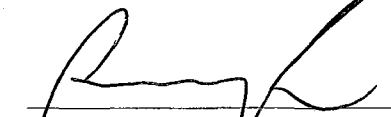
Aplicação do Método de Região de Confiança ao algoritmo Backpropagation

Izidro Avelino de Queiroz Neto

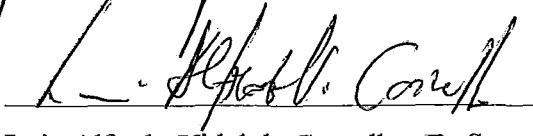
TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO EM ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por :

Paulo Roberto de Oliveira, Dr. Ing.



Ruy Luiz Milidiú, Ph. D.



Luiz Alfredo Vidal de Carvalho, D. Sc.

RIO DE JANEIRO, RJ - BRASIL

ABRIL DE 1995

QUEIROZ NETO, IZIDRO AVELINO DE

Aplicação do Método de Região de Confiança ao algoritmo Backpropagation (Rio de Janeiro) 1995.

VII, 70 p. 29,7 cm (COPPE/UFRJ, M. Sc., Engenharia de Sistemas e Computação, 1995)

Tese - Universidade Federal do Rio de Janeiro, COPPE.

1. Redes Neuronais 2. Métodos de Região de Confiança

I. COPPE/UFRJ II. Título (Série).

A Jaqueline e a nossos filhos : William e Amanda

Agradecimentos

Em primeiro lugar, à minha família (Jaqueline, William e Amanda), pela compreensão, carinho e incentivo nos momentos difíceis.

Ao meu orientador Paulo Roberto de Oliveira, pela ajuda e amizade, sempre.

Aos amigos Alexandre e Cristina, pelo estímulo no momento certo.

À Petrobrás e ao BNDES, que tornaram possível este mestrado.

Por último, à Internet, que permitiu o acesso a informações fundamentais para este trabalho.

Resumo da Tese apresentada à COPPE como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M. Sc.)

APLICAÇÃO DO MÉTODO DE REGIÃO DE CONFIANÇA AO ALGORITMO BACKPROPAGATION

Izidro Avelino de Queiroz Neto

Abril de 1995

Orientador : *Paulo Roberto de Oliveira*
Programa : *Engenharia de Sistemas e Computação*

RESUMO

Neste trabalho estudamos a Aplicação do Método de Região de Confiança (para Otimização irrestrita) ao algoritmo Backpropagation, com o objetivo de acelerar o aprendizado e obter maior robustez em sua convergência. Foi efetuado um teste comparativo com o algoritmo original e a variante RPROP (proposta em 1993), utilizando cinco problemas : ou exclusivo (XOR), codificadores 10-5-10 e 20-10-20, classificação de eletrofácies e previsão de preços de derivados de petróleo.

Abstract of Thesis presented to COPPE/UFRJ as partial fulfillment of the requirements for the degree of Master of Science (M. Sc.)

APPLICATION OF THE TRUST REGION METHOD TO BACKPROPAGATION ALGORITHM

Izidro Avelino de Queiroz Neto

April 1995

Thesis Supervisor : *Paulo Roberto de Oliveira*
Department : *Systems Engineering and Computer Science*

ABSTRACT

In this work we study the application of the Trust Region Method (in Unconstrained Optimization) to Backpropagation algorithm. Our goal is to accelerate the learning and to obtain more robustness in convergence. We do a comparative test with the original algorithm and the RPROP variation (introduced in 1993), using five benchmark problems : exclusive or (XOR), 10-5-10 and 20-10-20 encoders, eletrofacies identification and prediction of the price of oil products in the international market.

ÍNDICE

Capítulo I - Introdução **1**

Capítulo II - Redes Neuronais **3**

II.1 Origem 3

II.2 Estrutura 8

II.3 Principais Paradigmas 12

II.4 Exemplos de aplicações 16

Capítulo III - Backpropagation **19**

III.1 Origem 19

III.2 Descrição 20

III.3 Variantes do algoritmo 25

Capítulo IV - Métodos de Região de Confiança **29**

IV.1 Introdução 29

IV.2 Descrição 31

IV.3 Algoritmo Geral 32

IV.4 Estratégias para o Cálculo do Passo 34

Capítulo V - Aplicação do Método de Região de Confiança ao algoritmo Backpropagation **46**

V.1 Introdução 46

V.2 Implementação 47

V.3 Resultados 52

Capítulo VI - Conclusão **61**

Bibliografia **63**

Capítulo I

Introdução

Em 1990, tive a oportunidade de participar do primeiro Curso de Especialização em Redes Neuronais (CERN), ministrado na Petrobrás pelo Prof. Manoel Tenorio, da Purdue University. O curso tinha o objetivo de formar técnicos capazes de aplicar a tecnologia de redes neuronais na resolução de problemas na área de petróleo.

Ao final do curso, foram propostos diversos projetos (MORAES, 1990; CARDADOR, 1990; GOLDSCHMIDT, 1991; MELO, 1991; SATUF, 1993), entre os quais estava o de Classificação de Eletrofácies (RODRIGUES e QUEIROZ NETO, 1991; QUEIROZ NETO e RODRIGUES, 1991; RODRIGUES e QUEIROZ NETO, 1992), que desenvolvi em conjunto com Fernando da Silva Rodrigues, físico do Departamento de Exploração (DEPEX). Ao longo do projeto, observamos os problemas existentes com o algoritmo backpropagation, tais como o elevado tempo de treinamento e a dificuldade de ajustar a taxa de aprendizado. Isto despertou a curiosidade com relação ao algoritmo, levando a estudos que culminaram nesta tese de mestrado.

As *redes neuronais* pertencem à linha conexionista da Inteligência Artificial, cujo objetivo é desenvolver sistemas "inteligentes", a partir de modelos baseados nos neurônios do cérebro. No Capítulo II é apresentada uma visão geral das redes neuronais, incluindo sua origem, estrutura e principais regras de aprendizado.

Entre os diversos paradigmas de redes neuronais, o mais aplicado é, sem dúvida, o algoritmo **backpropagation**. Ele tem sido usado nas mais variadas tarefas, como : reconhecimento de voz, previsão no mercado de ações, identificação de rochas, etc.

O algoritmo backpropagation, no entanto, tem alguns problemas : o tempo de "aprendizado", com freqüência, é bastante elevado, e o algoritmo possui parâmetros (taxa de

aprendizado e momento) cujo ajuste não é trivial, sendo definidos via tentativa e erro. Assim, têm sido estudadas diversas variantes, objetivando solucionar os problemas citados. No Capítulo III, o algoritmo é descrito, apresentando as expressões de atualização dos pesos e principais variantes hoje existentes.

O algoritmo original se baseia em um método de otimização de primeira ordem (método do gradiente), onde é efetuado um passo de tamanho fixo (taxa de aprendizado), na direção oposta à do gradiente. Assim, algumas variantes propostas utilizam métodos de segunda ordem, na tentativa de acelerar a convergência. Em uma dessas variantes, proposta por BECKER e Le CUN (1989), devido a alguns problemas encontrados, sugere-se que a aplicação de uma estratégia que concilie o método de Newton com o método do gradiente poderia trazer bons resultados. Como esta é uma das características do Método de Região de Confiança, decidiu-se estudar a aplicação deste método ao algoritmo backpropagation.

O Método de Região de Confiança é descrito no Capítulo IV, sendo mostradas as estratégias existentes para o cálculo do passo ("hook step", "dogleg" e "double dogleg"), com alguns exemplos.

O Capítulo V apresenta os resultados encontrados na aplicação do Método de Região de Confiança ao algoritmo backpropagation. Para avaliar o desempenho da variação proposta, foram codificados os seguintes algoritmos : backpropagation e RPROP (RIEDMILLER e BRAUN, 1993), sendo este último a mais recente variante encontrada na literatura. Cada algoritmo foi testado em cinco tarefas : "ou exclusivo" (**XOR**), codificadores 10-5-10 e 20-10-20 (FAHLMAN, 1988), classificação de eletrofácies e previsão de preços de derivados de petróleo (MELO, 1991).

Finalmente, o Capítulo VI apresenta as conclusões encontradas.

Capítulo II

Redes Neuronais

II.1 Origem

Segundo HECHT-NIELSEN (1990), o início da neurocomputação é freqüentemente associado ao artigo de Warren MCCULLOCH e Walter PITTS (1943), embora estes autores nunca tenham mencionado usos práticos do seu trabalho. Este artigo, que mostrou que mesmo tipos simples de redes neuronais podiam, em princípio, calcular qualquer função aritmética ou lógica, foi amplamente divulgado e teve grande influência. Outros pesquisadores, principalmente John VON NEUMANN (1951, 1956), escreveram livros e artigos nos quais sugeriam que o estudo de computadores inspirados no cérebro podia ser interessante. Estas sugestões foram amplamente apreciadas, mas pouco de concreto aconteceu como resultado delas por um longo tempo.

Em 1949, Donald HEBB escreveu um livro intitulado *The Organization of Behavior,* no qual perseguia a idéia de que o condicionamento psicológico clássico está presente em todos os animais porque é uma propriedade dos neurônios individuais. Esta idéia não era intrinsecamente nova, mas Hebb a desenvolveu mais do que qualquer um antes dele, propondo uma regra de aprendizado específica para as sinapses dos neurônios. Hebb então usou esta regra de aprendizado para elaborar uma explanação qualitativa de alguns resultados experimentais da psicologia. Isto serviu para inspirar muitos outros pesquisadores a perseguir este mesmo tema - o que formou a base para o advento da neurocomputação.

O primeiro neurocomputador bem sucedido (o *Perceptron Mark I*) foi desenvolvido durante 1957 e 1958 por Frank Rosenblatt, Charles Wightman e outros (ROSENBLATT, 1958). Devido à sua profunda perspicácia, suas contribuições técnicas e sua maneira moderna de pensar sobre redes neuronais, muitos vêem Rosenblatt como o fundador

da neurocomputação como nós a conhecemos hoje. Seu principal interesse era o reconhecimento de padrões. Além de inventar o perceptron, Rosenblatt também escreveu um dos primeiros livros sobre neurocomputação, *Principles of Neurodynamics* (ROSENBLATT, 1961) - um livro que ainda é de leitura lucrativa.

Pouco depois de Rosenblatt, mas de importância semelhante, estava Bernard Widrow. Trabalhando com seus estudantes de graduação (mais notavelmente Marcian E. "Ted" Hoff, que mais tarde veio a inventar o microprocessador), desenvolveu um tipo distinto de rede neuronal chamado de ADALINE (WIDROW e HOFF, 1960), que era equipado com uma nova e poderosa regra de aprendizado que, ao contrário da regra de aprendizado do perceptron, está ainda sendo usada. Widrow e seus alunos aplicaram o ADALINE com sucesso a um grande número de problemas, e produziram vários filmes de seus sucessos. Widrow também fundou a primeira companhia de hardware para neurocomputadores (a Memistor Corporation), que realmente produziu neurocomputadores e componentes para venda na primeira metade da década de 60.

Apesar do considerável sucesso destes primeiros pesquisadores da neurocomputação, a área sofria de dois problemas óbvios. Primeiro, a maioria dos pesquisadores abordava o tema de um ponto de vista qualitativo e experimental, ao invés de usar um ponto de vista analítico (embora houvesse notáveis exceções, tais como Widrow). Esta ênfase experimental resultou em uma significativa perda de rigor científico e uma negligência nos conceitos que incomodava muitos cientistas e engenheiros que observavam a nova área.

Segundo, uma fração infelizmente grande dos pesquisadores da neurocomputação (e os jornalistas que com eles conversavam) se deixaram arrebatar pelo entusiasmo em seus textos e afirmações. Por exemplo, havia predições publicadas de que cérebros artificiais estariam disponíveis em poucos anos, e outras afirmações semelhantes. Este exagero desacreditou a área e irritou pessoas de outros campos.

Além do exagero e da perda de rigor geral, em meados dos anos 60 os pesquisadores tinham esgotado as boas idéias. Estava claro que, para que progressos adicionais fossem alcançados, algumas idéias radicalmente novas tinham que ser introduzidas - e não era aparente que estas idéias pudessem surgir em breve. Esta exaustão intelectual fez

com que muitos dos melhores pesquisadores deixassem a área. Muitos se transferiram para campos relacionados como o reconhecimento de padrões, processamento de imagem e processamento de sinais.

Em meados dos anos 60, estava claro que a era dos primeiros sucessos da neurocomputação estava próxima de terminar. O episódio final desta era foi a publicação do livro *Perceptrons* (MINSKY e PAPERT, 1969), no qual foi provado matematicamente que um perceptron não podia implementar a função lógica OU EXCLUSIVO (XOR) ($f(0,0) = f(1,1) = 0$, $f(0,1) = f(1,0) = 1$), nem muitas outras funções predicadas (funções escalares binárias de vetores binários). A tese implícita de *Perceptrons* era que essencialmente todas as redes neuronais sofriam do mesmo "defeito fatal" do perceptron; isto é, a inabilidade de calcular certos predicados essenciais como o XOR. Para reforçar este ponto de vista, os autores examinaram diversos melhoramentos propostos ao perceptron e mostraram que estes também não eram capazes de funcionar bem. Eles deixaram a impressão de que a pesquisa com redes neuronais não tinha futuro, levando à redução dos fundos que a financiavam.

Entre 1967 e 1982, pouco em redes neuronais foi produzido nos Estados Unidos (a pesquisa no Japão, Europa e União Soviética foi menos afetada pelo impacto do livro *Perceptrons*). Entretanto, uma boa parte dos resultados em redes neuronais estava sob as denominações de processamento adaptativo de sinais, reconhecimento de padrões e modelagem biológica. Um aspecto interessante deste período de "pesquisa silenciosa" foi o grande fluxo de novos e talentosos pesquisadores durante o período de 1966 e 1969, como por exemplo Shun-ichi AMARI (1967), James ANDERSON (1968), Kunihiko FUKUSHIMA (1969), Stephen GROSSBERG (1969), Harry KLOPF (1969), Teuvo KOHONEN (1970) e David WILLSHAW (1969). Estes, e os que vieram nos 13 anos seguintes, estabeleceram uma base sólida para a neurocomputação e prepararam o caminho para o renascimento da área.

No início dos anos 80, diversos pesquisadores da neurocomputação começaram a apresentar novas propostas para o desenvolvimento de neurocomputadores e aplicações de redes neuronais. A primeira ruptura aconteceu na Agência de Projetos Avançados de Pesquisa da Defesa (Defense Advanced Research Projects Agency - DARPA), onde Ira Skurnick se recusou a seguir cegamente os padrões estabelecidos, e passou a ouvir os argumentos que os pesquisadores da neurocomputação apresentavam a favor dos seus projetos. Divergindo da tradição, Skurnick começou a patrocinar a pesquisa em neurocomputação em 1983. Dada a

importância do DARPA como referência para outros centros de pesquisa, esta atitude abriu as comportas. Em meses, executivos de outros centros de pesquisa (muitos dos quais já tinham demonstrado interesse na área, mas não tiveram a iniciativa de Skurnick) haviam aderido.

Outra fonte de estímulo dos anos de 1983 a 1986 foi John Hopfield, um conhecido físico que se interessou por redes neuronais poucos anos antes. Hopfield escreveu dois artigos (HOPFIELD, 1982 e 1984) que, em conjunto com suas muitas conferências através do mundo, persuadiram centenas de cientistas, matemáticos e tecnólogos altamente qualificados a se juntar ao campo emergente das redes neuronais. De fato, no início de 1986, aproximadamente um terço das pessoas da área tinham sido trazidos diretamente por Hopfield ou por um dos seus primeiros "convertidos". O trabalho de Hopfield como recrutador foi talvez a contribuição individual mais importante para o ressurgimento da área. Na verdade, em alguns círculos, surgiu a confusão de que Hopfield tinha *inventado* a neurocomputação (ou que tinha pelo menos feito as descobertas fundamentais que levaram à revitalização da área). Esta crença causou considerável desconforto em parte dos verdadeiros pioneiros - particularmente aqueles que suportaram os "anos silenciosos" na obscuridade e estavam agora esperando que seu esforço fosse devidamente reconhecido (o que efetivamente aconteceu, mas não antes de 1987).

Em 1986, com a publicação do "PDP" (*Parallel Distributed Processing, Volumes I e II*, editados por David RUMELHART e James MCCLELLAND), a pesquisa explodiu. Em 1987, a primeira conferência sobre redes neuronais nos tempos modernos (IEEE International Conference on Neural Networks - 1700 participantes) foi realizada em San Diego, e a Sociedade Internacional de Redes Neuronais (International Neural Network Society - INNS) foi formada. Em 1988 o periódico da INNS *Neural Networks* foi fundado, seguido pelo *Neural Computation* em 1989 e pelo *IEEE Transactions on Neural Networks* em 1990 (e subseqüentemente, por muitos outros). No início de 1987, várias das principais universidades anunciaram a formação de institutos de pesquisa e programas educacionais ligados à neurocomputação.

No Brasil, diversos pesquisadores têm se destacado, como Armando Freitas (UNICAMP), Ricardo Machado e Valmir Barbosa (IBM) e Luís Alfredo V. de Carvalho (COPPE). Entre as diversas aplicações desenvolvidas, pode-se citar a previsão de séries temporais (MELO, 1991; RAPOSO, 1992; ZANDONADE, 1993; ABELEM, 1994),

interpretação de imagens de satélite (MACHADO e BARBOSA, 1992), integração com sistemas especialistas (MACHADO, FERLIN, ROCHA e SIGULEM, 1991) e em análise combinatória (CARVALHO e BARBOSA, 1989; BARBOSA e CARVALHO, 1990; CARVALHO e BARBOSA, 1990; BARBOSA e CARVALHO, 1991).

II.2 Estrutura

Antes de descrever os componentes e o funcionamento de uma rede neuronal, é instrutivo examinar brevemente as estruturas correspondentes do cérebro que inspiraram a neurocomputação. A figura abaixo resume a descrição a seguir, que é o modelo clássico para o neurônio.

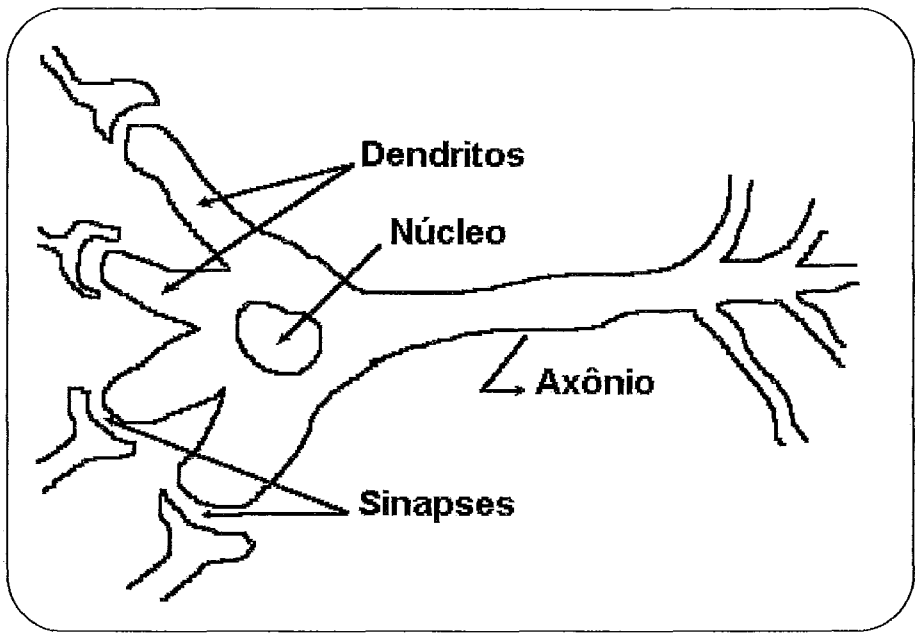


Figura II.1 : O neurônio

O *neurônio* é a célula fundamental do sistema nervoso e, em particular, do cérebro. Seu núcleo é uma unidade de processamento simples que recebe e combina sinais de muitos outros neurônios através de filamentos de entrada chamados *dendritos*. Se o sinal resultante é forte o suficiente, ele ativa o neurônio, que produz um sinal de saída; o filamento que transmite este sinal de saída é chamado de *axônio*.

O cérebro consiste de dezenas de bilhões de neurônios densamente interconectados. O axônio (saída) de um neurônio se divide e se conecta aos dendritos (entrada) de outros neurônios através de uma junção chamada de *sinapse*. A transmissão através desta junção é química em sua natureza e a quantidade de sinal transferida depende da força sináptica da junção. É esta força sináptica que é modificada quando o cérebro aprende e a sinapse pode ser considerada a unidade básica de memória do cérebro.

Em uma rede neuronal, a unidade análoga ao neurônio biológico é chamada de elemento de processamento (EP). Um EP pode ter várias entradas (equivalentes aos dendritos), formando um vetor $\mathbf{X} = (x_0, x_1, x_2, \ldots, x_n)$, e possui apenas um único valor de saída y_j (equivalente ao axônio), também chamado de grau de ativação (veja Figura II.2). A cada componente de **X** é associado um peso w_{ij}, que regula a influência daquele componente na ativação do EP, de forma semelhante às sinapses.

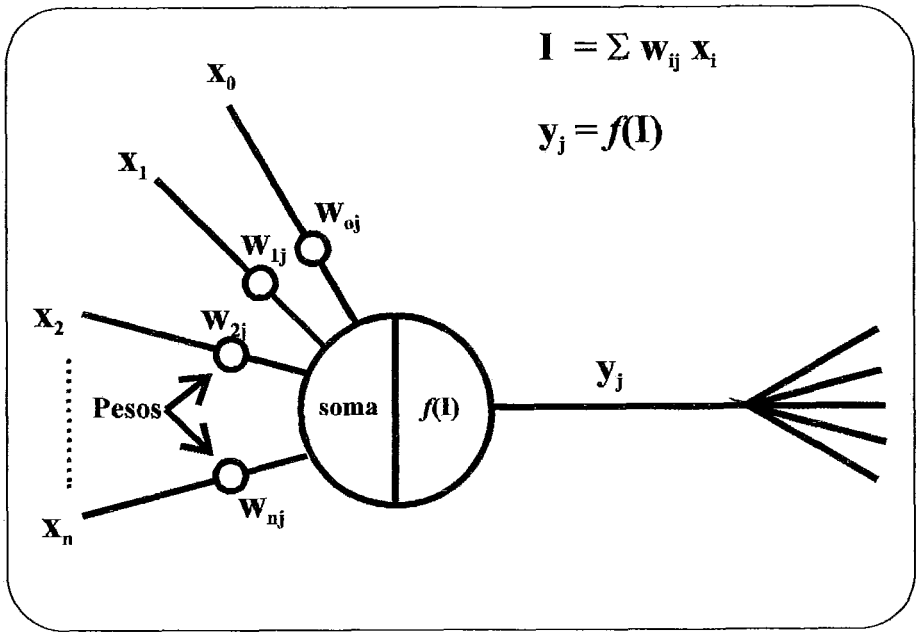


Figura II.2 : Elemento de processamento

A entrada total (I) é definida como :

$$I = \sum w_{ij} x_i$$

A esta entrada total é aplicada uma função de transferência f, que pode ser uma função limiar que somente passa informação caso I ultrapasse um certo nível, ou uma função contínua. Assim :

$$y_j = f(I)$$

Os EP's são normalmente organizados em uma sequência de camadas, de forma que os valores de saída de uma camada servem de entrada para a camada seguinte. A Figura II.3 mostra a arquitetura de uma rede neuronal simples. A primeira camada é chamada de camada de entrada, que recebe os estímulos apresentados à rede; a camada seguinte é chamada de camada escondida ("hidden layer"), pois não tem contato com o exterior; e a última é chamada de camada de saída, e apresenta a resposta da rede para um determinada

entrada.

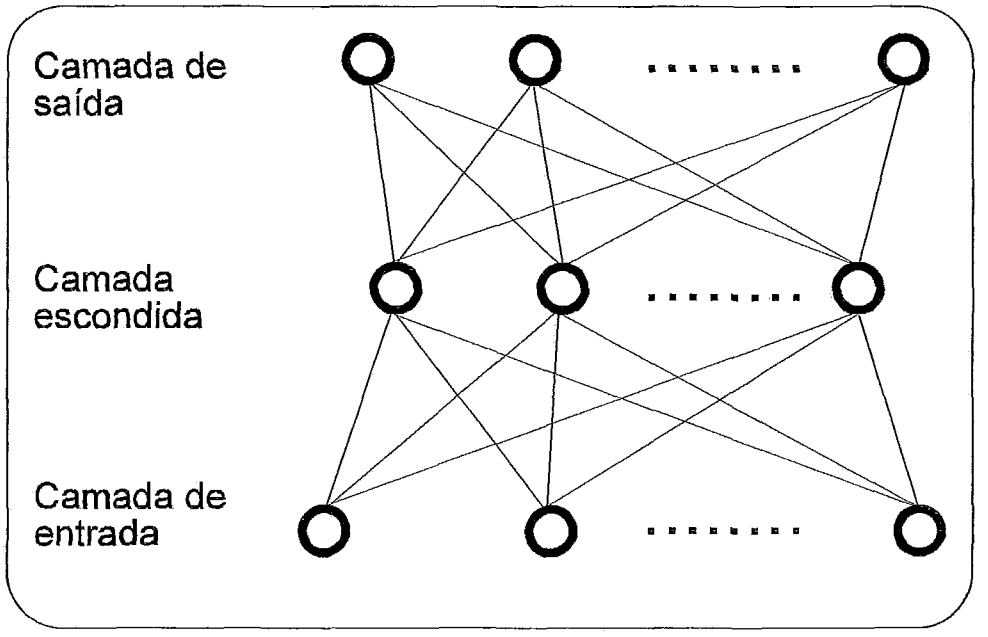


Figura II.3 : Arquitetura de uma rede neuronal

Há duas fases distintas no funcionamento de uma rede - *aprendizado* e *recordação* ("recall"). O aprendizado é o processo de adaptação dos pesos em resposta aos estímulos apresentados à camada de entrada. Os estímulos utilizados no aprendizado formam um conjunto de exemplos, onde a cada entrada corresponde uma saída desejada, ou seja, a resposta que deve ser apresentada pela rede neuronal. Neste tipo de aprendizado, chamado de *supervisionado*, cada exemplo é apresentado à rede, calculando-se a resposta correspondente; esta resposta é comparada com a saída desejada, determinando o nível de *erro* atual. Este valor do erro é utilizado para modificar os pesos, de forma a minimizá-lo. Este processo se repete até que o nível de erro alcance um valor aceitável, ou algum critério de convergência seja atingido.

Se o número de elementos de saída é diferente do número de elementos de entrada, a rede treinada é chamada de rede *heteroassociativa*. Se, para todos os exemplos de treinamento, a dimensão do vetor desejado de saída é igual à do vetor de entrada, a rede é chamada de *autoassociativa*. Se saídas desejadas não são apresentadas à rede, o aprendizado é dito *não supervisionado*.

Um terceiro tipo de aprendizado situado entre o supervisionado e o não supervisionado é o chamado *aprendizado de reforço,* onde um "professor" externo apenas indica se a resposta a uma determinada entrada está correta ou não.

Independentemente do tipo de aprendizado usado, uma característica essencial da qualquer rede é a sua *regra de aprendizado,* que especifica como os pesos vão se adaptar em resposta a um exemplo. Freqüentemente, o aprendizado requer que os exemplos sejam apresentados à rede milhares de vezes. Os parâmetros que governam uma regra de aprendizado podem mudar significativamente o tempo necessário para que este aprendizado seja bem-sucedido.

A recordação consiste na apresentação a uma rede já treinada de valores de entrada, aos quais a rede responderá de maneira coerente com os exemplos vistos no aprendizado. Normalmente, a recordação é uma parte do processo de aprendizado, uma vez que a resposta desejada deve ser comparada com a saída produzida pela rede.

II.3 Principais Paradigmas

II.3.1 Perceptron

O Perceptron, desenvolvido por Rosenblatt, foi projetado para modelar a capacidade de reconhecimento de padrões do sistema visual. Este era um objetivo bastante ambicioso. No entanto, o paradigma é bem simples e ilustra várias das idéias básicas relacionadas com elementos de processamento.

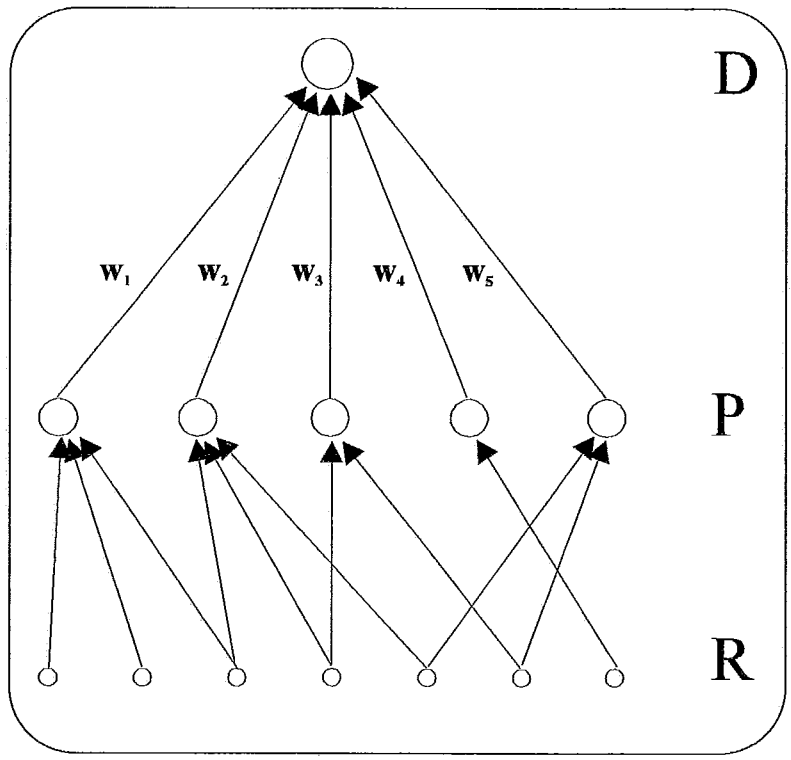


Figura II.4 : Perceptron

A rede neuronal do tipo Perceptron é basicamente uma rede de três camadas. A primeira camada possui unidades de entrada - originalmente unidades R (de retina) -, a segunda unidades chamadas de predicados (P) e a última unidades de decisão (D) (Figura II.4). Na versão mais tradicional todas as unidades são binárias - no entanto, o sistema funciona também para entradas contínuas (LIPPMAN, 1987). As unidades do tipo P e D empregam função de saída com limiar. Os pesos das ligações R $\rightarrow$ P são fixos ao longo de todo o funcionamento do sistema, enquanto que as ligações P $\rightarrow$ D são variáveis; só existem ligações R $\rightarrow$ P e P $\rightarrow$ D. Ou seja, há apenas *uma* camada de pesos variáveis. Assim, na

prática (e no algoritmo abaixo) considera-se que o sistema possui apenas duas camadas de unidades.

Um Perceptron é, basicamente, um *classificador de padrões*. Quando há apenas uma unidade de decisão, o sistema deve separar os estímulos de entrada em duas categorias distintas. Dada uma entrada, o sistema responde com a saída 1 (um) se o estímulo pertence a uma determinada classe e 0 (zero) caso contrário. O fato que gerou interesse nos Perceptrons foi a capacidade destes aprenderem um conjunto de pesos capaz de classificar a entrada em duas classes distintas ou, mais precisamente, que um Perceptron é um dispositivo capaz de aprender um discriminante linear.

O algoritmo proposto por Rosenblatt está descrito abaixo. O passo mais importante é o Passo 4; nele os pesos são ajustados de uma forma bastante simples. Como em todo algoritmo de correção de erros, os pesos são alterados de acordo com a discrepância do resultado desejado e o efetivamente gerado pela rede - só ocorre alteração do peso no caso do sistema fazer a classificação incorreta. O peso é incrementado de η quando a rede gerou uma saída zero e a saída desejada era um, ou seja, os pesos não eram suficientemente fortes para gerar uma saída um. Este incremento só é efetivado quando a unidade correspondente está ativa; isto garante que ela realmente contribuiu para o erro. A idéia de alterar os pesos apenas nos casos onde a unidade pré-sináptica está ativa é bastante importante e é empregada na grande maioria dos algoritmos de aprendizado - sejam eles supervisionados ou não. Um raciocínio análogo, mas inverso, permite concluirmos que no caso complementar de discrepância o peso é decrementado de η.

- **Algoritmo de Aprendizado do Perceptron**

  - **Passo 1. Inicialização**

    Os pesos e o limiar são inicializados. w_i e θ recebem valores aleatórios pequenos diferentes de zero. O índice i varia nas unidades do tipo predicado e θ é o limiar da unidade de decisão, d.

  - **Passo 2. Entrada**

    Apresentação de entrada e saída desejada. Entrada: o padrão (binário) S_k é apresentado gerando o vetor (o_1 , ... , o_n) da camada de entrada; saída desejada : t_d.

- **Passo 3. Cálculo da saída efetiva**

  Saída efetiva o_d é dada por

  $$o_d = f((\sum w_i(t).o_i) - \theta)$$

  onde f é a função limiar, ou seja, o_d valerá um se a entrada na unidade exceder θ, zero caso contrário; t representa o número da iteração.

- **Passo 4. Adaptação**

  Os pesos w_i são alterados :

  $$w_i(t+1) = w_i(t) + \eta(t_d - o_d).o_i$$

- **Passo 5. Loop**

  Vá para o Passo 2. (Pode-se interromper o algoritmo quando os pesos não mais se alteram ou quando um contador exceder um certo valor).

Este algoritmo, extremamente simples, é a base de muitos outros algoritmos supervisionados. Apesar de sua simplicidade, ele é capaz de implementar qualquer função linearmente separável.

II.3.2 ART

A Teoria de Ressonância Adaptativa (Adaptive Resonance Theory - ART) se baseia no trabalho desenvolvido no início da década de 60 em reconhecimento de padrões e modelagem neuronal, e foi introduzida por GROSSBERG (1976) e posteriormente elaborada por CARPENTER e GROSSBERG (1987). Os objetivos desta teoria incluem o desenvolvimento de detectores biologicamente plausíveis e classificadores de padrões.

A plausibilidade biológica impôs diversas restrições. Primeiro, os EP's obedecem a equações diferenciais ("equações de membrana"), baseadas em equações não lineares que descrevem a condutância, permeabilidade e atividade das membranas neuronais.

Outra restrição é a *localidade*, que estipula que a informação requer um substrato físico para sua transmissão. Por exemplo, o estado de uma sinapse pode ser influenciado apenas pelos estados dos componentes em contato físico com ela, e não pode ser diretamente influenciada por componentes fisicamente separados.

A restrição final é o chamado *Dilema da estabilidade-plasticidade* e se refere ao fato de que uma rede, em um ambiente desconhecido, deve ser estável o suficiente para não

perder as categorias já aprendidas, e ainda deve ter plasticidade para assimilar um estímulo suficientemente diferente das categorias já estabelecidas.

Além de ser biologicamente plausível, no algoritmo ART o aprendizado ocorre sem um professor explícito - o chamado "aprendizado não supervisionado".

Os sistemas ART são usados em muitos modelos psicológicos/biológicos, tais como Condicionamento Pavloviano (GROSSBERG e SCHMAJUK, 1987), Percepção visual e Reconhecimento de palavras (GROSSBERG e STONE, 1986), dentre outros. Outra aplicação bastante importante é o reconhecimento de padrões, como a classificação de imagens de radar. Em todos estes casos, o modelo geral inclui um ou mais módulos ART acoplados aos outros elementos do sistema. Por exemplo, no reconhecimento de padrões, o módulo ART realiza as categorizações após um módulo pré-processador ter tornado os estímulos de entrada invariantes à translação, escala e rotação.

II.3.3 Backpropagation

Sem dúvida, o algoritmo de aprendizado mais difundido é o algoritmo backpropagation. Esta popularidade advém da habilidade deste tipo de rede de aprender complicados mapeamentos multidimensionais. Nas palavras de WERBOS, Backpropagation vai "além da regressão".

Foi o algoritmo backpropagation que resolveu o problema teórico apresentado pelo livro *Perceptron*, que causou a "depressão" nas pesquisas com redes neuronais no final da década de 60. No caso, propôs-se a inclusão de uma camada intermediária, entre a camada de entrada e a de saída, e um algoritmo para atualizar seus respectivos pesos. Sua aplicação bem-sucedida a diversos problemas contribuiu significativamente para a "reabilitação" das redes neuronais.

O algoritmo backpropagation é descrito detalhadamente no **Capítulo III**.

II.4 Exemplos de aplicações

II.4.1 Conversão texto-voz

Terrence SEJNOWSKY e Charles ROSENBERG (1986) desenvolveram uma rede neuronal capaz de ler textos em voz alta. Isto foi feito de modo semelhante à forma como uma criança aprende a ler.

Primeiro, um texto simples foi transcrito por um lingüista em seus componentes silábicos (fonemas). À rede neuronal era então apresentado o texto, que começava a "ler" com a ajuda de um sintetizador de voz, chamado DECtalk. Este era usado apenas para converter fonemas em som; a rede de Sejnowsky e Rosenberg convertia o texto em fonemas, e era chamada NETtalk.

Inicialmente, o computador lia todas as palavras de forma contínua, porque os pesos entre os elementos de processamento eram aleatórios. Durante o processo de aprendizado supervisionado, eram apresentados ao sistema os fonemas transcritos pelo lingüista, os quais eram comparados com o seu próprio conjunto de fonemas para ajustar os pesos, corrigindo seus erros.

Após várias sessões de treinamento como esta, o computador era também capaz de inferir regras sobre a leitura de textos em Inglês, inicialmente fazendo distinções óbvias tais como a diferença entre vogais e consoantes, mas eventualmente aprendendo distinções mais difíceis. Inicialmente balbuciando como um bebê, depois de uma noite de aprendizado, ele podia falar como uma criança de seis anos.

Apesar da conversão texto-voz já ter sido feita antes, usando outros sistemas computacionais, a abordagem via redes neuronais é vantajosa pois elimina a necessidade de programar um conjunto complexo de regras de pronúncia em um computador.

II.4.2 Reconhecimento de Caracteres

Redes neuronais podem tratar grande quantidade de informação de uma só vez e são especialmente hábeis em reconhecimento de padrões. Estas características permitiram o desenvolvimento de interessantes aplicações na área de reconhecimento de caracteres.

Pesquisadores da NESTOR INC. (1988) desenvolveram uma rede neuronal que aceita manuscritos digitalizados como entrada. Após ser treinada a interpretar um conjunto de caracteres manuscritos, a rede neuronal foi capaz de interpretar tipos de letras que ela não tinha visto antes. A acurácia da rede pode ser muito melhorada por uma rápida sessão de treinamento com estes caracteres.

Outros sistemas, via métodos tradicionais, são também capazes de reconhecer caracteres manuscritos; a qualidade que distingue o método via redes neuronais é a flexibilidade. Após o treinamento, a rede neuronal é capaz de reconhecer uma variedade de estilos de manuscritos e de fazer melhores escolhas quando confrontada com um caracter confuso. Esta flexibilidade permitiu à Nestor Inc. desenvolver uma rede neuronal que aceita caracteres kanji como entrada.

II.4.3 Reconhecimento de Imagens

GORMAN e SEJNOWSKI (1988) aplicaram redes do tipo backpropagation para classificar alvos de sonar. A rede treinada possui desempenho semelhante a operadores de sonar treinados com os mesmos dados e significativamente melhor que um classificador convencional.

Os dados do sonar, que consistia de sinais gerados por um cilindro de metal e por uma rocha de tamanho similar, foi preprocessado através da integração do sinal ao longo do tempo de cada uma de 60 faixas de freqüência, de forma a produzir uma seqüência de 60 valores de energia espectral. Estes valores formavam a entrada para 60 elementos de processamento na camada de entrada da rede neuronal. Os dois elementos de saída codificavam a classificação - (1,0) para o cilindro, (0,1) para a rocha. Uma camada escondida foi usada nos testes. O número de elementos nesta camada foi variado, sendo que 12 unidades produziram os melhores resultados.

II.4.4 Processamento de Sinais

LAPEDES e FARBER (1987) mostraram que, para séries temporais "caóticas", as redes do tipo backpropagation ultrapassam os métodos preditivos convencionais (linear e

polinomial) em muitas ordens de magnitude. Estas redes aparentemente "aprendem" uma aproximação da regra que gera tal série.

Um exemplo deste tipo de série é a gerada pela seguinte equação :

$$x(t+1) = 4.x(t).(1-x(t))$$

Esta tem uma estrutura "escondida" muito simples, mas gera uma série complexa. Por causa da estrutura polinomial deste exemplo, resultados preditivos acurados podem também ser alcançados usando métodos polinomiais. Entretanto, Lapedes e Farber examinaram exemplos mais complexos que não eram adequados à predição polinomial. O método de implementação é o seguinte : os EP's de entrada representavam uma seqüência de amostras igualmente espaçadas da série :

$$x(t),\ x(t-d),\ x(t-2d),\ \ldots,\ x(t-nd)$$

A saída é uma amostra em um tempo à frente :

$$x(t+d)$$

A rede é treinada com conjuntos de amostras previamente observadas.

II.4.5 Problemas Combinatórios

Sistemas baseados em redes neuronais têm demonstrado grande potencial para resolver certos problemas combinatórios como o problema do caixeiro viajante. Este pertence a uma classe de problemas chamados de "NP-completos". "NP" significa não polinomial; soluções conhecidas para esta classe de problemas têm tempo de computação que cresce exponencialmente com o número de entradas. No problema do caixeiro viajante, o objetivo é escolher a menor rota possível tal que o caixeiro viajante passe por todas as cidades de uma determinada região.

John HOPFIELD e David TANK (1985) desenvolveram uma rede neuronal cujo objetivo é resolver o problema acima. Eles representaram as distâncias entre as cidades como pesos das ligações entre elementos de processamento. Quando estes EP's atingem uma configuração estável, uma solução foi encontrada. Esta configuração corresponde a um mínimo de uma função de energia que é definida para cada estado da rede.

Capítulo III

Backpropagation

III.1 Origem

A rede neuronal **backpropagation** é um dos mais importantes desenvolvimentos da neurocomputação. É uma potente rede associativa que tem sido aplicada com sucesso a uma grande variedade de problemas, que vai desde apuração de risco de crédito até compressão de imagem.

Inicialmente se acreditava que backpropagation havia sido introduzida por Paul WERBOS em 1974, por David PARKER em 1984/1985 e por David Rumelhart, Ronald Williams e outros membros do "PDP group" em 1985. No entanto, um algoritmo recursivo, matematicamente similar, foi apresentado por Arthur BRYSON e Yu-Chi HO em 1969.

De qualquer modo, não há dúvida de que o crédito por desenvolver o algoritmo backpropagation como uma técnica utilizável, assim como divulgar a arquitetura para uma larga audiência, pertence inteiramente a Rumelhart e outros membros do "PDP group". Antes do seu trabalho, backpropagation era pouco apreciado e obscuro. Hoje, é um dos pilares da neurocomputação.

III.2 Descrição

A Figura III.1 mostra a arquitetura de uma rede backpropagation típica. Ela sempre tem uma camada de entrada, uma de saída e pelo menos uma camada "escondida", onde cada camada está totalmente conectada à seguinte. Não existe limite teórico para o número de camadas escondidas, mas tipicamente usa-se uma ou duas. Para simplificar a notação, utilizamos apenas uma camada escondida. As convenções adotadas são as seguintes :

o_{pk} - valor de saída (resposta) do elemento de processamento (EP) k da camada de saída.

w_{jk} - peso entre o EP j da camada escondida e o EP k da camada de saída.

h_{pj} - valor de saída do EP j da camada escondida.

v_{ij} - peso entre o EP i da camada de entrada e o EP j da camada escondida.

x_{pi} - componente i do vetor de entrada x_p.

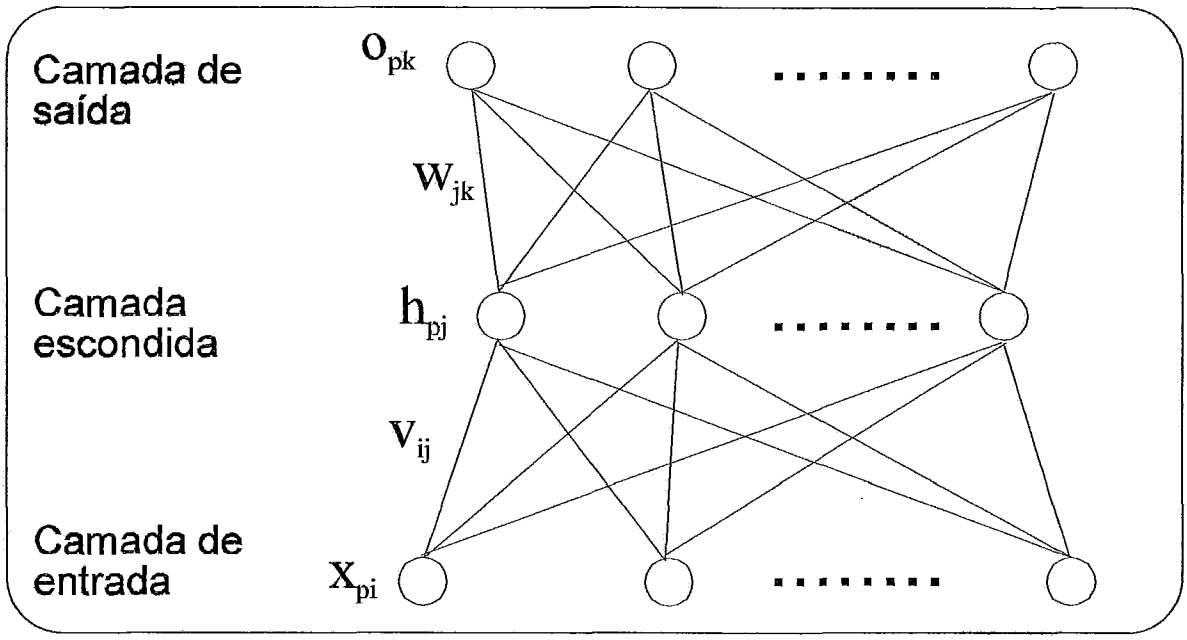


Figura III.1 - Rede Backpropagation

O objetivo deste tipo de rede é obter uma aproximação de uma função $f: \Re^n \rightarrow \Re^m$, através do "aprendizado" de exemplos (x_1, y_1), (x_2, y_2), ... , (x_p, y_p), ..., (x_e, y_e), onde $y_p = f(x_p)$. As dimensões da função definem a entrada e a saída da rede : n elementos na camada de entrada, m na camada de saída. O aprendizado consiste na minimização da seguinte função objetivo, que representa o nível de erro total da rede :

$$E = \frac{1}{2} \sum_p \sum_k (y_{pk} - o_{pk})^2 \qquad (1)$$

onde y_{pk} é o componente k de y_p.

o_{pk} é o componente k da resposta da rede neuronal à entrada x_p.

Um elemento de processamento da camada escondida é apresentado na figura III.2. Seu valor de saída (h_{pj}) é definido como o valor da função de transferência s (descrita a seguir), aplicada à soma ponderada de seus valores de entrada (net_{pj}) :

$$net_{pj} = \sum_i v_{ij} x_{pi}$$

$$h_{pj} = s(net_{pj})$$

onde v_{ij} é o peso entre o EP i da camada de entrada e o EP j da camada escondida.

x_{pi} é o componente i de x_p.

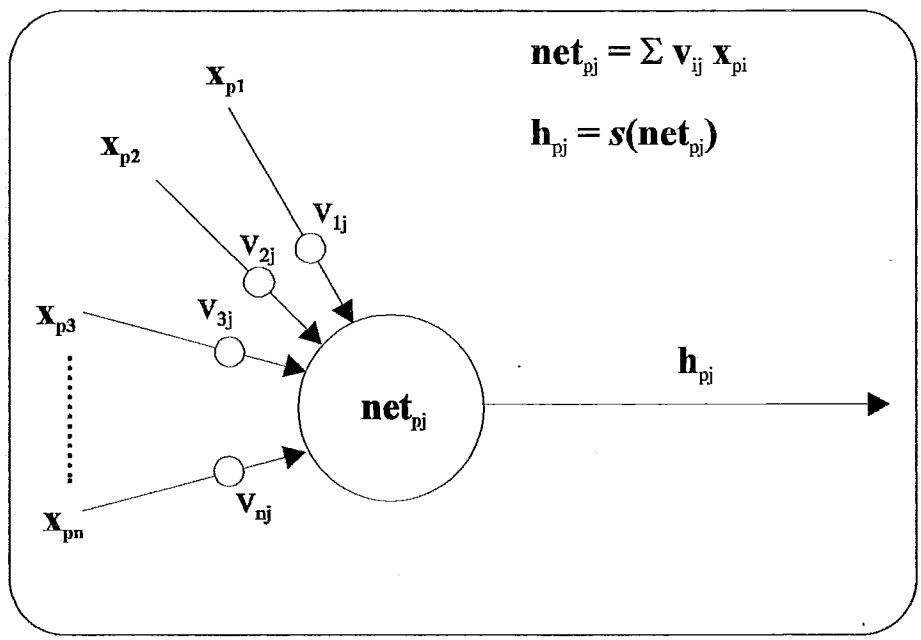


Figura III.2 - Elemento de processamento da camada escondida.

A função s é tradicionalmente a função sigmóide (Fig. III.3), mas pode ser qualquer função diferenciável. A função sigmóide é definida como :

$$s(x) = (1 + e^{-x})^{-1}$$

A derivada de s pode ser expressa como uma função simples dela mesma :

$$s'(x) = s(x)(1 - s(x))$$

O cálculo de o_{pk} é feito a partir dos valores de saída da camada escondida :

$$net_{pk} = \sum_j w_{jk} h_{pj}$$

$$o_{pk} = s(net_{pk})$$

onde w_{jk} é o peso entre o EP j da camada escondida e o EP k da camada de saída.

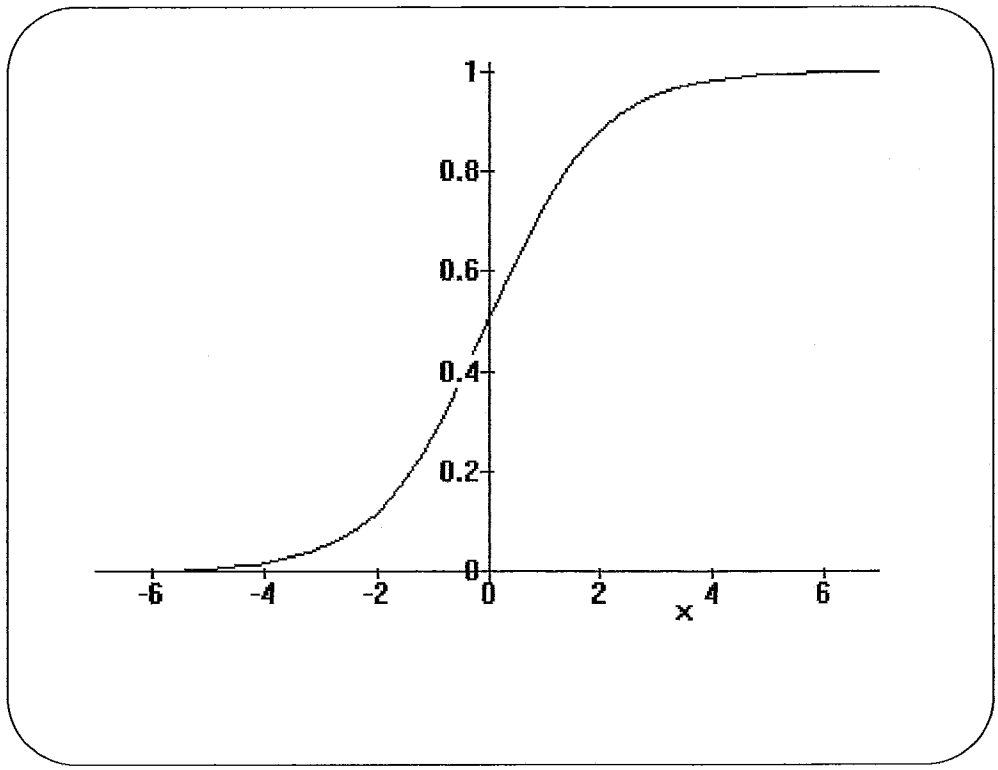


Figura III.3 - Função sigmóide.

O algoritmo padrão se baseia na minimização, em relação aos pesos, da função objetivo na direção oposta à do gradiente, usando um passo fixo η, que é chamado de "taxa de aprendizado". Ou seja, deseja-se obter um conjunto de pesos v_{ij} e w_{jk} correspondente ao mínimo da função de erros **(1)**. Assim :

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}} \qquad \textbf{(2)}$$

$$\Delta v_{ij} = -\eta \frac{\partial E}{\partial v_{ij}} \qquad \textbf{(3)}$$

Calculando-se o gradiente em relação aos pesos, tem-se :

- em relação a w_{jk} :

$$\frac{\partial E}{\partial w_{jk}} = -\sum_p (y_{pk} - o_{pk}) \frac{\partial o_{pk}}{\partial w_{jk}} \qquad \textbf{(4)}$$

$$\frac{\partial o_{pk}}{\partial w_{jk}} = \frac{\partial o_{pk}}{\partial net_{pk}} \frac{\partial net_{pk}}{\partial w_{jk}} = s'(net_{pk}).h_{pj} \qquad \textbf{(5)}$$

Define-se o sinal de erro em um EP da camada de saída como :

$$\delta_{pk} = s'(net_{pk})(y_{pk} - o_{pk}) \quad \textbf{(6)}$$

Substituindo **(5)** e **(6)** em **(4)**, tem-se :

$$\frac{\partial E}{\partial w_{jk}} = -\sum_{p} \delta_{pk} h_{pj} \quad \textbf{(7)}$$

- em relação a v_{ij} :

$$\frac{\partial E}{\partial v_{ij}} = -\sum_{p} \sum_{k} (y_{pk} - o_{pk}) \frac{\partial o_{pk}}{\partial v_{ij}}$$

$$\frac{\partial o_{pk}}{\partial v_{ij}} = \frac{\partial o_{pk}}{\partial net_{pk}} \frac{\partial net_{pk}}{\partial h_{pj}} \frac{\partial h_{pj}}{\partial net_{pj}} \frac{\partial net_{pj}}{\partial v_{ij}}$$

$$\frac{\partial o_{pk}}{\partial net_{pk}} = s'(net_{pk})$$

$$\frac{\partial net_{pk}}{\partial h_{pj}} = w_{jk}$$

$$\frac{\partial h_{pj}}{\partial net_{pj}} = s'(net_{pj})$$

$$\frac{\partial net_{pj}}{\partial v_{ij}} = x_{pi}$$

Define-se o sinal de erro em um EP da camada escondida como :

$$\delta_{pj} = s'(net_{pj}) \sum_{k} \delta_{pk} w_{jk}$$

Substituindo as expressões acima, chega-se a :

$$\frac{\partial E}{\partial v_{ij}} = -\sum_{p} \delta_{pj} x_{pi} \quad \textbf{(8)}$$

Como se pode observar, o cálculo do sinal de erro δ_{pj} é feito a partir dos sinais obtidos na camada anterior (δ_{pk}). Isto caracteriza a "propagação" do erro ao longo da rede, de onde vem o nome do algoritmo.

Substituindo as equações **(7)** e **(8)** em **(2)** e **(3)**, respectivamente, obtém-se o incremento a ser aplicado aos pesos da rede :

$$\Delta w_{jk} = \eta \sum_{p} \delta_{pk} h_{pj} \quad \textbf{(9)}$$

$$\Delta v_{ij} = \eta \sum_{p} \delta_{pj} x_{pi} \quad \textbf{(10)}$$

Em resumo, o algoritmo pode ser descrito como abaixo :

Faça

Para cada exemplo p

- Calcular os valores de saída da camada escondida (h_{pj}), a partir do vetor de entrada x_p;
- Calcular as respostas dos elementos da camada de saída (o_{pk});
- Calcular os sinais de erro da camada de saída (δ_{pk}), utilizando o vetor de saída desejado y_p;
- Calcular os sinais de erro da camada escondida (δ_{pj});
- Acumular o gradiente, de acordo com **(9)** e **(10)**;

Incrementar os pesos w_{jk} e v_{ij};

Calcular a função objetivo **(1)**;

até que seja atingido o critério de convergência ou o limite máximo de iterações.

III.3 Variantes do algoritmo

O algoritmo descrito na seção anterior consiste na busca do mínimo na superfície de erro. Esta superfície possui várias características que a tornam difícil de percorrer. Por exemplo, ela tem muitas regiões planas, onde o aprendizado é lento (HUSH, 1988). Como η é fixo, o tamanho real do passo é determinado pela magnitude do gradiente. Isto significa que o progresso nas regiões planas da superfície, onde o gradiente é pequeno, será relativamente lento. Desafortunadamente, a superfície de erro tende a ter um grande número de regiões planas.

Vários estudos mostraram que o algoritmo original possui problemas sérios :

i. é muito lento (FAHLMAN, 1988);
ii. quando aplicado a redes maiores o algoritmo não se comporta tão bem (FANTY, 1988);
iii. o algoritmo pode fornecer uma configuração de pesos que corresponde a um mínimo local da função de erro.

Isto ocorre porque o algoritmo utiliza o método do gradiente e, a princípio, a superfície de erro possui uma forma qualquer. Na prática, utilizando-se representações suficientemente distribuídas o algoritmo se comporta bem (McCLELLAND e RUMELHART, 1988). Assim, têm surgido diversas variantes do algoritmo, na busca da solução dos problemas acima.

III.3.1 "Batching"

Como se pode observar na definição do algoritmo, o valor do gradiente é obtido apenas depois que todos os exemplos são processados, o que corresponde ao valor expresso em **(9)** e **(10)**. No entanto, alguns pesquisadores (inclusive os formuladores originais do "PDP group") adotaram um procedimento alternativo, no qual a cada n exemplos (onde n é chamado de "batch size") o gradiente parcial é usado para incrementar os pesos, aumentando sua freqüência de atualização. Há inclusive uma forma desta heurística, usada com freqüência, na qual o "batch size" vale 1, ou seja, a cada exemplo os pesos são recalculados. De modo geral, este procedimento é levemente superior em desempenho que o algoritmo original.

III.3.2 Momento

Um dos problemas de um algoritmo que use o método do gradiente é escolher uma taxa de aprendizado apropriada. Alterar os pesos como uma função linear da derivada parcial de E (como definido em **(2)** e **(3)**) assume que a superfície de erro é localmente linear, onde "localmente" é definido pelo tamanho da taxa de aprendizado. Em pontos com curvatura alta, esta hipótese de linearidade não é válida e um comportamente divergente pode ocorrer perto destes pontos. Portanto, é importante manter a taxa de aprendizado pequena para evitar este comportamento.

Entretanto, uma taxa pequena pode ocasionar um aprendizado muito lento. O conceito de **momento** foi introduzido para resolver esta dicotomia. As equações **(9)** e **(10)** são modificadas de forma que uma parte da variação anterior é somada à variação atual :

$$\Delta w_{jk}(t) = \eta \sum_{p} \delta_{pk} h_{pj} + \alpha \Delta w_{jk}(t-1)$$

$$\Delta v_{ij}(t) = \eta \sum_{p} \delta_{pj} x_{pi} + \alpha \Delta v_{ij}(t-1)$$

Este termo age como um filtro, de forma que tendências gerais são reforçadas, enquanto que oscilações são canceladas. Isto permite uma taxa pequena, embora com aprendizado mais rápido.

III.3.3 Quickprop

Proposta por FAHLMAN (1988), baseia-se na aplicação do método de aproximação de Newton, a partir de duas hipóteses :

i. a curva erro x peso pode ser aproximada por uma parábola, e;
ii. o gradiente da curva acima não é afetado pelos outros pesos que estão sendo alterados ao mesmo tempo.

Para cada peso, independentemente, Fahlman usa os gradientes atual e anterior e a última variação do peso para determinar a parábola; ele passa então diretamente para o seu mínimo. Isso é feito através do processo iterativo :

$$\Delta w_{ij}(t) = \frac{\frac{\partial E}{\partial w_{ij}}(t)}{\frac{\partial E}{\partial w_{ij}}(t-1) - \frac{\partial E}{\partial w_{ij}}(t)} \Delta w_{ij}(t-1)$$

$$w_{ij}(t) = \begin{cases} w_{ij}(t-1) + \Delta w_{ij}(t) - \varepsilon \frac{\partial E}{\partial w_{ij}}(t) & \text{se } \frac{\partial E}{\partial w_{ij}}(t-1).\frac{\partial E}{\partial w_{ij}}(t) > 0 \\ w_{ij}(t-1) + \Delta w_{ij}(t) & \text{caso contrário} \end{cases}$$

O algoritmo Quickprop possui dois parâmetros : a taxa de aprendizado ε e o valor ν, chamado de "fator máximo de crescimento", tal que nenhum passo pode ser maior que ν vezes em magnitude o passo anterior. O valor default para ν é 1.75.

O algoritmo apresenta uma melhora considerável em comparação com o backpropagation padrão, fazendo com que o Quickprop seja uma das variantes mais usadas atualmente.

III.3.4 RPROP

O principal objetivo do algoritmo RPROP, proposto por RIEDMILLER e BRAUN (1993), é eliminar a influência indesejável do tamanho do gradiente no cálculo do passo. Como conseqüência, somente o sinal da derivada parcial é considerado para indicar a direção da atualização dos pesos. Trata-se de um algoritmo que se enquadra na família dos "métodos de descida por coordenadas" (veja, por exemplo, LUENBERGER (1984)). O tamanho do incremento a ser aplicado é determinado exclusivamente por um valor Δ_{ij} , específico para cada peso :

$$\Delta w_{ij}(t) = \begin{cases} -\Delta_{ij}(t) & se \ \frac{\partial E}{\partial w_{ij}}(t) > 0 \\ +\Delta_{ij}(t) & se \ \frac{\partial E}{\partial w_{ij}}(t) < 0 \\ 0 & \text{caso contrário} \end{cases}$$

O passo seguinte é determinar os novos valores $\Delta_{ij}(t)$, da seguinte forma :

$$\Delta_{ij}(t) = \begin{cases} \eta^+.\Delta_{ij}(t-1) & se \ \frac{\partial E}{\partial w_{ij}}(t-1).\frac{\partial E}{\partial w_{ij}}(t) > 0 \\ \eta^-.\Delta_{ij}(t-1) & se \ \frac{\partial E}{\partial w_{ij}}(t-1).\frac{\partial E}{\partial w_{ij}}(t) < 0 \\ \Delta_{ij}(t-1) & \text{caso contrário} \end{cases}$$

onde $0 < \eta^- < 1 < \eta^+$

O algoritmo possui dois parâmetros : Δ_0, que é o valor inicial assumido pelos incrementos Δ_{ij}; e Δ_{max}, valor máximo que pode ser atribuído a Δ_{ij}. Os fatores η^+ e η^- são fixados em 1.2 e 0.5, respectivamente.

Observa-se da fórmula de atualização do passo que :

i. se na direção ij E não oscilar ($\frac{\partial E}{\partial w_{ij}}(t-1).\frac{\partial E}{\partial w_{ij}}(t) > 0$), o modelo gradiente é considerado bom e se aumenta Δ_{ij};

ii. na situação oposta, considera-se que o modelo é ruim e se diminui Δ_{ij};

iii. enfim, próximo a direções críticas ($\frac{\partial E}{\partial w_{ij}} = 0$), Δ_{ij} é mantido.

Em resumo, o princípio básico do algoritmo é a adaptação direta dos incrementos Δ_{ij}. Devido à simplicidade de suas regras de aprendizado, há apenas um leve acréscimo no custo de cada iteração, comparado com o consumido pelo algoritmo backpropagation padrão.

III.3.5 Outras Variantes

Muitos pesquisadores têm proposto variantes baseadas em técnicas de busca mais sofisticadas. A maioria delas pode ser vista como diferentes maneiras de variar a taxa de aprendizado (η). Uma forma de variar η é realizar uma *busca linear*, ou seja, uma vez definida a direção (no caso, o gradiente), escolhe-se o valor de η que minimiza a função objetivo nesta direção. Existem várias formas de executar este busca. Uma abordagem é começar com η pequeno e gradualmente incrementá-lo até que o mínimo seja encontrado. Uma segunda abordagem, que usualmente é mais eficiente, é ajustar uma quadrática na direção de busca e executar um passo até o mínimo da quadrática. Outras tentativas de melhorar o aprendizado usaram informações de segunda ordem (da Hessiana). A maioria destas tentativas apresentaram melhoras significativas, usualmente em problemas pequenos. Freqüentemente, estas variantes enfrentam dificuldades para convergir, quando aumenta o tamanho do problema. Além disso, algumas destas abordagens tendem a ser menos robustas que o backpropagation (HUSH, 1991).

Capítulo IV

Métodos de Região de Confiança

IV.1 Introdução

Os Métodos chamados de *Região de Confiança* (TRUST-REGION METHODS) são uma opção menos usual na Programação Não Linear. Na maioria dos Algoritmos de Otimização se determina primeiramente uma *direção de descida*, ao longo da qual se faz uma busca unidimensional (também chamada de *busca linear*) com o objetivo de encontrar uma aproximação do mínimo nesta direção. A seguir, esta aproximação é considerada como o novo ponto de partida. Nesta categoria, aparecem os algoritmos do tipo Gradientes Conjugados e aqueles do tipo Newton e Quasi-Newton.

A idéia da iteração dada pelo Método de Região de Confiança é a seguinte : constrói-se no ponto x_c um modelo (em geral quadrático) da função objetivo e define-se, ao redor desse ponto x_c, uma bola de raio Δ_c, na qual "acredita-se" que este modelo seja uma aproximação adequada da função. Essa bola é chamada de Região de Confiança. O algoritmo deve garantir um decréscimo suficiente do modelo quadrático na intersecção desta Região de Confiança e do conjunto viável. A função objetivo é avaliada neste ponto. Se seu valor decresceu o suficiente, o novo ponto é aceito como o seguinte iterado, podendo-se eventualmente aumentar o raio da Região de Confiança. Caso contrário, o novo ponto é descartado e o raio da Região de Confiança é reduzido.

A diferença fundamental com os chamados Métodos de Busca Linear está na forma como eles utilizam o modelo quadrático para a eleição do comprimento do passo. Assim, enquanto na "busca linear" o modelo quadrático m_c é usado para obter a direção de busca através da minimização deste modelo, no Método de Região de Confiança escolhe-se

primeiro um passo de prova de comprimento Δ_c e a seguir se usa o modelo quadrático para selecionar a melhor direção com esse comprimento. Para isso se resolve o problema :

$$\mathbf{P}: \quad \text{Mín } m_c(x_c + s) = f(x_c) + g_c^t s + \tfrac{1}{2} s^t H_c s$$

$$s \in \Re^n$$

$$\text{s. a. } \| s \|_2 \leq \Delta_c \qquad \textbf{(1)}$$

onde g_c é o gradiente da função f avaliada no ponto x_c e H_c é a matriz Hessiana definida no ponto x_c.

A idéia aqui é que o passo de prova Δ_c seja considerado um estimador *de até onde* podemos "confiar" no modelo quadrático.

Assim, depois que s_c foi encontrado (sendo s_c a solução do problema de minimização **P**), o Método de Região de Confiança efetua uma avaliação de $f(x_c + s_c)$ para verificar se $x_c + s_c$ é satisfatório ou não.

P é em geral resolvido aproximadamente. As duas relaxações abaixo são as mais utilizadas :

i. Fazendo que a restrição **(1)** ativa só possa ser cumprida de forma aproximada; isso dá origem aos chamados Métodos de passo localmente restrito (*HOOK*), e

ii. O modelo quadrático é minimizado em uma aproximação poligonal da curva $s(\Delta_c)$, o que dá origem aos chamados Métodos "dogleg" e "double dogleg".

Os Métodos de Região de Confiança são atrativos, em particular por tratarem de modo "automático" de pontos em que a Hessiana de f seja indefinida (veja, por exemplo, DENNIS e SCHNABEL (1989) e MORÉ e SORENSEN (1984)).

IV.2 Descrição

No Método de Newton, com busca linear, a Hessiana é modificada quando não é suficientemente definida positiva. Esta modificação do modelo quadrático garante convergência, mas parece ignorar o papel do modelo quadrático como uma aproximação local da função objetivo. Vamos aqui considerar uma aproximação alternativa, na qual o modelo quadrático não é modificado diretamente, porém através de sua inserção em uma *região de confiança* restrita.

Seja f: $\Re^n \to \Re$ uma função duas vezes continuamente diferenciável. No Método de Newton, com uma estratégia de região de confiança, cada iterado x_k tem uma cota associada Δ_k, tal que:

$$f(x_k + w) \approx f(x_k) + \psi_k(w); \ \|w\| \leq \Delta_k$$

onde : $\psi_k(w) = \nabla f(x_k)^t w + \frac{1}{2} w^t \nabla^2 f(x_k) w$

é o modelo quadrático da possível redução em f, dentro de uma vizinhança do iterado x_k.

Isto sugere que seria desejável calcular um passo s_k, o qual resolve aproximadamente o problema :

$$\min \{\psi_k(w) : \|w\| \leq \Delta_k\} \qquad \textbf{(2)}$$

Se o passo é satisfatório, no sentido que $x_k + s_k$ produza uma redução suficiente em f, então Δ_k pode ser aumentado. Se, ao contrário, o passo não é satisfatório, então a cota Δ_k deve ser diminuída.

No caso mais simples em que se tem f: $\Re^n \to \Re$ duas vezes continuamente diferenciável e $H_k \in \Re^{nxn}$ simétrica e definida positiva, com $\|\ .\ \|$ a norma l_2, então o problema **(2)** é resolvido por :

$$s(\mu) = -(H_k + \mu I)^{-1} \nabla f(x_k) \qquad \textbf{(3)}$$

para um único $\mu > 0$, tal que $\|s(\mu)\| = \Delta_k$, a menos que $\|s(0)\| \leq \Delta_k$, em cujo caso $s(0) = s_k^N$ é a solução.

IV.3 Algoritmo Geral

Seja $f: \Re^n \to \Re$, $\Delta_k > 0$, $x_k \in \Re^n$, $H_k \in \Re^{nxn}$ simétrica.

Repita :

(1) s_k = solução aproximada do problema **(2)**

$x_+ = x_k + s_k$

(2) Decidir se x_+ é aceitável e calcular um novo valor de Δ_k.

Quando x_+ é aceitável :

$\Delta_+ = \Delta_k$

De forma mais explícita este Algoritmo se escreve assim :

PASSO 0 : Definem-se as constantes : $\mu \in (0, 1)$; $\eta \in (\mu, 1)$; $\gamma_0, \gamma_1, \gamma_2$, as quais satisfazem :

$0 < \gamma_0 \leq \gamma_1 < 1 \leq \gamma_2$

O ponto de partida x_0, o valor da função neste ponto $(f(x_0))$ e o gradiente (g_0) da função em x_0 são dados. Também são dados o raio inicial Δ_0 da região de confiança e B_0, a aproximação inicial da Hessiana no ponto de partida.

Seja $k = 0$.

PASSO 1 : Obtém-se o passo s_k (solução aproximada do problema **(2)**).

PASSO 2 : Calcula-se $f(x_k + s_k)$, e

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{f(x_k) - m_c(x_k + s_k)}$$

onde :

$$m_c(x_k + s_k) = f(x_k) + g_k^t s_k + \frac{1}{2} s_k^t B_k s_k$$

Notar que ρ_k pode ser escrito também como :

$$\rho_k = \frac{f(x_k + s_k) - f(x_k)}{\psi_k(s_k)}$$

PASSO 3 :

i) No caso em que $\rho_k > \mu$, ou seja, quando :

$$f(x_k) - f(x_k + s_k) > f(x_k) - m_c(x_k + s_k) \qquad \textbf{(4)}$$

fazer : $x_{k+1} = x_k + s_k$

$g_{k+1} = \nabla f(x_{k+1})$

e $\Delta_{k+1} \in [\Delta_k, \gamma_2 \Delta_k]$ se $\rho_k \geq \eta$

ou $\Delta_{k+1} \in [\gamma_1 \Delta_k, \Delta_k]$ se $\rho_k < \eta$

ii) No caso em que $\rho_k \leq \mu$, ou seja, quando :

$f(x_k) - f(x_k + s_k) \leq f(x_k) - m_c(x_k + s_k)$

fazer : $x_{k+1} = x_k$

$g_{k+1} = g_k$

e $\Delta_{k+1} \in [\gamma_0 \Delta_k, \gamma_1 \Delta_k]$

PASSO 4 : Atualizar a matriz B_k.

Incrementar k em 1 e voltar ao passo 1.

Observação : Diremos que a iteração é vitoriosa se **(4)** se cumpre, ou seja, quando a função redução :

$$f(x_k) - f(x_k + s_k)$$

é grande comparada com a anteriormente apresentada :

$$f(x_k) - m_c(x_k + s_k)$$

Não estamos interessados em resolver o problema **(2)** com grande precisão. Em vez disto, estamos interessados em dar condições mais relaxadas, para aceitar uma solução aproximada s_k, as quais garantam que a sucessão $\{x_k\}$ gerada pelo Algoritmo é convergente ao ponto x^*, com $\nabla f(x^*) = 0$ e $\nabla^2 f(x^*)$ semi-definida positiva.

Um requisito mínimo para s_k é que existam constantes $\beta_1 > 0$ e $\beta_2 > 0$ tais que :

$$-\psi_k(s_k) \geq \beta_1 |\psi_k^*|, \qquad \text{com } \|s_k\| \leq \beta_2 \Delta_k \qquad \textbf{(5)}$$

onde ψ_k^* é o valor ótimo do problema **(2)**.

O seguinte Teorema (MORÉ e SORENSEN (1984)) estabelece a convergência do Algoritmo :

TEOREMA:

Seja $f : \Re^n \to \Re$ duas vezes continuamente diferenciável, sobre um conjunto aberto D e suponhamos que o ponto de partida x_0 é tal que o conjunto :

$$\Omega = \{x \in D : f(x) \leq f(x_0)\}$$

é compacto. Se a sucessão $\{x_k\}$ é produzida pelo Algoritmo, onde s_k satisfaz **(5)**, então o Algoritmo termina em $x_l \in \Omega$ já que $\nabla f(x_l) = 0$ e $\nabla^2 f(x_l)$ é semi-definida positiva ou $\{x_k\}$ tem um ponto limite x^* em Ω com $\nabla f(x^*) = 0$ e $\nabla^2 f(x^*)$ semi-definida positiva.

IV.4 Estratégias para o Cálculo do Passo

IV.4.1 Passo Localmente Restrito ("Hook Step")

A primeira aproximação para calcular o passo no Algoritmo descrito no item IV.3, quando $\|s(0)\|_2 > \Delta_k$ (onde $s(0) = H_k^{-1}\nabla f(x_k) = s_k^N$), é através da resolução aproximada da equação escalar

$$\phi(\mu) = \|s(\mu)\|_2 - \Delta_k = 0 \qquad (6)$$

onde $s(\mu) = -(H_k + \mu I)^{-1}\nabla f(x_k)$

Para resolver **(6)**, HEBDEN e REINSCH observaram independentemente que a função $\|s(\mu)\|^2$ é uma função racional em μ, com pólos de segunda ordem nos valores $-\lambda_i$ (onde λ_i é autovalor da matriz H_k).

Então, para achar o valor μ_* tal que **(6)** tem um zero, o método de Newton para busca de zeros de uma função unidimensional não vai ser eficiente no caso em que a solução μ_* estiver perto de $-\lambda_1$ (onde λ_1 é o menor autovalor de H_k).

Eles sugerem que é mais eficiente aplicar o método de Newton à função

$$\nu(\mu) = \frac{1}{\Delta_k} - \frac{1}{\|s(\mu)\|_2}$$

De fato, ν é uma função que não tem pólos, e é "quase linear" perto de $-\lambda_1$.

Assim :

$$\mu_+ = \mu_c - \frac{\nu(\mu_c)}{\nu'(\mu_c)} = \mu_c - \frac{\left(-\frac{1}{\|s(\mu_c)\|_2} + \frac{1}{\Delta_k}\right)}{\|s(\mu_c)\|_2^{-2}\left(\|s(\mu_c)\|_2\right)'} =$$

$$= \mu_c - \frac{\frac{-\Delta_k + \|s(\mu_c)\|_2}{\Delta_k \cdot \|s(\mu_c)\|_2}}{\|s(\mu_c)\|_2^{-3}\left(-s(\mu_c)^t (H_k + \mu_c I)^{-1} s(\mu_c)\right)} =$$

$$= \mu_c - \frac{\|s(\mu_c)\|_2\left(\|s(\mu_c)\|_2 - \Delta_k\right)}{\Delta_k \frac{\left(-s(\mu_c)^t (H_k+\mu_c I)^{-1} s(\mu_c)\right)}{\|s(\mu_c)\|_2}}$$

Utilizando a expressão de $\phi'(\mu_c)$, chega-se que o valor de μ_+ pode ser escrito como :

$$\mu_+ = \mu_c - \frac{\|s(\mu_c)\|_2}{\Delta_k}\frac{\phi(\mu_c)}{\phi'(\mu_c)} \qquad (7)$$

Observação : A forma de **(7)** mostra que :

i) Quando $\mu_c < \mu_*$ (onde μ_* corresponde à solução exata da equação escalar **(6)**), estamos assumindo uma correção maior que aquela que o Método de Newton aplicado a ϕ poderia dar, ou seja, **(7)** é um acelerador em relação ao Método de Newton aplicado a **(6)**. A conclusão é análoga para $\mu_c > \mu_*$.

ii) À medida que μ_c tende a μ_*, o fator $\frac{\|s(\mu_c)\|_2}{\Delta_k}$ tende para 1, e as duas abordagens de Newton pouco diferenciam.

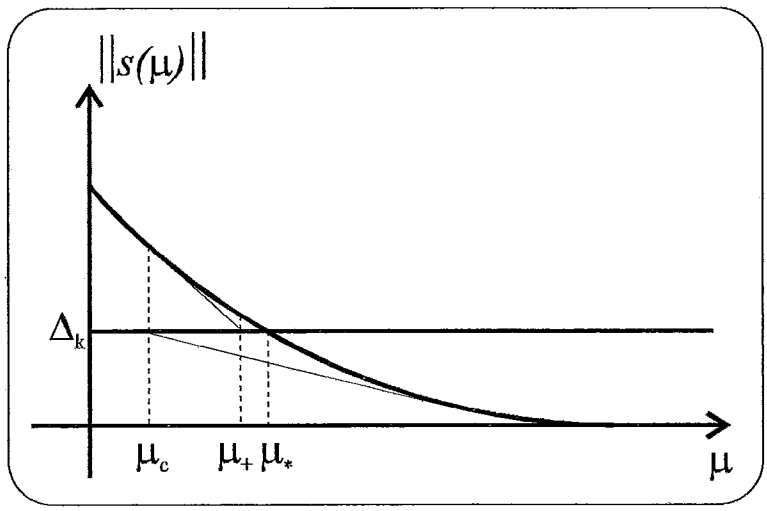


Figura IV.1 : Método de Newton aplicado a $\phi(\mu)$ e a $\nu(\mu)$

Daremos alguns resultados que transformam **(7)** em um algoritmo que pode ser implementado computacionalmente. Para isto, convém ter-se presente algumas considerações :

i) Com relação ao valor de partida para μ na resolução de **(6)** :

REINSCH (1971) demonstrou que se em **(7)** se começa com um valor inicial $\mu_0 = 0$, então a iteração converge monotonamente a μ_*. Entretanto, gostaríamos de ter um ponto inicial mais próximo, já que cada iteração de **(7)** resulta na resolução de um sistema linear. Para isto, MORÉ (1977) propôs o seguinte :

Se a cota do passo atual Δ_k é p vezes o último valor da cota Δ_-, então $\mu_0 = \frac{\mu_-}{p}$ é usado como ponto de partida em **(7)**.

Aqui se prefere uma estratégia diferente. Recordemos aonde estamos na iteração. Acabamos de achar um passo $s(\mu_-)$ de x_- para x_k e agora queremos x_+. Obtivemos Δ_k de Δ_-. Já temos H_k e também $(H_- + \mu_- I)$ de forma fatorada; calculamos :

$$\phi(\mu_-) = \|s(\mu_-)\| - \Delta_k$$

$$\phi'(\mu_-) = \frac{s(\mu_-)^t (H_- + \mu_- I)^{-1} s(\mu_-)}{\|s(\mu_-)\|}$$

Temos então, de (7), μ_0 dado por :

$$\mu_0 = \mu_- - \frac{\|s(\mu_-)\|}{\Delta_k} \frac{\phi(\mu_-)}{\phi'(\mu_-)} \qquad (8)$$

com custo computacional da ordem de n^2.

ii) Neste ponto construiremos um intervalo de confiança para μ, dado por cotas superior e inferior (u_+ e I_+).

Como a iteração de Newton aplicada a (6) sempre chega a μ_*, faz-se :

$$I_0 = -\frac{\phi(0)}{\phi'(0)}$$

Logo se calcula : $\mu_+^N = \mu_c - \frac{\phi(\mu_c)}{\phi'(\mu_c)}$, juntamente com cada cálculo de (7) e atualizamos a menor cota para $I_+ = \text{máx}\{I_c, \mu_+^N\}$, onde I_c é a menor cota atual. Também como:

$$\Delta_k = \left\|(H_k + \mu_+ I)^{-1} \nabla f(x_k)\right\|_2 < \frac{\|\nabla f(x_k)\|}{\mu_*}$$

(já que H_k é definida positiva e $\mu_* > 0$), tomamos $\frac{\|\nabla f(x_k)\|}{\Delta_k}$ como cota superior inicial u_0 para μ_*. Logo, em cada iteração, se $\phi(\mu_c) < 0$, atualizamos a cota superior para : $u_+ = \min\{u_c, \mu_c\}$ onde u_c é a cota superior atual.

Se, em qualquer iteração, μ_+ não está em $[I_+, u_+]$, seguimos MORÉ (1977, 1978) para escolher μ_+ como :

$$\mu_+ = \max\left\{(I_+ * u_+)^{\frac{1}{2}}, 10^{-3} u_+\right\} \qquad (9)$$

onde o segundo termo que aparece em (9) é uma precaução para os valores próximos de 0 e I_+. Na prática, estas cotas são usadas com bastante freqüência no cálculo de μ_0. Em particular, (9) é usada para definir μ_0 sempre que (8) não possa ser usada, porque a iteração prévia usou o passo de Newton.

iii) Finalmente não resolvemos (6) com grande exatidão, ao invés disto, fazemos $\|s(\mu)\|_2 \in \left\lfloor \frac{3}{4}\Delta_k, \frac{3}{2}\Delta_k \right\rfloor$. O motivo disto é que (como se verá mais adiante) a região de confiança nunca é aumentada ou diminuída por um fator menor que 2. Assim, se o raio da região de confiança em curso é Δ_k, um anterior teve que ser maior ou igual a $2\Delta_k$ ou menor ou igual a $\frac{\Delta_k}{2}$. Logo consideramos que o valor atual de Δ_k seja arbitrário dentro do intervalo $\left\lfloor \frac{3}{4}\Delta_k, \frac{3}{2}\Delta_k \right\rfloor$, no qual ajusta a diferença em uma ou outra direção e parece razoável aceitar que $\|s(\mu)\|_2$ tenha qualquer valor neste intervalo.

Veremos agora um exemplo onde se aplica o que foi dito anteriormente :

EXEMPLO 1 : Interessa-nos resolver o problema :

$$\min\ m_c(x_k+s) = f(x_k) + \nabla f(x_k)^t s + \tfrac{1}{2} s^t H_k s$$

$$\text{s. a. } \|s\|_2 \le \Delta_k$$

onde : $f(x_1, x_2) = x_1^4 + x_1^2 + x_2^2$

$$\Delta_k = \frac{1}{2}$$

$$\mu_- = 0$$

$$x_k = (1,1)^t$$

$$H_k = \nabla^2 f(x_k)$$

$$\nabla f(x_k) = \left(4x_1^3 + 2x_1, 2x_2\right)^t = (6,2)^t$$

$$\|\nabla f(x_k)\|_2 = \sqrt{40} \cong 6.325$$

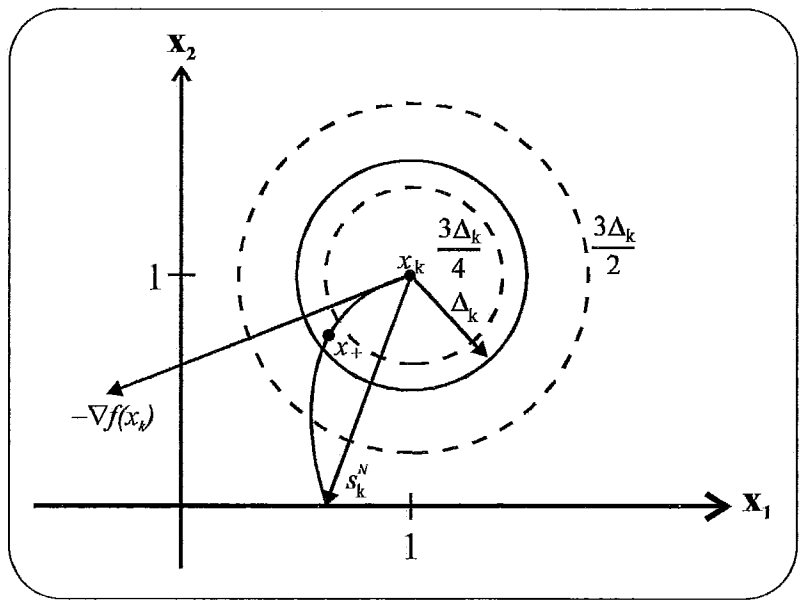


Figura IV.2 : Exemplo 1

$$H_k = \left[\begin{array}{cc} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{array} \right]_{x_c} = \left[\begin{array}{cc} 14 & 0 \\ 0 & 2 \end{array} \right]$$

$$s_k^N = -H_k^{-1} \nabla f(x_k) = -\left[\begin{array}{cc} \frac{1}{14} & 0 \\ 0 & \frac{1}{2} \end{array} \right] \left[\begin{array}{c} 6 \\ 2 \end{array} \right] = \left[\begin{array}{c} -\frac{3}{7} \\ -1 \end{array} \right]$$

$$\Rightarrow \left\|s_k^N\right\|_2 = \sqrt{\left(-\tfrac{3}{7}\right)^2 + (-1)^2} \cong 1.088$$

Como se deve ter que $\|s(\mu)\|_2 \in \left]\frac{3}{4}\Delta_k, \frac{3}{2}\Delta_k\right] = \left]\frac{3}{8}, \frac{3}{4}\right]$ e aqui ocorre que $\|s_k^N(\mu)\|_2 > \frac{3}{2}\Delta_k$ (o que significa que o passo de Newton é grande demais), então deve-se buscar a seguir algum $\mu > 0$ tal que :

$$s(\mu_i) = \left\|(H_k + \mu_i I)^{-1}\nabla f(x_k)\right\|_2 \in \left[\tfrac{3}{4}\Delta_k, \tfrac{3}{2}\Delta_k\right] = \left[\tfrac{3}{8}, \tfrac{3}{4}\right] = [0.375, 0.750]$$

com $i = 0, 1, 2, \ldots$

Para isto : como $\mu_- = 0$, então calculamos :

$$I_0 = -\frac{\phi(0)}{\phi'(0)} = \frac{-\|s(0)\|_2 + \Delta_k}{\frac{s(0)^t H_k^{-1} s(0)}{\|s(0)\|_2}} = \frac{-1.088 + 0.500}{\frac{-0.513}{1.088}} = \frac{-0.588}{-0.472} = 1.250$$

$$u_0 = \frac{\|\nabla f(x_k)\|_2}{\Delta_k} = \frac{6.325}{0.500} = 12.649$$

$$\mu_0 = \max\left\{(I_0 \times u_0)^{\frac{1}{2}}, 10^{-3}u_0\right\} = \max\left\{(1.250 \times 12.649)^{\frac{1}{2}}, 10^{-3} \times 12.649\right.$$

$$= \max\{3.97, 0.012\} = 3.97$$

Logo : $$s(\mu_0) = -(H_k + \mu_0 I)^{-1}\nabla f(x_k) = -\begin{bmatrix} 14+3.97 & 0 \\ 0 & 2+3.97 \end{bmatrix}^{-1}\begin{bmatrix} 6 \\ 2 \end{bmatrix} =$$

$$= -\left|\begin{array}{cc|c} 0.056 & 0 & 6 \\ 0 & 0.168 & 2 \end{array}\right| = (-0.334, -0.335)^t$$

$$\Rightarrow \|s(\mu_0)\|_2 = \sqrt{(-0.334)^2 + (-0.335)^2} \approx 0.473$$

Como $\|s(\mu_0)\|_2 \approx 0.473 \in [0.375, 0.750]$, fazemos então :

$\mu_c = \mu_0$ e

$$x_+ = x_k + s(\mu_c) = (1, 1)^t + (-0.334, -0.335)^t = (0.666, 0.665)^t$$

que corresponde à primeira iteração na obtenção de uma solução aproximada do problema :

$$\min\ m_c(x_k + s) = f(x_k) + \nabla f(x_k)^t s + \tfrac{1}{2}s^t H_k s$$

$$\text{s.a.: } \|s\|_2 \le \tfrac{1}{2}$$

Observamos que a solução exata é $\mu^* = 3.496$

IV.4.2 Passo "Dogleg" e "Double Dogleg"

Vejamos agora outra implementação do Modelo Região de Confiança, que foi proposto por POWELL (1970). Esta modificação também encontra uma solução aproximada do problema :

$$\min\ m_c(x_k + s) = f(x_k) + \nabla f(x_k)^t s + \tfrac{1}{2}s^t H_k s$$

$$\text{s.a.: } \|s\|_2 \le \Delta_k$$

Entretanto, em vez de encontrar um ponto $x_+ = x_k + s(\mu_c)$, sobre a curva $s(\mu)$, tal que $\|x_+ - x_k\| = \Delta_k$, aproxima esta curva por uma função seccionalmente linear, que conecta o chamado "ponto de Cauchy" (C.P.), que corresponde ao mínimo do modelo quadrático $m_c(x_k + s)$ na direção oposta à do gradiente $(-\nabla f(x_k))$, com o ponto de Newton para $m_c(x_k + s)$.

A modificação "duplo dogleg" (feita por DENNIS e MEI em 1979), muito usada na prática, leva o passo além da direção de Newton, (veja figura IV.3). A distância $\|\text{C.P.} - x_k\|_2$ é sempre menor que a distância $\|x^N - x_k\|_2$ (e algumas vezes, na prática, resulta ser bastante menor).

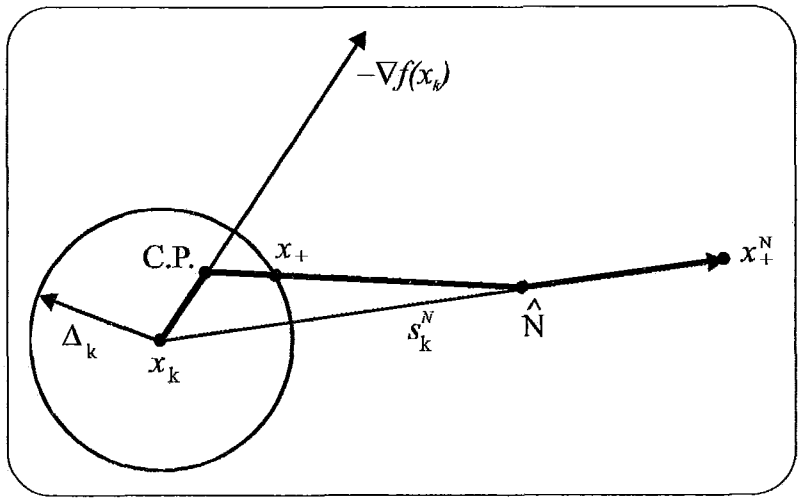


Figura IV.3 : "Duplo dogleg"

Aqui se escolhe x_+ como o ponto no arco poligonal ("duplo dogleg") tal que $\|x_+ - x_k\|_2 = \Delta_k$, salvo se $\|H_c^{-1}\nabla f(x_k)\|_2 < \Delta_k$, em cujo caso x_+ é o ponto de Newton.

A maneira específica de escolher a curva "duplo dogleg" faz com que ela tenha duas propriedades importantes :

i. à medida em que se avança ao longo da "curva linear" em pedaços de x_k a C.P., a $\hat{N}$ a x_+^N (melhor dizendo: de x_k até x_+^N, passando por C.P. e $\hat{N}$), a distância desde x_k aumenta monotonamente (mais adiante veremos que o ponto $\hat{N}$ corresponde a $\hat{N} = x_k - \eta H_k^{-1}\nabla f(x_k)$, para algum $0 < \gamma \leq \eta \leq 1$).
Assim, para qualquer $\Delta \leq \left\|H_k^{-1}\nabla f(x_k)\right\|$, existe um único ponto x_+ sobre a curva tal que $\|x_+ - x_k\| = \Delta$. Isto faz com que o processo esteja bem definido.

ii. o valor do modelo quadrático $m_c(x_k + s)$ decresce monotonamente quando s varia ao longo da curva de x_k até x_+^N, passando por C.P. e $\hat{N}$. Isto faz com que o processo seja bem definido.

O ponto C.P. resolve o problema :

$$\min\ m_c[x-\lambda\nabla f(x_k)] = f(x_k) - \lambda\|\nabla f(x_k)\|_2^2 + \tfrac{1}{2}\lambda^2\nabla f(x_k)^t H_k \nabla f(x_k)$$

$$\lambda \in \Re$$

Sua solução é obtida através de :

$$\nabla_\lambda m_c(x-\lambda\nabla f(x_k)) = 0 \Leftrightarrow -\|\nabla f(x_k)\|_2^2 + \lambda\nabla f(x_k)^t H_k \nabla f(x_k) = 0$$

$$\Rightarrow \lambda = \frac{\|\nabla f(x_k)\|_2^2}{\nabla f(x_k)^t H_k \nabla f(x_k)} \equiv \lambda_*$$

Logo, C.P. $= x_k - \lambda_* \nabla f(x_k)$, e se :

$$\Delta_k \le \lambda_* \|\nabla f(x_k)\|_2 = \frac{\|\nabla f(x_k)\|_2^3}{\nabla f(x_k)^t H_k \nabla f(x_k)},$$

o algoritmo assume um passo de tamanho Δ_k na direção oposta à do gradiente :

$$x_+ = x_k - \lambda_* \nabla f(x_k), \text{ onde } \lambda_* = \frac{\Delta_k}{\|\nabla f(x_k)\|_2}$$

O ponto $\hat{N}$ na curva "duplo dogleg" é escolhido agora e tem a forma : $\hat{N} = x_k - \eta H_k^{-1}\nabla f(x_k)$, para η tal que $\gamma \le \eta \le 1$, e $m_c(x)$ decresce monotonamente ao longo da linha que vai desde C.P. a N'.

Algumas observações :

i. POWELL originalmente escolheu $\eta = 1$, em cujo caso se tem a curva simples "dogleg";

ii. DENNIS e MEI (1979) sugerem escolher $\eta = 0.8\gamma + 0.2$, produzindo a curva "duplo dogleg", onde :

$$\gamma = \frac{\|\nabla f(x_k)\|_2^4}{(\nabla f(x_k)^t H_k \nabla f(x_k))(\nabla f(x_k)^t H_k^{-1} \nabla f(x_k))}$$

As escolhas de C.P. e $\hat{N}$ especificam completamente a curva "duplo dogleg". A escolha do ponto x_+ na curva tal que $\|x_+ - x_k\|_2 = \Delta_k$ é então um problema algébrico barato, como se observa no exemplo que se dará a seguir.

Note que o custo do algoritmo completo é somente da ordem de n^2 operações aritméticas, depois que s_k^N foi calculado.

EXEMPLO 2. Vejamos o que ocorre em uma iteração para o problema já apresentado :

$$\min\ m_c(x_k + s)$$

s.a.: $\|s\|_2 \le \Delta_k$

onde : $m_c(x_k + s) = f(x_k) + \nabla f(x_k)^t s + \frac{1}{2} s^t \nabla^2 f(x_k) s$, com os seguintes dados :

$$f(x_1, x_2) = x_1^4 + x_1^2 + x_2^2$$

$$x_k = (1, 1)^t$$

$$\nabla f(x_k) = (4x_1^3 + 2x_1, 2x_2)^t|_{x_c} = (6, 2)^t$$

$$\nabla^2 f(x_k) = H_k = \begin{bmatrix} 14 & 0 \\ 0 & 2 \end{bmatrix} \Rightarrow H_k^{-1} = \begin{vmatrix} \frac{1}{14} & 0 \\ 0 & \frac{1}{2} \end{vmatrix}$$

$$\Delta_k = 0.75$$

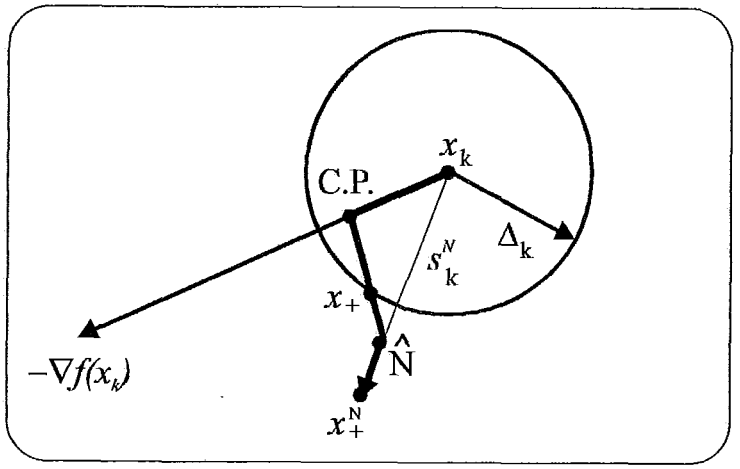


Figura IV.4 : Exemplo 2

a) Tem-se que :

$$s_k^N = -H_k^{-1} \nabla f(x_k) = \left(-\tfrac{3}{7}, -1\right)^t$$

$$\left\| s_k^N \right\|_2 = 1.088$$

b) Como ocorre que $\left\| s_k^N \right\|_2 > \Delta_k$, o algoritmo em conseqüência calcula o passo no Ponto de Cauchy (C.P.), ou seja, se calcula $s^{C.P.}$, recordando que C.P. é o mínimo do modelo quadrático $m_c(x_k + s)$, na direção oposta à do gradiente $(-\nabla f(x_k))$.

Para isto sabemos que :

$$s^{C.P.} = -\lambda_* \nabla f(x_k) = -\frac{\|\nabla f(x_k)\|_2^2}{\nabla f(x_k)^t H_k \nabla f(x_k)} \nabla f(x_k) =$$

$$= -\left(\frac{40}{512}\right) \begin{vmatrix} 6 \\ 2 \end{vmatrix} = \begin{vmatrix} -0.469 \\ -0.156 \end{vmatrix}$$

$$\Rightarrow \left\| s^{C.P.} \right\|_2 \approx 0.494$$

c) Como $\left\| s^{C.P.} \right\|_2 < \Delta_k$, calcula-se o passo ao ponto $\hat{N}$. Para isto :

$s^{\hat{N}} = \eta s_k^N$, onde $\eta = 0.8\gamma + 0.2$ (DENNIS e MEI), sendo que :

$$\gamma = \frac{\|\nabla f(x_k)\|_2^4}{(\nabla f(x_k)^t H_k \nabla f(x_k))(\nabla f(x_k)^t H_k^{-1} \nabla f(x_k))} = \frac{40^2}{(512)\left(\frac{92}{7}\right)} \approx 0.684$$

$$\Rightarrow \eta = 0.8 \times 0.684 + 0.2 = 0.747$$

$$s^{\hat{N}} = \eta s_k^N = \begin{bmatrix} -0.320 \\ -0.747 \end{bmatrix}$$

Recordemos que o ponto $\hat{N}$ é dado por $\hat{N} = x_k - \eta H_k^{-1} \nabla f(x_k)$ que, em nosso caso, corresponde a : $\hat{N} = (0.680, 0.253)^t$

d) Como $\left\|s^{\hat{N}}\right\|_2 = 0.813 > \Delta_k$, então o passo "duplo dogleg" deve estar na reta que une C.P. com $\hat{N}$, para o qual $\|s_k\|_2 = \Delta_k$. Isto significa dizer que :

$$s_k = s^{C.P.} + \lambda\left(s^{\hat{N}} - s^{C.P.}\right), \text{ com } 0 < \lambda < 1.$$

O valor de λ é encontrado resolvendo-se a equação :

$$\left\|s^{C.P.} + \lambda\left(s^{\hat{N}} - s^{C.P.}\right)\right\|_2 = \Delta_k$$

donde se obtém : $\lambda \approx 0.867$

Logo :

$$s_k = s^{C.P.} + \lambda\left(s^{\hat{N}} - s^{C.P.}\right) =$$

$$= \begin{bmatrix} -0.469 \\ -0.156 \end{bmatrix} + 0.867 \left[\begin{bmatrix} -0.320 \\ -0.747 \end{bmatrix} - \begin{bmatrix} -0.469 \\ -0.156 \end{bmatrix} \right] \approx \begin{bmatrix} -0.340 \\ -0.669 \end{bmatrix}$$

e) Finalmente :

$$x_+ = x_k + s_k \approx \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} -0.340 \\ -0.669 \end{bmatrix} = \begin{bmatrix} 0.660 \\ 0.331 \end{bmatrix}$$

Considerando que a solução exata do problema é $x_* = (0, 0)^t$, observa-se que o ponto obtido está efetivamente mais próximo da solução.

IV.4.3 Atualização da Região de Confiança

Agora estamos interessados em escolher se o ponto x_+ achado, usando qualquer dos métodos vistos anteriormente, é ou não um "ponto aceitável". A condição mais importante para que x_+ seja efetivamente um "ponto aceitável" é que se cumpra a desigualdade (condição de Armijo) :

$$f(x_+) \leq f(x_k) + \alpha g_k^t (x_+ - x_k) \qquad \textbf{(10)}$$

onde : $g_k = \nabla f(x_k)$ (ou uma aproximação); α é uma constante tal que $\alpha \in (0, \frac{1}{2})$ (de modo geral, escolhe-se $\alpha = 10^{-4}$)

1. Se x_+ não é um ponto aceitável, isto é, não satisfaz a desigualdade **(10)**, reduz-se a região de confiança por um fator que está entre $\frac{1}{10}$ e $\frac{1}{2}$ e volta-se a solução aproximada do problema de minimização, usando alguma das técnicas do passo vistas anteriormente (ou seja, pelo passo otimal localmente restrito do "HOOK", ou pelo método "dogleg" ou "duplo dogleg").

O fator de redução que se denotará por λ_*, se determina mediante a seguinte técnica : modela-se $f(x_k + \lambda(x_+ - x_k))$ usando para isto o modelo quadrático $m_q(\lambda)$, que ajusta $f(x_k)$, $f(x_+)$ e a derivada direcional de f em x_k, na direção $(x_+ - x_k)$, que denotaremos por $g_k^t(x_+ - x_k)$. A seguir fazemos que o novo raio de confiança Δ_k seja o mínimo deste modelo, o que ocorre quando :

$$\lambda_* = \frac{-g_k^t(x_+ - x_k)}{2[f(x_+) - f(x_k) - g_k^t(x_+ - x_k)]}$$

Então, o novo raio de confiança Δ_+ tem a forma :

$$\Delta_+ = \lambda_* \|x_+ - x_k\|_2$$

Observação : Se ocorre que este Δ_+ é tal que $\notin \left[\frac{\Delta_k}{10}, \frac{\Delta_k}{2}\right]$, então utiliza-se o limite deste intervalo mais próximo do valor encontrado; assim evita-se erros de precisão finita, especialmente quando o gradiente é calculado usando-se diferenças finitas.

2. Suponhamos agora que seja efetivamente encontrado um ponto x_+ que satisfaz (10).

Se x_+ é um passo de Newton desde x_k, então achamos o passo. Atualizamos Δ, construímos o novo modelo e continuamos com a iteração seguinte. Entretanto, se $(x_+ - x_k)$ não é o passo de Newton, tenta-se um passo maior a partir de x_k (dilatar-se o passo), usando o modelo geral. A justificativa para fazer isto é evitar ter que avaliar o gradiente (e a Hessiana) em x_+, a qual representa freqüentemente maior custo, em grandes problemas.

A razão para dilatar o passo é que a região de confiança pode chegar a se contrair durante o curso do algoritmo e pode-se necessitar que ela seja aumentada. Isto ocorre quando se tem uma região onde a cota do passo foi pequena, já que a função não estava bem representada por uma quadrática qualquer e entramos em uma região onde a função está bem representada por uma função quadrática.

Para se decidir se se faz ou não um passo maior desde x_k, compara-se a redução real definida por :

$$\Delta f = f(x_+) - f(x_k)$$

com a redução predita, definida por :

$$\Delta f_{pred} = m_c(x_+) - f(x_k)$$

e se aceita x_+ como o iterado, a menos que :

i. O ajuste seja tão bom, como por exemplo se
$|\Delta f_{pred} - \Delta f| \leq 0.1|\Delta f|$
e então Δ_k pode ser uma aproximação do raio em que m_c representa adequadamente f, ou

ii. Que a redução real (efetiva) de f seja tão grande como a presença de curvatura negativa e que portanto esteja implicado um decrescimento rapidamente contínuo em $f(x)$, ou seja :
$f(x_+) \leq f(x_k) + \nabla f(x_k)^t (x_+ - x_k)$

Em ambos os casos aproveitam-se x_+ e $f(x_+)$, mas em vez de mover-se diretamente para x_+, primeiro dobramos Δ_k e calcula-se um novo x_+, usando nosso modelo geral. Se **(10)** não é satisfeita para o novo x_+, volta-se então ao último "passo bom" já calculado. Mas se satisfaz, propõe-se dobrar novamente (Na prática, assim se pode evitar um número significativo de cálculos do gradiente).

EXEMPLO 3.

Seja $f(x_1, x_2) = x_1^4 + x_1^2 + x_2^2$

onde : $x_k = (1, 1)^t$

$\nabla f(x_k) = (6, 2)^t$

$$H_k = \nabla^2 f(x_k) = \begin{vmatrix} 14 & 0 \\ 0 & 2 \end{vmatrix}$$

$\Delta_k = 0.5$

e suponhamos, do EXEMPLO 1, visto anteriormente, que o passo s_k seja :

$s_k = (-0.334, -0.335)^t$

Logo :

$$x_+ = x_k + s_k = (1, 1)^t + (-0.334, -0.335)^t$$
$$= (0.666, 0.665)^t$$

Agora desejamos :

i. Decidir se x_+ é ou não um "ponto aceitável".

ii. Atualizar a região de confiança.

Assim :

a) Para que x_+ seja um ponto satisfatório, recordemos que se deve satisfazer a desigualdade :

$$f(x_+) \leq f(x_k) + \alpha \nabla f(x_k)^t (x_+ - x_k); \text{ com } \alpha = 10^{-4} \qquad \textbf{(11)}$$

Para isto : $f(x_+) = (0.666)^4 + (0.666)^2 + (0.665)^2 \approx 1.083$

$$f(x_k) = 1^4 + 1^2 + 1^2 = 3$$

$$f(x_k) + \alpha \nabla f(x_k)^t (x_+ - x_k) = 3 + 10^{-4} \begin{bmatrix} 6 & 2 \end{bmatrix} \begin{vmatrix} 0.666 - 1 \\ 0.665 - 1 \end{vmatrix} \approx 2.9997$$

Como $1.083 \leq 2.9997$ (ou seja, satisfaz a desigualdade **(11)**) então x_+ é efetivamente um ponto satisfatório.

b) Para atualizar a região de confiança devemos decidir se procuramos ou não um passo maior na iteração em curso. Para isto, vamos usar o teste da comparação :

$$\left| \frac{\Delta f_{pred}}{\Delta f} - 1 \right| \leq 0.1$$

Como $\Delta f = f(x_+) - f(x_k) = 1.083 - 3 = -1.917$

e $\Delta f_{pred} = m_c(x_+) - f(x_k) = \nabla f(x_k)^t s_k + \frac{1}{2} s_k^t H_k s_k$

$$\Delta f_{pred} = -2.673 + \frac{1}{2} \begin{bmatrix} -0.334 & -0.335 \end{bmatrix} \begin{bmatrix} 14 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} -0.334 \\ -0.335 \end{bmatrix} \approx -1.781$$

Então :

$$\left| \frac{\Delta f_{pred}}{\Delta f} - 1 \right| = \left| \frac{-1.781}{-1.917} - 1 \right| = 0.071 < 0.1$$

De tal modo que dobramos o raio de confiança, ou seja, fazemos $\Delta_k = 1$ e voltamos ao algoritmo de passo otimal localmente restrito ("HOOK") e temos que :

i) Nosso x_k é agora : $x_k = (0.666, 0.665)^t$ (que corresponde ao antigo x_+)

$$\nabla f(x_k) = \left(4x_1^3 + 2x_1, 2x_2\right)^t_{x_k} = (2.514, 1.33)^t$$

$$H_k = \nabla^2 f(x_k) = \begin{bmatrix} 12x_1^2 + 2 & 0 \\ 0 & 2 \end{bmatrix}_{x_k} = \begin{bmatrix} 7.323 & 0 \\ 0 & 2 \end{bmatrix}$$

$$H_k^{-1} = \begin{vmatrix} 0.137 & 0 \\ 0 & 0.5 \end{vmatrix}$$

ii) Logo $s_k^N = -H_k^{-1} \nabla f(x_k) = -\begin{bmatrix} 0.137 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} 2.514 \\ 1.330 \end{bmatrix} = \begin{bmatrix} -0.344 \\ -0.665 \end{bmatrix}$

$$\| s_k^N \|_2 = 0.749$$

Como $\| s_k^N \|_2 = 0.749 < 1$, o algoritmo "HOOK" (antes assinalado) seleciona o passo de Newton s_k^N.

Capítulo V

Aplicação do Método de Região de Confiança ao algoritmo Backpropagation

V.1 Introdução

Como foi dito no **Capítulo III**, o algoritmo **backpropagation** se baseia no Método do Gradiente, usando um passo fixo, chamado de "taxa de aprendizado". Um problema freqüentemente citado do algoritmo é seu elevado tempo de aprendizado, que pode ser explicado como uma conseqüência do uso do Método de Gradiente. Assim, com o objetivo de acelerar a taxa de convergência, diversas variantes foram propostas, usando métodos de segunda ordem (PARKER (1987), WATROUS (1987), BECKER e Le CUN (1988)). Neste último, são citados alguns problemas de convergência, e a conclusão de que poderia ser desejável usar uma técnica híbrida, combinando a aproximação de Newton com o Método do Gradiente. Como esta é precisamente uma das principais características do Método de Região de Confiança, decidimos investigar a sua aplicação ao algoritmo "backpropagation", na tentativa de obter melhores resultados.

V.2 Implementação

V.2.1 Conversão das matrizes *v* e *w*

Inicialmente, foi necessário adotar uma convenção que fizesse a correspondência entre as matrizes de pesos da rede neuronal (v e w) e um vetor (z), utilizado no Método de Região de Confiança.

Seja p o número de EPs da camada de entrada; q o número de EPs da camada escondida; e r o número de EPs da camada de saída. Definimos a seguinte correspondência entre as matrizes v (p x q) e w (q x r) e o vetor z :

$$z_1 = v_{11}$$
$$\vdots$$
$$z_{p\times q} = v_{pq}$$
$$z_{p\times q+1} = w_{11}$$
$$\vdots$$
$$z_{p\times q+q\times r} = w_{qr}$$

Portanto, o algoritmo original pode ser escrito da seguinte forma :

$$z_{t+1} = z_t - \eta \nabla E(z_t)$$

onde z_t representa o conjunto de pesos (v e w) da rede neuronal, no tempo t.

Uma vez adotada esta convenção, pode-se aplicar o Método de Região de Confiança, utilizando o vetor z.

V.2.2 Cálculo da Hessiana

Para aplicarmos o Método de Região de Confiança, foi necessário calcular a Hessiana da função E, que é definida como :

$$E = \frac{1}{2} \sum_p \sum_k (y_{pk} - o_{pk})^2$$

Sabe-se que o gradiente de E em relação a w_{jk} (veja **Cap. III**) é :

$$\frac{\partial E}{\partial w_{jk}} = -\sum_p \delta_{pk} h_{pj}$$

onde : $\delta_{pk} = s'(net_{pk})(y_{pk} - o_{pk}) = o_{pk}(1 - o_{pk})(y_{pk} - o_{pk})$

Calculando-se os elementos da Hessiana, tem-se :

- em relação a w_{jk} :

$$\frac{\partial E}{\partial w_{jk}} = -\sum_p \left(o_{pk} - o_{pk}^2\right)(y_{pk} - o_{pk})h_{pj}$$

$$= -\sum_p \left(o_{pk}^3 - (y_{pk}+1)o_{pk}^2 + y_{pk}o_{pk}\right)h_{pj}$$

$$\frac{\partial^2 E}{\partial w_{jk}\partial w_{j'k'}} = -\sum_p h_{pj}\left(3o_{pk}^2 - 2(y_{pk}+1)o_{pk} + y_{pk}\right)\frac{\partial o_{pk}}{\partial w_{j'k'}}$$

Sabe-se que :

$$\frac{\partial o_{pk}}{\partial w_{j'k'}} = \frac{\partial o_{pk}}{\partial net_{pk}}\frac{\partial net_{pk}}{\partial w_{j'k'}}$$

1º caso : $k \neq k'$

$$\Rightarrow \frac{\partial net_{pk}}{\partial w_{j'k'}} = 0 \Rightarrow \frac{\partial o_{pk}}{\partial w_{j'k'}} = 0 \Rightarrow \frac{\partial^2 E}{\partial w_{jk}\partial w_{j'k'}} = 0$$

2º caso : $k = k'$

$$\Rightarrow \frac{\partial net_{pk}}{\partial w_{j'k}} = h_{pj'} \Rightarrow \frac{\partial o_{pk}}{\partial w_{j'k}} = o_{pk}(1 - o_{pk})h_{pj'}$$

$$\Rightarrow \frac{\partial^2 E}{\partial w_{jk}\partial w_{j'k}} = -\sum_p h_{pj}h_{pj'}o_{pk}(1 - o_{pk})\left(3o_{pk}^2 - 2(y_{pk}+1)o_{pk} + y_{pk}\right)$$

Para simplificar a notação, vamos definir :

$$f_{pk} = 3o_{pk}^2 - 2(y_{pk}+1)o_{pk} + y_{pk}$$

Portanto :

$$\frac{\partial^2 E}{\partial w_{jk}\partial w_{j'k'}} = \begin{cases} 0 & \text{se } k \neq k' \\ -\Sigma_p f_{pk}h_{pj}h_{pj'}o_{pk}(1 - o_{pk}) & \text{se } k = k' \end{cases}$$

- em relação a v_{ij} :

$$\frac{\partial^2 E}{\partial w_{jk}\partial v_{ij'}} = -\sum_p \left[h_{pj} \underbrace{\left(3o_{pk}^2 - 2(y_{pk}+1)o_{pk} + y_{pk}\right)}_{f_{pk}} \frac{\partial o_{pk}}{\partial v_{ij'}} + \right.$$

$$\left. \underbrace{\left(o_{pk}^3 - (y_{pk}+1)o_{pk}^2 + y_{pk}o_{pk}\right)}_{\delta_{pk}} \frac{\partial h_{pj}}{\partial v_{ij'}} \right]$$

Sabe-se que (veja **Cap. III**) :

$$\frac{\partial o_{pk}}{\partial v_{ij'}} = \frac{\partial o_{pk}}{\partial net_{pk}} \frac{\partial net_{pk}}{\partial h_{pj'}} \frac{\partial h_{pj'}}{\partial net_{pj'}} \frac{\partial net_{pj'}}{\partial v_{ij'}}$$

onde :

$$\frac{\partial o_{pk}}{\partial net_{pk}} = s'(net_{pk}) = o_{pk}(1 - o_{pk})$$

$$\frac{\partial net_{pk}}{\partial h_{pj'}} = w_{j'k}$$

$$\frac{\partial h_{pj'}}{\partial net_{pj'}} = s'(net_{pj'}) = h_{pj'}(1 - h_{pj'})$$

$$\frac{\partial net_{pj'}}{\partial v_{ij'}} = x_{pi}$$

Portanto :

$$\frac{\partial o_{pk}}{\partial v_{ij'}} = o_{pk}(1 - o_{pk}) h_{pj'}(1 - h_{pj'}) w_{j'k} x_{pi}$$

Resta calcular a expressão de $\frac{\partial h_{pj}}{\partial v_{ij'}}$. Assim :

$$\frac{\partial h_{pj}}{\partial v_{ij'}} = \frac{\partial h_{pj}}{\partial net_{pj'}} \frac{\partial net_{pj'}}{\partial v_{ij'}}$$

1º caso : $j \neq j'$

Como $h_{pj} = s(net_{pj})$ (veja **Cap. III**), conclui-se que :

$$\Rightarrow \frac{\partial h_{pj}}{\partial net_{pj'}} = 0 \Rightarrow \frac{\partial h_{pj}}{\partial v_{ij'}} = 0$$

2º caso : $j = j'$

$$\Rightarrow \frac{\partial h_{pj}}{\partial v_{ij}} = h_{pj}(1 - h_{pj}) x_{pi}$$

Logo, a expressão de $\frac{\partial^2 E}{\partial w_{jk} \partial v_{ij'}}$ é :

$$\frac{\partial^2 E}{\partial w_{jk} \partial v_{ij'}} = \begin{cases} -\Sigma_p h_{pj} f_{pk} o_{pk}(1 - o_{pk}) h_{pj'}(1 - h_{pj'}) w_{j'k} x_{pi} & \text{se } j \neq j' \\ -\Sigma_p h_{pj} f_{pk} o_{pk}(1 - o_{pk}) h_{pj'}(1 - h_{pj'}) w_{j'k} x_{pi} + \\ \qquad \delta_{pk} h_{pj}(1 - h_{pj}) x_{pi} & \text{se } j = j' \end{cases}$$

Calculemos, finalmente, a expressão de $\frac{\partial^2 E}{\partial v_{ij} \partial v_{i'j'}}$. Sabe-se que :

$$\frac{\partial E}{\partial v_{ij}} = -\sum_p \delta_{pj} x_{pi}$$

onde $\delta_{pj} = h_{pj}(1-h_{pj})\sum_k \delta_{pk} w_{jk}$. Portanto :

$$\frac{\partial^2 E}{\partial v_{ij}\partial v_{i'j'}} = -\sum_p \left\{ \underbrace{\frac{\partial}{\partial v_{i'j'}}[x_{pi}h_{pj}(1-h_{pj})]}_{(1)} \sum_k \delta_{pk} w_{jk} + x_{pi}h_{pj}(1-h_{pj}) \underbrace{\frac{\partial}{\partial v_{i'j'}}\left[\sum_k \delta_{pk} w_{jk}\right]}_{(2)} \right\}$$

Desenvolvendo a expressão **(1)**, tem-se :

$$= x_{pi}(1-2h_{pj})\frac{\partial h_{pj}}{\partial v_{i'j'}} = \begin{cases} 0 & \text{se } j \neq j' \\ x_{pi}x_{pi'}(1-2h_{pj})h_{pj}(1-h_{pj}) & \text{se } j = j' \end{cases}$$

Quanto à expressão **(2)**, chega-se a :

$$= \sum_k w_{jk} \underbrace{\left(3o_{pk}^2 - 2(y_{pk}+1)o_{pk} + y_{pk}\right)}_{f_{pk}} \frac{\partial o_{pk}}{\partial v_{i'j'}} =$$

$$= \sum_k w_{jk} w_{j'k} f_{pk} o_{pk}(1-o_{pk}) h_{pj'}(1-h_{pj'}) x_{pi'}$$

Portanto, a expressão de $\frac{\partial^2 E}{\partial v_{ij}\partial v_{i'j'}}$ é :

$$\frac{\partial^2 E}{\partial v_{ij}\partial v_{i'j'}} = \begin{cases} -\Sigma_p x_{pi}x_{pi'}h_{pj}(1-h_{pj})h_{pj'}(1-h_{pj'})\Sigma_k w_{jk}w_{j'k}f_{pk}o_{pk}(1-o_{pk}) & \text{se } j \neq j' \\ -\Sigma_p x_{pi}x_{pi'}h_{pj}(1-h_{pj})\Sigma_k w_{jk}[\delta_{pk}(1-2h_{pj}) + \\ \quad f_{pk}o_{pk}(1-o_{pk})h_{pj'}(1-h_{pj'})w_{j'k}] & \text{se } j = j' \end{cases}$$

V.2.3 Codificação do algoritmo

Segundo BATTITI (1992), de modo geral a Hessiana descrita na seção anterior não é definida positiva. Assim, foi necessário utilizar uma implementação do Método de Região de Confiança que tratasse a possibilidade da Hessiana ser indefinida. GAY (1983) descreve uma subrotina (DHUMSL, em FORTRAN), com esta característica. Através da Internet (**http://gams.nist.gov**), foi obtida uma cópia desta subrotina, no GAMS (*Guide to Available Mathematical Software*), mantido pelo NIST (National Institute of Standards and Technology).

Assim, o algoritmo foi implementado em FORTRAN, da seguinte forma :

- BPTRUST - programa principal;
- DHUMSL - implementação do Método de Região de Confiança;
- FUNC - cálculo da função objetivo *E*;
- DFUNC - gradiente da função objetivo *E*;
- HESS - Hessiana (conforme seção **V.2.2)**;

V.3 Resultados

V.3.1 Metodologia

Para avaliar o desempenho da implementação descrita em **V.2** (BPTRUST), foram codificados os seguintes algoritmos : backpropagation (BPSTD) e RPROP (RIEDMILLER e BRAUN, 1993). Este último foi escolhido por ser a mais recente variação do algoritmo backpropagation encontrada na literatura. Em seguida, cada um dos algoritmos foi aplicado a cinco problemas : "ou exclusivo", codificador 10-5-10 e 20-10-20, classificação de eletrofácies e previsão de preços de derivados de petróleo, descritos nas seções **V.3.2**, **V.3.3**, **V.3.4** e **V.3.5**, respectivamente.

A metodologia adotada foi proposta por FAHLMAN (1988), e consiste em executar cada um dos algoritmos a partir do mesmo conjunto de pontos iniciais. Este conjunto possui dez pontos aleatórios e distintos. O tempo de aprendizado é apresentado como o número médio de épocas[1] necessário até que a tarefa seja aprendida. Caso não haja convergência, o tempo correspondente é definido como o número máximo de épocas, cujo valor depende de cada teste.

Para comparar o custo computacional de cada algoritmo, foi incluído o tempo médio de CPU gasto em cada teste (PRECHELT, 1994). Todos os testes foram executados em um IBM 9121 modelo 400.

Seguindo as sugestões de Fahlman, foi aplicado o critério "40-20-40" : um valor de saída é considerado como zero até o valor máximo de 0.4; caso esteja acima de 0.6, é considerado como um; e indeterminado, caso esteja no intervalo intermediário (0.4, 0.6). O aprendizado é considerado completo quando todo o conjunto de treinamento é classificado corretamente.

Ao final de cada teste, é apresentada uma tabela com as informações abaixo :

- N. Épocas - número de épocas necessário para completar o aprendizado;
- N. Funções - total de cálculos da função objetivo;
- N. Grad. - total de cálculos do gradiente;

[1] Uma época é definida como o período durante o qual todos os padrões do conjunto de treinamento são apresentados uma vez.

- N. Hess. - total de cálculos da Hessiana;
- CPU - tempo de CPU gasto no aprendizado;
- Sucesso - indica quantas execuções, a partir do conjunto de pontos iniciais, completaram o aprendizado (convergiram).

V.3.2 Ou exclusivo (XOR)

Um dos mais conhecidos problemas de teste é o do "**ou exclusivo**" (**XOR**), ou, em sua forma mais geral, o problema de paridade N. De acordo com Fahlman, este não deveria ser um teste típico para problemas do mundo real resolvidos por redes neuronais. A habilidade de generalização de uma rede, que é associar padrões similares de entrada a ativações similares de saída, não se aplica ao "ou exclusivo". A simples mudança de um bit no vetor de entrada causa uma classificação complementar. A razão pela qual se inclui problemas de paridade N neste estudo é o fato de que eles são freqüentemente usados na literatura para testar novos algoritmos de aprendizado.

A rede consiste em dois EPs de entrada, dois na camada escondida e um EP de saída, possuindo 9 pesos, que é a dimensão do problema (vetor gradiente). O conjunto de treinamento é :

ENTRADA	SAÍDA
0 0	0
1 0	1
0 1	1
1 1	0

O tempo máximo de aprendizado foi estabelecido como 1000 épocas.

A tabela abaixo mostra os resultados obtidos :

Algoritmo	N. Épocas	N. Funções	N. Grad.	N. Hess.	CPU (MM:SS)	Sucesso
BPSTD	531	531	531	-	00:27	09/10
RPROP	520	520	520	-	00:27	05/10
BPTRUST	48	115	48	48	00:06	10/10

Neste teste, nota-se que o algoritmo BPTRUST é mais robusto que os demais, convergindo em todo o conjunto de pontos iniciais. É interessante observar que, caso se considere apenas as execuções bem sucedidas, o algoritmo RPROP atinge uma média de apenas 40 épocas, com um consumo médio de CPU de 2 segundos. Portanto, seu desempenho é bastante prejudicado por sua sensibilidade ao ponto de partida.

V.3.3 Codificadores 10-5-10 e 20-10-20

Segundo Fahlman, uma classe mais adequada de problemas de teste é a família de problemas de codificação/decodificação do tipo N-M-N. A rede consiste de N EPs em cada uma das camadas de entrada e saída, e M EPs na camada escondida. O vetor de entrada inclui N bits, dos quais um vale '1' e os demais valem '0'. O vetor desejado de saída é idêntico ao de entrada; assim, o objetivo da rede é fazer uma "auto-associação" entre os vetores de entrada e de saída, aprendendo um mapeamento entre os N EPs de entrada e os M EPs da camada escondida (codificação) e um mapeamento entre estes e os N EPs da camada de saída (decodificação), onde, em geral, $M < N$.

O codificador 10-5-10 é um problema típico da família descrita acima. A rede possui 10 EPs de entrada, 5 na camada escondida e 10 na de saída, totalizando 115 pesos, que é a dimensão do problema (vetor gradiente). O conjunto de treinamento é :

ENTRADA	SAÍDA
1 0 0 0 0 0 0 0 0 0	1 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0	0 1 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0	0 0 1 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0	0 0 0 1 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0	0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0	0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 1 0 0 0	0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1 0 0	0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 1 0	0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 0 0 1

O tempo máximo de aprendizado foi definido como 500 épocas.

Os resultados encontrados foram os seguintes :

Algoritmo	N. Épocas	N. Funções	N. Grad.	N. Hess.	CPU (MM:SS)	Sucesso
BPSTD	180	180	180	-	00:13	10/10
RPROP	116	116	116	-	00:10	08/10
BPTRUST	50	100	50	50	02:32	10/10

Apesar de convergir em um número menor de épocas, o algoritmo BPTRUST é afetado pela dimensão do problema ($N = 115$), atingindo um tempo de CPU superior aos demais algoritmos. No entanto, mostrou-se mais robusto que o algoritmo RPROP. Em RIEDMILLER e BRAUN (1993), foi obtida uma média de 19 épocas. Convém ressaltar que foi utilizado um conjunto distinto de pontos iniciais, que não foi informado.

Com o objetivo de avaliar o algoritmo em um problema com dimensão maior, foi testado o codificador 20-10-20, que é semelhante ao problema anterior, tendo, no entanto, 20 elementos de entrada, 10 na camada escondida e 20 de saída (com 430 pesos), encontrando-se os resultados abaixo :

Algoritmo	N. Épocas	N. Funções	N. Grad.	N. Hess.	CPU (H:MM:SS)	Sucesso
BPSTD	15.014	15.014	15.014	-	52:22	10/10
RPROP	16	16	16	-	00:03	10/10
BPTRUST	154	339	154	154	6:32:12	10/10

Mais uma vez, observa-se a forte influência que a dimensão tem no algoritmo BPTRUST, levando a um significativo tempo de aprendizado. Pode-se notar também um dos problemas do algoritmo backpropagation padrão, que é o elevado número de épocas necessário para atingir a convergência.

V.3.4 Classificação de Eletrofácies

Com o intuito de observar o desempenho do algoritmo em estudo em uma tarefa real, escolheu-se o problema apresentado em QUEIROZ NETO e RODRIGUES (1991). O objetivo da rede proposta é indicar, a partir de perfis elétricos, qual a rocha predominante (eletrofácies) em cada profundidade. Os perfis elétricos (fig. V.1) são registros em profundidade das propriedades físicas das diferentes litologias presentes em um poço de petróleo. Estas propriedades podem ser a resistividade, densidade, porosidade, etc.

A identificação de eletrofácies a partir de perfis elétricos é um problema muito freqüente na indústria de petróleo. O geólogo responsável por este trabalho analisa uma série de perfis e dados de testemunhos[2], para determinar as litologias presentes nos poços estudados. É de fundamental importância para a delimitação das zonas produtoras de óleo a correta determinação das eletrofácies.

A rede possui a arquitetura abaixo (totalizando 95 pesos) :

- três EPs de entrada, correspondentes aos três perfis elétricos utilizados (raios gama, densidade e porosidade neutrônica);

[2] Amostras de rocha extraídas de poços de petróleo.

- dez EPs na camada escondida;
- cinco EPs de saída, correspondentes aos cinco tipos de rocha selecionados (anidrita, arenito, calcarenito, folhelho e siltito).

Durante o aprendizado, são apresentados à rede os valores dos perfis elétricos e a rocha correspondente para cada profundidade.

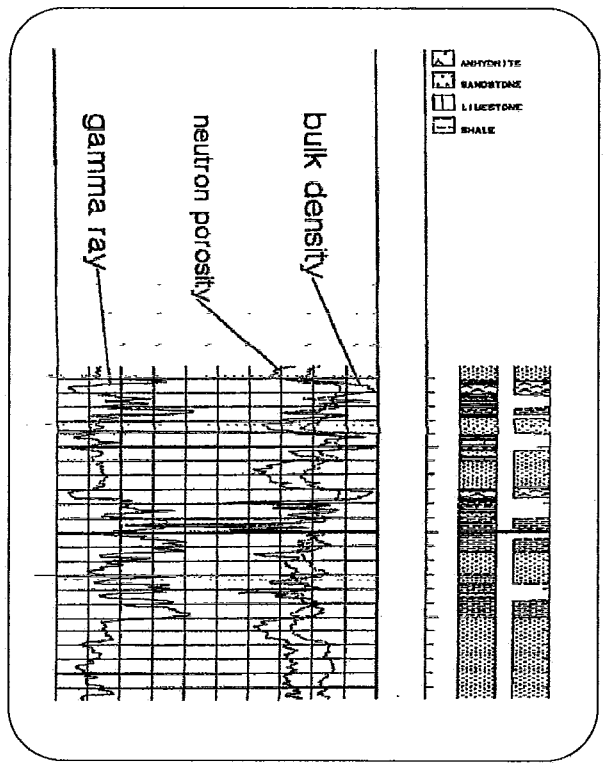


Figura V.1 : Exemplo de perfis elétricos

O conjunto de treinamento inclui 2022 exemplos.

Neste teste, foram adotados os mesmos critérios empregados em QUEIROZ NETO e RODRIGUES : o EP de saída com maior ativação corresponde ao tipo de rocha selecionado pela rede; o aprendizado é considerado completo quando mais de 80% do conjunto de treinamento é classificado corretamente.

Os resultados obtidos estão sumarizados abaixo :

Algoritmo	N. Épocas	N. Funções	N. Grad.	N. Hess.	CPU (H:MM:SS)	Sucesso
BPSTD	479	479	479	-	16:31	10/10
RPROP	34	34	34	-	01:14	10/10
BPTRUST	67	154	67	67	3:05:00	10/10

Apesar de convergir em um número menor de épocas que o algoritmo BPSTD, o algoritmo BPTRUST apresentou um elevado consumo de CPU. Chama a atenção o impressionante desempenho do algoritmo RPROP.

V.3.5 Previsão de Preços de Derivados de Petróleo

Proposto (no CERN) por Marcelo Pereira Melo (sob orientação do Prof. Ruy Milidiú) (MELO, 1991), o projeto acima estuda a aplicação de redes neuronais na previsão de preços de derivados de petróleo no mercado internacional, tendo sido observado um desempenho superior ao de especialistas de mercado na realização de previsões de curto prazo.

O principal objetivo destas previsões é balizar as decisões gerenciais de curto prazo e auxiliar a participação da Petrobrás no mercado futuro de derivados. Como a Petrobrás é grande compradora de petróleo, que vai ser transformado em derivados que podem ser exportados, as oportunidades comerciais ligadas às variações de preços de petróleo e derivados têm que ser aproveitadas ao máximo. Desta forma, previsões de preço são fundamentais para o planejamento operacional da empresa, sendo inclusive entradas para modelos de programação linear que auxiliam esse planejamento.

O mercado de Nova York é o principal mercado de exportação brasileira de derivados, e o mercado Noroeste da Europa serve de referência para as nossas exportações para a África Ocidental. Assim, são necessárias previsões de curto prazo para os seguintes produtos :

Nova York (FOB) : Série

- óleo combustível BTE 0.3 % HP (US$/barril) OC03NY
- óleo combustível BTE 1.0 % LP (US$/barril) OC10NY

(BTE = Baixo Teor de Enxofre)

Noroeste da Europa (CIF) :

- diesel (US$/tonel) DIESNW
- querosene de aviação (US$/tonel) QAVNW
- óleo combustível 3.5 % (US$/tonel) OC35NW

Convém ressaltar que as séries de preços em questão constituem seqüências de preços médios semanais. Assim, uma previsão um passo à frente corresponde nesse caso à previsão para uma semana à frente.

A solução proposta consiste em uma rede 7-3-1 (28 pesos), onde a entrada é formada pelos preços médios das sete semanas anteriores, e o valor desejado é o preço médio da semana atual.

O período escolhido para o ajuste de curvas foi de 04/01/85 a 25/12/87, portanto com duração de três anos (156 semanas).

Define-se *arv* (*average relative variance*) como :

$$arv = \frac{E}{n\sigma^2}$$

onde E é o valor da função objetivo;

n é o número de exemplos; e

σ^2 é a variância do conjunto de treinamento.

Em MELO (1991), considerou-se que o valor mínimo aceitável para *arv* é 0.04. Portanto, esta condição foi usada como critério de parada.

Os resultados obtidos para cada uma das cinco séries foi :

OC03NY	Algoritmo	N. Épocas	N. Funções	N. Grad.	N. Hess.	CPU (MM:SS)	Sucesso
	BPSTD	1.850	1.850	1.850	-	01:34	10/10
	RPROP	218	218	218	-	00:06	10/10
	BPTRUST	170	462	170	170	02:40	10/10

OC10NY	Algoritmo	N. Épocas	N. Funções	N. Grad.	N. Hess.	CPU (MM:SS)	Sucesso
	BPSTD	1.388	1.388	1.388	-	01:13	10/10
	RPROP	89	89	89	-	00:03	10/10
	BPTRUST	92	238	92	92	01:25	10/10

DIESNW	Algoritmo	N. Épocas	N. Funções	N. Grad.	N. Hess.	CPU (MM:SS)	Sucesso
	BPSTD	1.837	1.837	1.837	-	01:34	10/10
	RPROP	151	151	151	-	00:05	10/10
	BPTRUST	156	425	156	156	02:27	10/10

QAVNW	Algoritmo	N. Épocas	N. Funções	N. Grad.	N. Hess.	CPU (MM:SS)	Sucesso
	BPSTD	672	672	672	-	00:35	10/10
	RPROP	77	77	77	-	00:02	10/10
	BPTRUST	72	182	72	72	01:07	10/10

OC35NW	Algoritmo	N. Épocas	N. Funções	N. Grad.	N. Hess.	CPU (MM:SS)	Sucesso
	BPSTD	4.660	4.660	4.660	-	04:00	10/10
	RPROP	647	647	647	-	00:19	10/10
	BPTRUST	832	2.389	832	832	13:00	10/10

Observa-se que o número de iterações obtido pelo algoritmo BPTRUST é inferior (OC03NY e QAVNW) ou bastante próximo (OC10NY e DIESNW) do apresentado pelo algoritmo RPROP. No entanto, o custo computacional é muito mais elevado.

Capítulo VI

Conclusão

O algoritmo **backpropagation** se baseia em um método de otimização de primeira ordem (método do gradiente), onde é efetuado um passo de tamanho fixo (taxa de aprendizado), na direção oposta à do gradiente. Assim, algumas variantes do algoritmo utilizam métodos de segunda ordem, na tentativa de acelerar a convergência. Em uma dessas variantes, proposta por BECKER e Le CUN, devido a alguns problemas encontrados, sugere-se que a aplicação de um método que conciliasse o método de Newton com o método do gradiente poderia trazer bons resultados. Como esta é uma das características do Método de Região de Confiança, decidiu-se estudar a aplicação deste método ao algoritmo backpropagation.

Para avaliar o desempenho da variação proposta, foram codificados os seguintes algoritmos : backpropagation e RPROP, sendo este último a mais recente variação encontrada na literatura. Cada algoritmo foi testado em cinco tarefas : "ou exclusivo" (**XOR**), codificadores 10-5-10 e 20-10-20, classificação de eletrofácies e previsão de preços de derivados de petróleo.

Com a aplicação do Método de Região de Confiança ao algoritmo backpropagation, produziu-se uma variação com convergência superior em relação aos demais algoritmos testados, baseados apenas em métodos de primeira ordem. Este comportamento era esperado, uma vez que estes algoritmos não são capazes, por exemplo, de "escapar" de "falsos" mínimos locais, como pontos-sela. No entanto, esta robustez é obtida ao custo de iterações mais "pesadas" do ponto de vista computacional, nem sempre compensadas pelo menor número de iterações.

Apesar de apresentar alguns problemas de convergência, o algoritmo RPROP mostrou um excelente desempenho, principalmente nos problemas mais complexos (classificação de eletrofácies e previsão de preços de derivados de petróleo). Por usar apenas a direção do gradiente, ele se torna quase insensível às regiões da função objetivo que possuem gradientes com norma próxima de zero. A avaliação teórica que apresentamos junto ao método explica em parte aquele resultado.

Como descrito no **Cap. V**, foi utilizada uma subrotina (**DHUMSL**) que implementa o Método de Região de Confiança. No entanto, esta rotina é genérica, sendo aplicável a qualquer função duas vezes continuamente diferenciável. Considerando que a função objetivo E é uma soma de quadrados, poderia ser estudada a aplicação de um método de resolução específico para este tipo de problema de minimização, que é o *Método de Levenberg-Marquardt*, o que poderá diminuir o custo de cada iteração. Ainda com este objetivo, a modelagem quadrática por Quase-Newton merece ser comparada com a da Hessiana.

Conclui-se, portanto, que a interação entre Otimização e Redes Neuronais é frutífera e incentiva esforços de pesquisa adicionais. Em particular, a aplicação de técnicas de segunda ordem a problemas de grande escala (com centenas ou milhares de pesos e exemplos) é um tema que provavelmente levará a estudos muito produtivos.

Bibliografia

ABELÉM, A. J. G. (1994), "Redes Neurais Artificiais na Previsão de Séries Temporais", Dissertação de Mestrado, Departamento de Engenharia Elétrica, PUC/RJ.

AMARI, S. (1967), "A theory of adaptive pattern classifiers", *IEEE Trans. Eletronic Computers*, **EC-16(3)**, 299-307.

ANDERSON, J. A. (1968), "A memory storage model utilizing spatial correlation functions", *Kybernetik*, **5**, 113-119.

ASSIS, F. M. (1994), "Decodificação de Códigos de Geometria Algébrica e Uso de Redes Neurais para Cálculo em Corpo Finito", Dissertação de Doutorado, Departamento de Engenharia Elétrica, PUC/RJ.

BARBOSA, V. C. e CARVALHO, L. A. V. (1990), "Feasible Directions Linear Programming by Neural Networks", *International Joint Conference on Neural Networks (IJCNN)*.

BARBOSA, V. C. e CARVALHO, L. A. V. (1991), "Learning in Analog Hopfield Neural Networks", *International Joint Conference on Neural Networks (IJCNN)*.

BATTITI, R. (1992), "First- and Second-Order Methods for Learning : Between Steepest Descent and Newton's Method", *Neural Computation*, Vol. 4, 141-166.

BECKER, S. e Le CUN, Y. (1989), "Improving the convergence rate of back-propagation learning with second order methods", *Proceedings of the 1988 Connectionist Models Summer School*, 29-37.

BRYSON, A. E. e HO, Y-C. (1969), Applied Optimal Control, Hemisphere Publishing.

CARDADOR, D. M. (1990), "Representação do Conhecimento : Modelos Clássicos e Conexionistas", Dissertação de Mestrado, IME.

CARDOSO, L. A. L. S. (1992), "Análise de Modelos Plásticos de Redes Neurais para o Processamento de Sinais", Dissertação de Mestrado, Departamento de Engenharia Elétrica, PUC/RJ.

CARPENTER, G. A. e GROSSBERG, S. (1987), "A massively parallel architecture for a self-organizing neural pattern recognition machine", *Computer Vision, Graphics and Image Processing*, **37**, 54-115.

CARVALHO, L. A. V. e BARBOSA, V. C. (1989), "Towards a Sthocastic Neural Model for Combinatorial Optimization", *First International Joint Conference on Neural Networks*.

CARVALHO, L. A. V. e BARBOSA, V. C. (1990), "A TSP Objective Function That Ensures Feability at Stable Points", *International Neural Networks Conference (INNC)*, Paris, France.

CLARK, A. e THORNTON, C. (1994), "Trading Spaces : Computation, Representation and the Limits of Uninformed Learning", Washington University in St Louis.

DAHL, E. D. (1987), "Accelerated Learning Using the Generalized Delta Rule", Proceedings IEEE 1st International Conference on Neural Networks, Vol. 2, 523-530.

DENNIS, J. E. e MEI, H. H. (1979), "Two new unconstrained optimization algorithms which use function and gradient values", *Journal of Optimization Theory and its Aplications*, **28**, 453-482.

DENNIS, J. E. e SCHNABEL, R. B. (1983), **Numerical Methods for Unconstrained Optimization and Nonlinear Equations**, Englewood Cliffs, Prentice-Hall.

DENNIS, J. E. e SCHNABEL, R. B. (1989), "A View of Unconstrained Optimization", **Handbooks in Operation Research and Mathematical Software**, G. L. Newhauser, 1-72.

ELDREDGE, J. G. e HUTCHINGS, B. L. (1994), "RRANN : A Hardware Implementation of the Backpropagation using Reconfigurable FPGAs", *IEEE World Conference on Computational Intelligence*, 77-80, Orlando, Florida.

FAHLMAN, S. E. (1988), "An empirical study of learning speed in back-propagation networks", Technical Report CMU-CS-88-162, Carnegie-Mellon University.

FANTY, M. A. (1988), "Learning in structured connectionist networks", Technical Report 252, Computer Science Department, University of Rochester.

FUKUSHIMA, K. (1969), "Visual feature extraction by a multilayered network of analog threshold elements", *IEEE Trans. Systems, Sci. & Cyber.*, **SSC-5(4)**, 322-333.

GASPAR, E. R. (1993), "Solução do Problema Cinemático Inverso de Manipuladores Redundantes via Redes Neuronais", Dissertação de Mestrado, Programa de Engenharia Mecânica, COPPE/UFRJ.

GAY, D. M. (1983), "Subroutines for unconstrained minimization using a model/trust-region approach", *ACM Transactions on Mathematical Software* **9**, 503-524.

GOLDSCHMIDT, R. R. (1991), "Redes Neurais em Sistemas para Reconhecimento de Palavras Isoladas", Dissertação de Mestrado, IME.

GORMAN, P e SEJNOWSKI, T (1988), "Learned classification of sonar targets using a massively parallel network", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **36**, 1135-1140.

GROSSBERG, S. (1969), "Embedding fields : A theory of learning with physiological implications", *J. Math. Psych.*, **6**, 209-239.

GROSSBERG, S. (1976), "Adaptive pattern classification and universal recoding : I. Parallel development and coding of neural detectors", *Biological Cybernetics*, **23**, 121-134.

GROSSBERG, S. e SCHMAJUK, N. A. (1987), "Neural dynamics of attentionally modulated Pavlovian conditioning", *Psychobiology*, **15**, 195-240.

GROSSBERG, S. e STONE, G. (1986), "Neural dynamics of word recognition and recall : Attentional priming, learning and resonance", *Psychological Review*, **93**, 46-74.

HEBB, D. (1949), **The Organization of Behavior**, Wiley.

HEBDEN, M. D. (1973), "An algorithm for minimization using exact second derivatives", Atomic Energy Research Establishment, Report T.P. 515, Harwell, England.

HECHT-NIELSEN, R. (1990), **Neurocomputing**, Addison-Wesley.

HOPFIELD, J. J. (1982), "Neural networks and physical systems with emergent collective computacional abilities", *Proc. Natl. Acad. Sci.*, **79**, 2554-2558.

HOPFIELD, J. J. (1984), "Neurons with graded response have collective computacional properties like those of two-state neurons", *Proc. Natl. Acad. Sci.*, **81**, 3088-3092.

HOPFIELD, J. e TANK, D. (1985), "'Neural' computation of decisions in optimization problems", *Biological Cybernetics*, **52**, 141-152.

HUSH, D. R. e SALAS, J. M. (1988), "Improving the learning rate of back-propagation with the gradiente reuse algorithm", *Proceedings IEEE 2nd International Conference on Neural Networks*, Vol. 1, 441-448.

HUSH, D. R., SALAS, J. M. e HORNE, B. (1991), "Error surfaces for multi-layer perceptrons", *Proceedings of the IEEE International Conference on Neural Networks (ICNN)*, Vol. I, 759-764.

HWANG, J. N., LAY, S. R., MAECHLER, M., MARTIN, R. D. e SCHIMERT, J., (1994), "Regression Modeling in Back-propagation and Projection Pursuit Learning", *IEEE Transactions on Neural Networks*, vol. 5, n. 2, 342-346.

KALMAN, B. L e KWASNY, S. C. (1994), "TRAINREC : A System for Training Feedforward & Simple Recurrent Networks Efficiently and Correctly", Department of Computer Science, Washington University in St. Louis.

KLOPF, A. H. e GOSE, E. (1969), "An evolutionary pattern recognition network", *IEEE Trans. Systems, Sci. & Cyber.*, **SSC-5(3)**, 247-250.

KOHONEN, T. (1970), "Correlation matriz memories", Helsinki University of Technology Report TKK-F-A130.

LAPEDES, A. S. e FARBER, R. M. (1987), "Nonlinear signal processing using neural networks : prediction and system modelling", Technical Report LA-UR-87-2662, Los Alamos National Laboratory.

LIPPMANN, R. (1987), "An introduction to computing with neural nets", *IEEE ASSP Magazine*, **4**, 4-22.

LUENBERGER, D. G. (1984), **Linear and Nonlinear Programming**, 2nd ed., Addison-Wesley.

MACHADO, R. J. e BARBOSA, V. (1992), "Using Neural Networks for the Interpretation of Images from the Amazonia", *1º Workshop Nacional em Redes Neuronais*, 1-3, PUC/RJ.

MACHADO, R. J., FERLIN, C., ROCHA, A. F. e SIGULEM, D. (1991), "NEXTOOL : An Environment for Connectionist Expert Systems", *Latin American Conference on Artificial Inteligence in Petroleum Exploration and Production (LAIC-PEP'91)*, 121-132.

MARTINO, M. B. (1994), "Redes Neurais aplicadas à Aquisição do Conhecimento em Sistemas Elétricos de Potência", Dissertação de Mestrado, Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ.

McCLELLAND, J. L. e RUMELHART, D. E. (1988), **Explorations in Parallel Distributed Processing**, MIT Press.

McCULLOCH, W. S. e PITTS, W. (1943), "A logical calculus of the ideas immanent in nervous activity", *Bulletin of Math. Bio.*, **5**, 115-133.

MELO, M. P. (1991), "Redes Neurais Artificiais : Uma Aplicação à Previsão de Preços de Derivados de Petróleo", Dissertação de Mestrado, Departamento de Informática, PUC/RJ.

MINSKY, M. e PAPERT, S. (1969), **Perceptrons**, MIT Press.

MORAES, R. A. (1990), "Avaliação da Adequação dos Modelos de Redes Neurais para Reconhecimento de Padrões de Guerra Eletrônica", Dissertação de Mestrado, IME.

MORÉ, J. J. (1977), "The Levenberg-Marquardt algorithm : Implementation and theory", **Proceedings of the Dundee Conference on Numerical Analysis**, G. A. Watson, Springer-Verlag, 105-116.

MORÉ, J. J. e SORENSEN, D. C. (1984), "Newton's method", **Studies in Numerical Analysis**, G. H. Golub, 29-82.

NESTOR, INC. (1988), "The Nestor Development System" (company literature).

PARKER, D. B. (1985), "Learning-logic", Technical Report TR-47, Center for Computacional Res. in Economics and Management Sci., MIT.

PARKER, D. B. (1987), "Optimal Algorithms for Adaptive Networks : Second Order Back Propagation, Second Order Direct Propagation, and Second Order Hebbian Learning", Proceedings IEEE 1st International Conference on Neural Networks, Vol. 2, 593-600.

PARLOS, A. G., CHONG, K. I. e ATTYA, A. F. (1994), "Application of the Recurrent Multilayer Perceptron in Modeling Complex Process Dynamics", *IEEE Trans. on Neural Networks*, volume 5, number 2, 255-258.

POWELL, M. J. D. (1970), "A hybrid method for nonlinear equations", **Numerical Methods for Nonlinear Algebraic Equations**, P. Rabinowitz, Gordon and Breach, 87-114.

PRATT, L. Y., TRACY, L. C. e NOORDEWIER, M. (1994), "Back-propagation Learning on Ribosomal Binding Sites in DNA Sequences using Preprocessed Features", *Proceedings of the IEEE (ICNN-94)*, volume 5, 3332-3335.

PRECHELT, L. (1994), "PROBEN1 - A Set of Neural Network Benchmark Problems and Benchmarking Rules", Technical Report 21/94, Universität Karlsruhe.

QUEIROZ NETO, I. A. e RODRIGUES, F. S. (1991), "Eletrofacies Identification using Neural Networks", *Latin American Conference on Artificial Inteligence in Petroleum Exploration and Production (LAIC-PEP'91)*, 32-42.

RAPOSO, C. M. (1992), "Redes Neuronais na Previsão de Séries Temporais", Dissertação de Mestrado, COPPE/UFRJ.

REINSCH, C. H. (1971), "Smoothing by Spline functions II", *Numer. Math.*, **16**, 451-454.

RIEDMILLER, M. e BRAUN, H. (1993), "A direct adaptive method for faster backpropagation learning: The RPROP algorithm", *Proceedings of the IEEE International Conference on Neural Networks (ICNN)*, 586-591.

RIEDMILLER, M. (1994a), "RPROP - Description and Implementation Details", Technical Report, January 1994, University of Karlsruhe.

RIEDMILLER, M. (1994b), "Advanced Supervised Learning in Multi-layer Perceptrons - From Backpropagation to Adaptive Learning Algorithms", *Computer Standard & Interfaces*, volume 16, special issue on Neural Networks, Elsevier Science Publishers, Amsterdam.

ROCHA, A. F. et al. (1992), "A neural net for extracting knowledge from natural language data bases.", IEEE Transactions on Neural Networks, v. 3, n. 1, 1-10.

RODRIGUES, F. S. e QUEIROZ NETO, I. A. (1991), "Classificação de Eletrofácies Utilizando Redes Neuronais", *1º Encontro de Inteligência Artificial Aplicada à Indústria do Petróleo*, 132-145, Rio de Janeiro.

RODRIGUES, F. S. e QUEIROZ NETO, I. A. (1992), "Aplicação de Inteligência Artificial na Identificação de Eletrofácies - Redes Neuronais versus Análise Discriminante", *Boletim de Geociências da Petrobrás*, v. 6 (3/4), 155-161.

ROSENBLATT, F. (1958), "The perceptron : A probabilistic model for information storage and organization in the brain", *Psychol. Rev.*, **65**, 386-408.

ROSENBLATT, F. (1961), **Principles of Neurodynamics**, Spartan Books.

RUMELHART, D. E. e McCLELLAND, J. L. (1986), **Parallel Distributed Processing : Explorations in the Microstruture of Cognition, I e II**, MIT Press.

SATUF, E. N. (1993), "Estudo da Viabilidade de Redes Neurais na Solução do Problema de Decodificação Binária", Dissertação de Mestrado, Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ.

SEJNOWSKI, T. J. e ROSENBERG, C. R. (1986), "NETtalk : A parallel network that learns to read aloud", Johns Hopkins University EE & CS Technical Report JHU/EECS-86/01.

SILVA, V. N. A. L. (1994), "Diagnose em Sistemas de Potência utilizando Lógica Não-monotônica e Redes Neurais", Dissertação de Mestrado, Programa de Engenharia de Sistemas e Computação, PUC/RJ.

SOUZA JUNIOR, M. B. (1993), "Redes Neuronais Multicamadas aplicadas a Modelagem e Controle de Processos Químicos", Dissertação de Mestrado, Programa de Engenharia Química, COPPE/UFRJ.

SOVAT, R. B. (1994), "Retropropagação Recorrente : Aplicação na Previsão de Séries Temporais e Identificação de Assinaturas Sonoras", Dissertação de Mestrado, Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ.

VON NEUMANN, J. (1951), "The general and logical theory of automata", **Cerebral Mechanisms in Behavior**, 1-41, Wiley.

VON NEUMANN, J. (1956), "Probabilistic logics and the synthesis of reliable organisms from unreliable components", **Automata Studies**, 43-98, Princeton University Press.

WATROUS, R. L. (1987), "Learning Algorithms for Connectionist Networks : Applied Gradient Methods of Nonlinear Optimization", *Proceedings IEEE 1st International Conference on Neural Networks*, Vol. 2, 619-628.

WERBOS, P. J. (1974), "Beyond regression : New tools for prediction and analysis in the behavioral sciences", Dissertação de Doutorado, Appl. Math., Harvard University.

WIDROW, B. e HOFF, M. E. (1960), "Adaptive switching circuits", *1960 IRE WESCON Convention Record*, 96-104.

WILLSHAW, D. J., BUNEMAN, O. P. e LONGUET-HIGGINS, H. C. (1969), "Non-holographic associative memory", *Nature*, **222**, 960-962.

ZANDONADE, E. (1993), "Aplicação da Metodologia de Redes Neurais em Previsão de Séries Temporais", Departamento de Engenharia Elétrica, PUC/RJ.