Otimização da generalização de uma Rede Neural utilizando Algoritmos Genéticos

Erico S. Ventura, Cláudio A. Amorim

***Resumo*—A method to maximize the generalization capability of a Multilayer Perceptron (MLP) is purposed, and applied to the recognition of printed characters. A Genetic Algorithm is used to find an optimal match between the fonts in the training set and the number of hidden nodes in the MLP. The results obtained were satisfactory, and reinforce the idea that the number of hidden nodes of a neural network should vary in accordance with the specific data used to train it.**

***Index Terms*—Artificial Neural Networks, Genetic Algorithms, Generalization, Optimization, Character recognition**

I. Introdução

UMA das maiores dificuldades encontradas no projeto de uma rede neural artificial (RNA) é a otimização da estrutura e parâmetros junto com um conjunto de treinamento eficaz, a fim de que a rede possua uma boa capacidade de generalização. A busca por uma configuração ótima é essencialmente experimental, e por isso o projeto de uma RNA "é mais uma arte do que uma ciência"[10]. Existem alguns trabalhos teóricos sobre esse tema, mas a maioria não são muito úteis na prática [2].

Os perceptrons de múltiplas camadas (*MLP - Multi-Layer Perceptron*), tipo de rede comumente utilizado para classificação de padrões, podem apresentar problemas de convergência ou de má generalização quando não possuem uma quantidade adequada de neurônios ocultos. Uma alternativa para contornar este problema são os algoritmos de aprendizagem construtivos, que começam com uma rede pequena e introduzem novos neurônios ou conexões sob demanda, até que a rede se encontre em condições satisfatórias [8].

Segundo Simon Haykin [10], os fatores que influenciam diretamente a generalização são: o tamanho do conjunto de treinamento e a sua representatividade com relação à classe a que pertence, a arquitetura da rede e a complexidade física do problema tratado. Assim, para otimizar a generalização torna-se necessário atentar não apenas para as dimensões da rede, mas também para o seu conjunto de treinamento, conforme o SELF (*Selective Learning with Flexible neural architectures*), proposto por Byoung-Tak Zhang [2].

O presente trabalho propõe um método que utiliza um algoritmo genético para otimizar a capacidade de generalização de uma rede neural aplicada ao reconhecimento de caracteres, baseando-se na relação de dependência entre um dado conjunto de treinamento e quantidade suficiente de nós ocultos - que agem como detectores de características [1] [10] - para que a rede possa extrair satisfatoriamente as características deste conjunto.

E. S. Ventura, C. A. Amorim - Departamento de Ciências Exatas e da Terra, Universidade do Estado da Bahia
R. Silveira Martins, 2555 Cabula, Salvador - Bahia
CEP: 41.150-000. Phone: (71) 3117-2200
E-mail: ventura@ufba.br, claudio.a.amorim@terra.com.br

O método proposto difere do SELF por não ser um algoritmo construtivo - já que a rede é treinada com uma quantidade fixa de exemplos e de neurônios - e por realizar a busca pela melhor solução aplicando diretamente os parâmetros (a serem otimizados) na rede, dispensando o conhecimento prévio sobre a representatividade dos exemplos disponíveis.

Esta associação entre os neurônios ocultos e o conjunto de treinamento existe porque, durante a aprendizagem, a rede desenvolve relações internas, nem sempre evidentes, entre estes neurônios e características específicas dos dados de entrada [7]. Assim, é notório que a relação mais significativa está na dimensão da rede necessária para mapear uma característica específica, comum em um dado conjunto de treinamento.

II. Descrição do método proposto

Neste trabalho, o problema tratado é o reconhecimento de caracteres digitais numéricos, por ser uma típica aplicação de reconhecimento de padrões através das RNAs. O modelo de RNA utilizado é um perceptron de múltiplas camadas treinado com o algoritmo backpropagation. Para o treinamento, os exemplos disponíveis são vetores que representam imagens contendo um caractere numérico gerados a partir de um dos 16 tipos de fontes adotados.

A proposta deste trabalho é utilizar um algoritmo genético para otimizar a quantidade média de acertos para o conjunto de validação. O uso de tal método mostra-se adequado para tal objetivo, uma vez que a determinação de uma boa configuração para a rede depende fundamentalmente do problema a ser tratado [1]. Ademais, dentre as vantagens dos algoritmos genéticos, podemos destacar a capacidade de maximizar uma função objetivo, mesmo sem o conhecimento específico do problema ou de sua complexidade, aplicando diretamente valores à função objetivo [8] [9].

Uma parte de cada cromossomo representa os exemplos - que irão compor o conjunto de treinamento - enquanto a outra representa a quantidade de neurônios da camada oculta. Cada gene foi associado a um grupo de exemplos, dependendo do tipo de aplicação. No problema de reconhecimento de padrões tratado neste trabalho, cada gene foi associado um grupo de exemplos, ou seja, cada gene associado a um tipo de fonte com a qual serão gerados 10 exemplos, cada um contendo um caractere numérico.

Para se calcular o valor da função objetivo de um dado cromossomo configura-se uma rede com os parâmetros re-

presentados por este cromossomo, que é treinada até atingir algum de seus critérios de parada. Depois, calcula-se a média de acertos para o conjunto de validação e temos assim o valor da função objetivo: percentual que representa a capacidade de generalização da rede testada. Isso é feito para cada cromossomo da população, que são ordenados de acordo com sua função objetivo. Dessa forma, são atribuídos os valores de aptidão dos indivíduos da população, é feita a seleção e, em seguida, são aplicados operadores de *crossover* e *mutação* para gerar uma nova população. Quando o critério de parada do algoritmo genético for alcançado, o melhor cromossomo encontrado indicará a melhor combinação entre conjunto de treinamento e estrutura da rede para a obtenção de uma boa generalização.

III. Descrição da rede neural artificial utilizada

A rede neural artificial utilizada no experimento, um Perceptron de múltiplas camadas, é composto por apenas uma camada oculta, e treinado com o algoritmo de retropropagação de erro, que é o modelo usado com mais freqüência em aplicações de reconhecimento de caracteres [1] [6] [7] [11]. A arquitetura da rede baseia-se no exemplo descrito por Haykin [10]. Os sinais de entrada são imagens monocromáticas com 8 pixels de largura e 12 de altura, geradas dinamicamente e contendo um caractere numérico na sua área central. Cada pixel da imagem gerada corresponde a uma unidade da camada de entrada da rede, que conseqüentemente possui 96 nós, correspondentes ao produto de 12 x 8 pixels.

O algoritmo de aprendizagem utilizado foi o algoritmo de retropropagação de erro padrão, com a utilização da constante de momento (*momentum*) para melhorar o desempenho da rede durante o treinamento e, eventualmente, evitar a incidência de mínimos locais [1].

A camada de saída é constituída de 10 nós - cada um representando uma classe (ou seja, um dígito) - de forma que, quando apenas um desses nós é ativado, julgamos que a rede considerou o caractere apresentado como sendo pertencente a esta classe. Caso contrário, considera-se que a rede não foi capaz de classificar o dado.

A função de ativação utilizada em todos os nós da rede é a função sigmoidal logística, sendo que, na última camada, para que a saída de um determinado neurônio seja considerada ativa, seu valor deve ser superior a um limiar que é definido, depois de encerrado o treinamento, através de um método iterativo de bissecção, para encontrar o valor que possibilite a rede apresentar o maior percentual de acertos. O valor mínimo para este limiar foi definido como 0,5 e o máximo 0,99. Assim, neste intervalo inicial são feitas divisões sucessivas em duas partes iguais, e o intervalo é repetidamente reduzido à metade na qual o maior valor da função objetivo possa ser encontrado. Estas iterações se repetem até que o tamanho do intervalo seja menor ou igual a 0,001.

Os pesos das conexões e bias da rede são inicializados com valores aleatórios entre -0,5 e +0,5 [7]. O valor da constante de momento utilizada é 0,65 e a taxa de aprendizagem utilizada é 0,10, por ser um valor médio entre o menor e o maior valor da faixa que vai de 0,05 a 0,25, sugerida por James Freeman e David Skapura [7]. O critério de parada adotado para encerrar o treinamento da rede é uma combinação de critérios, que implica na minimização do erro médio quadrado e da norma euclidiana do vetor gradiente da superfície de erro em relação ao vetor de pesos, até que o valor de ambos alcance 0,0001, ou quando o número de épocas for igual a 3000.

TABELA I
Lista das fontes utilizadas para gerar os exemplos

Índice	Tipo de fonte
1	Times New Roman
2	Arial
3	Courier New
4	Futura Lt BT
5	Charlesworth
6	MS Sans Serif
7	Bookman Old Style
8	Arial Black
9	Garamond
10	Modern
11	Verdana
12	Fixedsys
13	Century Gothic
14	Comic Sans MS
15	Impact
16	Tahoma

Na rede neural projetada, foi utilizada apenas uma camada oculta, por ser geralmente suficiente [7], e foi definido que o seu tamanho máximo seria igual a 64 nós, quantidade que, em experiências anteriores com o modelo de rede neural utilizado, demonstrou ser suficiente [4]. O tamanho desta camada é representado pelos últimos 6 bits de cada cromossomo do algoritmo genético utilizado neste trabalho.

Para representar as camadas, foram utilizados vetores, estrutura empregada porque, segundo James Freeman e David Skapura, [7] ela é mais eficiente do que as listas ligadas. Isso se deve - além do fato dos vetores armazenarem dados homogêneos - à sua organização seqüencial e linear, o que torna muito mais rápido o acesso aos dados armazenados se comparado ás listas ligadas, em que é necessário procurar o endereço de todo novo valor.

Os tipos de fontes usados para gerar os caracteres que compõem o conjunto total de exemplos são mostrados na tabela I na mesma ordem em que são armazenadas em um vetor na aplicação e do lado esquerdo é mostrado o índice do gene que representa a fonte no cromossomo. Para evitar que a ordem de apresentação seqüencial dos exemplos influencie no mapeamento realizado pela rede, a ordem é aleatoriamente definida no inicio de cada época do treinamento [7] [10].

Para a geração dos dados apresentados à rede - seja

para treinamento ou para testes - é criada dinamicamente uma imagem bitmap usando um componente do Borland®Delphi™ chamado de Image. Então, o caractere de um determinado tipo de fonte é plotado na parte central do bitmap. Cada pixel dessa imagem é armazenado em uma das 96 posições de um vetor, conforme mostrado na figura 1, atribuindo-se o valor 0 para o pixel branco e 1 para o pixel preto.

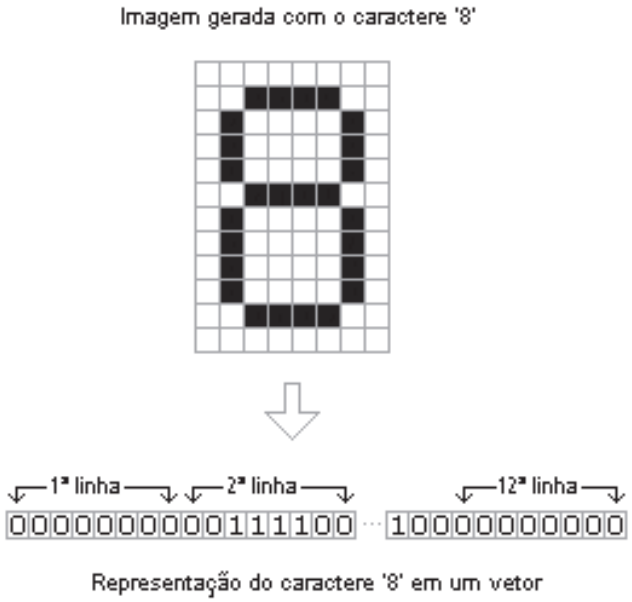

Figura 1. Exemplo de representação vetorial de um caractere

IV. Descrição do algoritmo genético utilizado

O algoritmo genético utilizado tem como função objetivo a ser maximizada a média dos percentuais de acertos da rede neural - usando como parâmetros de entrada um grupo de exemplos e o tamanho da camada oculta - para o conjunto de validação. Para representar estes dois parâmetros da rede, foram utilizados cromossomos com representação binária contendo 22 bits armazenados em *strings* [3]. Cada um dos 16 primeiros genes representam um grupo de exemplos (caracteres numéricos gerados com um tipo de fonte). Quando um certo gene g é igual a 1, isso significa que os 10 dígitos gerados a partir da g-ésima fonte do vetor que contém todas as fontes utilizadas farão parte do conjunto de treinamento. Os outros 6 genes restantes que compõem o cromossomo são a representação na base binária da quantidade de nós ocultos.

A população (*pool*) inicialmente definida era composta por 21 indivíduos, mas, para reduzir a ocorrência de convergência prematura, além de se modificar o tipo de seleção utilizada, seu tamanho foi aumentado para 51 indivíduos. Apesar da população inicial ser originada aleatoriamente, para que um indivíduo gerado possa fazer parte da população, ele deve ter sua factibilidade testada. Este teste consiste em verificar se o cromossomo gerado não é formado apenas por zeros e se o número de fontes utilizadas para produzir o conjunto de treinamento é igual ou inferior à quantidade máxima permitida, definida por um percentual na interface da aplicação.

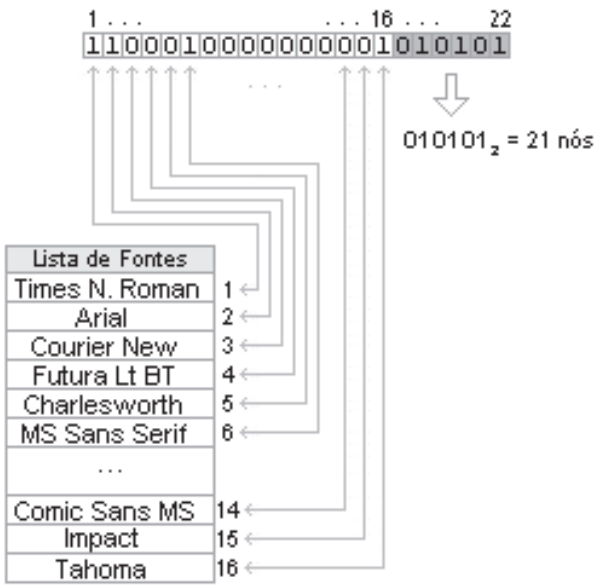

Figura 2. Exemplo de um cromossomo e sua configuração de rede correspondente

Na seleção dos indivíduos para a população intermediária (*matting pool*) foram utilizados os métodos da roleta e da seleção por torneio, sendo que este último reduziu a ocorrência de convergência prematura. A cada geração, é preservado o melhor cromossomo encontrado (*Elitismo*). Foram utilizados os operadores de *crossover, mutação, swap e inversão* [3].

O operador de *crossover* adotado é de 4 pontos, aleatoriamente determinados, com taxa igual a 80%, sendo garantido que, quando o operador é aplicado, dois pontos de corte ocorrem na primeira parte do cromossmo - que define o conjunto de treinamento - e os outros dois na segunda parte - a qual define a quantidade de nós ocultos. A aplicação do *crossover* é feita seqüencialmente aos pares da população intermediária, que contêm 1 cromossomo a menos que a população, resultando em um total de 25 pares. Um número aleatório é gerado para cada par e caso seja inferior a taxa de *crossover*, a aplicação do operador é repetida no cromossomo da população intermediária até que os dois filhos gerados sejam factíveis.

O operador de *mutação* é aplicado a cada gene dos cromossomos com taxa igual a 2%. Assim como o operador de *crossover*, a aplicação do operador de *mutação* no cromossomo se repete até que o novo cromossomo seja factível. Os operadores de *swap* e *inversão* possuem suas taxas iguais à taxa de *mutação*, e quando um número aleatório gerado para cada cromossomo é menor do que sua taxa, o operador é repetidamente aplicado até que o novo cromossomo produzido seja factível.

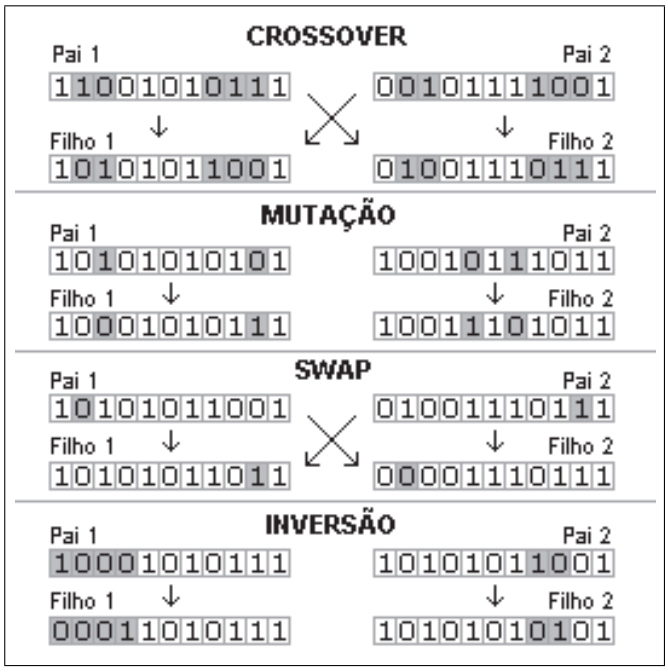

Figura 3. Exemplos dos quatro operadores genéticos utilizados

Para evitar processamento redundante, são armazenados em uma lista encadeada os cromossomos calculados com o seu respectivo valor da função objetivo. O valor da função objetivo só é calculado para um cromossomo caso este não esteja presente na lista. Este valor para um dado cromossomo pode variar a cada vez que for calculado, uma vez que, a cada novo treinamento da rede realizado, novos valores para os pesos serão gerados fortuitamente e a ordem de apresentação dos exemplos em cada época do treinamento será aleatoriamente definida. Por isso não há garantia de que o algoritmo de treinamento irá conduzir o vetor de pesos para o mesmo ponto de mínimo global na superfície de erro [7]. Entretanto, a possível variação não é significativa a ponto de justificar o custo computacional de recalcular os valores da função objetivo para indivíduos que já foram avaliados. As gerações do algoritmo genético utilizado são encerradas quando, na geração corrente, 90% dos indivíduos da população são iguais ao melhor indivíduo encontrado.

V. Descrição do Experimento

No desenvolvimento da aplicação, foi adotada a linguagem Object Pascal, usando como ambiente de desenvolvimento o Borland®Delphi™ 6 Enterprise Edition. A arquitetura de hardware utilizada possui um processador AMD Athlon® XP de 1,3 GHz e 128 MB de memória RAM. Os procedimentos executados não possuem nenhuma operação que seja fortemente dependente do hardware empregado, sendo assim, a mudança deste poderia afetar o desempenho da aplicação, mas não afetaria os resultados obtidos.

Na aplicação desenvolvida, antes de executar cada teste, é preciso definir o percentual máximo do conjunto total de exemplos que será utilizado no treinamento. Este valor será usado no teste de factibilidade de novos cromossomos gerados, a fim de garantir que a razão entre o número de fontes utilizadas (quantidade dos primeiros 16 genes do cromossomo que possuam valor igual a 1) e o número total de fontes seja igual ou inferior ao percentual definido. Quando o método é iniciado, é gerada uma população inicial de cromossomos factíveis que são avaliados e, de acordo com o seu valor da função objetivo, são colocados em ordem decrescente. Os operadores genéticos são aplicados e novas populações de indivíduos são geradas até que o critério de parada do algoritmo genético seja satisfeito.

TABELA II
Resultados obtidos nos testes

Fontes usadas (%)	Cromossomo	Valor da função	Gerações
10%	01000000000000000100	39,33	33
10%	00000100000000000011	39,33	23
10%	00010000000000000011	39,33	24
15%	00010010000000000101	56,43	57
15%	01000000000000001111	55,71	29
15%	00001010000000000100	57,14	18
20%	01000000100000001000	69,23	132
20%	01100001000000000011	66,92	35
20%	10001000000000001010	68,46	191
25%	01000000010100010000	77,5	190
25%	00001000010100010010	77,5	315
25%	00010000010100011011	76,66	313
35%	00010000011100011000	80,91	793
35%	00000101100000011001	80	1037
35%	01000000010100110101	81,82	46
40%	00110000011100011011	86	581
40%	00010001010100111010	86	1416
40%	00010000011100111010	86	1017
45%	01000001011100111001	91,11	6929
45%	11000000011100111011	90	3310
45%	10001000011100111001	90	4875

Para cada quantidade de fontes utilizadas no conjunto de treinamento - limitada pelo percentual que define o tamanho máximo do conjunto de treinamento - foram executados 3 testes, cujos resultados são apresentados na tabela II.

VI. Conclusão

Os resultados obtidos corroboram o método proposto como uma alternativa para otimização da generalização no problema do reconhecimento de caracteres. Os resultados obtidos caracterizam a dependência entre os dados utilizados no treinamento e o tamanho da camada oculta, visto que, para conjuntos de treinamento de mesmo tamanho, mas compostos por exemplos diferentes, a rede precisou de quantidades de neurônios ocultos significativamente diferentes como pode ser observado na tabela II.

Em redes com dimensões muito grandes, contudo, o

custo computacional pode inviabilizar o uso deste método. Por outro lado, é possível que a otimização da generalização justifique um longo tempo de processamento para a configuração da rede neural e definição dos conjuntos de treinamento. Vale ressaltar que este problema pode ser amenizado com o uso de processamento paralelo para o cálculo da função objetivo de cada cromossomo ou de uma plataforma de hardware com melhor desempenho. Outra possibilidade para tentar reduzir o custo computacional e o tempo despendido seria a otimização do algoritmo genético ou da própria rede neural.

A possibilidade de incidência de mínimos locais durante o treinamento da RNA também deve ser considerada. Apesar da utilização da constante de momento para evitar este problema, o fato dos pesos iniciais da rede serem definidos aleatoriamente, de não haver garantia de que ao final do treinamento o ponto na superfície de erro seja o mínimo global e de cada cromossomo ter a sua função objetivo calculada apenas uma vez, pode fazer com que cromossomos - que potencialmente seriam soluções ótimas - sejam descartados durante a seleção de indivíduos em uma dada geração do algoritmo genético devido à sua baixa aptidão.

Um fator fundamental para a eficácia do método é uma representação adequada do conjunto de exemplos no cromossomo. Este fator depende da natureza da aplicação e dos exemplos disponíveis para treinamento e validação. O critério de parada do treinamento da rede neural é um outro aspecto de extrema importância, uma vez que está diretamente relacionado com o valor da função objetivo obtido pelos cromossomos e com o custo computacional do método. Apesar do modelo de rede neural utilizado neste trabalho possuir apenas uma camada intermediaria, é possível representar mais de uma camada em um cromossomo, bastando para tal acrescentar a quantidade de genes necessária para representar na base binária a quantidade máxima de nós que esta camada irá possuir.

Referências

[1] Antônio de Pádua Braga, André Ponce de Leon F. de Carvalho, Teresa Bernarda Ludemir, *Redes Neurais Artificais: Teoria e Aplicações*, Rio de Janeiro-RJ, Editora LTC, 2000, pp. 11, 53, 56, 67, 72.

[2] Byoung-Tak Zhang, *An Incremental Learning Algorithm That Optimizes Network Size ans Sample Size in One Trial*, Sankt Augustin, Proceedings of IEEE International Conference on Neural Networks (ICNN'94), vol. 1, pp. 215-220, 1994.

[3] David E. Goldberg *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA, 1989.

[4] Erico da S. Ventura, *Um estudo sobre o problema da generalização em Redes Neurais*, Salvador-BA, 2004.

[5] Estéfane G. M. de Lacerda, André Carlos P. L. F. de Carvalho, *Sistemas Inteligentes: Aplicações a Recursos Hídricos e Ciências Ambientais*, Porto Alegre - RS, Editora da Universidade, 1999.

[6] Geoffrey E. Hinton, *How Neural Networks Learn from Experience*, In: Scientific American Magazine, set. 1992, pp. 106.

[7] James A. Freeman, David M. Skapura, *Neural networks: algorithms, applications, and programming techniques*, Addison-Wesley, Reading, MA, 1991, pp. 32-36, 89, 93, 103-105.

[8] Leandro Nunes de Castro, Eduardo Masato Iyoda, Eurípedes Pinheiro, Fernando Von Zuben, *Redes Neurais Construtivas: Uma Abordagem Comparativa*, São José dos Campos - SP, IV Congresso Brasileiro de Redes Neurais, 1999, pp. 102-107.

[9] Rodrigo Evangelista de Castro, *Otimização de estruturas com multi-objetivos via algoritmos genéticos*, Rio de Janeiro - RJ, 2001.

[10] Simon Haykin, *Redes Neurais: princípios e prática (Traduzido por Paulo Martins Engel)*, Editora Bookman, 2a. edição, Porto Alegre - RS, 2001, pp. 50, 205, 206, 225, 233.

[11] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, *Handwritten Digit Recognition with a Back-Propagation Network*, Holmdel, 1990.