Uma Abordagem Evolucionária Para o Aprendizado Semi-Supervisionado em Máquinas de Vetores de Suporte

Marcelo Mourão Silva

Universidade Federal de Minas Gerais
Escola de Engenharia
Programa de Pós-Graduação em Engenharia Elétrica

Uma Abordagem Evolucionária Para o Aprendizado Semi-Supervisionado em Máquinas de Vetores de Suporte

Marcelo Mourão Silva

Orientador: *Prof. Dr. Antônio de Pádua Braga*

Dissertação submetida à Banca Examinadora designada pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais - PPGEE/UFMG, como requisito parcial à obtenção do título de Mestre em Engenharia Elétrica.

Belo Horizonte
Novembro/2008

Ao Vô Mourão
e sua fábrica de doutores
e ao Vô Helvécio,
do seu xumbrega do brejo.

Agradecimentos

Quando, lá em 2003, saindo de uma aula de Sistemas Digitais, ouvi "Marcelo, passe na minha sala, por favor, que eu gostaria de conversar com você", não imaginaria o que essa conversa acarretaria. Entre aquele dia e este, em que termino esta dissertação, muita coisa aconteceu: inúmeros artigos lidos, várias páginas escritas, diversas reuniões de orientação... artigos rejeitados, mais artigos rejeitados... e artigos aceitos! Cresci bastante, como profissional e como pessoa, e muito disso se deve a um grande mestre que resolveu apostar naquele menino e mudou completamente a sua vida. Braga, muito obrigado pela oportunidade, pela enorme atenção que você me deu em todo o tempo em que trabalhamos juntos, e pelos incontáveis ensinamentos, que você transmite com uma naturalidade incrível!

Gostaria de agradecer também às mulheres da minha vida: mama Zezé, grande mulher que trabalha para que eu um dia me torne um grande homem; Vó Lulu, exemplo de vida que eu seguirei pra sempre; Vó Nely, carinho incondicional; Tia Alice, madrinha, mãe e amiga; e Gabi, presente maravilhoso que deu cor à minha vida.

Agradeço ao papai Beto, sábio que a cada dia se revela mais genial; ao Brou, muito mais que um irmão, um amigo para todas as horas; aos Vôs Helvécio e Mourão, que com certeza estão muito orgulhosos me assistindo de um camarote bem alto; ao Tio Tarcísio, padrinho que escolhi pela pessoa especial que é (e à afilhadinha Nina, que me escolheu sabe-se lá porquê!); aos incontáveis amigos que a vida me deu, que tornam o batente mais leve e os dias muito mais divertidos.

E à Mãozinha lá de cima, que sempre cuidou de mim!

Nothing is more practical than a good theory.

Vladimir Vapnik

Resumo

O paradigma de Aprendizado Semi-Supervisionado é bastante adequado a uma classe de problemas de crescente relevância no contexto do Aprendizado de Máquinas: aqueles onde há um grande desbalanceamento entre o conjunto de treinamento e o de teste, devido, entre outras coisas, ao alto custo de um classificador. Nessa classe de problemas, não se pode assegurar que os padrões rotulados representem adequadamente o sistema a ser aprendido, restringindo o uso do paradigma Indutivo Supervisionado. Utilizam-se, então, os padrões não-rotulados como fonte alternativa de informação sobre o problema a ser resolvido, garantindo maior capacidade de generalização à solução obtida.

As Máquinas de Vetores de Suporte (SVMs) são Redes Neurais Artificiais de ampla aceitação pela comunidade de Inteligência Computacional. Sua formulação baseada na Teoria do Aprendizado Estatístico e na maximização da margem de separação confere às SVMs altíssima capacidade de generalização.

As TSVMs (*Transductive Support Vector Machines*) ampliam a formulação das SVMs para a aplicação em problemas de aprendizado Semi-Supervisionado. Entretanto, a procura pelo conjunto de classificações que maximiza a margem de separação entre ambos os conjuntos de treinamento e de teste é realizada através de uma busca local exaustiva. A não-otimalidade desse processo motivou o desenvolvimento das GA3SVMs (*Genetic Algorithm Semi-Supervised Support Vector Machines*), propostas no presente trabalho.

Introduz-se, aqui, um Algoritmo Evolucionário na busca pelas classificações ótimas para os padrões de teste, de forma a induzir uma solução de separação máxima e alta capacidade de generalização. Um operador de mutação modificado, inspirado no método transdutivo *k-Nearest Neighbors*, é também apresentado, o qual adiciona informação ao processo de busca e acelera significativamente a convergência do Algoritmo Genético utilizado. Os resultados obtidos mostram a superioridade da metodologia proposta quando comparada às TSVMs tradicionais, para a classe de problemas estudada.

Abstract

The Semi-Supervised Learning paradigm is highly adequate for a class of problems with growing relevance in the context of Machine Learning: those in which there is a large unbalance between the training and the test data sets due to, among other things, the high cost of a classifier. In such class of problems, one cannot ensure that the labeled patterns appropriately represent the system to be learned, limiting the applicability of the Supervised Inductive paradigm. The unlabeled patterns are then used as an additional source of information about the problem being solved, providing increased generalization ability to the achieved solution.

The Support Vector Machines (SVMs) are Artificial Neural Networks widely accepted among the Computational Intelligence community. The formulation based on the Statistical Learning Theory and on the separating margin maximization provides the SVMs with extremely high generalization ability.

The TSVMs (Transductive Support Vector Machines) extend the SVMs formulation to the context of Semi-Supervised Learning. However, the search for the set of labels that maximize the separating margin between both the training and the test data is therein performed through an exhaustive local search. The non-optimality of such process motivates the development of the GA3SVMs (Genetic Algorithm Semi-Supervised Support Vector Machines), proposed in this piece.

An Evolutionary Algorithm is introduced in the search for the optima classifications for the test patterns, inducing a solution with maximum separating margin and high generalization ability. A modified mutation operator, based on the k-Nearest Neighbors transductive method, is also presented, which adds information to the search process and speeds up convergence significantly for the used Genetic Algorithm. Obtained results show the superiority of the proposed approach compared to the traditional TSVMs, for the class of problems studied.

Sumário

Lista de Abreviaturas

As principais abreviaturas utilizadas no presente trabalho encontram-se listadas abaixo.

k-NN	*k-Nearest Neighbor*
MCP	McCulloch & Pitts
RNA	Rede Neural Artificial
MLP	*Multi Layer Perceptron*
RBF	*Radial Basis Function*
MOBJ	Algoritmo Multiobjetivo
MSE	*Medium Squared Error*
VC	Vapnik-Chervonenkis
SI	Sistema Inteligente / *Swarm Intelligence*
DNA	Ácido desoxirribonucleico
GA	*Genetic Algorithm*
ACO	*Ant Colony Optimization*
PSO	*Particle Swarm Optimization*
i.i.d.	Independentes e identicamente distribuídos
SVM	*Support Vector Machine*
TSVM	*Transductive Support Vector Machine*
3SVM	*Semi-Supervised Support Vector Machine*
GA3SVM	*Genetic Algorithm Semi-Supervised Support Vector Machine*
GDMP	*Gene-Dependent Mutation Probability*
KFD	*Kernel Fisher Discriminant*

Lista de Símbolos

$\vec{x}$	Vetor de dados
$\vec{w}$	Vetor dos pesos
b	Termo de polarização
y	Saída da máquina de aprendizado
y_d	Saída desejada para a máquina de aprendizado
e	Erro da máquina de aprendizado
$f(.)$	Função de decisão ou disparo
N	Cardinalidade do conjunto de treinamento
n_{pop}	Número de indivíduos na população do GA
p_c	Probabilidade de cruzamento
p_m	Probabilidade de mutação
σ	Desvio-padrão do kernel RBF
ρ	Margem de separação entre duas classes
∇	Superfície linear de dimensão m
h	Dimensão VC da máquina de aprendizado
$\vec{\phi}(.)$	Mapeamento do espaço de entrada para o espaço de características
$K(.,.)$	Função de kernel da SVM
α	Multiplicador de Lagrange
$L(.)$	Função de perda de uma máquina de aprendizado
$J(.)$	Função Lagrangeana
ξ	Variável de folga da SVM
ξ^*	Variável de folga relativa a padrões de teste
C	Parâmetro regularizador da SVM
C^*	Parâmetro regularizador relativo aos padrões de teste
D	Matriz de distâncias entre os padrões de treinamento
$d_{i,j}$	Distância entre dois padrões de treinamento

Lista de Figuras

CAPÍTULO
1

Introdução

A ciência da *Inteligência Computacional* busca "estudar e desenhar agentes inteligentes" (Poole, Mackworth, & Goebel 1997). Um agente inteligente toma decisões apropriadas considerando as circunstâncias em que se encontra e objetivos que possui, mesmo com deficiências na percepção e na capacidade de cálculo, aprendendo com experiências passadas mas sendo flexível para se adaptar a ambientes dinâmicos e objetivos conflitantes (Poole, Mackworth, & Goebel 1997).

Tal definição é muitas vezes utilizada para o termo *Inteligência Artificial*. Entretanto, a utilização de tal termo traz algumas confusões, conforme discutido por Poole, Macworth e Goebel:

> Inteligência artificial é inteligência real? Talvez não, da mesma forma que uma pérola artificial é uma pérola falsa, e não uma pérola real. O nome “Inteligência Sintética” talvez fosse mais apropriado, assim como uma pérola sintética pode não ser uma pérola **natural**, mas é uma pérola **real**. Como buscamos “estudar e desenhar agentes inteligentes”, sejam eles naturais ou artificiais (ou sintéticos), preferimos o termo *Inteligência Computacional*. Outra vantagem da utilização de tal termo é que o agente computacional utilizado fica assim explícito.

O campo de *Aprendizado de Máquinas* é um sub-campo de Inteligência Computacional voltado ao desenvolvimento de algoritmos e técnicas que permitem às máquinas *aprender* (Alpaydin 2004).

Freqüentemente, a solução de problemas complexos por meio da construção de modelos para explicar o funcionamento de sistemas é uma tarefa extrema-

mente árdua e, muitas vezes, intratável dadas as restrições técnicas e tecnológicas ainda existentes. O objetivo da Aprendizagem de Máquinas é obter dispositivos que realizam tarefas complexas **aprendendo** a solução de um problema, e não **construindo-a** (Mika 2002).

Um exemplo de problema deste tipo, extraído de (Mika 2002), é o mapeamento do genoma humano. A comunidade de Ciências Biológicas e Genética compreende que há um mecanismo específico para o modo como o genoma armazena informações e como tais informações são decodificadas em algo útil. Sabe-se, ainda, que apenas partes do DNA, denominadas genes, são utilizadas nessa tarefa. Entretanto, ainda não se sabe, ao certo, onde os genes estão e como localizá-los (precisamente) ao longo do DNA. Os modelos biológicos disponíveis são extremamente complexos e explicam apenas parte do que está acontecendo. A Aprendizagem de Máquinas aborda esse problema de uma forma alternativa. Ao invés de buscar um modelo que explique corretamente os mecanismos implícitos à localização dos genes e possa, então, deduzir uma resposta, busca-se um conjunto de regras que seja capaz de, para cada pedaço de DNA, responder à seguinte pergunta: Isso é um gene ou não? De maneira mais genérica, a Aprendizagem de Máquinas aborda o problema de modelar relacionamentos entre objetos.

A dificuldade, agora, está em obter métodos que possibilitem às máquinas aprender. A complexidade de tal tarefa não é necessariamente menor que a solução do problema específico, mas traz uma vantagem significativa: uma vez resolvida, pode-se aplicar a mesma abordagem a uma imensa gama de problemas, diferentemente da solução específica, de aplicação limitada.

1.1 Motivação

Um dos paradigmas de aprendizado de máquinas mais utilizado é o de *Aprendizado por Exemplos*. A expectativa por trás dos métodos dessa classe é que, dado um conjunto restrito de observações de relacionamentos entre objetos (denominado conjunto de treinamento), pode-se estimar uma regra geral para estes relacionamentos. Os métodos dessa classe são usualmente denominados *Supervisionados* e *Indutivos*.

Entretanto, a obtenção de um conjunto de treinamento adequado à realização do aprendizado não é sempre uma tarefa trivial. Algumas vezes, a obtenção de parâmetros classificados (relacionamentos entre objetos) é demasiadamente cara ou complexa, limitando o tamanho dos conjuntos de treinamento. Em outras, não se pode garantir que a amostragem obtida representa adequadamente o sistema como um todo.

Outras classes de métodos, voltados exatamente à resolução de problemas

desse tipo, são os métodos *Semi-Supervisionados* e os *Transdutivos* (Cortes & Vapnik 1995; Mitchell 1999).

Um dos métodos dessa nova classe, de mais ampla aplicação e aceitação, é o algoritmo TSVM (*Transductive Support Vector Machine*), proposto por Joachims (Joachims 1999). Tal método estende as Máquinas de Vetores de Suporte, baseadas no aprendizado indutivo supervisionado, ao paradigma indutivo semi-supervisionado. Entretanto, o mecanismo pelo qual as TSVMs realizam tal procedimento é baseado em uma busca exaustiva local, abordagem não-ótima para a resolução de problemas complexos. O presente trabalho objetiva expandir a TSVM proposta por Joachims de forma a obter um algoritmo de maior capacidade de generalização e aplicabilidade a problemas onde o conjunto de treinamento tem tamanho limitado e não representa, necessariamente, uma observação adequada do sistema a ser aprendido.

1.2 Abordagem e Organização do Trabalho

O desenvolvimento deste trabalho partiu de uma compreensão do campo de Inteligência Computacional como um todo, seus diferentes métodos, paradigmas e abordagens, obtendo-se um mapa integrado de vantagens e desvantagens de cada um deles, inter-relacionamentos e classes de problema de maior aplicabilidade. As Máquinas de Vetores de Suporte foram, então, identificadas como um método de alta capacidade de generalização e histórico de sucesso em aplicações semi-supervisionadas, sendo, então, estudadas em mais detalhe. Os benefícios de otimalidade global das soluções obtidas pelos Algoritmos Evolucionários de busca levou à seleção de um Algoritmo Genético para utilização na otimização do aprendizado semi-supervisionado. Utilizando-se os conceitos obtidos na etapa inicial de investigação, pôde-se ainda identificar as vantagens da inserção de um método de um outro paradigma completamente diferente de aprendizado, a *Inferência Transdutiva*, na orientação do processo de busca do Algoritmo Genético implementado. O resultado é uma abordagem híbrida, que demonstra resultados superiores às tradicionais TSVMs para a classe de problemas estudada.

O trabalho é, então, organizado da seguinte forma:

- No Capítulo 2, os diferentes paradigmas de aprendizado de máquinas são divididos em *Supervisionados* ou *Não-Supervisionados*, e em *Indutivos* ou *Transdutivos*. Os principais métodos de aprendizado são então apresentados, dentre os quais as *Redes Neurais Artificiais*, os *Sistemas Nebulosos*, a *Swarm Intelligence* e a *Computação Evolucionária*;

- O Capítulo 3 parte da diferenciação entre Otimização Determinística e

Estocástica para a caracterização dos Algoritmos Evolucionários. Os Algoritmos Genéticos, utilizados neste trabalho, são, então, discutidos em detalhes. Apresentam-se também os métodos de computação evolucionária baseados em *Swarm Intelligence*, mais especificamente o *Ant Colony Optimization* e o *Particle Swarm Optimization*;

- As Máquinas de Vetores de Suporte, Redes Neurais Artificiais de alta capacidade de generalização e ampla aplicação, são apresentadas no Capítulo 4. Inicia-se a discussão pela descrição da Teoria do Aprendizado Estatístico, fundamentação teórica sobre a qual as SVMs são construídas. A classificação de margens máximas das SVMs é demonstrada passo a passo, partindo da formulação linear de margens rígidas, realizando-se a flexibilização das margens por meio da introdução das variáveis de folga e, por fim, possibilitando a aplicação a problemas não-lineares através do mapeamento ao espaço de características induzido por funções de Kernel;

- A nova abordagem é proposta no Capítulo 5. Inicialmente, modifica-se a formulação das SVMs apresentada no Capítulo 4 de forma a adaptá-la ao paradigma semi-supervisionado. Apresentam-se, então, as TSVMs e as limitações inerentes à sua formulação. Propõe-se então as *GA3SVMs* (*Genetic Algorithm Semi-Supervised Support Vector Machines*), e sua versão modificada, com operador de mutação orientado pelo método transdutivo dos *k-Nearest Neighbors*, denominada *GA3SVM-GDMP* (*Genetic Algorithm Semi-Supervised Support Vector Machines - Gene-Dependent Mutation Probability*);

- As principais conclusões e discussões são apresentadas no Capítulo 6. Inicialmente, discutem-se as principais contribuições e limitações da metodologia proposta. As modificações possíveis na abordagem para contornar tais limitações são, então, apresentadas. As propostas de continuidade versam sobre dois principais temas: as modificações e melhorias na metodologia proposta, e a extensão do paradigma Semi-Supervisionado a outras máquinas de aprendizado, a saber, os *Kernel Fisher Discriminants* e as redes *Multi-Layer Perceptron*.

1.3 Conclusões do Capítulo

O presente capítulo estabelece o contexto no qual o desenvolvimento deste trabalho está inserido. Descreve-se a Aprendizagem de Máquinas, sub-área da Inteligência Computacional voltada ao desenvolvimento de algoritmos de

aprendizado. A classe de problemas em que o conjunto de dados não incorpora informação suficiente para a definição de uma regra geral adequada e as limitações dos principais métodos disponíveis na literatura são apresentadas como as motivações do trabalho desenvolvido. A abordagem seguida no desenvolvimento do método proposto é, então, discutida, além das implicações da mesma na estrutura e organização do trabalho.

CAPÍTULO 2

Aprendizagem de Máquinas

No presente capítulo, são apresentados os conceitos básicos de aprendizado de máquinas, sub-área da Inteligência Computacional que busca possibilitar às máquinas "aprender". As diferentes teorias e os principais métodos de aprendizado de máquinas são aqui discutidos.

2.1 Introdução

Conforme discutido no Capítulo 1, o campo da Aprendizagem de Máquinas é uma sub-área da Inteligência Computacional voltada ao desenvolvimento de algoritmos que possibilitem às máquinas *aprender*.

Dentro do contexto de Inteligência Computacional, uma definição adequada de aprendizado, adaptada de (Mendel & McLaren 1994), é a seguinte:

> Aprendizado é o processo pelo qual um sistema inteligente é adaptado através do estímulo do ambiente no qual está inserido. O tipo de aprendizado é determinado pela maneira através da qual esta adaptação é realizada.

Assim, a definição do processo de aprendizado implica na análise dos seguintes aspectos (Haykin 1999):

1. O sistema inteligente (SI) é *estimulado* pelo ambiente;
2. O SI é *ajustado* como resultado dos estímulos recebidos;
3. O SI *responde de uma nova maneira* ao ambiente devido aos ajustes realizados.

A forma como relacionam-se os estímulos do ambiente e os ajustes realizados na Máquina de Aprendizado separam os diversos métodos de aprendizagem em, basicamente, Supervisionados ou Não-Supervisionados, e em Indutivos ou Transdutivos. As diferenças entre as possíveis classificações serão explicitadas na seção 2.2.

2.2 Teorias de Aprendizado

Um agente inteligente busca compreender o comportamento de um sistema com uma visão limitada sobre o mesmo, modelando e posteriormente emulando seu comportamento da forma mais coerente possível com o sistema original (Figura 2.1).

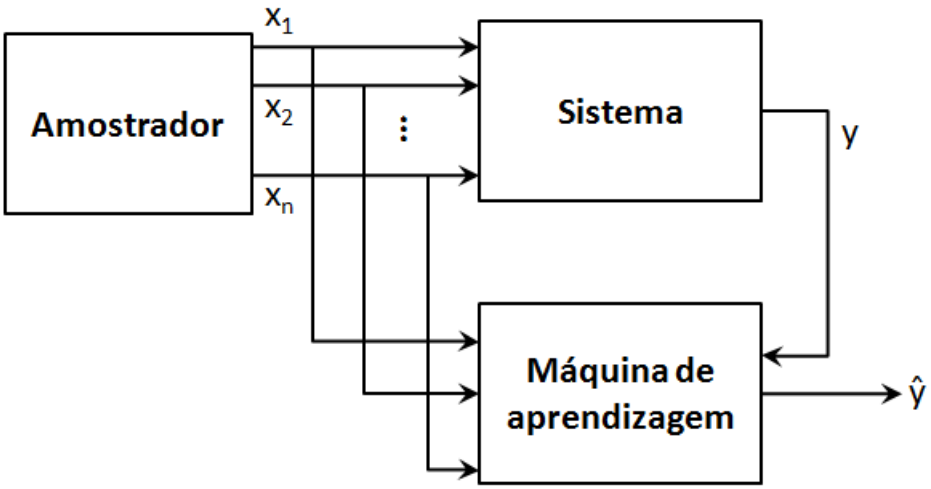


Figura 2.1: Esquema de um sistema e uma máquina de aprendizado

A Figura 2.1 ilustra tal mecanismo. As observações do sistema são dadas pelos pares de entrada x_i e saída y, e a resposta da máquina de aprendizado, buscando emular o comportamento do sistema, é dada por $\hat{y}$.

Uma das grandes vantagens do Aprendizado de Máquinas reside em sua utilização em problemas onde não é possível ou viável encontrar-se soluções exatas. Com tais técnicas, diminui-se a necessidade de inserção de conhecimento prévio de um especialista para a resolução do problema, além de permitir que se encontrem novas regras e relacionamentos, implícitos nos dados mas não facilmente observáveis por especialistas humanos.

Os mecanismos de aprendizado segundo os quais as máquinas de aprendizagem incorporam conhecimento acerca do problema (ou do sistema) podem ser separados em *Aprendizado Supervisionado e Não-Supervisionado*.

2.2.1 Aprendizados Supervisionado e Não-Supervisionado

Técnicas de *Aprendizado Supervisionado* são as mais comumente utilizadas no treinamento de *Redes Neurais Artificiais* e de *Árvores de Decisão*. Seu funcionamento consiste na inserção de um "Supervisor" no ciclo de aprendizado, responsável por dizer ao modelo se suas previsões estão corretas ou não.

Uma das formas de se implementar tal decisor se baseia na construção de um conjunto de dados denominado *conjunto de treinamento*, no qual amostras são retiradas do sistema e previamente classificadas pelo supervisor. Assim, durante seu processo de aprendizado, a máquina pode verificar se suas respostas estão coerentes ou não com o esperado, ajustando-se de forma a minimizar o erro para tal conjunto (os problemas inerentes a treinamentos baseados em simples minimização do erro de treinamento serão discutidos na seção 2.3.1.2) O esquema de aprendizado supervisionado pode ser observado na Figura 2.2.

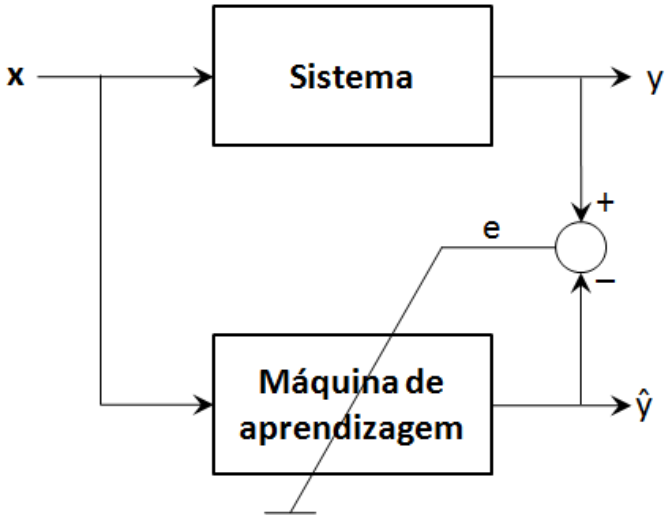


Figura 2.2: Esquema de aprendizado supervisionado

Tal abordagem pode ser bastante útil para problemas de classificação e regressão, em que já se sabe de antemão os resultados esperados para um dado número de pontos (o problema de reconhecimento de caracteres é um exemplo do tipo).

Entretanto, uma abordagem alternativa é a de se fazer com que o computador aprenda sem orientá-lo especificamente quanto ao que fazer. Uma das formas possíveis de aprendizado não-supervisionado consiste em criar sistemas onde o agente é premiado segundo a adequação de suas respostas, de forma que o mesmo busque a maximização das premiações, e não a solução do problema em si. Tal paradigma de aprendizado é usualmente chamado de *Aprendizado por Reforço* (*Reinforcement Learning*) (Sutton & Barto 1998).

Outra classe de problemas em que o aprendizado não-supervisionado é uti-

lizado é o *agrupamento (clustering)*. O problema de agrupamento é bastante similar ao de classificação de padrões: em ambos, deseja-se que o sistema seja capaz de determinar de que classe é um novo padrão apresentado, com base em observações anteriores. A diferença reside no fato de que, no aprendizado supervisionado, um conjunto de dados já classificados é fornecido ao sistema, enquanto no não-supervisionado tais classificações não existem - o sistema deve, por si só, decidir quais são as classes pertinentes, simplesmente agrupando padrões semelhantes.

No *Aprendizado Semi-Supervisionado*, uma parte dos dados utilizados no treinamento é classificada, enquanto outra consiste de dados não-rotulados. Tal paradigma de aprendizado é particularmante útil em casos onde a amostragem limitada do conjunto de treinamento não fornece informação suficiente para a indução de uma regra-geral. Assim, utiliza-se o conjunto de teste como fonte extra de informação para a resolução do problema.

Dentre os problemas em que essa abordagem é útil estão todos aqueles onde o espaço de amostragem é demasiadamente grande para ser possível gerar uma amostra estatisticamente representativa, ou ainda nos casos em que o classificador é demasiadamente caro (seja pelo custo computacional ou pelo alto grau de especialização). O problema de classificação de textos da internet é um exemplo adequado: a quantidade de textos disponíveis na rede é demasiadamente grande, especialmente quando relacionada ao custo de um classificador humano na classificação de tais textos quanto a, por exemplo, seu assunto (processo de geração de padrões de treinamento). Assim, uma máquina de busca pode baixar muito mais textos que um classificador humano pode lidar, e os textos não classificados podem ser utilizados na indução de uma regra-geral, mesmo sem um conjunto de treinamento suficiente para tal.

A Figura 2.3 será utilizada para ilustrar a diferença entre os esquemas supervisionado e semi-supervisionado de aprendizado. Tal figura representa um problema de aprendizado hipotético: os sinais $+$ e $-$ representam os pontos já classificados, presentes no conjunto de treinamento, enquanto os pequenos pontos não classificados fazem parte do conjunto de teste. A solução supervisionada, que não utiliza informações de tais pontos na indução da regra geral, é apresentada na forma de uma linha pontilhada, enquanto a solução semi-supervisionada é dada pela linha sólida. Observa-se que a solução supervisionada não resulta em boa separação para os conjuntos de treinamento e de teste juntos, enquanto a semi-supervisionada, sim. Pode-se perceber, entretanto, que os conjuntos de treinamento e de teste não representam amostragens *i.i.d.* dos padrões de entrada. É exatamente nessa situação que a utilização do aprendizado semi-supervisionado mostra-se mais interessante.

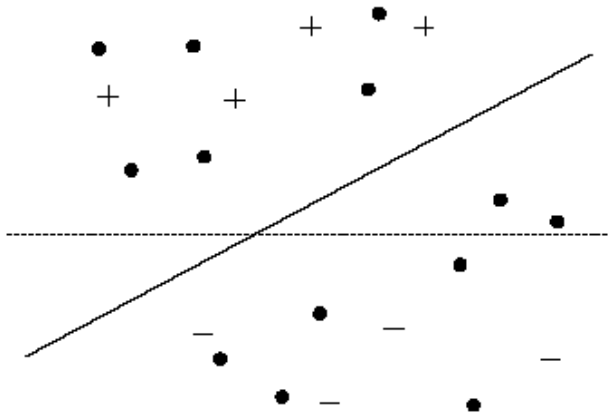


Figura 2.3: Separações possíveis: aprendizados supervisionado e semi-supervisionado

2.2.2 Inferências Indutiva e Transdutiva

No contexto de Inteligência Computacional, entende-se por *inferência* o processo de realizar predições baseadas em observações (Solomonoff 1964). Dois tipos básicos de inferência podem ser utilizados para aprendizagem de máquinas: as Inferências Indutiva e Transdutiva.

Na Inferência Indutiva, o agente em aprendizado busca **induzir** uma regra-geral (ou um relacionamento) baseado em um conjunto de observações do sistema, posteriormente aplicando tal regra na predição de valores para novos padrões apresentados. Tal abordagem é extremamente conveniente para problemas em que se busca uma solução com alta capacidade de generalização (seção 2.3.1.2).

Assim, o processo de aprendizado indutivo pode ser resumido nos seguintes passos:

1. Observação de um sistema;
2. Indução de um modelo (ou regra-geral) que aproxime (ou explique) tal sistema;
3. Utilização da regra induzida na predição do comportamento de tal sistema face a novos estímulos.

Entretanto, para uma outra classe de problemas, tal abordagem pode ser desnecessariamente (ou proibitivamente) complexa, ou até mesmo levar a comportamentos indesejados (Vapnik 1998). Uma abordagem alternativa é a de se "pular" o passo de indução de regras, realizando a predição de um fenômeno pela direta relação do mesmo com os padrões anteriormente observados. Tal princípio é denominado *Inferência Transdutiva*.

O exemplo mais comum de aprendizagem transdutiva é o Método dos k-Vizinhos mais próximos.

2.2.2.1 *Método dos k-Vizinhos mais Próximos*

O Método dos k-Vizinhos mais Próximos (*k-Nearest Neighbors* ou *k-NN*) (Cover & Hart 1967) está entre os mais simples e mais largamente utilizados métodos em classificação de padrões. Para o caso base, em que $k = 1$, seu funcionamento consiste em, para cada padrão apresentado, atribuir ao mesmo a classificação do seu vizinho mais próximo (Figura 2.4).

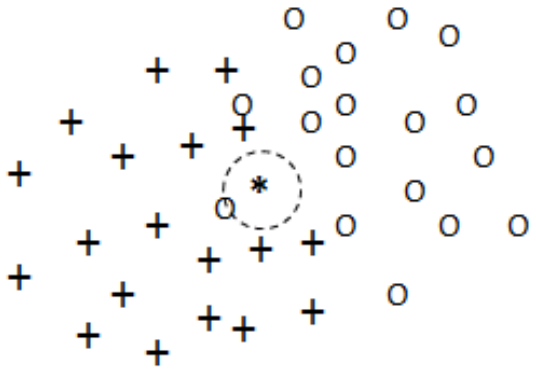


Figura 2.4: Regra do Vizinho Mais Próximo com $k = 1$

Pode-se variar o valor de k, de forma a considerar não apenas o vizinho mais próximo, mas os k vizinhos mais próximos. Assim, cada novo padrão será classificado segundo a classificação da maioria de seus k vizinhos (Figura 2.6).

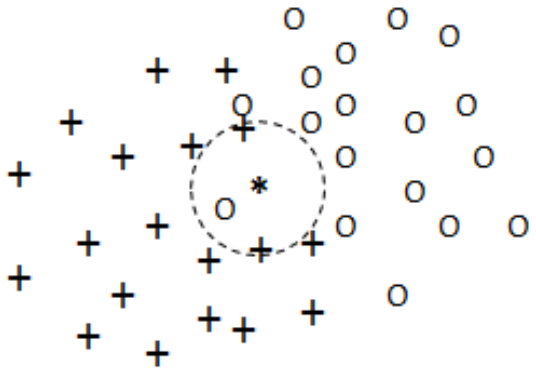


Figura 2.5: Regra do Vizinho Mais Próximo com $k = 3$

Para evitar "empates", pode-se utilizar sempre valores ímpares de k. Tal procedimento, entretanto, não é imprescindível, podendo-se utilizar valores pares de k, decidindo de maneira aleatória em caso de empate.

2.2.3 Aprendizado Semi-Supervisionado e Inferência Transdutiva

Atualmente vigora ainda certa confusão entre o Aprendizado Semi-Supervisionado e o Aprendizado Transdutivo (Chapelle, Schölkopf, & Zien 2006). Esta seção visa a esclarecer nosso entendimento sobre tal assunto.

A principal motivação dos dois métodos é fundamentalmente diferente: no aprendizado semi-supervisionado, o objetivo é utilizar informações do conjunto de teste de forma a auxiliar a indução de uma regra-geral para o problema enquanto, no aprendizado transdutivo, o foco está em não se resolver desnecessariamente um problema demasiadamente complexo, focando a solução nos pontos de interesse.

Assim, posteriormente à fase de treinamento, qualquer padrão não previamente apresentado pode ser classificado utilizando um agente de aprendizado semi-supervisionado. Entretanto, tal processo não é possível para um agente de aprendizado transdutivo - um novo processo de aprendizado é necessário a cada nova instância de teste. Assim, classificação de pontos *a posteriori* não é aceita no paradigma transdutivo, ou se cairia no seguinte caso: bastaria apresentar-se todos os possíveis pontos do espaço de entrada ao agente transdutivo e colher suas classificações para que fosse mapeado todo o espaço de soluções - e consequentemente uma regra de decisão, levando o método transdutivo a recair sobre a indução.

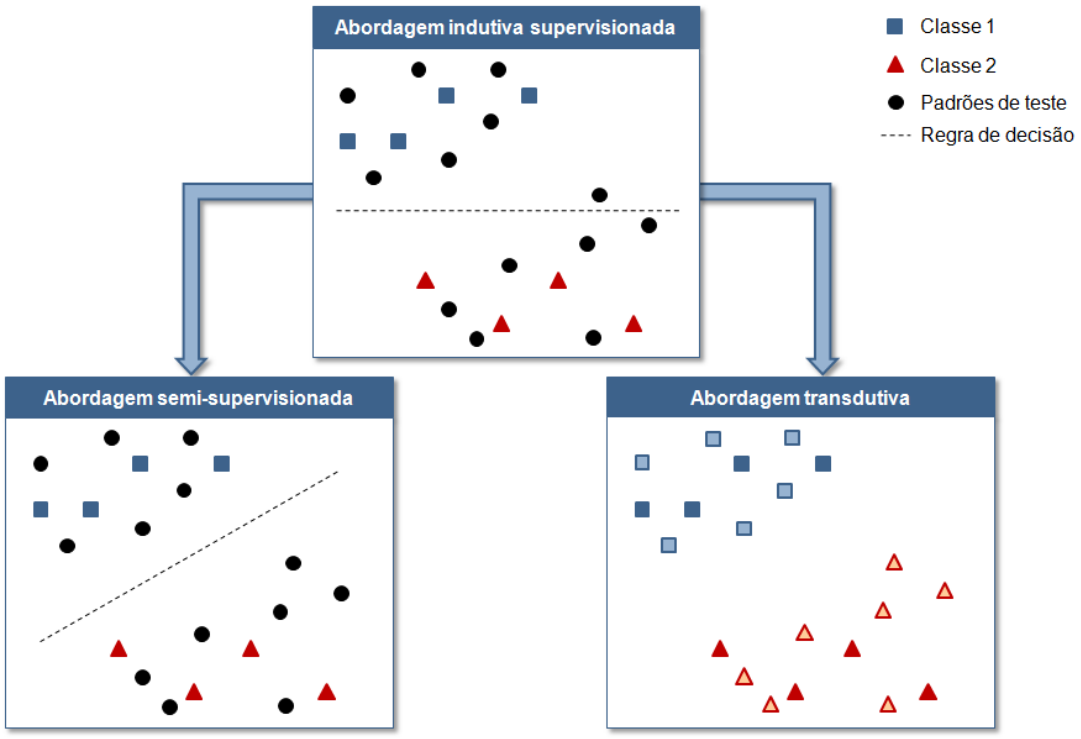


Figura 2.6: Exemplo de diferença entre aprendizado indutivo supervisionado e abordagens semi-supervisionada e transdutiva

2.3 Métodos de Aprendizado de Máquinas

Dentre as várias técnicas utilizadas para propiciar capacidade de aprendizado a uma máquina destacam-se as baseadas em sistemas naturais, sejam eles físicos ou biológicos. Dentre eles, podem-se citar as Redes Neurais Artificiais (Seção 2.3.1), os Sistemas Nebulosos (Seção 2.3.2), a *Swarm Intelligence* (Seção 2.3.3) e os métodos de Computação Evolucionária (Seção 2.3.4).

2.3.1 Redes Neurais Artificiais

O primeiro trabalho no campo das Redes Neurais Artificiais foi apresentado em 1943 por Wax Ten McCulloch e Walter Pitts (McCulloch & Pitts 1943), onde foi mostrado que unidades de processamento semelhantes aos neurônios biológicos podem realizar cálculos computacionais. Uma representação esquemática do chamado *Neurônio MCP*, ou *Perceptron Simples*, pode ser observada na Figura 2.7.

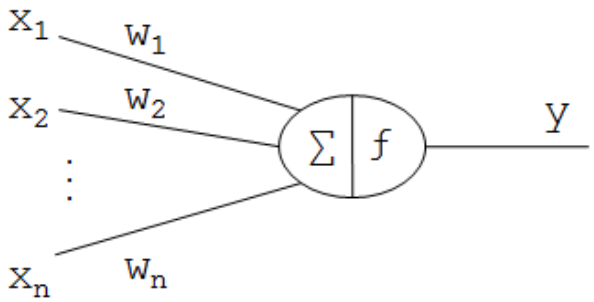


Figura 2.7: Representação esquemática de um Neurônio MCP

Tal modelo se assemelha bastante a um neurônio biológico. As entradas x_1 a x_n representam dendritos, prolongamentos dos neurônios naturais utilizados na recepção de estímulos nervosos. Os pesos w_1 a w_n representam as conexões sinápticas entre neurônios, responsáveis por enfatizar os estímulos vindos de alguns neurônios. A saída y representa o axônio, prolongamento do neurônio natural responsável pela condução do estímulo nervoso a outros neurônios. Ainda, a função de decisão $f(\vec{x})$ tem a mesma função da decisão de disparar (ou não) os neurônios naturais.

Entretanto, em 1969, com a publicação do livro *Perceptrons*, por Marvin Minsk e Seymour Papert (Minsky & Papert 1969), foi provado que o *Perceptron* simples só é capaz de solucionar problemas linearmente separáveis (não sendo capaz de solucionar, por exemplo, o clássico problema do OU-EXCLUSIVO). Tal constatação reduziu momentaneamente o interesse e o desenvolvimento das Redes Neurais Artificiais (RNAs) (Braga, Ludermir, & de Carvalho 2000).

Seu desenvolvimento foi retomado, entre outros motivos, devido ao desenvolvimento do método de retropropagação de erros e a introdução das Redes Perceptron Múltiplas Camadas (Multi-Layer Perceptron - MLP) (D. E. Rumelhart & Williams 1986), em 1986.

2.3.1.1 Redes MLP

Dentre as formas mais populares de RNAs se encontram as redes *Multi-Layer Perceptron (MLP)*. Tais redes são compostas por unidades funcionais constituídas no *Neurônio MCP* (McCulloch & Pitts 1943).

Uma representação esquemática de rede MLP pode ser observada na Figura 2.8.

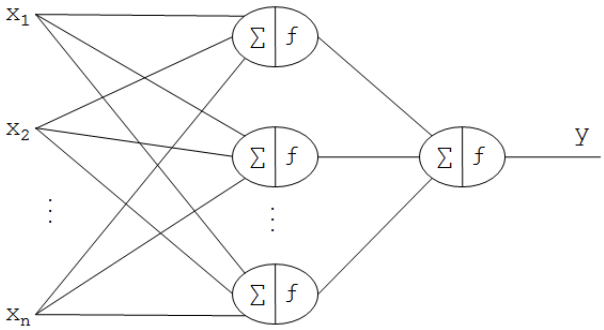


Figura 2.8: Rede MLP genérica

As camadas escondidas das Redes MLP são responsáveis por mapear o espaço de entrada no chamado *Espaço de Características*, onde problemas não lineares podem ser transformados em lineares e, então, facilmente resolvidos.

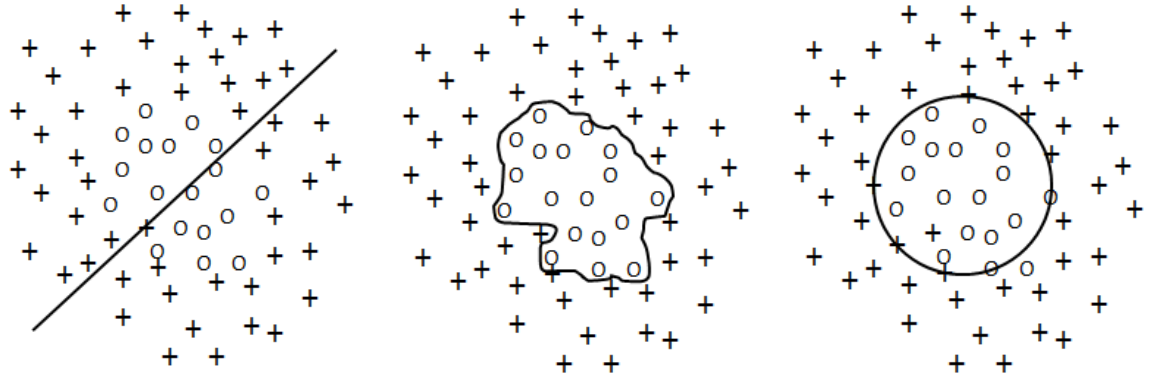
O processo de aprendizado de uma rede MLP consiste em se encontrar os pesos correspondentes às ligações entre os neurônios de forma a se obter o comportamento desejado.

2.3.1.2 Capacidade de generalização

O processo de aprendizado supervisionado consiste em encontrar uma regra-geral baseada em amostragens do espaço de entrada, buscando a maximização da capacidade de generalização do modelo.

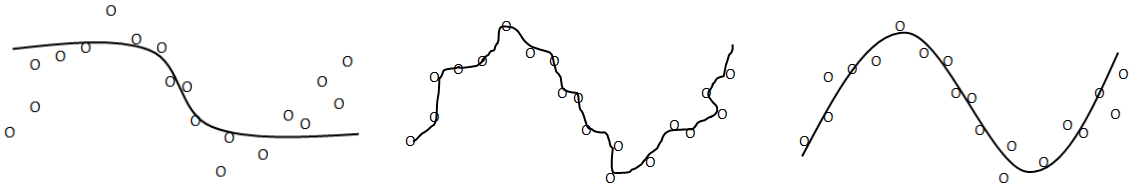
Dado um conjunto de treinamento, é possível construir um modelo que se ajuste perfeitamente aos padrões de entrada. Entretanto, na presença de ruído, inerente a todos os processos reais de amostragem, tal abordagem pode ser prejudicial na estimação de uma regra-geral adequada (Figuras 2.9 e 2.10).

Ambas as redes mostradas nas Figuras 2.9(b) e 2.10(b) sofrem de *overfitting*: complexidade do modelo superior à complexidade do sistema, resultando em memorização dos padrões de treinamento, incluindo o ruído presente nos mesmos. Entretanto, redes de complexidade demasiadamente baixa podem



(a) Sub-ajuste (b) Sobre-ajuste (c) Ajuste adequado

Figura 2.9: Sub-ajuste e sobre-ajuste para um problema de classificação



(a) Sub-ajuste (b) Sobre-ajuste (c) Ajuste adequado

Figura 2.10: Sub-ajuste e sobre-ajuste para um problema de regressão

também apresentar respostas pobres para tais problemas (Figuras 2.9(a) e 2.10(a)). É dito que tais redes sofrem de *underfitting*.

Na Figura 2.11, observa-se que o erro relativo a um conjunto de teste, utilizado freqüentemente como uma aproximação para a capacidade de generalização da rede, decresce com a diminuição do erro de treinamento apenas até certo ponto, a partir do qual menores erros de treinamento resultam em maiores erros sobre o conjunto de teste (Mackay 2002). A partir desse ponto de inflexão do erro de teste a rede apresenta *overfitting*.

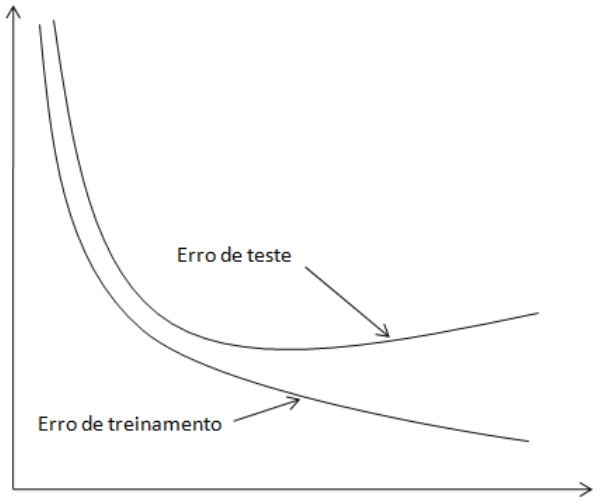


Figura 2.11: Evolução do erro de teste com o erro de treinamento

Estudos demonstram que, através da adição de pequenos ruídos aleatórios aos padrões de treinamento, pode-se melhorar a capacidade de generalização da rede (Reed & Robert J. Marks 1998; Bishop 1995). Assim, o número de padrões de treinamento sobe bastante e a rede tende a não receber o mesmo padrão várias vezes, o que dificulta a memorização do mesmo. Tal procedimento é equivalente a se impor a restrição de suavidade à rede, já que se está efetivamente dizendo à rede que para entradas levemente diferentes o resultado deve ser praticamente o mesmo. Resta ainda o problema de que o treinamento com ruído pode ser bastante mais lento, por multiplicar o tamanho do conjunto de treinamento, além da dificuldade de se decidir quanto ruído deve ser adicionado.

Assim, no processo de aprendizado, deve-se buscar soluções que encontrem regularidades, de forma a permitir a generalização do modelo e, dentre essas soluções encontradas, deve-se priorizar as de menor complexidade possível, desde que ainda se adeqüem ao problema proposto (Figuras 2.9(c) e 2.10(c)). Busca-se, então, ajustar a complexidade do modelo à complexidade do problema que se deseja resolver.

Para redes MLP, por exemplo, existem duas formas básicas de se controlar a complexidade: 1) controla-se o número de parâmetros livres da rede (Lawrence, Giles, & Tsoi 1996); 2) controla-se a magnitude desses parâmetros (Bartlett 1998). Baseado nessas premissas, o método MOBJ (de Albuquerque Teixeira 2001) realiza o controle de complexidade baseado na norma do vetor de pesos da rede.

2.3.1.3 O Treinamento MOBJ

O treinamento multiobjetivo (de Albuquerque Teixeira 2001) procura um compromisso entre minimizar, ao mesmo tempo, dois objetivos conflitantes: i)ϕ_1: erro de treinamento (média do erro quadrático); ii)ϕ_2: norma dos pesos. O controle da norma dos pesos expressa o controle da complexidade da rede. É importante ressaltar que o MOBJ difere de outros algoritmos de controle de norma ao trabalhar com erro e norma separadamente, e não em uma mesma função de custo, com ponderação para as duas contribuições, como é feito em algoritmos como o *Weight Decay* (Hinton 1989).

As duas etapas do treinamento MOBJ são descritas a seguir. Essas etapas consistem, primeiramente, na geração de um conjunto de soluções chamado Pareto-ótimo (Teixeira, Braga, Takahashi, & Saldanha 2000; Costa, Braga, Menezes, Teixeira, & Parma 2003), e posteriormente na obtenção da melhor solução de rede, presente nesse conjunto, através de uma regra de decisão (de Albuquerque Teixeira 2001).

O Conjunto Pareto-ótimo O primeiro passo do método MOBJ é a obtenção do conjunto Pareto-ótimo. Esse conjunto tem a propriedade de ser o limite entre as soluções viáveis e as não-viáveis. A figura 2.12 apresenta um esquema da curva do Pareto-ótimo. As soluções que pertencem à curva não podem ser melhoradas considerando os dois objetivos simultaneamente. O conjunto viável é aquele que se encontra acima da curva apresentada.

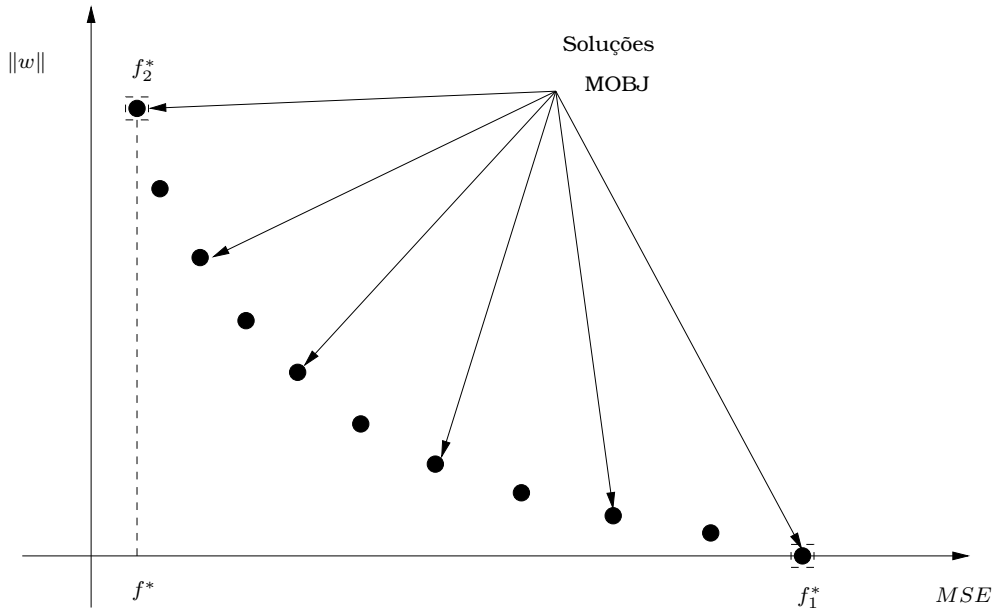


Figura 2.12: Conjunto Pareto-ótimo

Na figura 2.12, pode-se observar soluções f_1^* e f_2^* nos extremos da curva. A solução f_1^* corresponde à solução sub-ajustada, que pode ser obtida igualando os pesos a zero. Já a solução f_2^* corresponde à solução sobre-ajustada, que pode ser obtida pelo treinamento da rede por um algoritmo padrão como o *Backpropagation* (Rumelhart, Hinton, & Williams 1986). As soluções intermediárias são geradas pelo algoritmo MOBJ (Teixeira, Braga, Takahashi, & Saldanha 2001; Costa, Braga, Menezes, Teixeira, & Parma 2003; de Albuquerque Teixeira 2001; Teixeira, Braga, Takahashi, & Saldanha 2000). A solução f^* é a solução ideal (norma zero e erro mínimo) e que nunca pode ser obtida. O algoritmo MOBJ resolve o problema de otimização multi-objetivo através da transformação desse problema em um outro, mono-objetivo, onde os múltiplos objetivos aparecem como restrições (Takahashi, Peres, & Ferreira 1997). Os detalhes do algoritmo são apresentados em (de Albuquerque Teixeira 2001).

O Decisor O segundo passo do método MOBJ é a escolha da melhor solução, dentre aquelas presentes no conjunto Pareto-ótimo. Seja $\mathcal{W}^*$ o conjunto das soluções Pareto-ótimas para um dado conjunto de treinamento $\chi = \{\vec{x}_i, y_i\}_{i=1}^{N_T}$ estatisticamente representativo. Seja $\chi_V = \{\vec{x}_{V_i}, y_{V_i}\}_{i=1}^{N_V}$ um conjunto de vali-

dação também estatisticamente representativo. Em (de Albuquerque Teixeira 2001), demonstra-se matematicamente que a solução de pesos que minimiza o erro quadrático de validação é a que gera um modelo de rede $f(\vec{x};\vec{w}^*)$ que melhor aproxima a função geradora $f_g(\vec{x})$. Portanto, a regra de decisão é a seguinte:

$$\vec{w}^* = \arg\min e_V \,, \tag{2.1}$$

onde e_V é dado pela equação:

$$e_V = \frac{1}{N_V}\sum_{i=1}^{N_V}[y_{V_i} - f(\vec{x}_{V_i};\vec{w})]^2 \,, \tag{2.2}$$

onde $\vec{w} \in \mathcal{W}^*$.

2.3.1.4 O Teorema No Free Lunch

Entretanto, não há uma forma universal de se medir a complexidade de uma RNA. Na simples escolha da medida de complexidade já se está introduzindo conhecimento na rede, orientando, porém restringindo, o conjunto de soluções possíveis.

Tal problema pode ser formalizado pelo chamado Teorema *No Free Lunch* (Wolpert & Macready 1997). Segundo tal teorema, caso não se faça nenhuma suposição prévia de que observações passadas (i.e., conjunto de treinamento) estão relacionadas a observações futuras (i.e., conjunto de teste), qualquer predição é impossível. Além disso, caso não haja nenhuma restrição *a priori* de quais os fenômenos passíveis de serem observados no futuro, torna-se impossível generalizar e, assim, não há um algoritmo sempre melhor que outro (qualquer algoritmo pode ser batido por outro para alguns fenômenos observados).

Assim, é necessário realizar suposições para tornar possível a explicação de um sistema por um modelo restrito - entretanto, ao se fazer tais suposições, está-se inerentemente introduzindo conhecimento no sistema, e restringindo o conjunto de soluções possíveis.

2.3.2 Sistemas Nebulosos

A *Lógica Nebulosa* (ou *Lógica Fuzzy*) é provavelmente a área em Inteligência Computacional com mais aplicações já em uso em sistemas reais. Proposta inicialmente por Zadeh em 1965 (Zadeh 1965), a lógica *fuzzy* introduz a idéia de infinitos graus de incerteza que podem existir entre a certeza do sim e do não.

Considere, por exemplo, um elemento e e dois possíveis conjuntos de interseção vazia, $A \bigcap B = \emptyset$, e de união igual ao conjunto universo, $A \bigcap B = \mathbb{U}$. A teoria de conjuntos tradicional postula que, nessas condições, o elemento e está ou no conjunto A ($e \in A$) ou no B ($e \in B$).

Indo além da teoria de conjuntos tradicional, Zadeh argumenta que um elemento pode apresentar um grau de pertinência ao conjunto A ($e \rightarrow \mu_A$) e outro grau de pertinência ao conjunto B ($e \rightarrow \mu_B$), onde μ representa a função de pertinência.

Tal teoria se assemelha à maneira humana de se efetuar classificação. Pense em um bebê. Ele é jovem? Sim. Ele é velho? Não. Agora, pense em um senhor de 90 anos. Ele é jovem? Não. É velho? Sim. Mas em que idade uma pessoa se torna velha? Aos 40? 50? Talvez aos 60? A idéia exposta aqui é que, entre as certezas de jovem e velho, podem haver infinitos graus de pertinência. Um homem de 40 anos pode ser jovem em alguns contextos e velho em outros.

Mandami aplicou a lógica *fuzzy* ao introduzir as regras *fuzzy*. Tais regras seguem a seguinte estrutura:

- **Se** *Antecedente* **então** *Consequente*

Por exemplo, para o cenário de uma presa fugindo de um predador, poder-se-ia utilizar as seguintes regras para guiar o comportamento de um agente inteligente (no caso, uma presa):

- **Se** a distância ao predador mais próximo é pequena **então** fuja depressa;
- **Se** a distância ao predador mais próximo é grande **então** fique onde você está.

Essas regras simples utilizam o que chamamos de *variáveis linguísticas fuzzy*. O que é uma distância pequena? O que quer dizer *fuja depressa*? Métodos tradicionais tentariam relacionar com altíssima precisão a distância à velocidade. Mas isso não é, provavelmente, o que presas reais fariam. O seu comportamento se assemelha muito mais às regras *fuzzy* listadas acima.

2.3.3 Swarm Intelligence

Uma das mais novas e intrigantes áreas dentro da Inteligência Computacional é a *Swarm Intelligence* (numa tradução livre, "inteligência de multidões"), abordada nesta seção. Tal expressão foi primeiramente utilizada por Gerardo Beni e Jing Wang em 1989, no contexto de sistemas robóticos celulares (Beni & Wang 1989).

Animais que vivem em colônias têm recebido atenção especial da comunidade de *Swarm Intelligence*. Pássaros percorrem longas distâncias, mantendo a coesão para reduzir a resistência do ar. Peixes formam cardumes para se protegerem de ataques de predadores. Formigas deixam rastros químicos para guiarem outras formigas às novas fontes de comida. Tudo isso realizado sem a necessidade de coordenação centralizada ou supervisão. Regras simples guiam indivíduos simples, levando a colônias complexas e eficientes.

Swarm Intelligence é definida por Liu como "a inteligência coletiva emergente de grupos de agentes autônomos simples", em (Liu & Passino 2000).

Em (Bonabeau & Meyer 2001), Bonabeau lista as três características responsáveis pelo sucesso dos insetos sociais:

- flexibilidade (uma colônia se adapta facilmente a um ambiente em modificação);
- robustez (a falha de um indivíduo não compromete a eficiência da colônia como um todo);
- auto-organização (nenhuma coordenação centralizada ou supervisão é requerida).

Essas características tornam a *Swarm Intelligence* um campo interessante para engenheiros, especialmente para os de Otimização, onde robustez e flexibilidade são preocupações centrais. A habilidade de se auto-organizarem é bastante interessante, não sendo, entretanto, crucial para a Otimização. Aplicações de *Swarm Intelligence* a Otimização são discutidas no Capítulo 3 (seção 3.2.2).

2.3.3.1 Comportamento Emergente

Swarms são sistemas de múltiplos agentes, com um grande número de partículas seguindo regras simples e comunicando-se localmente umas com as outras, na busca de um determinado objetivo. O resultado de tais regras simples individuais pode ser um comportamento coletivo extremamente complexo.

Freqüentemente, o comportamento coletivo emergente das regras simples é difícil de ser predito. Outras vezes, variações pequenas nas regras podem resultar em mudanças significativas no comportamento coletivo. Tais dificuldades tornam o Comportamento Emergente uma das áreas mais desafiadoras de *Swarm Intelligence*.

A Emergência de Comportamentos pode ser vista de duas direções opostas: A *abordagem direta* está em prever o comportamento emergente por meio da análise das regras. Entretanto, muitas vezes o objetivo é exatamente o oposto:

Para um comportamento coletivo desejado, quais regras devem ser utilizadas? A essa abordagem dá-se o nome de *Abordagem Inversa*.

2.3.4 Computação Evolucionária

Outra classe de métodos baseados em sistemas naturais são os chamados *Algoritmos Evolucionários*, técnicas baseadas na Teoria da Evolução Natural e na Genética (Mitchell 1996).

Em sua célebre *Teoria da Evolução das Espécies* (Darwin 1859), Charles Darwin propõe que a ocorrência de pequenas variações nos indivíduos e a luta pela sobrevivência são os motivadores da evolução natural. Em um ambiente de competição por recursos (e por parceiros para o acasalamento), o ambiente exerceria pressão para que os indivíduos mais adaptados tenham maior probabilidade de sobreviverem e se reproduzirem, transmitindo as características que os fizeram superiores às futuras gerações. A tal princípio dá-se o nome de *Seleção Natural*.

Através dos estudos em genética, fundamentaram-se as hipóteses de Darwin sobre o mecanismo da hereditariedade, que explica como as características de um indivíduo são transmitidas a seus descendentes. Assim, a seleção natural explica a otimização de comportamentos, enquanto a genética fundamenta a otimização do material genético (De Jong 1994).

Os Algoritmos Evolucionários baseiam-se nos princípios de evolução apresentados na definição de métodos de busca e otimização, conforme discutido no Capítulo 3.

2.4 Conclusões do Capítulo

Uma visão ampla da área de Aprendizagem de Máquinas foi apresentada neste capítulo, estabelecendo as bases sobre as quais os demais capítulos são desenvolvidos. A compreensão dos diferentes paradigmas de aprendizado, suas motivações e implicações inerentes ao seu uso é fundamental na escolha do método mais apropriado ao tipo de problema que se deseja resolver. Metodologias híbridas, que incorporam vantagens de diferentes abordagens, são também possíveis, e muitas vezes apresentam resultados superiores a abordagens genéricas para problemas específicos. Deve-se atentar, entretanto, ao fato de que a simples escolha de métodos, seu projeto e ajuste resultam em adição de informação ao processo de busca, orientando os resultados ao que se deseja obter. Assim, deve-se ter cuidado para não generalizar a superioridade de um método sobre os outros. Conforme o Teorema *No Free Lunch*, há sempre uma classe de problemas para a qual um método se comporta pior que outros disponíveis.

CAPÍTULO 3

Computação Evolucionária

Este capítulo aborda a Computação Evolucionária, uma das áreas de Inteligência Computacional de maior inspiração biológica. Os conceitos de diferentes métodos baseados em populações são apresentados, desde os baseados no Evolucionismo Darwiniano, como os Algoritmos Genéticos (GAs), até os motivados por *Swarm Intelligence*, como o *Ant Colony Optimization* (*ACO*) e o *Particle Swarm Optimization* (*PSO*).

3.1 Introdução

Antes de deixar sua residência, um motorista pode ponderar distância, trânsito e número de pedágios a fim de decidir que caminho seguir para o destino desejado, minimizando tempo de viagem, consumo de combustível e/ou dinheiro gasto.

Após receber um financiamento, um empresário deve decidir como utilizar tais recursos de forma a maximizar o valor da companhia.

Em ambas as situações, e em várias outras, uma maneira eficiente de se tomar decisões, atingindo certos objetivos, é o problema principal. O processo de busca pelo conjunto de parâmetros que maximiza o desempenho de um sistema é denominado *Otimização*.

Assim, os problemas de otimização podem ser descritos da seguinte forma (Equação 3.1):

$$\begin{aligned} \text{Minimizar/Maximizar:} \quad & \vec{f}(\vec{x}) \\ \text{Sujeito a:} \quad & \vec{g}(\vec{x})\{.\}\vec{b} \end{aligned} \tag{3.1}$$

onde $\vec{f}$ é o vetor de funções objetivo, $\vec{x}$ as variáveis de otimização, e $\vec{g}\{.\}\vec{b}$ as restrições, onde $\{.\}$ pode representar quaisquer relações de igualdade ou desigualdade (i.e. $<$, $\leq$, $=$, $>$ e $\geq$).

Segundo tal definição, observa-se que o problema de aprendizagem de máquinas é, por si só, um problema de otimização. A busca pelo vetor de pesos de uma rede MLP que minimiza o erro relativo ao conjunto de treinamento é um exemplo, dentre diversos outros, de formulação de aprendizado bastante adequada à otimização.

3.1.1 Otimização Determinística e Otimização Estocástica

Os métodos de otimização podem ser divididos em determinísticos e estocásticos.

Os métodos determinísticos utilizam informações da função objetivo para buscar soluções ótimas. Dependem, portanto, de atributos da função, como suavidade e diferenciabilidade. Nessa classe de métodos podem-se citar o *Gradiente Descendente*, o *Simplex*, o *Algoritmo Elipsoidal* e os métodos de *Newton* e *Quasi-Newton*, dentre outros.

Entretanto, a utilização de métodos determinísticos pode não ser adequada a problemas onde se pode assumir pouco a respeito da função objetivo, ou em casos onde o ruído nos dados pode levar a grandes dificuldades. Além disso, os métodos desta classe sofrem gravemente com a existência de ótimos locais: uma vez posicionados dentro da "bacia de atração" de um ótimo local, a maioria dos métodos têm dificuldades em deixar tal bacia e, assim, têm sua capacidade de otimização global bastante reduzida.

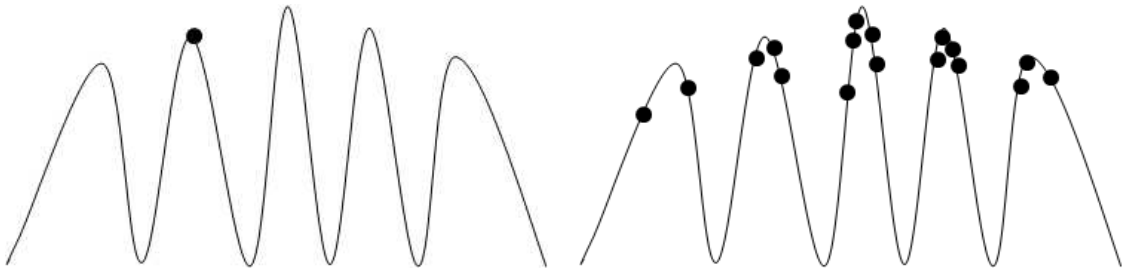
Na outra ponta estão os métodos de otimização estocástica. Tais métodos, geralmente baseados em sistemas físicos ou biológicos, realizam uma "busca orientada" pelo espaço de soluções, sem a necessidade de conhecimento prévio acerca da função objetivo.

De maneira geral, os métodos estocásticos preservam maior capacidade de otimização global, mas tendem a exigir um número muitas vezes mais elevado de avaliações da função objetivo, o que pode tornar sua utilização proibitiva em casos onde tal procedimento é demasiadamente caro.

Alguns dos métodos estocásticos mais conhecidos são o *Recozimento Simulado* (Kirkpatrick, Gelatt, & Vecchi 1983), o *Tabu Search* (Glover & Laguna 1993) e os métodos evolucionários, dentre eles os *Algoritmos Genéticos* (Holland 1975) e os *Sistemas Imunológicos Artificiais* (Dasgupta 1998). Os Métodos Evolucionários são mais profundamente abordados na Seção 3.2.

3.2 Métodos evolucionários

Os métodos evolucionários são denominados métodos baseados em populações. Nestes, ao invés de se trabalhar com uma única solução, caminhando com a mesma pelo espaço de soluções na busca do ótimo, utiliza-se uma população de soluções, que evoluem baseadas em algum princípio definido na busca pela solução ótima. Os esquemas de métodos de otimização clássicos e os baseados em populações podem ser observados na Figura 3.1.



(a) Métodos tradicionais (b) Métodos populacionais

Figura 3.1: Comparação ilustrativa entre métodos de otimização tradicionais e populacionais

As heurísticas segundo as quais os métodos evolucionários guiam suas buscas no espaço de soluções se baseiam nos mais variados processos biológicos. Os algoritmos genéticos, abordados na Seção 3.2.1, inspiram-se na *Teoria da Evolução das Espécies*, de Charles Darwin (Darwin 1859), para a evolução de suas populações. Os métodos *Ant Colony Optimization* (Dorigo, Maniezzo, & Colorni 1996) e *Particle Swarm Optimization* (Kennedy & Eberhart 1995) baseiam-se nos princípios de *Swarm Intelligence*, e são abordados na Seção 3.2.2.1. Outros exemplos de métodos evolucionários são o *Algoritmo Clonal* (Castro 2000), versão modificada do GA com a supressão do operador de cruzamento, e os *Sistemas Imunológicos Artificiais*, que modelam os sistema imunológico humano para uso em otimização (Dasgupta 1998).

3.2.1 Algoritmos Genéticos

Baseado na célebre *Teoria da Evolução das Espécies*, de Charles Darwin (Darwin 1859), Holland propõe, em 1975, os chamados *Algoritmos Genéticos* (Holland 1975).

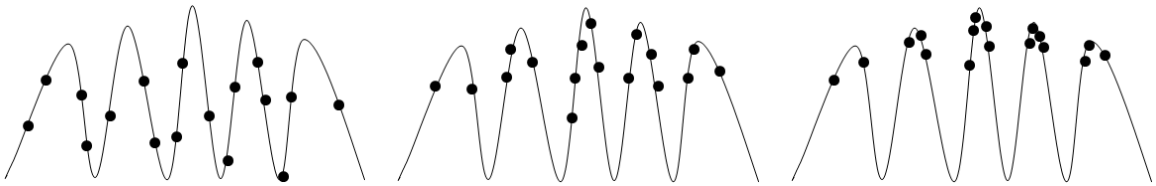
A idéia por trás desses algoritmos é a seguinte: dada uma população de indivíduos, a pressão ambiental estimula a seleção natural (*survival of the fittest*) e, assim, o desempenho geral da população cresce com o passar das gerações. Tal mecanismo é facilmente entendido como um processo de otimização. Dada uma função objetivo a ser maximizada (o análogo se aplica à minimização de funções), uma população de soluções é criada aleatoriamente

e, utilizando-se a função objetivo como uma medida do grau de adequação das soluções, os melhores candidatos tendem a ser escolhidos para formar a nova geração de soluções, por meio de procedimentos como o cruzamento e a mutação. Tal processo pode ser repetido até que se atinjam condições estabelecidas de convergência, nas quais espera-se que o ótimo global do problema esteja entre as soluções encontradas (Goldberg 1989).

O procedimento básico de um algoritmo genético pode ser expresso em pseudo-código na seguinte forma:

```
INICIALIZAR a população com indivíduos aleatórios
AVALIAR o desempenho dos mesmos
REPETIR ATÉ QUE (Condição de parada seja satisfeita)
    SELECIONAR indivíduos pais
    RECOMBINAR pares de indivíduos pais
    MUTAR indivíduos gerados
    AVALIAR novos indivíduos
FIM REPETIR
```

Assim, no processo de evolução, as soluções tendem a iniciar o processo dispersas, caminhando na direção dos ótimos com o passar das gerações (Figura 3.2).



(a) Etapa inicial (b) Etapa intermediária (c) Etapa final

Figura 3.2: Evolução de soluções durante processo de busca

3.2.1.1 Representação/Codificação de Indivíduos

O primeiro passo na definição de um algoritmo genético é a representação dos indivíduos. Nesta etapa, determina-se a relação entre o universo do problema e o universo onde a evolução será dada. Os elementos que formam possíveis soluções no universo do problema são denominados *fenótipos*, enquanto suas codificações, que levam ao universo da evolução, chamam-se *genótipos* (muitas vezes chamados também de *cromossomos*) (Eiben & Smith 2003). Assim, o passo de codificação consiste em definir um mapeamento entre os fenótipos e os genótipos que os representarão.

Dessa forma, num problema hipotético onde a busca seja por uma solução inteira, o conjunto dos números inteiros representa os fenótipos. Caso

se decidisse pela utilização de codificação binária para a representação dos genótipos, o número 23 seria representado pelo genótipo 10111.

Exemplos de codificações binária e *gray* podem ser observados na Figura 3.3.

	0	1	2	3	4	5	6	7
Binária	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
Gray	0 0 0	0 0 1	0 1 1	0 1 0	1 1 0	1 1 1	1 0 1	1 0 0

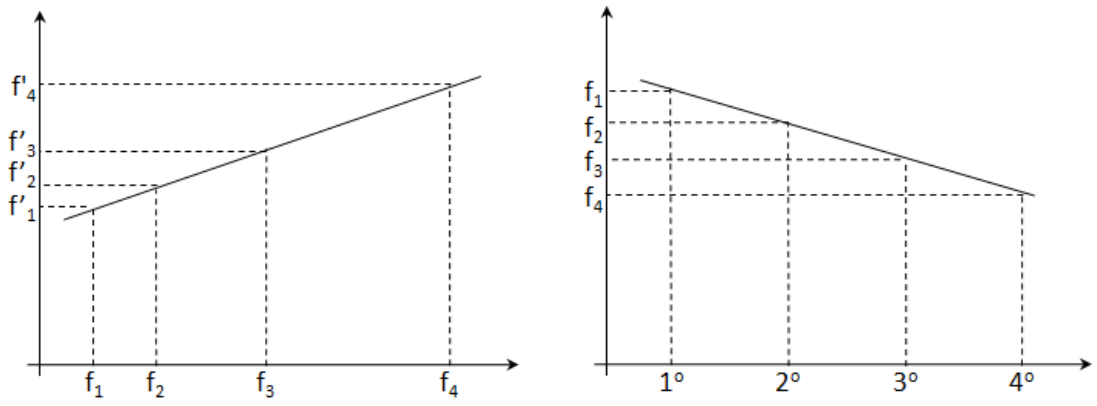
Figura 3.3: Exemplos de codificações binária e gray

3.2.1.2 Avaliação

O papel principal da função de desempenho é guiar o processo evolutivo, no espaço dos genótipos, à solução do problema de otimização, no espaço dos fenótipos. Assim, se desejássemos maximizar a função x^2 com $x \in \mathbb{I}$, o desempenho do genótipo 10111 seria dado pelo quadrado do fenótipo correspondente: no caso, $23^2 = 529$.

Muitas vezes, entretanto, o mapeamento direto entre os espaços de solução e de busca pode ocasionar convergência prematura, em casos de grandes diferenças nos valores da função de desempenho entre os indivíduos da população. Em um estágio inicial do desenvolvimento, o aparecimento de um indivíduo de desempenho algumas ordens de grandeza superior aos outros pode levar a uma importância demasiada desse indivíduo na formação da nova geração, focando a busca na região do mesmo e restringindo a capacidade de procura global do método.

Para evitar tal tipo de problema, pode-se recorrer, entre outros recursos, ao escalonamento da função objetivo. Como exemplo, pode-se citar dois métodos simples de escalonamento: o escalonamento linear e o ranqueamento. Ambos aparecem na Figura 3.4.



(a) Escalonamento linear (b) Escalonamento por ranqueamento

Figura 3.4: Escalonamentos linear e por ranqueamento

No escalonamento linear (Figura 3.4(a)), um valor máximo e um valor mínimo para a função de desempenho são definidos por parâmetros de entrada (v_{max} e v_{min}). Tais valores são atribuídos aos maior e menor desempenho dentre os indivíduos da população. Todos os outros valores são interpolados entre esses limites, no caso, de maneira linear. Isso limita a discrepância entre os desempenhos como a diferença entre o máximo e o mínimo definidos na entrada.

No escalonamento por ranqueamento (Figura 3.4(b)), os valores máximo e mínimo são novamente definidos como parâmetros de entrada mas, agora, a atribuição dos valores da função de *fitness* é feita seguindo um *ranking* de performance: o indivíduo de maior desempenho recebe o maior valor, enquanto o pior indivíduo recebe o menor valor. Aos indivíduos restantes são atribuídos valores da função de *fitness* em intervalos lineares entre v_{max} e v_{min}, segundo um *ranking* de desempenhos.

Outras abordagens, como o *niching* e o *crowding*, também atuam na prevenção de convergência prematura, com a vantagem adicional de forçarem a procura em diferentes regiões do espaço de busca, aumentando a probabilidade de convergência global e permitindo, ainda, a identificação de múltiplos ótimos locais. Para mais detalhes sobre tais métodos, ver (Oei, Goldberg, & Chang 1991).

3.2.1.3 Seleção

Uma vez realizada a avaliação da população e a atribuição de seus desempenhos, deve-se selecionar os indivíduos que formarão a próxima geração. Tal escolha deve seguir o princípio de seleção probabilística: indivíduos de maior desempenho devem ter **maior probabilidade** de seleção, mas deve ainda ser possível escolher indivíduos de desempenho inferior. Isso garante a diversidade genética da população, fator primordial para o sucesso de qualquer processo evolutivo.

Dentre os principais métodos de seleção, pode-se se citar o *Método da Roleta*, o *Torneio*, o *Deterministic Sampling*, o *Stochastic Remainder Sampling* e o *Stochastic Universal Sampling*. Os três primeiros serão abordados neste texto. Uma comparação entre métodos de seleção é apresentada em (Goldberg & Deb 1991). Para mais informações sobre os demais métodos de seleção, consultar (Baker 1987).

Método da Roleta No Método da Roleta, cada indivíduo tem a probabilidade de ser selecionado de acordo com o seu desempenho relativo ao da população, ou seja:

$$p_{sel} = \frac{f_1}{\sum f} \tag{3.2}$$

Desta forma, pode-se representar o processo de seleção como uma roleta, com as probabilidades de cada indivíduo ser selecionado sendo observadas na área de cada um, conforme observa-se na Figura 3.5.

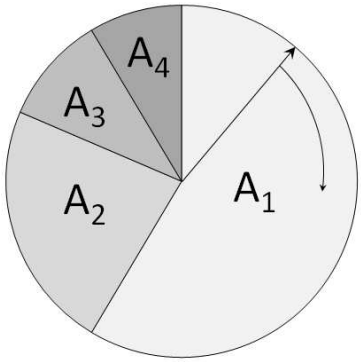


Figura 3.5: Método da Roleta

Tal roleta será "girada" n_{pop} vezes, sendo n_{pop} o número de indivíduos na população, e os indivíduos serão escolhidos de acordo com o local onde a roleta "parar". Assim, indivíduos de maior desempenho recebem uma área maior da roleta, tendo maior **probabilidade** de serem escolhidos para formar a nova população.

Torneio No método de torneio, selecionam-se aleatoriamente dois indivíduos de cada vez, e um "torneio" entre os dois é realizado, de forma que o melhor tenha maior probabilidade de ser escolhido. Tal torneio tem, usualmente, a seguinte forma: define-se um limiar de decisão ($L > 0.5$) e gera-se um número aleatório r. Caso $r < L$, o melhor indivíduo é escolhido. Caso contrário, o pior indivíduo é levado à próxima geração. Repete-se o processo n_{pop} vezes.

Tal procedimento pode ser representado pelo seguinte pseudo-código:

```
DEFINIR L
REPETIR Npop vezes
    SELECIONAR 2 indivíduos aleatoriamente
    r = valor aleatório entre 0 e 1
    SE r < L
        SELECIONAR o MELHOR indivíduo
    SENÃO
        SELECIONAR o PIOR indivíduo
    FIM SE
FIM REPETIR
```

Deterministic Sampling Para o método *Deterministic Sampling*, define-se uma população temporária de n_{temp} indivíduos, com o número de cópias de cada indivíduo dado por $n_i = INT(\frac{f_i}{\bar{f}})$, onde $\bar{f}$ representa o desempenho médio dos indivíduos. Assim, todos os indivíduos com desempenho acima da média são selecionados, com os indivíduos com desempenho m vezes superior à média sendo selecionados m vezes.

Caso $n_{temp} < n_{pop}$, seleciona-se mais $n_{pop} - n_{temp}$ indivíduos que contenham a maior parte fracionária de $f_i - INT(\frac{f_i}{\bar{f}})$.

Tal procedimento pode ser mais facilmente entendido pela Figura 3.6.

$f(x) = x^2 - x$

Genótipo	Fenótipo	$f(x)$	$f(x)/\bar{f}$	$INT\left(f(x)/\bar{f}\right)$	$f(x) - INT\left(f(x)/\bar{f}\right)$	**Indivíduos selecionados**
1 | 0 | 0 | 1	9	72	0.83	0	**0.83**	**1**
1 | 1 | 1 | 0	14	182	2.10	**2**	0.10	**2**
0 | 0 | 1 | 0	2	2	0.02	0	0.02	**0**
1 | 0 | 1 | 0	10	90	1.04	**1**	0.04	**1**

Figura 3.6: Exemplo de *Deterministic Sampling*

Elitismo Entretanto, tais processos não garantem que o melhor indivíduo seja levado à proxima geração, sendo então possível que se perca uma solução ótima nesse processo. Para evitar que isso ocorra, pode-se implementar o *Elitismo*, método pelo qual um número n de melhores indivíduos é sempre carregado, intacto, para a nova geração.

3.2.1.4 Cruzamento

O cruzamento é o principal operador dos Algoritmos Genéticos (Eiben & Smith 2003). Com ele, dois indivíduos têm seus genótipos recombinados, formando um ou dois novos indivíduos. Cruzamentos com mais de dois pais são também possíveis, mesmo não tendo embasamento biológico algum. Entretanto, são raramente utilizados, mesmo com alguns estudos indicando que os mesmos podem ter efeitos positivos na evolução (Eiben 1997).

O princípio por trás do cruzamento é o seguinte: dados dois indivíduos com características desejadas, um cruzamento entre eles poderia combinar ambas as características. Tal procedimento é usado há milênios pelo homem, com o cruzamento de animais ou plantas de características superiores, na tentativa de se obter variações ainda melhores.

Dentre os mais utilizados métodos de cruzamento podem-se citar o *cruzamento com um ponto de corte*, o *cruzamento com* n *pontos de corte*, o *cruzamento*

uniforme, o *cruzamento por variável* e o *cruzamento por variável entre vários indivíduos*. Os três primeiros serão introduzidos neste texto. Para informações sobre os demais, consultar (Bandyopadhyay & Pal 2007).

Cruzamento com um Ponto de Corte É o tipo de cruzamento mais simples entre todos. Sendo L o número de genes nos cromossomos, escolhe-se um ponto k, sendo $1 \leq k < L$, e, com probabilidade p_c de ocorrência, realiza-se a troca de material genético tendo esse ponto como *pivot*. Tal procedimento pode ser facilmente compreendido na Figura 3.7.

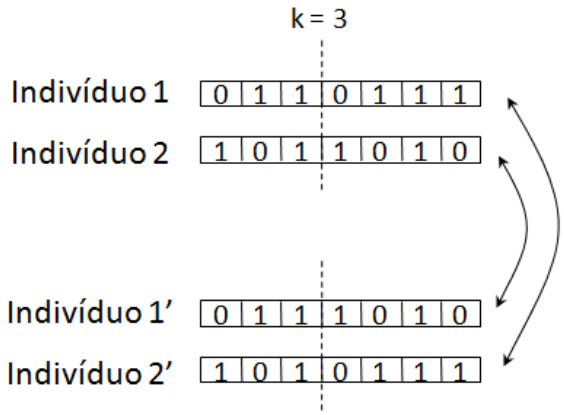


Figura 3.7: Cruzamento com 1 ponto de corte

Cruzamento com n Pontos de Corte Similar ao cruzamento com um ponto de corte, mas agora com n pontos. Selecionam-se os n pontos aleatoriamente, entre 1 e $L-1$, e, com probabilidade p_c, realiza-se a troca de material genético tendo tais pontos como *pivots*. Caso o mesmo ponto seja selecionado mais de uma vez, não se seleciona um novo ponto. Tal processo pode ser observado na Figura 3.8.

Cruzamento Uniforme Com probabilidade p_c, percorrendo os cromossomos de início ao fim, define-se a ocorrência ou não de um ponto de corte, a cada gene, com probabilidade de 50%. Caso o ponto seja escolhido como um ponto de corte, realiza-se a permuta do material genético naquela posição e prossegue-se para o próximo gene, repetindo esse processo até o final do comossomo. Espera-se, portanto, por $\frac{(L-1)}{2}$ pontos de corte nesse sistema.

3.2.1.5 Mutação

O operador mutação é um operador unário, ou seja, atua sobre apenas um genótipo de cada vez, gerando um novo indivíduo levemente modificado. O papel principal da mutação nos Algoritmos Genéticos é manter a diversidade

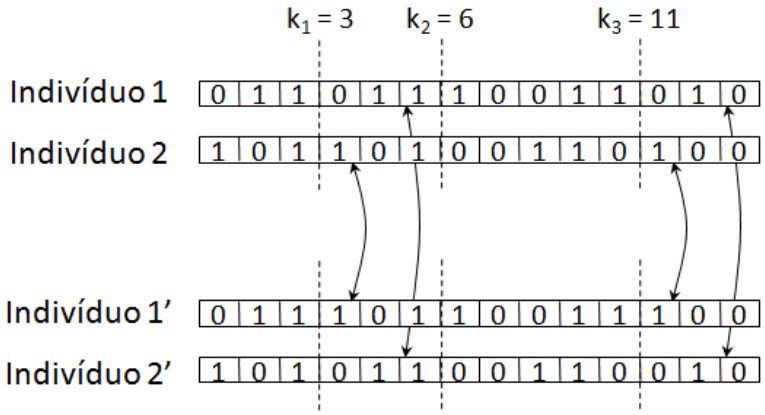


Figura 3.8: Cruzamento com n pontos de corte

genética da população. Assim, da mesma forma que o cruzamento tem o papel de levar o algoritmo a um novo ponto no espaço de buscas, a função da mutação é garantir que esse espaço esteja conectado (Eiben & Smith 2003). A importância de tal propriedade remonta aos teoremas que garantem que, dado um tempo suficiente, um GA atingirá o ótimo global do problema (Weishui & Chen 1996). Tais teoremas se apoiam na propriedade de que qualquer solução possível possa ser alcançada através dos operadores de cruzamento e mutação (Eiben & Smith 2003). Uma maneira imediata de garantir tal possibilidade é permitir que se "pule" de um local para outro no espaço de soluções, propriedade do operador de mutação.

Assim, para se realizar tal procedimento, basta decidir, com probabilidade p_m para cada gene, pela realização ou não da mutação. De maneira geral, p_m apresenta valores bastante baixos, usualmente entre 0.01 e 0.03.

3.2.2 *Métodos baseados em* Swarm Intelligence

O campo da *Otimização* é o que tem recebido maior contribuição dos estudos em *Swarm Intelligence* (Bonabeau, Dorigo, & Theraulaz 2000; Denby & Le Hégarat-Mascle 2003). O método *Ant Colony Optimization*, proposto por Dorigo (Dorigo, Maniezzo, & Colorni 1996), baseia a procura por ótimos globais nos mecanismos de busca por comida e deposição de feromônio de formigas. Tal método tem sido intensivamente aplicado no roteamento em redes de telecomunicações (Arabshahi, Gray, Kassabalidis, El-Sharkawi, II, Das, & Narayanan 2001; White 1997), entre várias outras aplicações. Tal método será abordado na Seção 3.2.2.1.

Observando a maneira como revoadas de pássaros se orientam, mudando de direção rapidamente de maneira coesa e organizada, sem a necessidade de coordenação centralizada, Eberhart propôs o método *Particle Swarm Optimiza-*

tion (Kennedy & Eberhart 1995). Tal método tem sido aplicado com sucesso na solução de uma extensa gama de problemas de otimização, desde o treinamento de Redes Neurais Artificiais (Duch & Korczak 1998) até a otimização de potência reativa (Wen Zhang & Clerc 2003). Uma introdução a tal método é dada na Seção 3.2.2.2.

3.2.2.1 Ant Colony Optimization

Formigas depõem uma substância química, denominada feromônio, marcando os caminhos seguidos enquanto buscam por fontes de comida. Elas tendem, também, a seguir caminhos previamente marcados com feromônio.

Baseado nisso, Deneubourg desenvolveu o seguinte experimento: ele construiu duas pontes entre um ninho de formigas e uma fonte de comida. Uma das pontes tinha o dobro do comprimento da outra. Em poucos minutos, a colônia tendia a selecionar sempre o caminho mais curto.

Isso acontece por duas razões. A primeira é que formigas seguindo o caminho mais curto retornarão mais rapidamente ao ninho do que aquelas que tomaram o percurso longo. Após algum tempo, o caminho curto terá sido visitado por mais vezes, resultando em trilhas mais fortes de feromônio, aumentando ainda mais sua probabilidade de seleção pelas novas formigas. A segunda razão é que feromônios evaporam com o tempo. Caminhos mais longos (resultando em tempos maiores de viagem), sofrerão mais com a evaporação.

O principal atrativo do comportamento de busca por comida das formigas é essa tendência de encontrar o caminho mais curto (através da deposição de feromônio), mantendo a habilidade de se adaptar a ambientes em modificação, como o aparecimento de um novo caminho mais curto (através da evaporação dos feromônios).

O método Ant Colony Optimization (ACO), proposto por Dorigo (Dorigo, Maniezzo, & Colorni 1996), baseia-se nessas características.

ACO - Algoritmo A modelagem matemática do comportamento de busca de comida das formigas seguirá o esquema abaixo:

Considere que entre dois nós, a e b, existam dois possíveis caminhos, 1 e 2, doravante referenciados como C_1 e C_2, respectivamente.

Alguns parâmetros devem ser definidos:

- L_1 e L_2 serão os comprimentos dos percursos C_1 e C_2;
- τ_1 e τ_2 serão os níveis de feromônio nos percursos C_1 e C_2;
- $\rho \in (0, 1]$ é o coeficiente de evaporação dos feromônios.

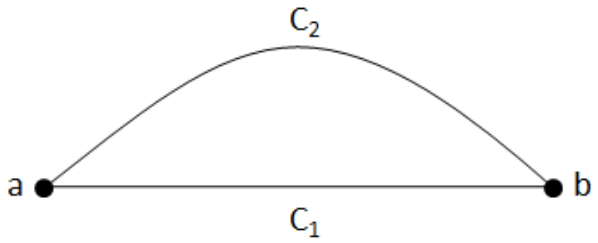


Figura 3.9: Caminhos hipotéticos C_1 e C_2

Apresentamos a estrutura do algoritmo:

1. Inicializar feromônios
   $\tau_1 = \tau_2 = c > 0$

2. Posicionar n_a formigas no nó a

3. Para cada uma das n_a formigas, atravessar do nó a ao nó b:

   - Por C_1 com probabilidade $P_1 = \frac{\tau_1}{\tau_1+\tau_2}$
   - Por C_2 com probabilidade $P_2 = 1 - P_1$

4. Evaporar os feromônios: $i = 1, 2$
   $\tau_i = (1 - \rho)\tau_i$

5. Para cada uma das n_a formigas, depositar feromônio nos percursos C_i da forma:
   $\tau_i = \tau_i + \frac{1}{L_i}$

Estendendo tal algoritmo para o número desejado de nós, e modelando o comprimento do caminho como a função objetivo a ser otimizada, tem-se a forma geral de tal método.

3.2.2.2 Particle Swarm Optimization

Inspirado na forma como revoadas de pássaros e cardumes de peixes se organizam, Eberhart propôs o método Particle Swarm Optimization (PSO) (Kennedy & Eberhart 1995).

Este é um método de otimização por agentes múltiplos, tal qual o algoritmo ACO discutido na Seção 3.2.2.1. Novamente, a busca pelo ótimo é realizada por partículas "voando" pelo espaço de busca, mas agora governadas por regras inspiradas em revoadas de pássaros e cardumes de peixes.

Cada partícula se move considerando três fatores: tende a seguir a trajetória que já vinha seguindo (inércia), enquanto se move em direção à melhor posição que já ocupou no espaço de busca (ótimo individual), e na direção da

melhor posição que qualquer outra partícula da colônia tenha ocupado (ótimo global).

A coesão com que revoadas de pássaros mudam de direção é interpretada nesse modelo como uma mudança no ótimo global.

Este é um algoritmo relativamente novo, tendo apresentado resultados altamente promissores.

3.3 Conclusões do Capítulo

Neste capítulo, apresentaram-se os conceitos da computação evolucionária, além de alguns de seus principais métodos, como os Algoritmos Genéticos, o *Ant Colony Optimization* e o *Particle Swarm Optimization*. A natureza estocástica dos AEs torna os métodos dessa classe mais propensos a alcançarem ótimos globais que os métodos de busca e otimização tradicionais. Entretanto, sua natureza populacional leva a complexidade e custo computacional maiores, especialmente em problemas onde a etapa de avaliação da função objetivo é demasiadamente cara computacionalmente.

CAPÍTULO 4

Máquinas de Vetores de Suporte

Neste capítulo, apresentam-se as Máquinas de Vetores de Suporte (*Support Vector Machines*, ou SVMs), um dos tipos de Redes Neurais Artificiais de maior sucesso entre a comunidade de Inteligência Computacional, seja por sua sólida formulação teórica ou pela ampla gama de aplicações práticas. Na seção 4.2, a Teoria do Aprendizado Estatístico é introduzida, definindo as bases sobre as quais a teoria das SVMs é construída. Parte-se do conceito da separação de margem máxima (seção 4.2.4) para a definição das SVMs de Margens Rígidas (seção 4.3.1), formulação mais restrita dessa máquina de aprendizado. Introduzem-se então as variáveis de folga, fundamentais ao desenvolvimento das SVMs de Margens Flexíveis (seção 4.3.2). Conclui-se estendendo a formulação para o caso mais geral, as SVMs Não-Lineares de Margens Flexíveis, através da definição do mapeamento para o espaço de características e das funções de Kernel (seção 4.4).

4.1 Introdução

As Máquinas de Vetores de Suporte (SVMs) constituem uma das técnicas de aprendizado de máquinas de maior sucesso e aplicação pela comunidade de Inteligência Computacional. Têm apresentado resultados equivalentes e, muitas vezes, superiores aos alcançados por outros algoritmos de aprendizado, inclusive outros tipos de RNAs (Braga, Ludermir, & de Carvalho 2000).

Voltadas para o projeto de Redes Neurais *feedforward* de uma camada escondida com neurônios não-lineares, as SVMs são um método de aprendizado de máquinas elegante e altamente fundamentado (Haykin 1999). Seguem o

paradigma de Aprendizado Supervisionado apresentado no Capítulo 2 (seção 2.2).

As SVMs têm forte embasamento conceitual proveniente da Teoria do Aprendizado Estatístico, descrita por Vapnik em (Vapnik 1998) e inicialmente investigada em (Vapnik & Chervonenkis 1971). Conforme descrito na seção 4.2, tal Teoria orienta a busca pela melhor solução para um dado problema de aprendizado através da minimização não só do erro de treinamento, mas também da complexidade do modelo obtido, o que resulta em um dos principais pontos fortes das SVMs: a **alta capacidade de generalização**.

A extração dos *vetores de suporte*, principais padrões de treinamento responsáveis pela definição da solução do problema de aprendizado, é outro ponto de destaque das SVMs, conforme discutido na seção 4.3.

4.2 Teoria do Aprendizado Estatístico

Conforme discutido no Capítulo 2 (seção 2.3.1.2), a maximização da capacidade de generalização de uma Rede Neural Artificial é um problema fundamental que deve guiar todo o seu processo de aprendizado. A Teoria do Aprendizado Estatístico apresenta uma formulação matemática para o controle da capacidade de generalização de uma máquina de aprendizado genérica (Haykin 1999), no contexto do aprendizado supervisionado, apresentado no Capítulo 2 (seção 2.2.1).

O problema do Aprendizado Supervisionado consiste em, dada uma série de observações do sistema $\{(\vec{x_i}, y_i)\}_{i=1}^{N}$ (pares ordenados de entrada e saída desejada), com função geradora $y = f(\vec{x})$, encontrar a função $F(\vec{x}, \vec{w})$ que melhor aproxima a função geradora $f(\vec{x})$, onde $\vec{w}$ é o vetor de pesos.

Dessa forma, surge no processo de aproximação da função geradora a necessidade de definição de uma medida de discrepância ou perda, dada por $L(y, F(\vec{x}, \vec{w}))$, a qual indica o quanto a saída obtida $F(\vec{x}, \vec{w})$ difere da saída esperada y (Vapnik 1982). Uma função comumente empregada como medida de discrepância é o erro quadrático, dado por:

$$L(y, F(\vec{x}, \vec{w})) = (y - F(\vec{x}, \vec{w}))^2 \tag{4.1}$$

Sendo $P(\vec{x}, y)$ a distribuição de probabilidade das amostras do problema de aprendizado que se deseja resolver, o valor esperado para o erro da máquina de aprendizagem obtida é definido pelo *Risco Funcional*:

$$R_{funcional}(\vec{w}) = \int L(y, F(\vec{x}, \vec{w}))dP(\vec{x}, y). \tag{4.2}$$

Assim, o objetivo do aprendizado supervisionado (a aproximação da função

geradora $f(\vec{x})$ através da função $F(\vec{x}, \vec{w})$) pode ser também expresso como a minimização do risco funcional $R(\vec{w})$. Caso a distribuição de probabilidade $P(\vec{x}, y)$ seja conhecida, a avaliação do risco funcional é imediata (Vapnik 1998). Entretanto, esta distribuição é, de maneira geral, desconhecida para problemas de aprendizado supervisionado. Assim, é necessária a utilização do princípio indutivo de minimização do *Risco Empírico*, o qual tenta extrair o comportamento de um sistema através de um conjunto restrito de observações.

4.2.1 Minimização do Risco Empírico

Dado um conjunto de N amostras de um sistema $\{(\vec{x_i}, y_i)\}_{i=1}^{N}$, originadas da mesma distribuição de probabilidade $P(\vec{x}, y)$ desconhecida, define-se o Risco Empírico através de (Vapnik 1995):

$$R_{empirico}(\vec{w}) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, F(\vec{x_i}, \vec{w})) \tag{4.3}$$

Conforme expresso por Haykin em (Haykin 1999), a utilização deste novo funcional $R_{emp}(\vec{w})$ no lugar de $R_{func}(\vec{w})$ traz dois principais benefícios:

- O risco empírico **não depende explicitamente** da distribuição de probabilidade $P(\vec{x}, y)$ desconhecida;
- Tal risco pode, em princípio, ser minimizado com respeito ao vetor de pesos $\vec{w}$.

Para o caso da função de erro quadrática expressa na Equação 4.1, o risco empírico é dado por:

$$R_{empirico}(\vec{w}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - F(\vec{x_i}, \vec{w}))^2 \tag{4.4}$$

De acordo com a *Lei dos Grandes Números* (Gray & Davisson 1986), quando o número de amostras N tende ao infinito, o risco empírico converge em probabilidade para o risco funcional (Vapnik 1999). Entretanto, dado o tamanho limitado dos conjuntos de treinamento em problemas reais, a minimização do risco empírico não implica, necessariamente, na minimização do risco funcional.

Uma interpretação alternativa é a seguinte: Para um dado problema de classificação de padrões, o Risco Empírico é o erro de treinamento da máquina de aprendizado, enquanto o Risco Funcional é a probabilidade de erro para novos padrões da mesma distribuição não apresentados durante a etapa de treinamento (erro de generalização).

4.2.2 Dimensão VC

Um conceito importante no estudo de máquinas de vetores de suporte é o da *Dimensão Vapnik-Chervonenkis* ou, simplesmente, *Dimensão VC* (Vapnik & Chervonenkis 1971). A dimensão VC é uma medida da capacidade de separação de uma família de funções classificadoras implementáveis por uma dada máquina de aprendizado (Vapnik 1982).

Dois conceitos são importantes para a definição da dimensão VC (Haykin 1999):

Dicotomia Dizemos que as funções de classificação binárias, ou seja, aquelas que dividem o espaço em dois conjuntos disjuntos, implementam uma *dicotomia*;

Fragmentação Uma máquina de aprendizado tem a capacidade de implementar certo número de dicotomias. Dado um conjunto $\Im$ de pontos, dizemos que a máquina de aprendizado *fragmenta* o conjunto $\Im$ se todas as dicotomia possíveis desse conjunto podem ser implementadas por tal máquina.

Sendo $|\Im|$ a cardinalidade de $\Im$, tem-se que o número N de dicotomias implementáveis pela máquina deve ser igual a $2^{|\Im|}$ para que tal máquina *fragmente* o conjunto $\Im$.

Agora, pode-se definir a dimensão VC (Vapnik & Chervonenkis 1971; Kearns & Vazirani 1994; Vidyasagar 1997; Vapnik 1998):

> *A dimensão VC de um conjunto de dicotomias $\mathbb{F}$ é a cardinalidade do maior conjunto $\Im$ que pode ser fragmentado por $\mathbb{F}$.*

Em outras palavras, a dimensão VC de uma máquina de aprendizado é o maior número de padrões de treinamento que esta pode aprender com erro zero para todas as possíveis classificações binárias destes padrões.

Tal definição pode ser melhor compreendida através da Figura 4.1 (Schölkopf & Smola 2002). Na Figura 4.1, observa-se que um conjunto de 3 pontos tem todas as suas dicotomias possíves (totalizando $2^3 = 8$) implementáveis por uma reta. Entretanto, a Figura 4.2 mostra um exemplo de dicotomia de um conjunto de quatro pontos não implementável por uma reta. Tem-se, então, que o maior conjunto que pode ser fragmentado por uma reta tem cardinalidade igual a 3.

Assim, seja ∇ uma superfície linear de dimensão m (para o caso da reta, $m = 2$). A dimensão VC de ∇ é dada por (Haykin 1999):

$$VC_{dim}(\nabla) = m + 1 \tag{4.5}$$

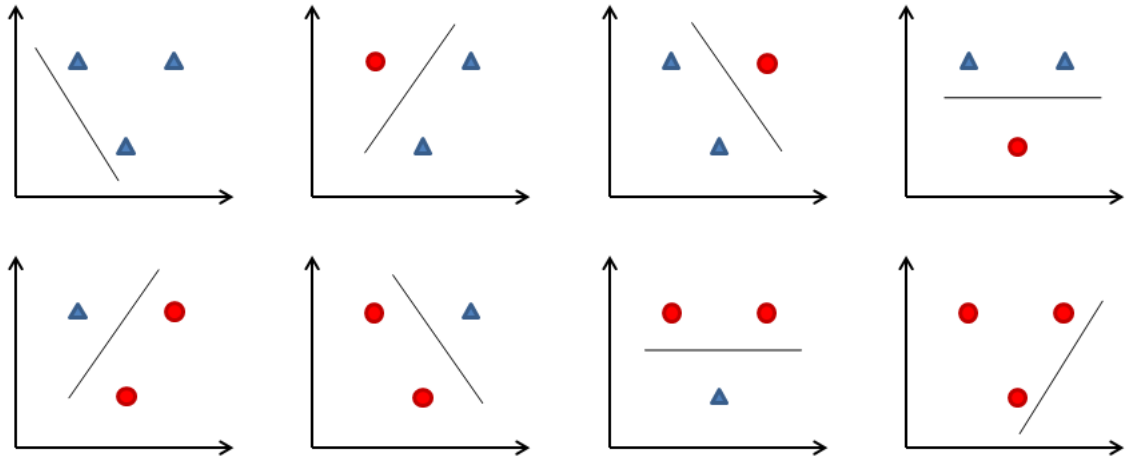


Figura 4.1: Dicotomias possíveis para um conjunto de 3 pontos em $\mathbb{R}^2$

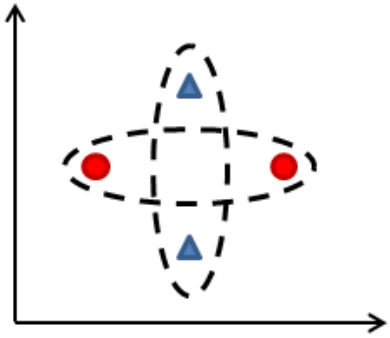


Figura 4.2: Exemplo de dicotomia de 4 pontos em $\mathbb{R}^2$ não-linearmente separável

A determinação analítica da dimensão VC de redes neurais é, na maioria das vezes, de muito difícil obtenção. Entretanto, dois resultados podem ser citados, os quais determinam limites superiores para a dimensão VC de dois tipos importantes de RNA's:

- A dimensão VC de uma rede *feedforward* arbitrária, com neurônios utilizando a função de ativação de *Heaviside* (função degrau) é $O(WlogW)$, onde W é o número de parâmetros livres da rede (Cover ; Baum & Haussler 1989);

- A dimensão VC de uma rede *MLP* com função de ativação sigmoidal é $O(W^2)$, onde W é o número de parâmetros livres da rede (Koiran & Sontag 1997).

A principal questão a ser notada desses resultados não são os limites em si, mas a constatação de que redes MLP apresentam dimensão VC *finita* (Haykin 1999).

4.2.3 Minimização do Risco Estrutural

Vapnik demonstra que o risco funcional é limitado pela seguinte expressão, com probabilidade $1 - \eta$ de ocorrer, dado um valor de $\eta \in [0, 1]$ (Vapnik 1998):

$$R_{funcional} \leq R_{empirico} + R_{estrutural}(h, N, \eta) \tag{4.6}$$

onde h é a dimensão VC da máquina de aprendizado, N é a cardinalidade do conjunto de treinamento e, η, um parâmetro escolhido.

Vapnik demostra também que o termo $R_{estrutural}(h, N, \eta)$ pode ser expresso por:

$$R_{estrutural} = \sqrt{\frac{h(log(\frac{2N}{h}) + 1) - log(\frac{\eta}{4})}{N}} \tag{4.7}$$

A Equação 4.6 mostra que a minimização do risco funcional, objetivo geral da aprendizagem de máquinas, é obtida pela minimização de dois termos, os riscos empírico e estrutural. A Equação 4.7 relaciona o risco estrutural à dimensão VC (h), de maneira direta: reduzindo-se a dimensão VC, ou a complexidade da máquina de aprendizado, diminui-se o risco estrutural. Entretanto, a redução do risco empírico se dá com o aumento da complexidade da rede e, portanto, com um aumento do risco estrutural. Assim, identifica-se um *trade-off* entre o risco empírico e o risco estrutural. O ponto que minimiza ambos os riscos (e assim, o risco funcional) é a complexidade ótima da máquina de aprendizado, como pode ser observado na Figura 4.3.

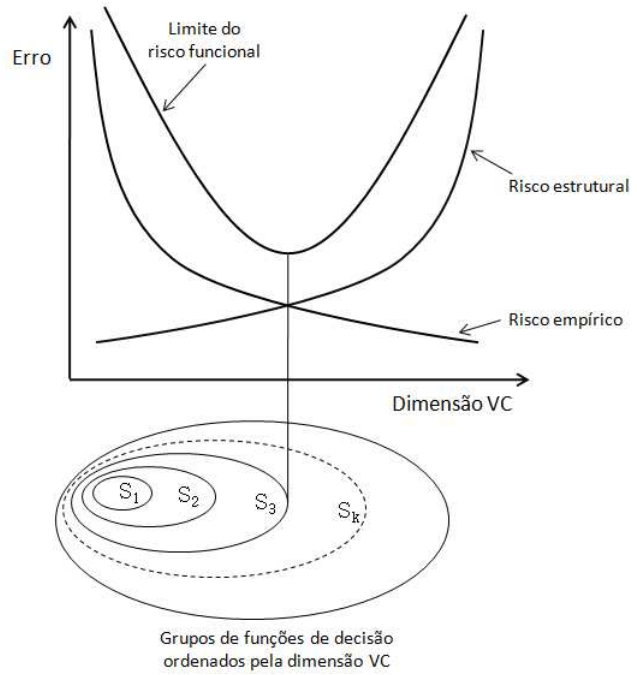
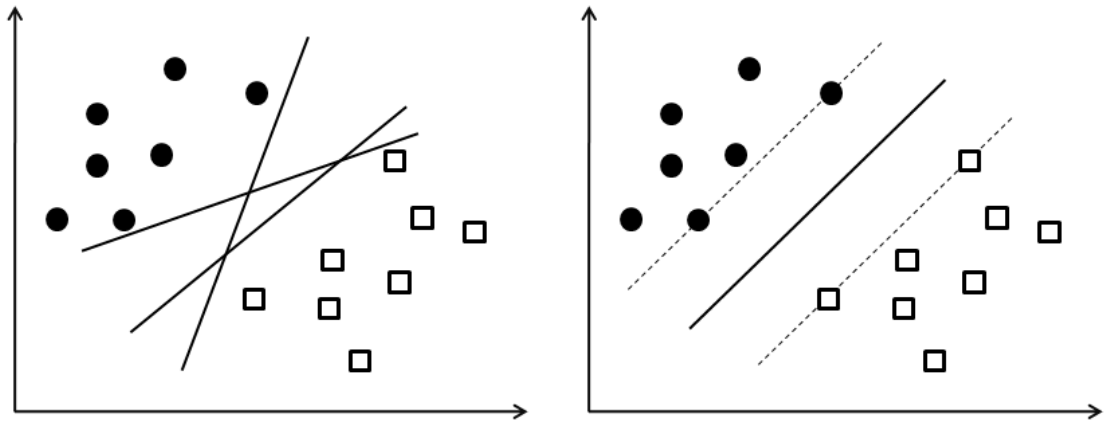


Figura 4.3: *Trade-off* entre os riscos empírico e estrutural na minimização do risco funcional

4.2.4 Separação com Margem Máxima

Para um dado problema de classificação, uma mesma dicotomia pode ser implementada por um conjunto de diferentes funções de decisão, conforme observado na Figura 4.4(a). Entretanto, há uma solução (Figura 4.4(b)) que maximiza a margem de separação entre as classes (aqui, margem de separação é definida como a menor distância euclideana entre a superfície de separação e o ponto mais próximo a ela).



(a) Exemplos de separações possíveis (b) Separação de margem máxima

Figura 4.4: Identificação da solução de máxima margem de separação

Mostra-se que (Vapnik, Levin, & Lecun 1994):

$$h \leq 1 + min(N, (\frac{R^2}{\rho^2})) \tag{4.8}$$

onde R é o menor raio possível de uma hiper-esfera circunscrita aos vetores de treinamento e ρ, a margem de separação entre as classes.

Daí conclui-se que a maximização da margem resulta em minimização da dimensão VC h e, conseqüentemente, do risco estrutural. As SVMs apoiam-se nesse princípio na orientação de seu processo de aprendizado, conforme detalhado na seção 4.3.

4.3 As Máquinas de Vetores de Suporte (SVMs) como Classificadores de Margens Máximas

Inicialmente desenvolvidas como classificadores binários (Boser, Guyon, & Vapnik 1992), as SVMs são poderosas máquinas de aprendizado utilizadas tanto em problemas de classificação (Burgess 1998) como de regressão (Smola & Scholkopf 1998). Apoiadas nos princípios da Teoria do Aprendizado Estatístico (seção 4.2), as SVMs buscam a maximização da margem de separação entre duas classes para a minimização do risco funcional, conforme discutido na

seção 4.2.4, tendo sido, portanto, originalmente denominadas "classificadores de margem ótima" (Boser, Guyon, & Vapnik 1992). Na seção 4.3.1, a formulação para problemas linearmente separáveis, denominada SVM de Margens Rígidas, é discutida. Em seguida, expande-se o conceito para casos mais gerais em que erros ou ruídos nos padrões de treinamento impossibilitam a completa separação das classes, apresentando-se a SVM de Margens Flexíveis (seção 4.3.2). Problemas não-linearmente separáveis, onde o mapeamento para um espaço de características de dimensão mais elevada é necessário, são discutidos na seção 4.4.

4.3.1 SVMs de Margens Rígidas

As SVMs lineares de margens rígidas definem fronteiras de separação planas entre duas classes linearmente separáveis. A equação de uma superfície de decisão implementada por um hiperplano é dada por

$$f(\vec{x}) = \vec{w}^T\vec{x} + b = 0 \tag{4.9}$$

onde $\vec{w}$ é o vetor normal ao hiperplano descrito por 4.9 e $\frac{b}{\|w\|}$ é a distância do hiperplano à origem, conforme observado na Figura 4.5.

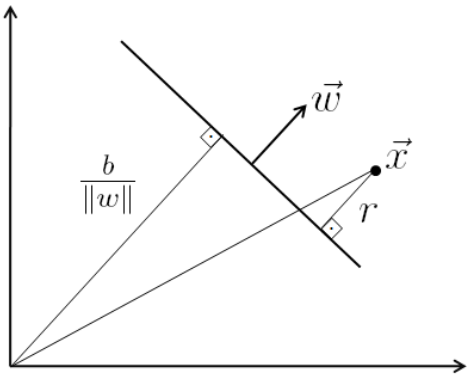


Figura 4.5: Representação geométrica de um hiperplano de duas dimensões

Tal resultado para b provém da constatação de que a distância r de um ponto à superfície expressa por 4.9 é dada por (Duda & Hart 1973):

$$r = \frac{\vec{w}^T\vec{x} + b}{\|\vec{w}\|} \tag{4.10}$$

Para utilização em um problema de classificação de padrões, por exemplo, observa-se que tal vetor divide o espaço em dois sub-espaços, um para cada classe. Assim, a classificação de cada padrão é dada por sua posição com relação às margens de separação, conforme expresso na Equação 4.11 (Haykin 1999):

$$\begin{cases} \vec{w}^T\vec{x}+b \geq +1 & se \quad y_i = +1 \\ \vec{w}^T\vec{x}+b \leq -1 & se \quad y_i = -1 \end{cases} \tag{4.11}$$

Para os pontos em que a primeira ou segunda expressão da Equação 4.11 são satisfeitos com uma igualdade, dá-se o nome de *vetores de suporte*, ou *support vectors* (Haykin 1999). Assim, da Equação 4.10, obtém-se que as distâncias dos vetores de suporte ao hiperplano de separação podem ser expressas por (Haykin 1999)(Equação 4.12):

$$\begin{cases} \frac{1}{\|\vec{w}\|} & se \quad y_i = +1 \\ -\frac{1}{\|\vec{w}\|} & se \quad y_i = -1 \end{cases} \tag{4.12}$$

Assim, a margem de separação entre as classes, conforme observado na Figura 4.6, pode ser expressa pela Equação 4.13, definida como a margem geométrica do classificador (Campbell 2001):

$$\rho = \frac{2}{\|\vec{w}\|} \tag{4.13}$$

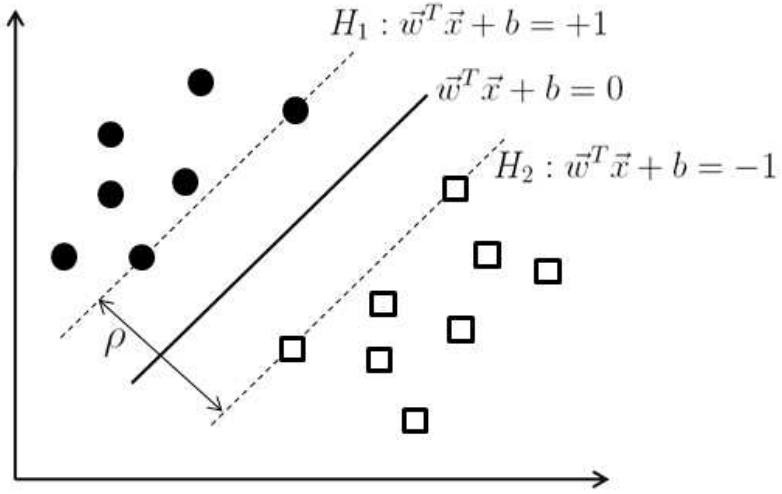


Figura 4.6: Determinação da margem ρ entre os hiperplanos H_1 e H_2

Conforme introduzido na seção 4.2.4, as SVMs obtêm sua solução através da maximização da margem de separação entre as classes. Em vista da Equação 4.13, conclui-se que tal resultado pode ser obtido por meio da minimização da margem do vetor de pesos $\|\vec{w}\|$ (Burgess 1998). Assim, obtém-se o seguinte problema de otimização primal (Schölkopf & Smola 2002):

$$\begin{aligned} \text{Minimizar através de } (\vec{w}, b): \quad & \frac{1}{2}\|\vec{w}\|^2 \\ \text{sujeito a:} \quad & \forall_{i=1}^{n} : y_i[\vec{w}.\vec{x_i}+b] \geq 1 \end{aligned} \tag{4.14}$$

As restrições garantem que não haja pontos entre as margens de separação das classes. Daí o nome de *SVMs de Margens Rígidas*.

4.3.1.1 O Problema Dual

Muitas vezes, a solução direta da Equação 4.14 é de difícil obtenção. Problemas quadráticos desse tipo, com função de custo convexa e restrições lineares em $\vec{w}$, podem ser solucionados utilizando o *método dos multiplicadores de Lagrange* (Bertsekas 1982), que transforma o problema primal em um novo problema, denominado dual, de mais simples solução. Tal método agrega as restrições à função de custo, associadas a variáveis auxiliares denominadas *multiplicadores de Lagrange*, resultando na função Lagrangeana expressa em 4.15 (Schölkopf & Smola 2002):

$$J(\vec{w}, b, \vec{x}_i, \vec{\alpha}) = \frac{1}{2}\|\vec{w}\|^2 - \sum_{i=1}^{N} \alpha_i[y_i(\vec{w}^T\vec{x}_i) + b) - 1] \tag{4.15}$$

onde os multiplicadores de Lagrange são expressos por α_i.

Tal equação deve, então, ser minimizada em relação a $\vec{w}$ e b, e maximizada em relação a $\vec{\alpha}$ (Muller, Mika, Ratsch, Tsuda, & Schölkopf 2001). Busca-se então o ponto de sela onde:

$$\frac{\partial J}{\partial \vec{w}} = 0 \quad e \quad \frac{\partial J}{\partial b} = 0 \tag{4.16}$$

A resolução dessas equações leva a (4.17 e 4.18):

$$\vec{w} = \sum_{i=1}^{N} \alpha_i y_i \vec{x}_i \tag{4.17}$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \tag{4.18}$$

Substituindo-se as Equações 4.17 e 4.18 na Equação 4.15 obtém-se o problema de otimização dual apresentado em 4.19:

$$\begin{aligned} \text{Maximizar através de } \vec{\alpha}: \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j y_i y_j \vec{x}_i^T \vec{x}_j \\ \text{sujeito a:} \quad & \sum_{i=1}^{N} \alpha_i y_i = 0 \\ & \alpha_i \geq 0, \forall i = 1, 2, ..., N \end{aligned} \tag{4.19}$$

Observa-se também que, por se tratar de um ponto de sela, o produto de cada multiplicador de Lagrange e sua restrição correspondente se anula,

resultando em (4.20):

$$\alpha_i[y_i(\vec{w}^T\vec{x}_i + b) - 1] = 0, \forall i = 1, 2, ..., N \tag{4.20}$$

Assim, α_i só pode ser diferente de 0 para dados que se encontram sobre as margens e, portanto, anulam o termo entre colchetes da Equação 4.20. Tais pontos são os que se localizam mais próximos à superfície de separação e através dos quais a solução do problema é construída. São, assim, denominados dos *vetores de suporte*, os dados mais importantes do conjunto de treinamento para as SVMs (Burgess 1998). Os demais pontos não influenciam a formulação da função de decisão e podem, portanto, ser descartados sob o ponto de vista de definição da solução.

Resta, agora, obter-se os parâmetros do problema primal através do resultado do problema dual. Para a obtenção de $\vec{w}^*$, basta a aplicação da Equação 4.17, previamente apresentada. Daí, nota-se que apenas os pontos com $\alpha \neq 0$, os vetores de suporte, participam do cálculo de $\vec{w}^*$, o que sustenta a afirmação de que os demais pontos podem ser descartados. O valor de b^* é obtido através da Equação 4.20, computando-se a média para todos os vetores de suporte (SVs) (Schölkopf & Smola 2002), conforme expresso na Equação 4.21:

$$b^* = \frac{1}{n_{SV}} \sum_{\vec{x}_j \in \mathbf{SV}} \frac{1}{y_j} - \vec{w}^*.\vec{x}_j = \frac{1}{n_{SV}} \sum_{\vec{x}_j \in \mathbf{SV}} \left(\frac{1}{y_j} - \sum_{\vec{x}_i \in \mathbf{SV}} \alpha_i^* y_i \vec{x}_i.\vec{x}_j \right) \tag{4.21}$$

Daí, obtém-se a função de decisão da SVM (Equação 4.22):

$$f(\vec{x}) = sign\left(f\left(x\right)\right) = sign\left(\sum_{\vec{x}_i \in \mathbf{SV}} \alpha_i^* y_i \vec{x}_i.\vec{x} + b^* \right) \tag{4.22}$$

4.3.2 SVMs de Margens Flexíveis

Entretanto, a restrição de separação completa das classes imposta pelas SVMs de margens rígidas limita bastante sua aplicação, uma vez que a maioria dos problemas práticos tendem a violá-la, seja por superposição inerente entre as classes ou por ruído presente nos dados. De forma a estender o conceito e ampliar sua aplicabilidade, foram desenvolvidas as SVMs de Margens Flexíveis, apresentadas nesta seção.

Para tanto, introduz-se o conceito das *variáveis de folga* $\xi_i \geq 0$, associadas a cada um dos i-ésimos padrões de treinamento. O nome *variáveis de folga* provém do fato de que tais variáveis "relaxam" as restrições impostas ao problema de otimização primal descrito em 4.14, resultando em (Schölkopf & Smola 2002):

$$y_i[\vec{w}^T\vec{x}_i + b] \geq 1 - \xi_i \tag{4.23}$$

Agora, permite-se a ocorrência de pontos situados entre as margens de separação ($0 < \xi_i \leq 1$), ou mesmo do lado errado da margem($\xi_i > 1$). Daí o nome de SVMs de Margens Flexíveis.

Introduzem-se então os valores de ξ_i, violações da linearidade de separação, na função de custo do problema de otimização 4.14, obtendo-se (Burgess 1998):

$$\begin{aligned} \text{Minimizar através de } (\vec{w}, b, \vec{\varepsilon}): \quad & \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{n}\varepsilon_i \\ \text{sujeito a:} \quad & \forall_{i=1}^{n} : y_i[\vec{w}.\vec{x}_i + b] \geq 1 - \varepsilon_i \end{aligned} \tag{4.24}$$

A constante C pondera os erros de treinamento ($\sum_{i=1}^{n}\varepsilon_i$) e a complexidade do modelo ($\frac{1}{2}\|\vec{w}\|^2$), atuando assim como um regularizador (Passerini 2004).

Novamente, tem-se um problema quadrático que atende às condições do Teorema do Lagrangeano. Por meio de procedimento análogo ao realizado na seção 4.3.1.1, obtém-se a formulação dual para a SVM de margens flexíveis (Equação 4.25):

$$\begin{aligned} \text{Maximizar através de } \vec{\alpha}: \quad & \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \vec{x}_i^T\vec{x}_j \\ \text{sujeito a:} \quad & \sum_{i=1}^{N}\alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C, \forall i = 1, 2, ..., N \end{aligned} \tag{4.25}$$

Tal problema difere do problema de margens rígidas apenas na restrição imposta aos multiplicadores de Lagrange α_i, agora limitados a C. Obtém-se os valores de $\vec{w}^*$ e b^* da mesma forma que o realizado para as SVMs de margens rígidas (conforme descrito na seção 4.3.1.1), com α_i determinado, agora, a partir da Equação 4.25 (Cristianini & Shawe-Taylor 2000; Pontil & Verri 1998).

4.4 Formulação Teórica para Problemas Não-Linearmente Separáveis

Apesar de solucionar a limitação de separabilidade de classes imposta às SVMs de margens rígidas, a formulação das SVMs de margens suaves ainda

incorre em uma severa restrição: a necessidade de linearidade de separação no *espaço de entrada*. Em muitos problemas, é necessário um mapeamento do espaço de entrada em um outro, geralmente (mas não necessariamente) de dimensão mais elevada, onde pode-se, aí sim, efetuar a separação das classes de maneira linear. Esse novo espaço é denominado *espaço de características*. Um problema desse tipo é ilustrado na Figura 4.7.

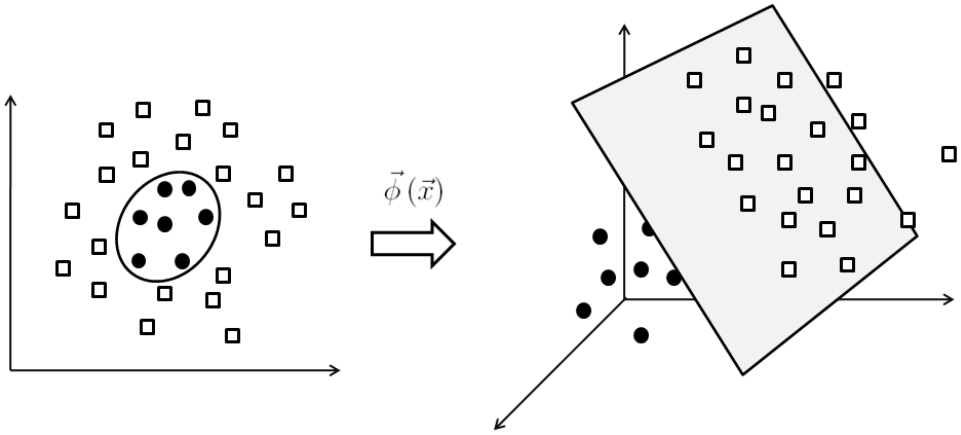


Figura 4.7: Mapeamento do espaço de entrada num espaço de características com separação linear entre as classes

O *Teorema de Cover da Separabilidade de Padrões* (Cover 1965) demonstra que:

> Um problema de classificação de padrões mapeado de maneira não-linear em um espaço de alta dimensão tem maior probabilidade de separação linear que em um espaço de baixa dimensão.

conforme transcrito de (Haykin 1999).

Assim, garantindo-se que o mapeamento realizado seja não-linear e seja efetuado para uma dimensão suficientemente elevada, tem-se uma alta probabilidade de viabilidade de uma solução linear. Nesse novo espaço, pode-se aplicar a mesma formulação de construção do hiperplano ótimo de separação das SVMs, minimizando a dimensão VC e o risco estrutural e maximizando, assim, a generalização da solução obtida.

4.4.1 SVMs Não-Lineares

Considere a função não-linear $\vec{\phi}(\vec{x})$, a qual realiza o mapeamento de $\vec{x}$ do espaço de entrada para o espaço de características, conforme expresso na Equação 4.26:

$$F = \{\vec{\phi}(\vec{x}) \,|\vec{x} \in X\} \tag{4.26}$$

Aplicando-se $\vec{\phi}(.)$ ao problema de otimização apresentado na Equação 4.25 (SVM linear de Margens Flexíveis), obtém-se a seguinte função de custo (Equação 4.27):

$$\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \left(\vec{\phi}(\vec{x}_i)^T \cdot \vec{\phi}(\vec{x}_j) \right) \tag{4.27}$$

Entretanto, o mapeamento para dimensões elevadas leva à chamada *Maldição da Dimensionalidade*, que ocasiona problemas de alta complexidade computacional e até a inviabilidade da solução.

Para a solução de tal limitação, utiliza-se o artifício das funções de Kernel. Um Kernel K é uma função que recebe dois pontos $\vec{x_i}$ e $\vec{x_j}$ do espaço de entradas e computa o produto escalar dos mesmos no espaço de características (Herbrich 2001) (Equação 4.28):

$$K\left(\vec{x_i}, \vec{x_j}\right) = \langle \vec{\phi}\left(\vec{x_i}\right), \vec{\phi}\left(\vec{x_j}\right) \rangle \tag{4.28}$$

Assim, a formulação do Problema de Otimização das SVMs Não-Lineares de Margens Flexíveis, caso mais geral das SVMs, pode ser assim expresso:

$$\begin{aligned} \text{Maximizar:} \quad & J\left(\vec{w}, b, \vec{\xi}, \vec{\alpha}, \vec{r}\right) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} y_i y_j \alpha_i \alpha_j K\left(\vec{x_i}, \vec{x_j}\right) \\ \text{sujeito a:} \quad & 0 \leq \alpha_i \leq C, i = 1, \ldots, \ell \\ & \sum_{i=1}^{\ell} \alpha_i y_i = 0, \end{aligned} \tag{4.29}$$

que resulta na seguinte regra de decisão:

$$f(\vec{x}) = \text{sign}\left(\sum_{i=1}^{\ell} y_i \alpha_i \cdot K\left(\vec{x}, \vec{x_i}\right) + b \right) \tag{4.30}$$

De forma a garantir a convexidade do Problema de Otimização expresso em 4.29 e a realização de um mapeamento no qual é viável o cálculo dos produtos escalares expressos em 4.28, devem-se utilizar *Kernels* que atendem às condições expressas no Teorema de Mercer (Mercer 1909). Algumas dessas funções, comumente utilizadas como funções de *Kernel*, são a RBF, a polinomial e a sigmoidal. As funções sigmoidais, em particular, atendem ao teorema de Mercer apenas para alguns valores de β_0 e β_1 (Haykin 1999).

A simplicidade de cálculo não é a única vantagem das funções de Kernel. Outro grande benefício em seu uso é que o mapeamento para o espaço de características não precisa ser explicitamente calculado, podendo induzir uma dimensão extremamente elevada, ou até mesmo infinita, sem implicar em au-

Tabela 4.1: Exemplos típicos de funções de Kernel

Função de Kernel:	Expressão para $K(x_i, x_j)$:	Parâmetros:
RBF	$e^{-\|x_i - x_j\|^2/2\sigma^2}$	σ^2
Polinomial	$(x_i^T x_j + a)^b$	a, b
Sigmóide	$tanh(\beta_0 x_i^T x_j + \beta_1)$	β_0, β_1

mento da complexidade de cálculo. Assim salienta-se outra diferença entre as redes MLP e as SVMs: Nas redes MLP, a dimensão do espaço de características é finita, controlada pelo número e a magnitude dos parâmetros livres da rede, cujo aumento demasiado leva a complexidade desnecessária da rede e potencialmente *overfitting*. Nas SVMs, tal dimensão pode ser, inclusive, infinita, e mesmo assim a complexidade é controlada através da minimização da dimensão VC e do risco estrutural, mantendo-se assim a capacidade de generalização. Em contrapartida, a resolução de um problema quadrático é exigida pelas SVMs, levando a uma mais alta complexidade computacional que para as redes MLP.

4.5 Conclusões do Capítulo

O profundo embasamento na Teoria de Aprendizado Estatístico observado nas Máquinas de Vetores de Suporte resulta em uma abordagem elegante e fortemente fundamentada para a obtenção de redes neurais de alta capacidade de generalização. Os princípios de minimização do risco estrutural baseados na dimensão VC garantem a robustez do método mesmo em problemas de alta dimensionalidade e complexidade.

Uma observação interessante é que as SVMs controlam a complexidade do modelo obtido independente da dimensionalidade do problema ou do mapeamento realizado, mantendo o poder de generalização mesmo onde a maioria dos algoritmos de aprendizado leva ao *overfitting*.

A natureza quadrática da função objetivo é outro ponto a ser destacado nas SVMs, que não sofrem de problemas de convergência para mínimos locais, comuns às Redes Neurais Artificiais com treinamento baseado no algoritmo *backpropagation* e suas variações.

CAPÍTULO 5

Abordagem proposta e Resultados Obtidos

O presente capítulo propõe uma abordagem evolucionária para o Aprendizado Semi-Supervisionado de Máquinas de Vetores de Suporte (SVMs). Um Algoritmo Genético é inserido na busca pelo conjunto de classificações para o conjunto de teste que maximiza a margem de separação entre as classes, garantindo melhor desempenho e globalidade das soluções obtidas. A natureza estocástica dos Algoritmos Evolucionários torna esta nova abordagem mais propensa a alcançar ótimos globais que as SVMs semi-supervisionadas tradicionais. Um operador de mutação adaptativo por gene, motivado pelo método transdutivo *k-nearest neighbors*, é utilizado como forma de adicionar informação ao processo de busca, acelerando significativamente a convergência.

5.1 Introdução

O objetivo principal da *inferência indutiva* é encontrar uma regra geral para um dado problema. Com base em uma amostragem reduzida dos dados de entrada, esta regra é induzida. Posteriormente, tal regra geral é aplicada para avaliar e classificar dados não apresentados durante a fase de treinamento. As suposições de que existe uma regra geral e de que tal regra pode ser aprendida através de uma amostragem restrita dos dados de entrada são princípios básicos do aprendizado indutivo.

Em algumas situações, entretanto, o desbalanceamento entre o número de padrões de treinamento e de teste restringe a capacidade do conjunto de

treinamento no fornecimento de informação suficiente para a definição de um modelo geral. Em várias situações, o conjunto de pontos não classificados pode ser muitas vezes maior que o de pontos classificados. As origens de tal desbalanceamento vão desde o alto custo na classificação até limitações inerentes ao próprio problema. Para este tipo de situação, uma abordagem alternativa é obter regras específicas para um dado conjunto de pontos que se deseja classificar, ao invés de se procurar por uma regra geral para todo o espaço de entrada. Esta é a inovação da *inferência transdutiva* (Gammerman, Azoury, & Vapnik 1998). Outra abordagem possível é utilizar informações do conjunto de teste, mais numeroso e estatisticamente representativo que o conjunto de treinamento, na composição de uma solução. Este esquema representa o *aprendizado indutivo semi-supervisionado*.

Apesar de o aprendizado indutivo semi-supervisionado ser um conceito geral, podendo ser aplicado a qualquer tipo de aprendizado de máquinas, os resultados mais expressivos foram obtidos com Máquinas de Vetores de Suporte (SVMs) (Cortes & Vapnik 1995; Vapnik 1982). A maximização da margem, implícita à formulação das SVMs, é utilizada como função objetivo aplicada ao conjunto de pontos não classificados para otimizar o desempenho do aprendizado. Nessa abordagem alternativa, denominada Máquinas de Vetores de Suporte Semi-Supervisionadas (*Semi-Supervised Support Vector Machines*, ou *3SVMs*), as classificações dos elementos do conjunto de teste se tornam variáveis para o problema de otimização geral. A solução ótima é aquela que minimiza o erro para o conjunto de treinamento e maximiza a margem dos conjuntos de treinamento e de teste simultaneamente.

As modificações na formulação do problema de otimização das SVMs, necessárias à sua adaptação ao paradigma semi-supervisionado de aprendizado, serão discutidas na seção 5.2. O método TSVM, implementação amplamente utilizada de SVMs semi-supervisionadas, será introduzido na seção 5.3. Na seção 5.4, uma nova abordagem híbrida é apresentada, o GA3SVM, que otimiza o aprendizado semi-supervisionado das SVMs por meio da inserção de um algoritmo evolucionário no processo de busca, contando ainda com um operador de mutação baseado em um método transdutivo, o K-NN. Os resultados obtidos são apresentados na seção 5.5.

5.2 SVMs e 3SVMs

Nas SVMs indutivas supervisionadas, o aprendizado visa a induzir uma função de decisão $f_L : \vec{x} \rightarrow \{-1, 1\}$ baseando-se apenas no conjunto de treinamento (Equação 5.1) (Kasabov & Pang 2003).

$$\begin{aligned} f_L &= L(S_{train}) \\ \text{onde: } S_{train} &= \{(\vec{x_1}, y_1), (\vec{x_2}, y_2), \ldots, (\vec{x_n}, y_n)\} \end{aligned} \tag{5.1}$$

O plano de separação ótimo é aquele que resulta em máxima margem de separação entre as duas classes. Para o caso de classes linearmente separáveis, sua determinação é o resultado da solução do seguinte problema de otimização restrita, já apresentado no Capítulo 4:

Problema de Otimização 1 - SVMs (caso linearmente separável)

$$\begin{aligned} \text{Minimizar através de } (\vec{w}, b): \quad & \frac{1}{2}\|\vec{w}\|^2 \\ \text{sujeito a:} \quad & \forall_{i=1}^{n} : y_i[\vec{w}.\vec{x_i} + b] \geq 1 \end{aligned} \tag{5.2}$$

Para o caso não-separável, modifica-se o Problema de Otimização 1 de forma a permitir pontos classificados erroneamente no treinamento. O resultado é um classificador de margem flexível que penaliza pontos mal classificados pela introdução de um conjunto de variáveis de folga ξ_i como uma medida da violação das restrições (Equação 5.3).

Problema de Otimização 2 - SVMs (caso não-separável)

$$\begin{aligned} \text{Minimizar através de } (\vec{w}, b, \vec{\xi}): \quad & \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{sujeito a:} \quad & \forall_{i=1}^{n} : y_i[\vec{w}.\vec{\phi}(\vec{x_i}) + b] \geq 1 - \xi_i \end{aligned} \tag{5.3}$$

O parâmetro C é um parâmetro de flexibilidade da margem utilizado para ponderar as variáveis de folga. Ele representa a influência dos erros de treinamento e permite ajustar o compromisso entre largura da margem e padrões de treinamento classificados erroneamente. O espaço de entrada e, conseqüentemente, o hiperplano ótimo de separação são projetados em um espaço de características induzido por funções de kernel. Os dados originais $\vec{x} \in X$ são então representados pela relação de mapeamento $\vec{\phi}(\vec{x})$ conforme descrito na Equação 5.4. Isto é equivalente a um mapeamento entre um espaço de entrada X e um novo espaço de características $F = \{\vec{\phi}(\vec{x}) \,|\vec{x} \in X\}$.

$$\vec{x} = (x_1, \ldots, x_\ell) \rightarrow \vec{\phi}(\vec{x}) = (\phi_1(\vec{x}), \ldots, \phi_\ell(\vec{x})). \tag{5.4}$$

Com a utilização de funções de kernel, os produtos internos dos vetores

mapeados $\langle\vec{\phi}(x_i), \vec{\phi}(x_j)\rangle$ podem ser computados diretamente. Os produtos internos $\langle\vec{x_i}, \vec{x_j}\rangle$ da formulação original das máquinas de vetores de suporte podem ser substituidos pela relação:

$$K\left(\vec{x_i}, \vec{x_j}\right) = \langle\vec{\phi}\left(\vec{x_i}\right), \vec{\phi}\left(\vec{x_j}\right)\rangle \tag{5.5}$$

Substituindo (5.5) na formulação dual do Lagrangiano do problema de minimização (5.3), chegamos à formulação tradicional das SVMs indutivas que se segue (já apresentada no capítulo 4, eq. 4.29).

$$\begin{aligned} \text{Maximizar:} \quad & J\left(\vec{w}, b, \vec{\xi}, \vec{\alpha}, \vec{r}\right) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} y_i y_j \alpha_i \alpha_j K\left(\vec{x_i}, \vec{x_j}\right) \\ \text{sujeito a:} \quad & 0 \leq \alpha_i \leq C, i = 1, \ldots, \ell \\ & \sum_{i=1}^{\ell} \alpha_i y_i = 0, \end{aligned}$$

que resulta na seguinte regra de decisão:

$$f(\vec{x}) = \text{sign}\left(\sum_{i=1}^{\ell} y_i \alpha_i \cdot K\left(\vec{x}, \vec{x_i}\right) + b\right) \tag{5.6}$$

Diferentemente do que ocorre no aprendizado supervisionado de SVMs, na abordagem semi-supervisionada o conjunto de teste é utilizado como fonte adicional de informação. Assim, a função a ser aprendida (Equação 5.1) pode ser reescrita como se observa na Equação 5.7:

$$\begin{aligned} f_L &= L(S_{train}, S_{test}) \\ \text{onde: } S_{train} &= \{(\vec{x_1}, y_1), (\vec{x_2}, y_2), \ldots, (\vec{x_n}, y_n)\} \\ S_{test} &= \{\vec{x_1^*}, \vec{x_2^*}, \ldots, \vec{x_k^*}\} \end{aligned} \tag{5.7}$$

O objetivo do aprendizado semi-supervisionado é utilizar informações dos pontos não classificados para a indução da regra de decisão. Para as 3SVMs (*Semi-Supervised Support Vector Machines*), tais classificações tornam-se variáveis para o problema de otimização, levando, para o caso linearmente separável, a (Vapnik 1998):

Problema de Otimização 3 - 3SVMs (caso linearmente separável)

$$\begin{aligned} \text{Minimizar através de } (y_1^*, \ldots, y_k^*, \vec{w}, b): \quad & \frac{1}{2}\|\vec{w}\|^2 \\ \text{sujeito a:} \quad & \forall_{i=1}^{n} : y_i[\vec{w}.\vec{x_i} + b] \geq 1 \\ & \forall_{j=1}^{k} : y_j^*[\vec{w}.\vec{x_j^*} + b] \geq 1 \end{aligned} \tag{5.8}$$

Para lidar com dados não-separáveis, variáveis de folga e mapeamento por funções de kernel são introduzidos, similarmente ao que é feito em SVMs indutivas (Equação 5.9).

Problema de Otimização 4 - 3SVMs (caso não-linearmente separável)

$$\begin{aligned} \text{Minimizar através de } (y_1^*, \ldots, y_k^*, \vec{w}, b, \vec{\xi}, \vec{\xi^*}): \quad & \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{n}\xi_i + C^*\sum_{j=1}^{k}\xi_j^* \\ \text{sujeito a:} \quad & \forall_{i=1}^{n} : y_i[\vec{w}.\vec{\phi}(\vec{x_i}) + b] \geq 1 - \xi_i \\ & \forall_{j=1}^{k} : y_j^*[\vec{w}.\vec{\phi}(\vec{x_j^*}) + b] \geq 1 - \xi_j^* \end{aligned} \tag{5.9}$$

Agora dois parâmetros regulam a flexibilidade da margem, C e C^*, controlando a influência dos erros de treinamento e de teste, respectivamente. Analogamente ao caso indutivo, multiplicadores de Lagrange e espaços de características kernel-induzidos são introduzidos:

$$\begin{aligned} \text{Maximizar:} \quad & J\left(\vec{w}, b, \vec{\xi}, \vec{\alpha}, \vec{r}\right) = \sum_{i=1}^{\ell}\alpha_i + \sum_{j=1}^{k}\alpha_j^* - \frac{1}{2}\sum_{i,r=1}^{\ell} y_i y_r \alpha_i \alpha_r K\left(\vec{x_i}, \vec{x_r}\right) \\ & -\frac{1}{2}\sum_{j,r=1}^{\ell} y_j^* y_r^* \alpha_j^* \alpha_r^* K\left(\vec{x_j^*}, \vec{x_r^*}\right) - \sum_{j=1}^{\ell}\sum_{r=1}^{k} y_j y_r^* \alpha_j \alpha_r^* K\left(\vec{x_j}, \vec{x_r^*}\right) \\ \text{sujeito a:} \quad & 0 \leq \alpha_i \leq C, i = 1, \ldots, \ell \\ & 0 \leq \alpha_j^* \leq C^*, j = \ell + 1, \ldots, \ell + k \\ & \sum_{i=1}^{\ell}\alpha_i y_i + \sum_{j=\ell+1}^{\ell+k}\alpha_j^* y_j^* = 0. \end{aligned} \tag{5.10}$$

Os parâmetros encontrados com a solução do problema (5.10) podem ser utilizados para construir uma regra de decisão, como no caso supervisionado:

$$f(\vec{x}) = \text{sign}\left(\sum_{i=1}^{\ell} y_i \alpha_i \cdot K\left(\vec{x}, \vec{x_i}\right) + \sum_{j=1}^{k} y_j^* \alpha_j^* \cdot K\left(\vec{x}, \vec{x_j^*}\right) + b\right). \tag{5.11}$$

5.3 TSVM - Transductive Support Vector Machine

Um dos métodos mais difundidos para a resolução do problema de aprendizado semi-supervisionado de SVMs apresentado na seção 5.2 é o algoritmo TSVM (*Transductive Support Vector Machine*), proposto por Joachims (Joachims 1999). Apesar de ter sido originalmente classificado como *transdutiva*, tal abordagem é, na realidade, semi-supervisionada, uma vez que define uma função de decisão aplicada a pontos não apresentados durante a etapa de treinamento, cenário não aderente ao paradigma transdutivo, conforme discutido no Capítulo 2 (seção 2.2.3).

O TSVM foi desenvolvido para aplicação em problemas de classificação de textos da internet. Tais problemas sofrem de um grande desbalanceamento entre o baixo número de padrões de treinamento e o elevado tamanho do conjunto de teste. É exatamente nesse contexto que a abordagem semi-supervisionada tem maior atratividade.

O algoritmo proposto por Joachims segue o seguinte esquema: partindo-se da classificação da SVM indutiva para ambos os conjuntos de treinamento e de teste, trocam-se os rótulos de pontos situados próximos à margem de separação, 2 a 2, de forma a buscar melhorias na função de custo apresentada na Equação 5.9, a margem de separação entre os conjuntos de treinamento e de teste. Apenas trocas que resultem em aumento da margem são mantidas. Repete-se o procedimento até que não haja pontos classificados erroneamente, aumentando-se o valor do parâmetro C^*, que regula a penalização imposta às classificações incorretas do conjunto de teste. Tal abordagem pode ser compreendida como uma busca local exaustiva nos pontos situados próximos às margens de separação.

O critério de escolha dos pontos a terem classificações trocadas é baseado nos valores das variáveis de folga ξ_i a eles associadas. Testam-se todos os pares em que cada variável de folga é superior a 0 e cuja soma das mesmas é superior a 2.

No Capítulo 4, mostrou-se que os pontos com $0 < \xi_i < 1$ estão situados do lado correto do hiperplano de separação, mas dentro da margem de sua classe, enquanto os pontos com $\xi_i > 1$ estão localizados do lado oposto do hiperplano de separação, o que sustenta a afirmação de que a busca é localizada nas proximidades das margens de separação entre as classes.

Essa não é uma abordagem ótima, já que sua busca exaustiva pode ser demasiadamente cara e ainda assim não resultar em soluções globalmente ótimas (convergindo para mínimos locais). Outra limitação das TSVMs propostas por Joachims é que deve-se definir *a priori* o número de padrões de teste a serem rotulados como pertencentes a cada uma das classes, o que

exige conhecimento prévio acerca da distribuição de probabilidade da amostra de teste, normalmente desconhecida em problemas semi-supervisionados.

No presente trabalho, a TSVM proposta por Joachims é estendida por meio da introdução de um *Algoritmo Genético (GA)* na busca pelo melhor conjunto de classificações. A utilização de um operador de mutação adaptativo por gene, motivado pelo método transdutivo k-NN, adiciona informação ao processo de busca e acelera significativamente a convergência.

5.4 O Método Proposto

5.4.1 Motivação

A solução das equações descritas na seção 5.2, ou seja, o treinamento semi-supervisionado de SVMs, consiste em se encontrar um conjunto de classificações $\{y_1^*, \ldots, y_k^*\}$ para os dados de teste e um hiperplano $< \vec{w}, b >$ de forma que tal hiperplano separe tanto o conjunto de treinamento quanto o de teste com máxima margem (Joachims 1999). Isto é obtido através da minimização da função objetivo (5.9), sendo esta, então, a função de desempenho para nosso problema de otimização.

Para o caso de reconhecimento de padrões onde $y \in \{-1, 1\}$, isto equivale a procurar entre 2^k possíveis combinações de classificações. Dessa forma, o treinamento semi-supervisionado de SVMs pode ser visto como a resolução de um problema *minimax* (Vapnik 1998), para o qual utilizamos algoritmos de heurística para lidar com tal espaço exponencial de busca. Para um pequeno número de pontos de teste, tal problema pode ser resolvido simplesmente testando todas as atribuições possíveis de $y_1^*, \ldots, y_k^*$ às duas classes (Joachims 1999). Entretanto, tal abordagem torna-se impraticável para grandes conjuntos de teste.

Conforme discutido na seção 5.3, no algoritmo TSVM proposto por Joachims (Joachims 1999), a procura pelo melhor conjunto de classificações para os dados de teste começa com a atribuição da SVM indutiva, melhorando a solução através da troca de classificações que resultem exclusivamente em decrescimento da função objetivo. É exatamente esta restrição à degradação momentânea do desempenho que torna tal abordagem tão propensa à convergência para mínimos locais. Um algoritmo iterativo similar é descrito em (Chen, Wang, & Dong 2003). Abordagens baseadas em técnicas de *clustering* são também comuns, como a encontrada em (Joachims 2003).

No presente trabalho propõe-se o uso de um Algoritmo Genético para efetuar a busca pelo melhor conjunto de classificações para os dados de teste. A natureza estocástica dos GAs torna essa abordagem mais propensa a atingir

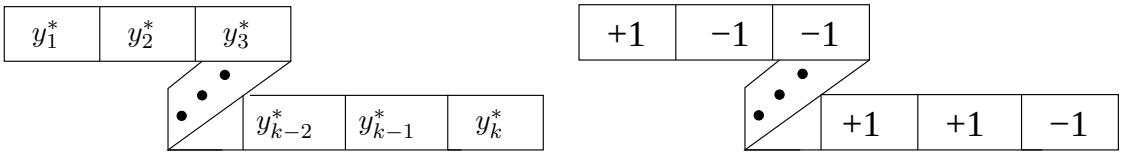
ótimos globais que as abordagens tradicionais. Utiliza-se, ainda, um operador de mutação baseado no método transdutivo k-NN, que orienta e adiciona informação ao processo de busca. Este método foi primeiramente proposto em (Silva, Maia, & Braga 2005).

5.4.2 GA3SVM

Algoritmos Genéticos (Holland 1975; Goldberg 1989) são métodos de otimização estocástica baseados na Teoria Evolucionista de Darwin. Conforme discutido no Capítulo 3, o GA inicia o processo de evolução com um conjunto de indivíduos (representados por cromossomos) denominado população. Alguns indivíduos são então selecionados, segundo seu desempenho, para reproduzirem e formarem a próxima geração. Quanto mais adaptados estão ao ambiente, maior chance terão de serem selecionados, carregando suas características para as gerações por vir. Através da utilização de operadores genéticos (tais como seleção, cruzamento e mutação), a população como um todo evolui, ao invés de uma única solução. A natureza estocástica dos GAs os torna menos propensos a ficarem presos a mínimos locais, que estão inevitavelmente presentes em qualquer problema prático de otimização.

As principais características do GA proposto são introduzidas nesta seção.

Representação : Cada conjunto possível de classificações para os padrões de teste será representado por um cromossomo. Assim, a classificação de cada ponto do conjunto de teste é dada por um gene nestes cromossomos, conforme observado na Figura 5.1.



(a) Cromossomo (b) Exemplo de solução

Figura 5.1: Exemplo de solução para o respectivo cromossomo

Inicialização : A população inicial pode conter um ou mais indivíduos definidos pela classificação de SVMs indutivas com aprendizado supervisionado utilizando o conjunto de treinamento. Através deste procedimento, a convergência tende a ser acelerada, com o contratempo de se introduzir um mínimo local no processo.

Avaliação : Para cada conjunto de classificações do conjunto de teste, representado por um indivíduo, sua função de desempenho será dada pelo valor da função de custo expressa na Equação 5.9. Para se evitar convergência prematura, utilizou-se escalonamento linear do desempenho.

Seleção : O Método da Roleta foi utilizado na seleção de indivíduos para formarem a nova geração.

Cruzamento : Foi utilizado cruzamento com dois pontos de corte aleatórios. Além disso, observando-se que o operador de cruzamento é altamente importante nos estágios iniciais de evolução, tendo sua importância reduzida com a população aproximando-se da convergência, utilizou-se o procedimento de redução linear da probabilidade de cruzamento (P_c) com o passar das gerações, conforme observado na Figura 5.2.

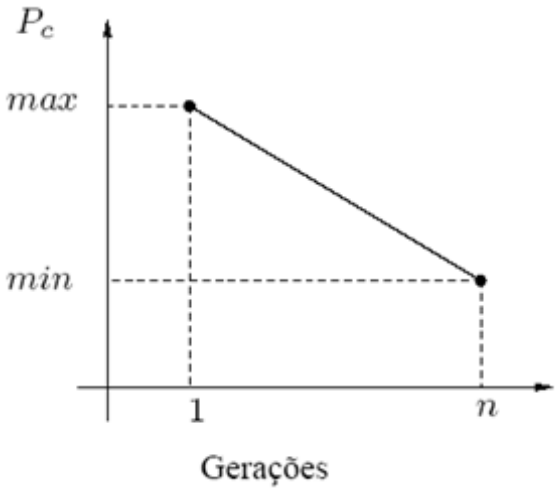


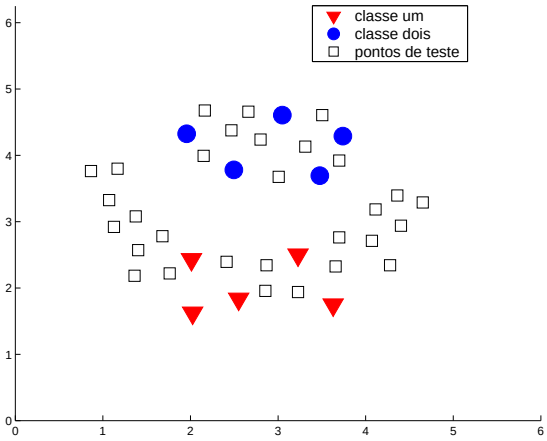
Figura 5.2: Probabilidade de cruzamento com o passar das gerações

Mutação : Em operadores padrão de mutação, uma probabilidade constante de ocorrência (P_m) é utilizada para cada gene de cada indivíduo de toda a população. No presente trabalho é proposta uma *probabilidade de mutação adaptativa por gene*, inspirada no método k-NN, conforme explicado na Seção 5.4.2.1.

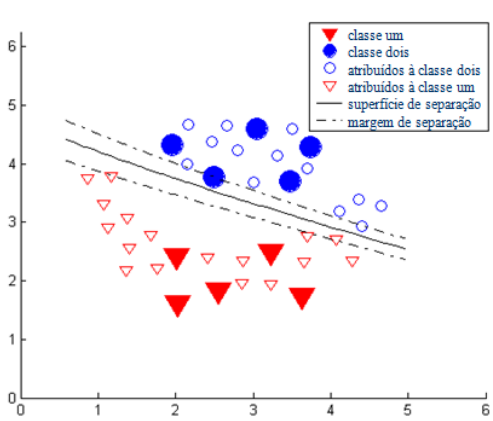
5.4.2.1 O Operador de Mutação Adaptativo por Gene

Nos estágios iniciais de teste do método proposto, algumas dificuldades de convergência foram encontradas. Um número demasiadamente alto de gerações era necessário para se atingir o ótimo global. Conforme observado, uma das principais razões para tal comportamento era a existência de *outliers* nos indivíduos. Isto é, algumas vezes, um único ponto mal classificado poderia levar o desempenho do indivíduo (a margem associada) a um valor extremamente pequeno. Para problemas linearmente separáveis, um único ponto mal classificado pode impactar ainda mais negativamente a margem, já que a superfície de separação deve envolvê-lo. Para problemas não-linearmente separáveis, um único *outlier*, distante da margem, pode resultar em uma alta penalidade à função objetivo. Em ambos os casos, uma boa solução poderia

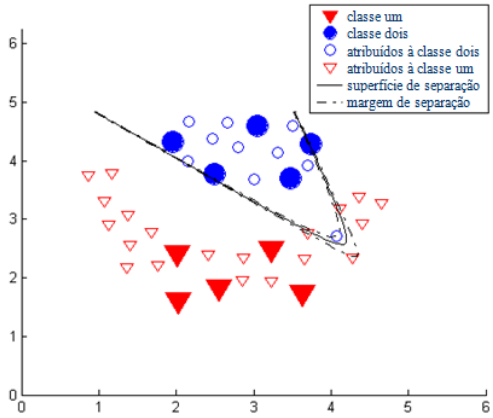
ser ignorada e até descartada com o passar das gerações. O exemplo a seguir ilustra esse problema.



(a) O problema do U



(b) O mínimo local



(c) A solução com um *outlier*

Figura 5.3: Motivação para o desenvolvimento do operador de mutação modificado

Considere o problema separável da Figura 5.3(a). Os pontos não classificados, representados por pequenos quadrados pretos, são os genes de nossos indivíduos, conforme explicado na Seção 5.4.2. Considere que, durante a evolução, uma solução como aquela apresentada na Figura 5.3(b) seja alcançada. A margem associada a esse indivíduo, que pode ser visualizada graficamente como a distância entre as linhas sólida e pontilhada, é relativamente grande, tornando essa solução um mínimo local (apenas para comparação numérica, o valor calculado para a margem, nesse caso, foi de $0,62$).

Assuma que, em uma geração subsequente, um cruzamento ou até mesmo uma mutação neste indivíduo o transforme na solução da Figura 5.3(c). Apesar desta última representar, claramente, uma solução muito superior, a margem a ela associada é bastante pequena (em termos numéricos, seu valor é de $0,01$). Assim, tal solução poderia ser negligenciada no processo de seleção, desaparecendo com o passar de gerações, a não ser que uma mutação "cirúr-

gica" ocorresse exatamente neste único ponto incorretamente classificado.

Situações como essa ocorrem diversas vezes ao longo do processo evolutivo, resultando no descarte de indivíduos de alto potencial e atrasando significativamente a convergência. Esta é a motivação do operador de mutação adaptativo por gene, proposto a seguir.

A motivação para esse operador é o algoritmo transdutivo k-NN, introduzido no Capítulo 2 (Seção 2.2.2.1). Para cada ponto, a probabilidade de mutação deve depender das classificações de seus vizinhos mais próximos. Caso, como observado no exemplo acima, todos os k vizinhos mais próximos tenham classificações diferentes daquela atribuída ao ponto em questão, a probabilidade de mutação deste gene deve ser elevada. No caso oposto, se todos os k vizinhos mais próximos tiverem a mesma classificação do ponto dado, sua probabilidade de mutação deve ser reduzida. Assim, a probabilidade de mutação seria um valor que, para cada gene, dependesse das classificações dos seus k-Vizinhos mais próximos (Figura 5.4).

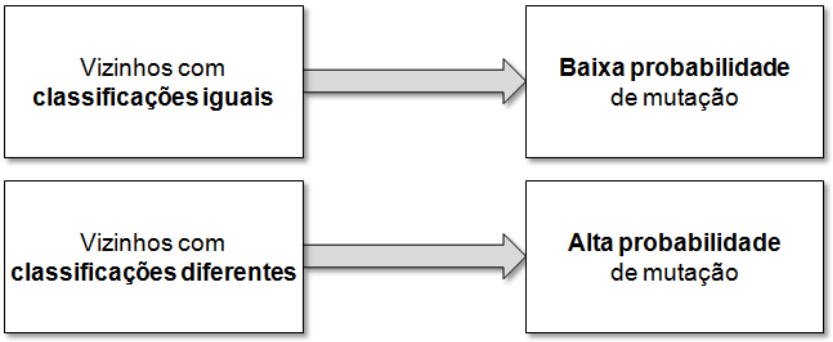


Figura 5.4: Probabilidade de mutação segundo classificações dos vizinhos

---

Definição: O Operador de Mutação Adaptativo por Gene

Seja y_i^* a classificação atribuída por um indivíduo a um dado ponto de teste (um gene). Chamamos D_{nXn} a matriz que comporta as distâncias entre todos os pares de genes no espaço de entrada ($D_{nXn} = \{d_{i,j}\}$, onde $d_{i,j}$ é a distância entre os genes i e j). Seja, então, D_{nXk}^{min} a matriz que compreende apenas as k menores distâncias (as distâncias aos k vizinhos mais próximos) de cada gene, e, Y_{nXk}^{NN}, os rótulos correspondentes (as classificações dos k vizinhos mais próximos), ou seja, $Y_{nXk}^{NN} = \{y_{i,j}^{NN}\}$ onde $y_{i,j}^{NN}$ é o rótulo do j-*ésimo* vizinho mais próximo ao gene i. A probabilidade de mutação p_i^m para um dado gene será dada, então, por:

$$p_m^i = p_m^{max} \frac{1}{k} \sum_{j=1}^{k} (y_i^* \neq y_{i,j}^{NN}) \tag{5.12}$$

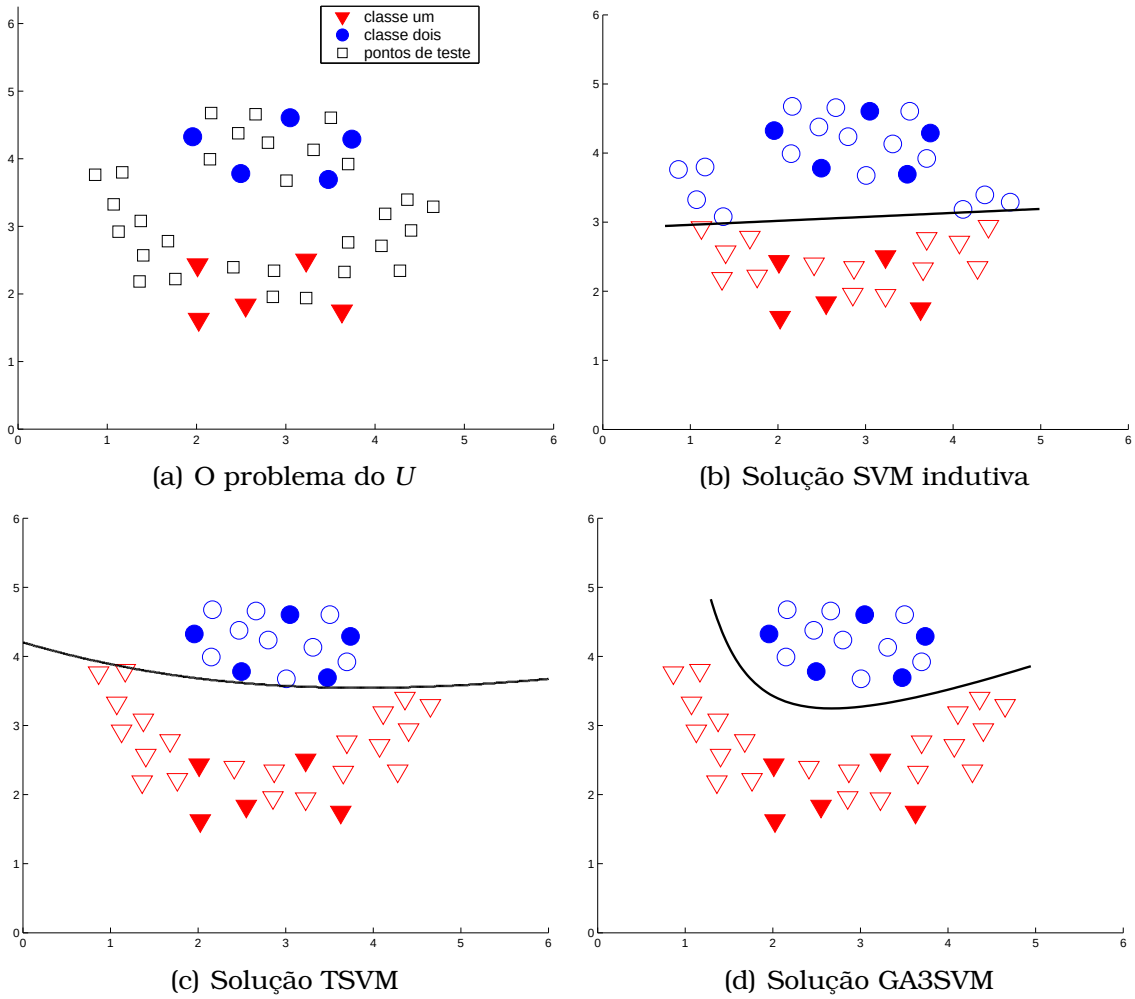
A desigualdade em parênteses resulta em 1 caso $(y_i^* \neq y_{i,j}^{NN})$ e em 0 caso $(y_i^* = y_{i,j}^{NN})$.

O valor de p_m^{max} é um parâmetro definido pelo usuário, análogo à constante p_m utilizada em algoritmos genéticos em geral.

5.5 Resultados

Para demonstrar a eficiência do método proposto, foram solucionados dois problemas artificialmente gerados, que possibilitam uma fácil, e bastante útil, visualização dos resultados, como pode ser observado nas Figuras 5.5(a) e 5.6(a). Ambos são problemas comumente utilizados para se demonstrar a eficiência de métodos transdutivos e semi-supervisionados.

O primeiro é usualmente denominado *Problema do U*. Nele, apenas 5 pontos pertencem ao conjunto de treinamento para cada classe, sugerindo uma separação linear entre as mesmas. Entretanto, ao se introduzir o conjunto de teste, observa-se que tais conjuntos não apresentam amostragens *i.i.d.* e, assim, a separação ideal se parece mais com a forma de um "U".



(a) O problema do *U*

(b) Solução SVM indutiva

(c) Solução TSVM

(d) Solução GA3SVM

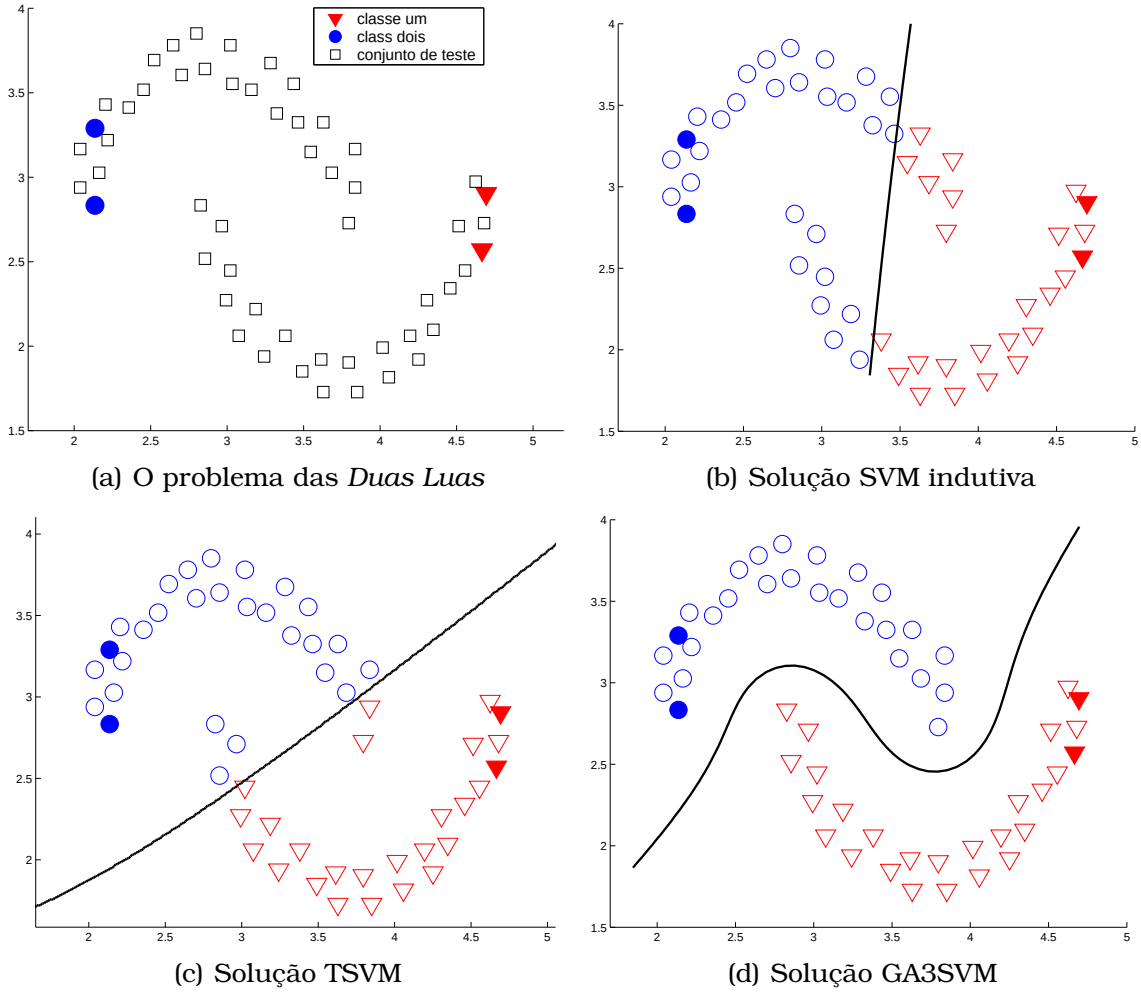
Figura 5.5: Problema do *U* solucionado pelos métodos comparados

O segundo é o *Problema das Duas Luas*. Neste, apenas dois pontos por classe contêm rotulos, sugerindo, novamente uma separação linear. Entre-

tanto, ao se introduzir os elementos do conjunto de teste, muito mais numeroso que o de treinamento, percebe-se que a separação ideal se parece mais com as “Duas Luas”.

Em ambos os problemas, a separação visual entre as classes é bastante clara. Entretanto, os conjuntos de treinamento não representam, de forma eficiente, as distribuições dos conjuntos de teste.

Para o problema do *U* (Figura 5.5(a)), pode-se observar que ambos TSVM (Figura 5.5(c)) e GA3SVM (Figura 5.5(d)) fornecem classificação correta para todo o conjunto de teste, superando de longe as SVMs indutivas tradicionais (Figura 5.5(b)). Entretanto, para o problema das *Duas Luas* (Figura 5.6(a)), o algoritmo TSVM (Figura 5.6(c)) tem desempenho superior ao da SVM indutiva (Figura 5.6(b)), ficando preso, porém, em um mínimo local durante o processo de busca. O algoritmo GA3SVM, novamente, atinge o mínimo global (margem máxima), resultando em classificação correta para todo o conjunto de teste (Figura 5.6(d)).



(a) O problema das *Duas Luas*

(b) Solução SVM indutiva

(c) Solução TSVM

(d) Solução GA3SVM

Figura 5.6: Problema das *Duas Luas* solucionado pelos métodos comparados

O efeito do operador de mutação adaptativo por gene pode ser observado na Figura 5.7. Nela, observa-se a evolução do desempenho do melhor indivíduo

no método GA3SVM e depois da inclusão de tal operador. O novo método foi chamado de *Genetic Algorithm Semi-Supervised Support Vector Machines - Gene-Dependent Mutation Probability*, ou *GA3SVM-GDMP*.

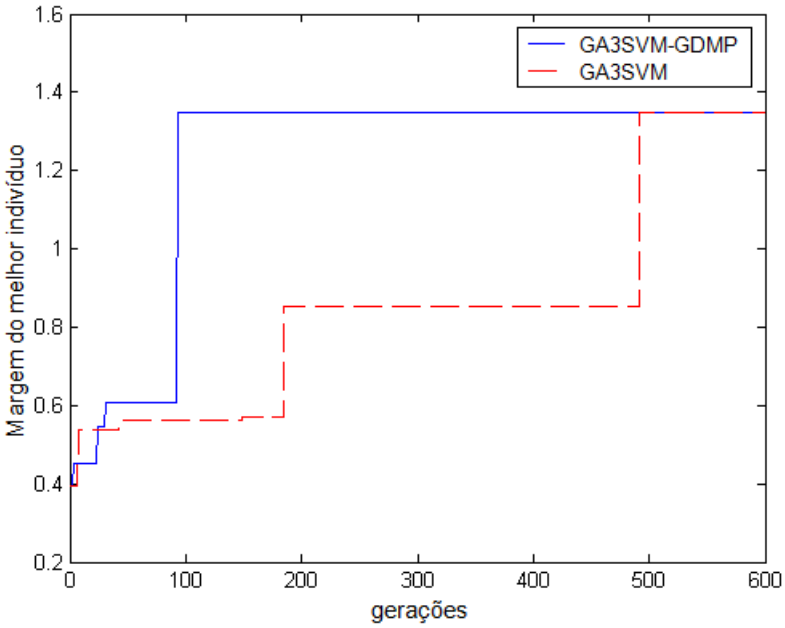


Figura 5.7: Comparação entre a velocidade de convergência dos métodos GA3SVM e GA3SVM-GDMP

Com os testes conduzidos, a utilização do operador de mutação proposto reduziu o número de gerações para alcançar a convergência em aproximadamente 5 vezes, o mesmo observado para esta execução do problema do U (Figura 5.7).

O próximo teste para a metodologia proposta seria a aplicação em bases de dados não-sintéticas, com grande número de padrões em alta dimensão. Dentre as bases a serem utilizadas para tais testes destacam-se aquelas utilizadas na proposição das TSVMs (classificação de textos da internet) (Joachims 1999). Entretanto, a metodologia proposta ainda apresenta custo computacional proibitivamente alto para a realização de tais testes, o que limita, ainda, sua utilização em aplicações reais com a tecnologia atual.

5.6 Conclusões do Capítulo

A utilização de informação adicional extraída do conjunto de teste, assim como a formulação geral de aprendizado semi-supervisionado, podem resultar em avanços significativos na área de aprendizado de máquinas. Esta abordagem pode resultar em grande ganho no tratamento de alguns problemas atualmente em voga, como Bioinformática e procura na Web, já que a imensa quantidade de dados não classificados pode ser, agora, utilizada na própria

formulação do problema. O grande desafio a esse novo paradigma está em definir que tipo de informação deve ser extraída desses dados não classificados. O presente enfoque é em maximização da margem, mas várias novas abordagens podem ainda ser exploradas.

O uso de algoritmos evolucionários na busca pela classificação correta dos dados de teste, apresentado no presente trabalho, pode fornecer novos caminhos para avanços futuros na área. A significativa redução no erro e a maior possibilidade de convergência globalmente ótima são algumas das contribuições da abordagem evolucionária apresentada. O alto custo computacional resultante da metodologia limita, entretanto, a aplicabilidade da abordagem a problemas reais, com o nível tecnológico atual.

CAPÍTULO 6

Conclusões e Propostas de Continuidade

Este capítulo apresenta as conclusões e principais discussões ocasionadas pelo trabalho realizado. As principais contribuições e limitações da abordagem proposta são descritas, assim como propostas de ajustes para contornar tais restrições. Outras propostas de continuidade mais amplas são também aqui relacionadas.

6.1 Conclusões

A ciência da Inteligência Computacional tem mostrado resultados promissores, com ampla difusão e diversas aplicações práticas de sucesso. Sistemas inteligentes ampliam sua participação, dia após dia, em nossas atividades cotidianas, maximizando nossa produtividade e facilitando nossas vidas. Os avanços no campo teórico são igualmente importantes ao desenvolvimento e à popularização do aprendizado de máquinas. Novos métodos, paradigmas, e até novos problemas ampliam o entendimento e o leque de alternativas disponíveis, levando a uma crescente qualidade e aplicabilidade das metodologias desenvolvidas.

Nesse processo de construção e desenvolvimento do conhecimento, a natureza tem se mostrado uma enorme fonte de inspiração para a comunidade de Inteligência Computacional. O funcionamento dos neurônios biológicos estimulou o desenvolvimento das Redes Neurais Artificiais. A evolução das espécies foi "replicada" para um ambiente simulado com a chamada Computação Evolucionária, levando a grandes ganhos no campo da otimização. Revoadas

de pássaros, colônias de formigas, células imunológicas humanas, virtualmente tudo o que se observa na natureza demonstra uma grande sabedoria e razão de ser, e uma imensa aplicabilidade a uma vasta gama de problemas.

O presente trabalho busca compreender as diferentes dimensões de aprendizagem de máquinas, as nuances de cada metodologia, suas vantagens, desvantagens e relacionamentos, na construção de uma nova e diferente abordagem. Parte-se de uma visão holística do campo de Inteligência Computacional, realizando-se relacionamentos e conexões entre diferentes linhas de pesquisa, levando à construção de uma proposta híbrida, que alia particularidades de cada método na construção de uma abordagem superior para a classe de problemas estudada, em que o conjunto de treinamento não fornece informação suficiente para a definição de uma superfície de separação entre duas classes com alta capacidade de generalização, devido ao seu limitado tamanho e à pequena capacidade do mesmo de descrever o sistema como um todo.

O resultado é um método que une conceitos de diferentes paradigmas: as *Máquinas de Vetores de Suporte*, Redes Neurais Artificiais de *aprendizado indutivo e supervisionado*, são aplicadas em um cenário *semi-supervisionado*, com um *algoritmo evolucionário* realizando a busca no espaço de soluções, orientada por um método *transdutivo*.

O célebre Teorema *No Free Lunch* nos mostra: não há uma abordagem genérica superior em toda gama de problemas. Abordagens híbridas, que constroem uma metodologia específica pela seleção de diferentes métodos segundo suas características e aquelas do problema que se deseja resolver, tendem a obter resultados superiores para uma dada tarefa de aprendizado. Não representam necessariamente, entretanto, abordagens superiores quando aplicadas a uma outra classe de problemas.

6.2 Principais contribuições e propostas de continuidade

O presente trabalho desafia um consagrado método de aprendizado de máquinas, as TSVMs, quanto à otimalidade do processo de busca por ele empregado. Introduz-se um Algoritmo Genético para a realização de tal tarefa, o que confere ao novo método maior capacidade de generalização e globalidade das soluções obtidas. Trabalha-se em um paradigma semi-supervisionado de aprendizado, em que os conjuntos de treinamento e de teste não representam, necessariamente, amostragens *i.i.d.* do espaço de entrada. Essa característica resulta em uma necessidade de *agrupamento* na orientação do processo de busca, introduzida no algoritmo genético utilizado através de um operador

de mutação modificado, que aumenta (ou diminui) a probabilidade de mutação de um dado gene de acordo com as classificações de seus vizinhos mais próximos. Os resultados obtidos demonstram a superioridade do método proposto em termos de capacidade de generalização para a classe de problemas estudada.

Entretanto, apesar de comprovadamente introduzir informação no processo de busca e resultar em convergência acelerada, a metodologia resultante ainda é demasiadamente cara computacionalmente. A necessidade de solução de uma nova SVM na avaliação do desempenho de cada indivíduo da população, a cada geração, introduz um custo elevadíssimo e, algumas vezes, proibitivo para aplicações reais.

6.2.1 Aprimoramento da abordagem proposta

De forma a contornar tal limitação, uma primeira alternativa é a implementação do algoritmo genético desenvolvido utilizando-se programação paralela, de implementação bastante simples e direta devido à própria natureza dos GA's. Como a etapa de maior custo computacional do método está na avaliação da função objetivo, que pode ser realizada de maneira completamente descentralizada, os ganhos de tal abordagem se aproximarão ao número de nodos de processamento utilizados, resultado excelente sob o ponto de vista da capacidade de paralelização, viabilizando sua aplicação em um maior conjunto de problemas.

Outra alternativa é o enfoque no desenho de novos operadores genéticos na aceleração da convergência do GA utilizado. Assim como no operador de mutação modificado proposto, novas heurísticas podem direcionar a seleção e o cruzamento de indivíduos na formação de novas gerações, utilizando, por exemplo, medidas de dispersão e distanciamento entre as classes. Dentre as ferramentas que realizam tal tarefa pode-se citar o Discriminante de Fisher (Fisher 1936).

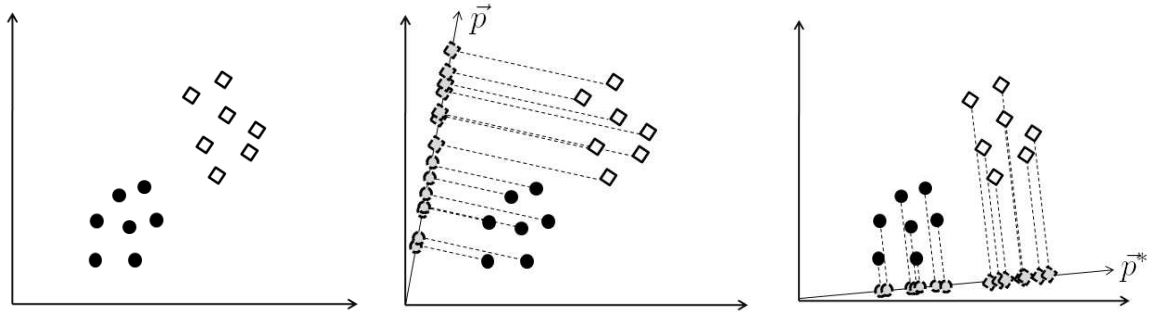
Pode-se, ainda, substituir o Algoritmo Genético utilizado por outros métodos de otimização no processo de busca pelo conjunto de classificações para os padrões de teste que maximiza a margem entre ambos os conjuntos de treinamento e teste. As abordagens evolucionárias *Particle Swarm Optimization* e *Ant Colony Optimization*, descritas no Capítulo 3 (seção 3.2.2), entre outras, poderiam ser utilizadas na realização dessa tarefa.

6.2.2 Extensão do aprendizado semi-supervisionado a outras máquinas de aprendizado

Os resultados mais expressivos da aplicação do paradigma semi-supervisionado foram obtidos com as Máquinas de Vetores de Suporte (Cortes & Vapnik 1995; Vapnik 1982). Entretanto, pode-se estender tal abordagem a outras classes de máquinas de aprendizado. Como propostas de continuidade deste trabalho, descrevem-se a aplicação do paradigma semi-supervisionado de aprendizado aos *Kernel Fisher Discriminants* e às Redes *Multi-Layer Perceptron*.

Semi-Supervised Kernel Fisher Discriminant Os métodos denominados *Discriminantes* abordam o seguinte problema: dado um problema de classificação, qual o melhor conjunto de "características" para separar (ou discriminar) as classes da melhor maneira? (Mika, Ratsch, Weston, Scholkopf, & Mullers 1999). Um dos métodos de maior utilização desta classe é o *Discriminante de Fisher* (Fisher 1936).

Neste método, utiliza-se um vetor $\vec{p}$ sobre o qual são projetados os pontos das classes a serem separadas. Busca-se a direção de $\vec{p}^*$ que maximiza a separação entre as médias das classes e minimiza, ao mesmo tempo, a dispersão de cada classe. Tal abordagem pode ser compreendida através da Figura 6.1.



(a) Classes a serem separadas (b) Projeção sobre um vetor $\vec{p}$ qualquer (c) Projeção sobre o vetor ótimo $\vec{p}^*$

Figura 6.1: Exemplo de aplicação do Discriminante de Fisher

Caso a separação linear não seja possível, pode-se estender tal método para a aplicação no espaço de características. Para tanto, o mesmo procedimento de indução do espaço de características através de funções de Kernel realizado pelas SVMs pode ser utilizado, resultando nos denominados *Kernel Fisher Discriminants*, conforme apresentado em (Mika 2002). No mesmo trabalho, mostra-se que os resultados dos KFDs são comparáveis aos obtidos pelas SVMs em um conjunto de problemas estudados.

Tal método poderia, então, ser adaptado ao paradigma Semi-Supervisionado. Ao se introduzir o conjunto de teste, com padrões não-classificados, no cál-

culo do discriminante de Fisher, buscar-se-ia o conjunto de classificações e o vetor $\vec{p}^*$ que maximizassem a separação e minimizassem a dispersão entre ambos os conjuntos de treinamento e de teste.

Redes MLP Semi-Supervisionadas Poder-se-ia, ainda, ampliar o conceito do discriminante de Fisher para a obtenção de Redes *Multi-Layer Perceptron* Semi-Supervisionadas. Conforme discutido no Capítulo 2, a camada escondida das redes MLP realiza um mapeamento do espaço de entrada em um espaço de características onde, espera-se, seja viável a separação linear entre as classes. Em um paradigma semi-supervisionado, o treinamento das redes seria agora realizado em conjunto com uma busca pelas melhores classificações para o conjunto de teste. Pode-se, então, utilizar o Discriminante de Fisher como uma medida da adequação do conjunto de classificações atribuído aos padrões de teste. A solução semi-supervisionada seria a Rede MLP treinada com os padrões de treinamento e de teste que maximizem o discriminante de Fisher no espaço de características induzido pela camada escondida.

Referências

Alpaydin, E. (2004). *Introduction to Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Arabshahi, P., A. Gray, I. Kassabalidis, M. El-Sharkawi, R. M. II, A. Das, & S. Narayanan (2001). Adaptive routing in wireless communication networks using swarm intelligence. In *9th AIAA Int. Communications Satellite Systems Conf., 17-20 April 2001, Toulouse, France*.

Baker, J. E. (1987). Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, Mahwah, NJ, USA, pp. 14–21. Lawrence Erlbaum Associates, Inc.

Bandyopadhyay, S. & S. Pal (2007). *Classification and Learning Using Genetic Algorithms*. Springer Berlin Heidelberg.

Bartlett, P. L. (1998). The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory 44*(2), 525–536.

Baum, E. B. & D. Haussler (1989). What size net gives valid generalization? *Neural Computing 1*(1), 151–160.

Beni, G. & J. Wang (1989). Swarm intelligence in cellular robotic systems. In *Proceedings of the NATO Advanced Workshop on Robots and Biological Systems*.

Bertsekas, D. P. (1982). *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. New York, NY, USA: Oxford University Press, Inc.

Bonabeau, E., M. Dorigo, & G. Theraulaz (2000, july). Inspiration for optimization from social insect behaviour. *Nature 406*, 39–42.

Bonabeau, E. & C. Meyer (2001, May). Swarm intelligence: A whole new way to think about business. *Harvard Business Review*.

Boser, B. E., I. Guyon, & V. Vapnik (1992). A training algorithm for optimal margin classifiers. In *Computational Learning Theory*, pp. 144–152.

Braga, A. P., T. B. Ludermir, & A. C. P. L. F. de Carvalho (2000). *Redes Neurais Artificiais: Teoria e aplicações*. LTC - Livros Técnicos e Científicos Editora S. A.

Burgess, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining And Knowledge Discovery 2*, 121–167.

Campbell, C. (2001). An introduction to kernel methods. pp. 155–192.

Castro, L. N. D. (2000). The clonal selection algorithm with engineering applications. In *In Proc. GECCO 2000 Workshop on Artificial Immune Systems*, pp. 36–37. Morgan Kaufmann.

Chapelle, O., B. Schölkopf, & A. Zien (Eds.) (2006, September). *Semi-Supervised Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Chen, Y., G. Wang, & S. Dong (2003). Learning with progressive transductive support vector machine. *Pattern Recogn. Lett. 24*(12), 1845–1855.

Cortes, C. & V. Vapnik (1995). Support-vector networks. *Machine Learning 20*(3), 273–297.

Costa, M. A., A. P. Braga, B. R. Menezes, R. A. Teixeira, & G. G. Parma (2003). Training neural networks with a multi-objective sliding mode control algorithm. *Neurocomputing 51*, 467–473.

Cover, T. & P. Hart (1967). Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on 13*(1), 21–27.

Cover, T. M. Capacity problems for linear machines. In L. Kanal (Ed.), *Pattern Recognition*.

Cover, T. M. (1965). Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *Electronic Computers, IEEE Transactions on EC-14*(3), 326–334.

Cristianini, N. & J. Shawe-Taylor (2000, March). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

D. E. Rumelhart, G. H. & R. Williams (1986). Learning internal representations through error propagation. *Parallel Distributed Processing: Experiments in the Microstructure of Cognition 1*.

Darwin, C. (1859). *The Origin of Species By Means of Natural Selection or the Preservation of Favoured Races in the Struggle for Life*. London: Penguin Books.

Dasgupta, D. (1998). *Artificial Immune Systems and Their Applications*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

de Albuquerque Teixeira, R. (2001). *Treinamento de Redes Neurais Artificiais Através de Otimização Multi-Objetivo: Uma Nova Abordagem Para o Equilíbrio Entre a Polarização e a Variância*. Tese de Doutorado, UFMG.

De Jong, K. A. (1994). Genetic algorithms: a 25 year perspective. In *Computational Intelligence: Imitating Life*, pp. 125–134.

Denby, B. & S. Le Hégarat-Mascle (2003, April). Swarm intelligence in optimisation problems. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 502*(2-3), 364–368.

Dorigo, M., V. Maniezzo, & A. Colorni (1996). The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics 26*(1), 29–41.

Duch, W. & J. Korczak (1998). Optimization and global minimization methods suitable for neural networks.

Duda, R. O. & P. E. Hart (1973). *Pattern Classification and Scene Analysis*. John Wiley & Sons Inc.

Eiben, A. (1997). Multi-parent recombination.

Eiben, A. E. & J. E. Smith (2003). *Introduction to Evolutionary Computing*. SpringerVerlag.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics 7*, 179–188.

Gammerman, A., K. Azoury, & V. Vapnik (1998). Learning by transduction. In *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence*, pp. 148–155.

Glover, F. & M. Laguna (1993). Tabu search. In C. Reeves (Ed.), *Modern Heuristic Techniques for Combinatorial Problems*, Oxford, England. Blackwell Scientific Publishing.

Goldberg, D. (1989). *Genetic Algorithms is search, Optimization and Machine Learning*. Addison Wesley.

Goldberg, D. E. & K. Deb (1991). *A Comparative Analysis of Selection Schemes Used in Genetic Algorithms*. Morgan Kaufmann.

Gray, R. M. & L. D. Davisson (1986). *Random Processes: A Mathematical Approach for Engineers*. Englewood Cliffs.

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation* (2nd ed.). Prentice-Hall, Inc.

Herbrich, R. (2001). *Learning Kernel Classifiers: Theory and Algorithms*. Cambridge, MA, USA: MIT Press.

Hinton, G. E. (1989). Connectionist learning procedures. *Artificial Inteligence 40*(1-3), 185–234.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. University of Michigan Press.

Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *Proceedings of the 16th International Conference on Machine Learning*.

Joachims, T. (2003). Transductive learning via spectral graph partitioning. In *Proceeding of The Twentieth International Conference on Machine Learning*.

Kasabov, N. & S. N. Pang (2003). Transductive support vector machines and applications in bioinformatics for promoter recognition. In *Proceedings of International Conference on Neural Networks & Signal Processing*.

Kearns, M. J. & U. V. Vazirani (1994). *An introduction to computational learning theory*. Cambridge, MA, USA: MIT Press.

Kennedy, J. & R. Eberhart (1995). Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, Volume 4, pp. 1942–1948.

Kirkpatrick, S., C. D. Gelatt, & M. P. Vecchi (1983). Optimization by simulated annealing. *Science, Number 4598, 13 May 1983 220, 4598*, 671–680.

Koiran, P. & E. D. Sontag (1997). Neural networks with quadratic vc dimension. *J. Comput. Syst. Sci. 54*(1), 190–198.

Lawrence, S., C. L. Giles, & A. Tsoi (1996). What size neural network gives optimal generalization? convergence properties of backpropagation. Umiacs-tr-96-22 and cs-tr-3617, Institute for Advanced Computer Studies, University of Maryland.

Liu, Y. & K. M. Passino (2000). Swarm intelligence: Literature overview. Technical report, Dept. of Electrical Engineering, Ohio State University.

Mackay, D. J. C. (2002). *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press.

McCulloch, W. T. & W. Pitts (1943). A logical calcus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics 5*, 115–133.

Mendel, J. M. & R. W. McLaren (1994). Reinforcement-learning control and pattern recognition systems. pp. 287–318.

Mercer, J. (1909). Functions of positive and negative types and their connection with the theory of integral equations. *Transactions of the London Philosophical Society (A) 209*, 415–446.

Mika, S. (2002). *Kernel Fisher Discriminants*. Tese de Doutorado, Technischen Universität Berlin.

Mika, S., G. Ratsch, J. Weston, B. Scholkopf, & K. R. Mullers (1999). Fisher discriminant analysis with kernels. In *Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop*, pp. 41–48.

Minsky, M. L. & S. A. Papert (1969, December). *Perceptrons*. The MIT Press.

Mitchell, M. (1996). *An introduction to genetic algorithms*. Cambridge, Mass., USA: MIT Press.

Mitchell, T. M. (1999). The role of unlabeled data in supervised learning. In *In Proceedings of the Sixth International Colloquium on Cognitive Science*.

Muller, K. R., S. Mika, G. Ratsch, K. Tsuda, & B. Schölkopf (2001). An introduction to kernel-based learning algorithms. *Neural Networks, IEEE Transactions on 12*(2), 181–201.

Oei, C., D. Goldberg, & S. Chang (1991). Tournament selection, niching, and the preservation of diversity. Technical report.

Passerini, A. (2004). *Kernel Methods, Multiclass Classification and Applications to Computational Molecular Biology*. Tese de Doutorado, Università Degli Studi di Firenze.

Pontil, M. & A. Verri (1998). Support vector machines for 3d object recognition. *IEEE Trans. Pattern Anal. Mach. Intell. 20*(6), 637–646.

Poole, D., A. Mackworth, & R. Goebel (1997). *Computational intelligence: a logical approach*. Oxford, UK: Oxford University Press.

Reed, R. & I. Robert J. Marks (1998). Neurosmithing: improving neural network learning. pp. 639–644.

Rumelhart, D. E., G. E. Hinton, & R. J. Williams (1986). Learning representations by back-propagating errors. *Nature 323*, 533–536.

Schölkopf, B. & A. J. Smola (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press.

Silva, M. M., T. T. Maia, & A. P. Braga (2005). An evolutionary approach to transduction in support vector machines. In *HIS '05: Proceedings of the Fifth International Conference on Hybrid Intelligent Systems*, Washington, DC, USA, pp. 329–334. IEEE Computer Society.

Smola, A. J. & B. Scholkopf (1998). A tutorial on support vector regression. Technical Report NC-TR-98-030, Royal Holloway College, University of London, UK.

Solomonoff, R. J. (1964). A formal theory of inductive inference. *Information and Control 7*(2), 224–254.

Sutton, R. & A. Barto (1998). *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

Takahashi, R. H. C., P. L. D. Peres, & P. A. V. Ferreira (1997). H2/h-infinity multiobjective pid design. *IEEE Control Systems Magazine 17*(5), 37–47.

Teixeira, R. A., A. P. Braga, R. H. C. Takahashi, & R. R. Saldanha (2000). Improving generalization of mlps with multi-objective optimization. *Neurocomputing 35*, 189–194.

Teixeira, R. A., A. P. Braga, R. H. C. Takahashi, & R. R. Saldanha (2001). Recent advances in the mobj algorithm for training artificial neural networks. *International Journal of Neural Systems 11*(3), 265–270.

Vapnik, V. (1982). *Estimation of dependencies based on empirical data*. Springer.

Vapnik, V. (1998). *Statistical Learning Theory*. Wiley.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. New York: Springer.

Vapnik, V. N. (1999). An overview of statistical learning theory. *IEEE Transactions on Neural Networks 10*, 988–999.

Vapnik, V. N. & A. Y. Chervonenkis (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Appl. 16*, 264–280.

Vapnik, V. N., E. Levin, & Y. Lecun (1994). Measuring the VC-dimension of a learning-machine. *Neural Computation 6*, 851–876.

Vidyasagar, M. (1997). *A Theory of Learning and Generalization: With Applications to Neural Networks and Control Systems*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Weishui, W. & X. Chen (1996). Convergence theorem of genetic algorithm. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, Volume 3.

Wen Zhang, Y. L. & M. Clerc (2003). An adaptive pso algorithm for reactive power optimization. In *Proceedings of the Sixth International Conference on Advances in Power System Control, Operation and Management (ASD-COM 2003)*, Volume 1, pp. 302–307.

White, T. (1997, Spring). Swarm intelligence and problem solving in telecommunications. *Canadian Artificial Intelligence Magazine* (41), 14–16.

Wolpert, D. H. & W. G. Macready (1997, April). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation 1*(1), 67–82.

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control 8*(3), 338–353.