de classificação, sendo este o principal enfoque desta técnica. Pode ainda ser aplicado em problemas de regressão (HAYKIN, 1999).

Devido ao fato de sua eficiência em problemas de alta dimensionalidade, a SVM vem obtendo grande sucesso em aplicações de visão computacional, que busca extrair informações a partir de imagens. Exemplos de aplicação em visão computacional são: classificação de impressões digitais (LIMA, 2002). Também são aplicadas em bioinformática (MA; HUANG, 2008) e classificação textual (TONG; KOLLER, 2000). A SVM também pode ser aplicada em regressão não-linear, como em (SUN; SUN, 2003).

Esta técnica tem seus resultados muitas vezes comparados com resultados

de outras técnicas como redes neurais.

6.6 Conclusão

As Máquinas de Vetor Suporte destacam-se pela forte fundamentação teórica existente, possuindo como base a teoria da aprendizagem estatística, sendo esta característica um diferencial sobre outras técnicas como redes neurais, que não possui um modelo teórico.

A capacidade em trabalhar com padrões de alta dimensionalidade é outra característica interessante desta técnica, sendo ideal para aplicação em problemas de visão computacional, como reconhecimento de padrões e filtragem.

Mesmo com características atrativas, algumas ressalvas devem ser feitas, como descreve (LORENA; CARVALHO, 2003b):

- Velocidade de classificação pode ser menor do que outras técnicas como Redes Neurais;
- Alta complexidade computacional na busca de soluções, agravando ainda mais quando um grande número de dados estão disponíveis para treinamento;
- Conhecimento adquirido não é facilmente interpretável.

Diversos estudos foram realizados com o intuito de minimizar estas deficiências, o que juntamente com a robustez desta técnica, faz da SVM uma das técnicas mais exploradas atualmente.

7 Estudos de Caso

As técnicas apresentadas possuem características e aplicações diferentes, sendo necessária a elaboração de dois estudos de caso distintos. Com base em suas características, dividiremos as abordagens em dois subconjuntos, sendo o primeiro aplicado em um problema de otimização e o outro em classificação de padrões.

Para otimização, aqui denominado de estudo de caso 1, serão consideradas as seguintes técnicas: Algoritmos Genéticos; Otimização por Enxame de Partículas (*Particle Swarm Optimization-PSO*) e Otimização por Colônia de Formigas (*Ant Colony Optimization-ACO*), sendo que as duas últimas estão dentro da área maior *Swarm Intelligence*. É importante salientar que tais abordagens não somente são aplicadas em problemas de otimização, como também em outros tipos: mineração de dados (PARPINELLI; LOPES; FREITAS, 2002), método de aprendizagem de uma rede neural (HARPHAM; DAWSON; BROWN, 2004), porém neste trabalho serão utilizadas com o objetivo de otimização.

Já para classificação de padrões, estudo de caso 2, serão analisadas e comparadas as seguintes técnicas: Redes Neurais, Redes Bayesianas e Máquina de Vetor Suporte. Destaca-se também que as mesmas podem ser aplicadas para outros fins.

7.1 Estudo de Caso 1

No estudo de caso 1 o problema abordado é a otimização de parâmetros configuráveis de uma rede neural do tipo *Multilayer Perceptron (MLP)*, treinada com algoritmo *backpropagation*. De acordo com (BASHEER; HAJMEER, 2000) e (SUKTHOMYA; TANNOCK, 2005) a configuração dos parâmetros da rede neural MLP é reportada por vários autores como um processo de tentativa e erro, o que se torna indesejável quando o espaço de busca (número de combinações possíveis dos parâmetros) é muito grande. Os parâmetros a serem otimizados são: taxa de aprendizagem, constante de momento e número de neurônios na camada oculta. Neste caso, a rede neural é constituída por uma única camada oculta.

Os valores ideais dos parâmetros dependem do problema aplicado, sendo que neste trabalho a rede neural é apta a classificar indivíduos entre saudáveis ou com Doença Pulmonar Obstrutiva Crônica (DPOC), com base em cinco medidas fisiológicas distintas, as quais são:

- Capacidade Vital Forçada (CVF);
- Volume Expiratório Forçado no Primeiro Segundo (VEF1);
- Pico de Fluxo Expiratório (PFE);
- Fluxo Expiratório Forçado Médio (FEF25-75%);
- Ventilação Voluntária Máxima (VVM).

Estas variáveis expressam a relação percentual do valor obtido pelo valor normal esperado para cada padrão de indivíduo (considerando idade e sexo) (OLIVEIRA et al., 2006). Estas medidas foram obtidas do Laboratório de Fisioterapia Pulmonar da Universidade Estadual de Londrina.

O espaço de busca, ou seja, o intervalo experimental de valores dos parâmetros onde o método tentará localizar a melhor solução é mostrado na tabela 5.

Parâmetros	Intervalo
Taxa de Aprendizagem	0-1
Constante de Momento	0-1
Número de Neurônios Ocultos	2-13

Tabela 5: Intervalo dos parâmetros otimizados

Os valores da tabela foram definidos com base em heurísticas e recomendações encontradas na literatura. Para a taxa de aprendizagem e constante de momento, os valores são aqueles recomendados por (FU, 1994 apud BASHEER; HAJMEER, 2000) e o número de neurônios ocultos foram obtidos pela heurística apresentada por (JADID; FAIRBAIRN, 1996), que determina o limite superior do NNO.

O problema de otimização de parâmetros da rede MLP foi realizado sob três diferentes técnicas: AG, PSO e ACO, sendo necessário a modelagem do problema com três ênfases distintas, as quais são detalhadas a seguir.

7.1.1 Modelagem com Algoritmos Genéticos

Para o algoritmo genético a principal dificuldade era trabalhar simultaneamente com três variáveis, ou seja, o cromossomo deveria conter informações sobre as três variáveis. Uma solução reportada na literatura foi utilizada, que consiste em codificar as variáveis e depois concatená-las, formando o cromossomo. Foi utilizada a codificação binária com *crossover* de um ponto e mutação que seleciona randomicamente um gene e inverte seu valor. Para testes foi utilizada uma taxa de *crossover* igual a 80% e para mutação taxa de 1%.

7.1.2 Modelagem com PSO

Para o PSO foi utilizado o conceito de busca no espaço de três dimensões onde cada dimensão representa uma variável do problema. Sendo assim o algoritmo PSO tenta encontrar a melhor posição (configuração ótima) dentro de um espaço tri-dimensional.

7.1.3 Modelagem com ACO

Existem diversos algoritmos pertencentes a metodologia de Otimização por Colônia de Formigas encontradas na literatura, dentre eles o *Ant System* que foi o primeiro algoritmo ACO proposto (COLORNI; DORIGO; MANIEZZO, 1992). Neste trabalho o *Ant System* é utilizado.

A novo modelo de otimização de parâmetros utilizando ACO, denominado de ASRNA (*Ant System* para Redes Neurais Artificiais), foi proposto neste trabalho, sendo apresentado na seção a seguir.

7.1.3.1 Modelo ASRNA

A principal dificuldade da utilização do AS é a modelagem do problema investigado como um problema de encontrar menores caminhos em grafos. Neste contexto, foi criado um grafo a partir dos possíveis valores de cada parâmetro da rede neural, como mostra a figura 34.

Na figura 34 os identificadores TA, CM e NNO representam taxa de aprendizagem, constante de momento e número de neurônios na camada oculta respectivamente e os números identificam os diversos valores possíveis para cada parâmetro dentro do intervalo definido.

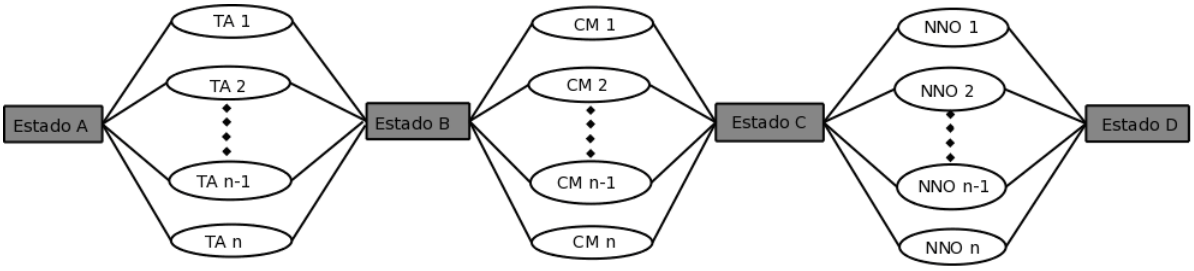

Figura 34: Grafo construído a partir dos parâmetros a serem otimizados

Seja $\tau(t)_{ij}$ a quantidade de ferormônio presente na trilha entre os estados i e j no tempo t. Inicialmente todas as formigas estão no estado A, a quantidade de ferormônios em cada trilha é inicialmente zero, ou seja, $\tau(0)_{ij}$. A partir disso, temos que a probabilidade de uma formiga escolher entre as n trilhas entre o estado A e o estado B é igual para todas n trilhas. Isto ocorre da mesma maneira entre as trilhas do estado B-C e C-D.

O cálculo do custo da trajetória de cada formiga é a porcentagem de acertos que a rede neural obteve utilizando a configuração, que equivale aos parâmetros presentes em cada trilha que determinada formiga utilizou. A figura 35 mostra uma possível rota realizada por uma determinada formiga k.

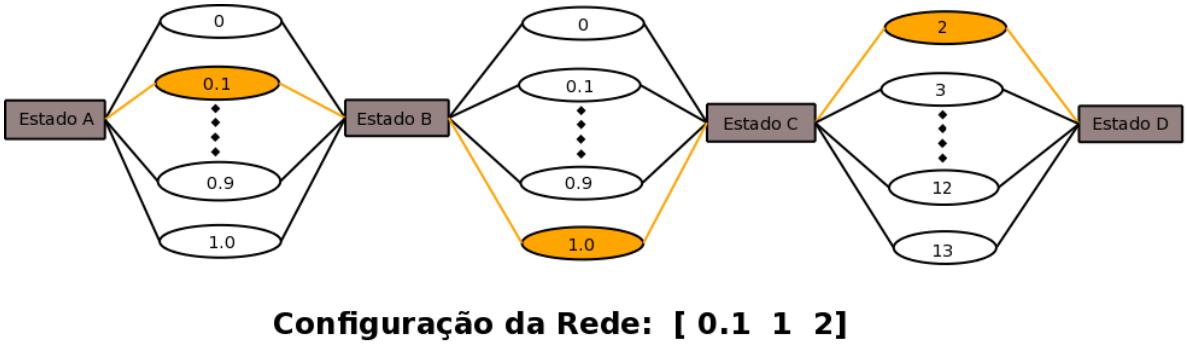

Figura 35: Exemplo de rota de uma determinada formiga k

Neste exemplo, a configuração da rede é [0.1 1 2], que são respectivamente os valores da taxa de aprendizagem, constante de momento e número de neurônios na camada oculta, que serão submetidos a RNA a fim de verificar o desempenho obtido com tal configuração.

Depois de realizado o cálculo dos custos do caminho percorrido por cada formiga, elas são novamente colocadas no estado A, simulando o processo natural de que novas formigas utilizariam o caminho para buscar mais alimento, e percorrendo o caminho novamente até chegarem ao estado D (fonte de alimento). Essa seqüência é realizada até que todas as formigas utilizem um mesmo caminho, ou seja, o sistema convergiu para uma solução

ótima ou próxima dela.

A atualização da quantidade de ferormônio depositado na trilha entre os estados i e j é realizada de acordo com a Eq. (7.1).

$$\tau(t+1)_{ij} \leftarrow (1-\rho) \cdot \tau(t)_{ij} + \sum_{k-1}^{m} \Delta\tau_{ij}^{k} \tag{7.1}$$

onde ρ é a taxa de evaporação, m é o número de formigas que utilizaram esta trilha e $\Delta\tau_{ij}^{k}$ é a quantidade de ferormônio deixado na trilha entre os estados i e j pela formiga k, que é calculada conforme a Eq. (7.2).

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{A_k}{Q} & \textit{, se a formiga } k \textit{ utilizou esta trilha} \\ 0 & \textit{, caso contrário} \end{cases} \tag{7.2}$$

onde A_k representa a acurácia obtida pela RNA utilizando a configuração gerada pelo trajeto da formiga k e Q é uma constante, a mesma descrita no AS. O valor da taxa de evaporação foi igual a 0.7 e para constante Q foi utilizado valor igual a 100, valores estes que foram identificados por (COLORNI; DORIGO; MANIEZZO, 1992), como bons parâmetros para o AS.

Na construção do conjunto de parâmetros resultante, ou seja, uma possível solução, as formigas vão selecionando as próximas trilhas a serem seguidas por meio de um processo estocástico. A probabilidade da formiga k utilizar uma determinada trilha entre os estados i e j é dado pela Eq. (7.3), onde n é o número de trilhas possíveis entre o estado i e j.

$$p_{ij}^{k} = \frac{\tau_{ij}}{\sum_{c=0}^{n} \tau_{ij}^{c}} \tag{7.3}$$

No modelo proposto não é utilizada a idéia de visibilidade, pois não há o conceito de comprimento da trilha. Com o objetivo de preservar as configurações com menor quantidade de neurônios na camada oculta que obtiveram mesma taxa de acerto de outras configurações com maiores quantidades, foi adicionado à acurácia um valor inversamente proporcional ao NNO, que é obtido pela Eq. (7.4), onde x representa o NNO.

$$T(x) = \exp^{\left(-\frac{x}{10}\right)} \tag{7.4}$$

Esta heurística foi adicionada, devido ao alto tempo de processamento gasto com uma grande quantidade de neurônios ocultos. Sendo assim, o custo da trilha s é calculado conforme a Eq. (7.5), onde A_r representa a acurácia real, obtida com a execução da rede neural.

$$C_s = A_r + T \tag{7.5}$$

Outras variações do modelo proposto foram abordadas, a primeira chamada de ASRNA1 utiliza, assim como em Algoritmos Genéticos, um operador de Elitismo, para preservar os melhores resultados. Este operador foi implementado de forma simples, apenas armazenando em uma estrutura de dados as melhores soluções encontradas até o momento; a taxa de Elitismo aplicada foi de 20%.

Em outra variação, denominada ASRNA2, utiliza o conceito de pesos nas respostas da rede, quanto maior a acurácia da rede utilizando um determinado caminho, maior será a quantidade de ferormônio depositado nesta trilha, a função que determina o peso da resposta é descrita pela Eq. (7.6), onde x é resposta (acurácia) da rede.

$$P(X) = \exp^{\left(-\frac{x}{50}\right)} + 1.1353353 \tag{7.6}$$

Esta função foi obtida de maneira empírica, objetivando encontrar uma função que aumente a influência de boas soluções e reduza a influência de solução piores, no processo de decisão das formigas de escolher qual trilha será seguida. Assim sendo, o valor da resposta da rede é calculado conforme a Eq. (7.7), onde A_r é igual à acurácia real obtida pela RNA.

$$R(x) = P(x) + A_r \tag{7.7}$$

A última variação, denominada de ASRNA3, é um modelo híbrido do ASRNA1 e ASRNA2, contendo tanto a heurística de elitismo, quanto de peso nas respostas.

7.1.4 Resultados

Os resultados serão analisados sob diferentes perspectivas. Todos os modelos utilizaram uma mesma quantidade de indivíduos e um mesmo número de ciclos, ambos iguais a 15. A tabela 6 apresenta os resultados em relação aos erros de teste e treinamento.

	Erro Teste	Erro Treinamento
PSO	0.048632	0.056570
AG	0.044122	0.056302
ASRNA	0.069582	0.055637
ASRNA1	0.041850	0.060172
ASRNA2	0.051649	0.060000
ASRNA3	0.041389	0.058251

Tabela 6: Análise através dos erros de teste e treinamento

Pode-se verificar que o menor erro de teste foi obtido com o modelo ASRNA3. Já no erro de treinamento o menor valor foi verificado com o modelo ASRNA. Em relação à acurácia da RNA utilizando a configuração retornada por cada modelo testado, com base na tabela 7 é possível constatar que as configurações obtidas com os modelos ASRNA obtiveram uma alta taxa de predições corretas, principalmente os modelos ASRNA1, ASRNA2 e ASRNA3. Também se destaca o bom resultado obtido com o AG.

	Percentagem de predições corretas (Acurácia)
PSO	93.68%
AG	96.96%
ASRNA	87.16%
ASRNA1	94.01%
ASRNA2	94.10%
ASRNA3	94.24%

Tabela 7: Análise através da percentagem de predições corretas

Um fator importante a ser considerado é o número de configurações que o modelo testou para que chegasse ao valor máximo (solução ótima ou próxima dela), denominado aqui de *taxa de execução*. A tabela 8 apresenta estes valores, que são mensurados pela percentagem de configurações testadas em relação ao número total de combinações possíveis dos parâmetros, que neste caso é de (11 x 11 x 12) = 1452, que representam os valores possíveis da taxa de aprendizagem, constante de momento e número de neurônios na camada oculta respectivamente.

	Taxa de Execução
PSO	5.92%
AG	6%
ASRNA	6.75%
ASRNA1	7.5%
ASRNA2	6.33%
ASRNA3	5.17%

Tabela 8: Análise através da taxa de execução

É possível identificar que o modelo ASRNA3 obteve a solução realizando um menor número de testes. Destaca-se também a baixa taxa de execução do modelo PSO.

7.2 Estudo de Caso 2

O problema de classificação de padrões é aqui abordado. O problema consiste em classificar tumores de mama entre benigno ou maligno, com base em nove características distintas.

Os dados deste problema foram obtidos de uma base de casos, disponibilizadas pelo UCI (ASUNCION; NEWMAN, 2007) (Machine Learning Repository) da Universidade da Califórnia. Esta base de câncer de mama foi obtida da Universidade de Hospitais de *Wisconsin*, Madison - USA, através de relatos de casos clínicos de pacientes do Dr. WIlliam H. Wolberg. A tabela 9 apresenta as características consideradas e o domínio de cada uma.

Atributo	Domínio
Sample code number	id number
Clump Thickness	1-10
Uniformity of Cell Size	1-10
Uniformity of Cell Shape	1-10
Marginal Adhesion	1-10
Single Epithelial Cell Size	1-10
Bare Nuclei	1-10
Bland Chromatin	1-10
Normal Nucleoli	1-10
Mitoses	1-10
Class	benign,malignant

Tabela 9: Características do problema e seus respectivos domínios

Neste trabalho foi desconsiderado o "Sample code number", pois este atributo não acrescentaria conteúdo informativo ao domínio. Além disso, foram retirados padrões com um ou mais valores de atributos não informados, os padrões com *"missing values"*, sendo assim

do total de 699 instâncias disponibilizadas pelo UCI, foram utilizados 683 padrões.

As técnicas de aprendizagem de máquina utilizadas neste problema foram: Redes Neurais, Máquina de Vetor Suporte e Rede Bayesiana, mais precisamente o classificador *Naive Bayes*.

A rede neural do tipo *Multilayer Perceptron (MLP)* treinada com algoritmo *backpropagation* foi utilizada. Alguns parâmetros da rede, como a taxa de aprendizagem, constante de momento e número de neurônios na camada oculta foram otimizados utilizando a técnica de ASRNA descrito em (GONÇALVES; CAMARGO-BRUNETO, 2008c). Os valores obtidos foram:

- Taxa de aprendizagem(η) = 0.5;
- Constante de momento(μ) = 0.7;
- Número de neurônios ocultos = 3;

Foram utilizadas 50 épocas de treinamento, obtidas pelo método descrito em (GONÇALVES; CAMARGO-BRUNETO, 2008b). O erro aceitável, entre o valor esperado y_i e o obtido d_i, foi de 0.2. Além disso, para medir se o padrão representa bem o domínio, foi utilizado um percentual de confiabilidade. Após a rede treinada aplicou-se o mesmo padrão n vezes. Se ele for classificado incorretamente mais do que a porcentagem de confiabilidade aceita, o padrão é dito não confiável e o mesmo é descartado. Foi utilizado um valor de porcentagem igual a 0.2, ou seja 20%.

A máquina de vetor suporte foi implementada por meio de uma Rede RBF. Neste modelo utiliza-se a função *Kernel* RBF. O número de funções base radiais e seus centros são determinados automaticamente pelo número de vetores suporte e pelos multiplicadores de Lagrange associados aos mesmos vetores (LORENA; CARVALHO, 2003b). Foram utilizados valor de largura $\sigma = 2$ e número de centros $C = 1000$.

Já para o classificador Bayesiano, *Naive Bayes*, foram atribuídas as probabilidades condicionais $P(Classe|Atributos)$ de valor zero, um valor pequeno da ordem $2 \cdot 10^{-5}$.

Todas as técnicas foram implementadas sobre o ambiente livre de programação numérica Scilab[1]. Também foi utilizado o método de validação cruzada do tipo K-fold, para melhor estimar a precisão das técnicas apresentadas, um valor de K igual a 10 foi utilizado.

[1] http://www.scilab.org

7.2.1 Resultados

Os resultados da aplicação das três técnicas são mostrados na tabela 10, onde são considerados a acurácia do método (número de predições corretas) e o tempo médio de execução.

Técnica	Acurácia (%)	Tempo de Execução (s)
Rede Neural MLP	98.98	55.30
Máquina de Vetor Suporte (RBF)	77.20	22.16
Naive Bayes	97.35	0.35

Tabela 10: Análise da acurácia e tempo de execução

Por meio da tabela 10 é possível identificar que todas as técnicas obtiveram boas acurácias, ou seja, obtiveram alta taxa de predições corretas sobre padrões desconhecidos. Destacando as Redes MLP e o *Naive Bayes*.

Com a alta acurácia do método *Naive Bayes*, há indícios de que as variáveis do problema são independentes, ou seja, as variáveis não influenciam umas as outras.

Em relação ao tempo de execução, destaca-se o baixo tempo do algoritmo *Naive Bayes*. Essa característica se deve ao fato da simplicidade do algoritmo, comparado as duas outras técnicas. Estes valores correspondem ao tempo médio entre as K iterações do método $K - fold$.

A partir deste estudo de caso foi possível ratificar a robustez e eficiência das técnicas de aprendizagem de máquina, em problemas de classificação ressaltando que este é um caso específico e que estas técnicas podem não ter o mesmo desempenho quando aplicadas em outros problemas.

Conclusão

Este trabalho apresentou os fundamentos, paradigmas e algumas das principais técnicas de Aprendizagem de Máquina (AM). Área esta que visa desenvolver programas capazes de aprender conceitos ou hipóteses a partir de um conjunto de exemplos ou casos observados.

Enraizada em diversas áreas, a AM se utiliza de diversas teorias e procedimentos de diferentes áreas do conhecimento. Dentre elas destacam-se: estatística, física, mecânica, biologia, medicina, entre outras.

O uso das técnicas de AM é indicado quando:

- Há um grande volume de dados, no qual se deseja extrair algum tipo de informação;
- Dados complexos e/ou de alta dimensionalidade, onde métodos clássicos, geralmente, não se mostram eficientes;
- Há necessidade de extrair regras do tipo "Se-Então";
- Soluções aproximadas são aceitáveis.

Várias aplicações de AM foram apresentadas no decorrer deste trabalho, sendo algumas técnicas destinadas a problemas específicos. De maneira geral, AM são aplicadas em:

- Problemas de Otimização;
- Problemas de Classificação de Padrões;
- Visão Computacional;
- Regressão Linear e Não-Linear;
- Previsão;
- Sistemas Controle Adaptativo; entre outros.

Algumas das aplicações acima descritas foram realizadas neste trabalho. As principais contribuições deste trabalho foram:

- Apresentar os diferentes paradigmas de aprendizagem de máquina;
- Mostrar pontos fortes e as limitações das principais técnicas de AM;
- Discutir sobre a aplicação destas técnicas em problemas reais;
- Elaboração de estudos de caso;
- Desenvolvimento de programas de computador para mostrar a aplicabilidade destas técnicas.

Não é possível afirmar que uma determinada técnica supera as outras em todos os problemas. Uma abordagem comumente utilizada e a implementação de diferentes algoritmos, estimar as acurácias e selecionar o algoritmo com melhor desempenho. Também existem sistemas que utilizam mais de uma técnica ao mesmo tempo, se aproveitando do melhor que cada uma possui, cobrindo a fragilidade de uma com o apoio de outra. Estes sistemas são chamados de *comitês de máquinas*.

Mesmo com os bem sucedidos resultados obtidos com a aplicação das técnicas de AM, algumas questões ainda estão em aberto, sendo necessário um maior aprofundamento teórico para identificar até onde estas técnicas podem chegar.

Embora esta área possua algumas deficiências, várias metodologias foram propostas com o intuito de minimizá-las. Em conjunto com suas características atrativas formam poderosas ferramentas, fazendo com que sejam exploradas em diversos problemas reais.

Referências

ACHARYA, U. R. et al. Classification of heart rate data using aritificial neural network and fuzzy equivalence relation. *Pattern Recognition*, v. 36, p. 61–68, 2003.

AMARI, S. et al. Asymptotic statistical theory of overtraining and cross-validation. In: *IEEE Transactions on Neural Networks*. New York: [s.n.], 1997. v. 5, n. 8, p. 985–996.

ANDROUTSOPOULOS, I. et al. An evaluation of naive bayesian anti-spam filtering. In: *Workshop on Machine Learning in the New Information Age*. [s.n.], 2000. p. 9–17. Disponível em: <http://arxiv.org/abs/cs.CL/0006013>.

ARBIB, M. A. *The Handbook Of Brain Theory and Neural Networks*. 2. ed. London, England: The MIT Press, 2002.

ASUNCION, A.; NEWMAN, D. *UCI Machine Learning Repository*. 2007. Disponível em: <http://www.ics.uci.edu/~mlearn/MLRepository.html>.

BARANAUSKAS, J. A.; MONARD, M. C. *Reviewing Some Machine Learning Concepts and Methods*. São Carlos - SP, Fevereiro 2000.

BASHEER, I. A.; HAJMEER, M. Artificial neural networks: fundamentals, computing, design, and application. *Journal of Microbiological Methods*, v. 43, p. 3–31, 2000.

BASYE, T. D. K.; VITTER, J. S. Coping with uncertainty in map learning. *Machine Learning*, Springer Netherlands, v. 29, n. 1, October 1997.

BLUM, A.; MITCHELL, T. Combining labeled and unlabeled data with co-training. In: *COLT' 98: Proceedings of the eleventh annual conference on Computational learning theory*. [S.l.]: Morgan Kaufmann Publishers, 1998. p. 92–100.

CARVALHO, F. *Aprendizagem Bayesiana*. Apresentação de Slides. Acessado em: 05 de Outubro de 2008. Disponível em: <http://www.cin.ufpe.br/~compint/aulas-IAS/kdd-011-/Bayes.ppt>.

CENTENO, T. M. et al. Applications on evolutionary computing. In: _____. [S.l.]: Springerlink, 2005. cap. Object Detection for Computer Vision Using a Robust Genetic Algorithm, p. 284–293.

CHARNIAK, E. Bayesian networks without tears: making bayesian networks more accessible to the probabilistically unsophisticated. *AI Mag.*, American Association for Artificial Intelligence, Menlo Park, CA, USA, v. 12, n. 4, p. 50–63, 1991. ISSN 0738-4602. Disponível em: <http://portal.acm.org/citation.cfm?id=122716>.

CHENG, J.; GREINER, R. Comparing bayesian network classifiers. In: . Morgan Kaufmann Publishers, 1999. p. 101–108. Disponível em: <http://citeseer.ist.psu.edu/115216.html>.

COLORNI, A.; DORIGO, M.; MANIEZZO, V. Distributed optimization by ant colonies. In: *Proceedings European Conference on Artificial Life*. Paris, França: Elsevier, Amsterdam, 1992. p. 134–142. Disponível em: <http://iridia.ulb.ac.be/~mdorigo/ACO% -/publications.html>.

COOPER, G. F.; HERSKOVITS, E. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, v. 09, n. 4, p. 309–347, October 1992. Disponível em: <http://www.springerlink.com/content/t2k011n123r16831/fulltext.pdf>.

COZMAN, F. G. Generalizing variable elimination in bayesian networks. In: *In Workshop on Probabilistic Reasoning in Artificial Intelligence*. [S.l.: s.n.], 2000. p. 27–32.

CRAMÉR, H. *Elementos da Teoria da Probabilidade e algumas de suas aplicações*. São Paulo: Mestre Jou, 1955.

DAI, Y. S. et al. Optimal testing-resource allocation with genetic algorithm for modular software systems. *Journal of Systems and Software*, v. 66, n. 1, p. 47–55, 2003.

DEJONG, K. A. *An Analisysof the Behavior of a Class of Genetic Adaptive Systems*. Tese (Doutorado) — University of Michigan, 1975.

DENEUBOURG, J.-L. et al. The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior*, v. 76, p. 159, 1990.

DING, C. H.; DUBCHAK, I. Multi-class protein fold recognition using support vector machines and neural networks. *Bioinformatics*, v. 17, n. 4, p. 349–358, 2001.

DORIGO, M.; BIRATTARI, M.; STUTZLE, T. Ant colony optimization: Artificial ants as a computational intelligence technique. *Computational Intelligence Magazine, IEEE*, v. 1, n. 4, p. 28–39, 2006. Disponível em: <"http://ieeexplore.ieee.org/xpls/abs_all-.jsp?arnumber=4129846">.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, v. 26, n. 1, p. 29–41, 1996.

DRÉO, J. et al. *Metaheuristics for Hard Optimization: Methods and Case Studies*. [S.l.]: Springer, 2005.

DUDA, R. O.; HART, P. E. *Pattern Classification and Scene Analysis*. [S.l.]: John Wiley Sons Inc, 1973. Hardcover.

FERRO, M. *Aquisição de conhecimento de conjuntos de exemplos no formato atributo valor utilizando aprendizado de máquina relacional*. Dissertação (Mestrado) — ICMC-USP, 2004.

FIDELIS, M. V.; LOPES, H. S.; FREITAS, A. A. Discovering comprehensible classification rules with a genetic algorithm. In: *Proceedings of the 2000 Congress on Evolutionary Computation*. [S.l.: s.n.], 2000. p. 805–810.

FRIEDMAN, J. H. *Another approach to polychotomous classification*. [S.l.], 1996. Disponível em: <http://www-stat.stanford.edu/~jhf/ftp/poly.ps.Z>.

FRIEDMAN, N.; GEIGER, D.; GOLDSZMIDT, M. Bayesian network classifiers. *Machine Learning*, v. 29, n. 2-3, p. 131–163, 1997. Disponível em: <http://citeseer.ist.psu.edu-/friedman97bayesian.html>.

FU, L. *Neural Networks in Computer Intelligence*. New York: MacGraw-Hill, 1994.

GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. [S.l.]: Addison-Wesley, 1989.

GOLDMAN, R. *Probabilistic Approach to Language Understanding*. [S.l.], 1990.

GONÇALVES, A. R.; CAMARGO-BRUNETO, M. A. de O. Aplicação de modelos computacionais evolutivos na otimização de parâmetros de uma rede neural multilayer perceptron treinada via backpropagation. In: *Aceito para: XVII Encontro Anual de Iniciação Científica*. Foz do Iguaçu, Paraná: [s.n.], 2008.

GONÇALVES, A. R.; CAMARGO-BRUNETO, M. A. de O. Automatização do processo de determinação do número de ciclos de treinamento de uma rede neural artificial. In: *Aceito para: II Congresso Nacional de Extensão Universitária e XI Encontro de Atividades Científicas da Unopar*. Londrina, Paraná: [s.n.], 2008.

GONÇALVES, A. R.; CAMARGO-BRUNETO, M. A. de O. Um novo modelo híbrido baseado em otimização por colônia de formigas e redes neurais para identificação de indivíduos com dpoc. In: *Aceito para: XI Congresso Brasileiro de Informática Aplicada a Saúde*. Campos do Jordão, São Paulo: [s.n.], 2008.

GUPTA, M. M.; JIN, L.; HOMMA, N. *Static and Dynamic Neural Networks: From Fundamentals to Advanced Theory*. [S.l.]: Wiley-IEEE Press, 2003.

HAN, S.-S.; MAY, G. S. Optimization of neural network structure and learning parameters using genetic algorithms. In: *ICTAI '96: Proceedings of the 8th International Conference on Tools with Artificial Intelligence (ICTAI '96)*. Washington, DC, USA: IEEE Computer Society, 1996. p. 200.

HARPHAM, C.; DAWSON, C.; BROWN, M. A review of genetic algorithms applied to training radial basis function networks. *Neural Computing & Applications*, Springer, v. 13, n. 3, p. 193–201, September 2004. ISSN 0941-0643. Disponível em: <http://dx.doi.org/10-.1007/s00521-004-0404-5>.

HAYKIN, S. *Redes Neurais, Princípios e prática*. 2. ed. [S.l.]: Bookman, 1999.

HAZZAN, S.; IEZZI, G. *Fundamentos de Matemática Elementar vol. 5*. [S.l.]: Atual, 2004.

HEARST, M. A. et al. Support vector machines. *IEEE Intelligent Systems*, IEEE Computer Society, Los Alamitos, CA, USA, v. 13, n. 4, p. 18–28, 1998. ISSN 1094-7167.

HEBB, D. O. *The Organization of Behavior: a neuropsychological theory*. New York: Willey, 1949.

HECKERMAN, D. E. *Probabilistic Similarity Networks*. [S.l.], 1990.

HINTON, G. E.; ACKLEY, D. H.; SEJNOWSKI, T. J. A learning algorithm for boltzmann machines. *Cognitive Science*, v. 9, p. 147–169, 1985.

HOLLAND, J. H. *Adaptation in Natural and Artificial Systems*. 2. ed. [S.l.]: The MIT Press, 1975.

HOPFIELD, J. J. Neural networks and physical systems with emergent collective computational abilities. *Biophysics*, v. 79, p. 2554–2558, Abril 1982.

HOPFIELD, J. J. Neurons with graded response have collecitve computational properties like those of two state neurons. *Biophysics*, v. 81, p. 3088–3092, Maio 1984.

HOPFIELD, J. J.; TANK, D. W. Computing with a neural circuits: a model. *Science*, v. 233, n. 4764, p. 625–633, Agosto 1986.

HRUSCHKA JR., E. R. *Imputação Bayesiana no contexto da Mineração de Dados*. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, Rio de Janeiro, Outubro 2003. Disponível em: <http://www.coc.ufrj.br/teses/doutorado/inter/2003/teses-/HRUSCHKA%20JUNIOR_ER_03_t_D_int.pdf>.

IYODA, E. M. *Inteligência Computacional no Projeto Automático de Redes Neurais Híbridas e Redes Neurofuzzy Heterogêneas*. Dissertação (Mestrado) — Universidade Estadual de Campinas (Unicamp), Campinas, São Paulo, 2000. Disponível em: <http://www.dca.fee-.unicamp.br/~vonzuben/research/emi_mest.html>.

JADID, M. N.; FAIRBAIRN, D. R. Neural network applications in predicting moment-curvature parameters from experimental data. *Engineering Applications of Artificial Intelligence*, v. 9, 1996.

JAIN, A. K.; MAO, J.; MOHIUDDIN, K. Artificial neural networks: A tutorial. *IEEE Computer*, v. 29, p. 31–44, 1996.

JAIN, A. K.; MURTY, M. N.; FLYNN, P. J. Data clustering: a review. *ACM Comput. Surv.*, ACM Press, v. 31, n. 3, p. 264–323, September 1999. Disponível em: <http://dx.doi.org-/10.1145/331499.331504>.

KENNEDY, J. The particle swarm: social adaptation of knowledge. In: *IEEE International Conference on Evolutionary Computation*. [S.l.: s.n.], 1997. p. 303–308.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: *Proceedings IEEE International Conference on Neural Networks*. [s.n.], 1995. v. 4, p. 1942–1948. Disponível em: <http://dx.doi.org/10.1109/ICNN.1995.488968>.

KOHAVI, R. A study a cross validation a bootstrap for accuracy estimation and a model selection. In: *International Joint Conference on Artificial Intelligence (IJCAI)*. [S.l.: s.n.], 1995.

KOHONEN, T. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, v. 43, p. 59–69, 1982.

KOVáCS, Z. L. *Redes Neurais Artificiais: fundamentos e aplicações*. São Paulo: Acadêmica, 1996.

LACERDA, E. G. M.; CARVALHO, A. C. Sistemas inteligentes: aplicações a recursos hídricos e ciências ambientais. In: _____. Porto Alegre, RS: Universidade/UFRGS, 1999. cap. Introdução aos Algoritmos Genéticos, p. 99–150. Disponível em: <http://www.dca.ufrn.br-/~estefane/metaheuristicas/ag.pdf>.

LEVITT, T. S.; AGOSTA, J. M.; BINFORD, T. O. Model-based influence diagrams for machine vision. In: *UAI '89: Proceedings of the Fifth Annual Conference on Uncertainty in Artificial Intelligence*. Amsterdam, The Netherlands: North-Holland Publishing Co., 1990. p. 371–388. ISBN 0-444-88738-5.

LIANG, J. J. et al. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *Evolutionary Computation, IEEE Transactions on*, v. 10, n. 3, p. 281–295, 2006.

LIMA, A. R. G. *Máquinas de Vetores Suporte na Classificação de Impressões Digitais*. Dissertação (Mestrado) — Universidade Federal do Ceará, Fortaleza, Ceará, 2002.

LIPPMANN, R. An introduction to computing with neural nets. *ASSP Magazine, IEEE [see also IEEE Signal Processing Magazine]*, v. 4, n. 2, p. 4–22, 1987.

LIPSCHUTZ, S. *Probabilidade*. 4. ed. São Paulo: Makron Books, 1993.

LONG, W. J.; FRASER, H. S. F.; NAIMI, S. Reasoning requirements for diagnosis of heart disease. *Artificial Intelligence in Medicine*, v. 10, n. 1, p. 5–24, 1997. Disponível em: <http://citeseer.ist.psu.edu/william97reasoning.html>.

LOPES, H. S. *Fundamentos da Computação Evolucionária e Aplicações*. Bandeirantes,Paraná, 2006. 52-106 p.

LORENA, A. C.; CARVALHO, A. C. P. L. de. *Introdução aos Classificadores de Margens Largas*. São Carlos - SP, Maio 2003.

LORENA, A. C.; CARVALHO, A. C. P. L. de. *Introdução às Máquinas de Vetores Suporte*. São Carlos - SP, Abril 2003.

MA, S.; HUANG, J. Penalized feature selection and classification in bioinformatics. *Brief Bioinform*, v. 9, n. 5, p. 392–403, September 2008. Disponível em: <http://dx.doi.org/10-.1093/bib/bbn027>.

MARQUES, R. L.; DUTRA, I. *Redes Bayesianas: o que são, para que servem, algoritmos e exemplos de aplicações*. Rio de Janeiro: [s.n.], 2008. Disponível em: <www.cos.ufrj.br/~ines-/courses/cos740/leila/cos740/Bayesianas.pdf>.

MATSUBARA, E. T. *O algoritmo de aprendizado semi-supervisionado co-training e sua aplicação na rotulação de documentos*. Dissertação (Mestrado) — Instituto de Ciências Matemáticas e de Computação (ICMC) - USP, 2004. Disponível em: <http://www.teses-.usp.br/teses/disponiveis/55/55134/tde-19082004-092311/>.

MCCALLUM, A.; NIGAM, K. *A comparison of event models for Naive Bayes text classification*. 1998. Disponível em: <http://citeseer.ist.psu.edu/489994.html>.

MCCULLOCH, W. S.; PITTS, W. H. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, v. 5, p. 115–133, 1943.

MEDEIROS, J. A. C. C. *Enxame de Partículas como ferramenta de Otimização em Problemas Complexos da Engenharia Nuclear*. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, 2005.

MEYER, P. L. *Probabilidade: Aplicações à Estatística*. 2. ed. [S.l.]: LTC, 2000.

MICHALEWICZ, Z. *Genetics Algorithms + Data Structures = Evolution Programs*. 3. ed. New York: Springer-Verlag Berlin Hildelberg, 1996.

MITCHELL, M. *An Introduction to Genetic Algorithms (Complex Adaptive Systems)*. London, England: The MIT Press, 1996.

MITCHELL, T. M. *Machine Learning*. [S.l.]: McGraw-Hill, 1997.

MORGADO, A. C. et al. *Análise Combinatória e Probabilidade*. Rio de Janeiro: SBM, 2001.

NEAPOLITAN, R. E. *Learning Bayesian Networks*. [S.l.]: Prentice Hall, 2003.

NODA, E.; FREITAS, A. A.; LOPES, H. S. Comparing a genetic algorithm with a rule induction algorithm in the data mining task of dependence modeling. In: *Genetic and Evolutionary Coomputation Conference*. [S.l.: s.n.], 2000. p. 1080.

OBITKO, M. *Introduction to Genetic Algorithms*. 1998. Website. Disponível em: <http://obitko.com/tutorials/genetic-algorithms>.

OLIVEIRA, C. et al. Identificação de doença pulmonar obstrutiva crônica através de redes neurais artificiais. *Anais do XX Congresso Brasileiro de Engenharia Biomédica / II Congresso Brasileiro de Engenharia Clínica*, p. 262–265, 2006.

PARPINELLI, R. S.; LOPES, H. S.; FREITAS, A. A. Data mining with an ant colony optimization algorithm. *Evolutionary Computation, IEEE Transactions on*, v. 6, n. 4, p. 321–332, 2002.

PAULINO, C. D.; TURKMAN, M. A. A.; MURTEIRA, B. *Estatística Bayesiana*. Lisboa: Fundação Calouste Gulbenkian, 2003.

PEARL, J. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. [S.l.]: Morgan Kaufmann, 1988. Paperback.

PENG, F.; SCHUURMANS, D. *Combining Naive Bayes and n-Gram Language Models for Text Classification*. 2003. Disponível em: <http://citeseer.ist.psu.edu/572782.html>.

PLATT, J.; CRISTIANINI, N.; SHAWE-TAYLOR, J. Large margin dags for multiclass classification. In: SOLLA, S.; LEEN, T.; MUELLER, K.-R. (Ed.). *Advances in Neural Information Processing Systems 12*. [S.l.: s.n.], 2000. p. 547–553.

RAHNAMAYAN, S.; TIZHOOSH, H.; SALAMA, M. Adaptive and natural computing algorithms. In: _____. [S.l.]: Springerlink, 2005. cap. Learning Image Filtering from a Gold Sample Based on Genetic Optimization of Morphological Processing, p. 478 – 481.

REIS, M. C. *A unidade básica do Sistema Nervoso: Neurónio*. 2008. Website.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 1958.

RUSSELL, S. J.; NORVIG, P. *Artificial Intelligence: a modern approach*. New Jersey: Prentice Hall, 1995.

SHMYGELSKA, A.; HOOS, H. An ant colony optimisation algorithm for the 2d and 3d hydrophobic polar protein folding problem. *BMC Bioinformatics*, v. 6, n. 1, 2005. Disponível em: <http://dx.doi.org/10.1186/1471-2105-6-30>.

SIM, K. M.; SUN, W. H. Ant colony optimization for routing and load-balancing: survey and new directions. *Systems, Man and Cybernetics, Part A, IEEE Transactions on*, v. 33, n. 5, p. 560–572, 2003.

SMOLA, A. J. et al. *Advances in Large Margin Classifiers*. [S.l.]: Morgan-Kauffman, 1999.

SMOLA, A. J. et al. Introduction to large margin classifiers. In: _____. [S.l.]: Morgan-Kauffman, 1999. cap. 1, p. 1–28.

SOCHA, K.; SAMPELS, M.; MANFRIN, M. Ant algorithms for the university course timetabling problem with regard to the state-of-the-art. In: *In Proc. Third European Workshop on Evolutionary Computation in Combinatorial Optimization (EvoCOP 2003*. [S.l.]: Springer Verlag, 2003. p. 334–345.

SOUSA, T.; SILVA, A.; NEVES, A. Particle swarm based data mining algorithms for classification tasks. *Parallel Computing*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 30, n. 5-6, p. 767–783, 2004.

SOUTO, M. de et al. Técnicas de aprendizado de máquina para problemas de biologia molecular. Universidade de São Paulo. 2003.

SUKTHOMYA, W.; TANNOCK, J. The optimisation of neural network parameters using taguchiś design of experiments approach: an application in manufacturing process modeling. *Neural Computing & Applications*, Amsterdam, n. 14, p. 337–344, 2005.

SUN, Z.; SUN, Y. Fuzzy support vector machine for regression estimation. In: *Systems, Man and Cybernetics, 2003. IEEE International Conference on*. [S.l.: s.n.], 2003. p. 3336–3341.

SUNG, A. H.; MUKKAMALA, S. Identifying important features for intrusion detection using support vector machines and neural networks. *Applications and the Internet, 2003. Proceedings. 2003 Symposium on*, p. 209–216, January 2003.

TANOMARU, J. Motivação, fundamentos e aplicações de algoritmos genéticos. *Anais do II Congresso Brasileiro de Redes Neurais*, 1995.

TONG, S.; KOLLER, D. Support vector machine active learning with applications to text classification. In: *Proceedings of ICML-00, 17th International Conference on Machine Learning*. Stanford, US: Morgan Kaufmann Publishers, San Francisco, US, 2000. p. 999–1006. Disponível em: <http://citeseer.ist.psu.edu/tong00support% -.html>.

UTZLE, S.; HOOS, T. *Improvements on the ant system: Introducing MAXMIN ant system*. 1997. Disponível em: <http://citeseer.ist.psu.edu/278024.html>.

VAPNIK, V. N. *The nature of statistical learning theory*. New York, NY, USA: Springer-Verlag New York, Inc., 1995. ISBN 0387945598. Disponível em: <http://portal.acm.org/citation-.cfm?id=211359>.

VON ZUBEN, F.; ATTUX, R. R. *Redes Neurais com Função de Base Radial*. 2008. Disponível em: <ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ia353_1s07/topico9_07.pdf>.

WEGENER, J. et al. Testing real-time systems using genetic algorithms. *Software Quality Journal*, v. 6, n. 2, p. 127–135, 1997.

WILSON, E. O. *Sociobiology: The New Synthesis*. Cambridge,MA: Harvard University Press, 1975.

YOUSEF, M. et al. Combining multi-species genomic data for microrna identification using a naive bayes classifier machine learning for identification of microrna genes. *Bioinformatics*, The Wistar Institute, Philadelphia, PA 19104, USA., March 2006. ISSN 1367-4803. Disponível em: <http://view.ncbi.nlm.nih.gov/pubmed/16543277>.

ZHANG, C.; SHAO, H.; LI, Y. Particle swarm optimisation for evolving artificial neural network. In: *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*. [s.n.], 2000. v. 4, p. 2487–2490 vol.4. Disponível em: <http://dx.doi.org/10.1109/ICSMC.2000.884366>.