TESE DE DOUTORADO Nº 018

TREINAMENTO DE REDES NEURAIS ARTIFICIAIS ATRAVÉS DE OTIMIZAÇÃO MULTI-OBJETIVO: UMA NOVA ABORDAGEM PARA O EQUILÍBRIO ENTRE A POLARIZAÇÃO E A VARIÂNCIA

Roselito de Albuquerque Teixeira

DATA DA DEFESA: 10.08.2001

Universidade Federal de Minas Gerais
Programa de Pós-Graduação em Engenharia Elétrica
Centro de Pesquisa e Desenvolvimento em Engenharia Elétrica

Treinamento de Redes Neurais Artificiais Através de Otimização Multi-objetivo: Uma Nova Abordagem para o Equilíbrio entre a Polarização e a Variância

Por
Roselito de Albuquerque Teixeira

Tese de Doutorado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais, como requisito parcial a obtenção de título de Doutor em Engenharia Elétrica

Orientador : **Antônio de Pádua Braga**
Co-Orientadores : **Ricardo H. C. Takahashi**
Rodney R. Saldanha

10 de agosto de 2001

"Treinamento de Redes Neurais Artificiais através de Otimização Multi-Objetivo: Uma nova Abordagem para o Equilíbrio entre a Polarização e a Variância"

Roselito de Albuquerque Teixeira

Tese de Doutorado submetida à banca examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais, como parte dos requisitos necessários à obtenção do grau de Doutor em Engenharia Elétrica.

Aprovada em 10 de agosto de 2001.

Por:

Antônio de Pádua Braga – Ph.D.
Prof. DELT/UFMG – orientador

Ricardo Hiroshi Caldeira Takahashi – Dr.
Prof. DEE/UFMG – co-orientador

Rodney Rezende Saldanha – Dr.
Prof. DEE/UFMG – co-orientador

Geraldo Robson Mateus – Dr.
Prof. DCC/ICEx/UFMG

Andre Carlos Ponce de Leon F. de Carvalho – Ph.D.
Prof. ICMC-USP/São Carlos

Takashi Yoneyama – Dr.
Prof. ITA – São José dos Campos

Agradecimentos

Temo ser injusto ao escrever esta seção, pois são inúmeras as pessoas que participaram de alguma forma deste trabalho e todas elas deveriam ser mencionadas. No entanto, algumas têm um peso muito grande neste processo, como a minha esposa Elizanete e meus filhos Mateus e Isac. A verdade é que tivemos momentos difíceis e quero que ela saiba o quanto sou feliz no meu casamento e quanto gosto dela e de meus filhos. Esta mulher se mostrou grande e forte, dispondo-se a ficar longe de suas raízes cuidando da nossa família.

Aos meus pais, pela forte presença, pela fé e até mesmo, pelo apoio financeiro nos momentos difíceis.

Aos meus amigos e colegas do LITC Marcelo Barros, Marcelo Costa, Eber, Jeremias, Regis e Parma que nunca pouparam esforços em me ajudar e especialmente ao meu amigo Roberto Mansur que fez o desenvolvimento da Interface para demonstração do método multi-objetivo proposto nesta Tese.

Aos Professores Antônio de Pádua Braga, Ricardo Hiroshi C. Takahashi e Rodney Rezende Saldanha pelo apoio técnico e moral, pela paciência, compreensão e tranquilidade com que conduziram a orientação deste trabalho.

Aos meus amigos Ramon e Marcelo, que sempre estiveram no mesmo barco e ainda assim sempre dispostos a dar uma palavra amiga.

Aos colegas de trabalho do UNILESTE que me apoiaram e incentivaram, especialmente a Professora Maria Cândida Belo Corrêa.

Ao UNILESTE - Centro Universitário do Leste Mineiro, Coronel Fabriciano, pelo apoio e por possibilitar minha dedicação exclusiva a este trabalho.

Ao CNPq pelo apoio e suporte financeiro.

À minha esposa Elizanete
Aos meus filhos Mateus e Isac

ALGO A SE PENSAR

Nosso maior medo não é o de sermos inadequados.
Nosso maior medo é o de sermos poderosos além da medida.
É nossa luz, nossa escuridão, o que mais nos apavora.
Perguntamos a nós mesmos:
Quem sou eu para ser brilhante, esplêndido, talentoso e fabuloso?
Na verdade, por que você não seria?
Você é um filho de Deus. Bancar o pequeno não serve ao mundo.
Nada nos esclarece no sentido de nos diminuirmos,
para que outras pessoas não se sintam inseguras em torno de nós.
Nascemos para tornar manifesta a glória de Deus que está dentro de nós.
Ela não está em alguns de nós; está em todos nós.
E quando deixamos nossa própria luz brilhar, inconscientemente
damos a outras pessoas permissão para fazer o mesmo.
Quando nos libertamos de nosso próprio medo,
nossa presença automaticamente liberta outros.

Nelson Mandela, 1994.

Resumo

Neste trabalho é desenvolvido um novo método para treinamento de Redes Neurais Artificiais (RNAs) do tipo Perceptron Multi-camadas (*Multilayer Perceptron*-MLP) utilizando-se técnicas de otimização multi-objetivo para encontrar soluções com alta capacidade de generalização.

Na abordagem proposta, além de ser utilizado o erro quadrático como função de custo, utiliza-se também a função norma do vetor de pesos como um segundo objetivo. Estas duas funções são minimizadas e como consequência deste processo, obtém-se um conjunto de soluções chamadas Pareto-ótimas.

Através de um decisor, estas soluções são avaliadas e apenas uma delas é escolhida como solução final, a qual equilibra os efeitos da polarização e da variância resultando em alta capacidade de generalização dada uma determinada realização do conjunto de treinamento.

O método multi-objetivo proposto controla a flexibilidade do modelo independentemente da quantidade de pesos existente na rede, a partir de uma estrutura mínima. Além disso, a utilização de parâmetros de treinamento apesar de ser necessária, influencia pouco a solução final, o que faz com que a escolha destes parâmetros seja uma tarefa simples.

Além do algoritmo multi-objetivo proposto e do algoritmo *Backpropagation*, métodos conhecidos os quais também visam ao aumento da capacidade de generalização como *10-Fold Cross-Validation*, *Early Stopping*, *Optimal Brain Damage*, *Weight Decay* e *Support Vector Machines* são abordados. Uma análise qualitativa e quantitativa dos algoritmos é feita, através da qual pode-se observar a superioridade do método proposto, sendo o mesmo capaz de gerar soluções com alta capacidade de generalização de forma simples e eficiente.

Abstract

This work presents a new learning scheme for improving generalization of Multilayer Perceptrons (MLPs). The proposed Multi-objective algorithm (MOBJ) approach minimizes both the sum of squared error and the norm of network weight vectors to obtain the Pareto-optimal solutions.

Since the Pareto-optimal solutions are not unique, we need a decision phase in order to choose the best one as a final solution by using a validation set. The final solution is expected to balance network variance and bias and, as a result, generates a solution with high generalization capacity, avoiding over and underfitting.

The proposed multi-objective method controls model flexibility independently of the number of network weights, although a minimal number of weights is needed. Also, the training parameters produce minor effects on the final solution and, consequently, tuning the best set of training parameters is an easy task.

The classification and regression MOBJ solutions are compared with Weight decay, Optimal Brain Damage, Early Stopping, 10-Fold Cross-Validation, Support Vector Machines and Backpropagation. It is concluded that the proposed method is able to generate high generalization solutions and its operation is simple and efficient.

Sumário

Abreviações

As principais abreviações usadas nesta Tese são listadas a seguir. Algumas siglas consagradas na literatura internacional foram mantidas em Inglês.

RNA	Redes Neurais Artificiais
SVM	Máquina de Vetores de Suporte (*Support Vector Machine*)
MOBJ	Algoritmo de treinamento Multi-objetivo
BP	Algoritmo *Backpropagation*
WD	Algoritmo de *pruning Weight Decay*
OBD	Algoritmo de *pruning Optimal Brain Damage*
ES	Algoritmo de treinamento com Interrupção Precoce (*Early Stopping*)
CV	Algoritmo de treinamento com Validação Cruzada (*Cross-validation*)
MLP	Redes Neurais Multi-Camadas (*Multilayer Perceptron*)
MSE	Média do Somatório dos Erros Quadráticos (*Mean Squared Errors*)

Lista de Símbolos

Nesta Tese, vetores são indicados por letras latinas minúsculas em negrito. Escalares são representados por letras minúsculas gregas ou latinas em itálico com ou sem sub-índice. A seguir são listados os principais símbolos usados neste trabalho. Símbolos específicos serão definidos in loco.

$\mathbf{x}$	Vetor de entrada;
$\mathbf{d}$	Vetor de saída desejado;
σ^2	Variância do ruído presente nos dados;
$purelin(\cdot)$	Função de ativação linear ($purelin(x) = x$);
$tanh(\cdot)$	Função de ativação tangente hiperbólica;
f_{ah}	Função de ativação da camada escondida;
f_{ao}	Função de ativação da camada de saída;
N_N	Número de neurônios na camada escondida;
$\delta\epsilon$	Variação da norma entre soluções para problema Pϵ;
$\mathbf{y}$	Vetor de saída da RNA;
$\mathbf{e}$	Vetor de erro;
e_V	Média do somatório dos erros quadráticos para padrões de validação;
e_T	Média do somatório dos erros quadráticos para padrões de treinamento;
e_E	Média do somatório dos erros quadráticos para padrões de teste;
ϕ_i^*	Valor ótimo para o i-ésimo objetivo;
ϕ_i	Valor do i-ésimo objetivo em um ponto qualquer;
$\mathbf{f}_i^*$	Vetor formado pelo ótimo de cada objetivo individual i e os valores Correspondentes aos demais objetivos;
$\mathbf{u}^*$	Solução utópica do problema multi-objetivo;
$\mathbf{w}$	Vetor de pesos de uma RNA;
$E[\cdot]$	Esperança matemática;
N_T	Número de padrões do conjunto de treinamento;
N_V	Número de padrões do conjunto de validação;
N_E	Número de padrões do conjunto de teste;
N	Número total de padrões de um conjunto de dados;

ξ	Ruído gaussiano com média zero e variância σ^2;
$f_a(\cdot)$	Função de ativação de um neurônio;
η	Variável auxiliar;
u_h	Número de neurônios na camada escondida;
u_i	Número de entradas da RNA;
u_o	Número de saídas da RNA;
n_w	Número de pesos e termos de polarização de uma RNA;
p	Número de variáveis do problema de otimização original;
q	Número de restrições do problema de otimização original;
$\mathcal{W}$	Conjunto dos vetores soluções factíveis;
$\mathcal{W}^*$	Conjunto Pareto-ótimo;
ζ	Número de soluções Pareto-ótimas;
$\mathbf{w}^*$	Vetor de pesos ótimo;
$N_{\mathbf{w}^*}$	Número da melhor solução escolhida;
$\mathcal{E}[f(\mathbf{x};\mathbf{w})]$	Funcional distância entre função geradora $f_g(\mathbf{x})$ e função estimada $f(\mathbf{x};\mathbf{w})$;
$f_g(\mathbf{x})$	Função geradora;
$f(\mathbf{x};\mathbf{w})$	Função estimada;
$\mathcal{T}$	Conjunto de dados genérico;
$\mathcal{T}_T$	Conjunto de treinamento;
$\mathcal{T}_V$	Conjunto de validação;
$\mathcal{T}_E$	Conjunto de teste;
$\mathcal{N}_0$	Norma ótima considerando o funcional $\mathcal{E}[f(\mathbf{x};\mathbf{w})]$;
$\mathcal{N}^*$	Norma ótima considerando o conjunto de validação $\mathcal{T}_V$;
e_V^*	Média do somatório dos erros quadráticos ótimo considerando o conjunto de validação;
^{T}	Transposição de vetores ou matrizes;

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O processo de obtenção de uma rede devidamente treinada quase sempre não é uma tarefa simples e geralmente requer um grande esforço por parte do projetista na determinação de parâmetros que irão fazer com que uma Rede Neural Artificial (RNA) aprenda adequadamente uma dada tarefa.

Uma rede devidamente treinada não só deverá responder adequadamente aos padrões utilizados no processo de treinamento mas também aos demais que porventura sejam mostrados a ela. A esta propriedade de resposta coerente a padrões desconhecidos dá-se o nome de capacidade de generalização de uma rede. Uma RNA que seja capaz de responder de maneira satisfatória aos padrões desconhecidos possui capacidade de generalização elevada e portanto está devidamente treinada.

Vários fatores podem influenciar na obtenção de redes com alta capacidade de generalização como por exemplo a seleção dos parâmetros ideais para o processo de treinamento e ainda a escolha da topologia adequada. A escolha da topologia de uma rede neural do tipo *Multilayer Perceptron* (MLP) [Rumelhart et al., 1986a] as quais são alvo deste trabalho, é tarefa do projetista e consiste basicamente na determinação do número de camadas da rede e do número de neurônios em cada camada. Neste trabalho, redes com topologias $u_i \times u_h \times u_o$ serão utilizadas. Portanto, uma rede com tal topologia possui entre pesos e termos de polarização um total n_w de parâmetros livres dado por $n_w = [(u_i + 1)u_h + (u_h + 1)u_o]$ e o treinamento de uma rede com n_w parâmetros se dá no espaço n_w-dimensional. O processo de aprendizagem é, consequentemente, considerado um processo difícil porque a procura da solução adequada ocorre em um universo de soluções possíveis de grande dimensão e a escolha entre as representações possíveis deve ser feita baseada em uma determinada realização do conjunto de treinamento [Hinton, 1989].

Diferentes realizações para um conjunto de treinamento de tamanho fixo podem fazer com que diferentes soluções sejam tomadas no universo possível de soluções, dada uma topologia de rede, principalmente quando os dados são ruidosos. Porém, em um processo de treinamento sempre se busca modelar a função geradora dos dados com base em uma

das possíveis realizações para o conjunto de treinamento. À variabilidade de soluções, dados conjuntos de treinamento diferentes para uma mesma tarefa, dá-se o nome de variância [Geman et al., 1992]. A variância das soluções é algo que deve ser minimizado para se garantir boa capacidade de generalização das redes, sendo tão maior quanto maior for o número de parâmetros n_w da rede pois a busca ocorre no espaço n_w-dimensional. Geralmente estas soluções apresentam super-ajuste aos dados de treinamento (*overfitting*). Quanto maior a dimensão deste espaço, maiores são as possibilibidades de representação de funções no mesmo.

Para resolver o problema da variância excessiva, uma saída seria fazer a redução do espaço de busca das soluções, reduzindo o número n_w de parâmetros da rede e com isto simplificando sua topologia. Com a redução da dimensão do espaço de busca, pode-se incorrer em outro problema que também prejudica a capacidade de generalização que é a polarização [Geman et al., 1992] das soluções. A polarização das soluções se dá quando mesmo para diferentes realizações para o conjunto de treinamento, a solução resultante do processo de treinamento no espaço de dimensão reduzida é praticamente a mesma. Se for tomada a média destas soluções ainda assim esta não aproximaria a função geradora. As soluções polarizadas são caracterizadas pelo sub-ajuste aos padrões de treinamento (*underfitting*). Logo, a polarização das soluções é também algo que deve ser minimizada para se garantir boa capacidade de generalização.

Pode parecer contraditório o fato de que deve-se reduzir a variância para se alcançar boa capacidade de generalização mas isto implica em aumento da polarização que também prejudica tal capacidade e portanto também deve ser minimizada. Logo, pode-se notar que deve haver um ponto de equilíbrio entre estes efeitos, que é conhecido na literatura como "o dilema entre polarização e variância" [Geman et al., 1992].

A determinação da topologia adequada para uma dada tarefa de aprendizagem a priori não é uma tarefa trivial. Muitos métodos visam ao controle de complexidade para obter sistemas com alta capacidade de generalização [Hinton, 1989, Cun et al., 1990, Cortes and Vapnik, 1995]. Porém, todos os métodos relatados na literatura necessitam ainda do conhecimento do projetista para obter as melhores soluções de forma que o processo de treinamento ainda é um ponto importante em RNAs. Isto motiva a busca de novos métodos que visam aumentar a capacidade de generalização das mesmas e diminuir a necessidade da intervenção do projetista neste processo.

Atualmente há uma grande variedade de algoritmos para o treinamento de RNAs. Alguns trabalham, simplesmente, no sentido de minimizar o somatório dos erros quadráticos, como o popular *Backpropagation* [Rumelhart and McClelland, 1986], bem como suas variações, tais como *Quickprop* [Fahlman, 1988], *Rprop* [Riedmiller and Braun, 1993a], *Levenberg Marquardt* [Hagan and Menhaj, 1994] e treinamento com modos deslizantes [Parma et al., 1998].

Outros algoritmos, além de minimizarem o somatório dos erros quadráticos (MSE),

atuam também no sentido de maximizar a capacidade de generalização, como os algoritmos de *Pruning* [Hinton, 1989, Cun et al., 1990, Mozer and Smolensky, 1989, Karnin, 1990], que iniciam o treinamento com uma topologia de rede superdimensionada e eliminam pesos e nodos até que se consiga uma boa capacidade de generalização. O algoritmo *Weight Decay* [Hinton, 1989] em especial é abordado neste trabalho.

As técnicas de *pruning* fazem ajuste da complexidade das redes alterando a estrutura das mesmas. Porém, algumas técnicas conseguem alcançar soluções adequadas sem que haja alteração da topologia das RNAs, como por exemplo, o método da interrupção precoce do treinamento (*Early Stopping*) [Weigend et al., 1990] e o método da validação cruzada (*Cross-Validation*) [Stone, 1974, Stone, 1978]. Encontram-se na literatura também as *Support Vector Machines* [Boser et al., 1992, Cortes and Vapnik, 1995, Burges, 1998, Osuna et al., 1997a, Osuna et al., 1997b], as quais são capazes de retornar soluções de alta capacidade de generalização baseadas no princípio da aprendizagem estatística.

Apesar de os métodos acima serem capazes de retornar soluções com alta capacidade de generalização, alguns são mais sensíveis à quantidade de padrões utilizados no processo de treinamento, como por exemplo o *Early Stopping* e o *Cross-Validation* e outros bastante sensíveis aos parâmetros de treinamento, como o *Weight Decay* e as *Support Vector Machines*.

Algoritmos de treinamento de redes neurais que utilizam apenas o conjunto de treinamento e o erro relacionado a ele, para ajuste dos pesos não são sempre capazes de encontrar uma solução com alta capacidade de generalização. Estes algoritmos são muito sensíveis ao número de parâmetros livres n_w, aos valores de inicialização dos mesmos e também ao instante de parada do treinamento como é o caso dos algoritmos *Backpropagation*, *Quickprop*, *Rprop*, *Levenberg Marquardt* e treinamento por modos deslizantes. Dependendo destes parâmetros, as redes treinadas podem ficar sub ou super-ajustadas aos dados de treinamento, prejudicando a capacidade de generalização das mesmas.

As discussões anteriores deixam claro a necessidade de ajuste da complexidade da rede em relação à complexidade requerida para que um dado problema de aprendizagem seja resolvido com a capacidade de generalização maximizada. Pesos cujos valores podem variar em grandes faixas resultam em alta variância e pesos cujos valores estão restritos em faixas menores resultam em alta polarização [Geman et al., 1992] de forma que a magnitude dos pesos é mais importante que a quantidade deles presentes em uma rede [Bartlett, 1997] para uma boa capacidade de generalização. A magnitude dos elementos do vetor de pesos de uma rede é diretamente refletida na função norma deste vetor que será tão maior quanto maiores forem as magnitudes dos pesos e vice-versa.

Soluções que se apresentam super-ajustadas aos dados são soluções com complexidade além da necessária e portanto estas soluções possuem normas mais elevadas que a necessária. Por outro lado, soluções de normas baixas são aquelas que se apresentam sub-ajustadas ao conjunto de treinamento, ou seja, possuem baixa complexidade.

Neste sentido, verificando que a norma do vetor de pesos é uma medida de complexidade da função mapeada pela rede, surge a idéia da utilização desta medida como função custo a ser minimizada em conjunto com o erro em relação aos padrões de treinamento. Portanto, pode-se formalizar um problema de otimização multi-objetivo onde são considerados dois objetivos a serem minimizados.

Através da otimização multi-objetivo e da utilização das funções norma do vetor de pesos e erro de treinamento como funções de custo, podem-se gerar soluções de complexidades diferentes entre os extremos de alta complexidade e de baixa complexidade, de forma que uma destas soluções tenha a complexidade adequada e portanto com as características desejáveis. Ao processo de geração de soluções e a escolha da solução final dá-se o nome de método multi-objetivo para treinamento de RNAs ou simplesmente método MOBJ.

Diferentemente dos métodos acima mencionados, o método multi-objetivo proposto trabalha com a imposição de limites à norma para controle de complexidade, restringindo o universo de funções possíveis na dimensão n_w, sendo que para uma solução de norma $\mathcal{N}$, o erro para os padrões de treinamento será o menor possível.

Neste trabalho, soluções com boa capacidade de generalização são encontradas a partir de uma RNA superdimensionada. O modelo resultante tem o mesmo número de parâmetros livres que o modelo inicial, porém, o sistema se comporta como se tivesse menos parâmetros livres, porque muitos destes se anulam ou se aproximam de zero. A implementação da abordagem proposta é feita utilizando-se técnicas de otimização multi-objetivo como o problema ε-restrito [Chankong and Haimes, 1983] e uma variante deste problema chamada de técnica de relaxação [Takahashi et al., 1997].

Para validar o algoritmo multi-objetivo proposto, problemas de classificação e de regressão são abordados e os resultados são comparados com os algoritmos *Backpropagation* (BP), *Early Stopping*, *Cross-Validation*, *Optimal brain damage*, *Weight Decay* e SVMs.

A Figura 1.1 ilustra soluções hipotéticas geradas pelo método MOBJ. Este conjunto de soluções é chamado de conjunto Pareto-ótimo. É característica das soluções pertencentes a este conjunto, o fato de que não se pode melhorar um dos objetivos sem que outro seja degradado. Em outras palavras, não se pode reduzir o erro de treinamento sem que haja um incremento no valor da norma ou de complexidade. Por outro lado, não se pode diminuir a norma, fazendo com que a complexidade efetiva do sistema diminua, sem que haja um incremento no erro de treinamento.

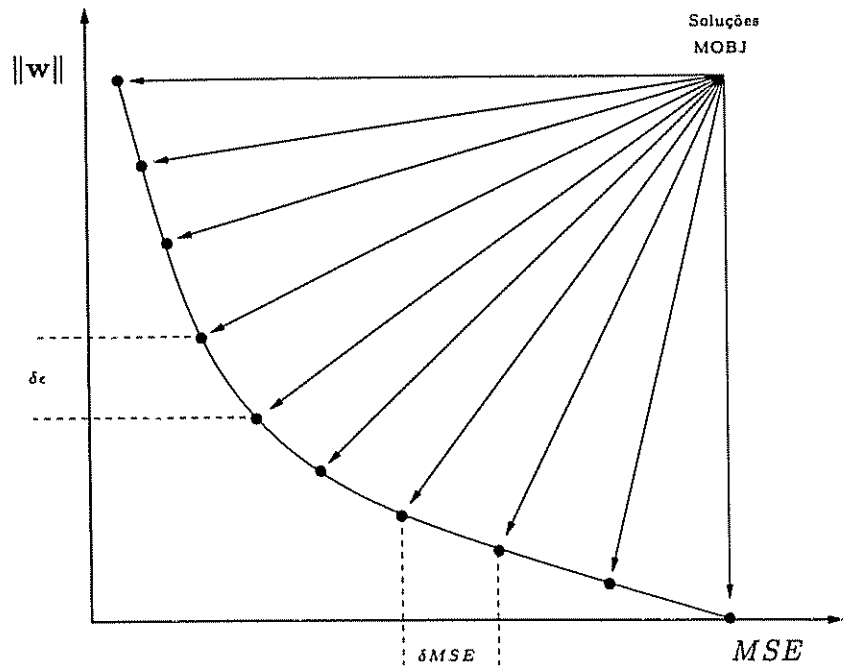


Figura 1.1: Detalhamento do conjunto Pareto-ótimo

É importante observar o compromisso entre o erro de treinamento e a norma do vetor de pesos. Note-se que, para as soluções pertencentes a este conjunto, existe uma diferença de erro δMSE e uma diferença de norma $\delta\epsilon$ entre elas e se uma solução possui erro para os padrões de treinamento menor, consequentemente a norma dos pesos para esta solução é maior. O conjunto Pareto-ótimo ilustrado na Figura 1.1 deixa explícita a relação entre erro e norma do vetor de pesos e entre as soluções pertencentes a este conjunto.

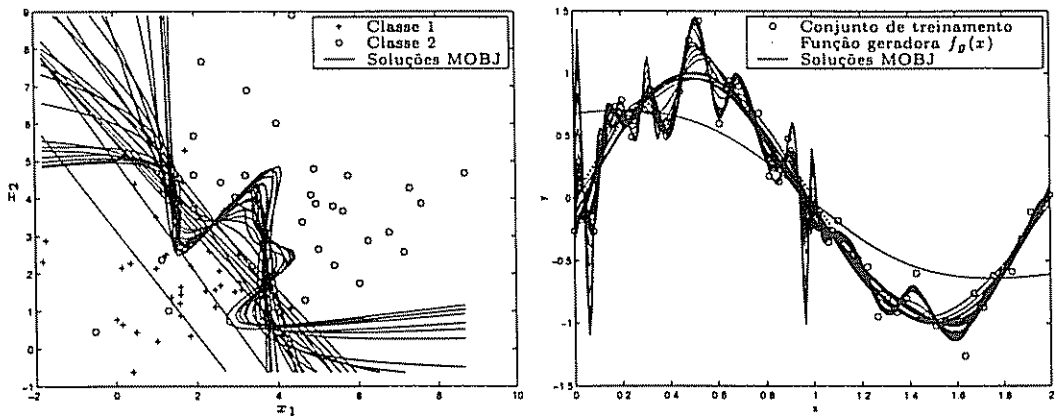
O método MOBJ é capaz de gerar soluções de forma eficiente e possui grande robustez em relação aos parâmetros de treinamento. Como este método é capaz de encontrar soluções que estão entre os extremos citados anteriormente, é preciso que um decisor atue no sentido de escolher uma delas. É tarefa do decisor avaliar as soluções geradas e escolher a melhor segundo algum critério. Neste trabalho é implementado um eficiente decisor baseado em dados de validação, que faz sua busca no conjunto Pareto-ótimo e não no espaço n_w-dimensional.

1.1 Contribuições

Neste trabalho é desenvolvido um método de treinamento de RNAs capaz de gerar soluções com capacidade de generalização elevada utilizando técnicas de otimização multi-objetivo. Na abordagem proposta, é gerado um conjunto de redes (soluções) sendo que uma delas é escolhida como solução final com base em sua capacidade de generalização. É importante ressaltar que, apesar de o algoritmo MOBJ utilizar uma estrutura de rede

que deve ser superdimensionada e informada a priori, procurou-se minimizar o número de parâmetros de treinamento a serem informados pelo usuário. Logo, a partir de uma RNA superdimensionada, obtém-se ζ soluções sem a necessidade da intervenção do projetista.

As Figuras 1.2(a) e 1.2(b) ilustram as soluções Pareto-ótimas obtidas para um problema de classificação e outro de regressão respectivamente. Em cada problema, o decisor deve escolher uma das soluções MOBJ como solução final, sendo que, é escolhida aquela que apresentar menor erro em relação a um determinado conjunto de validação. Note-se, através das ilustrações, que em ambos os casos existem desde soluções sub-ajustadas até soluções super-ajustadas. A solução a qual possui o ajuste adequado pertence ao conjunto de soluções MOBJ e deve ser escolhida pelo decisor proposto.



(a) Soluções MOBJ - Problema de classificação

(b) Soluções MOBJ - Problema de regressão

Figura 1.2: Geração de soluções MOBJ em problemas de classificação e regressão.

O método multi-objetivo para treinamento de RNAs proposto é novo e abre uma linha de pesquisa para futuros trabalhos na área de generalização em RNAs. Neste trabalho é desenvolvida uma teoria que aborda as propriedades e características do método em questão sendo que a eficiência do decisor implementado é provada. Os resultados obtidos através do método multi-objetivo são em muitos casos superiores aos demais métodos testados e nos casos onde não são superiores, são equivalentes, no que diz respeito à capacidade de generalização das soluções obtidas. No que diz respeito à necessidade de conhecimento por parte do projetista para se obter soluções com capacidade de generalização elevada, o método proposto se mostra superior aos demais.

1.2 Organização do Texto

O presente texto está organizado em sete capítulos da seguinte forma:

No Capítulo 2 são abordadas questões importantes relativas ao processo de treinamento de redes neurais como o dilema entre polarização e variância, capacidade de generalização das redes neurais, *overfitting* e *underfitting*. É feita também uma revisão bibliográfica de alguns dos algoritmos conhecidos na literatura que buscam encontrar soluções de alta capacidade de generalização como o *Weight Decay*, *Optimal Brain Damage*, *Early Stopping*, *Cross-Validation* e as *Support vector machines*.

No Capítulo 3 são abordados conceitos relativos à otimização multi-objetivo necessários à compreensão do algoritmo proposto. As técnicas de otimização multi-objetivo denominadas de método de relaxação e método ε-restrito são também abordadas.

O Capítulo 4 trata da geração das soluções MOBJ para problemas de aprendizagem de redes do tipo MLP. Neste capítulo é feito o detalhamento da implementação do algoritmo multi-objetivo proposto através da técnica de relaxação e do método ε-restrito. Ainda neste capítulo são feitas discussões sobre a eficiência computacional do algoritmo proposto, sobre as possíveis formas do conjunto Pareto-ótimo e ainda sobre o comportamento das soluções no espaço dos objetivos.

No Capítulo 5 é abordado o algoritmo "decisor". Também é feita a demonstração do teorema que comprova a eficiência do processo de escolha da melhor solução. Vários fatores que podem influenciar neste processo como intensidade de ruído nos dados, quantidade de padrões, resolução do conjunto Pareto-ótimo e a topologia da rede são discutidos em detalhe.

No Capítulo 6 o algoritmo MOBJ é utilizado para resolver alguns problemas de classificação e de regressão com diferentes complexidades. Os demais algoritmos abordados no Capítulo 2 são também utilizados para resolver os mesmos problemas para fins de comparação com o algoritmo proposto. Os problemas de classificação são num total de 4, sendo que os três primeiros são simulações e o quarto é um problema real de análise de risco de crédito. Os problemas de regressão são dois e tratam-se de problemas simulados por computador.

No Capítulo 7 são feitas as discussões sobre as características do algoritmo multi-objetivo proposto e também são discutidas algumas características dos demais métodos abordados que podem fazer com que os mesmos não encontrem soluções com capacidades de generalização elevadas. Conclusões sobre o método proposto são feitas e são apresentadas as propostas de continuidade deste trabalho.

1.3 Conclusões do Capítulo

Neste capítulo é dada uma visão geral sobre os principais métodos de treinamento encontrados na literatura bem como o problema inerente ao processo de treinamento das RNAs que é o equilíbrio entre a polarização e a variância das mesmas. Tendo em vista este equilíbrio, é proposto um método utilizando técnicas de otimização multi-objetivo o qual visa à obtenção de RNAs com elevada capacidade de generalização.

Capítulo 2

O Problema de Generalização em RNAs

Existem autores que afirmam que as redes neurais artificiais (RNAs) são capazes de generalizar [Caudill and Butler, 1990]. Esta afirmativa porém, nem sempre é verdadeira, uma vez que esta propriedade não é inerente às RNAs. Boa capacidade de generalização é conseguida se alguns fatores são levados em consideração, como por exemplo, o número de padrões utilizados no processo de treinamento e a capacidade do modelo de se ajustar a estes dados. Considerando que o tamanho do conjunto de treinamento é adequado para uma dada tarefa de aprendizagem, ou seja, é estatisticamente representativo, a capacidade de generalização pode ser conseguida se o modelo tiver complexidade ótima. Modelos muito complexos (os quais se ajustam bem aos dados) e modelos pouco complexos (os quais se ajustam mal aos dados) frequentemente apresentam baixa capacidade de generalização. Entretanto, deve haver um equilíbrio entre a complexidade do modelo e a complexidade inerente à tarefa. Mas, a determinação a priori da complexidade do modelo não é uma tarefa trivial.

Considerando um ajuste polinomial, a complexidade do modelo é dada pela ordem do polinômio e em RNAs, a complexidade do modelo pode ser dada pelo número de parâmetros livres da rede, sendo que os parâmetros livres são os pesos e os termos de polarização. Existem outras formas para a caracterização da complexidade de um modelo e uma medida mais adequada para problemas de aprendizagem é dada pelo *VC-dimension*[Vapnik, 1995, Vapnik et al., 1994], porém, em muitos casos, o cálculo desta quantidade é inviável. Também, medidas clássicas de complexidade podem ser feitas em função do número de parâmetros livres, como citado anteriormente, pelo número de derivadas contínuas [Girosi et al., 1995] e ainda pela magnitude da transformada de Fourier [Barron, 1993]. Estas medidas clássicas não são tão adequadas quanto o *VC-dimension* quando a dimensionalidade do modelo é elevada.

Contudo, a complexidade efetiva pode ser controlada através da imposição de restrições (regularização) e, neste caso, o número de parâmetros não é importante, podendo-se trabalhar com modelos complexos. Intuitivamente, redes neurais com mesmos números

de parâmetros livres podem se comportar como sistemas com diferentes complexidades. Considere um conjunto $\mathcal{W}_1$ constituído por vetores $\mathbf{w}$ tais que $\|\mathbf{w}\| \leq \mathcal{N}_1$ para todo $\mathbf{w}$ e uma função de penalidade $f(\cdot) = \|\cdot\|$. Se um outro conjunto $\mathcal{W}_2$ constituído por vetores $\mathbf{w}$ tais que $\|\mathbf{w}\| \leq \mathcal{N}_2$ para todo $\mathbf{w}$ e $\mathcal{N}_1 < \mathcal{N}_2$, o conjunto $\mathcal{F}_2$ constituído pelas funções $f(\mathbf{x};\mathbf{w})$ tais que $\mathbf{w} \in \mathcal{W}_2$ contém o conjunto $\mathcal{F}_1$, constituído pelas funções $f(\mathbf{x};\mathbf{w})$ tais que $\mathbf{w} \in \mathcal{W}_1$. Isto ocorre devido à imposição de limites para as magnitudes dos pesos que para o conjunto $\mathcal{W}_1$ é menor que para o conjunto $\mathcal{W}_2$ e portanto, $\mathcal{F}_2 \supset \mathcal{F}_1$. A Figura 2.1 ilustra os conjuntos de funções S_1, S_2 e S_3 para $\mathcal{N}_1 < \mathcal{N}_2 < \mathcal{N}_3$.

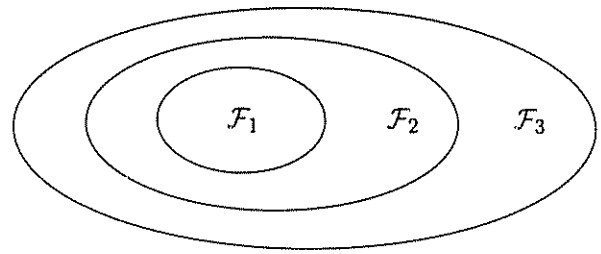


Figura 2.1: Conjuntos de soluções com diferentes complexidades efetivas

Tendo os conceitos anteriores em mente, é feita uma análise sobre a qualidade das soluções geradas pelas RNAs. Através desta análise, pode-se discutir o que afeta a capacidade de generalização das redes.

Matematicamente, o treinamento de uma MLP consiste em minimizar a função de custo dada pela Equação (2.1) [Bengio, 1996],

$$e_V(\mathbf{w}) = \int_{\mathcal{X},\mathcal{D}} e(f(\mathbf{x};\mathbf{w}),\mathbf{d})p(\mathcal{X},\mathcal{D})d\mathbf{x}d\mathbf{d} \tag{2.1}$$

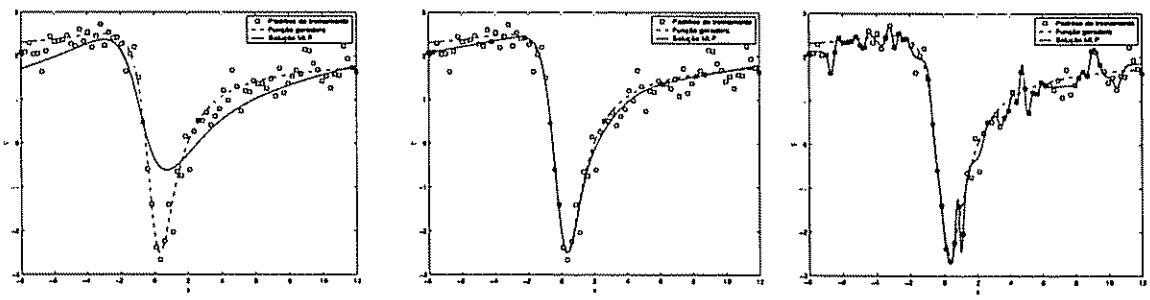
onde $\mathcal{X}$ e $\mathcal{D}$ são variáveis aleatórias e $\mathbf{x} \in \mathcal{X}$ e $\mathbf{d} \in \mathcal{D}$, e é uma função de custo local, $f(\mathbf{x};\mathbf{w})$ é a função implementada pela RNA, $\mathbf{x}$ é o vetor de entrada, $\mathbf{d}$ é o vetor de saída desejada, $\mathbf{w}$ denota o vetor de pesos da rede, p representa a distribuição de probabilidade conjunta de $\mathbf{x}$ e $\mathbf{d}$. O objetivo do treinamento é encontrar um vetor $\mathbf{w}$ tal que e_V seja minimizado:

$$\mathbf{w}^* = \underset{\mathbf{w}}{arg\ min}\ e_V(\mathbf{w})$$

onde e_V é o erro de generalização [Bengio, 1996], ou seja, o desempenho esperado da rede para novos padrões escolhidos aleatoriamente com probabilidade $p(\mathcal{X},\mathcal{D})$. Na prática $p(\mathcal{X},\mathcal{D})$ não é conhecida. Ao invés disso é dado o conjunto de treinamento $\mathcal{T} = \{\mathbf{x}_p, \mathbf{d}_p\}_1^{N_p}$ onde N_p é o número de padrões e uma aproximação de e_V é minimizada a qual é chamada de "erro empírico" [Vapnik, 1982] ou erro de treinamento [Bengio, 1996], dado pela Equação (2.2).

$$e_T = \sum_{p=1}^{N_p} e(f(\mathbf{x}_p; \mathbf{w}), \mathbf{d}_p) \tag{2.2}$$

Um modelo treinado de forma que o erro empírico e_T seja minimizado não garante que e_V também o seja porque é levado em consideração um conjunto de N_p padrões para o cálculo de e_T. O fato de se fazer e_T pequeno não garante que e_V também será e o valor de e_T que faz com que e_V seja minimizado não é conhecido a priori. Para exemplificar o que pode ocorrer com o erro de generalização quando o treinamento de uma rede é feita somente através da minimização do erro empírico, é dado um exemplo.



(a) *Underfitting* (b) Ajuste adequado (c) *Overfitting*

Figura 2.2: Mapeamento de Funções - os efeitos *underfitting* e *overfitting*

Considere um caso simples de mapeamento de função com apenas uma entrada e uma saída mostrado na Figura 2.2 onde os padrões de treinamento contêm ruído. A função geradora dos dados corresponde à linha tracejada e a linha contínua corresponde à resposta da RNA em três situações distintas. Na primeira delas, Figura 2.2(a), a RNA está sub-ajustada aos dados e portanto sofre do efeito chamado *underfitting*. Observe que o erro médio para os padrões de treinamento é o maior, em relação aos outros dois casos. Por outro lado, na Figura 2.2(c) pode-se observar que o erro médio para os padrões de treinamento é o menor. Diz-se para este caso que o modelo está super-ajustado aos dados, inclusive ao ruído presente nos mesmos. Este fenômeno é chamado de *overfitting*. Note que o erro para os padrões de treinamento pode ser pequeno mas o erro para padrões de teste pode ser elevado. Os efeitos *overfitting* ou *underfitting* quando presentes no modelo degradam a capacidade de generalização. A Figura 2.2(b) ilustra a situação onde o ajuste está adequado. A qualidade do modelo obtido pode ser influenciada pelo número de padrões utilizados no treinamento, o número de épocas do treinamento e o número de parâmetros livres (pesos) da rede. Uma boa escolha para estes valores pode permitir um equilíbrio adequado entre polarização e variância [Geman et al., 1992] e, consequentemente, alto desempenho para padrões de teste. Na Figura 2.2(a) o modelo apresenta alta polarização e baixa variância. Na Figura 2.2(c), a polarização é baixa

mas a variância é alta. Existe um ótimo entre dois extremos e o que se pretende com a abordagem multi-objetivo é encontrar um equilíbrio entre estes efeitos, os quais, diga-se de passagem, são contraditórios.

Para melhor explicar os fenômenos *underfitting* e *overfitting* e suas relações com os efeitos da variância e da polarização, é abordado a seguir o dilema entre polarização e variância [Geman et al., 1992]. Este dilema é central em RNAs e uma rede com alta capacidade de generalização precisa equilibrá-lo bem.

2.1 O dilema entre polarização e variância

Para melhor entendimento dos efeitos da polarização e da variância, é dada a seguir uma noção intuitiva. Como este dilema é central em RNAs, é feita também uma abordagem formal na Seção 2.1.2.

2.1.1 Noção intuitiva

Em problemas de aprendizagem onde se faz uso do somatório dos erros quadráticos como função de custo, o erro pode ser decomposto em dois termos, um devido à escolha de um certo conjunto de treinamento (variância) e outro devido à diferença entre a função que se deseja e a função estimada (polarização). O conjunto de dados utilizado para treinamento de uma RNA é apenas um entre os infinitos conjuntos que podem ser gerados a partir da função geradora. Naturalmente, diferentes conjuntos de treinamento extraídos a partir de uma mesma função geradora em condições de ruído podem fazer com que a RNA represente diferentes funções. Na maioria dos problemas práticos de aprendizagem o conjunto de treinamento tem número de pontos finito e a distribuição dos mesmos não é conhecida, o que torna impossível a determinação da variância e da polarização. A Figura 2.3 ilustra um comportamento hipotético dos erros devido à polarização e à variância em função da complexidade do modelo. Os efeitos devido à polarização e à variância somados formam o erro quadrático. Note que na medida em que o erro devido à polarização diminui, o erro devido à variância aumenta e na medida em que a complexidade C do modelo aumenta, os efeitos da polarização são minimizados e os da variância são maximizados.

Para demonstrar o erro devido à variância e à polarização, considere o exemplo a seguir.

Um conjunto de treinamento é amostrado na função geradora descrita pela Equação (2.3) mais um termo de ruído ξ.

$$d(x) = \frac{(x-2)(2x+1)}{(1+x^2)} + \xi \tag{2.3}$$

O termo ξ denota um ruído gaussiano de variância $\sigma^2 = 0.2^2$ e média $\mu = 0$. A

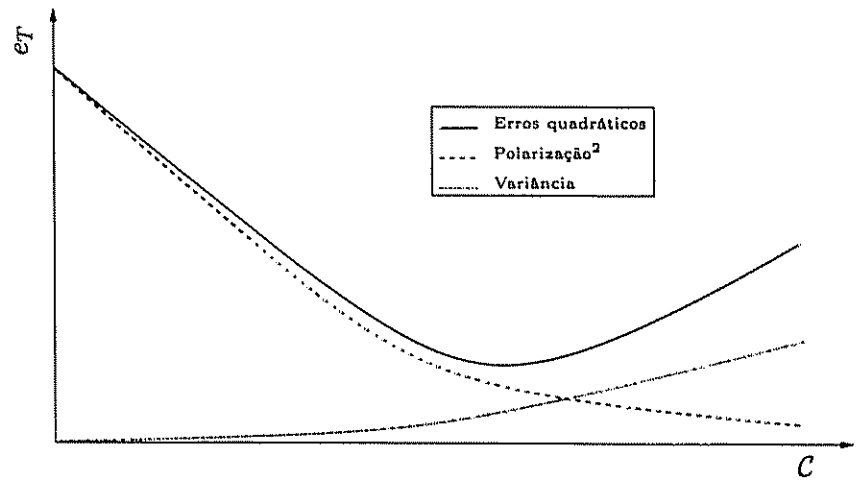
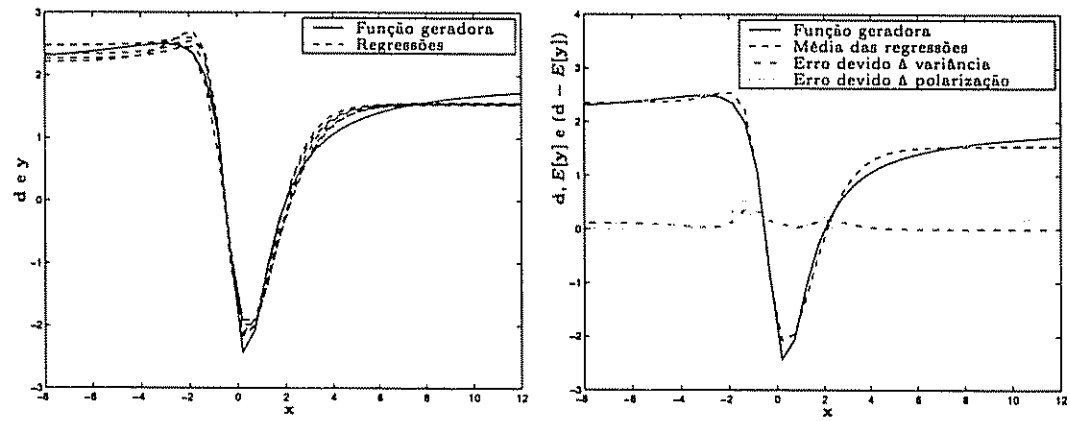


Figura 2.3: Decomposição do somatório dos erros quadráticos no quadrado da polarização mais a variância em função da complexidade C da rede.

variável x possui distribuição uniforme no intervalo entre $[-8, 12]$. Cinco conjuntos de dados são gerados com 40 padrões cada e dois procedimentos são usados para fazer a regressão.

Procedimento 1: RNAs com topologia $1 \times 2 \times 1$ são treinadas com cada um dos cinco conjuntos de dados de treinamento. A função de ativação do neurônio da camada intermediária é tangente hiperbólica ($tanh(\cdot)$) e do neurônio de saída é linear.

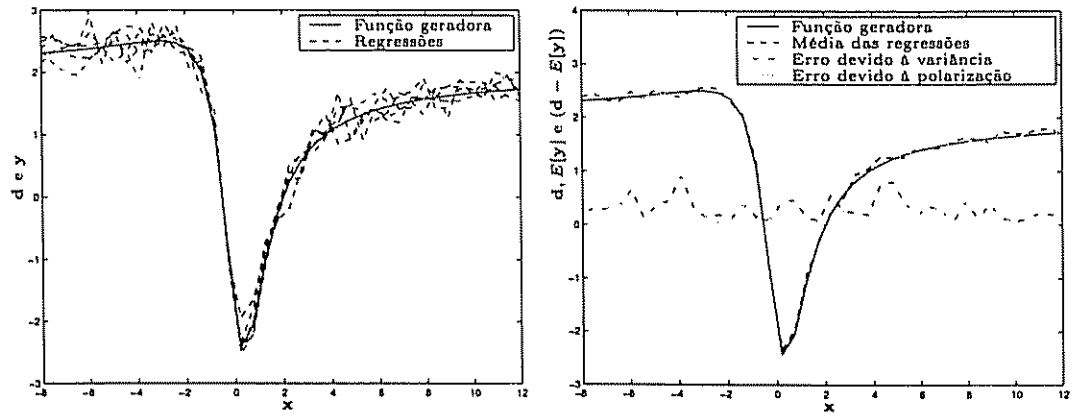


(a) Cinco regressões (b) Média das regressões e erro

Figura 2.4: Predominância do erro devido à polarização

Procedimento 2: RNAs com topologia $1 \times 30 \times 1$ são treinadas com cada um dos cinco conjuntos de dados de treinamento. A função de ativação do neurônio da camada intermediária é tangente hiperbólica ($tanh(\cdot)$) e do neurônio de saída é linear.

Nas Figuras 2.4(b) e 2.5(b), os erros devido à variância e à polarização foram multiplicados por 10 para serem melhor visualizados.



(a) Cinco regressões (b) Média das regressões e erro

Figura 2.5: Predominância do erro devido à variância

A Figura 2.4(a) ilustra as cinco aproximações conseguidas com os cinco conjuntos de dados diferentes seguindo o procedimento 1. Note que apesar de os conjuntos de treinamento serem diferentes, há uma tendência em se obter aproximações semelhantes mas com um erro comum em relação à função sem ruído. Este erro é chamado erro de polarização. Por outro lado, a Figura 2.5(a) ilustra a variabilidade das aproximações conseguidas através do procedimento 2. Esta variabilidade das aproximações para diferentes conjuntos de treinamento é chamada de variância.

A Figura 2.4(b) mostra a média das aproximações conseguidas com o procedimento 1. Note que em média, não se consegue uma aproximação correta através do procedimento 1. A Figura 2.5(b) ilustra a média das aproximações conseguidas com o procedimento 2. Note que a média das aproximações com variabilidade elevada se ajusta bem à função geradora, portanto baixo erro devido à polarização.

Para os exemplos abordados, o procedimento 1 fornece aproximações com erros elevados devido à polarização. Diz-se que estes modelos estão sub-ajustadas aos dados (*underfitting*) e não apresentam boa capacidade de generalização porque a complexidade da função a ser modelada é superior à complexidade do modelo. As aproximações conseguidas através do procedimento 2 apresentam elevados erros devido à variância (*overfitting*) não constituindo também bons modelos porque o resultado pode variar muito em função do conjunto de dados.

2.1.2 Abordagem formal

O problema de aprendizagem é construir uma função $f(\mathbf{x})$ baseada em um conjunto de treinamento $\mathcal{T} = \{\mathbf{x}_i, d_i\}_{i=1}^{N_T}$ com o propósito de aproximar a função $d(\mathbf{x})$ em observações futuras de $\mathbf{x}$. Tipicamente, $f(\mathbf{x})$ é escolhida de tal forma que uma função de custo seja

minimizada. Em RNAs do tipo MLP [Rumelhart et al., 1986b, Rumelhart et al., 1986c], é muito usual a utilização do somatório dos erros quadráticos observados

$$e_T = \sum_{i=1}^{N} [d_i - f(\mathbf{x}_i)]^2$$

como função de custo e $f(\mathbf{x})$ é escolhida para fazer tal função tão pequena quanto possível. A minimização desta função de custo não é sobre todas as funções f possíveis e sim sobre uma classe de funções geradas em função dos valores que os parâmetros de f possam assumir. A regressão de $d(\mathbf{x})$ em $\mathbf{x}$ é $E[d|\mathbf{x}]$ que é a função de $\mathbf{x}$ que dá o valor médio de d condicionado a $\mathbf{x}$. Como a função $f(\mathbf{x})$ é dependente do conjunto de treinamento $\mathcal{T} = \{\mathbf{x}_i, d_i\}_{i=1}^{N_T}$, a função será denotada por $f(\mathbf{x}, \mathcal{T})$ ao invés de simplesmente $f(\mathbf{x})$.

Dado um conjunto $\mathcal{T}$ e dado um vetor $\mathbf{x}$ particular, uma medida do desvio de $f(\mathbf{x}, \mathcal{T})$ enquanto um preditor de $d(\mathbf{x})$ é dada pela Equação (2.4) onde $E[\cdot]$ denota a esperança matemática.

$$E[(d(\mathbf{x}) - f(\mathbf{x}; \mathcal{T}))^2 | \mathbf{x}, \mathcal{T}] \tag{2.4}$$

A média dos erros quadráticos de f em relação ao conjunto de treinamento $\mathcal{T}$ como um estimador da regressão $E[d|\mathbf{x}]$ é dada pela Equação (2.5),

$$E_{\mathcal{T}}[(f(\mathbf{x}; \mathcal{T}) - E[d|\mathbf{x}])^2] \tag{2.5}$$

onde $E_{\mathcal{T}}$ representa a esperança em relação ao conjunto de treinamento $\mathcal{T}$, que é a média sobre todas as possíveis realizações de $\mathcal{T}$ com tamanho N_T fixo.

Para um determinado conjunto de treinamento $\mathcal{T}$, $f(\mathbf{x}; \mathcal{T})$ pode ser uma boa aproximação para $E[d|\mathbf{x}]$, logo, um preditor de $d(\mathbf{x})$ quase ótimo. Entretanto, pode ser também que $f(\mathbf{x}; \mathcal{T})$ seja muito diferente para uma outra realização de $\mathcal{T}$ e geralmente o é. Isto ocorre principalmente quando se tem a complexidade do modelo acima da necessária, fazendo com que $f(\mathbf{x}; \mathcal{T})$ varie muito em função de $\mathcal{T}$. Também, pode ser que a média (sobre todas as possíveis realizações de $\mathcal{T}$) de $f(\mathbf{x}; \mathcal{T})$ seja distante de $E[d|\mathbf{x}]$. Neste caso, a complexidade do modelo está aquém da necessária fazendo as regressões $f(\mathbf{x}; \mathcal{T})$ serem aproximadamente as mesmas para diferentes conjuntos de dados $\mathcal{T}$ e ainda assim, distantes de $E[d|\mathbf{x}]$. Estas circunstâncias contribuirão com valores elevados para a Equação (2.5), fazendo $f(\mathbf{x}; \mathcal{T})$ ser um preditor ruim de d.

Uma forma usual de tratar o erro dado pela Equação (2.5) é através da decomposição do mesmo em polarização e variância. Esta decomposição foi reescrita do trabalho

[Geman et al., 1992] e pode ser vista a seguir.

$$
\begin{aligned}
E_{\mathcal{T}}[(f(\mathbf{x};\mathcal{T}) - E[d|\mathbf{x}])^2] &= E_{\mathcal{T}}[((f(\mathbf{x};\mathcal{T}) - E_{\mathcal{T}}[f(\mathbf{x};\mathcal{T})]) + (E_{\mathcal{T}}[f(\mathbf{x};\mathcal{T})] - E[d|\mathbf{x}]))^2] \\
&= E_{\mathcal{T}}[(f(\mathbf{x};\mathcal{T}) - E_{\mathcal{T}}[f(\mathbf{x};\mathcal{T})])^2] + E_{\mathcal{T}}[(E_{\mathcal{T}}[f(\mathbf{x};\mathcal{T})] - E[d|\mathbf{x}])^2] \\
&\quad +2E_{\mathcal{T}}[(f(\mathbf{x};\mathcal{T}) - E_{\mathcal{T}}[f(\mathbf{x};\mathcal{T})])(E_{\mathcal{T}}[f(\mathbf{x};\mathcal{T})] - E[d|\mathbf{x}])] \\
&= E_{\mathcal{T}}[(f(\mathbf{x};\mathcal{T}) - E_{\mathcal{T}}[f(\mathbf{x};\mathcal{T})])^2] + (E_{\mathcal{T}}[f(\mathbf{x};\mathcal{T})] - E[d|\mathbf{x}])^2 \\
&\quad +2E_{\mathcal{T}}[f(\mathbf{x};\mathcal{T}) - E_{\mathcal{T}}[f(\mathbf{x};\mathcal{T})]] \cdot (E_{\mathcal{T}}[f(\mathbf{x};\mathcal{T})] - E[d|\mathbf{x}]) \\
&= \underbrace{(E[d|\mathbf{x}] - E_{\mathcal{T}}[f(\mathbf{x};\mathcal{T})])^2}_{\text{Polarizacão}} + \underbrace{E_{\mathcal{T}}[(f(\mathbf{x};\mathcal{T}) - E_{\mathcal{T}}[f(\mathbf{x};\mathcal{T})])^2]}_{\text{Variância}}
\end{aligned}
$$

Se em média $f(\mathbf{x};\mathcal{T})$ for diferente de $E[d|\mathbf{x}]$, então se diz que o estimador de $E[d|\mathbf{x}]$ é polarizado. Mesmo se o estimador for não polarizado, ainda assim o somatório dos erros quadráticos pode ser grande devido à variância. Mesmo com $E[f(\mathbf{x};\mathcal{T})] = E[d|\mathbf{x}]$, $f(\mathbf{x};\mathcal{T})$ pode ser altamente sensível aos dados e tipicamente, distante distante de $E[d|\mathbf{x}]$. Contudo, o erro devido à polarização e o erro devido à variância podem contribuir para o baixo desempenho do modelo.

Existe um equilíbrio entre a contribuição da polarização e da variância para o erro de estimação. Se este equilíbrio for alcançado, a função $f(\mathbf{x};\mathcal{T})$ será um estimador adequado de $E[d|\mathbf{x}]$.

Uma forma de minimizar o efeito da variância é aumentar o conjunto de dados usado para treinamento [Bishop, 1995] ou ainda fazer uma redução da complexidade do modelo ou da rede. Aumentar o conjunto de dados nem sempre é possível e a redução da complexidade da rede pode levá-la a aumentar o erro devido à polarização. Logo, a rede não pode ser demasiadamente complexa para não se ajustar aos dados que quase sempre estão contaminados com ruído e também não pode ter complexidade demasiadamente baixa, o que resultaria em soluções polarizadas. Encontrar o equilíbrio entre polarização e variância significa obter soluções com elevada capacidade de generalização.

Nesta Tese é proposto um método de treinamento de RNAs através do qual conseguem-se equilibrar os efeitos da variância e da polarização, o que possibilita o alcance de soluções com alta capacidade de generalização além de não exigir que o usuário tenha muito conhecimento do problema de aprendizagem a ser tratado, sendo a solução final pouco sensível à escolha de um determinado conjunto de parâmetros de treinamento fornecidos pelo usuário.

Existem na literatura várias abordagens para treinamento de RNAs que visam a minimizar os efeitos provocados pelo *underfitting* e *overfitting*. Algumas abordagens fazem ajuste de complexidade através de alterações na estrutura física das redes e outras se baseiam em métodos estatísticos. Estas abordagens caminham no sentido de equilibrar

os efeitos da polarização e da variância. Análises críticas de cinco destas abordagens são feitas nas subseções seguintes.

Dentre os vários métodos que visam à melhoria da capacidade de generalização das RNAs, os cinco abordados para efeito de comparação são:

- *Weight Decay* [Hinton, 1989];
- *Optimal Brain Damage* [Cun et al., 1990];
- Interrupção do treinamento (*Early Stopping*) [Weigend et al., 1990];
- Validação cruzada (*Cross-Validation*) [Stone, 1974, Stone, 1978];
- *Support Vector Machines* [Cortes and Vapnik, 1995].

Uma análise de cada método acima relacionada é feita a seguir.

2.2 O Algoritmo de *Pruning Weight Decay*

Os algoritmos de *pruning* podem ser classificados em dois grupos sendo que no primeiro deles, é feita uma estimativa da sensibilidade da função de erro em relação a um dado elemento. Os elementos que afetam menos a função de erro são removidos [Cun et al., 1990, Mozer and Smolensky, 1989]. O outro grupo adiciona termos de penalidade à função objetivo [Hinton, 1989, Weigend et al., 1991], proporcionando um controle de complexidade do modelo através da alteração da estrutura da rede, o que pode levar a uma solução eficiente. Por exemplo, um termo proporcional à magnitude dos pesos pode ser adicionado, favorecendo soluções com pesos de baixa magnitude, ou seja, com norma menor para o vetor de pesos. Aqueles elementos com magnitudes inferiores a um δ especificado podem ser removidos por influenciarem pouco a saída. Em geral, métodos de *pruning* fazem uma redução da estrutura das RNAs, reduzindo com isto complexidade do modelo.

O algoritmo *Weight Decay* [Hinton, 1989] é um método de *pruning* que modifica a função de custo penalizando as soluções com normas elevadas. A modificação consiste em adicionar um termo de regularização que penaliza os pesos com grandes magnitudes conforme pode ser visto na Equação (2.6).

$$J(\mathbf{w}) = e(\mathbf{w}) + \frac{\lambda}{2} \|\mathbf{w}\|^2 \tag{2.6}$$

O termo $e(\mathbf{w})$ é o somatório dos erros quadráticos e é dado pela Equação (2.7).

$$e(\mathbf{w}) = \sum_{i=1}^{N} (d_i - y_i)^2 \tag{2.7}$$

Aqui, λ representa a importância relativa entre o termo de complexidade ou regularização e o termo somatório dos erros quadráticos $e(\mathbf{w})$. Considerando o gradiente para a nova função de custo e caminhando no sentido de minimizá-la, encontra-se a regra para ajuste dos pesos dada pela Equação (2.8).

$$\Delta w_k = -\rho \frac{\partial e}{\partial w_k} - \rho \lambda w_k \tag{2.8}$$

A função de custo dada pela Equação (2.6) leva à geração de soluções onde os pesos assumem valores pequenos, resultando em baixos valores para a mesma. Soluções onde ocorre a presença de pesos com valores elevados não são obtidas porque levariam a função de custo para valores mais altos.

Uma outra forma de se adicionar um termo de penalidade mas de modo que a magnitude dos pesos seja regulada foi proposta por [Weigend et al., 1991]. O procedimento para eliminação de pesos é dado pela minimização da Equação (2.9).

$$J(\mathbf{w}) = e(\mathbf{w}) + \frac{\lambda}{2} \sum_k \frac{w_k^2}{w_0^2} \left(1 + \frac{w_k^2}{w_0^2}\right)^{-1} \tag{2.9}$$

w_0 é um parâmetro positivo a ser determinado. Para valores grandes de w_0, este procedimento equivale ao descrito pela Equação (2.6) favorecendo soluções com pesos pequenos. Se w_0 é pequeno, soluções com alguns pesos com valores grandes são favorecidas. Na prática, é usado um valor para w_0 próximo da unidade. Deve-se notar que o procedimento de eliminação de pesos é muito sensível à escolha de λ.

O termo de penalidade faz com que os pesos tenham uma tendência de convergir para o menor valor absoluto e por outro lado, os pesos grandes podem comprometer a capacidade de generalização das redes. Em outras palavras, pesos grandes resultam em variância excessiva da saída [Geman et al., 1992] e pesos muito pequenos podem resultar em polarização das soluções. Por isto, a sintonia da constante de queda λ deve ser bem feita para que o algoritmo *Weight Decay* retorne uma boa solução. De acordo com [Bartlett, 1997], para uma boa capacidade de generalização é mais importante a magnitude dos pesos do que sua quantidade. Soluções com capacidade de generalização elevada podem ser alcançadas através do controle da magnitude dos parâmetros livres, sem que haja a necessidade da eliminação de parâmetros da rede. Baseado nesta afirmativa, o método MOBJ proposto neste trabalho consegue soluções que equilibram bem os efeitos da variância e da polarização controlando a complexidade efetiva das redes, uma vez que o número de parâmetros é mantido fixo.

Um problema fundamental com a técnica *Weight Decay* é que a constante de queda λ é aplicada em todos os pesos da rede mas estes, para uma boa generalização, requerem diferentes constantes de queda. Se ajustar uma única constante de queda já é uma tarefa difícil, ajustar várias constantes inviabiliza o método.

Ao contrário do algoritmo *Weight Decay*, o método MOBJ não tem uma constante de queda a ser aplicada aos pesos mas o processo de obtenção da solução se comporta como se cada peso tivesse sua própria constante de queda, de forma que a redução da norma ocorre através da redução da magnitude de pesos mas não implica em que todos os pesos sofrão redução com a mesma taxa. Com isto, o algoritmo MOBJ encontra facilmente soluções que estão entre os extremos de complexidade excessiva e de complexidade muito baixa.

2.3 O Algoritmo de *Pruning Optimal Brain Damage*

O algoritmo de *Pruning Optimal Brain Damage* [Cun et al., 1990] faz ajuste de complexidade alterando a estrutura da rede. Inicialmente, treina-se uma rede superdimensionada para o problema em questão e após este treinamento, os pesos são eliminados temporariamente e um cálculo indicando como cada peso da rede afeta a função de erro é feito. Este cálculo recebe o nome de cálculo de saliência e os pesos que apresentarem menor valor para esta quantidade são eliminados. Uma vez eliminados, um retreinamento deve ser feito para a nova topologia.

Para o cálculo da saliência é utilizada uma aproximação por série de Taylor da função de custo pela qual pode-se predizer o efeito de uma perturbação no vetor de pesos. Considerando uma perturbação δP no vetor de parâmetros, a função de custo é dada pela Equação (2.10),

$$\delta e = \sum_i g_i \delta p_i + \frac{1}{2} \sum_i h_{ii} \delta p_i^2 + \frac{1}{2} \sum_{i \neq j} h_{ij} \delta p_i \delta p_j + S(\|\delta P\|^3) \qquad (2.10)$$

onde, os elementos de δP são denotados por δp_i, g_i são os componentes do gradiente G de e em relação a P e h_{ij} são os elementos da matriz Hessiana H de e em relação a P.

$$g_i = \frac{\partial e}{\partial p_i} \quad \text{e} \quad h_{ij} = \frac{\partial^2 e}{\partial p_i \partial p_j}$$

A meta é encontrar um conjunto de parâmetros tais que quando estes são eliminados, fazem com que a função e tenha um incremento dentro de uma faixa permitida. O cálculo da matriz H é de alto custo e às vezes impossível devido ao número de parâmetros da rede. Contudo, algumas simplificações podem ser feitas como efetuar o cálculo apenas da diagonal da matriz H, o que faz o terceiro termo do lado direito da Equação (2.10) se anular. Considerando que os parâmetros só são eliminados depois de um processo de treinamento, pode-se ignorar o primeiro termo do lado direito. Por último, considerando que a função de custo seja quadrática, o último termo do lado direito também pode ser ignorado. Desta forma, a Equação (2.10) pode ser reescrita conforme a Equação (2.11).

$$\delta e = \frac{1}{2}\sum_i h_{ii}\delta p_i^2 \tag{2.11}$$

Em [Cun et al., 1990] é mostrada uma forma eficiente de calcular a diagonal da matriz Hessiana H. O método OBS (*Optimal Brain Surgeon*) [Hassibi and Stork, 1993] também é um algoritmo de *pruning* mas neste a matriz H é inteiramente calculada, evitando-se os erros provenientes da aproximação feita pelo algoritmo OBD. A dificuldade básica de se lidar com o método OBD assim como outros algoritmos de *pruning* é a necessidade de ajuste fino de parâmetros de usuário. Estes, se mal ajustados, podem produzir efeitos indesejáveis como eliminação total dos parâmetros da rede ou até mesmo não se conseguir o equilíbrio correto entre polarização e variância.

Pode-se melhorar também a generalização de redes do tipo *feedforward* interrompendo o treinamento precocemente. A seguir, um algoritmo que possibilita este tipo de abordagem é descrito.

2.4 Interrupção do Treinamento (*Early Stopping*)

Interrupção precoce do treinamento (*Early Stopping*) [Weigend et al., 1990] é também uma estratégia para obtenção de redes com boa capacidade de generalização, sendo sugerida pelo comportamento do erro de generalização em relação a um dado conjunto de validação $\mathcal{T}_V$ que passa por um ponto de mínimo durante o processo de minimização do erro de treinamento e_T em relação ao conjunto de dados $\mathcal{T}_T$.

No processo de treinamento de redes multi-camadas onde os dados são ruidosos pode-se notar que, à medida em que o treinamento evolui, o erro de validação (generalização) decresce até um ponto de mínimo e começa a crescer novamente apesar de o erro para os padrões de treinamento continuar decrescendo. Este fenômeno pode ser observado na Figura 2.6, que ilustra o comportamento dos erros de treinamento e validação para uma tarefa de aprendizagem hipotética.

Considerando o comportamento do erro de validação, observa-se que o seu ponto de mínimo não corresponde ao ponto de mínimo da curva erro de treinamento. Consequentemente, se a opção for treinar uma rede para que o erro de treinamento seja minimizado, esta rede não responderá da melhor forma possível aos padrões desconhecidos. Para resumir, quando se está trabalhando com dados ruidosos, o treinamento excessivo traz como consequência o *overfitting*. Para se evitar este efeito, deve-se treinar a rede até que o erro de validação comece a crescer. Neste ponto, o erro de treinamento não é mínimo mas é o menor possível para os padrões de validação e então o treinamento deve ser interrompido.

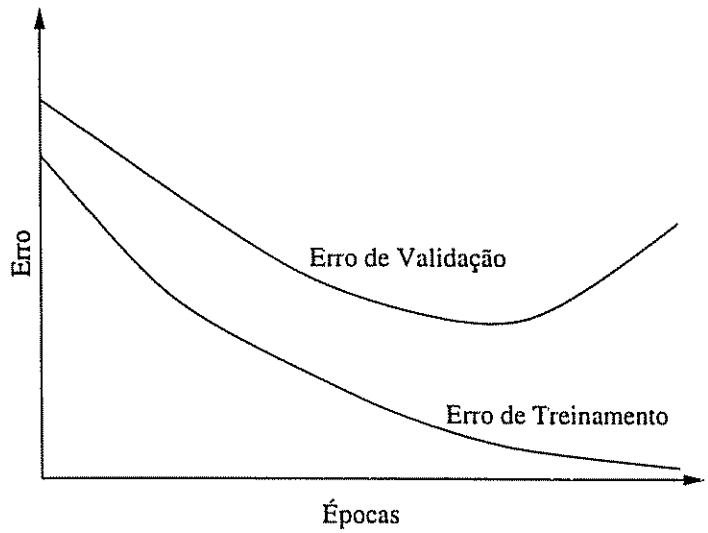


Figura 2.6: Dilema entre o MSE e a capacidade de generalização das RNAs.

Observando a Figura 2.6 pode-se notar que o processo de treinamento pode ser dividido em duas partes. Na primeira parte, caracterizada pelo início do treinamento até o ponto de mínimo da curva erro de validação, a rede se adapta somente às características principais dos dados, ou seja, aprende a função geradora. Na segunda parte, a medida em que o treinamento prossegue, o ruído também começa a ser mapeado pela rede.

Se uma rede tem complexidade adequada para que somente a função geradora do conjunto de treinamento seja capaz de ser aprendida então não há com o que se preocupar. O que ocorre na prática é que não se sabe a priori qual deve ser esta complexidade. Na maioria das vezes, escolhe-se uma rede mais complexa que o necessário. Neste caso, o que deve ser feito para evitar que o ruído presente nos dados seja modelado é interromper o treinamento quando o erro para os padrões de validação estiver no seu ponto de mínimo. A interrupção do treinamento antes que o mínimo "global" para o erro de treinamento seja alcançado tem o efeito de ajuste da complexidade da rede.

Se uma análise relativa ao valor da norma do vetor de pesos for feita, verifica-se que no ponto de máxima capacidade de generalização o vetor de pesos terá uma norma menor que no ponto onde o erro de treinamento é mínimo. Isto significa que as soluções onde o ruído foi modelado possuem normas mais elevadas.

Fato importante no treinamento de uma RNA com função de custo somatório dos erros quadráticos é que à medida em que o treinamento evolui, a magnitude dos pesos cresce. Quando se interrompe o treinamento, está se impondo forçosamente uma restrição para norma do vetor de pesos, evitando que ela cresça ainda mais, o que faria com que a rede modelasse não só a função que gerou os dados mas também o ruído quase sempre presente nos mesmos.

Para que o treinamento possa ser interrompido, é necessária a avaliação do erro de

validação. Esta avaliação do erro de generalização pode ser feita dividindo-se o conjunto de dados em duas partes. A primeira parte é utilizada para treinamento e a segunda parte é utilizada para cálculo do erro de validação da rede à cada época do treinamento. Nem sempre o erro de generalização tem comportamento monotonicamente decrescente até seu ponto de mínimo, podendo existir mínimos locais. Esta característica dificulta a determinação do melhor momento de interrupção. O fato de que a amostragem é única sobre o conjunto de dados total para constituir os conjuntos de treinamento e validação também podem prejudicar a qualidade da solução final. Além disso, o usuário deve decidir qual deve ser o tamanho dos conjuntos a serem utilizados.

Uma outra estratégia de escolha de um modelo com boa capacidade de generalização é conhecida como validação cruzada. Esta estrégia é também baseada na divisão do conjunto de dados para cálculo do erro de generalização. A seguir, este método é abordado.

2.5 Validação Cruzada (*Cross-Validation*)

Outra forma de se avaliar o erro de generalização para se conseguir sistemas com boa capacidade de generalização é conhecida na literatura como validação cruzada [Stone, 1978]. Esta estratégia é apropriada para melhorar a capacidade de generalização de redes com complexidade acima da necessária para uma dada tarefa evitando o *overfitting*. Validação cruzada é uma melhoria da validação por divisão da amostra como é feito no método da interrupção do treinamento, permitindo que todo o conjunto de dados seja usado para treinamento. Este método é superior ao método da interrupção precoce principalmente em casos onde o conjunto de dados é pequeno [Goutte, 1997]. A desvantagem da validação cruzada é a necessidade de se treinar várias redes. Existem várias formas de se implementar tal estratégia sendo que duas delas são abordadas a seguir.

Na primeira estratégia de implementação, chamado de *k-fold Cross-Validation*, o conjunto de dados é dividido em k partes de forma a constituir k conjuntos diferentes de treinamento e validação, que são utilizados para treinar k redes. Se k é igual ao tamanho total do conjunto de dados, este método se chama *leave-one-out Cross-Validation*.

Leave-one-out Cross-Validation apresenta boas estimativas para o erro de generalização para funções de erro contínuas como média dos erros quadráticos. Para funções de erro descontínuas como padrões mal classificados, este método pode apresentar baixo desempenho. Para este último caso, *k-fold Cross-Validation* pode apresentar melhores resultados. O valor 10 para k é frequentemente usado para se estimar o erro de generalização.

Sendo N o número total de padrões do conjunto de dados, cada conjunto de treinamento é constituído de $\frac{N}{k}(k-1)$ padrões e cada conjunto de validação é constituído de $\frac{N}{k}$ padrões, que são utilizados para cálculo do erro de generalização.

Este método tem a vantagem sobre o *Early Stopping* por trabalhar com amostragens

diferentes sobre o mesmo conjunto de dados. Como desvantagens pode-se citar a necessidade de se constituir vários conjuntos de dados, de se treinar várias redes que estão sujeitas aos parâmetros de treinamento e ainda ficarem estagnadas em mínimos locais. A validação cruzada fornece melhores resultados quando o conjunto de dados não é muito pequeno.

2.6 *Support Vector Machines*

Support Vector Machines (SVMs) [Boser et al., 1992, Cortes and Vapnik, 1995] é mais uma categoria de redes *feed-forward*. Foi apresentada por Vapnik pela primeira vez em 1992 e pode ser usada em problemas de classificação e regressão não linear. É baseada na teoria de aprendizagem estatística sendo mais precisamente uma aproximação do método Minimização do Risco Estrutural [Vapnik, 1995].

A idéia central das SVMs é mapear o espaço de entrada descrito por um vetor de entrada $\mathbf{x}$ em espaço de características $\mathbf{Z}$ de alta dimensão através de mapeamentos não lineares escolhidos a priori. No espaço de características, é construído um hiperplano ótimo de separação. A Figura 2.7 ilustra este mapeamento e o hiperplano de separação.

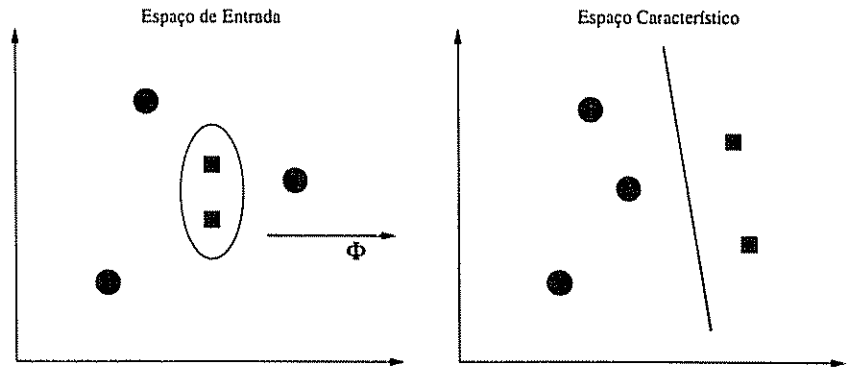


Figura 2.7: O princípio das SVMs. Mapeamento do espaço não linear de entrada no espaço de características de alta dimensão através da transformação $\Phi(\cdot)$

Um hiperplano de separação é considerado ótimo quando este maximiza a margem de separação entre as classes. A Figura 2.8 ilustra duas situações onde na primeira delas tem-se um hiperplano de separação não ótimo e portanto a margem de separação não é máxima e na segunda situação é mostrado um hiperplano ótimo, onde a margem está maximizada.

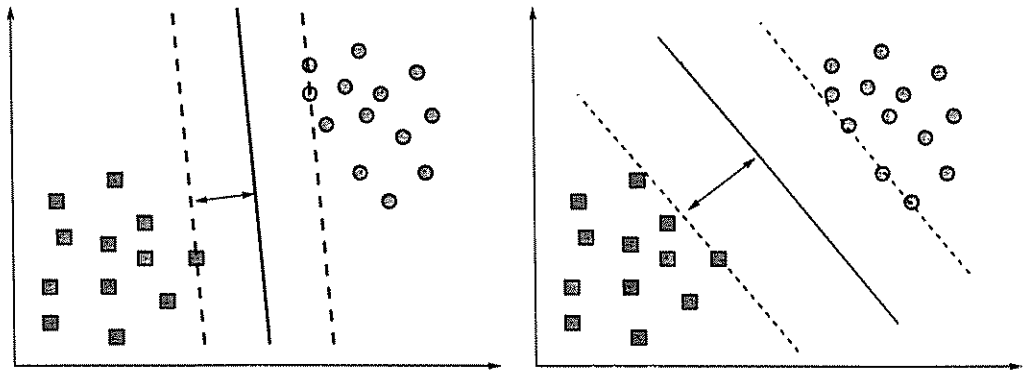


Figura 2.8: Ilustração de hiperplanos: (a) hiperplano não ótimo, margem não maximizada; (b) hiperplano ótimo, margem maximizada.

A noção central para construção de uma máquina de aprendizagem SVM é o *kernel* produto interno entre um vetor suporte $\mathbf{x}_i$ e um vetor $\mathbf{x}$ do espaço de entrada. Os vetores de suporte constituem o menor subconjunto extraído do conjunto de treinamento pelo algoritmo. Dependendo de como o *kernel* produto interno é gerado, pode-se construir diferentes tipos de máquinas como máquinas de aprendizagem polinomiais, função de base radial e *perceptron* 02 camadas (uma camada escondida). Para cada uma destas máquinas de aprendizagem, pode-se fazer uso do algoritmo de aprendizagem *Support Vector* onde o número de unidades escondidas é determinado automaticamente em função dos parâmetros de treinamento.

As SVMs diferem das abordagens convencionais para se treinar redes do tipo MLPs em um ponto fundamental. Na abordagem convencional, a complexidade do modelo é controlada pelo número de neurônios na camada escondida e SVM oferece uma forma para projeto de máquinas de aprendizagens onde o controle da complexidade do modelo é feito independente da dimensionalidade [Vapnik, 1995, Vapnik, 1998].

Conceitualmente, a dimensão do espaço de características (escondido) é feita propositadamente muito elevada para permitir a superfície de decisão na forma de hiperplano naquele espaço. Para garantir boa capacidade de generalização, a complexidade do modelo é controlada através da imposição de restrições na construção do hiperplano de separação, resultando na extração de parte do conjunto de treinamento como vetores de suporte.

Para a construção do hiperplano de separação, é necessário que seja resolvido um problema de otimização. Como otimização numérica no espaço de características de alta dimensão é muito cara, utiliza-se a noção de *kernel* produto interno e resolve-se um problema dual de otimização restrita formulado no espaço de entrada. Estes procedimentos resolvem a questão da dimensionalidade elevada do espaço de características.

Para o treinamento de uma SVM, a função de custo a ser minimizada é dada pela Equação 2.12, na sua forma primal.

$$J(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N_T} \alpha_i \left[d_i(\mathbf{w}^T\mathbf{x} + b) - 1\right] \tag{2.12}$$

O problema de otimização restrita em sua forma dual para construção da superfície de decisão é descrito a seguir onde a Equação (2.13) deve ser maximizada. Para resolver o problema de otimização, utiliza-se o método os multiplicadores de Lagrange [Bertsekas, 1995].

Dado um conjunto de treinamento $\{(\mathbf{x}_i, d_i)\}_{i=1}^N$, *encontre os multiplicadores de Lagrange* $\{\alpha_i\}_{i=1}^N$ *que maximiza a função objetivo descrita pela Equação (2.13).*

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i\alpha_j d_i d_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{2.13}$$

Sujeito às restrições:

1. $\sum_{i=1}^{N} \alpha_i d_i = 0$
2. $0 \leq \alpha_i \leq C$

onde C é um parâmetro especificado pelo usuário que denota o limite para os multiplicadores de Lagrange α, $\mathbf{d}$ denota a saída desejada, $\mathbf{x}$ denota o conjunto de entrada e $K(\mathbf{x}_i, \mathbf{x}_j)$ é o *kernel* produto interno.

O *kernel* produto interno $K(\mathbf{x}_i, \mathbf{x}_j)$ deve satisfazer ao teorema de Mercer [Mercer, 1909, Courant and Hilbert, 1970] e, dependendo da sua escolha, as máquinas de aprendizagem podem ser polinomial, função de base radial, *perceptron* multi-camadas, etc.

A Tabela 2.1 ilustra três tipos comuns de *Kernels*.

Tabela 2.1: Resumo dos *Kernels* produto interno

Tipo de SVM	*Kernel* produto interno $K(\mathbf{x}, \mathbf{x}_i)$, $i = 1, 2, \ldots, N$	Comentários
Máquina Polinomial	$(\mathbf{x}^T\mathbf{x}_i + 1)^P$	P é especificado pelo usuário
RBF	$\exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x} - \mathbf{x}_i\|^2\right)$	σ^2 é especificado pelo usuário
Perceptron 2 camadas	$\tanh(\beta_0\mathbf{x}^T\mathbf{x}_i + \beta_1)$	O teorema de Mercer é satisfeito para alguns valores de β_0 e β_1

Destes *Kernels*, alguns pontos são importantes:

1. O *Kernels* produto interno para SVMs polinomiais e RBFs sempre satisfazem o teorema de Mercer [Mercer, 1909, Courant and Hilbert, 1970]. Para o *kernel Perceptron* duas camadas este teorema nem sempre é satisfeito, dependendo dos valores de β_0 e β_1.

2. Em todos os tipos de SVM, a dimensão do espaço de características é determinada pelo número de vetores de suporte.

3. Para SVMs do tipo RBF, o número de funções de base radial e seus centros são determinados automaticamente pelo número de vetores de suporte e seus valores respectivamente.

4. Para SVMs do tipo *Perceptron* duas camadas, o número de neurônios da camada escondida e seus vetores de pesos são determinados automaticamente pelo número de vetores de suporte e seus valores, respectivamente.

A Figura 2.9 mostra a arquitetura de uma SVM.

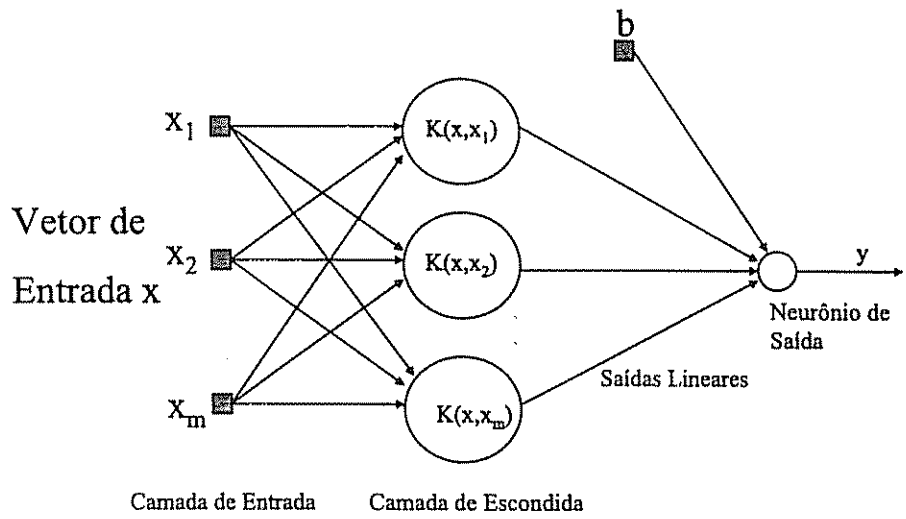


Figura 2.9: Arquitetura de uma SVM

Existem vários pacotes comerciais que podem ser usados para solucionar o problema de otimização quadrática [Luenberger, 1984]. Entretanto, as rotinas utilizadas possuem limitações. A quantidade de memória necessária para se resolver um problema de otimização quadrática cresce com o quadrado do número de pontos utilizados no treinamento. Como em aplicações reais geralmente se tem que trabalhar com alguns milhares de pontos, o problema de otimização quadrática não pode ser resolvido diretamente com rotinas de otimização comerciais. Em [Osuna et al., 1997a] desenvolveu-se um algoritmo que alcança a solução resolvendo uma sequência de sub-problemas menores. Com a alteração proposta em [Osuna et al., 1997a] obtiveram-se resultados satisfatórios com até 100 mil pontos de dados. Em termos de tempo de execução, as SVMs são mais lentas do que por exemplo, as redes MLPs devido ao número de vetores de suporte selecionados no processo de treinamento.

A questão de como controlar a seleção de vetores de suporte não é fácil, principalmente se os padrões a serem classificados são não separáveis e existe ruído. Um método

eficiente para seleção de vetores de suporte pode ser visto em [Barros et al., 2001] e em [Osuna and Girosi, 1998] é investigado o problema da redução do tempo de execução para problemas de classificação.

2.7 Conclusões do Capítulo

Os algoritmos abordados nas seções anteriores têm como objetivo obter modelos com boa capacidade de generalização. Para o caso do *Weight Decay*, boas soluções são conseguidas se o ajuste da constante de queda dos pesos é bem estimado. Para o caso da interrupção do treinamento, tem-se o problema da amostragem única para os conjuntos de validação e de treinamento além do comportamento do erro de generalização que pode possuir mínimos locais. No caso da validação cruzada, melhores resultados são fornecidos quando se tem conjuntos maiores de dados. Outro ponto importante quando se trabalha com divisões do conjunto de dados é determinar qual deve ser o tamanho de cada subconjunto de treinamento e validação e como os padrões devem ser amostrados. No caso das SVMs, a determinação do limite para os multiplicadores de Lagrange não é uma tarefa simples e influencia diretamente a capacidade de generalização destas máquinas.

O algoritmo MOBJ proposto não altera a estrutura da RNA ao longo do treinamento. Este algoritmo é capaz de encontrar uma boa solução mesmo que a complexidade da RNA seja elevada em relação ao problema de aprendizagem. O controle da complexidade efetiva é feito através da imposição de limites à norma do vetor de pesos. Não possibilitar que os pesos de uma rede variem livremente em uma larga faixa implica em obter soluções de variâncias reduzidas. Portanto, soluções de normas menores são também de complexidades efetivas menores.

Neste capítulo, foram abordadas questões importantes relativas ao processo de treinamento de redes neurais como complexidade, capacidade de generalização e "*overfitting*". É feita também uma revisão bibliográfica dos métodos que visam a maximização da capacidade de generalização de RNAs e que terão seus resultados posteriormente comparados com os obtidos pelo algoritmo MOBJ. São eles o *Weight Decay*, *Optimal Brain Damage*, *10-Fold Cross-Validation*, *Early stopping* e as SVMs. Vantagens e desvantagens destes métodos são relatadas além de uma breve descrição de cada algoritmo.

No Capítulo seguinte serão abordados alguns conceitos relativos à otimização multiobjetivo. Através destes conceitos, os objetivos erro e norma serão tomados para a formalização do método proposto.

Capítulo 3

Fundamentos da Otimização Multi-objetivo

Muitos algoritmos para treinamento supervisionado de RNAs consideram apenas o somatório dos erros quadráticos para os padrões de treinamento como função de custo [Rumelhart et al., 1986b, Riedmiller and Braun, 1993b, Fahlman, 1988, Hagan and Menhaj, 1994, Parma et al., 1998], o que não necessariamente faz com que estes algoritmos encontrem soluções com boa capacidade de generalização, principalmente quando se trabalha com dados ruidosos. Conforme já discutido no Capítulo 2, para que uma RNA tenha uma boa capacidade de generalização, esta não pode ser nem muito nem pouco complexa para evitar que o ruído presente nos dados seja modelado e também possibilitar a modelagem completa das características da função geradora dos dados. Portanto, para se conseguir boa capacidade de generalização, deve haver uma adequação entre a complexidade do modelo e a tarefa de aprendizagem em questão.

Existem duas formas básicas de se controlar a complexidade de uma RNA. A primeira delas é controlar o número de parâmetros livres [Lawrence et al., 1996] da rede, pois redes com muitos parâmetros são mais complexas do que as redes com poucos parâmetros. Outra forma é controlar a magnitude dos parâmetros [Bartlett, 1997], pois redes com muitos parâmetros mas porém com magnitudes reduzidas se comportam como sistemas menos complexos.

Neste trabalho, o controle da complexidade efetiva do modelo é feito através do redução da magnitude dos parâmetros livres. Considerando uma RNA com vetor de pesos dado por $\mathbf{w}$, uma forma de se medir simultaneamente a magnitude dos parâmetros livres é através da função $\|\mathbf{w}\|$. Qualquer variação em qualquer elemento do vetor $\mathbf{w}$ é refletida na função $\|\mathbf{w}\|$. Portanto, reduzir a norma do vetor de pesos implica em redução dos elementos que constituem o vetor $\|\mathbf{w}\|$.

Com a abordagem multi-objetivo pode-se considerar não só o erro devido aos padrões de treinamento como também a magnitude dos pesos expressos através da função

norma. No processo de treinamento proposto, as funções norma e erro devem ser minimizadas. Entretanto, estas funções são conflitantes em uma determinada faixa resultando em crescimento do erro quanto ocorre o decrescimento da norma e vice-versa. Esta região conflitante é a região de interesse onde pode ser encontrada a solução que melhor equilibra a complexidade e erro.

Assim, a otimização multi-objetivo se torna uma ferramenta importante para solucionar o problema de treinamento de redes neurais. Para facilitar o entendimento das técnicas multi-objetivos abordadas neste trabalho, são vistos a seguir alguns conceitos básicos em otimização multi-objetivo.

3.1 Conceitos Básicos

Um problema multi-objetivo parte da constatação de que, na existência de múltiplos objetivos, como é o caso do problema de treinamento de RNAs onde a norma e o erro são tomados como funções objetivo, existirão soluções que farão com que todos os objetivos melhorem simultaneamente implicando na existência de soluções melhores e ainda, existirão soluções que, comparadas com outras soluções, serão melhores quando um dos objetivos for considerado mas piores quando for considerado o outro. A geração desse último conjunto de soluções, denominadas soluções Pareto-ótimas, é um dos principais problemas da otimização multi-objetivo.

Para se saber o quanto é "boa" uma determinada solução, tem-se que avaliá-la utilizando algum critério. Estes critérios podem ser expressos como funções matemáticas, que possuem como argumento as variáveis de decisão e são chamados de "funções objetivo". Neste trabalho, são utilizadas duas funções objetivo e estas são conflitantes em uma determinada faixa. A Equação (3.1) descreve o problema multi-objetivo para treinamento de RNAs colocado neste trabalho.

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \begin{cases} f_1(\mathbf{w}) = e(\mathbf{w}) \\ f_2(\mathbf{w}) = \|\mathbf{w}\| \end{cases} \tag{3.1}$$

onde f_1 e f_2 são as duas funções objetivos, $e(\cdot)$ é um funcional erro e $\|\mathbf{w}\|$ é a norma Euclidiana do vetor de pesos da rede $\mathbf{w}$.

Denominam-se "Variáveis de decisão" as quantidades numéricas cujos valores são escolhidos no processo de otimização. Estas quantidades serão denotadas por w_j, $j = 1, 2, \ldots, z$.

O vetor para z variáveis de decisão $\mathbf{w}$ é representado pela Equação (3.2).

$$\mathbf{w} = [w_1, w_2, \ldots, w_z]^T \tag{3.2}$$

Em problemas de engenharia podem existir as chamadas restrições, impostas pelas características particulares do ambiente, pelos recursos disponíveis e etc. Elas devem ser satisfeitas e limitam o conjunto de soluções possíveis. O problema específico de treinamento de RNAs não possui restrições.

As funções objetivo denotadas por $f_1(\mathbf{w})$ e $f_2(\mathbf{w})$ podem ser escritas na forma de um vetor de funções $\mathbf{f}(\mathbf{w})$ como mostrado através da A Equação (3.3).

$$\mathbf{f}(\mathbf{w}) = [f_1(\mathbf{w}), f_2(\mathbf{w})]^T \tag{3.3}$$

Neste trabalho, são considerados dois espaços Euclidianos:

- O espaço z-dimensional das variáveis de decisão no qual cada coordenada corresponde a uma componente do vetor $\mathbf{w}$;
- O espaço k-dimensional das funções objetivo no qual cada coordenada corresponde a uma componente do vetor $\mathbf{f}(\mathbf{w})$. Este espaço é frequentemente chamado de espaço dos objetivos.

Qualquer ponto no espaço z-dimensional representa uma solução que mapeia um ponto no espaço k-dimensional, onde se representa a solução em termos dos valores das funções objetivo.

As soluções multi-objetivo são as melhores soluções entre as quais não há como definir a partir das funções objetivo, que uma solução é melhor que outra. Para estas soluções não existe um ordenamento.

Um conjunto $\mathcal{H}$ é ordenado de acordo com a relação de ordem "$\leq$" se dados quaisquer dois elementos $x, y \in \mathcal{H}$ é sempre verdade que $x \leq y$ ou $y \leq x$ e as seguintes propriedades com relação a "$\leq$" são válidas:

i. $x \leq x$

ii. $x \leq y$ e $y \leq q \Rightarrow x \leq q$

iii. $x \leq y$ e $y \leq x \Rightarrow x = y$

Para o caso multi-objetivo, a definição relevante é a de *conjunto parcialmente ordenado* feita a seguir:

Um conjunto $\mathcal{H}$ é parcialmente ordenado se valem as propriedades (i), (ii) e (iii) mas nem sempre $x \leq y$ ou $y \leq x$, isto é, nem sempre x e y são comparáveis. Em $\mathbb{R}$ os conjuntos são ordenados e em $\mathbb{R}^z$, $z \geq 2$ os conjuntos são parcialmente ordenados.

Para vetores em $\mathbb{R}^z$, a seguinte notação é empregada:

$$\mathbf{x} \leq \mathbf{y} \Rightarrow \{x_i \leq y_i, \quad i = 1, 2, \ldots, z\}$$
$$\mathbf{x} < \mathbf{y} \Rightarrow \{x_i < y_i, \quad i = 1, 2, \ldots, z\}$$
$$\mathbf{x} = \mathbf{y} \Rightarrow \{x_i = y_i, \quad i = 1, 2, \ldots, z\}$$

Os operadores $\geq$ e $>$ são definidos analogamente e o operador $\neq$ é definido da seguinte forma:

$$\mathbf{x} \neq \mathbf{y} \Rightarrow \{\nexists\, i \mid x_i = y_i\}$$

ou seja:

$$\mathbf{x} \neq \mathbf{y} \nRightarrow \{x_i \neq y_i \; \forall \; i = 1, 2, \ldots, z\}$$

Esta definição de conjunto parcialmente ordenado é utilizada para definir o conjunto Pareto-ótimo.

Um problema de otimização multi-objetivo genérico pode ser definido da seguinte forma:

Encontre o vetor $\mathbf{w}^* = [w_1^*, w_2^*, \ldots, w_n^*]^T$ *que otimize o vetor de funções dado pela Equação (3.4).*

$$\mathbf{f}(\mathbf{w}) = [f_1(\mathbf{w}), f_2(\mathbf{w}), \ldots, f_k(\mathbf{w})]^T \tag{3.4}$$

onde $\mathbf{w} = [w_i, w_2, \ldots, w_n]^T$ *é o vetor das variáveis de decisão, sujeito às m restrições de desigualdade descritas pela Equação (3.5),*

$$g_i(\mathbf{w}) \geq 0 \qquad i = 1, 2, \ldots, m \tag{3.5}$$

e às p restrições de igualdade descritas pela Equação (3.6).

$$h_i(\mathbf{w}) = 0 \qquad i = 1, 2, \ldots, p \tag{3.6}$$

Em outras palavras, deseja-se determinar o conjunto $\{w_1^*, w_2^*, \ldots, w_n^*\}$ que satisfaça as Equações (3.5) e (3.6) de forma que as funções objetivo não possam ser melhorados simultaneamente ainda mais.

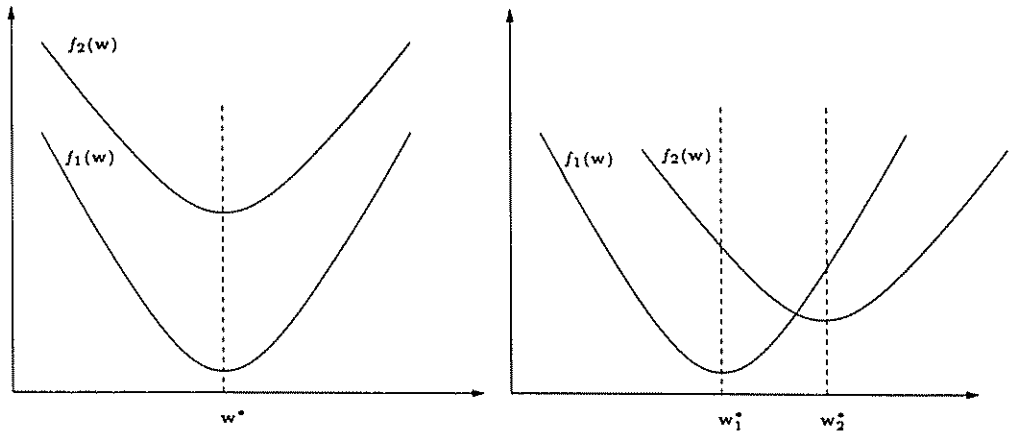
As restrições dadas por (3.5) e (3.6) definem a "Região Factível" $\mathcal{W}$ e qualquer ponto

$\mathbf{w} \in \mathcal{W}$ define uma "Solução Factível". O vetor de funções $\mathbf{f}(\mathbf{w})$ faz o mapeamento do conjunto $\mathcal{W}$ no conjunto Ω, com Ω representando todas as valores possíveis para as funções objetivo. As k componentes do vetor $\mathbf{f}(\mathbf{w})$ representam os critérios que devem ser considerados e, $g_i(\mathbf{w})$ e $h_i(\mathbf{w})$ representam as restrições impostas às variáveis de decisão. O vetor $\mathbf{w}^*$ denota a solução ótima (podendo existir mais de uma).

O significado de "solução ótima" não é bem definido neste contexto, uma vez que raramente se tem um vetor $\mathbf{w}^*$ tal que para todo $i = 1, 2, \ldots, k$ a desigualdade descrita pela Equação (3.7) se mantém.

$$\underset{\mathbf{w}\in\mathcal{W}}{\wedge} (f_i(\mathbf{w}^*)) \leq (f_i(\mathbf{w})) \tag{3.7}$$

Se isto fosse o caso, então $\mathbf{w}^*$ seria a solução desejada, mas normalmente, não se tem uma situação, semelhante a esta, na qual todas as $f_i(\mathbf{w})$ têm um mínimo em $\mathcal{W}$ num ponto comum $\mathbf{w}^*$. Exemplos onde a solução utópica pode ser factível e infactível podem ser vistos nas Figuras 3.1(a) e 3.1(b).



(a) Solução utópica factível (b) Solução utópica não factível

Figura 3.1: Soluções utópicas factíveis e não factíveis

Contudo, esta situação raramente acontece em problemas práticos e por isto, critérios devem ser estabelecidos para decidir o que é solução ótima.

Considere que se possa encontrar separadamente os mínimos ou os máximos das funções objetivo. Então, a Equação (3.8) descreve um vetor de variáveis o qual otimiza a i-ésima função objetivo $f_i(\mathbf{w})$. Em outras palavras, o vetor $\mathbf{w}^u \in \mathcal{W}$ é tal que a Equação (3.9) é verdadeira.

$$\mathbf{w}^u = [w_1^u, w_2^u, \ldots, w_n^u]^T \tag{3.8}$$

$$f_i(\mathbf{w}^u) = \min_{\mathbf{w}\in\mathcal{W}} f_i(\mathbf{w}), \quad \text{para } i = 1, 2, \ldots, k \tag{3.9}$$

Assim, o vetor $\mathbf{f}^u = [f_1(\mathbf{w}^u), f_2(\mathbf{w}^u), \ldots, f_k(\mathbf{w}^u)]^T$ é a solução ideal para o problema multi-objetivo e é denominada de "Solução Utópica". Este conceito é importante apesar de esta solução ser geralmente não factível.

Como foi dito anteriormente, não se tem um conceito preciso de ótimo em otimização multi-objetivo. A interpretação deste termo é feita aqui através do conceito de Pareto-ótimo. Este conceito foi formulado por Vilfredo Pareto em 1896 [Pareto, 1896] e constitui a origem da pesquisa nesta área.

Considerem-se $\mathcal{W} \subset \mathbb{R}^z$ e $\mathbf{f} : \mathbb{R}^z \mapsto \mathbb{R}^k$. Assim, uma definição para o conjunto Pareto-ótimo pode ser dada segundo a Equação (3.10).

$$\mathcal{W}^* \triangleq \{\mathbf{w}^* \in \mathcal{W} \mid \nexists\, \mathbf{w} \text{ tal que } \mathbf{f}(\mathbf{w}) \leq \mathbf{f}(\mathbf{w}^*) \text{ e } \mathbf{f}(\mathbf{w}) \neq \mathbf{f}(\mathbf{w}^*)\} \tag{3.10}$$

Em outras palavras, esta definição diz que $\mathbf{w}^*$ é Pareto ótimo se não há mais como melhorar todos os objetivos simultaneamente, ou seja, não existe outro vetor $\mathbf{w}$ que decrementaria alguns critérios sem provocar simultaneamente o incremento em pelo menos um outro critério. Quase sempre o Pareto-ótimo não é constituído por uma única solução, mas por um conjunto de soluções chamadas não-inferiores ou não-dominadas. Para exemplificar, considere as duas funções dadas pelas Equações (3.11) e (3.12).

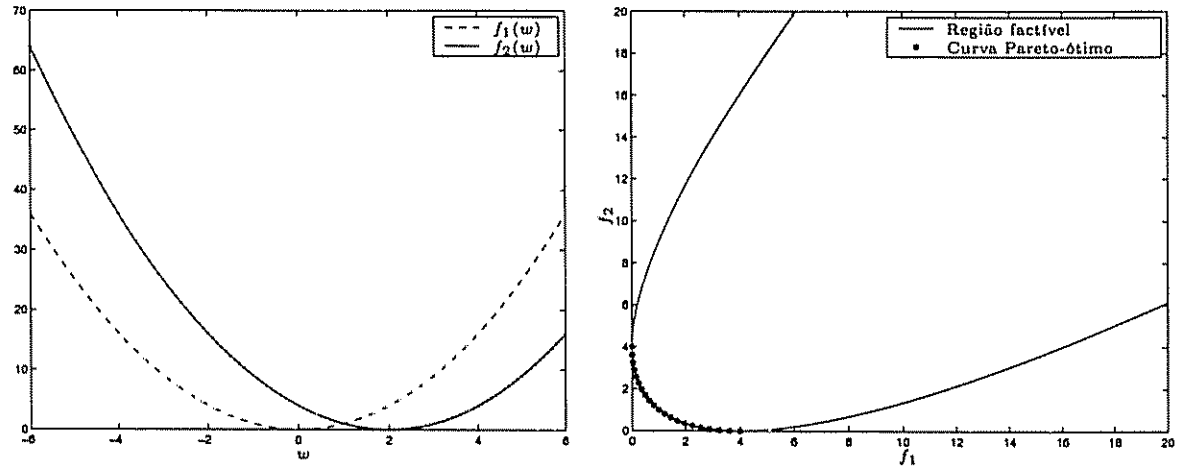
$$f_1(w) = w^2 \tag{3.11}$$

$$f_2(w) = (w-2)^2 \tag{3.12}$$

Estas duas funções são mostradas na Figura 3.2(a) para $w \in [-6, 6]$. Na Figura 3.2(b) é mostrado o espaço formado pelas duas funções $f_1(w)$ e $f_2(w)$, chamado de espaço dos objetivos. Neste espaço, para $w \in [a, b]$, a e $b \in \mathbb{R}$, um conjunto de pontos com coordenadas $(f_1(w), f_2(w))$ é obtido constituindo a região factível. Neste caso em particular onde $w \in \mathbb{R}$, esta região é uma curva. Tomando-se agora a Figura 3.2(a), pode-se perceber claramente que para $w < 0$ e $w > 2$, ambas as funções crescem e no intervalo $0 \leq w \leq 2$, um incremento em uma das funções resulta em um decremento da outra. Portanto, esta é a região chamada Pareto-ótima. Na Figura 3.2(b) a curva Pareto-ótima é mostrada e neste exemplo, as soluções não-dominadas ou não-inferiores ou Pareto-ótimas são as soluções pertencentes ao intervalo $[0, 2]$. Todas as soluções pertencentes a este intervalo são ótimas do ponto de vista da otimização multi-objetivo ou seja, $w^* \in [0, 2]$.

Continuando com os conceitos preliminares, um ponto $\mathbf{w}^* \in \mathcal{W}$ é uma solução fracamente não-dominada se não há um $\mathbf{w} \in \mathcal{W}$ tal que $f_i(\mathbf{w}) < f_i(\mathbf{w}^*)$, para $i = 1, 2, \ldots, k$. Um ponto $\mathbf{w}^* \in \mathcal{W}$ é uma solução fortemente não-dominada se não há um $\mathbf{w} \in \mathcal{W}$ tal que $f_i(\mathbf{w}) \leq f_i(\mathbf{w}^*)$, para $i = 1, 2, \ldots, k$ e para pelo menos um valor de i, $f(\mathbf{w}) < f(\mathbf{w}^*)$.

Também, se $\mathbf{w}^*$ é fortemente não-dominada, é também fracamente não-dominada



(a) Funções objetivo (b) Pareto-ótimo

Figura 3.2: Ilustração do conceito Pareto-ótimo - Caso bi-objetivo

mas o contrário não é necessariamente verdade. Soluções não-dominadas para o caso bi-objetivo podem ser representadas graficamente no espaço dos objetivos. A região dos pontos fortemente não-dominados corresponde à chamada "Curva mínima" e o lugar geométrico dos pontos fracamente não-dominados corresponde à chamada "Curva fracamente mínima". Estas duas curvas podem ser vistas na Figura 3.3. O conjunto das soluções não-inferiores ou não-dominadas é denotado por $\mathcal{W}^*$. Em otimização em Engenharia, soluções fortemente não-dominadas são procuradas e a qualificação "fortemente" é geralmente omitida.

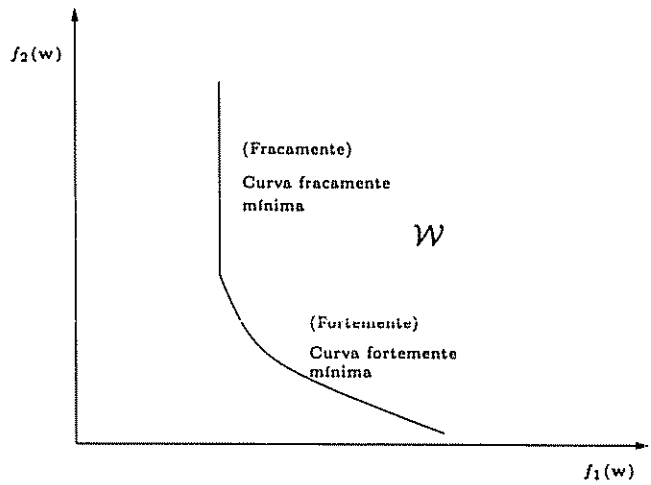


Figura 3.3: Curvas fortemente e fracamente não-dominadas para o caso bi-objetivo.

Existem várias técnicas de otimização multi-objetivo fundamentadas nos conceitos acima citados e uma delas é utilizada neste trabalho para resolver o problema do treinamento de RNAs do tipo MLP. A técnica utilizada é uma variante do problema ϵ-restrito

[Chankong and Haimes, 1983] conhecido na literatura chamada de técnica de relaxação [Takahashi et al., 1997]. A seguir é feito o detalhamento da técnica acima mencionada e da sua variação.

3.2 O Problema ε-restrito

Este método é baseado na minimização de um dos objetivos sendo que os demais são considerados como restrições. Logo, um problema restrito de otimização mono-objetivo é resolvido para diferentes valores de ε. A variação paramétrica de ε permite a geração de todo o conjunto Pareto-ótimo. Soluções Pareto-ótimas podem ser geradas a partir do seguinte resultado:

Teorema 1 *Se* $\mathbf{w}^* \in \mathcal{W}$ *é eficiente então existem inteiros* $i \in \{1, 2, \ldots, m\}$ *e números reais* ε_j, $j = 1, \ldots, m$ $(j \neq i)$ *tais que* $\mathbf{w}^*$ *resolve:*

$$\begin{aligned} &\min_{\mathbf{w} \in \mathcal{W}} \; f_i(\mathbf{w}) \\ &\text{sujeito a: } \{f_j(\mathbf{w}) \leq \varepsilon_j, \quad j = 1, \ldots, m \;\; (j \neq i)\} \end{aligned} \tag{3.13}$$

DEMONSTRAÇÃO: Defina $f_i^* = f_i(\mathbf{w}^*)$, $i = 1, \ldots, m$ e suponha que $\mathbf{w}^*$ não resolve o problema acima para nenhum $i \in [1, 2, \ldots, m]$ e números reais ε_j, $j = 1, \ldots, m$. Neste caso, para $i = 1$ e $\varepsilon_j = f_j^*$, $j = 2, \ldots, m$ deve existir $\mathbf{w}^o \in \mathcal{W}$ tal que:

$$f_1(\mathbf{w}^o) < f_1(\mathbf{w}^*) \quad e \quad f_j(\mathbf{w}^o) \leq f_j^*, \quad j = 2, \ldots, m$$

contradizendo a hipótese de que $\mathbf{w}^*$ é eficiente. ∎

Como implicação prática do teorema 1 conclui-se que, variando parametricamente ε_j, $\forall i$ e $\forall j \neq i$, é possível gerar completamente o conjunto $\mathcal{W}^*$.

Problema ε-restrito (Pε):

$$\min_{\mathbf{w} \in \mathcal{W}} \; f_i(\mathbf{w}) \tag{3.14}$$

$$\text{sujeito a: } f_j(\mathbf{w}) \leq \varepsilon_j, \quad j = 1, \ldots, m$$

Através do problema (Pε) descrito na Equação (3.14) é possível gerar $\mathcal{W}^*$ completamente, mesmo que o problema seja não-convexo. Para resumir, o problema (Pε) possui como características básicas a baixa complexidade computacional para problemas lineares, complexidade relativamente alta para problemas não lineares e a determinação dos reais

ε_j que geram soluções eficientes não é uma tarefa trivial. O ponto obtido pode não ser eficiente e dependendo do valor de ε_j, o problema pode se tornar infactível. Geralmente, estes problemas ocorrem quando existe um número muito grande de objetivos, o que não se configura para o problema abordado neste trabalho. Para a aplicação em treinamento de RNAs, o objetivo erro é tomado como função objetivo e o objetivo norma é tomado como restrição. Os detalhes da implementação podem ser vistos no Capítulo 4.

3.3 Uma Variante do Problema ε-restrito

Em [Takahashi et al., 1997] é proposta uma variação da abordagem do problema (Pε), contornando o problema básico da geração de problemas infactíveis.

A abordagem proposta por [Takahashi et al., 1997] é assim construída:

- Seja $\mathbf{u}^* \in \mathbb{R}^m$ o vetor de objetivos correspondente à "solução utópica" do problema.
- Seja $\mathbf{f}_i^* \in \mathbb{R}^m$; $i = 1, \ldots, m$ o vetor de objetivos associado a cada mínimo individual do problema.
- Seja $\mathcal{C}$ o cone gerado pelos vetores $(\mathbf{f}_i^* - \mathbf{u}^*)$, com origem em $\mathbf{u}^*$.
- Seja $\mathbf{v} \in \mathcal{C}$ um vetor construído segundo a Equação (3.15).

$$\mathbf{v} = \mathbf{u}^* + \sum_{i=1}^{m} \gamma_i (\mathbf{f}_i - \mathbf{u}^*) \tag{3.15}$$

para $\gamma_i > 0$.

O problema é então redefinido em um problema mono-objetivo com a função objetivo descrita pela Equação (3.16) e funções de restrição que incorporam as funções objetivo do problema multi-objetivo original descritas pela Equação (3.17).

$$\mathbf{w}^* = arg_{\mathbf{w}} \min_{\mathbf{w},\eta} \eta \tag{3.16}$$

$$\text{sujeito a} : \mathbf{f}(\mathbf{w}) \leq \mathbf{u}^* + \eta \mathbf{v} \tag{3.17}$$

sendo $\mathbf{w} \in R^n$, $f(\cdot) : \mathbb{R}^n \longmapsto \mathbb{R}^m$.

Este problema pode ser colocado no formato irrestrito através de um método de penalidade ou barreira [Luenberger, 1984] ou ainda ser resolvido através de um método de otimização não linear restrita como por exemplo, o método "elipsoidal" [Shor, 1977].

O método proposto por [Takahashi et al., 1997] é capaz de gerar todo o espaço de soluções Pareto-ótimas. Trabalha com o espaço de busca com $n + 1$ variáveis e $m + n$

restrições. Dessa forma, ele possibilita a organização da busca que seria realizada pelo método ε-restrito convencional, o qual corresponde a uma combinação de m problemas em n variáveis e $m+n-1$ restrições.

O método ε-restrito toma apenas um dos objetivos como função objetivo a ser otimizada e considera os demais objetivos como restrições. No método de relaxação [Takahashi et al., 1997], nome dado à variação do método ε-restrito, é criada uma nova função auxiliar linear como função objetivo e todos os objetivos originais do problema são colocados em funções de restrição, as quais são linearmente dependentes da variável auxiliar η. Este procedimento cria um novo problema sendo que o espaço de parâmetros possui a ordem do espaço original mais um e possui também um mínimo bem definido para a função objetivo auxiliar.

Todas as soluções pertencentes ao conjunto Pareto-ótimo geradas pelo método ε-restrito podem ser geradas pelo método de relaxação. Ambos os métodos podem gerar todo o conjunto Pareto-ótimo.

Uma propriedade inconveniente do método ε-restrito é que podem ser geradas soluções fracamente não-dominadas. Diferentemente deste, o método de relaxação não apresenta esta séria inconveniência além de não ter o problema de se encontrar soluções não factíveis em certas regiões do conjunto Pareto-ótimo. Entretanto, podem ser geradas soluções não eficientes.

Tendo em vista essas considerações, a heurística a seguir pode ser utilizada para gerar um conjunto representativo de pontos do espaço de soluções eficientes.

Os passos são os seguintes:

Passo 1: Determinar $\mathbf{u}^*$ e $\mathbf{f}_i^*$; $i = 1, \ldots, m$.

Passo 2: Escolher aleatoriamente um conjunto $\mathcal{V}$ de vetores $\mathbf{v}$, utilizando a Equação (3.15) e vetores γ gerados através de um gerador de números aleatórios com distribuição uniforme.

Passo 3: Para cada vetor $\mathbf{v} \in \mathcal{V}$, escolher um número η_0 suficientemente grande para tornar o problema descrito pela Equações (3.16) e (3.17) factível. Note-se que, sendo γ_i na Equação (3.15) um número estritamente positivo, sempre haverá algum valor de η que torna o problema factível, desde que o conjunto $\mathcal{W}$ não seja vazio.

Passo 4: Resolver cada problema assim construído. Dependendo do método utilizado, pode-se inicializar o problema com condições iniciais arbitrárias.

Vale dizer que muitas das afirmações feitas sobre o método ε-restrito não são válidas para o problema de treinamento de redes uma vez que as soluções fracamente não dominadas que forem geradas em determinados problemas são descartadas no processo de decisão. A variação paramétrica dos reais ε que geram as soluções eficientes no caso de treinamento de RNAs é uma tarefa trivial e será abordada no capítulo seguinte.

3.4 Conclusões do Capítulo

Neste Capítulo foram abordados todos os conceitos relativos à otimização multi-objetivo necessários à compreensão do algoritmo proposto bem como a descrição da técnica de otimização multi-objetivo ε-restrito e sua variação, chamada de método de relaxação. No Capítulo seguinte, as técnicas descritas neste Capítulo são aplicadas ao problema de treinamento de redes.

No Capítulo seguinte, tanto o método ε-restrito na sua forma original quanto sua variante serão utilizados para resolver o problema de treinamento de RNAs.

Capítulo 4

O Método Multi-objetivo MOBJ

4.1 Introdução

O que se busca em um processo de aprendizagem é reduzir o erro para os padrões de treinamento sem perder de vista que o modelo também deverá responder de forma satisfatória mediante à apresentação de padrões desconhecidos. O algoritmo MOBJ oferece uma forma para projeto de redes onde o controle da complexidade do modelo é feito independente da dimensionalidade. Através deste algoritmo, pode-se ter muitos parâmetros em uma rede e mesmo assim conseguir a complexidade adequada para uma dada tarefa controlando a magnitude dos parâmetros livres.

A relação entre o comportamento da magnitude dos pesos e o comportamento do erro para os padrões de treinamento é conflitante, ou seja, na medida em que a magnitude dos pesos decresce, o modelo se comporta como um sistema menos complexo, e o erro para os padrões de treinamento cresce. Por outro lado, na medida em que a magnitude dos pesos cresce, o erro de treinamento diminui e a rede se comporta como um sistema mais complexo. O que se deseja é encontrar uma solução tal que esta não apresente erros elevados para os padrões de treinamento mas também que não fique super-ajustada a estes. Deve haver um ponto de equilíbrio entre o erro para os padrões de treinamento e a complexidade da rede para que os efeitos do *overfitting* ou do *underfitting* sejam minimizados. Em outras palavras, os efeitos da variância e da polarização do modelo devem ser equilibrados.

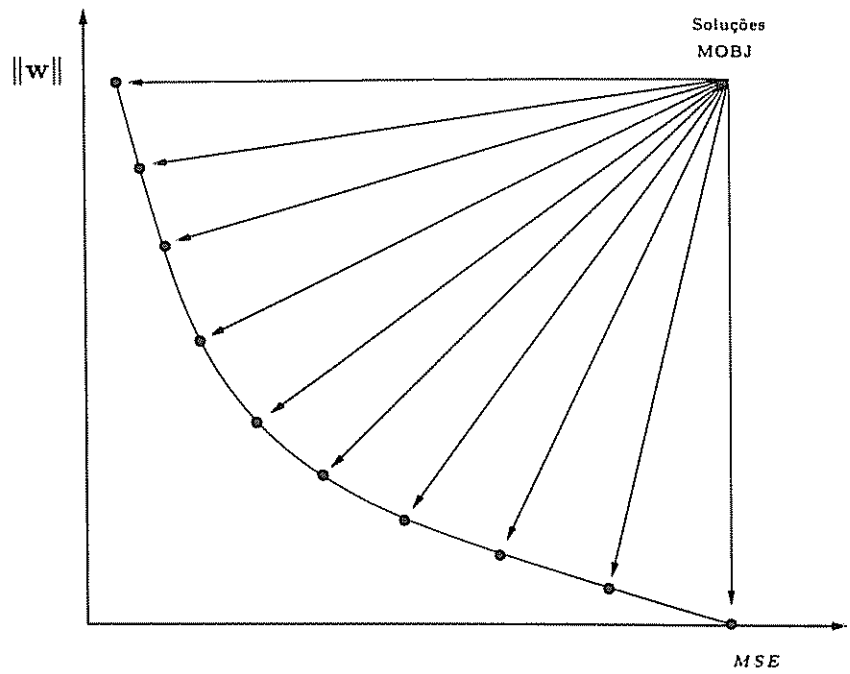


Figura 4.1: Conjunto Pareto-ótimo

A Figura 4.1 mostra o compromisso entre o erro para os padrões de treinamento e a norma do vetor de pesos. As soluções de normas mais elevadas apresentam erros menores para os padrões de treinamento, o que não significa erros menores para os padrões desconhecidos ou de validação. Por outro lado, as soluções de normas menores apresentam erros elevados para os padrões de treinamento e também para os padrões de validação devido à polarização. Entre estas soluções existe uma que melhor equilibra estes erros resultando em uma melhor capacidade de generalização. As soluções ilustradas na Figura 4.1 constituem o conjunto Pareto-ótimo e podem ser obtidas pelo método multi-objetivo de treinamento de redes do tipo MLP proposto.

4.2 Descrição do Método

O método de treinamento multi-objetivo proposto consiste em controlar a complexidade das redes através da minimização simultânea do erro para os padrões de treinamento e da norma do vetor de pesos [Teixeira et al., 2001]. A minimização destes objetivos é feita até que um ponto de equilíbrio seja alcançado, obtendo-se as soluções chamadas de não-dominadas ou Pareto-ótimas. Estas soluções são aquelas as quais não há mais como melhorar um dos objetivos sem que haja uma degradação do outro. Logo, considerando um determinado erro ϵ, a norma correspondente a este é a menor norma possível. Por outro lado, considerando uma determinada norma $\mathcal{N}$, o erro correspondente a esta é o menor erro possível. A Figura 4.2 mostra as soluções Pareto-ótimas e algumas soluções dominadas, como as soluções s_a, s_b, s_c e s_d. Para estas soluções, ambos os objetivos

podem ser minimizados simultaneamente e na abordagem proposta elas não podem ser geradas.

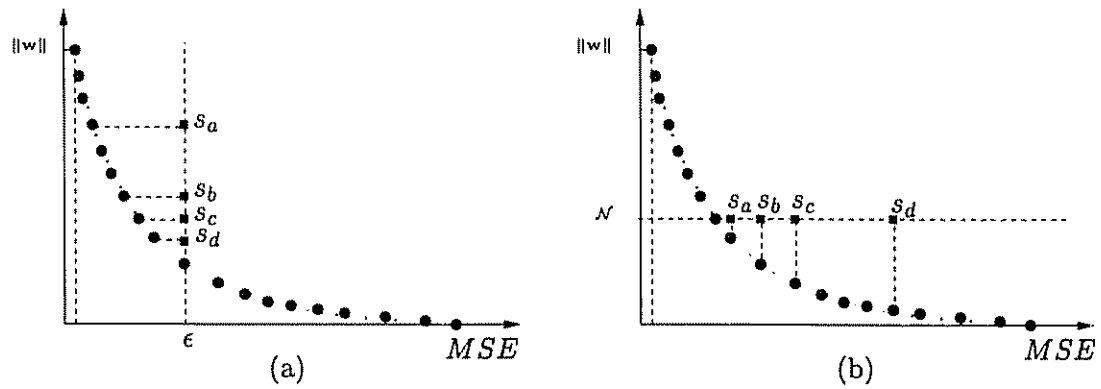


Figura 4.2: Demonstração do conjunto Pareto-ótimo e de soluções dominadas. (a) Soluções dominadas de mesmo erro. (b) Soluções dominadas de mesma norma.

Uma das vantagens do método é que o compromisso entre o erro e a complexidade, expressa através da norma, fica explícito. Outra vantagem importante é que o método de treinamento é capaz de encontrar um conjunto de soluções chamadas de soluções Pareto-ótimas que têm como principal característica o fato de que estas soluções são tais que o erro e a norma não podem mais serem minimizados simultaneamente, ou seja, uma melhora no objetivo norma resultará em uma piora do objetivo erro e vice-versa. Logo, para cada solução pertencente ao conjunto Pareto-ótimo, existirá um valor para o erro e para a norma específico. Este conjunto é constituído então de soluções que estão entre soluções de baixa complexidade com erro elevado e alta complexidade com erro baixo. É possível que, entre estas soluções, existirá uma com a complexidade efetiva adequada. Note-se na Figura 4.2 que, partindo-se da solução de norma mínima e de maior erro até a solução de norma máxima e de menor erro, poderá haver uma solução que não apresenta erros muito elevados para os padrões de treinamento nem norma muito alta, indicando que a rede não está super-ajustada aos dados. Em outras palavras, entre os extremos de norma máxima e norma mínima, poderá existir uma solução que possui a norma adequada, com um erro menor possível para a mesma.

A Figura 4.3 mostra o conjunto Pareto-ótimo. Neste, uma possível solução que equilibra o erro e a norma é mostrada, sendo esta chamada de ótima. Esta solução é denotada por ($\diamond$).

É importante notar que o conjunto Pareto-ótimo contém soluções sub-ajustadas e soluções super-ajustadas. As soluções denotadas por ($\bullet$) são as sub-ajustadas e apresentam erros grandes para os padrões de treinamento devido à falta de complexidade das mesmas. Para estas soluções, o erro predominante é devido ao efeito da polarização. As soluções denotadas por ($\circ$) são as super-ajustadas e apresentam erros pequenos para os padrões de

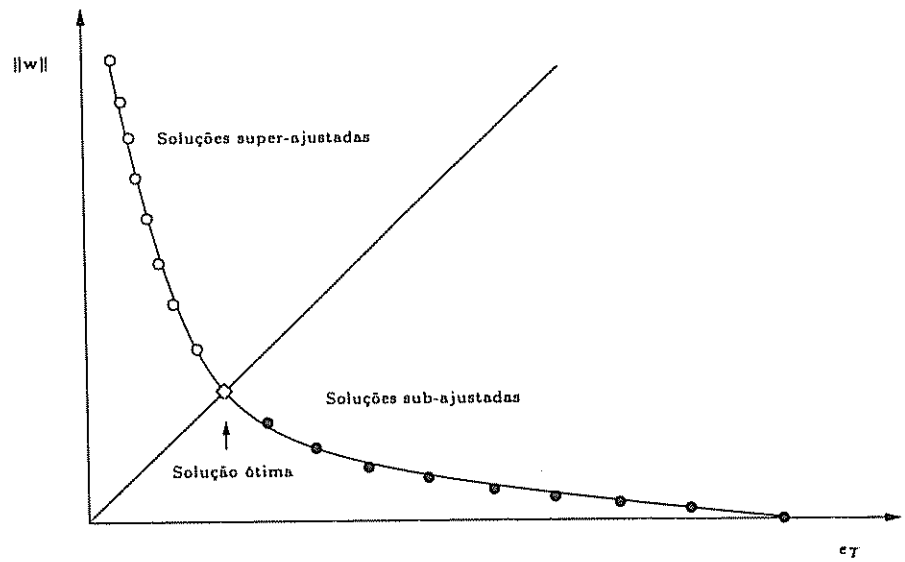


Figura 4.3: Localização da solução ótima $\mathbf{w}^*$ (o), das soluções super-ajustadas (o) e das soluções sub-ajustadas (•) aos dados no espaço dos objetivos.

treinamento mas erros maiores para os padrões de validação devido ao efeito chamado de *overfitting*. Para estas soluções, a variância é elevada.

O primeiro passo quando se usa otimização multi-objetivo é obter o conjunto Pareto-ótimo $\mathcal{W}^*$ [Chankong and Haimes, 1983], o qual é constituído de soluções eficientes $\mathbf{w}^*$. O passo seguinte é a seleção da melhor solução entre as soluções pertencentes ao conjunto Pareto-ótimo. Este processo é chamado de decisor e será abordado com detalhe no Capítulo 5. Na literatura podem-se encontrar muitos algoritmos para tratar problemas multi-objetivos [Chankong and Haimes, 1983]. Neste trabalho são utilizados dois métodos sendo o primeiro deles proposto por [Takahashi et al., 1997] chamado método de relaxação e o segundo chamado de ε-restrito [Duckstein, 1984] conhecido na literatura.

A seguir é feita uma descrição detalhada dos algoritmos implementados para resolver o problema de otimização multi-objetivo para o treinamento de redes, onde as duas funções de custo, erro e norma, são consideradas para que as soluções com boa capacidade de generalização sejam alcançadas.

4.3 Implementação através do algoritmo de Relaxação

O algoritmo multi-objetivo implementado através do método de relaxação [Takahashi et al., 1997] resolve o problema de otimização em duas fases. Na primeira fase, uma RNA do tipo MLP com um número relativamente alto de parâmetros é treinada fazendo-se uso de um algoritmo de treinamento como o *Backpropagation* [Rumelhart et al., 1986b]. Na segunda fase, a RNA treinada é passada como parâmetro para o algoritmo que tem como função encontrar um determinado número de soluções pertencentes ao conjunto Pareto-ótimo. Nesta fase, o problema multi-objetivo é reescrito

como mono-objetivo e um algoritmo de otimização restrita deve ser utilizado. Neste trabalho, o problema mono-objetivo é resolvido com o método elipsoidal [Shor, 1977].

A RNA superdimensionada a ser treinada inicialmente representa um custo adicional ao algoritmo MOBJ, mas não representa uma tarefa difícil. É necessário que esta rede tenha a complexidade acima da necessária para a tarefa em questão e que seja treinada até que o menor erro possível seja alcançado. Estas condições fazem com esta rede fique super-ajustada aos dados de treinamento, apresentando *overfitting*. Vale ressaltar que conseguir uma rede com excesso de complexidade é uma tarefa bem mais simples que conseguir uma com a complexidade adequada. Esta rede pode ser treinada fazendo-se uso do algoritmo *Backpropagation* ou qualquer uma de suas variações.

A partir desta rede inicial, o algoritmo MOBJ obtém um número ζ de novas redes, número este a ser fornecido pelo usuário, todas com complexidade inferior à complexidade da rede inicial. Cada uma destas redes é uma solução não-dominada constituindo o conjunto Pareto-ótimo.

A formulação do método de relaxação aplicado ao problema de treinamento de RNAs é feita a seguir:

- $\mathbf{u}^* \in \mathbb{R}^m$ é o vetor de objetivos correspondente à solução utópica [Chankong and Haimes, 1983] do problema;
- ϕ_i^* é o valor ótimo para o i-ésimo objetivo;
- ϕ_i é o valor do i-ésimo objetivo em um ponto qualquer;
- $\mathbf{f}_i^* \in \mathbb{R}^m$; $i = 1, \ldots, m$ é o vetor formado pelo ótimo de cada objetivo individual i e os valores correspondentes para os demais objetivos.
- $\mathcal{C}$ é o cone gerado pelos vetores $(\mathbf{f}_i^* - \mathbf{u}^*)$, com origem em $\mathbf{u}^*$;
- $\mathbf{v}_k \in \mathcal{C}$ é um vetor construído de acordo com a Equação 4.1, sendo este uma combinação convexa entre os dois vetores formados pelo ótimo de cada objetivo individual i e os valores correspondentes para os demais objetivos. A Figura 4.4 ilustra o processo de formação do vetor $\mathbf{v}_k$ para diferentes valores de γ_k.

$$\mathbf{v}_k = \mathbf{u}^* + \gamma_k(\mathbf{f}_1^* - \mathbf{u}^*) + (1 - \gamma_k)(\mathbf{f}_2^* - \mathbf{u}^*), \quad \text{para } 0 < \gamma_k < 1 \tag{4.1}$$

Os objetivos somatório dos erros quadráticos e norma do vetor de pesos a serem otimizados são descritos pelas Equações (4.2) e (4.3) respectivamente. A saída da RNA é descrita pela Equação (4.4).

$$f_1(\mathbf{x}; \mathbf{w}) = \frac{1}{N_T} \sum_{j=1}^{N_T} (d_j - y_j)^2 \tag{4.2}$$

$$f_2(\mathbf{w}) = \|\mathbf{w}\| \tag{4.3}$$

$$y_j = f(\mathbf{x}_j; \mathbf{w}) \tag{4.4}$$

onde $\mathbf{w}$ é o vetor de pesos da RNA, N_T é o número de padrões no conjunto de treinamento, d_j e y_j são, respectivamente, a saída desejada e a saída real correspondente ao padrão j.

A formulação apresentada através das Equações (4.5), (4.6), (4.7) e (4.8) mostra como os $\mathbf{f}_i^*$ podem ser obtidos. Neste caso particular onde se tem duas funções objetivo, a formulação a seguir ilustra como se pode obter $\mathbf{f}_1^*$, $\mathbf{f}_2^*$ e $\mathbf{u}^*$.

$$\mathbf{w}_1^* = \arg\min_{\mathbf{w}} f_1 \tag{4.5}$$

$$\begin{aligned} \phi_1^* &= f_1(\mathbf{x}; \mathbf{w}_1^*), \\ \phi_2 &= f_2(\mathbf{w}_1^*) \end{aligned}$$

$$\mathbf{f}_1^* = \begin{bmatrix} \phi_1^* \\ \phi_2 \end{bmatrix} \tag{4.6}$$

$$\mathbf{w}_2^* = \arg\min_{\mathbf{w}} f_2 \tag{4.7}$$

$$\begin{aligned} \phi_1 &= f_1(\mathbf{x}; \mathbf{w}_2^*), \\ \phi_2^* &= f_2(\mathbf{w}_2^*) \end{aligned}$$

$$\mathbf{f}_2^* = \begin{bmatrix} \phi_1 \\ \phi_2^* \end{bmatrix} \tag{4.8}$$

O cálculo de $\mathbf{w}_2^*$ na Equação (4.7) é trivial, basta que os pesos sejam nulos. O cálculo de $\mathbf{w}_1^*$ na Equação (4.5) deve ser feito utilizando-se de um algoritmo de treinamento como o *backpropagation* ou uma de suas variações.

A solução utópica denotada por $\mathbf{u}^*$ é um vetor formado por dois elementos:

1. Valor mínimo para a função objetivo f_1 que é ϕ_1^*;
2. Valor mínimo para a função objetivo f_2 que é ϕ_2^*.

A Equação (4.9) mostra a solução utópica $\mathbf{u}^*$.

$$\mathbf{u}^* = \begin{bmatrix} \phi_1^* \\ \phi_2^* \end{bmatrix} \tag{4.9}$$

A Equação (4.1) mostra como são construídos os ζ vetores pertencentes ao cone $\mathcal{C}$. Cada vetor $\mathbf{v}_k$ é responsável pela geração de uma solução pertencente ao conjunto Pareto-ótimo. O número ζ de vetores $\mathbf{v}$ a ser gerado e consequentemente o número ζ de soluções não-dominadas é um parâmetro a ser fornecido pelo usuário. Tipicamente, este parâmetro assume valores entre 20 e 50 e influencia na resolução do conjunto Pareto-ótimo.

A Equação (4.1) difere da sua forma original descrita pela Equação (3.15) porque são considerados aqui apenas os objetivos erro quadrático e norma do vetor de pesos.

A Equação (4.1) resulta sempre em um vetor pertencente ao cone das soluções factíveis. Para qualquer γ_k, existe um vetor $\mathbf{v}_k$ pertencente a este cone que corresponderá a uma solução pertencente ao conjunto Pareto-ótimo. Neste trabalho, para que todo o conjunto Pareto-ótimo possa ser gerado, γ_k é inicializado com $\frac{1}{\zeta}$ e então incrementado uniformemente por $\frac{\zeta-2}{\zeta^2}$ até $1-\frac{1}{\zeta}$. A Figura 4.4 ilustra como os vetores $\mathbf{v}_k$ são gerados a partir da Equação (4.1).

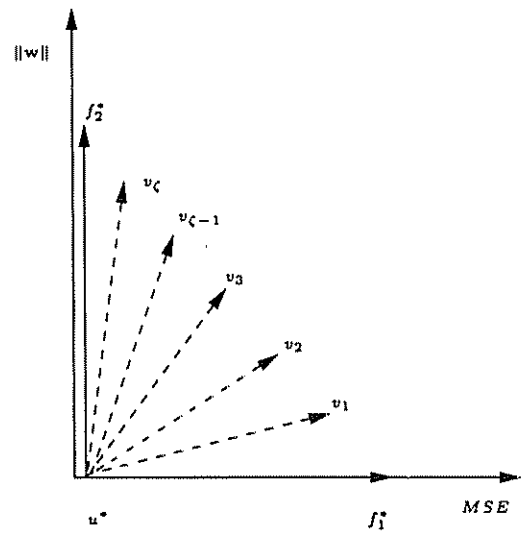


Figura 4.4: Interpretação gráfica da Equação (4.1)

Na abordagem segundo o método de relaxação, o problema multi-objetivo é redefinido em termos de um único objetivo, inserindo os objetivos descritos pelas Equações (4.2) e (4.3) como funções de restrição. O problema de otimização pode então ser resolvido por algum método restrito de otimização mono-objetivo como por exemplo, fazendo-se uso do algoritmo Elipsoidal [Shor, 1977]. O problema multi-objetivo é reescrito de forma mono-objetivo pelas Equações (4.10) e (4.11).

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \min_{\mathbf{w},\eta} \eta, \tag{4.10}$$

$$\text{Sujeito a : } \mathbf{f}_i(\mathbf{w}) \leq \mathbf{u}^* + \eta \mathbf{v}_k \tag{4.11}$$

onde η é uma variável auxiliar. Substituindo-se as Equações (4.2) e (4.3) na Equação (4.11), tem-se o problema restrito descrito pelas Equações (4.12) e (4.13).

$$\mathbf{w}^* = \arg \min_{\mathbf{w},\eta} \eta \tag{4.12}$$

$$\text{Sujeito a : } \begin{cases} g_1(\mathbf{w},\eta) = \frac{1}{N_T}\sum_{j=1}^{N_T}(d_j - y_j)^2 - \phi_1^* - \eta v_{k_1} \leq 0 \\ g_2(\mathbf{w},\eta) = \|\mathbf{w}\| - \phi_2^* - \eta v_{k_2} \leq 0 \end{cases} \tag{4.13}$$

Como pode ser observado na Figura 4.1, os dois extremos do conjunto Pareto-ótimo são duas soluções importantes. Em um extremo, a solução $\mathbf{f}_1^*$ possui erro quadrático pequeno e norma do vetor de pesos elevada, o que geralmente resulta em má capacidade de generalização (*overfitting*). No outro extremo, a solução $\mathbf{f}_2^*$ possui norma igual a zero e erro quadrático elevado, o que resultaria em *underfitting*. Uma solução bem balanceada pertencente ao conjunto Pareto-ótimo está entre estes dois extremos.

O Algoritmo (1) mostra como o método multi-objetivo para o treinamento de redes foi implementado fazendo-se uso da técnica de relaxação. Este algoritmo não contempla a primeira fase do método MOBJ que é o treinamento de uma rede superdimensionada sendo mostrada somente a parte onde as soluções Pareto-ótimas são calculadas. A listagem do programa em linguagem Matlab que implementa este método pode ser vista no Apêndice A, Seção A.3.1.

O Algoritmo (3), o qual implementa o método elipsoidal e faz a otimização mono-objetivo restrita pode ser visto no Apendice A na Seção A.3.3. A listagem do programa em Matlab também está no Apendice A na Seção A.3.4.

Após obtenção do conjunto Pareto-ótimo, um decisor deve escolher entre as soluções candidatas, a solução final. Para que esta escolha seja feita, um ou vários critérios de seleção devem ser embutidos no decisor. No capítulo 5, o critério de decisão implementado neste trabalho é abordado.

Uma outra forma possível de se obter o conjunto Pareto-ótimo é através da utilização do método ε-restrito. Neste trabalho, em algumas situações, este método apresentou melhores resultados que o método de relaxação anteriormente abordado. A seguir é feito o detalhamento da implementação do algoritmo MOBJ fazendo-se uso do método ε-restrito.

Algoritmo 1 Lançador MOBJ - Obtenção do Conjunto Pareto-ótimo - Método relaxação

Carregar conjunto de treinamento; {Vetores de Entrada e Saída}
Treinar uma RNA superdimensionada para o problema; {RNA com complexidade alta}
$\zeta \Leftarrow$ Número de soluções a serem geradas; {Soluções Pareto}
$\mathbf{w}_1^* \Leftarrow$ Vetor de pesos solução considerando Objetivo 1; {Proveniente da rede já treinada}
$\phi_1^* \Leftarrow f_1(\mathbf{w}_1^*)$ {Valor ótimo para a função f_1}
$\phi_2 \Leftarrow f_2(\mathbf{w}_1^*)$ {Valor da função f_2 no ponto $\mathbf{w}_1^*$}
$\mathbf{f}_1^* \Leftarrow \left[\begin{array}{cc} \phi_1^* & \phi_2 \end{array} \right]^T$
$\mathbf{w}_2^* \Leftarrow$ Vetor de pesos solução considerando Objetivo 2; {Solução trivial}
$\phi_1 \Leftarrow f_1(\mathbf{w}_2^*)$ {Valor da função f_1 no ponto $\mathbf{w}_2^*$}
$\phi_2^* \Leftarrow f_2(\mathbf{w}_2^*)$ {Valor ótimo para a função f_2}
$\mathbf{f}_2^* \Leftarrow \left[\begin{array}{cc} \phi_1 & \phi_2^* \end{array} \right]^T$
$\mathbf{u}^* \Leftarrow \left[\begin{array}{cc} \phi_1^* & \phi_2^* \end{array} \right]^T$; {Solução utópica}
Inicializar η; {Suficientemente grande para tornar o problema factível}
Inicializar $\mathbf{w}_0$; {Pesos ramdômicos, média zero e variância pequena}
$\gamma_{inc} \Leftarrow \frac{\zeta-2}{\zeta^2}$; {Passo para incremento do valor de γ}
$\gamma \Leftarrow \frac{1}{\zeta}$; {$\gamma$ inicial}
$C_{po} \Leftarrow 1$; {Contador de soluções Pareto}
Enquanto $C_{po} \leq \zeta$ **do**
  $\mathbf{v} \Leftarrow \mathbf{u}^* + \gamma * (\mathbf{f}_1^* - \mathbf{u}^*) + (1-\gamma) * (\mathbf{f}_2^* - \mathbf{u}^*)$; {Combinação convexa}
  $\gamma \Leftarrow \gamma + \gamma_{inc}$;
  Obter $\mathbf{w}^*$ que minimize η sujeito a: {Utiliza Algoritmo Elipsoidal}
  Restrição 1 : $g_1(\mathbf{w}, \eta) = \frac{1}{N_T} \sum_{j=1}^{N_T} (d_j - y_j)^2 - \phi_1^* - \eta v_1 \leq 0$;
  Restrição 2 : $g_2(\mathbf{w}, \eta) = \|\mathbf{w}\| - \phi_2^* - \eta v_2 \leq 0$;
  $\mathbf{w}_0 \Leftarrow \mathbf{w}^*$; {A próxima busca se iniciará no ponto de ótimo $\mathbf{w}^*$ da busca anterior}
  Guardar solução $\mathbf{w}^*$; {$\mathbf{w}^*$ é uma solução do conjunto Pareto-Ótimo}
  $C_{po} = C_{po} + 1$;
fim Enquanto

4.4 Implementação através do método ε-restrito

O algoritmo multi-objetivo implementado através do método ε-restrito, diferentemente do método abordado na seção anterior, necessita apenas de uma fase para que o conjunto Pareto-ótimo possa ser calculado, não sendo necessário o custo adicional do treinamento de uma rede superdimensionada. Através do método ε-restrito, o problema de otimização multi-objetivo é também reescrito na forma mono-objetivo sendo resolvido pelo algoritmo elipsoidal [Shor, 1977].

Para o caso particular deste trabalho onde somente dois objetivos são necessários, o objetivo somatório dos erros quadráticos é otimizado sendo o objetivo norma do vetor de pesos considerado como restrição. Logo, um problema restrito de otimização mono-objetivo é resolvido para diferentes valores de ε. A variação paramétrica de ε permite a geração de todo o conjunto Pareto-ótimo.

O algoritmo MOBJ obtém um determinado número ζ de soluções a ser informado pelo usuário com diferentes complexidades. Cada uma destas soluções é uma solução não-dominada constituindo o conjunto Pareto-ótimo. A formulação do método ε-restrito aplicado ao problema de treinamento de RNAs é feita a seguir:

$$\begin{aligned} &\min_{\mathbf{w}\in\mathcal{W}} \; f_1(\mathbf{x};\mathbf{w}) \\ &\text{sujeito a}: \; f_2(\mathbf{w}) \leq \varepsilon \end{aligned} \tag{4.14}$$

Através do problema (Pε) descrito na Equação 4.14 é possível gerar $\mathcal{W}^*$ completamente, mesmo que o problema seja não-convexo. Como

$$f_1(\mathbf{x};\mathbf{w}) = \frac{1}{N_T}\sum_{j=1}^{N_T}(d_j - f(\mathbf{x}_j;\mathbf{w}))^2$$

e

$$f_2(\mathbf{w}) = \|\mathbf{w}\|,$$

a Equação (4.14) fica

$$\begin{aligned} &\min_{\mathbf{w}\in\mathcal{W}} \; \tfrac{1}{N_T}\sum_{j=1}^{N_T}(d_j - f(\mathbf{x}_j;\mathbf{w}))^2 \\ &\text{sujeito a}: \; \|\mathbf{w}\| \leq \varepsilon \end{aligned} \tag{4.15}$$

onde $\mathbf{w}$ é o vetor de pesos da RNA, N_T é o número de padrões no conjunto de treinamento, x_j e d_j são, respectivamente, a j-ésima entrada e a saída do conjunto de treinamento.

Para o algoritmo MOBJ via Pε calcular um número ζ de soluções, é necessário que o usuário informe este número e como deverá ser feita a variação paramétrica de ε. Além disso, o número de neurônios na camada escondida deve ser informado. Tipicamente, o parâmetro ζ é um número entre 20 e 30, o qual determina quantas soluções não-dominadas serão geradas. O número de neurônios a ser informado também não representa uma tarefa difícil uma vez que o usuário deve se preocupar exclusivamente em superdimensionar a rede. A variação paramétrica de ε é também feita de uma forma muito simples. Iniciando-se de $\delta\varepsilon$ a ser informado, cada solução estará distante $\delta\varepsilon$ uma da outra porque a cada problema mono-objetivo escrito segundo a Equação 4.15 o parâmetro ε é igual ao ε anterior mais o $\delta\varepsilon$.

A Figura 4.5 ilustra como pode ser obtida cada solução pertencente ao conjunto Pareto-ótimo através da variação paramétrica de ε.

A escolha de $\delta\varepsilon$ é feita de modo que não haja um salto muito grande de norma entre as soluções obtidas. Tipicamente, o valor deste parâmetro é um número entre 0.5 e 1.

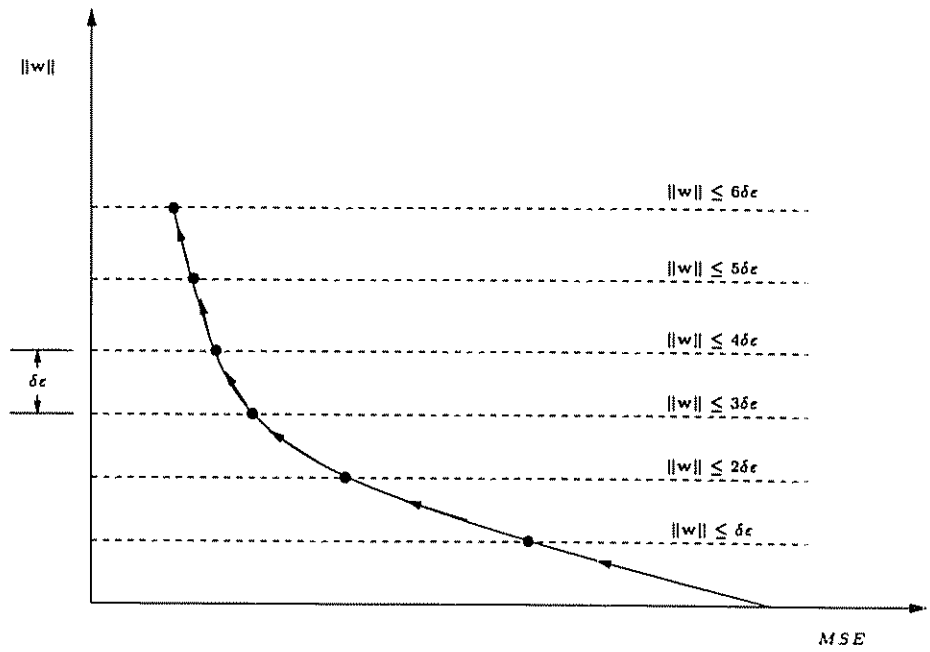


Figura 4.5: Geração de soluções via Pε

Note-se que para um $\delta\varepsilon = 0.5$ e $\zeta = 20$, a máxima norma que poderá ser alcançada será igual a 10. Se no processo de decisão todas estas soluções não apresentarem resultados satisfatórios, significa que a norma correta ainda é maior e um ajuste em $\delta\varepsilon$ para 1 mantendo-se $\zeta = 20$, por exemplo, faz com que a maior norma obtida agora seja de 20 ou por outro lado, mantém-se $\delta\varepsilon = 0.5$ e ajustando-se $\zeta = 40$, a maior norma também será 20. O custo computacional para este último caso será mais elevado uma vez que haverá a necessidade de se calcular 40 soluções não-dominadas.

O método Pε, apesar de sua simplicidade, é bastante poderoso e ainda possibilita que somente parte do conjunto Pareto-ótimo possa ser gerado. Como discutido anteriormente, se $\delta\varepsilon = 0.5$ e $\zeta = 20$, a solução de maior norma terá norma igual 10 e se no processo de decisão uma solução com norma inferior a 10 é escolhida, significa que não há a necessidade de cálculo de soluções com normas superiores a 10 pois estas soluções com certeza estariam super-ajustadas aos dados e seriam descartadas no processo de decisão.

O Algoritmo 2 mostra como o método multi-objetivo para treinamento de redes foi implementado fazendo-se uso do método ε-restrito. A listagem do programa em Matlab que implementa este método pode ser vista no Apêndice A, Seção A.3.2.

Para o algoritmo MOBJ implementado a partir do método Pε foram desenvolvidas interfaces para demonstração. Através destas, alguns problemas de classificação de duas classes e de regressão podem ser resolvidos. Nas Seções A.1 e A.2 do Apêndice A estas interfaces são mostradas.

Algoritmo 2 Lançador MOBJ - Obtenção do Conjunto Pareto-ótimo - Método Pε

Carregar conjunto de treinamento; {Vetores de Entrada e Saída}
$N_N \Leftarrow$ Número de neurônios da camada intermediária;
$\zeta \Leftarrow$ Número de soluções a serem geradas; {Soluções Pareto}
$\delta\varepsilon \Leftarrow$ Diferença entre as normas das soluções obtidas;
Inicializar ε; $\{\varepsilon = \delta\varepsilon\}$
Inicializar $\mathbf{w}_0$; {Pesos ramdômicos, média zero e variância pequena}
$C_{po} \Leftarrow 1$; {Contador de soluções Pareto}
Enquanto $C_{po} \leq \zeta$ **do**
  Obter $\mathbf{w}^*$ que minimize $f_1(\mathbf{x};\mathbf{w})$ sujeito a: {Utiliza Algoritmo Elipsoidal}
  Restrição 1 : $g_1(\mathbf{w}) = \|\mathbf{w}\| \leq \varepsilon$;
  $\mathbf{w}_0 \Leftarrow \mathbf{w}^*$; {A próxima busca se iniciará no ponto de ótimo $\mathbf{w}^*$ da busca anterior}
  Guardar solução $\mathbf{w}^*$; {$\mathbf{w}^*$ é uma solução do conjunto Pareto-Ótimo}
  $\varepsilon = \varepsilon + \delta\varepsilon$;
  $C_{po} = C_{po} + 1$;
fim Enquanto

4.5 Caracterização das soluções no espaço dos objetivos

O algoritmo MOBJ, seja ele implementado através do método de relaxação ou através do método ε-restrito, é capaz de gerar ζ soluções não-dominadas, as quais constituem o conjunto Pareto-ótimo. Estas soluções representam vetores de pesos no espaço das variáveis de decisão e têm como principal característica a diferença de complexidade entre as mesmas. Neste conjunto, pode-se ter, desde soluções super-ajustadas até as sub-ajustadas, além da solução que apresenta a complexidade compatível com a tarefa em questão.

As soluções Pareto-ótimas são aquelas que constituem a fronteira do Pareto e a finalidade do algoritmo MOBJ é encontrar soluções exatamente nesta fronteira. Note-se que nesta região, não se pode reduzir ainda mais a norma sem incrementar o erro ou reduzir o erro sem que haja um aumento da norma. Na região factível, pode-se reduzir ambos os objetivos além de ser possível a redução de apenas um objetivo deixando o outro constante. A Figura 4.6 ilustra as regiões não factível a fronteira do conjunto Ω, a qual constitui a região Pareto-ótima e algumas soluções ($\diamond$) pertencentes ao conjunto Ω fora da fronteira. Através do método MOBJ proposto, não se pode obter tais soluções e quanto mais distantes elas estão da fronteira, observa-se que piores são.

Os métodos atualmente conhecidos na literatura para treinamento de redes neurais artificiais não garantem por construção que a solução obtida através dos mesmos esteja sobre a região Pareto-ótima ou pelo menos nas vizinhanças da mesma. Este resultado é forte mas não significa que qualquer um destes métodos possa alcançar uma solução com alta capacidade de generalização, o que dependerá apenas de um conjunto de parâmetros de treinamento adequado. Por exemplo, considere os algoritmos de treinamento *Back-propagation*, *Weight Decay*, *Optimal Brain Damage*, *Cross-Validation*, *Early Stopping* e *Support Vector Machine*. Algumas discussões sobre como os parâmetros de cada algoritmo

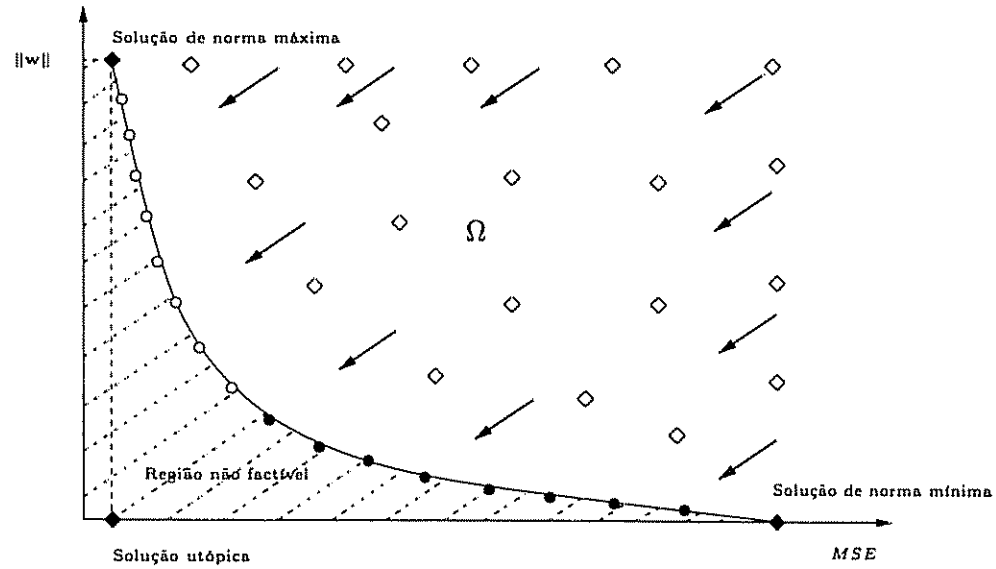


Figura 4.6: Caracterização das soluções no espaço dos objetivos

pode influenciar na suas respectivas capacidades de generalização e, consequentemente, como as soluções destes algoritmos se comportam no espaço dos objetivos são feitas a seguir:

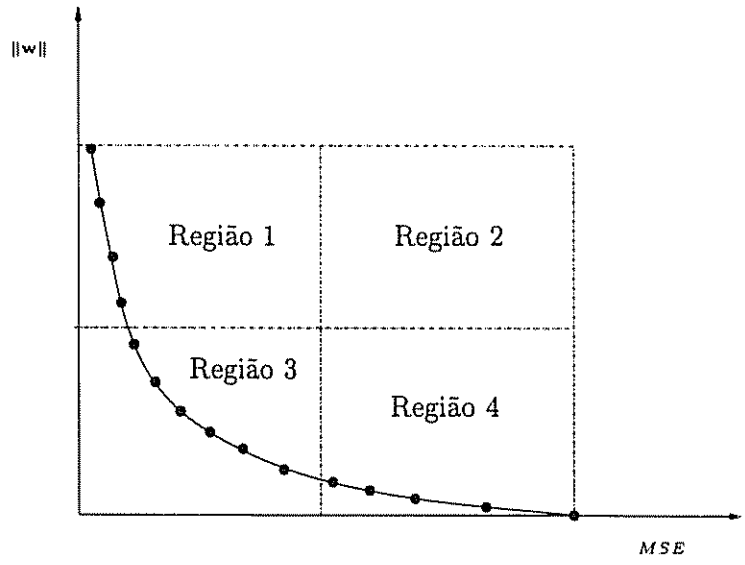


Figura 4.7: Localização de soluções no espaço dos objetivos

1. *Backpropagation* e suas variações:
   A capacidade de generalização de uma solução *Backpropagation* pode ser influenciada pela topologia da rede e pelo valor do erro para os padrões de treinamento desejado. A tendência deste algoritmo em casos onde a rede é pouco complexa, é de se prender em mínimos locais e para os casos onde a rede possui complexidade além do necessário, a tendência é de se obter uma solução com norma elevada e erro pequeno. A Figura 4.7 mostra a posição das soluções no espaço dos objetivos

e para o primeiro caso, as soluções seriam mapeadas na região de norma baixa e erro grande (região 4), caracterizando soluções sub-ajustadas ou polarizadas. No segundo caso, a soluções seriam mapeadas na região onde a norma é elevada e o erro é pequeno (região 1), caracterizando soluções super-ajustadas. Para resumir, conseguir um solução com boa capacidade de generalização com um algoritmo deste tipo não é uma tarefa trivial.

2. *Weight Decay*

   A capacidade de generalização de uma solução *Weight Decay* pode ser influenciada pelo valor da constante de queda dos pesos. Valores altos para este parâmetro levam a uma grande penalização da norma. A Figura 4.7 mostra a posição das soluções *Weight Decay* no espaço dos objetivos e para o casos onde o valor da constante de queda é grande, as soluções seriam mapeadas na região de norma baixa e erro grande (região 4), caracterizando soluções sub-ajustadas ou polarizadas. Nos casos onde as constantes de queda são pequenas, as soluções seriam mapeadas na região onde a norma é elevada e o erro é pequeno (região 1), caracterizando soluções super-ajustadas. Um bom ajuste no valor desta constante pode fazer com as soluções fiquem na região 3, onde os efeitos da polarização e da variância são melhor equilibrados. O custo para se conseguir uma solução na região 3 é elevado uma vez que, geralmente, várias tentativas de ajuste da constante são feitas.

3. *Optimal Brain Damage*

   A capacidade de generalização de uma solução *Optimal Brain Damage* pode ser influenciada pelo número de neurônios que se permite eliminar. Se poucos pesos são eliminados, a segunda fase do treinamento pode fazer com que a solução se super-ajuste ao conjunto de treinamento. Portanto o número de épocas para a segunda fase do treinamento também é importante. Logo, o número elevado de épocas em conjunto com o pequeno número de pesos eliminados pode fazer com a solução seja mapeada na região 1 da Figura 4.7. Se muitas conexões são eliminadas, a rede pode ficar pouco complexa para que seja possível realizar adequadamente o aprendizado. Neste caso, a solução seria mapeada na região 4. Para que uma boa solução possa ser alcançada, é necessário um conjunto de parâmetros adequado. Neste caso, a solução seria mapeada na região 3. O custo de uma boa solução seria em função do número de tentativas a serem feitas para se conseguir um bom conjunto de parâmetros de treinamento.

4. *Early Stopping*

   A capacidade de generalização de uma solução *Early Stopping* pode ser influenciada pelo número de padrões, pela forma na qual são constituídos os conjuntos de treinamento e validação, pela topologia da rede e pelo momento exato de se interromper

o treinamento. A Figura 4.7 mostra as posições das soluções para cada situação. Se a rede é superdimensionada e o treinamento é interrompido depois do momento exato, leva a solução para a região 1 e se for interrompido antes do momento exato, a solução é mapeada na região 2. Se a rede não é muito complexa e o treinamento é interrompido antes do momento correto, a solução é mapeada na região 4. Para que a solução seja mapeada na região 3, é preciso que a rede não seja muito complexa e que o treinamento seja interrompido em um momento adequado e que o conjunto de dados seja suficientemente grande.

5. *Cross-Validation*

   A capacidade de generalização de uma solução *Cross-Validation* pode ser influenciada pelo número de padrões para treinamento, número de divisões a serem feitas no conjunto de dados e pela topologia da rede. Neste método de treinamento são treinadas muitas redes e consequentemente haverá redes super-ajustadas, sub-ajustadas e possivelmente redes com um ajuste adequado ou próximo do adequado. A existência de uma rede na região 3 da Figura 4.7 é condicionada aos fatores que influenciam no processo de treinamento. Tipicamente, haverá redes nas regiões 1, 4 e 3. A região 2 caracteriza redes não treinadas. Contudo, dependendo da complexidade da rede, pode-se ter soluções apenas na região 4, e portanto, sub-ajustadas ou ainda apenas na região 1 e portanto, super-ajustadas. Para se conseguir melhores resultados, é bom que a rede não tenha complexidade muito acima da necessária. Isto faz com que as soluções tenham uma variância bastante elevada sendo necessária um quantidade de dados maior para diminuir a variância.

6. *Support Vector Machines*

   A capacidade de generalização de uma SVM pode ser influenciada pelo valor máximo dos multiplicadores de Lagrange além dos parâmetros do *kernel* adotado. Considera-se aqui somente o primeiro parâmetro, os parâmetros do *kernel* são considerados ideais. Valores altos para os multiplicadores de Lagrange conduz à soluções de normas mais elevadas e neste caso, as soluções seriam mapeadas na região de norma alta e erro pequeno, região 1, caracterizando soluções super-ajustadas. Nos casos onde o valor máximo para os multiplicadores de Lagrange é pequeno, as soluções seriam mapeadas na região onde a norma é baixa e o erro é elevado, região 4, caracterizando soluções polarizadas. Um bom ajuste no valor deste parâmetro pode fazer com as soluções fiquem na região 3. O custo de uma boa solução estaria relacionado com o número de tentativas de ajustes dos parâmetros de treinamento.

4.6 Formas do conjunto Pareto-ótimo

O conjunto Pareto-ótimo não apresenta sempre o formato apresentado nas Figuras 4.6 e 4.7. Algumas mudanças em sua forma podem ocorrer em virtude das não linearidades de cada problema de aprendizagem. Na Figura 4.8 é mostrada a forma mais comum do conjunto Pareto-ótimo. Neste caso, tanto a função de erro quanto a função norma do vetor de pesos apresentam valores baixos para suas derivadas.

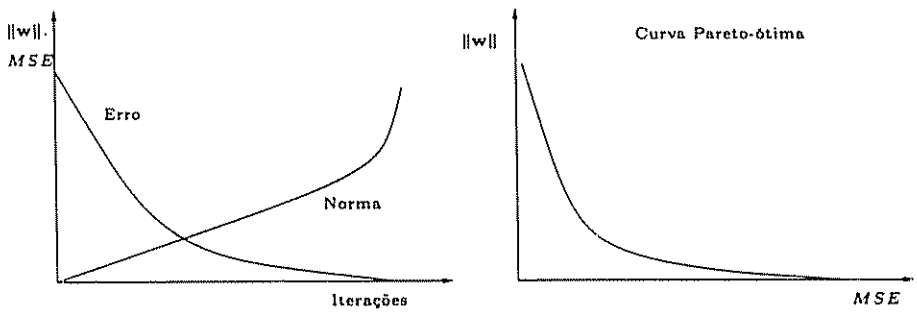


Figura 4.8: Possível forma do conjunto Pareto-ótimo - Sem variações bruscas no erro e na norma

Na Figura 4.9 é mostrada uma outra possível forma do conjunto Pareto-ótimo. Note-se que neste caso ocorre uma variação brusca no erro para uma variação pequena da norma. Este tipo de forma é observado em problemas onde é necessária uma complexidade relativamente alta do modelo para se resolver o problema de aprendizagem.

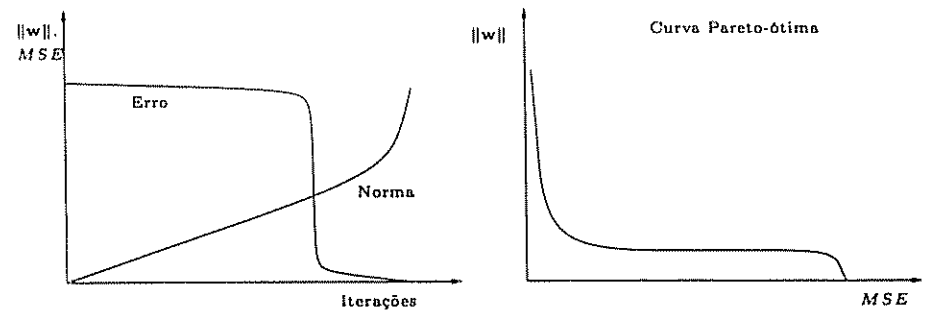


Figura 4.9: Possível forma do conjunto Pareto-ótimo - Variação brusca no erro

Na Figura 4.10 é mostrada uma outra forma possível, porém pouco comum. Neste caso, ocorre uma variação brusca da norma em um dado momento para uma variação pequena do erro. Para resumir, qualquer que seja a forma do conjunto Pareto-ótimo ou da fronteira da região factível, o algoritmo proposto é capaz de encontrar soluções sobre esta fronteira. Geralmente, as não linearidades dificultam a escolha de um conjunto adequado de parâmetros de treinamento a ser feita pelo usuário quando métodos convencionais de aprendizagem são empregados.

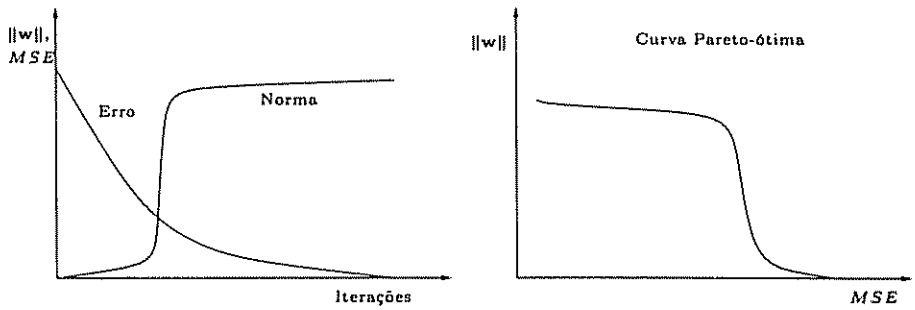


Figura 4.10: Possível forma do conjunto Pareto-ótimo - variação brusca na norma

4.7 Eficiência computacional dos algoritmos

O método MOBJ de treinamento foi implementado fazendo-se uso de duas técnicas diferentes, sendo a primeira delas a técnica de relaxação e a segunda, a técnica ε-restrito. Em ambos os algoritmos, o usuário deve informar o número ζ de soluções Pareto-ótimas a serem geradas e o custo para se obter estas soluções em qualquer um dos casos é o mesmo, mas, por motivos que serão explicados a seguir, o método MOBJ implementado com a técnica ε-restrito apresentou melhor desempenho. Outro ponto importante é que não pode ser feita uma análise comparativa de tempo de processamento em relação aos demais métodos mencionados neste trabalho, porque com exceção do método *Cross-Validation*, todos os outros geram apenas uma única solução e o algoritmo MOBJ gera ζ soluções, o que faz com que o algoritmo MOBJ tenha um maior custo que os demais, se considerado apenas tempo de processamento. Se for considerado também o tempo que o usuário gasta para ajustar os parâmetros de treinamento dos demais métodos para se obter uma boa solução além do conhecimento que o mesmo deve ter sobre os métodos, o algoritmo MOBJ se mostra superior, uma vez que, pouco conhecimento é necessário para se obter os parâmetros exigidos e, o processo de tentativa e erro não é utilizado.

4.7.1 Algoritmo de Relaxação

Através deste algoritmo o problema multi-objetivo é reescrito na forma mono-objetivo e um problema de otimização restrita deve ser resolvido. Para isto, foi utilizado o algoritmo elipsoidal. Para o algoritmo de relaxação empregado neste trabalho, tem-se uma função a ser otimizada e duas restrições. O algoritmo elipsoidal necessita do gradiente da função objetivo e das duas restrições e, além disso, necessita de um teste para verificação se existe restrição violada e qual é esta. Não há a necessidade de inversão de matrizes nem de cálculo de matriz Hessiana. A limitação do algoritmo elipsoidal existe no que diz respeito ao tamanho do vetor de variáveis de decisão, ou seja, ao número de pesos da rede. Para redes muito grandes, 5000 pesos por exemplo, é necessário uma matriz 5000×5000 e a necessidade da utilização de memória aumenta.

O cálculo dos gradientes é um ponto chave no que diz respeito ao desempenho do método proposto. Inicialmente, os gradientes estavam sendo calculados de forma numérica, com a utilização de algumas estruturas de repetição. Isto provocava uma grande queda no desempenho. Depois que se passou a fazer o cálculo de forma analítica, o desempenho como um todo foi melhorado.

O método MOBJ implementado através do método de relaxação necessita da existência de uma rede superdimensionada previamente treinada. Isto demanda um custo computacional adicional e além disso, o usuário deve ter conhecimento de algum método de treinamento para efetuar a tarefa.

4.7.2 Algoritmo ε-Restrito

Através deste algoritmo o problema multi-objetivo também é reescrito na forma mono-objetivo e um problema de otimização restrita deve ser resolvido. Para isto, faz-se o algoritmo elipsoidal. Para o problema ε-restrito empregado neste trabalho, tem-se uma função a ser otimizada e uma única restrição. O algoritmo elipsoidal necessita do gradiente da função objetivo e da única restrição. Com este algoritmo, necessita-se apenas de um teste para verificar se a restrição está violada. Para cada solução MOBJ gerada, a rotina de cálculo dos gradientes é executada muitas vezes pelo algoritmo elipsoidal, da mesma forma que na abordagem via técnica de relaxação. A limitação continua sendo o número de pesos da rede em caso de pouca memória disponível.

O cálculo dos gradientes continua sendo um ponto chave no que diz respeito ao desempenho do método proposto. A vantagem de se utilizar o problema ε-restrito é que deixa de ser necessária a rede treinada previamente, reduzindo este custo adicional.

4.8 Conclusões do Capítulo

Neste capítulo, o método de otimização multi-objetivo para treinamento de RNAs visando a otimização da capacidade de generalização foi abordado. Através deste método, é possível gerar soluções Pareto-ótimas para o problema de treinamento de redes neurais. Duas técnicas foram utilizadas para implementação do método MOBJ. A primeira delas é denominado de técnica de relaxação e a segunda, de problema ε-restrito ou Pε. Foi feita também uma análise de qualidade de soluções em relação ao posicionamento das mesmas no espaço dos objetivos e ainda demonstradas formas possíveis do conjunto Pareto-ótimo naquele espaço. Ainda neste capítulo, a eficiência computacional do algoritmo proposto é discutida.

Neste capítulo procurou-se mostrar como as soluções Pareto-ótimas são geradas mas, uma fase importante ainda esta por vir, abordando o processo de escolha da melhor solução ou da solução com máxima capacidade de generalização. Este processo é chamado

de decisor e é apresentado no Capítulo seguinte.

Capítulo 5

O Decisor

No capítulo anterior, os métodos para obtenção do conjunto Pareto-ótimo $\mathcal{W}^*$ foram abordados. Esta foi a primeira fase do método MOBJ proposto e, como resultado, tem-se um número ζ de redes treinadas e algumas destas são sub-ajustadas, outras super-ajustadas e também redes com ajuste adequado. A próxima fase é escolher a solução ótima $\mathbf{w}^* \in \mathcal{W}^*$ como solução final. Esta solução deverá possuir um ajuste adequado à função geradora $f_g(\mathbf{x})$, o que resulta em respostas satisfatórias mediante padrões desconhecidos, ou seja, a solução final deverá possuir alta capacidade de generalização. Este processo de escolha da melhor solução é chamado de decisor e é abordado neste capítulo.

O decisor implementado neste trabalho é baseado no erro de validação para tomada de decisão. Um conjunto de validação $\mathcal{T}_V = \{\mathbf{x}_i, \mathbf{d}_i + \boldsymbol{\xi}_i\}_{i=1}^{N_T}$ é apresentado a todas as redes calculadas na primeira fase, que são as soluções Pareto-ótimas e a que apresentar o menor erro para os padrões de validação é escolhida como solução final. A eficiência do processo de decisão é provado através de um teorema o qual mostra que no conjunto Pareto-ótimo existe uma solução que faz o ajuste adequado aos dados relativos à uma determinada tarefa de aprendizagem e que esta solução pode ser selecionada utilizando um conjunto de validação.

As amostras que constituem tanto o conjunto de treinamento como o de validação são geralmente contaminadas com um termo de ruído, denotado aqui por $\boldsymbol{\xi}$. Para qualquer efeito, este ruído é considerado normalmente distribuído com média zero e variância σ^2.

5.1 Considerações preliminares

É importante dizer que as soluções Pareto-ótimas no espaço das variáveis de otimização não são únicas, ou seja, soluções diferentes podem mapear um mesmo ponto no espaço dos objetivos. A seguir são feitas algumas demonstrações para explicar esta afirmação.

5.1.1 Multiplicidade de soluções

Como o conjunto Pareto-ótimo é constituído de soluções fortemente não-dominadas, todas as soluções $\mathbf{w} \in \mathcal{W}^*$ possuem normas e erros distintos. Neste conjunto, não se tem soluções de mesma norma e erro diferentes ou normas diferentes e mesmo erro. Cada uma destas soluções com erro e norma correspondente não é uma solução única, ou seja, várias soluções no espaço das variáveis de otimização podem mapear o mesmo ponto no espaço dos objetivos.

Para demonstrar esta afirmação, um único nodo com saída linear, duas entradas e termo de polarização igual a zero é tomada. A saída desta rede é dada pela Equação (5.1).

$$\mathbf{y} = \mathbf{w}^T\mathbf{x} \tag{5.1}$$

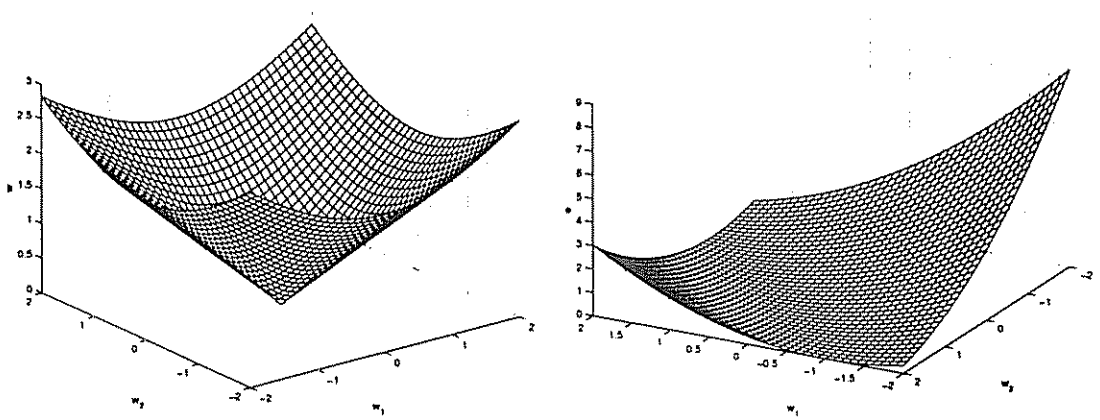
O erro em relação ao conjunto de treinamento é dado por:

$$e = \sum_{i=1}^{N_T} (d_i - y_i)^2$$

A norma do vetor de pesos é dada por:

$$\mathcal{N} = (w_1^2 + w_2^2)^{1/2}$$

Os comportamentos da norma e do erro em função dos pesos são mostrados nas Figuras 5.1(a) e 5.1(b).



(a) Comportamento da norma em função de w_1 e w_2

(b) Comportamento do erro em função de w_1 e w_2

Figura 5.1: Norma e erro em função dos pesos

A Figura 5.2 mostra as curvas de nível para a função norma e para a função erro. As linhas circulares são as curvas de nível correspondente à função norma e as retilíneas são as curvas correspondentes à função erro. Note-se que existem dois pontos de intersecção entre uma determinada curva de nível da função erro e outra da função norma. Isto significa que dois valores distintos para o vetor de pesos podem resultar em mesmos valores para as funções erro e norma. Para casos mais complexos onde o vetor de pesos é multi dimensional, existe uma grande variedade de soluções as quais mapeiam um único ponto no espaço dos objetivos.

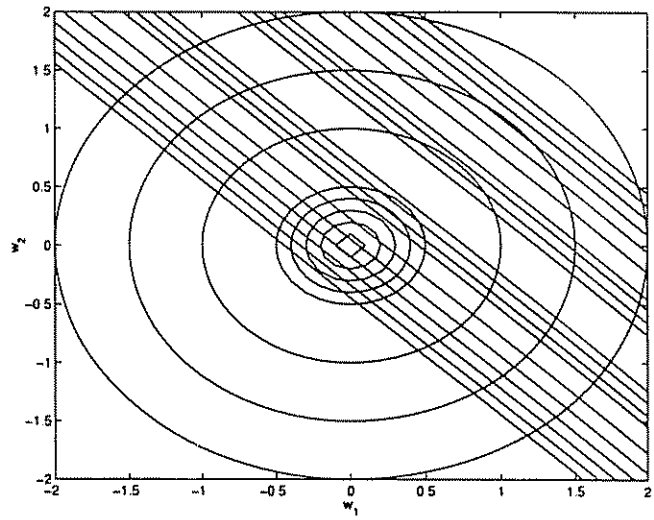


Figura 5.2: Norma e erro em função dos pesos - curvas de nível

Note-se que, apesar de a possibilidade de duas ou mais redes distintas serem mapeadas sobre o mesmo ponto no espaço dos objetivos, estas podem corresponder a um mesmo funcional ou a funcionais diferentes. No primeiro caso, se as redes correspondem a um mesmo funcional não importa qual delas seja gerada pelo algoritmo MOBJ mas, no segundo caso, se as redes representam funcionais diferentes, a geração de uma ou outra rede pode influenciar na qualidade da solução gerada. Porém, partindo da premissa de que o erro é mínimo, este fator não influência a qualidade da solução gerada. A demonstração a seguir corrobora esta afirmativa.

5.1.2 Redes que apresentam mesmo erro e norma podem representar funcionais diferentes

Considere um problema de regressão construído da seguinte forma: Um conjunto de dados com 10 padrões foi gerado a partir da Equação (5.2).

$$d(x) = 2; \tag{5.2}$$

Uma RNA $1 \times 2 \times 1$ foi treinada com funções de ativação tangente hiperbólica na camada intermediária, linear na saída e $MSE \leq 0.001$. O seguinte conjunto de pesos foi obtido:

$$\begin{aligned} \mathbf{w}_1 &= [-0.6764 \;\; 0.7170]^T \\ \mathbf{w}_2 &= [0.1746 \;\; 0.1729] \\ \mathbf{b}_1 &= [0.5664 \;\; -0.6426]^T \\ b_2 &= 1.9983 \end{aligned}$$

A partir dos pesos acima, um outro conjunto foi criado invertendo-se os pesos $\mathbf{w}_2$ e $\mathbf{b}_1$, ficando assim:

$$\begin{aligned} \mathbf{w}_1 &= [-0.6764 \;\; 0.7170]^T \\ \mathbf{w}_2 &= [-0.1746 \;\; -0.1729] \\ \mathbf{b}_1 &= [-0.5664 \;\; 0.6426]^T \\ b2 &= 1.9983 \end{aligned}$$

Tomando estes dois conjuntos de pesos e avaliando a norma considerando também os termos de polarização obteve-se:

$$\mathcal{N}_1 = \|\mathcal{N}_{conjunto_1}\| = 2.3998$$

$$\mathcal{N}_2 = \|\mathcal{N}_{conjunto_2}\| = 2.3998$$

Os erros em relação aos dados desejados são:

$$MSE_1 = \|MSE_{conjunto_1}\| = 4.9747 \times 10^{-05}$$

$$MSE_2 = \|MSE_{conjunto_2}\| = 4.9747 \times 10^{-05}$$

Apesar de as normas serem as mesmas e os erros serem os mesmos, os funcionais descritos pelas RNAs são diferentes. A Figura 5.3 ilustra estes dois funcionais. É importante notar a escala do eixo das ordenadas .

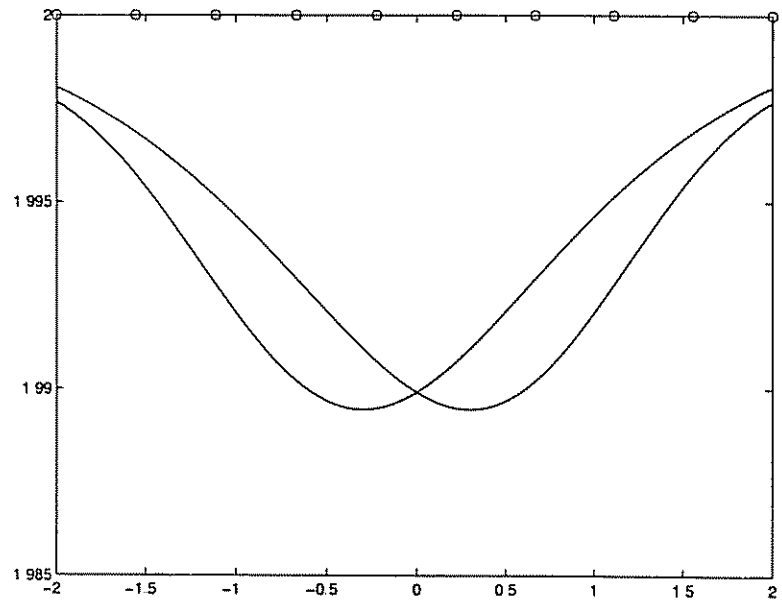


Figura 5.3: Funcionais diferentes com mesmo erro e norma

Como conclusão direta da demonstração, pode-se ter RNAs com mesmo erro e norma representando diferentes funcionais. Esta conclusão faz surgir o conceito de equivalência entre os funcionais. Por exemplo, apesar de os funcionais acima serem diferentes, pode-se dizer que, praticamente, não há diferença entre os mesmos, pois o erro de cada um em relação ao conjunto de treinamento é menor ou igual a um determinado valor mínimo. Desta forma, é dito que estes funcionais são equivalentes. Levando-se em consideração que o Pareto é a região onde o erro é mínimo para cada norma, todos os diferentes funcionais de mesma norma e de mesmo erro se equivalem. Estes conceitos de equivalência entre redes e entre funcionais são importantes e portanto, são formalizados a seguir.

Definição 1: Redes equivalentes.

Considere duas redes descritas pelos funcionais $f(\mathbf{x};\mathbf{w}_1)$ e $f(\mathbf{x};\mathbf{w}_2)$ respectivamente e um conjunto $\Omega = \{\delta, \epsilon_1, \epsilon_2\}$ de parâmetros. Estas redes são ditas Ω-equivalentes se $f(\mathbf{x};\mathbf{w}_1) \equiv f(\mathbf{x};\mathbf{w}_2)$. As condições necessárias para a equivalência entre os funcionais são dadas pela Equação (5.3).

$$\begin{cases} |\|\mathbf{w}_1\| - \|\mathbf{w}_2\|| < \delta \\ e_1,\ e_2 < \epsilon_1 \\ |e_1 - e_2| < \epsilon_2 \end{cases} \tag{5.3}$$

Onde

$$e_r = \sum_{i=1}^{N} [d(\mathbf{x}_i) - f(\mathbf{x}_i; \mathbf{w}_r)]^2$$

$$\|\mathbf{w}_r\| = \sqrt{w_{r_1}^2 + w_{r_2}^2 + \cdots + w_{r_n}^2}$$

■

A Figura 5.4 ilustra o lugar geométrico do espaço dos objetivos onde os funcionais descritos pelas redes são equivalentes.

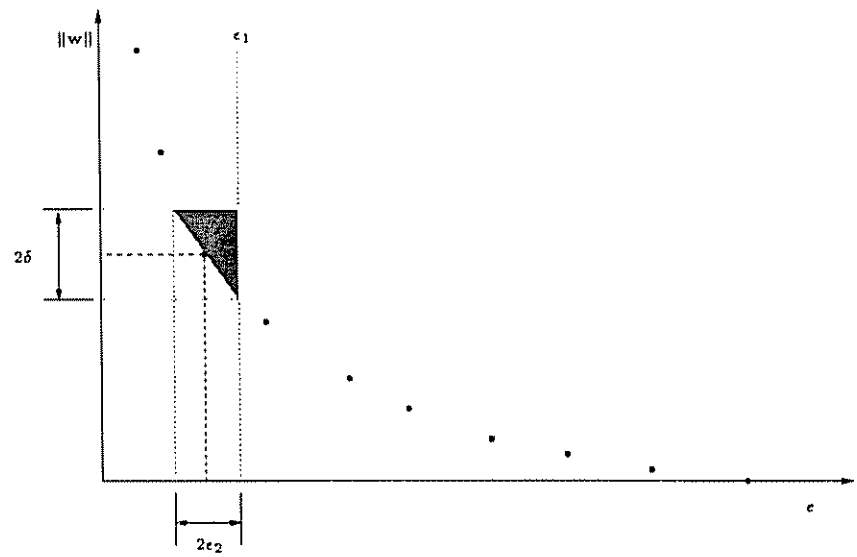


Figura 5.4: Região de equivalência entre os funcionais

Definição 2: Considere os funcionais $f(\mathbf{x}; \mathbf{w}_1)$ e $f(\mathbf{x}; \mathbf{w}_2)$. Diz-se que $f(\mathbf{x}; \mathbf{w}_1)$ é uma aproximação para $f(\mathbf{x}; \mathbf{w}_2)$ se estes funcionais são $(\delta, \epsilon_1, \epsilon_2)$-equivalentes, satisfazendo as condições descritas pela Equação (5.3).

■

Definição 3: Seja um conjunto $\mathcal{T} = \{\mathbf{x}_i, (d_i + \xi_i)\}_{i=1}^{N_T+N_V}$. Sejam I_T e I_V tais que:

$$\begin{cases} \{I_T + I_V\} = \{1, \ldots, N_T + N_V\} \\ I_T \cap I_V = \emptyset \\ \rho(I_T) = N_T; \quad \rho(I_V) = N_V \end{cases}$$

onde $\rho(\cdot)$ denota a cardinalidade do argumento.

Definam-se:

- O conjunto de treinamento;

$$\mathcal{T}_T = \{(\mathbf{x}_i, d_i + \xi_i) \mid i \in I_T\}$$

- O conjunto de validação.

$$\mathcal{T}_V = \{(\mathbf{x}_{V_i}, d_{V_i} + \xi_i) \mid i \in I_V\}$$

■

A seguir é abordada a regra de decisão na qual se baseia o decisor e com base nas definições anteriores, demonstra-se a viabilidade do conjunto Pareto-ótimo para escolha da solução final e que esta pode ser eficientemente escolhida utilizando-se um conjunto de validação.

5.2 A regra de decisão

A regra de decisão é dada pela Equação (5.4),

$$\mathbf{w}^* = \underset{\mathbf{w} \in \mathcal{W}^*}{\arg \min}\ e_V \tag{5.4}$$

onde e_V é dado pela Equação (5.5).

$$e_V = \frac{1}{N_V} \sum_{i=1}^{N_V} [(d_{V_i} + \xi_i) - f(\mathbf{x}_{V_i}; \mathbf{w})]^2 \tag{5.5}$$

O vetor $\mathbf{w}^*$ que minimizar o funcional dado pela Equação (5.5) é a melhor solução pertencente ao conjunto Pareto-ótimo e sua alta capacidade de generalização é assegurada. O Teorema 2 mostra a viabilidade do Pareto para escolha da melhor solução.

Teorema 2 *Seja $\mathcal{W}^*$ o conjunto das soluções Pareto-ótimas para um dado conjunto de treinamento $\mathcal{T}_T = \{\mathbf{x}_i, (\mathbf{d}_i + \xi_i)\}_{i=1}^{N_T}$ estatisticamente representativo*[1]*. Seja $\mathcal{T}_V = \{\mathbf{x}_{V_i}, (\mathbf{d}_{V_i} + \xi_{V_i})\}_{i=1}^{N_V}$ um conjunto de validação também estatisticamente representativo. Seja $e_V = \frac{1}{N_V} \sum_{i \in I_V} [(\mathbf{d}_{V_i} + \xi_i) - f(\mathbf{x}_{V_i}; \mathbf{w})]^2$ o erro em relação aos padrões de validação, ξ um ruído com distribuição normal, média zero e variância σ^2 presente nos*

[1]Um conjunto de dados é estatisticamente representativo quando este possui o número de elementos suficientemente grande para representar o funcional que se deseja modelar em toda a faixa de interesse.

dados $\mathbf{w} \in \mathcal{W}^*$. *Se* $\mathbf{w}^* \in \mathcal{W}^*$ *é o vetor que minimiza* e_V *em* $\mathcal{W}^*$ *para qualquer*[2] $\mathcal{T}_V = \{\mathbf{x}_{V_i}, (\mathbf{d}_{V_i} + \xi_i)\}_{i=1}^{N_V}$, *então* $f(\mathbf{x}; \mathbf{w}^*)$ *é a aproximação para a função geradora* $f_g(\mathbf{x})$ *que minimiza*

$$E\left[\int [f_g(\mathbf{x}) - f(\mathbf{x}; \mathbf{w})]^2 d\mathbf{x}\right]$$

em $\mathbf{w}$.

DEMONSTRAÇÃO: Considere o funcional descrito pela Equação (5.6):

$$\mathcal{E}[f(\mathbf{x}; \mathbf{w})] = E\left[\int [f_g(\mathbf{x}) - f(\mathbf{x}; \mathbf{w})]^2 d\mathbf{x}\right] \tag{5.6}$$

Considere o espaço $\mathcal{W}$ constituído de vetores $\mathbf{w}$ quaisquer, tais que um conjunto $\mathcal{F}$ é formado por todas as funções $f(\mathbf{x}; \mathbf{w})$ que possam ser mapeadas por uma RNA de topologia $u_i - u_h - u_o$. Se a RNA possui u_h suficientemente grande, existe um vetor $\mathbf{w}^o$ tal que a função $f(\mathbf{x}; \mathbf{w}^o)$ minimiza o funcional $\mathcal{E}[f(\mathbf{x}; \mathbf{w})]$ [Cybenko, 1989] e $\mathcal{N}^o = \|\mathbf{w}^o\|$.

Seja $\mathcal{W}^*$ o conjunto Pareto-ótimo e $\mathcal{T}_V = \{\mathbf{x}_{V_i}, (d_{V_i} + \xi_i)\}_{i=1}^{N_V}$ um conjunto de validação constituído por N_V padrões amostrados na função geradora $f_g(\mathbf{x})$ mais um termo de ruído.

Seja $\mathbf{w}^* \in \mathcal{W}^*$ o vetor de pesos que minimiza o funcional descrito pela Equação (5.5), que descreve o erro de validação para cada $\mathbf{w} \in \mathcal{W}^*$.

Considerando um conjunto de validação com $\sigma^2 = 0$, então d_V é a própria função $f_g(\mathbf{x})$. Logo, $f(\mathbf{x}; \mathbf{w}^*) \equiv f(\mathbf{x}; \mathbf{w}^o)$ e, pela Definição 01,

$$\begin{aligned} |\|\mathbf{w}^*\| - \|\mathbf{w}^o\|| &\leq \delta \text{ e} \\ |e^* - e^o| &\leq \epsilon_2. \end{aligned}$$

Como $(e^o, \mathbf{w}^o)$ mapeia a função geradora $f_g(\mathbf{x})$ no espaço dos objetivos, $(e^*, \mathbf{w}^*)$ mapeia a melhor aproximação para a função geradora neste mesmo espaço sendo que $\mathbf{w}^* \in \mathcal{W}^*$.

Se um conjunto de validação possui a variância do ruído diferente de zero, o vetor de pesos $\mathbf{w}^*$ continua sendo o argumento que minimiza o funcional descrito pela Equação (5.5) e o ponto de mínimo deste funcional não se altera conforme pode ser visto no desenvolvimento a seguir.

Seja

$$e_V = \frac{1}{N_V} \sum_{i=1}^{N_V} [(d_{V_i} + \xi_i) - f(\mathbf{x}_{V_i}; \mathbf{w})]^2 \tag{5.7}$$

[2]Tem-se que e_V é igual para todo $\mathcal{T}_V$, a partir da premissa da "representatividade estatística" desse conjunto.

Expandindo o funcional dado pela Equação (5.7) tem-se:

$$e_V = \frac{1}{N_V}\sum_{i=1}^{N_V}[(d_{V_i}+\xi_i)^2 + f(\mathbf{x}_{V_i};\mathbf{w})^2 - 2(d_{V_i}+\xi_i)f(\mathbf{x}_{V_i};\mathbf{w})]$$

$$e_V = \frac{1}{N_V}\sum_{i=1}^{N_V}[d_{V_i}^2 + \xi_i^2 + 2d_{V_i}\xi_i + f(\mathbf{x}_{V_i};\mathbf{w})^2 - 2d_{V_i}f(\mathbf{x}_{V_i};\mathbf{w}) - 2\xi_i f(\mathbf{x}_{V_i};\mathbf{w})]$$

$$\begin{aligned} e_V &= \frac{1}{N_V}\sum_{i=1}^{N_V}[d_{V_i}^2 + f(\mathbf{x}_{V_i};\mathbf{w})^2 - 2d_{V_i}f(\mathbf{x}_{V_i};\mathbf{w})] + \\ &+ \frac{1}{N_V}\sum_{i=1}^{N_V}[\xi_i^2 + 2d_{V_i}\xi_i - 2\xi_i f(\mathbf{x}_{V_i};\mathbf{w})] \end{aligned}$$

$$e_V = \frac{1}{N_V}\sum_{i=1}^{N_V}[d_{V_i} - f(\mathbf{x}_{V_i};\mathbf{w})]^2 + \frac{1}{N_V}\sum_{i=1}^{N_V}[\xi_i^2 + 2\xi_i(d_{V_i} - f(\mathbf{x}_{V_i};\mathbf{w}))]$$

$$e_V = \frac{1}{N_V}\sum_{i=1}^{N_V}[d_{V_i} - f(\mathbf{x}_{V_i};\mathbf{w})]^2 + \frac{1}{N_V}\sum_{i=1}^{N_V}(\xi_i)^2 + \frac{2}{N_V}\sum_{i=1}^{N_V}[\xi_i(d_{V_i} - f(\mathbf{x}_{V_i};\mathbf{w}))]$$

Como

$$\frac{1}{N_V}\sum_{i=1}^{N_V}(\xi_i)^2 \cong \sigma^2$$

Tem-se,

$$e_V = \frac{1}{N_V}\sum_{i=1}^{N_V}[d_{V_i} - f(\mathbf{x}_{V_i};\mathbf{w})]^2 + \sigma^2 + \frac{2}{N_V}\sum_{i=1}^{N_V}[\xi_i(d_{V_i} - f(\mathbf{x}_{V_i};\mathbf{w}))] \qquad (5.8)$$

Como o erro em relação ao conjunto de validação e o ruído não são correlacionados, o termo $\frac{2}{N_V}\sum_{i=1}^{N_V}[\xi_i(d_{V_i} - f(\mathbf{x}_{V_i};\mathbf{w}))]$ tende para zero quando $N_V \to \infty$. Então, a expressão final para o erro de validação pode ser reescrita conforme Equação (5.9).

$$e_V = \frac{1}{N_V} \sum_{i=1}^{N_V} [d_{V_i} - f(\mathbf{x}_{V_i}; \mathbf{w})]^2 + \sigma^2 \tag{5.9}$$

Logo, a equivalência entre $f(\mathbf{x}; \mathbf{w}^*)$ e $f(\mathbf{x}; \mathbf{w}^o)$ se mantém e portanto, $f(\mathbf{x}; \mathbf{w}^*)$ continua sendo a melhor aproximação para a função geradora pertencente ao conjunto $\mathcal{W}^*$ mesmo para um conjunto de validação ruidoso.

■

Com a finalidade de ilustrar alguns pontos do Teorema 2, seguem algumas análises gráficas.

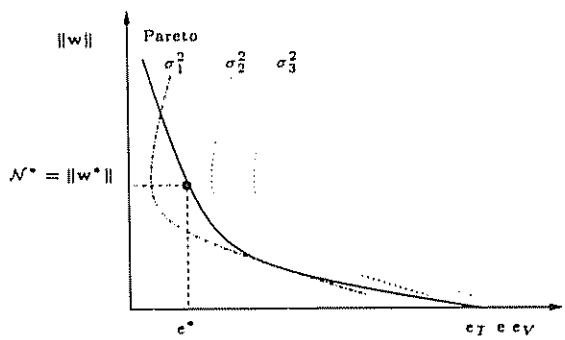


Figura 5.5: Conjunto Pareto-ótimo e curvas de validação para diferentes variâncias.

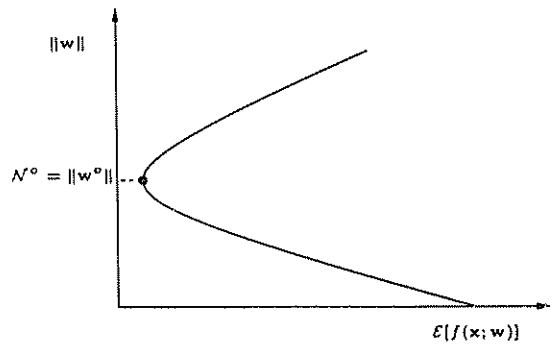


Figura 5.6: Erro entre a função geradora $f_g(\mathbf{x})$ e um conjunto de funções $f(\mathbf{x}; \mathbf{w})$, $\mathbf{w} \in \mathcal{W}$.

A Figura 5.5 ilustra a solução ótima $\mathbf{w}^*$ no conjunto Pareto-ótimo bem como suas coordenadas $(e^*, \mathcal{N}^*)$. As curvas em traço-ponto e pontilhadas são respectivamente as curvas de validação para dados sem ruído, $\sigma_1^2 = 0$ e com ruídos de variâncias σ_2^2 e $\sigma_3^2 \neq 0$ sendo $\sigma_3^2 > \sigma_2^2$. Note-se que o ponto de mínimo na coordenada $\|\mathbf{w}\|$ de cada curva de validação não se altera em função da variância do ruído como demonstra o Teorema 2.

A Figura 5.6 ilustra o ponto de mínimo de norma $\mathcal{N}^o$ do funcional dado pela Equação (5.6). Este ponto indica que existe um funcional $f(\mathbf{x}; \mathbf{w}^o)$ que aproxima a função $f_g(\mathbf{x})$ em um universo $\mathcal{F}$ de funções possíveis. O Teorema 2 mostra que existe uma solução $\mathbf{w}^* \in \mathcal{W}^*$ tal que $f(\mathbf{x}; \mathbf{w}^*) \equiv f(\mathbf{x}; \mathbf{w}^o)$ e, portanto, $f(\mathbf{x}; \mathbf{w}^*)$ também é uma aproximação para $f_g(\mathbf{x})$ pertencente ao conjunto Pareto-ótimo.

A Figura 5.7 ilustra a solução ótima $(e^o, \|\mathbf{w}^o\|)$, a solução pertencente ao conjunto Pareto-ótimo $(e^*, \|\mathbf{w}^*\|)$ e a região de equivalência entre os funcionais. Note-se que $(e^*, \|\mathbf{w}^*\|)$ e $(e^o, \|\mathbf{w}^o\|)$ pertencem a esta região o que faz $f(\mathbf{x}; \mathbf{w}^*) \equiv f(\mathbf{x}; \mathbf{w}^o)$.

Considerando agora a possibilidade de o ruído ser correlacionado com o erro, o ponto de mínimo da curva de validação poderá sofrer um deslocamento. Se a correlação for baixa, o deslocamento poderá ainda assim não afetar o resultado devido à escolha de uma solução final pertencente à região de equivalência. A Figura 5.8 ilustra esta situação.

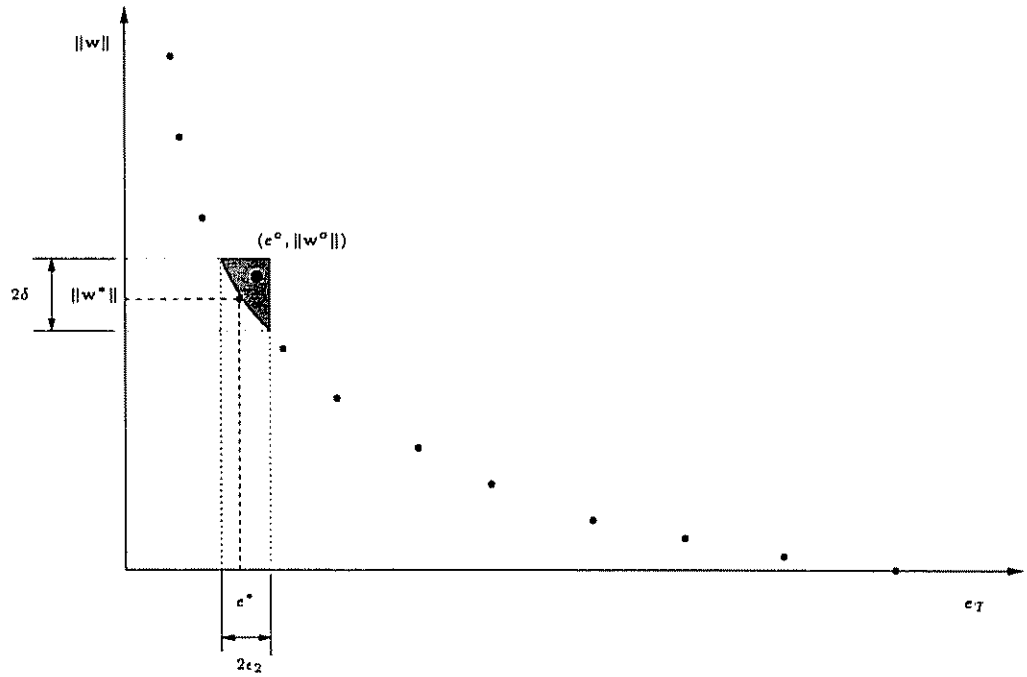


Figura 5.7: Ilustração da solução ótima $(e^o, \|\mathbf{w}^o\|)$, da melhor solução encontrada pelo algoritmo MOBJ $(e^*, \|\mathbf{w}^*\|)$ e da região hachurada onde possivelmente estará mapeada a solução ótima.

Tomando-se a Equação (5.8) e considerando-se que o ruído apresente correlação com o erro, tem-se duas possibilidades onde na primeira delas o ponto de mínimo será deslocado para cima fazendo o decisor optar por uma solução de norma ligeiramente maior ou na segunda possibilidade, o ponto de mínimo será deslocado para baixo fazendo o decisor optar por uma solução de norma ligeiramente inferior. Este deslocamento não afeta a qualidade da solução final escolhida desde que o ruído não esteja altamente correlacionado com o erro. Caso contrário, o decisor iria escolher uma solução fora da região de equivalência entre os funcionais.

Além dos fatores já discutidos, existem outros fatores que podem influenciar no processo de escolha da solução MOBJ final, como por exemplo, a intensidade do ruído presente nos dados, a escolha de uma das infinitas realizações para o conjunto de validação de tamanho finito, a resolução do conjunto Pareto-ótimo e ainda, a dimensão da RNA a ser treinada. A seguir estes fatores serão discutidos.

5.2.1 Efeito da intensidade de ruído nos dados

Através da Equação (5.9) pode ser visto que o valor do erro de validação para a melhor solução MOBJ escolhida pelo decisor é igual à variância do ruído presente nos dados, sendo independente de outros parâmetros. Portanto, o processo de decisão não sofre influência devido a variância do ruído e o ponto de mínimo da curva de validação, em

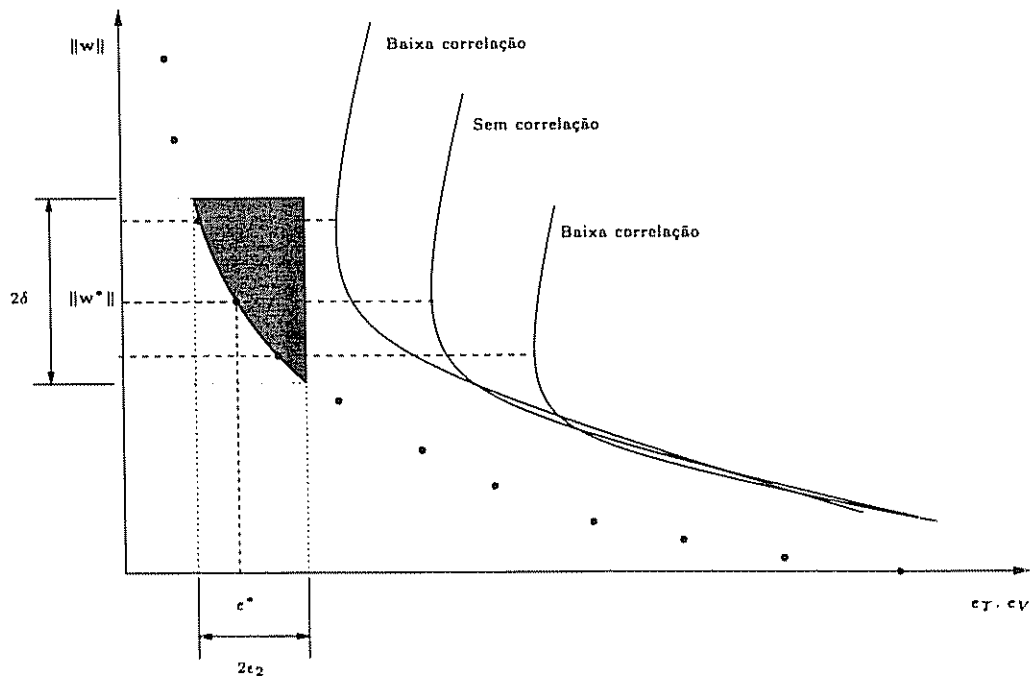


Figura 5.8: Deslocamento do ponto de mínimo da curva de validação em função da correlação entre ruído e erro.

tese, acontece sempre no mesmo ponto. Contudo, na maior parte dos casos, o número de amostras do conjunto de validação é pequeno e uma determinada realização deste conjunto pode influenciar na escolha da melhor solução. Isto se explica devido ao "afrouxamento" da premissa de não-correlação à medida em que o tamanho da amostra diminui. Como existe uma região de equivalência entre os funcionais, qualquer uma das soluções tomada como solução final resolve o problema de treinamento.

Para ilustrar o efeito da variância do ruído no processo de escolha da melhor solução, foram treinadas 05 redes através do algoritmo MOBJ implementado através do problema ϵ-restrito. O treinamento de cada rede se deu da seguinte forma: o conjunto Pareto-ótimo é constituído de 20 soluções, portanto tem-se que escolher uma entre 20 redes sendo que estas possuem topologias $1 \times 10 \times 1$. Os demais parâmetros envolvidos no processo são listados na Tabela 5.1. Na penúltima linha da mesma pode ser visto o número da solução final N_{w^*} escolhida em cada caso e na última linha pode ser visto o valor da norma para a solução escolhida $\mathcal{N}^*$. Note-se que, como o $\delta\varepsilon$ é 0.5 em todos os casos, as normas das soluções ótimas estão muito próximas.

As Figuras 5.9–5.13 ilustram os resultados das simulações realizadas para verificação do efeito do ruído. A função cuja regressão foi feita é descrita pela Equação (5.10).

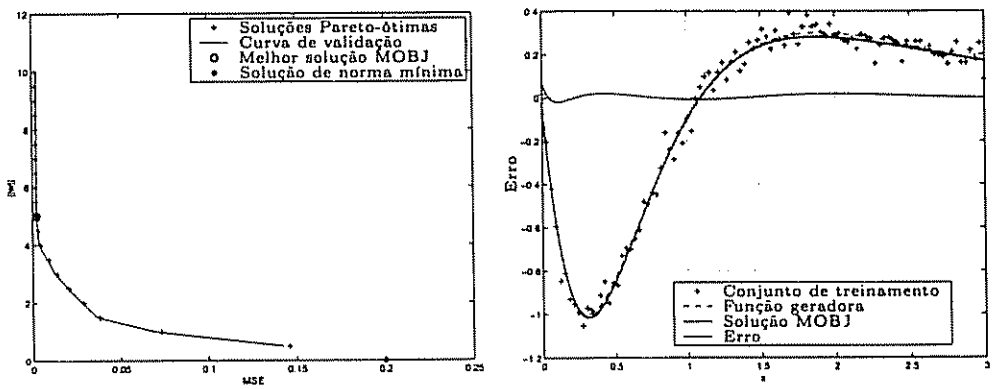
$$f(x) = 4,26(e^{-x} - 4e^{-2x} + 3e^{-3x}) \tag{5.10}$$

Nas Figuras 5.9–5.13, o gráfico do lado direito representa o conjunto Pareto-ótimo

Tabela 5.1: Parâmetros das redes treinadas - Variação do ruído

Parâmetro	Teste 1	Teste 2	Teste 3	Teste 4	Teste 5
f_{ah}	tanh(·)	tanh(·)	tanh(·)	tanh(·)	tanh(·)
f_{ao}	linear	linear	linear	linear	linear
N_N	10	10	10	10	10
$\delta\epsilon$	0.5	0.5	0.5	0.5	0.5
σ^2	0.05^2	0.1^2	0.2^2	0.3^2	0.4^2
N_V	100	100	100	100	100
N_T	100	100	100	100	100
N_{w^*}	10	11	10	11	09
$\mathcal{N}^*$	5	5.5	5	5.5	4.5

encontrado em cada caso onde também pode ser visto a curva de validação calculada a partir dos dados de validação e ainda a melhor solução encontrada pelo decisor. No lado direito de cada figura pode ser vista a resposta da rede escolhida como solução final.



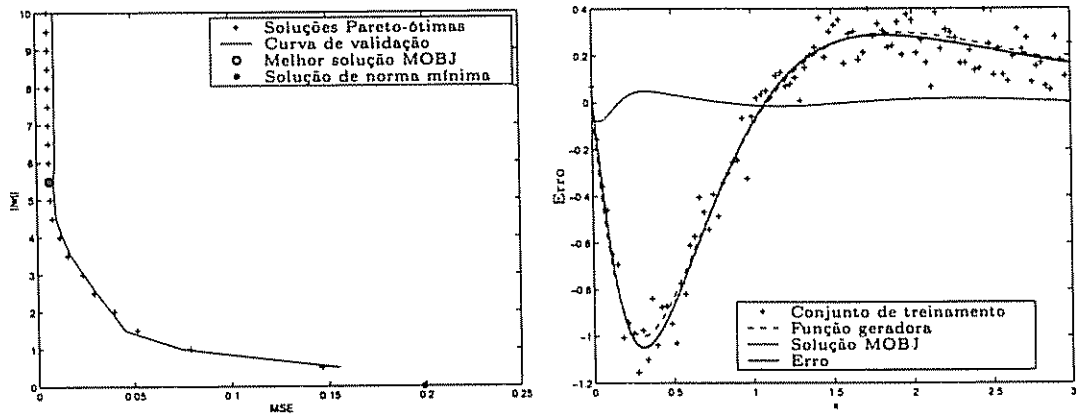
(a) Conjunto Pareto-ótimo (b) Melhor solução MOBJ, $N_{w^*} = 10$

Figura 5.9: Regressão da função $f(x)$, $\zeta = 20$, $\delta\epsilon = 0.5$, $N_T = 100$, $\sigma^2 = 0.05^2$

Nas Figuras 5.9(b)–5.12(b) pode-se notar que a solução MOBJ ecolhida está livre de *overfitting* mesmo com a variância crescente.

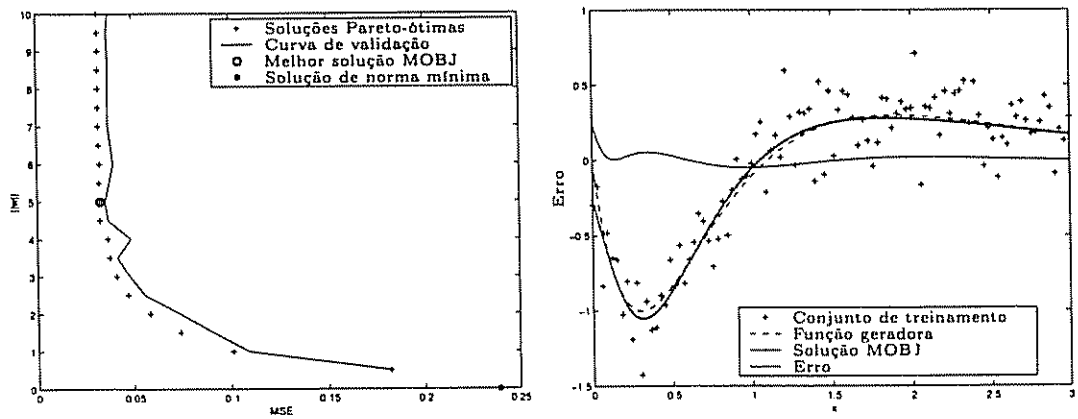
Na Figura 5.13(b) pode-se notar um certo grau de *underfitting*. Isto pode ocorrer principalmente devido à alta variância do ruído que neste caso é de 0.4^2, devido ao número baixo de amostras no conjunto de treinamento e ainda devido a uma determinada realização do conjunto de treinamento. Um outro conjunto de tamanho N_T pode possibilitar que o algoritmo MOBJ encontre uma solução melhor.

A Figura 5.14 mostra o conjunto Pareto-ótimo gerado a partir de 100 padrões com $\sigma^2 = 0.4^2$ e 4 curvas de validação geradas com diferentes conjuntos $\mathcal{T}_V$ de variância $\sigma^2 = 0.4^2$. O número de amostras de cada conjunto é 100. Pela Equação (5.9), o ponto de mínimo de cada curva de validação deveria acontecer em σ^2 mas, apesar de as curvas manterem o ponto de mínimo fixo, os valores para os erros de validação são diferentes



(a) Conjunto Pareto-ótimo (b) Melhor solução MOBJ, $N_{w^*} = 11$

Figura 5.10: Regressão da função $f(x)$, $\zeta = 20$, $\delta\epsilon = 0.5$, $N_T = 100$, $\sigma^2 = 0.1^2$



(a) Conjunto Pareto-ótimo (b) Melhor solução MOBJ, $N_{w^*} = 10$

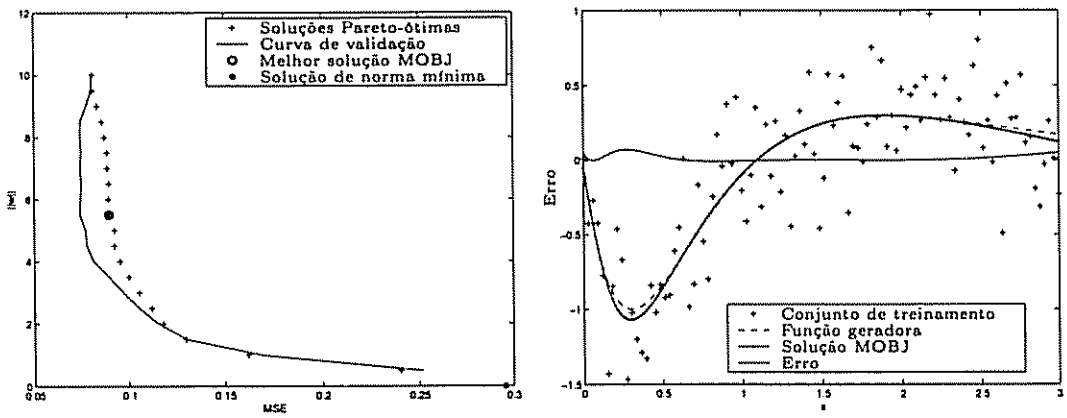
Figura 5.11: Regressão da função $f(x)$, $\zeta = 20$, $\delta\epsilon = 0.5$, $N_T = 100$, $\sigma^2 = 0.2^2$

para cada uma das 4 curvas. Este fato ocorre devido às realizações diferentes do conjunto de validação com tamanho fixo $N_V = 100$ padrões.

Para mostrar a influência do número de padrões do conjunto de validação na forma da curva de validação, o experimento anterior foi repetido para 4 conjuntos diferentes com variâncias $\sigma^2 = 0.4^2$ mas $N_V = 10.000$ padrões. Note-se que quando o número de padrões cresce, cada curva de validação tem seu mínimo exatamente no mesmo ponto e o valor do erro de generalização dado pela Equação 5.9 tende para σ^2.

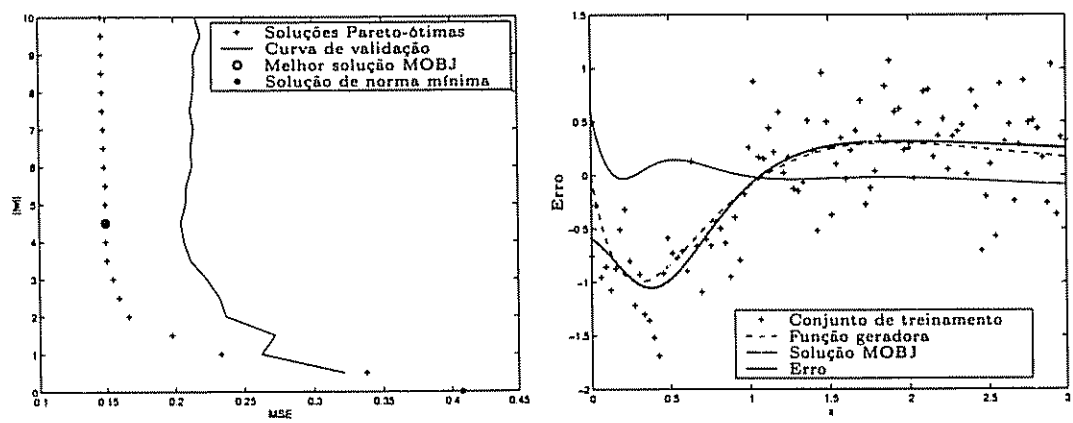
A Figura 5.16 mostra diferentes curvas de validação para variâncias diferentes. Os cinco conjuntos de validação utilizados possuem $N_T = 10.000$ e σ^2 igual à 0.05^2, 0.1^2, 0.2^2, 0.3^2 e 0.4^2. O ponto de mínimo de cada curva também se manteve fixo, mas o valor do erro de generalização em cada caso é igual a σ^2.

Para resumir, o processo de decisão é pouco influenciado pelo ruído presente nos dados



(a) Conjunto Pareto-ótimo (b) Melhor solução MOBJ, $N_{\mathbf{w}^*} = 11$

Figura 5.12: Regressão da função $f(x)$, $\zeta = 20$, $\delta\epsilon = 0.5$, $N_T = 100$, $\sigma^2 = 0.3^2$



(a) Conjunto Pareto-ótimo (b) Melhor solução MOBJ, $N_{\mathbf{w}^*} = 9$

Figura 5.13: Regressão da função $f(x)$, $\zeta = 20$, $\delta\epsilon = 0.5$, $N_T = 100$, $\sigma^2 = 0.4^2$

principalmente se os conjuntos de treinamento e validação possuem número de padrões suficientes para representar bem a função que se deseja mapear. A seguir é discutido o efeito de uma determinada realização do conjunto de treinamento no processo de decisão.

5.2.2 Efeito da escolha do conjunto de Treinamento

Dado uma determinado realização do conjunto de treinamento, o algoritmo MOBJ encontra um conjunto Pareto-ótimo que não é exatamente o mesmo se for considerada uma outra realização do conjunto de treinamento. Portanto, a posição exata do conjunto Pareto-ótimo é influenciada pelos dados. A Figura 5.17 ilustra a posição de 5 conjuntos Pareto calculados a partir de cinco conjuntos de treinamento diferentes com 80 padrões cada. O conjunto de validação utilizado no processo de decisão é o mesmo em todos os

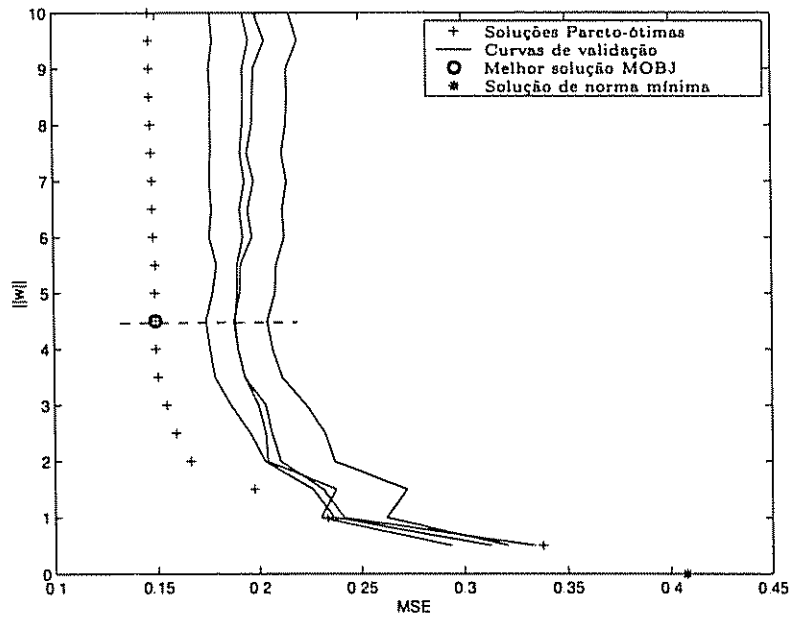


Figura 5.14: Influência de conjuntos de validação diferentes com mesmas variâncias, 100 padrões de validação.

casos mas as soluções escolhidas pelo decisor variam em torno de um determinada norma, embora estejam próximas. Esta diferença entre as normas das soluções finais não chega a afetar a qualidade das mesmas.

A Figura 5.18 ilustra um conjunto Pareto gerado a partir de um conjunto de treinamento com 400 padrões e também a média dos Paretos ilustrados na Figura 5.17. Note-se que na média, o conjunto Pareto-ótimo converge para a região Pareto-ótima real (região Pareto-ótima teórica obtida se um conjunto de treinamento de tamanho infinito fosse utilizado). Para o problema em questão, 400 amostras já possibilita o cálculo de um conjunto Pareto-ótimo que pode ser chamado de real. Por ora, vale dizer que a utilização de um conjunto de tamanho finito resulta em uma boa aproximação do Pareto real e soluções com alta capacidade de generalização podem ser escolhidas. Note-se na Figura 5.18, que a posição da solução média calculada a partir da posição da solução ótima obtida em cada Pareto da Figura 5.17 e a posição da solução ótima obtida através do conjunto com 400 padrões são praticamente as mesmas, não havendo grande diferença entre as normas de cada solução.

Portanto, uma determinada realização do conjunto de treinamento influencia na posição das soluções Pareto-ótimas principalmente quando o número de padrões adotado é pequeno mas ainda assim pode-se escolher uma boa solução. De qualquer forma, a solução final escolhida é pouco influenciada por uma determinada realização do conjunto de treinamento desde que este seja suficientemente grande para representar bem a função que se deseja mapear.

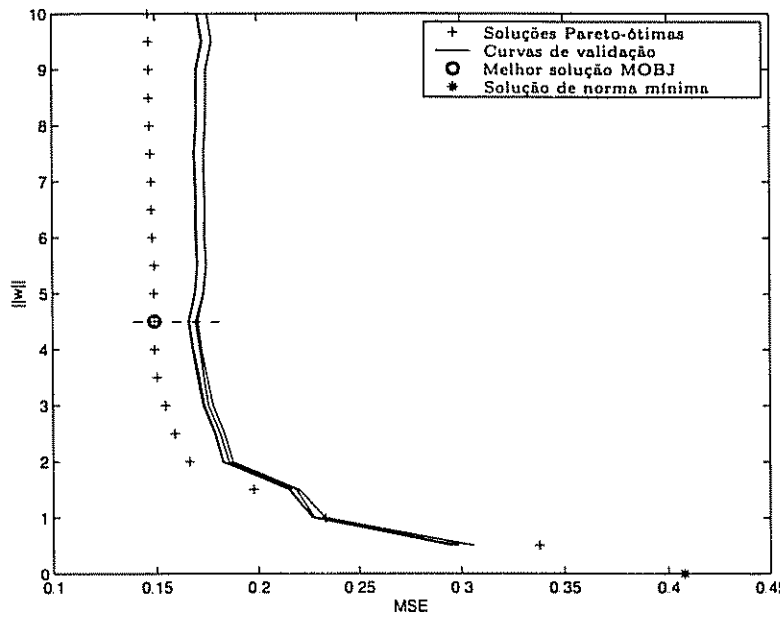


Figura 5.15: Influência de conjuntos de validação com variâncias iguais, 10000 padrões de validação.

5.2.3 Efeito da resolução do conjunto Pareto-ótimo

Outro fator que pode influenciar na qualidade da solução escolhida pelo decisor é a resolução do Pareto. Para o método de relaxação, a resolução do Pareto é dada pelo parâmetro ζ, que é o número de soluções a serem geradas. Tipicamente, este parâmetro deve assumir valores entre 20 e 50 e as soluções geradas alteram pouco a complexidade de uma para outra, passando pela complexidade desejada. Para o método Pϵ, o parâmetro $\delta\varepsilon$ controla o incremento de complexidade de uma solução para outra. Tipicamente este parâmetro deve assumir valores entre 0.5 e 2. Alguns experimentos com o algoritmo MOBJ implementado a partir do problema Pϵ foram realizados para verificação da influência da resolução do Pareto na qualidade da solução final escolhida. Os parâmetros utilizados no treinamento estão relacionados na Tabela 5.2 e para cada teste as redes tinham como topologia $1 \times 10 \times 1$. O parâmetro ζ foi escolhido de forma que a maior norma obtida fosse igual a 20.

A Figura 5.19(a) mostra o conjunto Pareto-ótimo com suas 20 soluções geradas com diferença entre normas de 1. Para este caso, o decisor escolheu a solução de número 5 como sendo a melhor o que corresponde a $\mathcal{N}^* = 5$. A Figura 5.19(b) mostra a solução final escolhida.

A Figura 5.20(a) mostra o conjunto Pareto-ótimo com 10 soluções geradas com diferença entre normas de 2. Para este caso, o decisor escolheu a solução de número 3 como sendo a melhor o que corresponde a $\mathcal{N}^* = 6$. A Figura 5.20(b) mostra a solução final escolhida.

Por último, a Figura 5.21(a) mostra o conjunto Pareto-ótimo com 5 soluções geradas

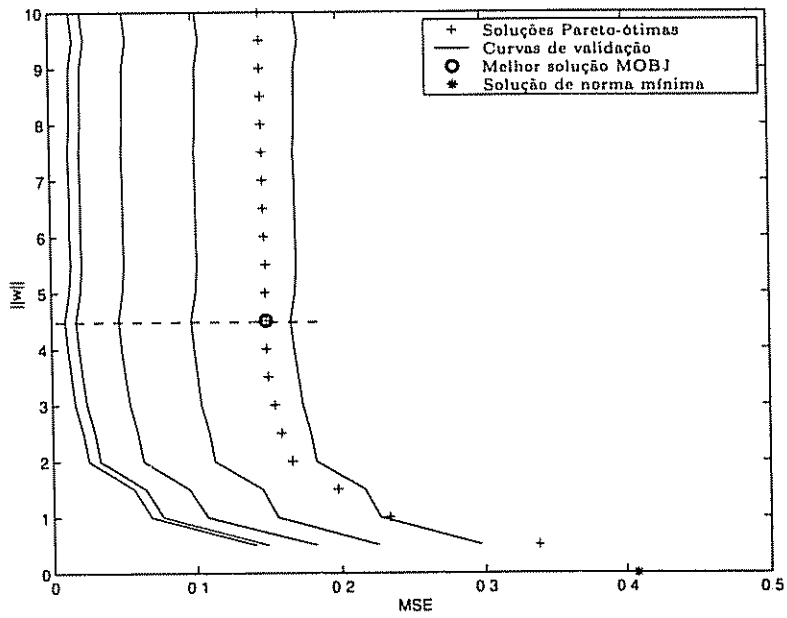


Figura 5.16: Influência de conjuntos de validação com variâncias diferentes, 10000 padrões de validação.

com diferença entre normas de 4. Para este caso, o decisor escolheu a solução de número 1 como sendo a melhor o que corresponde a $\mathcal{N}^* = 4$. A Figura 5.21(b) mostra a solução final escolhida. Note-se que nesta simulação a solução final já apresenta um determinado grau de *underfitting* e se a solução final escolhida fosse a de número 2, o que corresponde à $\mathcal{N}^* = 8$, a solução apresentaria um determinado grau de *overfitting*. O problema da resolução é que as soluções obtidas pode não estar exatamente sobre a complexidade ótima para o problema em questão. Para este caso específico, $\mathcal{N}^* = 4$ resulta em uma complexidade abaixo da necessária e $\mathcal{N}^* = 8$ seria acima da necessária, uma vez que a adequada para este problema está em torno de $\mathcal{N}^* = 6$.

Com estes resultados pode-se concluir que uma diferença grande entre as normas das

Tabela 5.2: Parâmetros das redes treinadas - Variação da resolução

Parâmetro	Teste 1	Teste 2	Teste 3
f_{ah}	$\tanh(\cdot)$	$\tanh(\cdot)$	$\tanh(\cdot)$
f_{ao}	linear	linear	linear
N_N	10	10	10
ζ	20	10	5
$\delta\varepsilon$	1	2	4
σ^2	0.2^2	0.2^2	0.2^2
N_V	80	80	80
N_T	80	80	80
N_{w^*}	5	3	1
$\mathcal{N}^*$	5	6	4

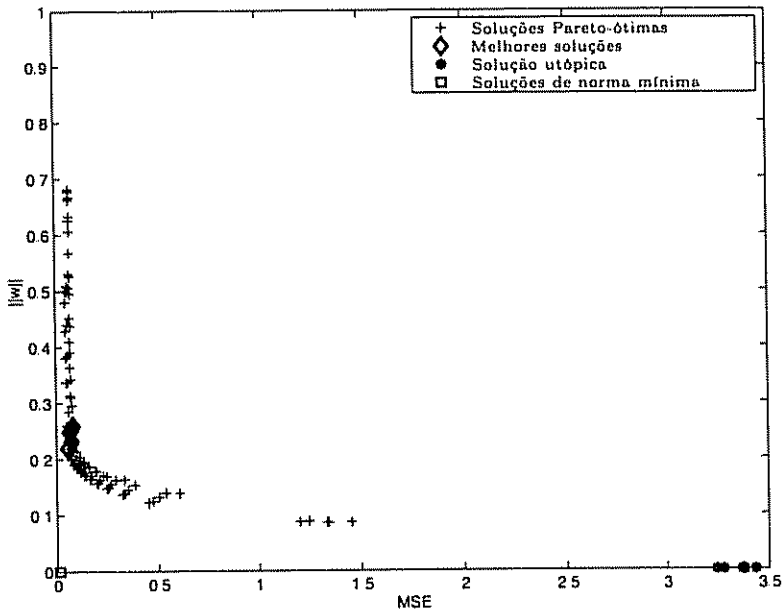


Figura 5.17: Influência da escolha particular de um conjunto de treinamento

soluções geradas pode fazer com que estas saltem o ponto de complexidade adequada. Outro fator também discutível é qual seria a norma da solução mais complexa a ser gerada para cada problema. Nos exemplos acima, a solução MOBJ de maior norma foi igual 20 e a solução adequada possui norma em torno de 6. Neste caso, foram geradas soluções MOBJ de norma 0 até 20. Mas como saber se a solução ótima a ser escolhida pelo decisor esta compreendida neste intervalo? Por exemplo, se a solução ótima tivesse como norma o valor 30, neste caso ela não seria gerada e a solução final obtida pelo decisor seria a última solução gerada, de norma igual a 20.

Este problema é facilmente contornado se o usuário verificar a curva de validação. Por exemplo, a Figura 5.22(a) mostra um caso onde fica fácil perceber que a solução ótima esta fora do conjunto de soluções geradas e então seria necessária a geração de soluções de normas mais elevadas. Com a observação do ponto de inflexão da curva de validação fica fácil perceber se a solução ótima foi gerada e pertence ao conjunto Pareto-ótimo ou não. Para casos onde este fato ocorre, a solução escolhida pelo decisor será sempre uma solução sub-ajustada. A Figura 5.22(b) mostra a solução final escolhida, com um certo grau de *underfitting*.

5.2.4 Efeito da dimensão da RNA

Para que o algoritmo MOBJ controle a complexidade e encontre desde soluções sub-ajustadas até as super-ajustadas é necessário que a topologia da rede a ser treinada ou a dimensão da mesma permita tal ajuste. Se a topologia da rede a ser treinada já é de antemão pouco complexa para o dado problema de aprendizagem, não há como o algoritmo

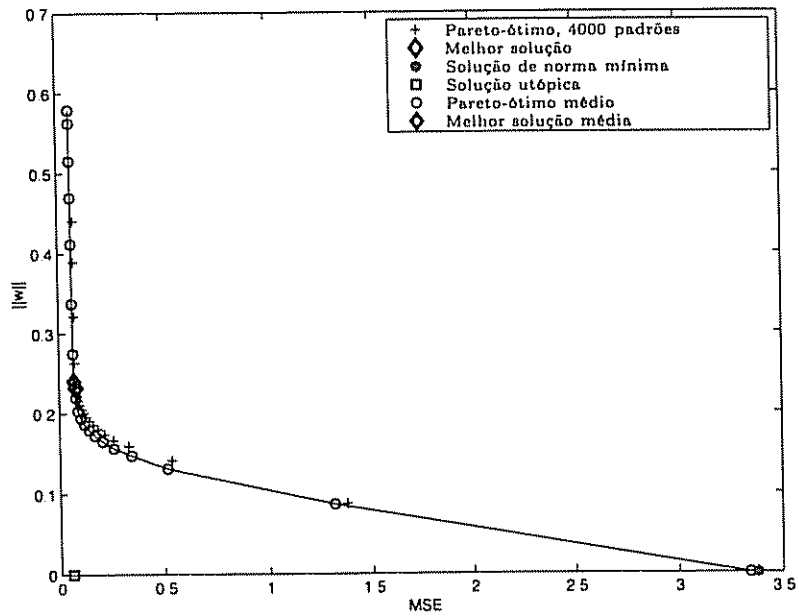


Figura 5.18: Conjunto Pareto-ótimo para 4.000 padrões e conjunto Pareto-ótimo médio

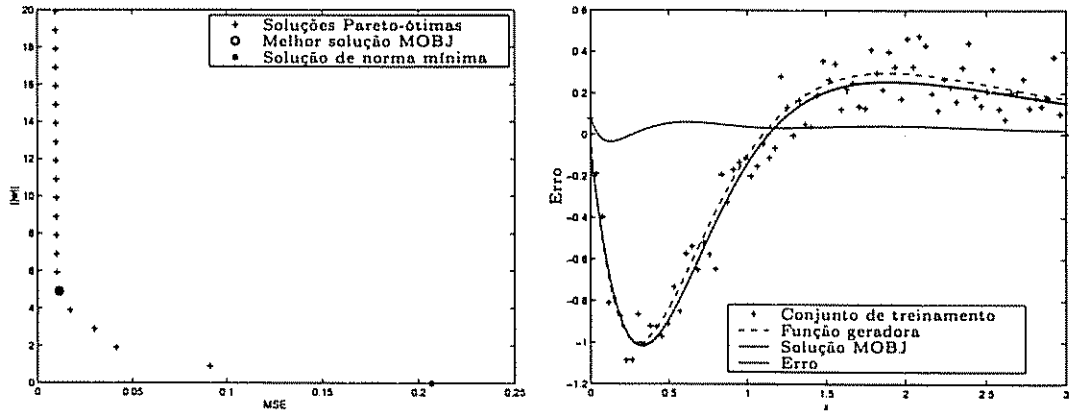
MOBJ encontrar uma solução com complexidade adequada mas se a complexidade da mesma é suficiente ou mais do que suficiente para a dada tarefa, o algoritmo MOBJ é capaz de fazer o controle da complexidade e encontrar uma solução adequada. Portanto, se a rede não é a priori pouco complexa, o algoritmo MOBJ pode encontrar uma solução com alta capacidade de generalização independentemente da dimensão da rede, simplesmente controlando a magnitude dos pesos da mesma.

Alguns testes foram realizados para ilustrar o efeito da dimensão da RNA na obtenção da solução final. Os parâmetros dos testes realizados são mostrados na Tabela 5.3, onde as N_N denota o número de neurônios na camada escondida, que foi variado de 5 a 40.

Tabela 5.3: Parâmetros das redes treinadas - Alteração da topologia

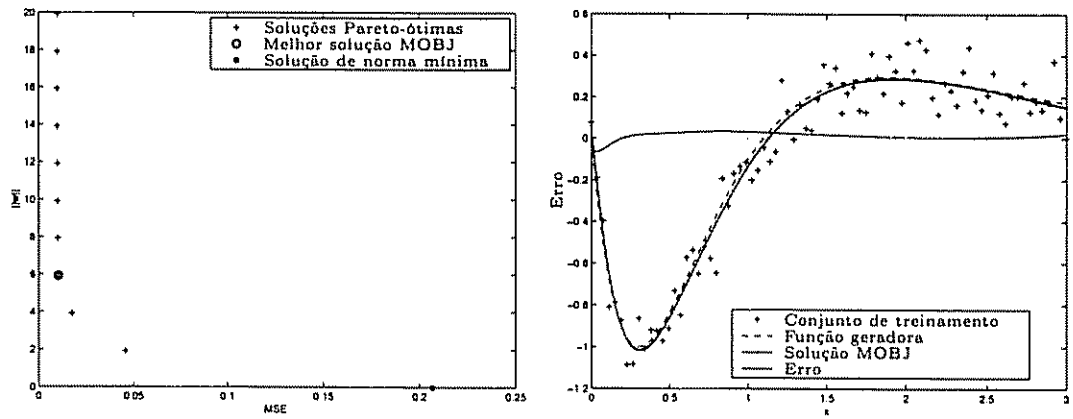
Parâmetro	Teste 1	Teste 2	Teste 3	Teste 4
f_{ah}	$\tanh(\cdot)$	$\tanh(\cdot)$	$\tanh(\cdot)$	$\tanh(\cdot)$
f_{ao}	linear	linear	linear	linear
N_N	5	10	20	40
ζ	20	20	20	20
$\delta\epsilon$	0.5	0.5	0.5	0.5
σ^2	0.2^2	0.2^2	0.2^2	0.2^2
N_V	80	80	80	80
N_T	80	80	80	80
N_{w^*}	09	11	09	10
$\mathcal{N}^*$	4.5	5.5	4.5	5.0

As Figuras 5.23–5.26 mostram os resultados dos experimentos. Para cada caso, o algoritmo MOBJ foi capaz de encontrar uma solução com alta capacidade de generalização



(a) Conjunto Pareto-ótimo (b) Melhor solução MOBJ

Figura 5.19: Regressão da função $f(x)$, $\zeta = 20$, $\delta\epsilon = 1$, $N_T = 80$



(a) Conjunto Pareto-ótimo (b) Melhor solução MOBJ

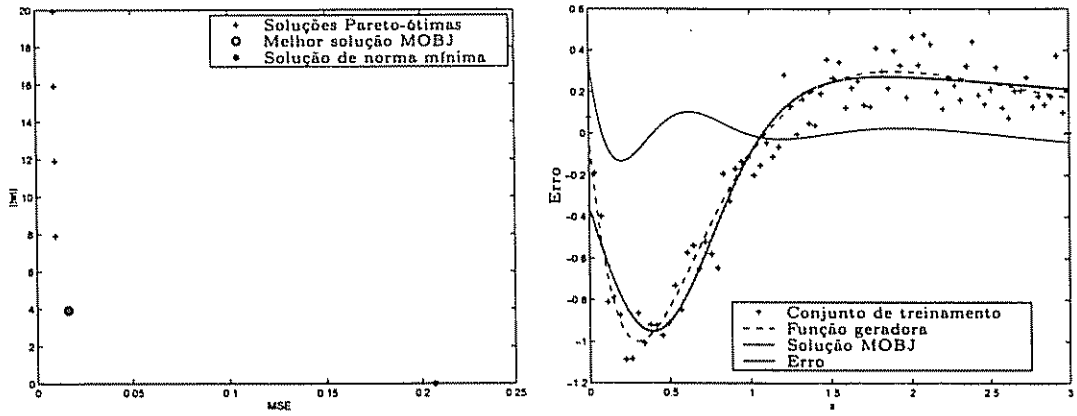
Figura 5.20: Regressão da função $f(x)$, $\zeta = 10$, $\delta\epsilon = 2$, $N_T = 80$

independente da dimensionalidade da rede.

Tendo em mente as discussões envolvendo o processo de obtenção das soluções Pareto-ótimas e o processo de decisão, pode-se concluir que o método MOBJ incluindo geração do conjunto Pareto e decisor é bastante robusto além de garantir que soluções com alta capacidade de generalização possam ser alcançadas com relativo esforço computacional mas sem a necessidade de sensibilidade por parte do usuário.

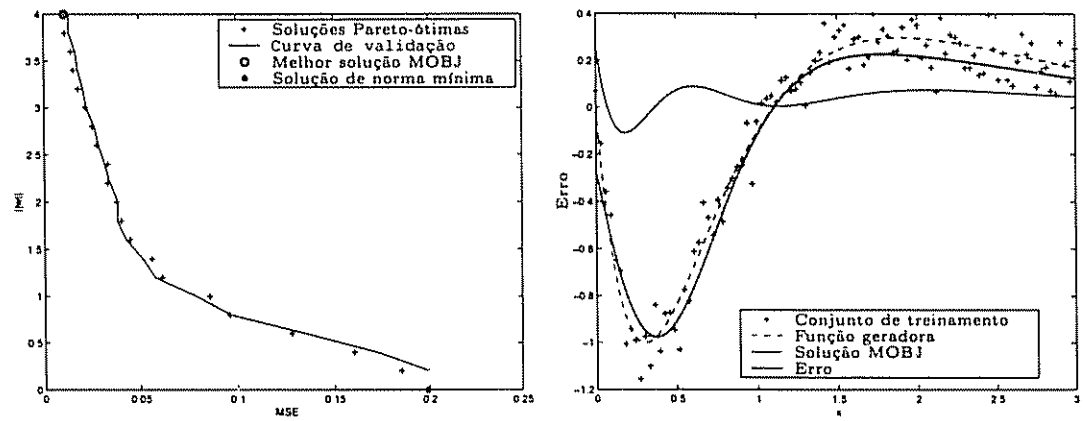
5.3 Conclusões do capítulo

Neste capítulo foi demonstrado o processo de decisão chamado de decisor bem como sua efetividade. Vários fatores que podem influenciar tanto no processo de geração das soluções Pareto-ótimas como no processo de decisão da melhor solução foram discutidos



(a) Conjunto Pareto-ótimo (b) Melhor solução MOBJ

Figura 5.21: Regressão da função $f(x)$, $\zeta = 4$, $\delta\epsilon = 4$, $N_T = 80$

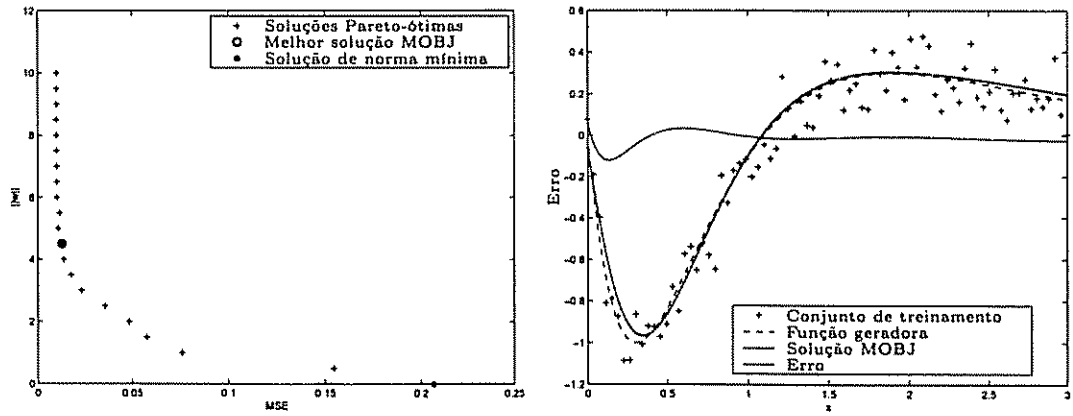


(a) Conjunto Pareto-ótimo (b) Melhor solução MOBJ

Figura 5.22: Regressão da função $f(x)$, $\zeta = 20$, $\delta\epsilon = 0.2$, $N_T = 100$

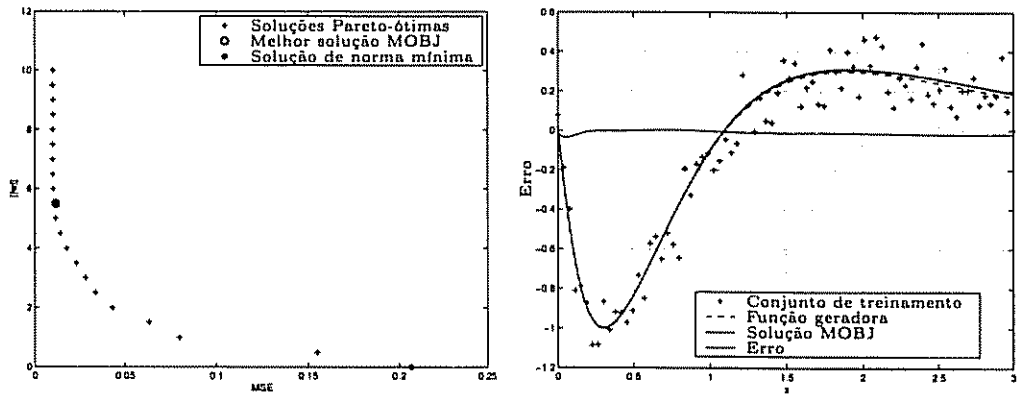
e ilustrados. São eles a intensidade do ruído presente nos dados, a escolha de uma determinada realização do conjunto de treinamento e validação, a resolução do conjunto Pareto-ótimo, o valor da norma máxima a ser gerada e ainda a dimensão da RNA a ser treinada. Após várias discussões conclui-se que o algoritmo proposto é bastante insensível à variações de parâmetros sendo capaz de gerar uma solução com alta capacidade de generalização com pouca intervenção do usuário.

No Capítulo 6 o método MOBJ é aplicado em problema de classificação e de regressão de diferentes complexidades. As soluções MOBJ obtidas são comparadas com as soluções geradas pelos algoritmos *Backpropagation* (BP) [Rumelhart et al., 1986b], *Weight Decay* (WD) [Hinton, 1989, Moody, 1992],*Optimal Brain Damage* [Cun et al., 1990] (OBD), *Cross-Validation* (CV) [Stone, 1974, Stone, 1978], *Early Stopping* (ES) [Weigend et al., 1990] e *Support Vector Machine* (SVM) [Cortes and Vapnik, 1995].



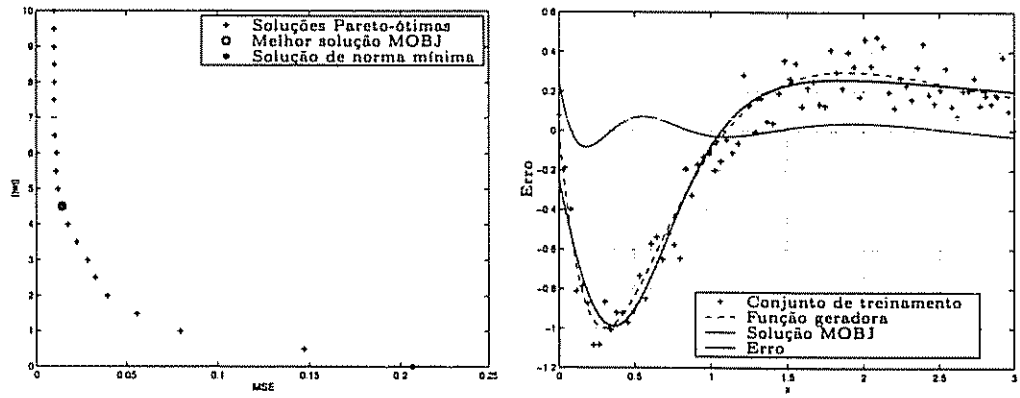
(a) Conjunto Pareto-ótimo (b) Melhor solução MOBJ

Figura 5.23: Regressão da função $f(x)$, $N_N = 5$ neurônios



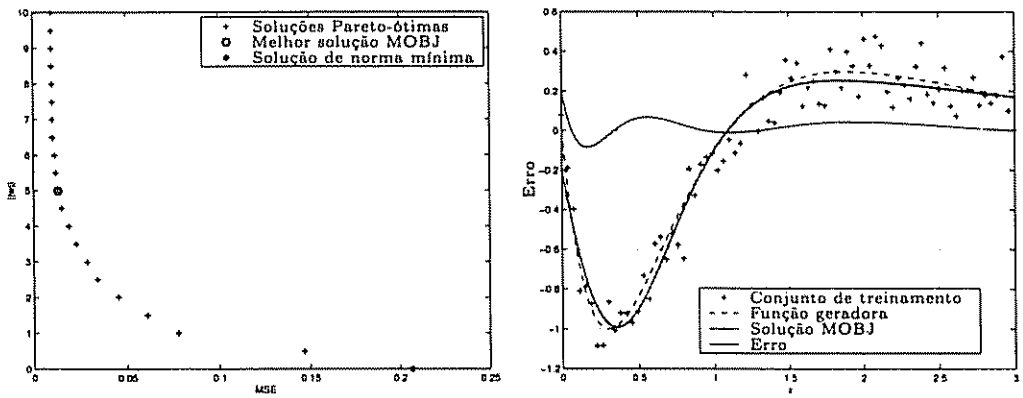
(a) Conjunto Pareto-ótimo (b) Melhor solução MOBJ

Figura 5.24: Regressão da função $f(x)$, $N_N = 10$ neurônios



(a) Conjunto Pareto-ótimo (b) Melhor solução MOBJ

Figura 5.25: Regressão da função $f(x)$, $N_N = 20$ neurônios



(a) Conjunto Pareto-ótimo (b) Melhor solução MOBJ

Figura 5.26: Regressão da função $f(x)$, $N_N = 40$ neurônios

Capítulo 6

Aplicação em Problemas de Classificação e Regressão

Neste Capítulo são abordados problemas de classificação e de regressão de diferentes complexidades para verificação da capacidade de aprendizagem e de generalização do método MOBJ.

6.1 Problemas de Classificação

Os problemas de classificação abordados são três sendo que os dois primeiros são problemas onde os dados de treinamento, validação e teste foram gerados por computador e apresentam superposição entre as classes. O último exemplo se trata de uma aplicação real de análise de crédito [Blake and Merz, 1998].

Em problemas de classificação, a rede aprende a classificar um determinado padrão entre várias classes ou categorias. Para avaliação do desempenho do classificador, um conjunto de padrões desconhecidos é mostrado ao mesmo e o número de acertos é armazenado. Nenhum dos padrões utilizados na fase de treinamento deve ser utilizado nesta fase. A classificação correta ocorre quando a rede associa um padrão desconhecido à sua verdadeira classe. Quando o classificador atribui o padrão desconhecido à classe errada, a classificação é dita incorreta.

Pode-se encontrar uma grande quantidade de aplicações de RNAs como classificadores na literatura em diversas áreas como reconhecimento de imagens [Nomura, 1999], reconhecimento de sons [Kepuska and Gowdy, 1989], classificação financeira [Horst et al., 1998, Lacerda et al., 2001], etc. Qualquer que seja a aplicação, a capacidade de generalização da rede é sempre uma propriedade muito procurada pelos projetistas mas nem sempre é fácil de ser obtida.

As soluções MOBJ obtidas para solucionar os problemas de classificação colocados nos exemplos 1, 2 e 3 são comparadas com as soluções dos algoritmos *Backpropagation*,

Weight Decay, *Optimal Brain Damage*, *Early Stopping*, *10-Fold Cross-Validation* e SVM.

Para todos os algoritmos abordados procurou-se manter a mesma topologia de rede MLP a ser treinada. Estas redes tinham tanto para os neurônios da camada intermediária quanto para os neurônios da camada de saída funções de ativação $f_a(\cdot) = tanh(\cdot)$.

A topologia das redes adotada é tal que a complexidade das mesmas excede à complexidade das tarefas de aprendizagem. Durante o processo de treinamento, cada algoritmo faz o ajuste de complexidade buscando encontrar a melhor solução em cada caso. Nas simulações seguintes, RNAs com topologias $u_i \times u_h \times u_o$ [1] são treinadas sendo u_i dado pelo número de entradas, u_h dado pelo número de neurônios da camada escondida N_N e u_o dado pelo número de saídas da rede.

O algoritmo utilizado para obtenção das soluções *backpropagation* (BP) foi o de Levenberg-Marquardt [Hagan and Menhaj, 1994]. Para obtenção das soluções *Early Stopping* (ES), o ajuste de pesos também foi feito pelo algoritmo de Levenberg-Marquardt sendo executado o número de iteração M_{fase_1} indicado em cada caso e a cada iteração foi apresentado um conjunto de validação para verificação do comportamento da capacidade de generalização. Conceitualmente, o algoritmo ES interrompe o treinamento quando o primeiro mínimo é encontrado. Como isto pode fazer com que a interrupção do treinamento ocorra antes do melhor momento, neste trabalho o algoritmo ES não é executado na forma em que foi concebido. Sendo assim, ao final das M_{fase_1} iterações, a rede que apresentar o menor erro de generalização é tomada como solução final. Portanto, a interrupção do treinamento não se dá no momento onde ocorre o primeiro ponto de mínimo do erro de validação. Para obtenção das soluções *10-Fold cross validation* (CV) foram utilizados dois conjuntos de dados, um de treinamento e outro de validação, utilizado para a escolha da melhor entre as 10 redes treinadas. O conjunto de treinamento foi dividido em 10 subconjuntos sendo que para cada uma das 10 redes treinadas foi utilizado um destes 10 subconjuntos para parada do treinamento por erro de generalização e os demais subconjuntos para o treinamento.

Para o treinamento da rede através do algoritmo *Weight Decay* (WD) o ponto fundamental é a escolha da constante de queda λ. Para as simulações seguintes, várias tentativas foram feitas com diferentes constantes até se obter uma rede considerada satisfatória. O algoritmo WD utilizado é o encontrado no *Toolbox* de identificação utilizando RNAs [Nørgaard, 1997]. O algoritmo *Optimal Brain Damage* (OBD) utilizado neste trabalho também faz parte do *Toolbox* de identificação utilizando RNAs e para o treinamento de redes utilizando este algoritmo são necessários dois conjuntos de dados. São treinadas várias redes com diferentes topologias iniciando a partir da topologia proposta e ao final deste processo, utiliza-se um conjunto de validação para escolha da melhor solução OBD. Para este algoritmo *pruning*, após a eliminação de um peso ocorre uma segunda fase de

[1] A notação RNA $u_i \times u_h \times u_o$ denota uma rede com u_i níveis de entrada, u_h nodos na camada escondida e u_o nodos de saída.

treinamento com o número de iterações dado por M_{fase_2}.

Para o treinamento das redes através do algoritmo SVM, tem-se que escolher o *kernel* a ser utilizado, neste caso foi utilizado um *kernel* RBF além do ajuste dos parâmetros do mesmo que neste caso é a variância σ^2 e finalmente, ajustar o limite para os multiplicadores de Lagrange C. É importante ressaltar que apesar da quantidade baixa de parâmetros a serem ajustados pelo usuário, uma má escolha destes pode resultar em uma solução com baixa capacidade de generalização, apresentando *overfitting* ou *underfitting*. Para obtenção da rede final houve a necessidade de treinamento de várias SVMs com diferentes conjuntos de parâmetros até que uma SVM considerada satisfatória fosse alcançada uma vez que a solução SVM é muito sensível aos parâmetros de treinamento.

Para o treinamento das RNAs através do algoritmo MOBJ, apenas dois parâmetros devem ser fornecidos pelo usuário sendo que a solução final não é muito sensível a estes parâmetros. O primeiro parâmetro é a quantidade ζ de soluções MOBJ a serem geradas e o parâmetro $\delta\varepsilon$ determina a diferença de norma entre estas soluções. Uma discussão detalhada da influência destes parâmetros na solução final foi feita no Capítulo 5. O primeiro problema de classificação é abordado a seguir.

6.1.1 Exemplo 1

Neste primeiro exemplo é abordado um problema de classificação constituído de duas classes cujos padrões foram amostrados aleatoriamente em duas distribuições normais bivariadas [Soares et al., 1991] sendo que para a primeira classe, as médias em relação a cada variável são $\mu_{x_1}^{C_1} = 2$ e $\mu_{x_2}^{C_1} = 2$ e para a segunda classe, as médias são $\mu_{x_1}^{C_2} = 4$ e $\mu_{x_2}^{C_2} = 4$. Ambas as classes possuem as mesmas variâncias em relação a cada variável, ou seja, $\sigma_1^2 = \sigma_2^2 = 1.5^2$. Para a classe 1 a resposta da rede deve ser $d = -1$ e para a classe 2, a resposta da rede deve ser $d = +1$. Para as simulações a seguir, foram gerados conjuntos de treinamento com 50 padrões por classe e com as variâncias utilizadas, as classes apresentam superposição.

O processo de treinamento de redes pelos algoritmos resulta em superfícies de separação entre as classes. Os valores desta superfície iguais a zero representam a sua curva de nível zero (curva de decisão). Esta curva de nível zero determina a curva de separação entre as classes.

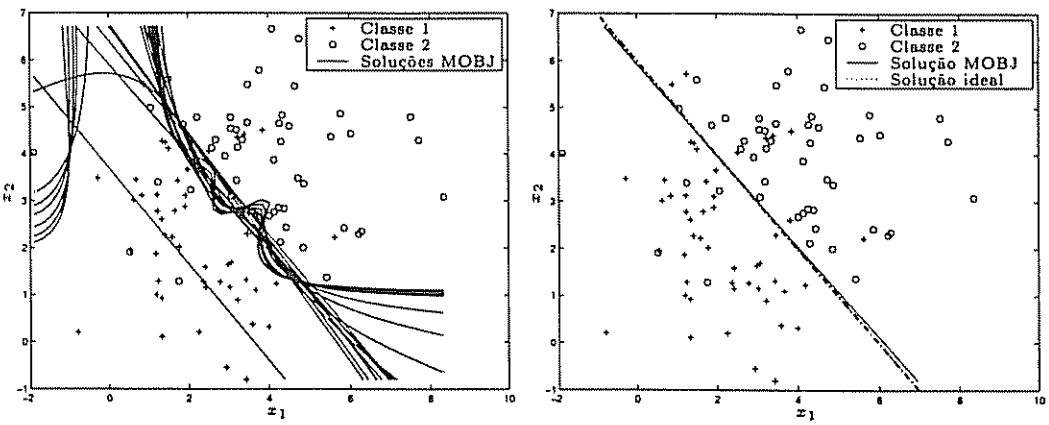
Para cada algoritmo em questão é necessário um conjunto de parâmetros de treinamento e estes foram ajustados segundo a Tabela 6.1 para o exemplo 1.

Tabela 6.1: Parâmetros das redes treinadas para o Exemplo 1

Parâmetro	BP	WD	OBD	ES	CV	SVM	MOBJ
λ	–	4	–	–	–	–	–
M_{fase_1}	250	200	100	100	50	–	–
M_{fase_2}	–	–	10	–	–	–	–
N_T	100	100	100	100	100	100	100
N_V	–	–	50	50	50	–	50
Kernel	–	–	–	–	–	rbf	–
C	–	–	–	–	–	4	–
σ^2 do *Kernel*	–	–	–	–	–	1	–
N_N	30	30	30	30	30	–	30
ζ	–	–	–	–	–	–	30
$\delta\varepsilon$	–	–	–	–	–	–	0.5

Obs.: O símbolo "–" denota que o parâmetro não é definido para o método.

As ilustrações a seguir referem-se ao problema de classificação colocado no exemplo 1. A Figura 6.1(a) ilustra todas as soluções MOBJ geradas e a Figura 6.1(b) mostra a melhor solução MOBJ escolhida e a solução analítica para este problema de classificação.



(a) Soluções MOBJ (b) Solução MOBJ escolhida

Figura 6.1: Problema de classificação Exemplo 1 - Soluções MOBJ

- **Comentário das Figuras 6.1(a) e 6.1(b):**

  Na Figura 6.1(a) são mostradas todas as 20 soluções obtidas através do algoritmo MOBJ. Estas soluções foram obtidas com diferença de norma entre elas de 0.5. É característico do método proposto a geração de soluções desde sub-ajustadas até super-ajustadas, conforme pode ser visto.

  Na Figura 6.1(b) é mostrada a melhor solução MOBJ, escolhida no conjunto Pareto-ótimo de 20 soluções através de teste de generalização utilizando um conjunto com

50 padrões. É mostrada também a solução analítica para o problema que é praticamente igual à solução MOBJ escolhida. Note-se que a solução MOBJ foi obtida a partir de um conjunto restrito de padrões de treinamento ($N_T = 100$) e validação $N_V = 50$.

O cálculo da solução analítica para este problema pode ser feito da seguinte forma:

Sejam x_1 e x_2 duas variáveis independentes. A distribuição normal bivariada considerando estas variáveis é descrita pela Equação (6.1)

$$p(x_1, x_2) = \frac{1}{2\pi\sigma_{x_1}\sigma_{x_2}} \exp\left[-\frac{1}{2}\left\{\frac{(x_1 - \mu_{x_1})^2}{\sigma_{x_1}^2} + \frac{(x_2 - \mu_{x_2})^2}{\sigma_{x_2}^2}\right\}\right] \tag{6.1}$$

O problema de classificação em questão é constituído de duas distribuições normais bivariadas em $\mathbb{R}^2$ sendo que cada distribuição possui o mesmo desvio padrão em relação à cada variável, ou seja, $\sigma_{x_1} = \sigma_{x_2}$. As distribuições são dadas pelas Equações (6.2) e (6.3).

$$p_1(x_1, x_2) = \frac{1}{2\pi\sigma_1^2} \exp\left[-\frac{1}{2}\left\{\frac{(x_1 - \mu_{x1}^{C_1})^2}{\sigma_1^2} + \frac{(x_2 - \mu_{x2}^{C_1})^2}{\sigma_1^2}\right\}\right] \tag{6.2}$$

$$p_2(x_1, x_2) = \frac{1}{2\pi\sigma_2^2} \exp\left[-\frac{1}{2}\left\{\frac{(x_1 - \mu_{x1}^{C_2})^2}{\sigma_2^2} + \frac{(x_2 - \mu_{x2}^{C_2})^2}{\sigma_2^2}\right\}\right] \tag{6.3}$$

Considerando que cada distribuição corresponde a uma classe, a melhor curva de separação corresponde ao lugar geométrico onde x_1 e x_2 são tais que $p_1(x_1, x_2) = p_2(x_1, x_2)$. A Equação (6.4) descreve a curva de separação que é a solução ótima para o problema.

$$\frac{(x_1 - \mu_{x1}^{C_1})^2 + (x_2 - \mu_{x2}^{C_1})^2}{\sigma_1^2} = \frac{(x_1 - \mu_{x1}^{C_2})^2 + (x_2 - \mu_{x2}^{C_2})^2}{\sigma_2^2} + 2\ln\left(\frac{\sigma_1^2}{\sigma_2^2}\right) \tag{6.4}$$

Neste exemplo, as variâncias das classes são iguais a 1.5^2 e as médias são $\mu_{x1}^{C_1} = 2$, $\mu_{x2}^{C_1} = 2$, $\mu_{x1}^{C_2} = 4$ e $\mu_{x2}^{C_2} = 4$.

Fazendo as substituições, tem-se:

$$(x_1 - 2)^2 + (x_2 - 2)^2 = (x_1 - 4)^2 + (x_2 - 4)^2$$

A solução analítica para o problema de classificação em questão é dada pela Equação (6.5).

$$x_2 = -x_1 + 6 \tag{6.5}$$

Este resultado corrobora a hipótese de que no conjunto Pareto-ótimo pode ser encontrada uma solução que é no mínimo Ω-equivalente à solução analítica para o problema de aprendizagem, conferindo alta capacidade de generalização para a mesma.

O conjunto Pareto-ótimo e a posição das soluções geradas pelos demais algoritmos no espaço dos objetivos podem ser vistos na Figura 6.2.

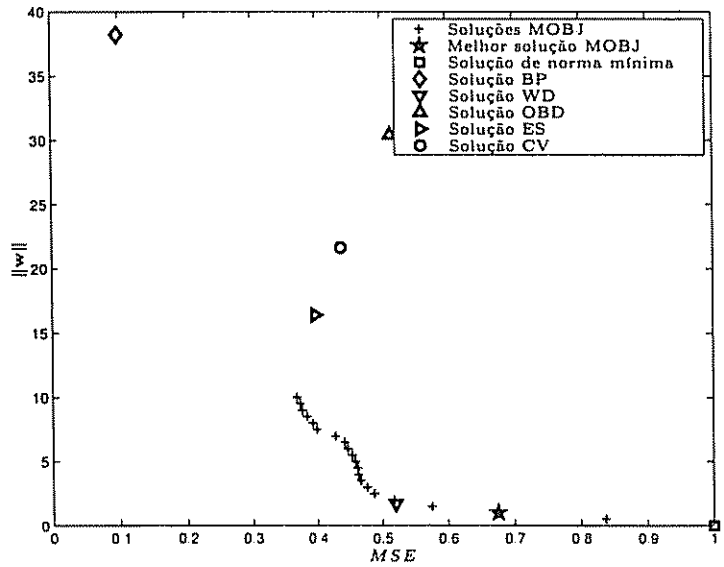


Figura 6.2: Conjunto Pareto-ótimo e demais soluções no espaço dos objetivos - Exemplo 1

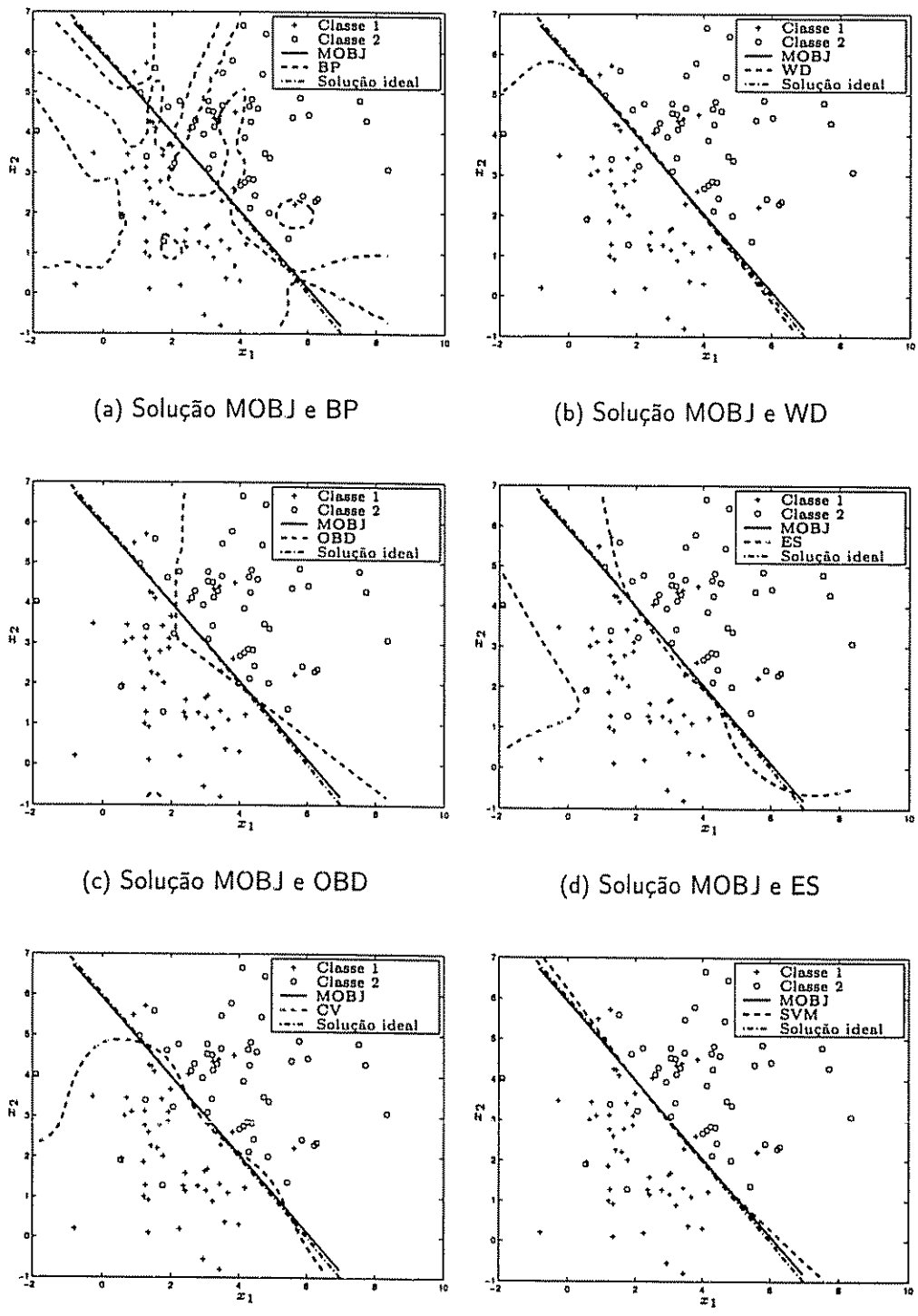
- **Comentário da Figura 6.2:**

  Nesta Figura é mostrado o conjunto Pareto-ótimo para o problema. Este conjunto é constituído das soluções obtidas pelo algoritmo MOBJ e pela solução de norma mínima. A melhor solução MOBJ escolhida no processo de decisão é denotada por uma estrela ($\star$).

  As soluções dos demais algoritmos também foram mapeadas no espaço dos objetivos para que se possa visualizar o comportamento do erro e da norma destas soluções. Verifica-se que as soluções BP, OBD, CV e ES possuem normas maiores que a solução MOBJ, o que significa que estas tendem a apresentar capacidades de generalização mais pobres. A norma da solução WD esta muito próxima da norma da solução MOBJ porém um pouco maior indicando que esta solução apresentará capacidade de generalização semelhante à apresentada pelo algoritmo MOBJ. Na Tabela 6.2 pode-se verificar quantitativamente o que esta diferença de norma significa. Como a solução BP é a de maior norma, é de se esperar que a superfície de decisão seja a menos suave de todas e portanto apresente maiores erros de generalização devido ao super-ajuste ao conjunto de treinamento.

  A solução SVM não é mostrada no espaço dos objetivos por se tratar de uma máquina bastante diferente das demais. Uma SVM não possui pesos entre os nodos de entrada e nodos escondidos e o nodo de saída é sempre linear. Portanto, existe apenas um conjunto de pesos entre a camada escondida e o nodo de saída e não faz

sentido comparar o erro e a norma relativos às SVMs com redes treinadas através dos demais métodos.



(a) Solução MOBJ e BP

(b) Solução MOBJ e WD

(c) Solução MOBJ e OBD

(d) Solução MOBJ e ES

(e) Solução MOBJ e CV

(f) Solução MOBJ e SVM

Figura 6.3: Comparações entre as soluções para o exemplo 1

As curvas de decisão correspondentes a cada método de treinamento abordado são

ilustradas nas Figuras 6.3(a)–6.3(f). As superfícies de decisão geradas pelos algoritmos são ilustradas nas Figuras 6.4–6.9. Através destas figuras pode-se fazer uma análise qualitativa das soluções.

- **Comentário das Figuras 6.3(a)–6.3(f):**

  Nas Figuras 6.3(a)–6.3(f) são ilustradas as curvas de decisão de cada algoritmo. Em cada figura podem ser vistas também as soluções MOBJ e a ideal. Note-se que para os demais métodos a curva de decisão difere da curva ideal calculada analiticamente. Como o algoritmo BP não faz nenhum controle de complexidade e a RNA é bastante complexa para esta tarefa, a função mapeada está super-ajustadas aos padrões de treinamento. A solução SVM se aproxima muito da solução MOBJ, embora as formas de obtê-las sejam bastante diferentes. Em problemas onde não se pode observar visualmente a qualidade da solução SVM a tarefa de ajustar um conjunto adequado de parâmetros de treinamento não é uma tarefa simples, o que não ocorre quando se utiliza o algoritmo MOBJ.

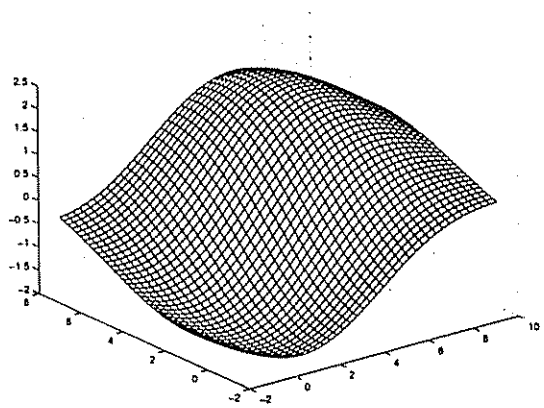


Figura 6.4: Solução SVM

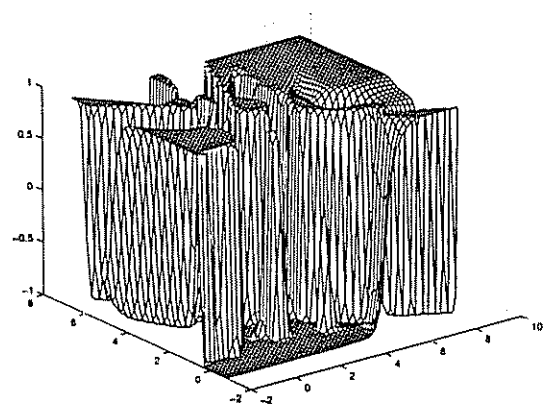


Figura 6.5: Solução BP

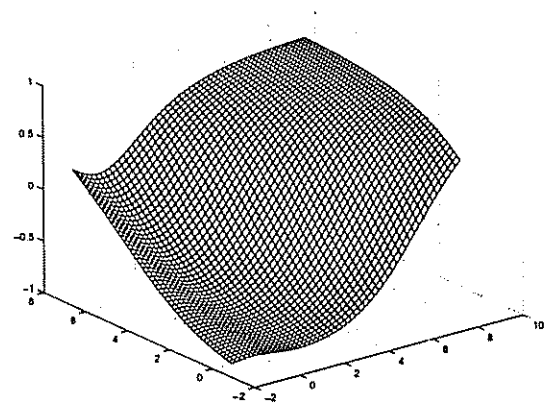


Figura 6.6: Solução WD

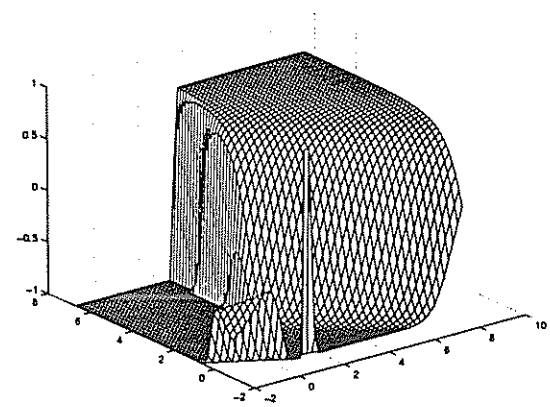


Figura 6.7: Solução OBD

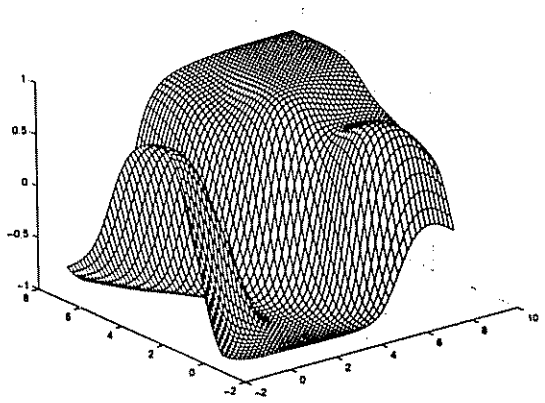


Figura 6.8: Solução ES

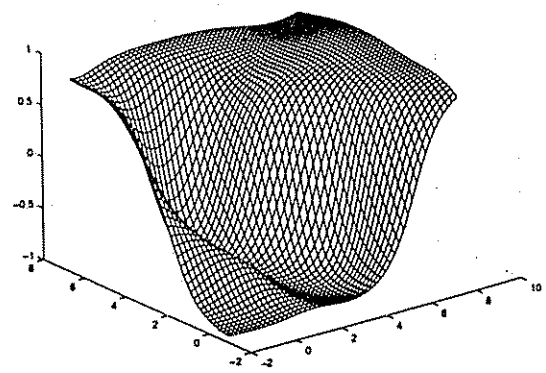


Figura 6.9: Solução CV

A Figura 6.10 ilustra a superfície de decisão obtida através do algoritmo MOBJ.

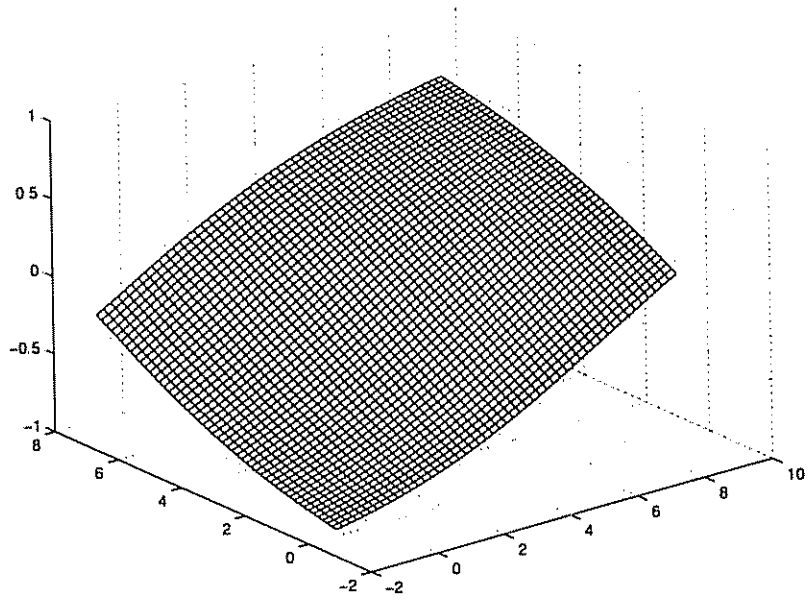


Figura 6.10: Superfície de decisão MOBJ para o exemplo 1

- **Comentário das Figuras 6.4–6.9 e da Figura 6.10:**

  Nestas figuras estão ilustradas as funções mapeadas por cada algoritmo. O que se deseja com esta ilustração é mostrar a capacidade de suavização do algoritmo MOBJ bem como a qualidade de suas soluções. As superfícies SVM, WD e CV são menos suaves que a superfície MOBJ embora sejam mais suaves que as soluções BP, ES e OBD. Estes três algoritmos geralmente retornam soluções de menor capacidade de generalização. Para resumir, com o algoritmo MOBJ pode-se partir de uma RNA capaz de gerar uma solução conforme ilustra a Figura 6.5 e chegar em uma solução conforme ilustra a Figura 6.10 sem alteração da estrutura da rede. O controle de complexidade é feito controlando somente a magnitude dos pesos sem que haja

grande esforço por parte do projetista.

A Tabela 6.2 traz os percentuais de acerto e os desvios padrão para as soluções obtidas através dos algoritmos abordados. Para o cálculo dos percentuais de acerto foram utilizados 100 conjuntos de dados constituídos de 2000 padrões cada O desvios são denotados por σ.

Tabela 6.2: Acertos para os padrões de teste.(100 conjuntos diferentes - 2000 pontos)

Algoritmos	Acerto	σ
Backpropagation	70.1885%	±0.9826%
Weight Decay	82.6710%	±0.8536%
Optimal Brain Damage	80.3945%	±0.7904%
Early Stopping	80.1750%	±0.8869%
Cross-Validation	82.0510%	±0.8696%
SVM	82.6845%	±0.7972%
MOBJ	82.7045%	±0.8481%

Na Tabela 6.2 podem-se observar os resultados do teste de generalização para as diversas soluções. Vale ressaltar que boas soluções podem ser encontradas utilizando-se de qualquer um dos métodos abordados, até mesmo com o algoritmo BP. A principal diferença está na facilidade ou não de se encontrar bons ajustes para cada algoritmo. Observe através dos resultados que o MOBJ apresentou melhor desempenho e é importante ressaltar o fato de que a solução MOBJ final não apresenta grande sensibilidade a parâmetros de treinamento como ocorre nos algoritmos WD e SVM.

- **Comentários finais da Subseção 6.1.1:**

  Como já mencionado anteriormente, todos os algoritmos abordados são capazes de alcançar soluções com boa capacidade de generalização. Entretanto, algumas dificuldades devem ser mencionadas. Por exemplo, a dificuldade de se ajustar a constante de queda para o algoritmo *Weight Decay*. Para se encontrar uma boa solução, é preciso um valor adequado para este parâmetro que é dependente do problema. O algoritmo ES é sensível à escolha dos padrões de treinamento e validação e o comportamento do erro de generalização não é monotonicamente decrescente até seu ponto de mínimo global, o que pode fazer com que o treinamento seja interrompido antes do momento exato.

  A solução SVM é bastante sensível ao *kernel* escolhido e também aos seus parâmetros. Neste caso particular, foram treinadas várias SVMs até se chegar a uma boa solução ajustando-se os valores da variância do *kernel* RBF e do limite para os multiplicadores de Lagrange.

A seguir, o segundo problema de classificação é abordado.

6.1.2 Exemplo 2

Este problema de classificação de duas classes tem como caracteristica a norma não muito baixa da solução, em torno de 20. No primeiro exemplo, a melhor solução tem norma aproximadamente igual a 1, portanto a solução de alta capacidade de generalização é mais complexa que no exemplo anterior.

Considere os seguintes conjuntos:

$$\begin{aligned} C_1 &= \{(x_1, x_2) | (x_1^2 + x_2^2) \leq 0.65\} \\ C_2 &= \{(x_1, x_2) | 0.35 \leq (x_1^2 + x_2^2) \leq 2.15\} \\ C_3 &= \{(x_1, x_2) | (x_1^2 + x_2^2) \geq 1.85\} \end{aligned}$$

O vetor de entrada é formado pelas variáveis x_1 e x_2 com distribuição normal de média zero e variâncias iguais à 0.5^2, 1.5^2 e 2^2 para os conjuntos C_1, C_2 e C_3 respectivamente. A Classe 1 é constituída pelo conjunto formado por $C_1 \cup C_3$ e a classe 2 é constituída pelo conjunto C_2. A saída desejada é dada pela Equação (6.6).

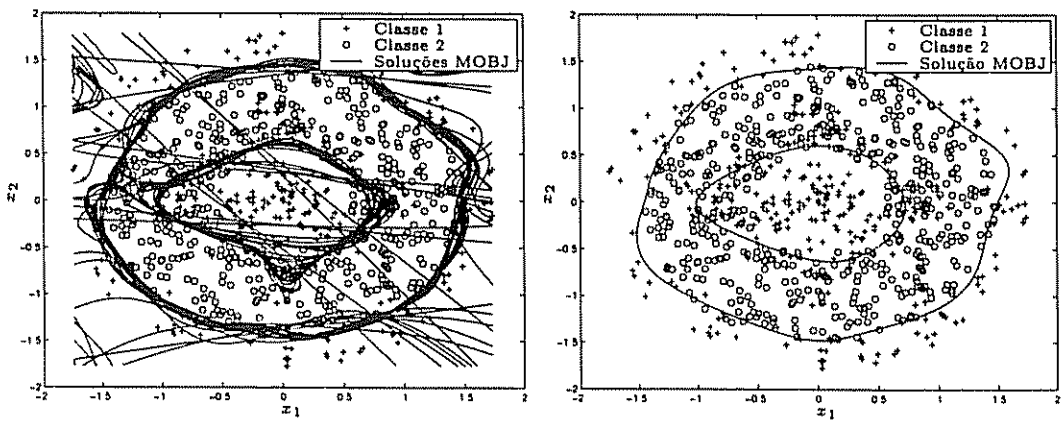
$$d = \begin{cases} +1 \ \ se \ \ x \in C_1 \cup C_3 \\ -1 \ \ se \ \ x \in C_2 \end{cases} \tag{6.6}$$

Ao conjunto de treinamento foram adicionados padrões ruidosos para verificar a insensibilidade dos métodos a estes. Os padrões ruidosos foram feitos com médias em $\mu_{r_1} = (0.75, 0)$, $\mu_{r_2} = (0, -0.75)$, $\mu_{r_3} = (-0.75, 0)$ e $\mu_{r_4} = (0, 0.75)$ e as variâncias foram feitas todas iguais a 0.5^2. A partir dos conjuntos de treinamento e validação assim constituídos, foram obtidas as soluções utilizando-se os diferentes algoritmos. A Tabela 6.3 traz os parâmetros de treinamento utilizados em cada caso.

Tabela 6.3: Parâmetros das redes treinadas para Exemplo 2

Parâmetro	BP	WD	OBD	ES	CV	SVM	MOBJ
λ	–	0.5	–	–	–	–	–
M_{fase_1}	250	400	100	100	50	–	–
M_{fase_2}	–	–	10	–	–	–	–
N_T	850	850	850	850	850	850	850
N_V	–	–	425	425	425	–	425
Kernel	–	–	–	–	–	rbf	–
C	–	–	–	–	–	0.6	–
σ^2 do *Kernel*	–	–	–	–	–	5	–
N_N	30	30	30	30	30	–	30
ζ	–	–	–	–	–	–	30
$\delta\varepsilon$	–	–	–	–	–	–	1

As ilustrações a seguir referem-se ao problema de classificação proposto no exemplo 2. Todas as soluções MOBJ são apresentadas na Figura 6.11(a) e a melhor solução MOBJ pode ser vista na Figura 6.11(b).



(a) Soluções MOBJ (b) Solução MOBJ escolhida

Figura 6.11: Problema de classificação. Exemplo 2 - Soluções MOBJ

- **Comentário das Figuras 6.11(a) e 6.11(b):**

  Na Figura 6.11(a) podem ser vistas as 30 soluções MOBJ geradas. Estas soluções foram obtidas com diferença de norma entre elas de 1. A solução MOBJ foi obtida a partir de um conjunto de treinamento com 850 padrões e de um conjunto de validação com 425 padrões. Mais uma vez pode-se notar a presença de soluções sub-ajustadas até soluções super-ajustadas.

  Na Figura 6.11(b) é mostrada a melhor solução MOBJ, escolhida no conjunto Pareto-ótimo de 30 soluções através de teste de generalização utilizando um conjunto com 425 padrões. Note-se que os ruídos normais adicionados nos pontos de média $\mu_{r_1} = (0.75, 0)$, $\mu_{r_2} = (0, -0.75)$ e $\mu_{r_4} = (0, 0.75)$ foram completamente ignorados e o ruído adicionado no ponto de média $\mu_{r_3} = (0, 0.75)$ foi parcialmente levado em consideração. Isto ocorreu devido ao número de padrões ruidosos naquela região ser maior que o número de padrões não ruidosos, ou seja, insuficiência de informação correta naquela região.

O conjunto Pareto-ótimo e a posição das soluções geradas pelos demais algoritmos no espaço dos objetivos podem ser vistos na Figura 6.12.

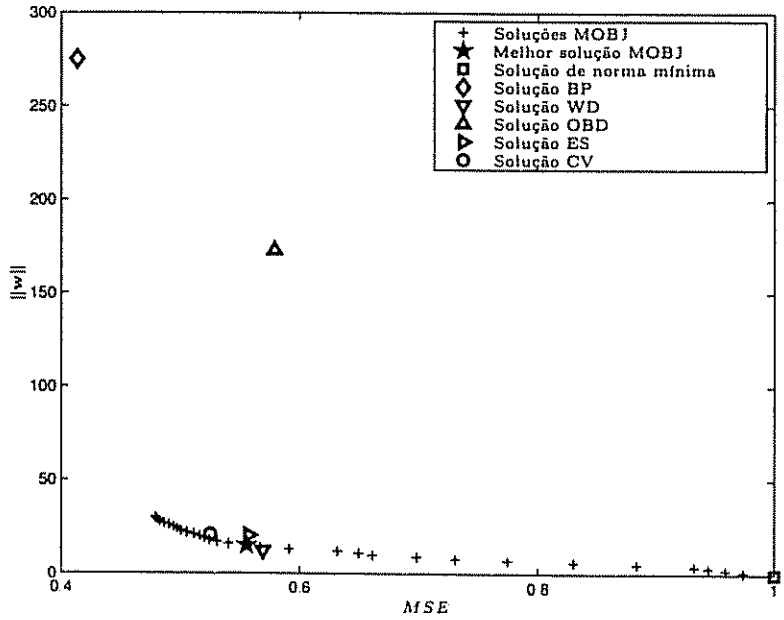


Figura 6.12: Conjunto Pareto-ótimo e demais soluções no espaço dos objetivos - Exemplo 2

- **Comentário da Figura 6.12:**

  Nesta Figura é mostrado o conjunto Pareto-ótimo para o problema e a melhor solução MOBJ escolhida no processo de decisão é denotada por uma estrela ($\star$).

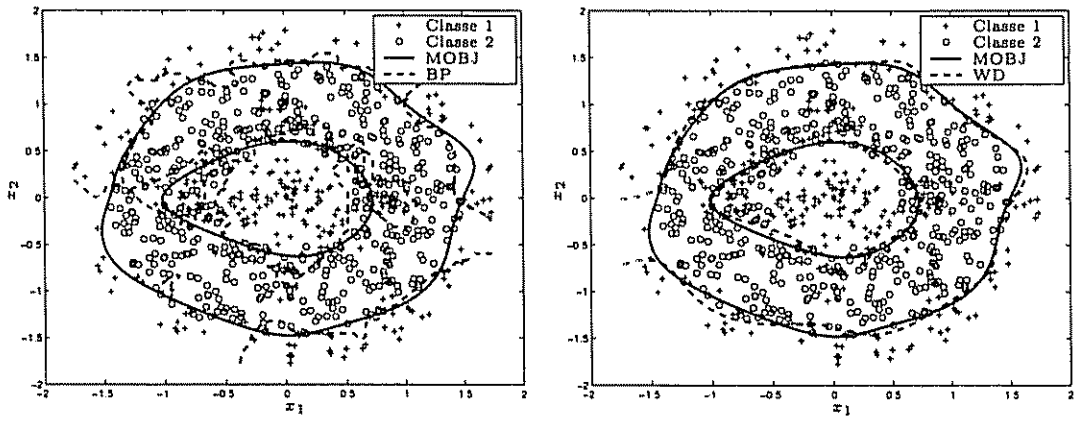
  As soluções dos demais algoritmos com exceção das soluções BP e OBD foram mapeadas próximas à solução MOBJ. Isto sugere que as soluções BP e OBD irão retornar erros maiores para os padrões de teste. Na Tabela 6.4 pode-se verificar quantitativamente este comportamento.

As curvas de decisão correspondentes a cada algoritmo de treinamento são ilustradas nas Figuras 6.13–6.15.

- **Comentário das Figuras 6.13–6.15:**

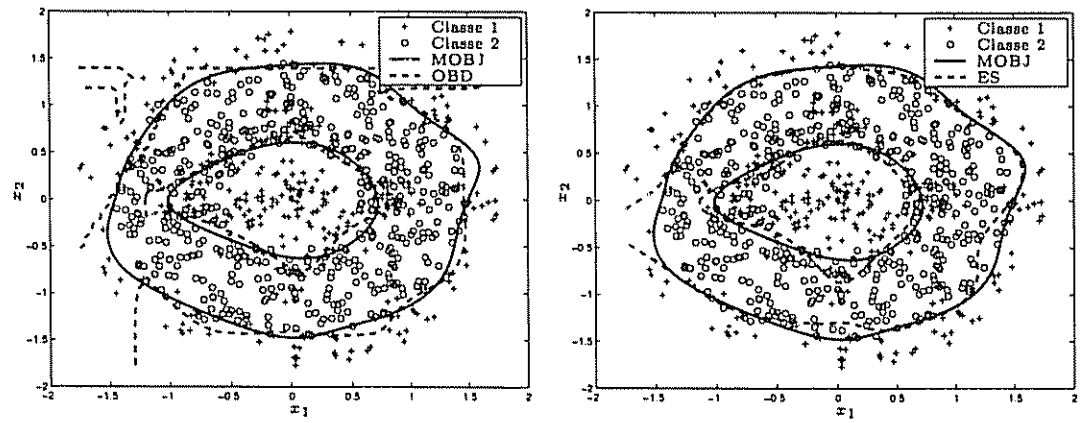
  Nas Figuras 6.13–6.15 são ilustradas as curvas de decisão de cada algoritmo. Note-se que para os demais algoritmos as curvas de decisão apresentaram distorções em relação à solução MOBJ. Alguns métodos além de mapearem parcialmente o ruído, apresentaram um abertura da curva no lado esquerdo, significando que para estes algoritmos, a informação contida nos conjuntos de treinamento e validação não foi suficiente.

As superfícies mapeadas pelas redes BP e MOBJ são ilustradas nas Figuras 6.16(b) e 6.16(a).



(a) Solução MOBJ e BP (b) Solução MOBJ e WD

Figura 6.13: Comparações com as soluções BP e WD para o exemplo 2



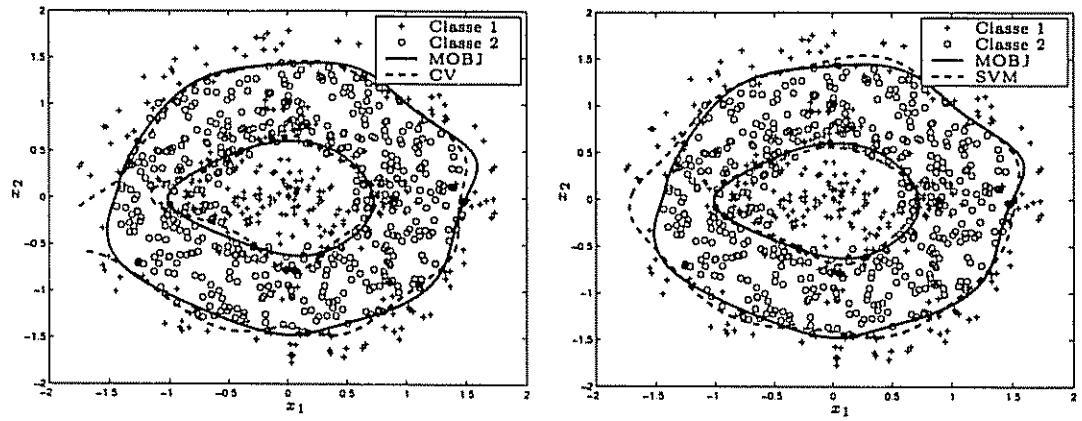
(a) Solução MOBJ e OBD (b) Solução MOBJ e ES

Figura 6.14: Comparações com as soluções OBD e ES para o exemplo 2

- **Comentário das Figuras 6.16(a) e 6.16(b):**

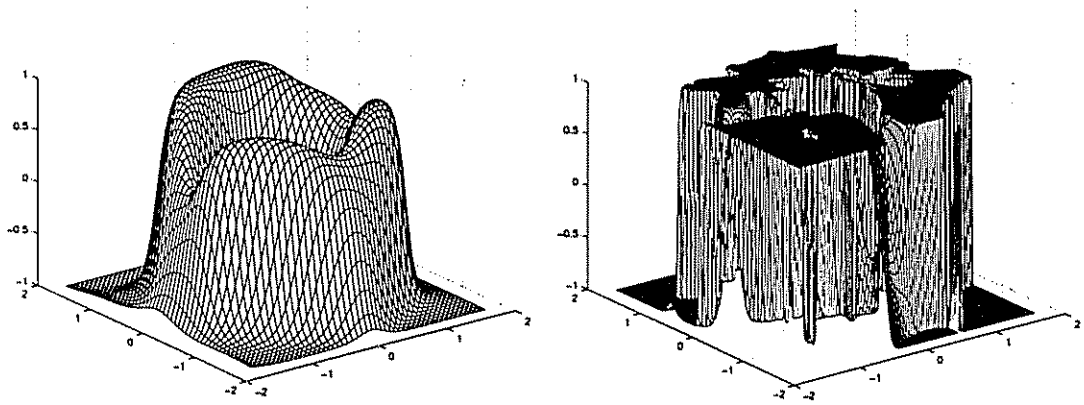
  Nestas figuras estão ilustradas as superfícies mapeadas pelos algoritmos MOBJ e BP. Mais uma vez é importante dizer que a rede com topologia $2 \times 30 \times 1$ é bastante complexa para a tarefa em questão sendo capaz de mapear funções altamente complexas como a mostrada na Figura 6.16(b) e ainda assim o algoritmo MOBJ foi capaz de mapear uma função relativamente mais suave como pode ser visto na Figura 6.16(a) não permitindo o super-ajuste aos dados utilizados no processo de treinamento.

A Tabela 6.4 traz os percentuais de acerto e os desvios padrão para as soluções obtidas através dos algoritmos abordados. Para o cálculo dos percentuais de acerto foram utilizados 100 conjuntos de dados constituídos de 850 padrões cada.



(a) Solução MOBJ e CV (b) Solução MOBJ e SVM

Figura 6.15: Comparações com as soluções CV e SVM para o exemplo 2



(a) Solução MOBJ (b) Solução BP

Figura 6.16: Superfícies de decisão dos métodos BP e MOBJ para o exemplo 2

Na Tabela 6.4 pode-se observar os resultados do teste de generalização para as diversas soluções e a solução MOBJ é a que resultou em maiores acertos. Para os demais métodos várias tentativas de treinamento foram feitas até que um nível de desempenho satisfatório fosse alcançado.

- **Comentários finais da Subseção 6.1.2:**

  Para obtenção de uma solução mais complexa é maior a necessidade de parâmetros bem ajustados e portanto, maior deve ser o conhecimento sobre o problema por parte do usuário quando se trabalha com os demais métodos citados. Para o algoritmo MOBJ, a complexidade da função a ser mapeada pouco influencia no processo de aprendizagem e o nível de rejeição ao ruído é bastante elevado.

A seguir é abordado um problema de classificação mais complexo que os dois anteri-

Tabela 6.4: Acertos para os padrões de teste.(100 conjuntos diferentes - 850 padrões cada)

Algoritmos	Acerto	σ
Backpropagation	66.6933%	±1.9745%
Weight Decay	75.3409%	±1.9876%
Optimal Brain Damage	73.4839%	±1.7514%
Early Stopping	76.1540%	±1.8470%
Cross-Validation	77.1718%	±1.8036%
SVM	75.0263%	±1.5290%
MOBJ	77.8673%	±1.6903%

ormente tratados.

6.1.3 Exemplo 3

Este problema de classificação denominado de "Tabuleiro de Xadrez 4 × 4" foi extraído do artigo [Barros et al., 2000]. São duas classes com superposição e o conjunto de treinamento é constituído de 960 padrões. A Figura 6.17 ilustra a distribuição das classes.

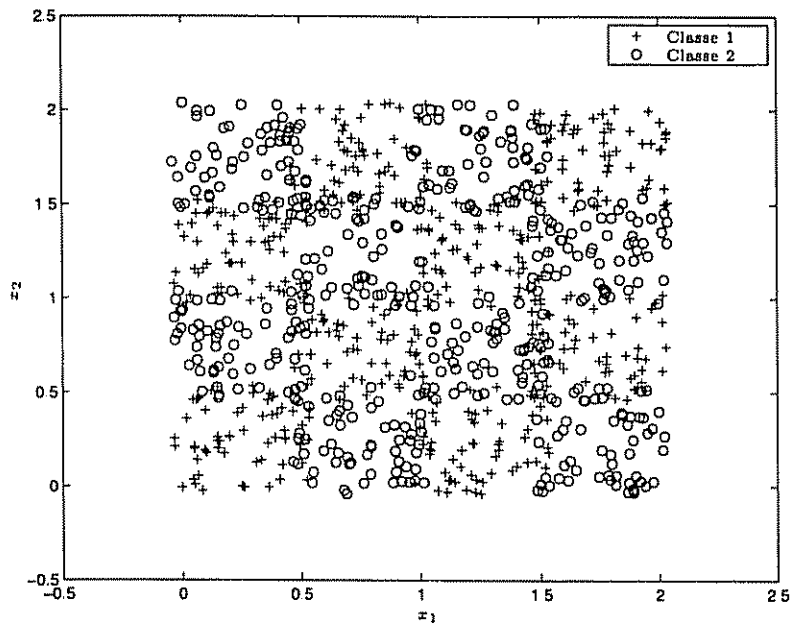


Figura 6.17: Ilustração do problema de classificação do exemplo 3

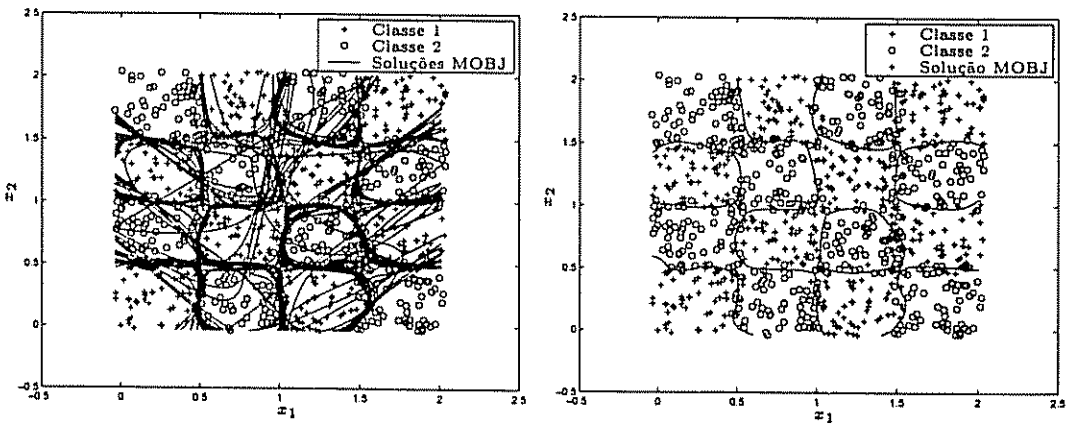
Os algoritmos utilizados tiveram seus parâmetros ajustados conforme mostra a Tabela 6.5. Todas as soluções MOBJ são apresentadas na Figura 6.18(a) e a melhor solução MOBJ pode ser vista na Figura 6.18(b).

- **Comentário das Figuras 6.18(a) e 6.18(b):**

  Na Figura 6.18(a) podem ser vistas as 30 soluções MOBJ geradas. Estas soluções

Tabela 6.5: Parâmetros das redes treinadas para Exemplo 3

Parâmetro	BP	WD	OBD	ES	CV	SVM	MOBJ
λ	–	0.09	–	–	–	–	–
M_{fase_1}	1000	200	120	200	200	–	–
M_{fase_2}	–	–	15	–	–	–	–
N_T	960	960	960	960	960	960	960
N_V	–	–	960	960	960	–	960
Kernel	–	–	–	–	–	rbf	–
C	–	–	–	–	–	7	–
σ^2 do *Kernel*	–	–	–	–	–	0.25	–
N_N	30	30	30	30	30	–	30
ζ	–	–	–	–	–	–	30
$\delta\varepsilon$	–	–	–	–	–	–	2



(a) Soluções MOBJ (b) Solução MOBJ escolhida

Figura 6.18: Problema de classificação. Exemplo 3 - Soluções MOBJ

foram obtidas com diferença de norma entre elas de 2. Na Figura 6.18(b) é mostrada a melhor solução MOBJ, escolhida no conjunto Pareto-ótimo de 30 soluções através de teste de generalização utilizando um conjunto com 960 padrões.

O conjunto Pareto-ótimo gerado, a posição da solução MOBJ escolhida e as soluções dos demais métodos podem ser vistos na Figura 6.19. Mais uma vez somente as soluções BP e OBD ficaram distantes da solução MOBJ. As demais soluções podem ser vistas com mais detalhe na ampliação mostrada na Figura 6.20.

- **Comentário da Figura 6.19:**

  Nesta Figura é mostrado o conjunto Pareto-ótimo para o problema e a melhor solução MOBJ escolhida no processo de decisão é denotada por uma estrela ($\star$). As soluções dos demais algoritmos com exceção das soluções BP e OBD foram mapeadas próximas à solução MOBJ. Este fato se explica através do número de

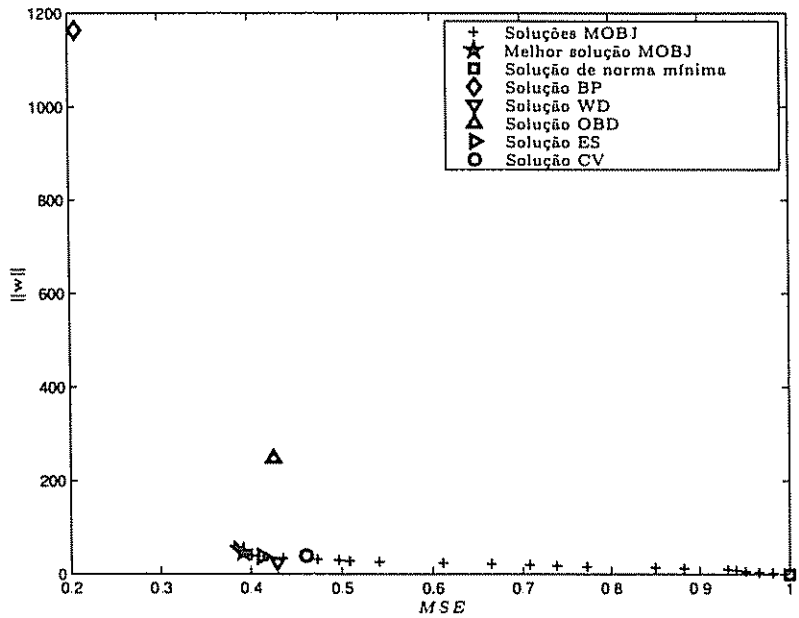


Figura 6.19: Conjunto Pareto-ótimo e demais soluções no espaço dos objetivos - Exemplo 3

padrões utilizados para o treinamento e a validação que neste caso foi suficiente para representar bem o problema fazendo com que todos os métodos atingissem soluções próximas.

As curvas de decisão correspondentes a cada algoritmo de treinamento são ilustradas nas Figuras 6.21–6.23.

- **Comentário das Figuras 6.21–6.23:**

  Nestas figuras são ilustradas as curvas de decisão para cada algoritmo sendo estas muito próximas com exceção da solução BP.

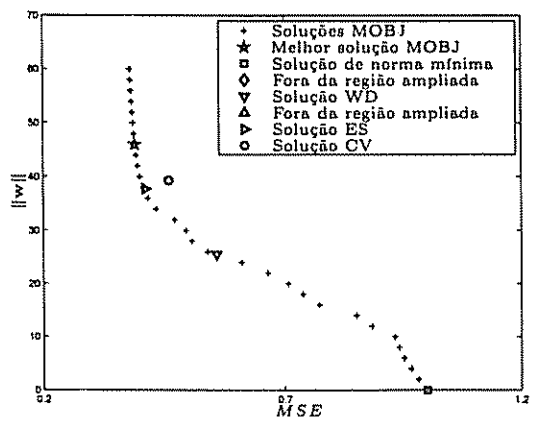
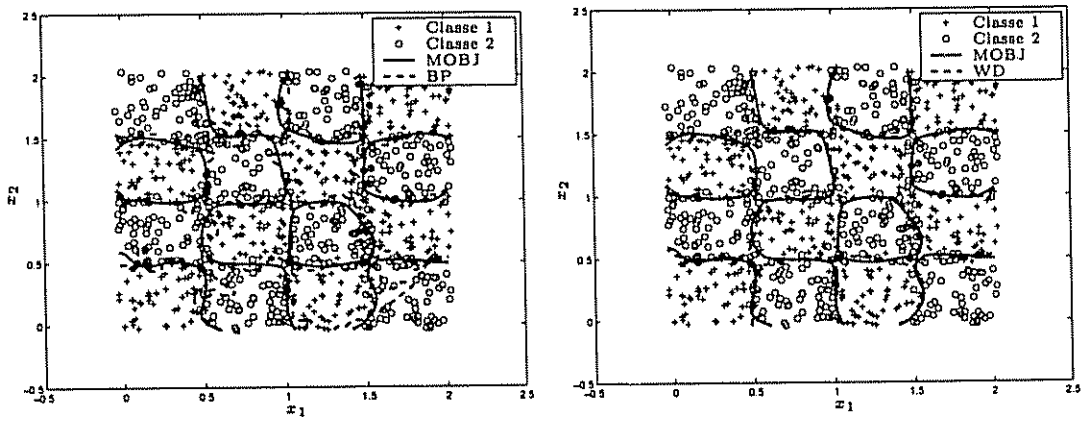
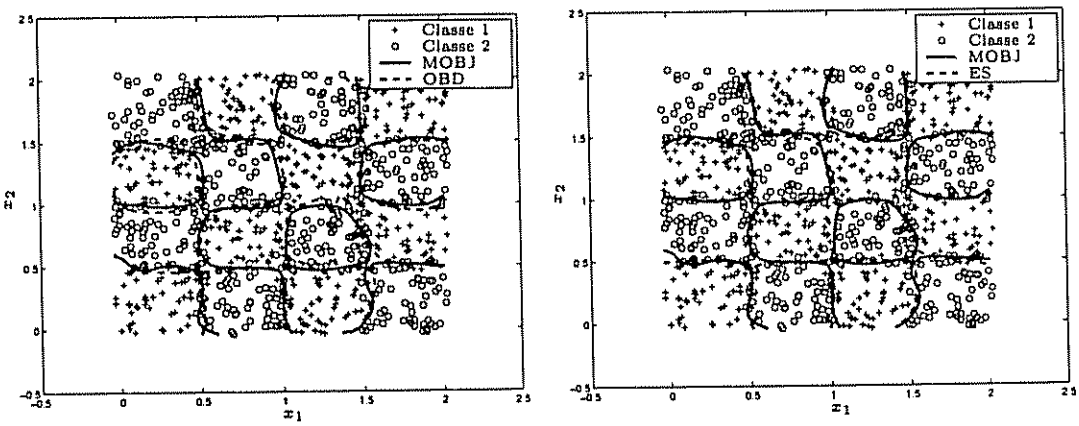


Figura 6.20: Ampliação da Figura 6.19 - Exemplo 3



(a) Solução MOBJ e BP (b) Solução MOBJ e WD

Figura 6.21: Comparações com as soluções BP e WD para o exemplo 3



(a) Solução MOBJ e OBD (b) Solução MOBJ e ES

Figura 6.22: Comparações com as soluções OBD e ES para o exemplo 3

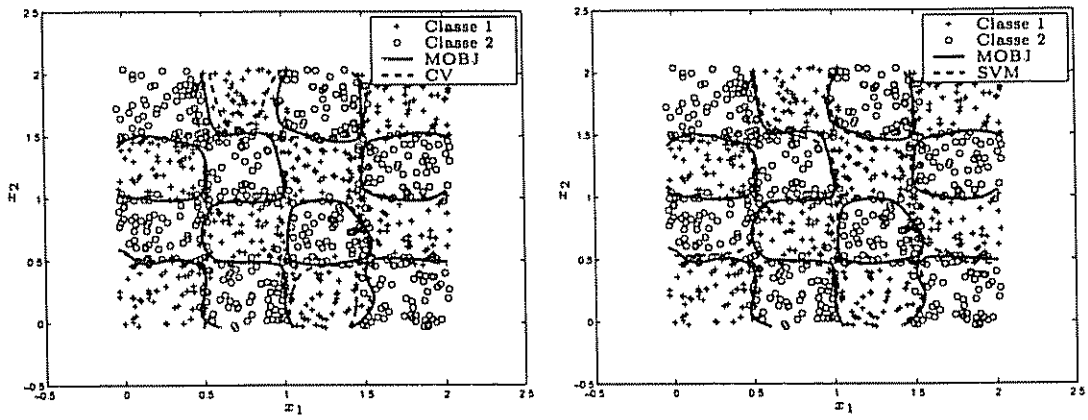
As superfícies mapeadas pelas redes BP e MOBJ são ilustradas pelas Figuras 6.24(b) e 6.24(a).

- **Comentário das Figuras 6.24(a) e 6.24(b):**

  Nestas figuras estão ilustradas as superfícies mapeadas pelos algoritmos MOBJ e BP sendo a solução retornada pelo algoritmo MOBJ (6.24(b)) relativamente mais suave que a solução BP (6.24(a)).

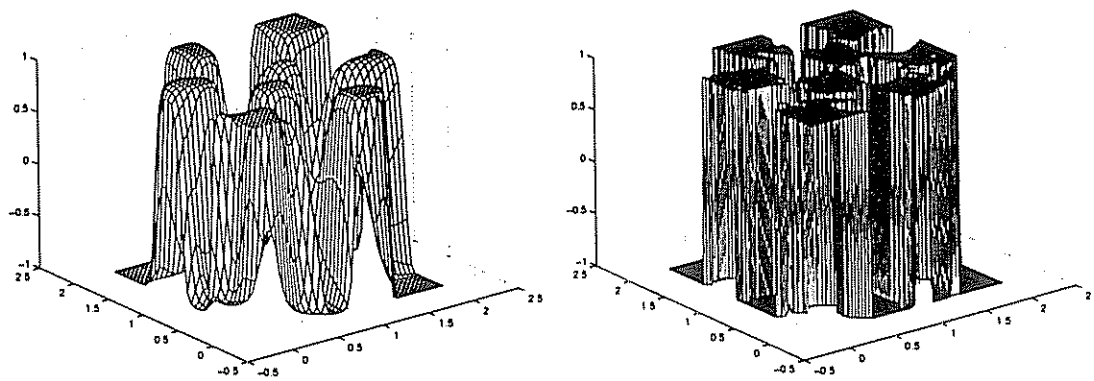
A Tabela 6.6 traz os percentuais de acerto e os desvios padrão para as soluções obtidas através dos algoritmos abordados. Para o cálculo dos percentuais de acerto foram utilizados 100 conjuntos de dados constituídos de 960 padrões cada.

Note-se que devido à proximidade das soluções no espaço dos objetivos mostrada na Figura 6.19, a capacidade de generalização entre as soluções é semelhante. Analisando



(a) Solução MOBJ e CV (b) Solução MOBJ e SVM

Figura 6.23: Comparações com as soluções CV e SVM para o exemplo 3



(a) Solução MOBJ (b) Solução BP

Figura 6.24: Superfícies de decisão dos métodos BP e MOBJ para o exemplo 3

os indices mostrados na Tabela 6.6 pode-se perceber que as soluções retornadas pelos algoritmos com exceção do BP estão dentro da região de equivalência descrita no Capítulo 5, o que na prática significa que todas elas são soluções de alta capacidade de generalização. Os conjuntos de dados utilizados possuem muita informação sobre a função que se deseja mapear e portanto, todos os métodos apresentaram bons resultados. A diferença entre eles está no esforço necessário para que cada um encontre sua respectiva solução. A seguir são feitos alguns comentários sobre os resultados obtidos.

- **Comentários finais da Subseção 6.1.2:**

  O algoritmo MOBJ se mostrou insensível ao aumento da complexidade do problema, alcançando uma solução com boa capacidade de generalização, compatível com os demais algoritmos testados. Este exemplo serviu para mostrar que o método

Tabela 6.6: Acertos para os padrões 100 conjuntos de teste (960 padrões).

Algoritmos	Acertos	σ
Backpropagation	75.9635%	±1.3103%
Weight Decay	77.8958%	±1.2566%
Optimal Brain Damage	77.2031%	±1.2293%
Early Stopping	77.7312%	±1.3177%
Cross-Validation	77.7604%	±1.3257%
SVM	77.7583%	±1.3525%
MOBJ	77.9667%	±1.2863%

proposto é capaz de alcançar soluções mesmo quando o problema é complexo.

A seguir são mostrados alguns resultados obtidos através de vários métodos para um problema de análise de risco de crédito onde é feita também uma comparação com os resultados do algoritmo MOBJ para o mesmo problema.

6.1.4 Aplicação à análise de crédito

Um problema de avaliação de risco de crédito é abordado em [Lacerda et al., 2001] onde redes do tipo RBF são treinadas para resolver tal problema de classificação. A base de dados utilizado nos testes foi obtida na *UCI Machine Learning Repository* [Blake and Merz, 1998] e consiste em 690 padrões com 51 entradas e duas saídas. Destes 690 padrões, 50% das amostras constituem o conjunto de treinamento, 25% o conjunto de validação e 25% o conjunto de teste. Para comparação, três redes com topologia $51 \times 20 \times 2$ foram treinadas com o algoritmo MOBJ utilizando-se de três conjuntos diferentes de treinamento, validação e teste. Os resultados para cada solução MOBJ são mostrados na Tabela 6.7 e a média dos resultados é comparada com os resultados obtidos em [Lacerda et al., 2001] na Tabela 6.8.

Tabela 6.7: Acertos para os três conjuntos diferentes considerando cada rede.

	Acertos
Solução MOBJ 1	90.1734%
Solução MOBJ 2	89.5954%
Solução MOBJ 3	91.9075%
Média	90.5587%
Desvio padrão	±1.2033

Para resolver o problema de análise de risco em [Lacerda et al., 2001], além do algoritmo RBF Evolucionário proposto, foram feitos testes com vários outros algoritmos como redes RBF com os algoritmos *batch K-means* [Lloyd, 1982], *on-line K-means* [MacQueen, 1967], *the iterative optimization (IO)* [Duda and Hart, 1973], *depth-first search (DF)* [Ismail et al., 1984], duas combinação dos algoritmos IO e DF (DFIO)

[Ismail and Kamel, 1989] e (IODF) [Ismail and Kamel, 1989] e *optimal adaptive K-means* [Chinrungrueng and Séquin, 1995] para os cálculos dos centros, alguns algoritmos construtivos como *Cascade Correlation* [Fahlman and Lebiere, 1990] *Tower* e *Pyramid* [Parekh et al., 1987] além das *Support Vector Machines*. Os resultados obtidos são aqui comparados com o resultado conseguido com o algoritmo MOBJ proposto.

Tabela 6.8: Comparação com os resultados apresentados em [Lacerda et al., 2001].

Algoritmos	Acertos	Desvio padrão
(RBF) *batch K-means*	16.67%	±3.87
(RBF) DF	16.28%	±2.54
(RBF) IO	17.83%	±3.96
(RBF) DFIO	16.67%	±4.28
(RBF) IODF	17.25%	±4.44
(RBF) *on-line K-means*	16.86%	±4.39
(RBF) *Optimal*	15.89%	±4.66
(RBF) GA	13.95%	±3.53
Backpropagation	17.05%	±1.77
Cascade correlation	18.02%	±3.03
Tower	14.73%	±3.20
Pyramid	16.86%	±2.10
SVM	16.67%	±2.63
MOBJ	9.44%	±1.20

Como pode ser visto na Tabela 6.8, o índice de acerto conseguido com o algoritmo MOBJ superou todos os demais métodos testados. A seguir são abordados alguns problemas de regressão.

6.2 Problemas de Regressão

Nesta seção são tratados dois problemas de regressão onde redes 1-u_h-1 são treinadas. Os conjuntos de treinamento, validação e teste utilizados em cada caso são constituídos de N_T, N_V e N_E amostras respectivamente feitas sobre as funções geradoras dadas pelas Equações (6.7) e (6.8) e a estas amostras foi adicionado um termo de ruído normalmente distribuído com média zero e variância $\sigma^2 = 0.2^2$.

$$f_1(x) = \sin(\pi x) \tag{6.7}$$

$$f_2(x) = \frac{(x-2)(2x+1)}{(1+x^2)} \tag{6.8}$$

Para todos os algoritmos abordados procurou-se manter a mesma topologia da rede MLP a ser treinada. Os neurônios da camada escondida tinham como função de ativação a função tangente hiperbólica ($f_a(\cdot) = tanh(\cdot)$) e os nodos de saída possuíam função de

ativação linear. A topologia das redes adotada é tal que a complexidade das mesmas excede à complexidade das tarefas de aprendizagem em questão.

O algoritmo utilizado para obtenção das soluções *Backpropagation* (BP) foi o de Levenberg-Marquardt [Hagan and Menhaj, 1994] sendo este algoritmo também utilizado para ajuste dos pesos nos métodos *Early Stopping* (ES) e *10-Fold Cross Validation* (CV). Para o treinamento das redes através do algoritmo *Weight Decay* (WD) houve a necessidade de se fazer várias tentativas para encontrar um ajuste adequado para a constante de queda λ e o algoritmo utilizado é o encontrado no *Toolbox* de identificação utilizando RNAs [Nørgaard, 1997]. O algoritmo *Optimal Brain Damage* (OBD) também faz parte deste *toolbox*. Com este algoritmo OBD são treinadas várias redes com diferentes topologias iniciando a partir da topologia proposta e ao final deste processo, utiliza-se um conjunto de validação para escolha da melhor solução OBD. Para este algoritmo de *pruning*, após a eliminação de um peso ocorre uma segunda fase de treinamento com o número de iterações definido por M_{fase_2}.

Para o treinamento das redes através do algoritmo SVM, tem-se que escolher o *kernel* a ser utilizado, neste caso foi utilizado um *kernel* RBF além do ajuste dos parâmetros do mesmo que neste caso é a variância σ^2 e finalmente, ajustar o limite para os multiplicadores de Lagrange C. Para este algoritmo também houve a necessidade de se fazer várias tentativas até que um conjunto adequado de parâmetros de treinamento fosse encontrado.

Para o treinamento das RNAs através do algoritmo MOBJ, apenas dois parâmetros devem ser fornecidos pelo usuário sendo que a solução final não é muito sensível à estes. O primeiro parâmetro é a quantidade ζ de soluções MOBJ a serem geradas e o segundo parâmetro determina o incremento $\delta\varepsilon$ que será dado na norma para cada solução. Influências destes parâmetros na solução final são discutidas no Capítulo 5.

As regressões feitas pelo método MOBJ são comparadas com as soluções retornadas pelos demais algoritmos abordados. Na seção seguinte, o primeiro problema de regressão é tratado.

6.2.1 Exemplo 1: função $f_1(x)$

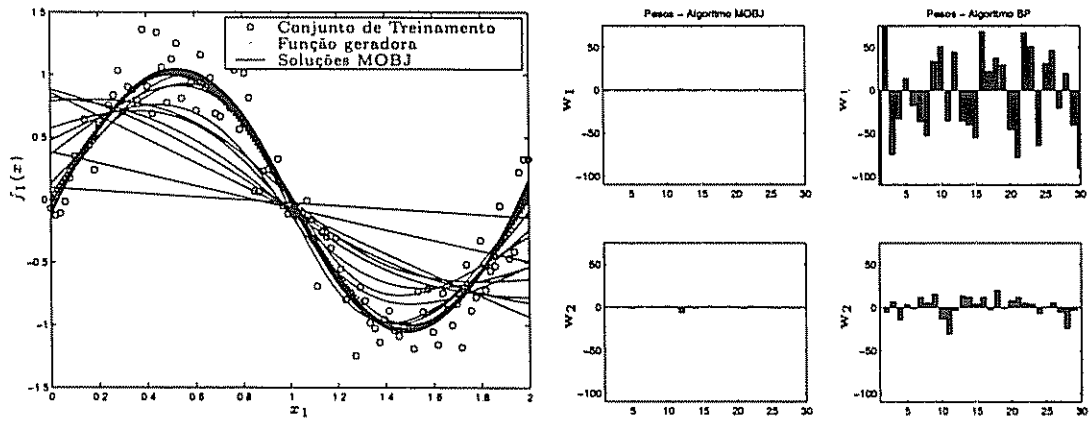
Para a regressão da função $f_1(x)$, RNAs com topologia $1 \times 30 \times 1$ são treinadas pelos diversos métodos abordados sendo que estes tiveram seus parâmetros de treinamento ajustados segundo a Tabela 6.9.

Foram geradas 20 soluções MOBJ com diferença de norma entre elas de 0.5. Todas as soluções MOBJ podem ser vistas na Figura 6.25(a) e os pesos da melhor solução MOBJ escolhida pelo decisor é mostrado na Figura 6.25(b), onde também pode ser visto o vetor de pesos referente à solução BP.

- **Comentário das Figuras 6.25(a) e 6.25(b) :**

Tabela 6.9: Parâmetros das redes treinadas para regressão. Exemplo 1

Parâmetro	BP	WD	OBD	ES	CV	SVM	MOBJ
λ	–	0.26	–	–	–	–	–
M_{fase_1}	400	200	100	200	100	–	–
M_{fase_2}	–	–	10	–	–	–	–
N_T	100	100	50	100	100	100	100
N_V	–	–	50	50	50	–	50
$Kernel$	–	–	–	–	–	rbf	–
C	–	–	–	–	–	2	–
σ^2 do $Kernel$	–	–	–	–	–	0.5	–
N_N	30	30	30	30	30	–	30
ζ	–	–	–	–	–	–	20
$\delta\varepsilon$	–	–	–	–	–	–	0.5



(a) Soluções MOBJ (b) Vetores de pesos MOBJ e BP

Figura 6.25: Problema de regressão. Exemplo 1

Na Figura 6.25(a) são mostradas as 20 soluções MOBJ geradas e na Figura 6.25(b) pode-se ver a magnitude dos elementos dos vetores de pesos $\mathbf{w}_1$ e $\mathbf{w}_2$ da melhor solução MOBJ e da solução BP. Como a escala foi mantida a mesma para ambos os vetores, quase não se pode perceber a magnitude dos pesos correspondente à solução MOBJ. Como a solução deste problema é de norma baixa, os pesos da solução MOBJ possuem valores muito pequenos em relação a solução BP.

O conjunto Pareto-ótimo bem como as soluções dos demais métodos no espaço dos objetivos podem ser vistos na Figura 6.26.

- **Comentário da Figura 6.26:**

Nesta figura pode-se notar que a solução MOBJ denotada por ($\star$) possui norma baixa em relação às soluções BP, ES, CV e OBD, as quais apresentaram soluções de normas bem mais elevadas e consequentemente, as soluções destes métodos apresentaram

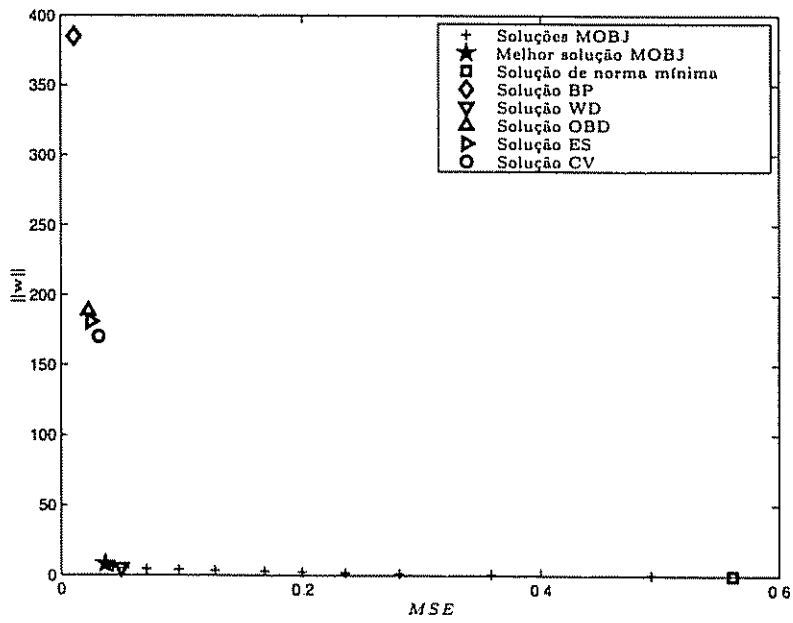


Figura 6.26: Conjunto Pareto-ótimo e demais soluções no espaço dos objetivos - Exemplo 1

algum grau de super-ajuste aos dados. A solução WD foi a que mais se aproximou da solução MOBJ. A implicação destas posições na qualidade das soluções é discutida a seguir.

- **Comentário da Figura 6.27:**

  Nesta figura pode-se verificar o comportamento das soluções WD quando foi feita uma variação paramétrica na constante de queda dos pesos que assumiu valores de acordo com o conjunto $\{0.005,\ 0.05,\ 0.1,\ 0.2, \ldots, 1.0\}$. Note-se que não há uniformidade na distribuição das soluções no espaço dos objetivos. Para o algoritmo MOBJ, um incremento de $\delta\varepsilon$ na norma é seguido de um decremento do erro mas, para o algoritmo WD, mesmo que λ varie uniformemente, um incremento desta implica necessariamente em redução de norma mas não necessariamente em aumento ou queda de erro, ou seja, não há nenhuma uniformidade no comportamento das soluções no espaço dos objetivos. Em outras palavras, quando se faz um ajuste no valor da constante de queda, não se sabe o quanto este ajuste refletirá na norma e no erro da solução WD.

As Figuras 6.28–6.30 ilustram a melhor solução MOBJ escolhida pelo decisor, a função geradora e as soluções dos demais algoritmos.

- **Comentário das Figuras 6.28–6.30:**

  Nestas figuras pode-se notar a qualidade da solução MOBJ gerada em relação aos

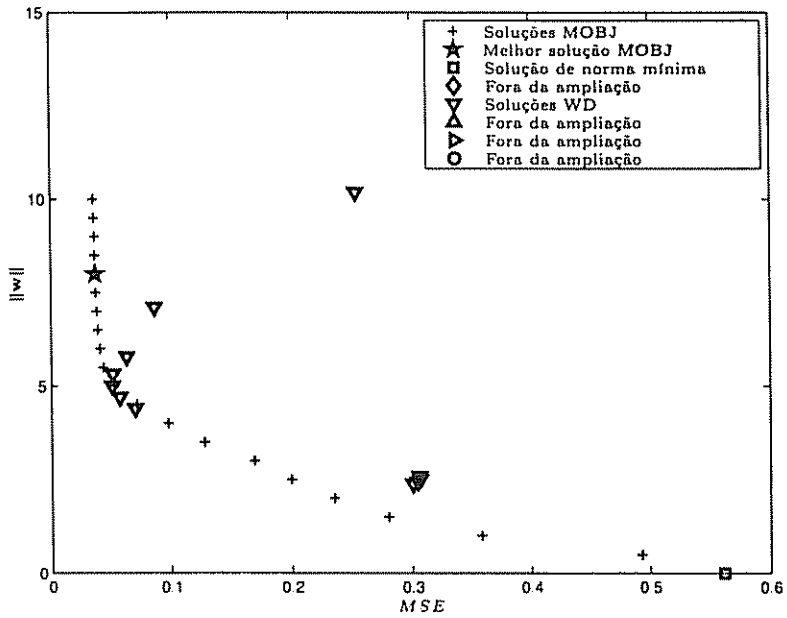


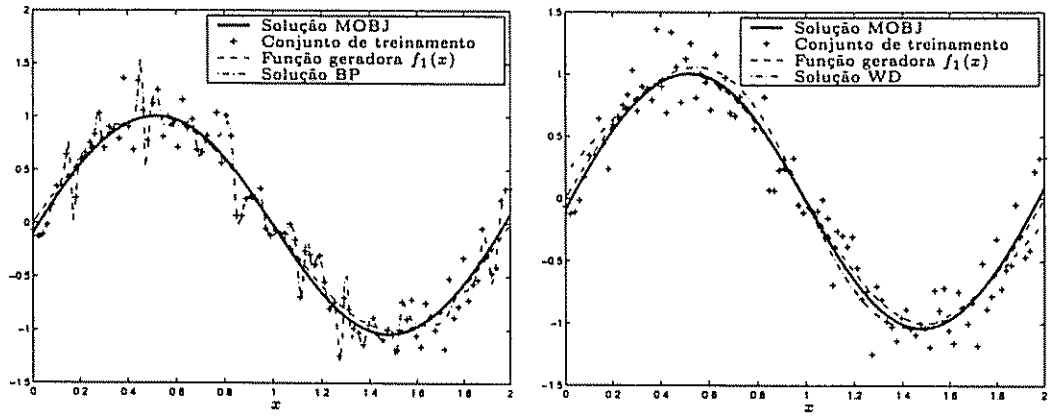
Figura 6.27: Conjunto Pareto-ótimo e demais soluções WD no espaço dos objetivos - Exemplo 1

demais métodos. Com exceção dos algoritmos MOBJ, SVM e WD, os quais apresentaram bons resultados para regressão da função $f_1(x)$, os demais ficaram super-ajustados aos dados de treinamento. Isto se deve ao fato de que a topologia da rede adotada é bastante complexa para o problema e ao número pequeno de amostras nos conjuntos de treinamento ($N_T = 100$) e validação ($N_V = 50$). Os métodos ES, CV e OBD falharam na regressão, pois não deveriam apresentar *overfitting*.

Note-se também na Figura 6.28(b) que as extremidades da solução WD estão distantes da função geradora. Para este algoritmo, quando se tentou ajustá-lo para mapear bem tal região, a região central da curva apresentava erros elevados. Portanto, várias tentativas foram feitas e o resultado mostrado foi o melhor que se conseguiu.

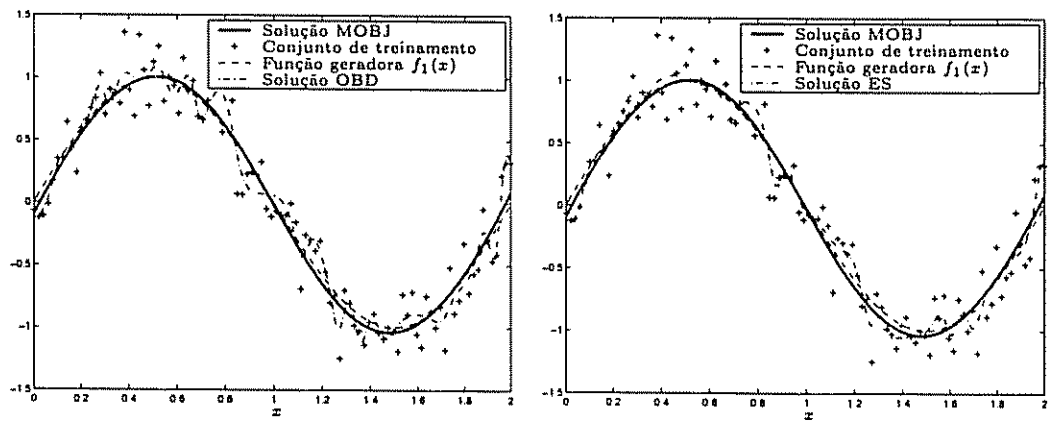
Nestas figuras pode-se notar também a implicação do fato das soluções dos métodos BP, WD, OBD, ES e CV estarem mais próximas ou mais distantes da solução MOBJ. Como a solução MOBJ possui alta capacidade de generalização, o fato de alguma outra solução estar próxima da solução MOBJ implica em apresentar melhor qualidade que as soluções mais distantes. Com exceção das soluções SVM e WD, todas as demais apresentaram um determinado grau de super-ajuste aos dados, e como pode-se notar na Figura 6.26, elas estão mais distantes da solução MOBJ.

Na Tabela 6.10 são mostrados os somatórios dos erros quadráticos (MSE) para todos os métodos abordados para fins de comparação com o algoritmo MOBJ. Estes erros foram calculados considerando um conjunto de teste constituído de 1000 padrões e corresponde



(a) Solução MOBJ e BP (b) Solução MOBJ e WD

Figura 6.28: Comparações com as soluções BP e WD para o exemplo 1



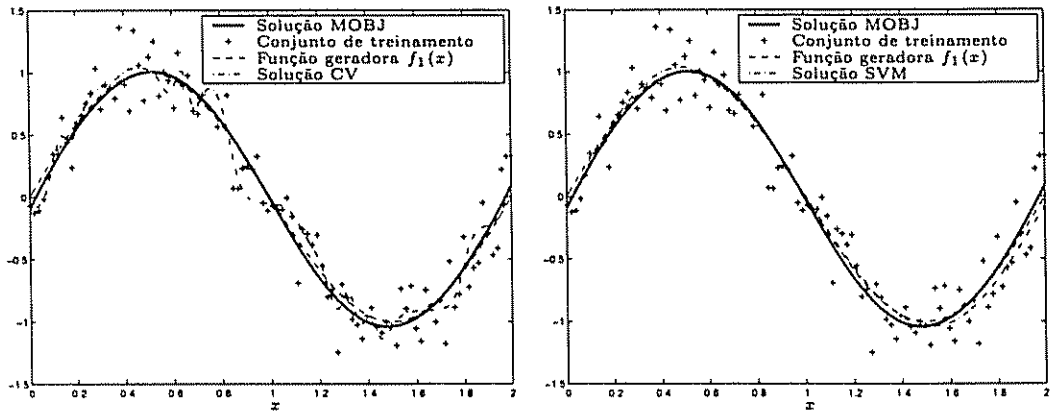
(a) Solução MOBJ e OBD (b) Solução MOBJ e ES

Figura 6.29: Comparações com as soluções ES e OBD para o exemplo 1

ao erro de generalização. A seguir, o segundo problema de regressão é tratado.

6.2.2 Exemplo 2: função $f_2(x)$

A função $f_2(x)$ é particularmente importante para se testar a capacidade de mapeamento dos métodos por apresentar regiões de suavidade, nos extremos, e uma região central mais íngreme. Aqui, o objetivo é testar se os métodos são capazes de fazer o mapeamento de todas as regiões da função. Logo, se a rede tiver complexidade suficientemente alta, conseguirá mapear a região íngreme mas pode apresentar *overfitting* na região onde ela deveria ser suave e por outro lado, se a rede for pouco complexa, consegue mapear bem a região suave da curva mas não mapeia a região íngreme. Para este problema foram utilizadas redes de topologia $1 \times 30 \times 1$ em todos os casos onde a topologia deve ser



(a) Solução MOBJ e CV (b) Solução MOBJ e SVM

Figura 6.30: Comparações com as soluções CV e SVM para o exemplo 1

Tabela 6.10: MSE para os padrões de teste.(100 conjuntos diferentes)

Algoritmos	MSE	σ
Backpropagation	0.0702	±0.0028
Weight Decay	0.0482	±0.0019
Optimal Brain Damage	0.0547	±0.0021
Early Stopping	0.0496	±0.0021
Cross-Validation	0.0522	±0.0022
SVM	0.0418	±0.0018
MOBJ	0.0419	±0.0018

determinada e os parâmetros de treinamento dos diversos métodos são listados na Tabela 6.11.

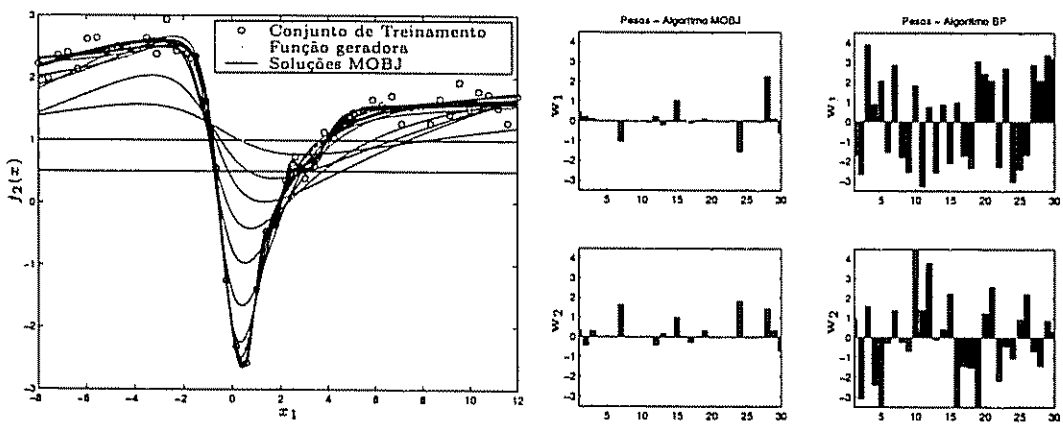
A Figura 6.31(a) ilustra as soluções MOBJ geradas e a Figura 6.31(b) ilustra o vetores de pesos da melhor solução MOBJ e da solução BP.

- **Comentário das Figuras 6.31(a) e 6.31(b):**

  Na Figura 6.31(a) são mostradas as 20 soluções MOBJ geradas e na Figura 6.31(b) são ilustradas as magnitudes dos pesos da melhor solução MOBJ e da solução BP. Pode-se notar nesta ilustração que alguns pesos correspondentes à solução MOBJ não tiveram suas magnitudes reduzidas. Isto significa que na abordagem multi-objetivo, apesar de se buscarem soluções de norma mínima para o problema, a redução da norma é feita reduzindo-se os pesos não significativos. Os pesos importantes para o mapeamento adequado da função geradora algumas vezes não sofrem grandes alterações de magnitude e podem às vezes até sofrer incrementos. Isto permite que a saída da rede seja suave em uma determinada faixa e íngreme em outras.

Tabela 6.11: Parâmetros das redes treinadas para regressão. Exemplo 2

Parâmetro	BP	WD	OBD	ES	CV	SVM	MOBJ
λ	–	0.9	–	–	–	–	–
M_{fase_1}	200	400	100	100	100	–	–
M_{fase_2}	–	–	10	–	–	–	–
N_T	50	50	50	50	50	50	50
N_V	–	–	25	25	25	–	25
N_E	1000	1000	1000	1000	1000	1000	1000
Kernel	–	–	–	–	–	rbf	–
C	–	–	–	–	–	1	–
σ^2 do *Kernel*	–	–	–	–	–	2	–
N_N	30	30	30	30	30	–	30
ζ	–	–	–	–	–	–	30
$\delta\varepsilon$	–	–	–	–	–	–	0.5



(a) Soluções MOBJ (b) Vetores de pesos MOBJ e BP

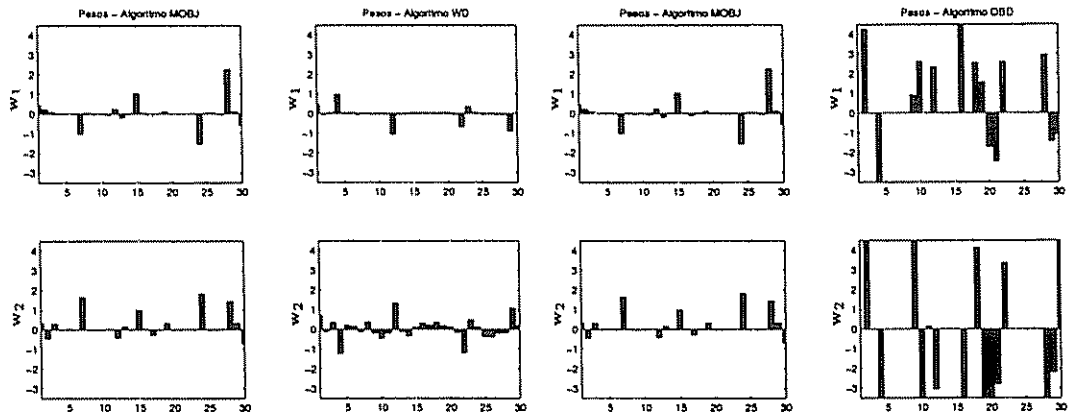
Figura 6.31: Problema de regressão. Exemplo 2

As Figuras 6.32(a)–6.33(b) ilustram os pesos obtidos com o algoritmo MOBJ e os pesos obtidos pelos algoritmos WD, OBD, ES e CV. Pode-se notar que os vetores correspondentes às soluções OBD, ES e CV possuem elementos com magnitudes mais elevadas principalmente aqueles que interligam o nível de entrada e à camada intermediária, denotados por $\mathbf{w}_1$.

A Figura 6.34 ilustra o conjunto Pareto-ótimo obtido bem como as soluções dos demais métodos no espaço dos objetivos.

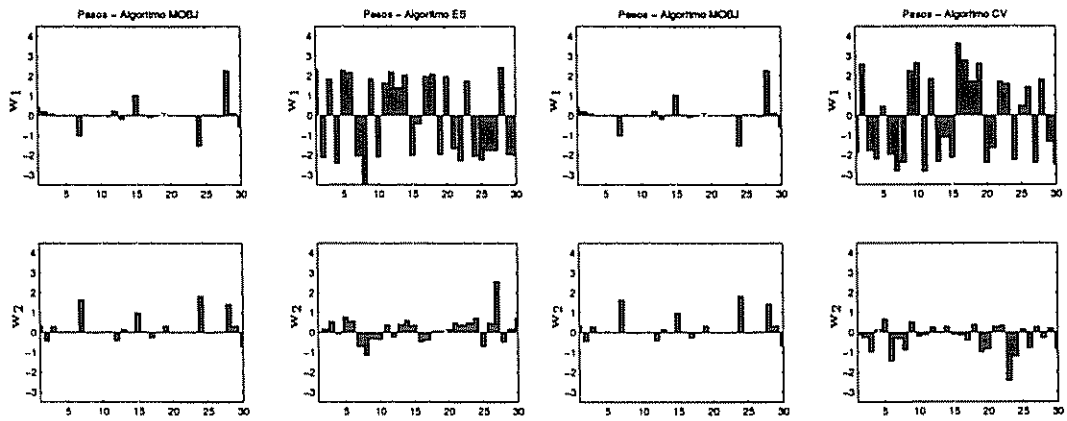
- **Comentário da Figura 6.34:**

  Nesta ilustração as posições das soluções correspondentes aos demais métodos ficaram mais distantes da melhor solução MOBJ. A solução WD foi a que mais se aproximou no valor da norma, mas se distanciou no valor do erro. Através da análise



(a) Vetores de pesos MOBJ e WD (b) Vetores de pesos MOBJ e OBD

Figura 6.32: Comparação entre os vetores de pesos obtidos pelos métodos WD, OBD e MOBJ. Exemplo 2



(a) Vetores de pesos MOBJ e ES (b) Vetores de pesos MOBJ e CV

Figura 6.33: Comparação entre os vetores de pesos obtidos pelos ES, CV e MOBJ. Exemplo 2

baseada na posição da solução WD no espaço dos objetivos pode-se inferir que a solução não está super-ajustada aos dados mas apresenta erro elevado em relação à função que se deseja mapear. As soluções ES, CV, BP e OBD apresentaram normas elevadas e certamente são soluções super-ajustadas.

As Figuras 6.35–6.37 ilustram a melhor solução MOBJ escolhida pelo decisor além de ilustrar a função geradora e as soluções obtidas pelos demais métodos.

- **Comentário das Figuras 6.35–6.37:**

  Nestas figuras pode-se notar que a solução MOBJ superou todas as demais, as quais apresentaram algum grau de *overfitting* ou *underfitting*, como é o caso da solução

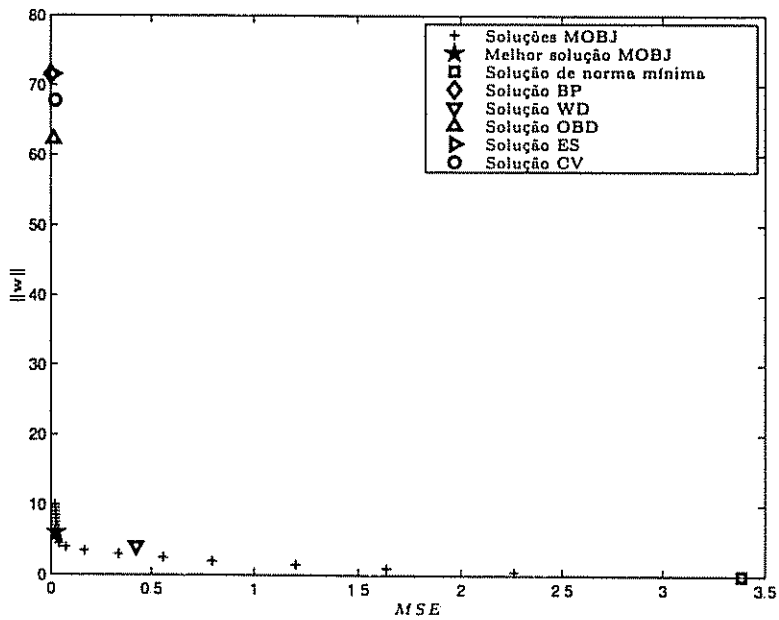


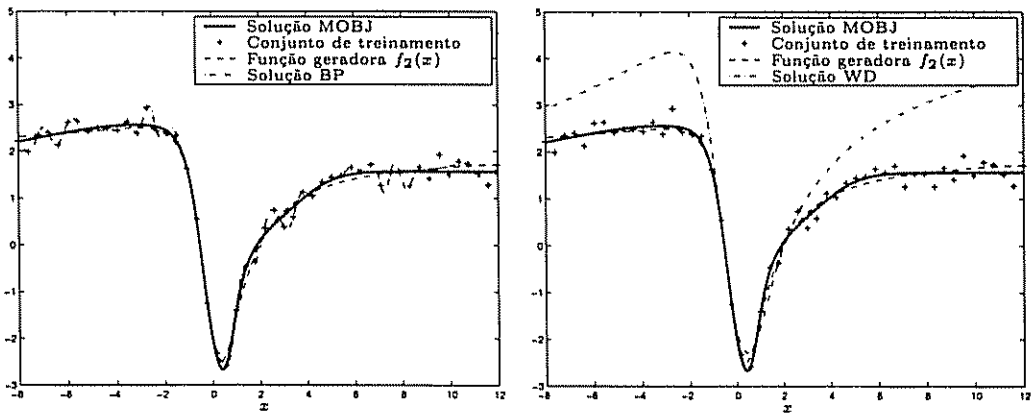
Figura 6.34: Conjunto Pareto-ótimo e demais soluções no espaço dos objetivos - Exemplo 2

WD. Nos exemplos anteriores, os métodos SVM e WD deram bons resultados mas, para esta função, apesar de terem sido feitas várias tentativas de ajuste, não se conseguiram valores para os parâmetros que pudessem melhorar tais soluções. Para problemas desta natureza, quando se ajusta os parâmetros dos algoritmos SVM e WD para que estes modelem a região suave, ocorre super-ajuste na região íngreme e quando esta região é visada, ocorre o super ou o sub-ajuste na região mais suave. Vale dizer que a solução MOBJ foi conseguida da mesma forma que nos problemas anteriores, sem que houvesse necessidade de sintonia fina de algum parâmetro para que a solução de alta capacidade de generalização fosse alcançada, o que deixa clara a superioridade do método proposto.

Na Tabela 6.12 pode-se verificar o desempenho dos métodos. Note-se que a solução MOBJ é a de maior desempenho.

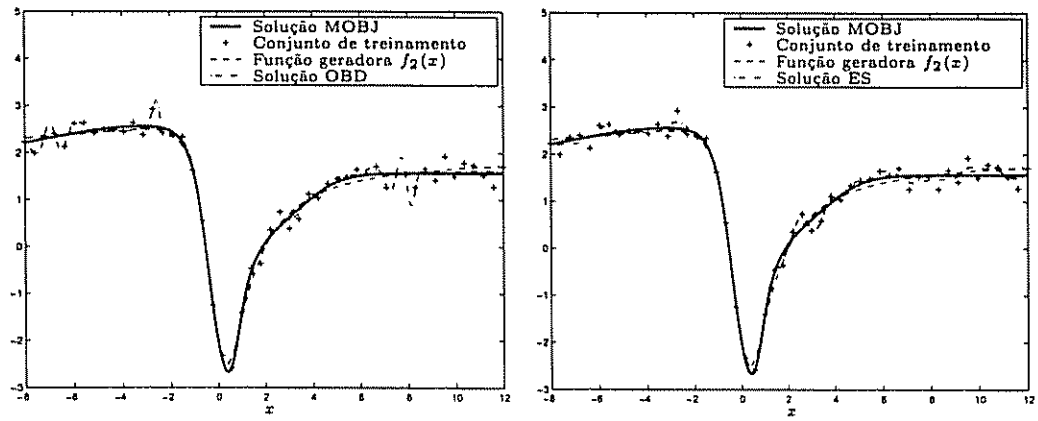
Tabela 6.12: MSE para os padrões de teste.(100 conjuntos diferentes)

Algoritmos	MSE	σ
Backpropagation	0.0787	±0.0028
Weight Decay	1.4627	±0.0157
Optimal Brain Damage	0.0699	±0.0028
Early Stopping	0.0595	±0.0022
Cross-Validation	0.0582	±0.0022
SVM	0.0783	±0.0032
MOBJ	0.0486	±0.0019



(a) Solução MOBJ e BP (b) Solução MOBJ e WD

Figura 6.35: Comparações com as soluções BP e WD para o exemplo 2

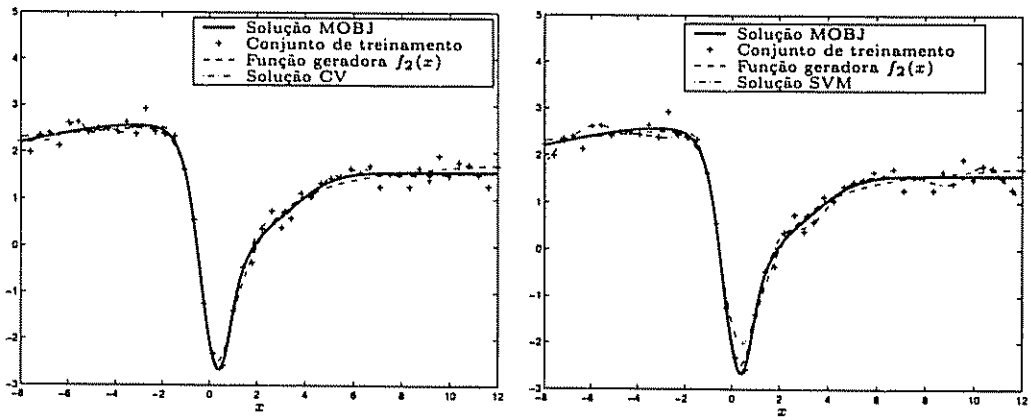


(a) Solução MOBJ e OBD (b) Solução MOBJ e ES

Figura 6.36: Comparações com as soluções OBD e ES para o exemplo 2

- **Comentários finais da Seção 6.2:**

  Verificou-se a boa capacidade de mapeamento de funções de diferentes níveis de complexidade através do algoritmo MOBJ mesmo quando se utiliza um número relativamente baixo de padrões. O processo de decisão foi validado para problemas de regressão uma vez que soluções com alta capacidade de generalização foram sempre escolhidas a partir do conjunto Pareto-ótimo. Em nenhum momento soluções com baixa capacidade de generalização foram escolhidas como solução final.



(a) Solução MOBJ e CV (b) Solução MOBJ e SVM

Figura 6.37: Comparações com as soluções CV e SVM para o exemplo 2

6.3 Conclusões do Capítulo

Neste capítulo o método multi-objetivo de treinamento de RNAs proposto foi aplicado a problemas de classificação e de regressão com diferentes níveis de complexidade, conseguindo retornar soluções com alta capacidade de generalização em todos os casos abordados. Os algoritmos *Backpropagation*, *Weight Decay*, *Optimal Brain Damage*, *Early Stopping*, *10-Fold Cross-Validation* e SVM tiveram suas soluções comparadas com as soluções MOBJ e fatores que podem fazer estes algoritmos falharem foram discutidos e demonstrados através de simulações. Outro ponto importante é a determinação dos parâmetros de treinamento destes algoritmos, os quais nem sempre são fáceis de serem sintonizados, necessitando de várias tentativas para tanto, além da experiência e do conhecimento do projetista.

Capítulo 7

Discussões, Conclusões e Propostas de Continuidade

7.1 Discussões e Conclusões

O método multi-objetivo para treinamento de RNAs proposto se mostrou bastante eficiente no mapeamento de funções no que diz respeito à qualidade da função mapeada, a qual não deve apresentar efeitos de sub ou super-ajuste aos dados de treinamento. Estes efeitos, quando presentes na função mapeada, fazem com que a sua capacidade de generalização diminua.

Para conseguir soluções com capacidade de generalização elevada, o método MOBJ trabalha com a minimização simultânea de dois objetivos, sendo eles o somatório dos erros quadráticos para os padrões de treinamento e a norma do vetor de pesos. As soluções MOBJ resultantes do processo de otimização são chamadas de soluções Pareto-ótimas e estas soluções são tais que nenhum dos objetivos pode mais ser melhorado sem que haja uma degradação do outro. Portanto, para uma solução $\mathbf{w} \in \mathcal{W}^*$, e_T é o erro para os padrões de treinamento e $\mathcal{N}$ é a menor norma possível para que e_T seja alcançado e por outro lado, sendo $\mathcal{N}$ a norma correspondente ao vetor $\mathbf{w}$, e_T é o menor erro possível para uma solução de norma $\mathcal{N}$. Tomando-se um conjunto $\mathcal{W}^*$, as soluções $\mathbf{w} \in \mathcal{W}^*$ terão normas e erros distintos sendo que as soluções de normas inferiores são também as de maiores erros para os padrões de treinamento sendo estes erros referentes aos efeitos da polarização e consequentemente estas soluções não têm a complexidade adequada para modelar o problema em questão. Por outro lado, as soluções de normas mais elevadas são as de menores erros para os padrões de treinamento mas possuem variâncias elevadas, ou seja, o fato de a norma ser elevada permite que uma grande variabilidade de soluções de mesma norma e erro possa ser encontrada. A Figura 7.1 ilustra algumas soluções Pareto-ótimas encontradas para o problema de regressão da função $f(x) = sen(2\pi x) + \xi$.

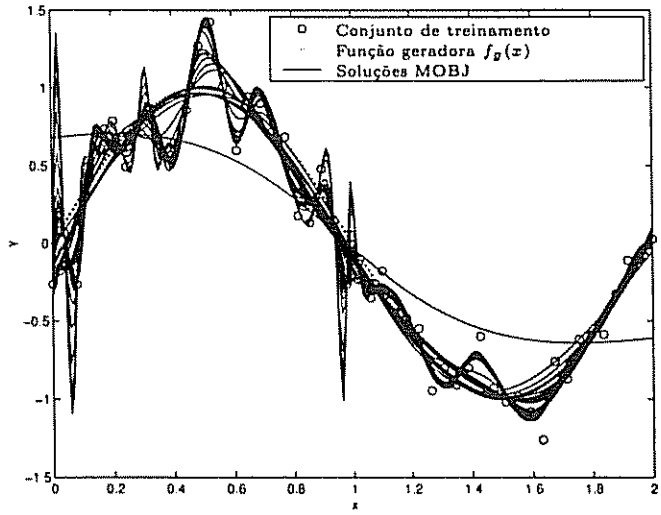


Figura 7.1: Soluções MOBJ

Para resolver o dilema entre polarização e variância, a rede não deve ser muito complexa para diminuir a variância e com isto diminuir o *overfitting*, nem ser pouco complexa, para diminuir os efeitos da polarização e com isto diminuir o *underfitting*. Apesar da utilização de uma rede com topologia fixa pelo método MOBJ, a complexidade da mesma é controlada através da imposição de restrições à norma do vetor de pesos. Redes com normas menores se comportam como redes menos complexas do que redes com normas mais elevadas. Como se pode verificar na Figura 7.1, existem desde soluções pouco complexas até soluções com complexidade elevada. Portanto, o método MOBJ é capaz de gerar soluções entre estes limites passando em algum momento por uma solução de complexidade adequada à tarefa em questão, a qual resolve o dilema entre polarização e variância. A solução que resolve tal dilema não é a que apresenta o menor erro para os padrões de treinamento.

Como o conjunto Pareto-ótimo é geralmente constituído por várias soluções, existe a necessidade de se estabelecer um processo de escolha da melhor solução. Este processo recebe o nome de "decisor" e foi desenvolvido baseado nos dados de validação. A efetividade do decisor no que diz respeito à utilização das soluções Pareto-ótimas, bem como à utilização de dados de validação para obtenção da solução final, foi demonstrada através de um teorema que prova que o conjunto Pareto-ótimo é a região do espaço que contém a melhor solução possível para o problema de aprendizagem e que através de dados de validação pode-se selecionar esta solução de forma eficiente. A Equação 5.7 descreve a função de custo e_V utilizada pelo decisor e tomando-se as soluções Pareto-ótimas $\mathbf{w}$ como argumento, o ponto de mínimo de e_V ocorre em um $\mathbf{w}$ tal que $f(\mathbf{x}_V;\mathbf{w})$ corresponde à melhor aproximação para a função geradora $f_g(\mathbf{x})$ pertencente ao conjunto Pareto-ótimo $\mathcal{W}^*$.

A seguir é reescrita a função de custo utilizada pelo decisor para demonstração da

localização do seu mínimo global.

$$e_V = \frac{1}{N_V} \sum_{i=1}^{N_V} [(d_{V_i} + \xi_i) - f(\mathbf{x}_{V_i}; \mathbf{w})]^2$$

De acordo com as manipulações feitas no Capítulo 5, a função de custo e_V pode ser reescrita conforme a Equação (5.9). Esta equação é novamente escrita a seguir.

$$e_V = \frac{1}{N_V} \sum_{i=1}^{N_V} [d_{V_i} - f(\mathbf{x}_{V_i}; \mathbf{w})]^2 + \sigma^2, \quad \mathbf{w} \in \mathcal{W}^*$$

Através do funcional e_V, pode-se verificar que quando um determinado $\mathbf{w} \in \mathcal{W}^*$ é a melhor solução pelo critério de validação, $f(\mathbf{x}_V; \mathbf{w})$ é a melhor aproximação para $f_g(\mathbf{x})$ e e_V se torna igual à variância σ^2 do ruído ξ. Logo, qualquer outro $\mathbf{w} \in \mathcal{W}^*$ que não seja uma boa aproximação para a função $f_g(\mathbf{x})$ faz com que e_V seja maior que a variância σ^2 do ruído, sendo por isto descartados pelo decisor.

O processo de escolha da melhor solução se mostrou imune ao ruído presente nos dados, principalmente quando existe um número não muito pequeno de padrões no conjunto de validação. Os resultados obtidos no Capítulo 6 mostraram que os demais algoritmos abordados neste trabalho baseados em validação falham quando a quantidade de padrões de treinamento e de validação é pequena e, ainda, os dados são ruidosos. Vale ressaltar que para os exemplos tratados no Capítulo 6 o método proposto não falhou em nenhum deles sendo que houve falhas dos demais métodos, principalmente em problemas de regressão com baixo número de padrões.

O processo de geração de soluções pode ser influenciado por uma dada realização do conjunto de treinamento N_T mas desde que este conjunto seja suficientemente grande para representar bem o problema, soluções com alta capacidade de generalização podem ser alcançadas. As simulações realizadas no Capítulo 6 mostraram que mesmo quando se tem uma quantidade de dados pequena, ainda podem-se gerar soluções Pareto-ótimas sendo que pelo menos uma delas aproxima adequadamente a função geradora $f_g(\mathbf{x})$.

Outros fatores que poderiam comprometer o desempenho do método MOBJ seriam a geração de soluções dominadas e fracamente não dominadas. Contudo, não se constatou em nenhum experimento a geração de soluções dominadas e mesmo que estas fossem geradas, seriam descartadas no processo de decisão. Em alguns casos, soluções fracamente não dominadas foram geradas mas, no processo de decisão, estas soluções também foram descartadas.

De maneira geral, o processo de geração de soluções Pareto-ótimas para resolver o problema de treinamento de RNAs é bastante simples no que diz respeito à quantidade de

parâmetros de treinamento a serem ajustados além de a solução final não ser muito sensível a estes. Para a implementação do algoritmo MOBJ através do método de relaxação o único parâmetro a ser informado é o número ζ de soluções a serem geradas. Tipicamente este número é um valor entre 20 e 50 e a qualidade da solução final não é muito sensível a este parâmetro. O problema deste método é a necessidade de se treinar uma RNA através de um algoritmo qualquer como o *backpropagation* ou uma de suas variações para se calcular um dos extremos do conjunto Pareto-ótimo que corresponde à solução de norma máxima. Nas simulações apresentadas neste trabalho utilizou-se apenas o método ε-restrito para obtenção de soluções, não havendo a necessidade do custo adicional de treinamento de uma RNA a priori. Porém, neste método tem-se que fornecer valores para três parâmetros. O primeiro deles diz respeito à topologia da rede a ser treinada (este parâmetro também deve ser informado para os demais métodos abordados com exceção das SVMs) e para que o método funcione adequadamente o usuário deve superdimensionar a topologia para permitir soluções super-ajustadas. A topologia adotada para as redes a serem treinadas é $u_i \times u_h \times u_o$, sendo que, u_i e u_o são dados pela dimensão das entradas e das saídas do problema e u_h é feito igual ao número de neurônios da camada escondida N_N fornecido pelo usuário. É importante dizer que obter o número de neurônios na camada escondida para que uma rede aprenda adequadamente uma dada tarefa não é um processo trivial, mas superdimensionar este número, é o que a maioria dos projetistas fazem na prática, é uma tarefa bastante simples. O segundo parâmetro é o número ζ de soluções MOBJ a serem geradas. O ajuste deste é feito conforme já discutido anteriormente. O outro parâmetro é o incremento $\delta\epsilon$ que a norma deve sofrer de uma solução para uma outra. O valor para este parâmetro é tipicamente um valor entre 0.5 e 3 devendo ser menor para problemas cuja solução é de norma baixa e maior quando o problema possui solução de norma elevada. No Capítulo 5 foi discutido em detalhe o efeito da escolha deste parâmetro na resolução do conjunto Pareto-ótimo. Para resumir, o ajuste dos parâmetros N_N, ζ e $\delta\epsilon$ não é um processo crítico para obtenção de soluções com alta capacidade de generalização.

É importante ressaltar que a qualidade das soluções quando foram utilizados os demais métodos de treinamento se mostrou extremamente dependente dos valores dos parâmetros externos e do tamanho dos conjuntos de treinamento e validação utilizados. Algumas das razões que podem fazer com que os algoritmos BP (*backpropagation*), WD (*Weight Decay*), OBD(*optimal brain damage*), ES(*Early Stopping*), CV(*Cross-Validation*) e SVM(*Support Vector Machines*) não retornem soluções com alta capacidade de generalização são abordadas a seguir.

O algoritmo BP efetua o treinamento de redes do tipo MLP simplesmente minimizando o somatório dos erros quadráticos em relação aos padrões de treinamento. A solução BP $\mathbf{w}^{BP}$ é dada pela Equação (7.1).

$$\mathbf{w}^{BP} = \arg\min_{\mathbf{w}} \sum_{i=1}^{N_T} [\mathbf{d}_{T_i} - f(\mathbf{x}_{T_i}; \mathbf{w})]^2, \quad \mathbf{w} \in \mathbb{R}^z \tag{7.1}$$

A qualidade de uma solução BP é extremamente sensível ao número de pesos da rede, ou seja, da topologia, do erro colocado como meta para interrupção do treinamento, do número de épocas de treinamento além do efeito de mínimos locais. Portanto, obter uma solução $\mathbf{w}^{BP}$ com alta capacidade de generalização é quase sempre muito difícil principalmente quando o problema de aprendizagem é multidimensional.

Uma tentativa de eliminar o problema do treinamento excessivo que ocorre geralmente quando se tem topologias de redes superdimensionadas para o problema em questão, além da imposição de erros pequenos e número grande de épocas, é fazer a interrupção precoce do treinamento. O algoritmo ES trabalha neste sentido e uma solução $\mathbf{w}^{ES}$ pode ser obtida segundo a Equação (7.2),

$$\begin{aligned} \mathbf{w}^{ES} = & \arg\min_{\mathbf{w}} \sum_{i=1}^{N_T} [\mathbf{d}_{T_i} - f(\mathbf{x}_{T_i}; \mathbf{w})]^2, \quad \mathbf{w} \in \mathbb{R}^z \\ \text{Sujeito a :} & \quad \sum_{i=1}^{N_V} [\mathbf{d}_{V_i} - f(\mathbf{x}_{V_i}; \mathbf{w})]^2 = e_V^* \end{aligned} \tag{7.2}$$

onde e_V^* é o valor mínimo para o somatório dos erros quadráticos em relação aos padrões de validação e z é o número total de pesos incluindo os termos de polarização da rede.

A função de restrição a qual é responsável pela interrupção do treinamento pode apresentar mínimos locais e fazer com que o algoritmo ES interrompa o treinamento antes do momento exato. Outro fator ainda seria a constituição do conjunto de validação que, dependendo do número de padrões e de uma dada realização para o mesmo, o ponto de mínimo da curva de validação pode ser deslocado. Experimentos realizados com este algoritmo onde tanto o conjunto de treinamento quanto o conjunto de validação tinham poucas amostras resultaram em redes com algum grau de *overfitting*.

O algoritmo CV *k-fold Cross-Validation* trabalha fazendo k divisões sobre o conjunto de treinamento na tentativa de evitar o problema de uma dada realização para o conjunto de treinamento e validação onde k redes são treinadas e uma delas é escolhida em função do erro de generalização apresentado. Portanto, este algoritmo deve apresentar melhores resultados que o algoritmo ES acima citado. Cada solução CV é conseguida segundo a Equação (7.3),

$$\begin{aligned} \mathbf{w}_j = & \arg\min_{\mathbf{w}} \sum_{i=1}^{N_{T_j}} \left[\mathbf{d}_{T_{j_i}} - f(\mathbf{x}_{T_{j_i}}; \mathbf{w})\right]^2, \quad \mathbf{w} \in \mathbb{R}^z \\ \text{Sujeito a :} & \quad \sum_{i=1}^{N_{V_j}} \left[\mathbf{d}_{V_{j_i}} - f(\mathbf{x}_{V_{j_i}}; \mathbf{w})\right]^2 = e_{V_j}^*, \quad j = 1, 2, \ldots, k \end{aligned} \tag{7.3}$$

onde $e^*_{V_j}$ é o valor mínimo para o somatório dos erros quadráticos em relação aos padrões de validação para o j-ésima rede e z é o número total de pesos incluindo os termos de polarização da rede. A solução $\mathbf{w}^{CV}$ final é pela Equação (7.4).

$$\mathbf{w}^{CV} = \arg\min_{\mathbf{w}_j} \sum_{i=1}^{N_V} [\mathbf{d}_{V_i} - f(\mathbf{x}_{V_i}; \mathbf{w}_j)]^2, \quad \mathbf{w}_j \in \mathcal{W}^{CV} \tag{7.4}$$

Para o algoritmo CV, a qualidade da solução final também pode ser influenciada pelo número de padrões nos conjuntos de treinamento e validação. Os resultados obtidos neste trabalho para este algoritmo deixaram a desejar quando o número de padrões foi feito pequeno em cada conjunto.

O algoritmo OBD trabalha no sentido de alterar a topologia da rede para se alcançar uma solução com alta capacidade de generalização. Uma rede inicialmente superdimensionada é adotada e, durante o treinamento, conexões são eliminadas. Sendo z o número de pesos inicial da rede e r o número de conexões eliminadas durante o treinamento, a solução $\mathbf{w}^{OBD}$ é dada pela Equação (7.5).

$$\mathbf{w}^{OBD} = \arg\min_{\mathbf{w}} \sum_{i=1}^{N_T} [\mathbf{d}_{T_i} - f(\mathbf{x}_{T_i}; \mathbf{w})]^2, \quad \mathbf{w} \in \mathbb{R}^{z-r} \tag{7.5}$$

Note-se que se a redução do número de parâmetros não for suficiente, o algoritmo procura por uma solução em $\mathbb{R}^{z-r}$ fazendo minimização de erro quadrático com um determinado número de épocas. Isto pode fazer com que a solução $\mathbf{w}^{OBD}$ fique super-ajustada aos dados de treinamento. Outro fator ainda é que a redução do número de parâmetros pode ser muito grande de forma que a complexidade da rede fique insuficiente para mapear a função que se deseja. Os resultados obtidos com este algoritmo são melhores que os obtidos com o algoritmo BP mas geralmente são piores que os obtidos com os algoritmos ES e CV.

O algoritmo WD trabalha diretamente com redução da norma do vetor de pesos durante o treinamento. Uma solução $\mathbf{w}^{WD}$ pode ser alcançada segundo a Equação (7.6),

$$\mathbf{w}^{WD} = \arg\min_{\mathbf{w}} \sum_{i=1}^{N_T} [\mathbf{d}_{T_i} - f(\mathbf{x}_{T_i}; \mathbf{w})]^2 + \lambda \|\mathbf{w}\|, \quad \mathbf{w} \in \mathbb{R}^z \tag{7.6}$$

onde λ é a constante de queda dos pesos e z é o número total de pesos incluindo os termos de polarização da rede.

O problema do algoritmo WD é a determinação da constante de queda λ, a qual geralmente é feita a mesma para todos os pesos da rede, em todas as camadas. Para cada problema de aprendizagem a constante λ deve assumir valores diferentes e, se este valor for grande para um dado problema, a rede tenderá a ficar sub-ajustada aos dados e por outro lado, se esta constante é pequena para o problema, a rede tenderá a ficar super-ajustada

as dados de treinamento. Portanto, principalmente para problemas multidimensionais, há a necessidade de várias tentativas para um ajuste adequado deste parâmetro. Nas simulações realizadas neste trabalho, na maioria dos casos se alcançaram boas soluções com este algoritmo, mas um caso específico foi tratado na Seção 6.2.2 para mostrar o problema de se aplicar a mesma constante sobre todos os pesos. Naquele exemplo, a função era constituída de regiões suaves e íngremes e não se conseguiu um valor para λ que possibilitasse o mapeamento de ambas as regiões simultaneamente.

O algoritmo MOBJ também trabalha com redução da norma do vetor de pesos mas não faz aplicação de uma constante de queda sobre os mesmos. A norma é reduzida de forma geral reduzindo-se a magnitude dos pesos não relevantes para o problema podendo até incrementar a magnitude de outros. Desta forma, o algoritmo MOBJ se comporta como se cada elemento do vetor de pesos tivesse uma constante de queda própria podendo esta ser positiva ou até negativa. Para o exemplo mostrado na Seção 6.2.2, o algoritmo MOBJ fez o mapeamento de todas as regiões da curva.

O algoritmo SVM também trabalha com redução da norma do vetor de pesos como pode ser visto na Equação (7.7) que traz o problema na sua forma primal.

$$\mathbf{w}^{SVM} = \arg\min_{\mathbf{w}} \min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{N_T} \alpha_i \left[d_i(\mathbf{w}^T\mathbf{x} + b) - 1\right] \tag{7.7}$$

Os parâmetros de treinamento para este algoritmo surgem quando o problema de otimização é escrito na sua forma dual, conforme a expressão seguinte,

$$\alpha = \arg\max_{\alpha} \quad \sum_{i=1}^{N_T} \alpha_i - \frac{1}{2}\sum_{i=1}^{N_T}\sum_{j=1}^{N_T} \alpha_i\alpha_j d_i d_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{Sujeito a :} \quad \begin{cases} \sum_{i=1}^{N_T} \alpha_i d_i = 0 \\ 0 \leq \alpha_i \leq C \end{cases}$$

onde C é um parâmetro especificado pelo usuário que denota o limite para os multiplicadores de Lagrange α, $\mathbf{d}$ denota a saída desejada, $\mathbf{x}$ denota o conjunto de entrada e $K(\mathbf{x}_i, \mathbf{x}_j)$ é o *kernel* produto interno.

Para a obtenção de uma rede com alta capacidade de generalização, o usuário deve escolher o *kernel* e consequentemente seus parâmetros (variância para *kernel* RBF) e um valor para o parâmetro C. A solução $\mathbf{w}^{SVM}$ é altamente sensível a estes parâmetros e principalmente para problemas multidimensionais o ajuste destes é feito através de múltiplas tentativas.

O método MOBJ também trabalha com redução da norma do vetor pesos e as soluções Pareto-ótimas podem ser conseguidas através da Equação (7.8),

$$\mathbf{w}_j = \arg\min_{\mathbf{w}} \sum_{i=1}^{N_T} [\mathbf{d}_{T_i} - f(\mathbf{x}_{T_i}; \mathbf{w})]^2, \quad \mathbf{w} \in \mathbb{R}^z \tag{7.8}$$
$$\text{Sujeito a :} \qquad \|\mathbf{w}\| \leq \mathcal{N}_j \quad j = 1, 2, \ldots, \zeta$$

onde $\mathcal{N}_j$ é a norma para o j-ésimo problema de otimização e ζ é o número total de soluções a serem geradas. A solução MOBJ $\mathbf{w}^*$ é dada pelo decisor implementado segundo a Equação (7.9).

$$\mathbf{w}^* = \arg\min_{\mathbf{w}_j} \sum_{i=1}^{N_V} [\mathbf{d}_{V_i} - f(\mathbf{x}_{V_i}; \mathbf{w}_j)]^2, \quad \mathbf{w}_j \in \mathcal{W}^* \tag{7.9}$$

Por construção, o algoritmo MOBJ encontra soluções Pareto-ótimas para o problema e o decisor deve escolher uma entre estas. O conjunto Pareto-ótimo $\mathcal{W}^*$ é limitado a um número ζ de soluções e o processo de validação é eficiente como demonstra o Teorema 2 no Capítulo 5.

Os demais métodos baseados em validação como o ES e o CV utilizam dados de validação para buscar uma solução em um universo irrestrito de soluções de dimensão $\mathbb{R}^z$. Portanto, se a rede é demasiadamente complexa, existe a possibilidade dos métodos ES e CV alcançarem uma solução que esteja super-ajustada aos conjuntos de treinamento e de validação ao mesmo tempo, principalmente quando se tem números pequenos de padrões. Os algoritmos ES e CV não fazem redução de norma por construção sendo que soluções de normas adequadas podem ser conseguidas por validação. O algoritmo CV é mais eficiente que o ES uma vez que este utiliza vários conjuntos de treinamento e validação. As simulações mostraram que, nos casos onde as soluções ES e CV se aproximaram da melhor solução MOBJ no espaço dos objetivos, estas soluções resultaram em capacidades de generalização semelhantes.

Os métodos WD e OBD fazem redução de norma, o primeiro pela redução da magnitude e o segundo pela eliminação de pesos. Ambos os métodos podem alcançar soluções Pareto-ótimas dependendo da escolha dos valores dos parâmetros de cada método. Nas simulações realizadas, o algoritmo OBD retornou soluções de normas mais elevadas que o necessário já o algoritmo WD retornou soluções semelhantes às soluções MOBJ no que diz respeito à capacidade de generalização com exceção do caso de regressão onde não se conseguiu um ajuste adequado para a constante de queda dos pesos.

O algoritmo BP também pode alcançar a região Pareto-ótima se a escolha da topologia, do número de iterações, do número de padrões e do erro a ser alcançado forem bem feitos. Como isto é uma tarefa difícil, o processo de obtenção de uma solução que esteja sobre a região Pareto-ótima com o algoritmo BP é bastante difícil.

Soluções sobre a região Pareto-ótima também podem ser alcançadas com o algoritmo SVM, dependendo da escolha do conjunto de parâmetros de treinamento. As soluções

SVM não foram desenhadas no espaço dos objetivos para as comparações com os demais métodos por se tratar de uma máquina diferente das demais apesar de se configurar numa rede com topologia $u_i \times u_h \times u_o$. Para problemas de classificação o somatório dos erros quadráticos em relação aos padrões de treinamento é bastante elevado uma vez que os dados estão na faixa de ± 1 e a função mapeada pode estar na faixa de, por exemplo, ± 10. O importante para as SVMs é a passagem por zero da função, ou seja, a curva de nível onde os valores das variáveis de entrada fazem a função mapeada se tornar igual a 0 e a margem de separação, dada pelos valores de entrada que fazem a função mapeada assumir os valores ± 1. Para problemas de regressão, como a função de ativação no nodo de saída das SVMs é linear, a máquina SVM se parece um pouco mais com as redes treinadas através dos demais métodos mas ainda assim não podem ser comparadas.

A quantidade de padrões utilizados no processo de treinamento influencia na qualidade das soluções geradas pelos demais métodos abordados neste trabalho. Porém, na medida que esta quantidade cresce, as soluções fornecidas por estes métodos ficam cada vez mais agrupadas no espaço dos objetivos e cada vez mais próximas da solução MOBJ final. Ressalta-se aqui a eficiência do algoritmo proposto quando se trabalha com poucos padrões de treinamento e validação.

O método MOBJ têm várias vantagens sobre os demais métodos abordados neste trabalho uma vez que soluções com alta capacidade de generalização são facilmente obtidas. As vantagens são:

I. Possibilidade de se lidar com conjuntos de treinamento e validação pequenos sem muita influência no resultado final;

II. Baixa sensibilidade ao ruído geralmente presente nos dados, principalmente quando se trabalha com um número não muito pequeno de padrões;

III. Como vantagem principal, tem-se a facilidade de ajuste dos parâmetros de treinamento uma vez que a solução final não é muito sensível aos mesmos.

Como desvantagem do método proposto pode-se citar o tempo de execução do mesmo, uma vez que ζ soluções devem ser geradas e para cada uma das ζ soluções um problema de otimização deve ser resolvido com a necessidade de cálculo de gradientes das funções objetivos, que são erro e norma do vetor de pesos. O cálculo das ζ soluções MOBJ pode se tornar desnecessário se uma heurística para determinação de qual solução deve ser computada for criada. Esta heurística será colocada como proposta de continuidade deste trabalho mais adiante.

Para resumir, o método MOBJ se mostrou bastante robusto e eficiente, gerando boas soluções independentemente da complexidade do problema de aprendizagem. Alguns pontos, entretanto, merecem ainda ser estudados e fazem parte das propostas de continuidade deste trabalho.

7.2 Propostas de continuidade

Existem alguns pontos importantes a serem investigados para que o método proposto seja implementado de forma eficiente. Estes pontos são relacionados a seguir:

- Implementação de heurísticas que torne desnecessário o cálculo das ζ soluções MOBJ;
  Conforme mencionado anteriormente, a necessidade de cálculo das ζ soluções MOBJ possui um custo relativamente elevado principalmente se comparado ao custo computacional de métodos que geram apenas uma solução baseada em parâmetros. Os demais métodos abordados foram concebidos para serem eficientes, computacionalmente falando. O algoritmo proposto utiliza técnicas de otimização genéricas e foi concebido para obtenção de uma solução com capacidade de generalização elevada porém, não foram tratados fatores que possam resultar em melhoria da eficiência computacional.

  A implementação de heurísticas que possibilitem o cálculo de apenas algumas soluções Pareto-ótimas pode proporcionar um grande ganho em eficiência computacional para o método MOBJ. Uma alternativa para reduzir o número total de iterações pode ser feita, por exemplo, através da realização de busca pela "seção áurea". Desta forma, um número ótimo de iterações seria realizado para se obter a solução final.

- Possibilidade de se efetuar *pruning*;
  Conforme pode ser visto na Figura 7.2, a qual compara os vetores de pesos de uma solução MOBJ e de uma solução BP, muitos elementos do vetor correspondente à solução MOBJ possuem valores muito baixos, dando a indicação de que estes elementos não contribuem ou pouco contribuem para o cálculo dos valores de saída da RNA. Portanto, a eliminação destes pesos abaixo de uma certa magnitude poderá reduzir a topologia da rede, proporcionando soluções com topologias mais simples.

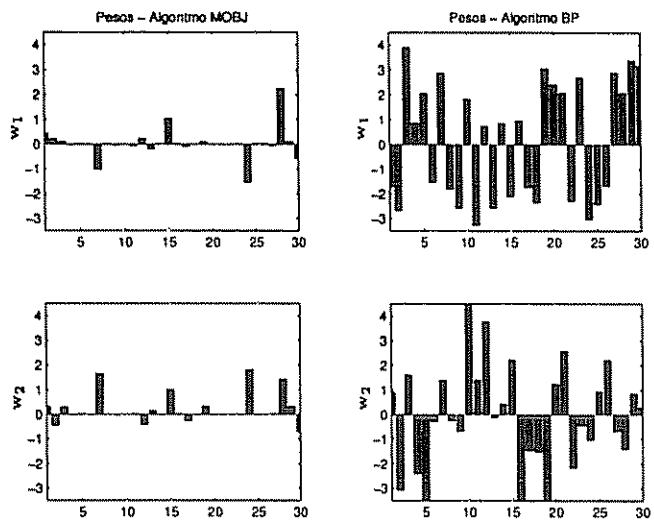


Figura 7.2: Vetores de pesos MOBJ e BP

- Implementação de novos decisores;
  O decisor implementado se baseia em dados de validação para escolha da melhor solução MOBJ. Este decisor é eficiente mas necessita de um conjunto de validação. Por isto, outras formas de decisão podem ser pesquisadas e, por exemplo, fazer decisões baseadas nos resíduos entre as soluções MOBJ e os dados de treinamento. Estes resíduos podem trazer alguma forma de se escolher a melhor solução.

- Utilização em outros tipos de redes;
  O método criado neste trabalho foi aplicado apenas a redes do tipo MLP porém aplicações deste método a outros tipos de redes como RBF por exemplo ainda são questões totalmente abertas.

Referências Bibliográficas

[Barron, 1993] Barron, A. R. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945.

[Barros et al., 2000] Barros, M. A., Braga, A. P., and Braga, J. P. (2000). SVM-KM: speeding SVMs learning with a priori cluster selection and k-means. In *Proceedings of Sixth Brazilian Symposium on Neural Networks (SBRN'2000*, volume I, pages 162–167, Rio de Janeiro, RJ, Brazil. IEEE Computer Society Press.

[Barros et al., 2001] Barros, M. A., Braga, A. P., and Braga, J. P. (2001). Training SVMs with EDR algorithm. *International Journal of Neural Systems (To appear)*.

[Bartlett, 1997] Bartlett, P. L. (1997). For valid generalization the size of the weights is more important than the size of the network. In Mozer, M. C., Jordan, M. I., and Petsche, T., editors, *Advances in Neural Information Processing Systems*, volume 9, page 134. The MIT Press.

[Bengio, 1996] Bengio, Y. (1996). *Neural Networks for Speech and Sequence Recognition*. Thomson.

[Bertsekas, 1995] Bertsekas, D. P. (1995). *Nonlinear Programming*. MA: Athenas Scientific, Belmont.

[Bishop, 1995] Bishop, C. M. (1995). *Neural Network for Pattern Recognition*. Oxford University Press, Oxford, UK.

[Blake and Merz, 1998] Blake, C. and Merz, C. (1998). UCI repository of machine learning databases.

[Boser et al., 1992] Boser, B., Guyon, I., and Vapnik, V. (1992). A training algorithm for optimal margin classifiers. *Fifth Annual Workshop on Computational Learning Theory*, pages 144–152.

[Burges, 1998] Burges, C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2).

[Caudill and Butler, 1990] Caudill, M. and Butler, C. (1990). *Naturally Intelligent Systems*. MIT Press, Cambridge, Massachusetts.

[Chankong and Haimes, 1983] Chankong, V. and Haimes, Y. Y. (1983). *Multiobjective Decision Making: Theory and Methodology*, volume 8. North-Holland (Elsevier), New York.

[Chinrungrueng and Séquin, 1995] Chinrungrueng, C. and Séquin, C. H. (1995). Optimal adaptive k-means algorithm with dynamic adjustment of learning rate. *IEEE Transactions on Neural Networks*, 6:157–169.

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. (1995). Support vector networks. *Machine Learning*, 20:273–279.

[Courant and Hilbert, 1970] Courant, R. and Hilbert, D. (1970). *Methods of Mathematical Phisics*, volume 1 and 2. Wiley Interscience, New York.

[Cun et al., 1990] Cun, Y. L., Denker, J. S., and Solla, S. A. (1990). Optimal brain damage. In *Advances in Neural Information Processing Systems 2*, pages 598–605.

[Cybenko, 1989] Cybenko, G. (1989). Approximation by superpositions of a sigmoid function. *Mathematics of Control, Signals and Systems*, 2:303–314.

[Duckstein, 1984] Duckstein, L. (1984). *Multiobjective optimization in structural design: The model choice problem*. John Wiley& Sons.

[Duda and Hart, 1973] Duda, R. O. and Hart, P. E. (1973). *Pattern Classification and Scene Analysis*. Wiley-Interscience Publication.

[Fahlman, 1988] Fahlman, S. E. (1988). Faster-learning variations on back-propagation: an empirical study. In Touretzky, D., Hinton, G., and Sejnowski, T., editors, *Proceedings of the 1988 Connectionist Models Summer School, Pittsburg*, pages 38–51, San Mateo, CA. Morgan Kaufmann.

[Fahlman and Lebiere, 1990] Fahlman, S. E. and Lebiere, C. (1990). The cascade-correlation learning architecture. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 2*. Morgen Kaufmann.

[Geman et al., 1992] Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58.

[Girosi et al., 1995] Girosi, F., Jones, M., and Poggio, T. (1995). Regularization theory and neural networks architectures. *Neural Computation*, 7:219–269.

[Goutte, 1997] Goutte, C. (1997). Note on free lunches and cross-validation. *Neural Computation*, 9(6):1245–1249.

[Hagan and Menhaj, 1994] Hagan, M. T. and Menhaj, M. B. (1994). Training feedforward networks with the Marquardt algorithm. *IEEE Transactions on Neural Networks*, 5(6):989–993.

[Hanson et al., 1993] Hanson, S. J., Cowan, J. D., and Giles, C. L., editors (1993). *Advances in Neural Information Processing Systems 5*, San Mateo, CA. Morgan Kaufman Publishers Inc.

[Hassibi and Stork, 1993] Hassibi, B. and Stork, D. G. (1993). Second order derivatives for network pruning: Optimal brain surgeon. In *[Hanson et al., 1993]*, pages 164–171.

[Hinton, 1989] Hinton, G. E. (1989). Connectionist learning procedures. *Artificial Intelligence*, 40:185–234.

[Horst et al., 1998] Horst, P., Padilha, T., Rocha, C., Rezende, S., and de Carvalho, A. (1998). Knowledge acquisition using symbolic and connectionist algorithms for credit evaluation. In Simpson, P., editor, *IEEE World Congress on Computational Intelligence*, Anchorage, EUA.

[Ismail and Kamel, 1989] Ismail, M. A. and Kamel, M. S. (1989). Multidimensional data clustering utilizing hybrid search strategies. *Pattern Recognition*, 22:75–89.

[Ismail et al., 1984] Ismail, M. A., Selim, S. Z., and Arora, S. K. (1984). Efficient clustering of multidimensional data. In *Proceedings of the IEEE International Conference on Systems Man and Cybernetics*, pages 120–123.

[Karnin, 1990] Karnin, E. D. (1990). A simple procedure for pruning back-propagation trained neural networks. *IEEE Trans. Neural Networks*, 1(2):239–242.

[Kepuska and Gowdy, 1989] Kepuska, V. Z. and Gowdy, J. N. (1989). Phonemic speech recognition system based on a neural network. In *SOUTHEASTCON '89 Proceedings. Energy and Information Technologies in the Southeast*, volume 2, pages 770–5, New York, NY, USA. IEEE.

[Lacerda et al., 2001] Lacerda, E., Carvalho, A., Braga, A. P., and Ludermir, T. B. (2001). Using evolutionary RBF networks for credit assessment. *(Submitted)*.

[Lawrence et al., 1996] Lawrence, S., Giles, C. L., and Tsoi, A. (1996). What size neural network gives optimal generalization? Convergence properties of backpropagation. Technical Report UMIACS-TR-96-22 and CS-TR-3617, Institute for Advanced Computer Studies, University of Maryland, College Park MD 20742.

[Lloyd, 1982] Lloyd, S. P. (1982). Least square quantization in pcm. ieee transactions on information theory. *IEEE Transactions on Information Theory*, 28(2):129–137.

[Luenberger, 1984] Luenberger, D. G. (1984). *Linear and Nonlinear Programming*. Addison-Wesley Publishing Company, Stanford, California, 2 edition.

[MacQueen, 1967] MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkley Symposium Math*, volume 1, pages 281–297.

[Mercer, 1909] Mercer, J. (1909). Functions of positive and negative type and their conections with the theory of integral equations. *Philos. Trans. Roy. Soc. London*, A(209):415–446.

[Moody, 1992] Moody, J. E. (1992). The effective number of parameters: An analysis of generalization and regularization in nonlinear systems. In Moody, J. E., Hanson, S. J., and Lippmann, R. P., editors, *Advances in Neural Information Processing Systems*, volume 4, pages 847–854. Morgan Kaufmann, San Mateo, CA.

[Mozer and Smolensky, 1989] Mozer, M. C. and Smolensky, P. (1989). Skeletonization: A technique for trimming the fat from a network via relevance assessment. *Advances in Neural Information Processing (1)*, pages 107–115.

[Nørgaard, 1997] Nørgaard, M. (1997). Neral network based system identification toolbox. Technical Report 97-E-851, Technical University of Denmark.

[Nomura, 1999] Nomura, M. (1999). A comfortable brain-interface to video displays. *Neural Networks*, 12(2):347–354.

[Osuna et al., 1997a] Osuna, E., Freund, R., and Girosi, F. (1997a). An improved training algorithm for support vector machines. *Neural Networks for Signal Processing VII, Proceedings of the 1997 IEEE Workshop*, pages 276–285.

[Osuna et al., 1997b] Osuna, E., Freund, R., and Girosi, F. (1997b). Support vector machines: training and aplications. *MIT AI Memo 1602*.

[Osuna and Girosi, 1998] Osuna, E. and Girosi, F. (1998). Reducing the run-time complexity of support vector machines. *ICPR 98*.

[Parekh et al., 1987] Parekh, R., Yang, J., and Honavar, V. (1987). Constructive neural network learning algorithms for multi-category real-valued pattern classification. Technical report, Iowa State University. Department of Computer Science.

[Pareto, 1896] Pareto, V. (1896). *Cours D'Economie Politique*, volume I and II. Rouse, Lausanne,.

[Parma et al., 1998] Parma, G. G., Menezes, B. R., and Braga, A. P. (1998). Sliding mode algorithm for training multi-layer neural networks. *IEE Electronics Letters*, 38(1):97–98.

[Riedmiller and Braun, 1993a] Riedmiller, M. and Braun, H. (1993a). A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proc. of the IEEE Intl. Conf. on Neural Networks*, pages 586–591, San Francisco, CA.

[Riedmiller and Braun, 1993b] Riedmiller, M. and Braun, H. (1993b). A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proc. of the IEEE Intl. Conf. on Neural Networks*, pages 586–591, San Francisco, CA.

[Rumelhart et al., 1986a] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986a). Learning internal representation by error propagation. In Rumelhart, D. E. and McClelland, J. L., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, pages 318–362. Cambridge, MA: MIT Press.

[Rumelhart et al., 1986b] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986b). Learning representations by back-propagating errors. *Nature*, 323:533–536.

[Rumelhart et al., 1986c] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986c). *Parallel Distributed Processing*, volume 1: Foundations, chapter Learning internal representations by error propagation, pages 318–362. The MIT Press.

[Rumelhart and McClelland, 1986] Rumelhart, D. E. and McClelland, J. L. (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*. MIT Press, Cambridge, MA.

[Shor, 1977] Shor, N. Z. (1977). Cut-off method with space extension in convex programming problems. *Cybernetics*, 12:94–96.

[Soares et al., 1991] Soares, J. F., Farias, A. A., and Cesar, C. C. (1991). *Introdução à Estatística*. Guanabara Koogan.

[Stone, 1974] Stone, M. (1974). Cross-validation choice and assessment of statistical predictions. *Journal of the Royal Statistical Society*, B-36:111–147.

[Stone, 1978] Stone, M. (1978). Cross-validation: A review. *Mathematische Operationsforschung Statistischen*, 9:127–140.

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

[Teixeira et al., 2001] Teixeira, R. A., Braga, A. P., Takahashi, R. H. C., and Saldanha, R. R. (2001). Recent advances in the mobj algorithm for training artificial neural networks. *International Journal of Neural Systems (To appear)*.

[Vapnik, 1982] Vapnik, V. (1982). *Estimation of Dependencies Based on Empirical Data*. Springer-Verlag.

[Vapnik, 1995] Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag.

[Vapnik et al., 1994] Vapnik, V., Levin, E., and Cun, Y. L. (1994). Measuring the VC-dimension of a learning machine. *Neural Computation*, 6(5):851–876.

[Vapnik, 1998] Vapnik, V. N. (1998). *Statistical Learning Theory*. New York:Wiley.

[Weigend et al., 1990] Weigend, A. S., Huberman, B. A., and Rumelhart, D. E. (1990). Predicting the future: a connectionist approach. *International Journal of Neural Systems*, 1:193–209.

[Weigend et al., 1991] Weigend, A. S., Rumelhart, D. E., and Huberman, B. A. (1991). Generalization by weight-elimination with application to forecasting. In *Advances in Neural Information Processing Systems 3*, pages 875–882.

Apêndice A

Apêndice : Rotinas MOBJ

A.1 Interface para o algoritmo MOBJ para problemas de classificação

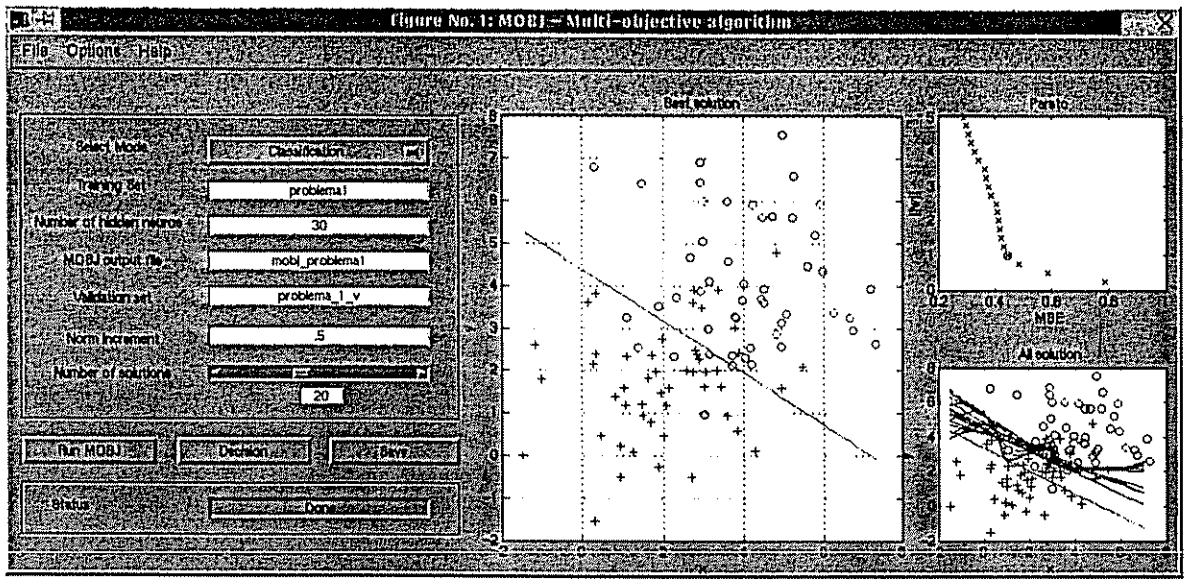


Figura A.1: Interface para o método MOBJ para problemas de classificação

A Figura A.1 traz a interface para problemas de classificação de duas classes. Basicamente, o usuário poderá entrar com os parâmetros de treinamento e as saídas do programa são três gráficos. No maior deles é mostrada a curva de decisão para o problema de classificação, o segundo no canto superior direito mostra o conjunto Pareto-ótimo para o problema em questão e no último, todas as soluções MOBJ computadas são mostradas.

A.2 Interface para o algoritmo MOBJ para problemas de regressão

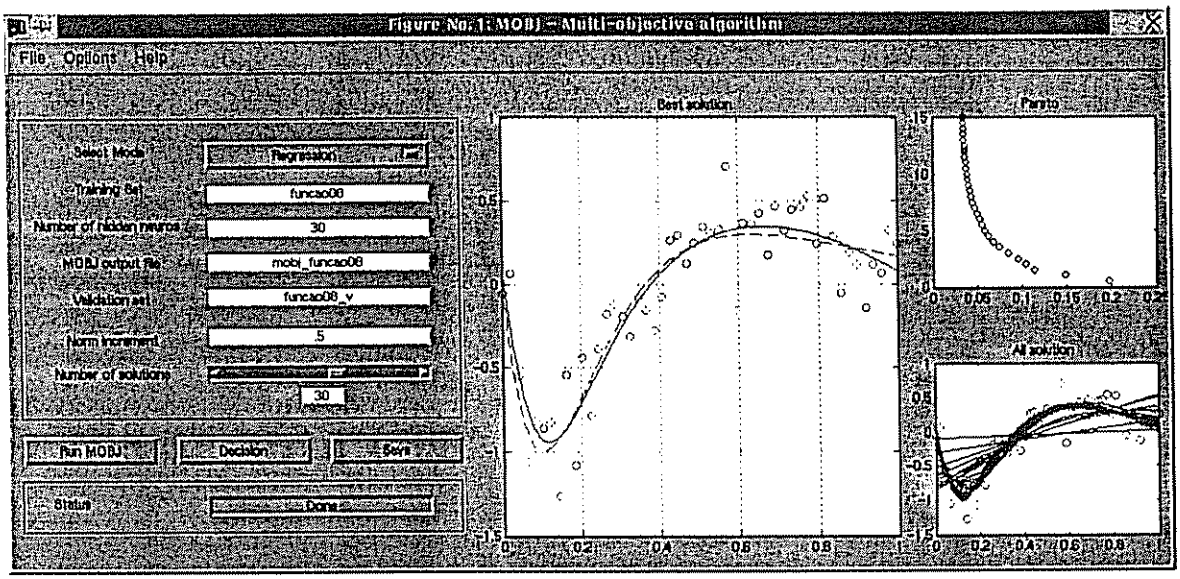


Figura A.2: Interface para o método MOBJ para problemas de regressão

A Figura A.2 traz a interface para problemas de regressão para uma variável de entrada e uma de saída. Nesta, o usuário poderá entrar com os parâmetros de treinamento e as saídas do programa são três gráficos. No maior deles é mostrada a regressão para o problema em questão, o segundo no canto superior direito mostra o conjunto Pareto-ótimo e no último, todas as soluções MOBJ computadas são mostradas.

A.3 Listagem dos programas mais importantes - Linguagem: MATLAB

As Rotinas foram implementadas em MATLAB e estão preparadas para treinamento de redes com duas camadas, sendo uma intermediária. O número de neurônios do nível de entrada, da camada intermediária e da camada de saída é flexível deixando o usuário livre para a escolha da topologia.

Os principais programas são:

- Lançador do MOBJ implementado via método de relaxação;
- Lançador do MOBJ implementado via problema ε-restrito;
- Método Elpsoidal;
- Cálculo dos Gradientes.

A.3.1 Lançador do MOBJ - Método de relaxação

```
%-------------------------------------------------------------------------------
% Programa mobjr.m
% Objetivo     : Aplicacao de tecnicas de çotimizaao multi-objetivo para treinamento de rnas
%                a partir de dados de uma RNA superdimensionada, encontrar o Pareto-otimo
%
% Roselito de A Teixeira
% roselito@cpdee ufmg br
%-------------------------------------------------------------------------------
clear;
nntwarn off

disp('===============================================================================')
disp('_____________________________MOBJ__-__REGRESSION_____________________________')
disp('-------------------------------------------------------------------------------')
arqd = input('Entre_com_arquivo_de_dados                    :_','s');
arqr = input('Entre_com_arquivo_da_RNA                      :_','s');
arqs = input('Entre_com_arquivo_para_salvar_simulacao       :_','s');
arqt = input('Entre_com_arquivo_de_dados_de_teste(op)       :_','s');

disp('===============================================================================')
disp('Intervalos_para_geracao_do_vetor_v');
disp('-------------------------------------------------------------------------------')
var1 = input('Entre_com_um_vetor_3_posicoes_para_intervalo_1_[inicio_fim_n_pontos]_:_');
var2 = input('Entre_com_um_vetor_3_posicoes_para_intervalo_2_[inicio_fim_n_pontos]_:_');

disp('===============================================================================')
disp('Tamanho_dos_elipsoides');
disp('-------------------------------------------------------------------------------')


if isempty(var1) & isempty(var2)
        nr    = input('Entre_com_o_únmero_de_çôrepeties_________:_');
end

if ~isempty(var1)
        if length(var1) ~= 3
                disp('O_vetor_deve_possuir_tres_elementos_!');
                break;
        end
end

if ~isempty(var2)
        if length(var2) ~= 3
                disp('O_vetor_deve_possuir_tres_elementos_!');
                break;
        end
end

if ~isempty(arqt)
        feval('load',arqt);
        X_t = X'; % dados arquivo de teste com ruido
        D_t = D'; % dados arquivo de teste com ruido
end

if isempty(arqr) | isempty(arqd)
   help mobjr
   break;
else
   feval('load',arqr);
   feval('load',arqd);
end


% avaliar J1 no ponto f1star
f1star = w2wk(b1s1,w1s1,b2s1,w2s1);

J1star = (1/length(D)) * sum((D' - simuff(X',w1s1,b1s1,'tansig',w2s1,b2s1,'purelin')).^2);
J2corr = norm(f1star);


% f2* - çãsoluo que minimiza a segunda funcao objetivo-------------------------
f2star = zeros(size(f1star));
b1s2=zeros(size(b1s1));
w1s2=zeros(size(w1s1));
b2s2=zeros(size(b2s1));
w2s2=zeros(size(w2s1));
```

```
% avaliar J1 no ponto f2star
J1corr =  (1/length(D)) * sum((D' - simuff(X',w1s2,b1s2,'tansig',w2s2,b2s2,'purelin')) ^2);

% avaliar J2 no ponto f2star
J2star = norm(f2star);


% vetores de objetivos associados a cada fmnimo
f1 =[J1star J2corr]';
f2 =[J1corr J2star]';

% f* vetor de objetivos correspondente a  solucao ótpica do problema----------------------
futopi = [J1star J2star]';

% escolha de eta suficientemente grande
eta0 = 0.1;

w0 = 0.0001*randn(size(f1star));
w0 = [w0 ; eta0];

% gera alpha nao igualmente espacado-----------------------------------------------
if isempty(var1)
        alpha = linspace(0,1,nr);
elseif isempty(var2)
        alpha =  linspace(var1(1),var1(2),var1(3));
else
        alpha =  linspace(var1(1),var1(2),var1(3));
        alpha = [alpha linspace(var2(1),var2(2),var2(3))];
end
%-------------------------------------------------------------------------------

% Para otimizar, utilizacao do metodo elipsoidal
gfuncao = 'gfobjr1';   % calculo dos gradientes

Q0 = relip(1)*eye(length(w0),length(w0));

epsil(1) = 1e-9;
epsil(2) = 1e-3;
epsil(3) = 1e-4;

cpo = 1;
PO = [];
vx = [];

minimo_erro = 100;
fim =1;
relip = [10 2];

while (cpo <= length(alpha)) & fim == 1,

   v = futopi + alpha(cpo)*(f1 - futopi) + (1-alpha(cpo))*(f2 - futopi);
   [wstar] = elposcx(gfuncao,w0,Q0,epsil,D,X,eta0,v,futopi,cpo,neuro);

   Q0 = relip(2)*eye(length(w0),length(w0));  % reducao da regiao de procura
   w0 = wstar;                                % partindo do minimo da iteracao anterior

   PO = [PO wstar];

   % avaliacao se o alg. MOBJ ja encontrou a melhor solucao-----------------------------
   if ~isempty(arqt)
           [w1,b1,w2,b2,eta] = wk2w(PO(:,cpo),neuro,X,D);
           s_mobj = simuff(X_t,w1,b1,'tansig',w2,b2,'purelin');
           e =  (1/length(D_t))*sum((D_t - s_mobj) ^2);
           if e > minimo_erro
                fim = 0;
           else
                minimo_erro = e;
                cpo = cpo +1;
           end
   else
           cpo = cpo +1;
   end
   %-------------------------------------------------------------------------------
end
feval('save',arqs);
%-------------------------------<<fim arquivo>>-------------------------------------
```

A.3.2 Lançador do MOBJ (Método ϵ-restrito)

```
%-------------------------------------------------------------------------
% Programa mobjr_new_1 m
% Objetivo     : Aplicacao de tecnicas de çotimizaao multi-objetivo para treinamento de rnas
% é              mtodo Pe
%
% Roselito de A  Teixeira
% roselto@cpdee ufmg br
%-------------------------------------------------------------------------
clear;
nntwarn off

disp('=============================================================================')
disp('_____________________________MOBJ__-__REGRESSION_____________________________')
disp('-----------------------------------------------------------------------------')
arqd = input('Entre_com_arquivo_de_treinamneto ..... :_','s');
arqs = input('Entre_com_arquivo_para_salvar_simulacao :_','s');

neuro  = input('Entre_com_o_numero_de_neuronios_na_ce_:_');
dnorma = input('Entre_com_o_Delta_norma_ ...          :_');
miter  = input('Numero_de_maximo_de_Solucoes_PO_       :_');

parada = input('Usar_parada_por_validacao_(s/n):_','s');

if parada == 's'
        arqv = input('Entre_com_arquivo_de_Validacao          :_','s');
        load(arqv,'Xt','Dt','Xauxt','Dauxt');
else
        arqv = [];
end

load(arqd,'X','D','Xaux','Daux');

[w1s1,b1s1,w2s1,b2s1] = initff(X,neuro,'tansig',D,'purelin');
f1star = w2wk(b1s1,w1s1,b2s1,w2s1);

% f2* - solucao que minimiza a segunda funcao objetivo-------------------------
f2star = zeros(size(f1star));

b1s2=zeros(size(b1s1));
w1s2=zeros(size(w1s1));
b2s2=zeros(size(b2s1));
w2s2=zeros(size(w2s1));


% avaliar J1 no ponto f2star
J1corr =  (1/length(D)) * sum((D' - simuff(X',w1s2,b1s2,'tansig',w2s2,b2s2,'purelin')) ^2);

J2star =  0;

f2 =[J1corr ; J2star];

w0 = 0.001*randn(size(f1star));

% Para otimizar, utilizacao do metodo elipsoidal
gfuncao = 'gfobjr_new';   % gradiente

Q0 = 20*eye(length(w0),length(w0));

epsil(1) = 1e-9;
epsil(2) = 1e-5;
epsil(3) = 1e-7;

cpo = 1;
PO = [];
fim = 1;
minimo_erro = 100;
v = 0;

while (cpo <= miter)  &  fim,
   v = v + dnorma;

   disp(['Iteracao_:_' num2str(cpo)])

   [wstar] = elip_reg_new(gfuncao,w0,Q0,epsil,D,X,v,neuro);
   Q0 = 20*eye(length(w0),length(w0));
   w0 = wstar;                          % partindo do minimo da iteracao anterior
   PO = [PO wstar];
```

```
   % avaliacao se o alg  MOBJ ja encontrou a melhor solucao------------------------------
   if ~isempty(arqv)
           [w1,b1,w2,b2,eta] = wk2w(PO(:,cpo).neuro.X,D);
           s_mobj = simuff(Xt',w1,b1,'tansig',w2,b2,'purelin');
           e =  (1/length(Dt))*sum((Dt' - s_mobj)^2);
           if e > minimo_erro
                fim = 0;
           else
                minimo_erro = e;
                cpo = cpo +1;
           end
   else
           cpo = cpo +1;
   end
end
feval('save',arqs);
%-------------------------------<<fim arquivo>>-----------------------------------------
```

A.3.3 Método Elipsoidal - Algoritmo

Algoritmo 3 Elipsoidal - Minimizar η

Ajustar parâmetros iniciais; {Dados provenientes do Algoritmo 1}
$fim \Leftarrow 0$; {Variável auxiliar para parada}
$s \Leftarrow 0$
$s_1 \Leftarrow \frac{1}{(n+1.1)}$
Enquanto $fim \neq 1$ **do**
  Avaliar o gradiente da função a ser otimizada; {Ver Apêndice A}
  $mg_k \Leftarrow max(\mathbf{g}_k)$; {$\mathbf{g}_k$ é o gradiente da função}
  $\mathbf{g}_k \Leftarrow \mathbf{g}_k / mg_k$; {Gradiente normalizado}
  $\nu_k \Leftarrow \mathbf{g}_k^T * \mathbf{Q}_k * \mathbf{g}_k$;
  se $\nu_k \leq \epsilon_1$ **então**
    $\mathbf{d} \Leftarrow -\mathbf{Q}_k * \mathbf{g}_k / \sqrt{\nu_k}$;
    $\mathbf{w}_k \Leftarrow \mathbf{w}_k + \mathbf{d} * (1 + n * s)/(n+1)$; {Com *deep cut*, $s = 0.1$}
    $\mathbf{Q}_k \Leftarrow (n^2 * (1 - s^2)/(n^2 - 1)) * (\mathbf{Q}_k - (2 * (1 + n * s)/((n+1) * (1+s))) * \mathbf{d} * \mathbf{d}^T)$;
    $k \Leftarrow k + 1$;
  senão se $mg_k \geq \epsilon$ **então**
    $fim \Leftarrow 1$
  senão
    $fim \Leftarrow 1$
  fim se
  se $(k > num_iter$) ou $(((max(abs(\mathbf{w}_k - \mathbf{w}_{ka})) < \epsilon)$ e $(mg_k < \epsilon)))$ **então**
    $fim \Leftarrow 1$
  fim se
 fim Enquanto

A.3.4 Método Elipsoidal - Listagem do programa em Matlab

```
%-----------------------------------------------------------------------------
% Programa elip_reg_new  - metodo elipsoidal
%-----------------------------------------------------------------------------

function [wk] = elip_reg_new(gfuncao,wk,Qk,epsil,D.X,v,neuro)

k   = 0;
fim = 0;
n = length(wk);
s =  .01;
```

```
while (~fim)
        gk = feval(gfuncao,X,D,wk,v,neuro);
        mgk = max(abs(gk));
        gk = gk/mgk;
        vk = gk'*Qk*gk;
        if (vk > epsil(1))
                d = -Qk*gk/sqrt(vk);
                wka = wk;
                wk = wk + d*(1+n*s)/(n+1);
                Qk = (n^2*(1-s^2)/(n^2-1))*(Qk-(2*(1+n*s)/((n+1)*(1+s)))*d*d');
                Qk = 1e-10*eye(size(Qk)) + Qk;
                k = k + 1;

                if abs(norm(wk)-v)<.001
                        fim = 1;
                end
        else
                fim = 1;
        end

        if (k > 6000) | ((max(abs(wk-wka)) < epsil(3)) & (  mgk < epsil(2)  ))
                fim = 1;
                disp(['k : ' num2str(k) ' max(abs(wk-wka))= ' num2str(max(abs(wk-wka))) ' mgk : ' num2str(mgk)])
        end
end
%-------------------------------------------------------------------------------
```

A.3.5 Cálculo dos Gradientes

```
%-------------------------------------------------------------------------------
% Funcao [gk] = gfobjr_new(X,D,wk,v,neuro)
%
% Objetivo : funcao para calculo dos gradientes
%-------------------------------------------------------------------------------

function [gk] = gfobjr_new(X,D,wk,v,neuro)

length_D = length(D);
wkx = wk;
nkx = norm(wk);

[W1,B1,W2,B2,eta] = wk2w(wk,neuro,X',D');


erro   = D' - simuff(X',W1,B1,'tansig',W2,B2,'purelin');
aaaux  = tanh(W1*X'+B1*ones(size(X')));
aaux   = 1 - aaaux^2;

%-------------------------------------------------------------------------------
% gradiente em relacao aa W1
dJ_W1 = (-2/length_D) * erro*X'*aaux';
dJ_W1 = W2' *dJ_W1';
%-------------------------------------------------------------------------------
%-------------------------------------------------------------------------------
% gradiente em relacao aa B1
dJ_B1 = (-2/length_D) * erro*aaux';
dJ_B1 = W2' *dJ_B1';
%-------------------------------------------------------------------------------
%-------------------------------------------------------------------------------
% gradiente em relacao aa W2
dJ_W2 = (-2/length_D) * erro*aaaux';
dJ_W2 = dJ_W2';
%-------------------------------------------------------------------------------
%-------------------------------------------------------------------------------
% gradiente em relacao aa B2
dJ_B2 = (-2/length_D) * sum(D' - W2 * aaaux - B2);
%-------------------------------------------------------------------------------

% gradiente  da funcao objetivo
gk = [dJ_B1;dJ_W1;dJ_B2;dJ_W2];
%-------------------------------------------------------------------------------

%-------------------------------------------------------------------------------
% calculo do gradiente da restricao
Dg1 = wkx/nkx;
%-------------------------------------------------------------------------------

SEQ = (1/length_D) * sum((D' - simuff(X',W1,B1,'tansig',W2,B2,'purelin')) ^2);
g1 = nkx -v;
```

```
disp ([' erro : ' num2str(SEQ) ' == Norma: ' num2str(v) '/' num2str(nkx) ' »  Gl: ' num2str(gl)])

if gl > 0
      gk = Dgl;
end
%-------------------------------------------------<< fim arquivo>>-------------------------------
```