Proceedings of the IV Brazilian Conference on Neural Networks - IV Congresso Brasileiro de Redes Neurais
pp. 102-107, July 20-22, 1999 - ITA, São José dos Campos - SP - Brazil

Redes Neurais Construtivas: Uma Abordagem Comparativa

Leandro Nunes de Castro Eduardo Masato Iyoda Eurípedes Pinheiro Fernando Von Zuben
{lnunes,emi,epsantos,vonzuben}@dca.fee.unicamp.br

Departamento de Engenharia de Computação e Automação Industrial (DCA)
Faculdade de Engenharia Elétrica e de Computação (FEEC)
Universidade Estadual de Campinas - UNICAMP

Abstract

Until recently, the determination of a proper dimension for an artificial neural network in a given application task usually involved only the designer's experience in implementing trial and error procedures. Nowadays, automatic design of neural network architectures is becoming part of the training process, by means of a more efficient exploration of the available information for supervised learning. Among the already proposed alternatives to automatic design, this paper emphasizes methods founded on the constructive paradigm. We chose three constructive algorithms for comparison: A heuristic search (A*), Cascade-Correlation with different activation functions (CASCOR) and Projection Pursuit Learning (PPL). A brief review of these constructive strategies in the solution of regression problems is presented, and their performance in terms of the parsimony of the resultant architecture is verified in benchmark problems.*

1. Introdução

Embora seja muito utilizada no tratamento de problemas de classificação de padrões, aproximação de funções e regressão não-linear, a rede neural do tipo MLP (perceptron de múltiplas camadas), pode apresentar-se mal-condicionada caso não seja feita uma escolha adequada de sua dimensão, que vai definir o espaço de parâmetros a ser explorado, diretamente vinculado ao número de neurônios da rede [15]. Isso se deve ao fato de que, redes com dimensão muito reduzida podem não apresentar flexibilidade suficiente para tratar determinados problemas, enquanto redes sobre-dimensionadas geralmente apresentam pouca capacidade de generalização [10].

A utilização de procedimentos de validação cruzada (VC) ou técnicas de regularização [11,16], diminuem os problemas em relação ao sobre-treinamento da rede e consequentemente amenizam os efeitos das redes sobre-dimensionadas, mas estas ainda continuam a exigir um esforço computacional excessivo e desnecessário. Portanto, algoritmos capazes de determinar automaticamente a arquitetura da rede são desejáveis.

Uma forma muito estudada de determinação automática da dimensão de uma arquitetura de rede neural são os chamados métodos de poda (*pruning methods*), cuja idéia é iniciar com uma arquitetura de dimensão elevada e ir retirando unidades ou conexões até que se chegue a uma dimensão adequada [17,18]. No entanto, estes métodos sempre trabalham com um esforço computacional adicional, exceto no final do processo de poda e além disso, se existir mais de uma arquitetura adequada para a solução, sempre vai se convergir para a de maior dimensão.

Estes são os principais fatores que justificam o emprego de métodos construtivos junto a este processo de definição automática da dimensão da rede neural. Dentre os métodos construtivos existentes na literatura, a arquitetura chamada *cascade-correlation* é provavelmente a mais conhecida [8]. Outros métodos como algoritmos que trabalham com *busca de projeção* (*Projection Pursuit Learning - PPL*) [12] e *métodos heurísticos* [5] estão sendo também bastante difundidos. A idéia básica é começar com uma arquitetura de rede de dimensão reduzida, e ir adicionando unidades intermediárias e/ou conexões até que uma solução adequada seja apresentada.

Neste trabalho serão feitas comparações entre estes métodos construtivos. O objetivo é determinar a dimensão da arquitetura resultante a partir da aplicação de cada estratégia aos mesmos problemas de teste.

2. Métodos Construtivos

Os *métodos construtivos* começam com uma arquitetura mínima de rede e adicionam neurônios até que uma solução adequada seja encontrada. A determinação da arquitetura de rede inicial nestes métodos é imediata. Além disso, os algoritmos construtivos sempre efetuam a busca a partir de arquiteturas de dimensão reduzida, resultando em um menor esforço computacional. Redes com dimensões reduzidas efetuam os passos *forward* e *backward*, necessários no processo de treinamento, com menor esforço computacional e podem ser descritas utilizando um conjunto menor de parâmetros. Além disso, o papel individual de cada neurônio torna-se mais evidente.

2.1 Estratégia de busca no espaço de estados

Nesta seção o problema de construção da rede é visualizado como uma busca no espaço de *estados* da rede neural [13, 14]. Os ingredientes essenciais em quaisquer problemas de busca são o espaço de estados, o estado inicial, a estratégia de busca e o critério de terminação da busca. Nesta discussão sobre os métodos construtivos, será dada ênfase especial à estratégia de busca. Esta formulação fornece uma estrutura conveniente para a análise de algoritmos construtivos.

Espaço de Estados

Dado um espaço de funções G, para implementar um elemento deste espaço utilizando uma arquitetura do tipo MLP, é necessário especificar:

1. *l*: a quantidade de unidades intermediárias da rede;
2. *C*: o grafo de conectividade, especificando como as entradas, as unidades intermediárias e as saídas estão interconectadas;
3. Γ: a forma das funções de ativação dos neurônios da camada intermediária; e
4. *W*: os parâmetros de toda a rede, incluindo o vetor de pesos (θ) e outros parâmetros associados a Γ.

Assim, uma quádrupla (*l*, *C*, Γ, *W*) pode ser unicamente utilizada para especificar um elemento $\hat{g} \in \mathsf{G}$. Note que os elementos da quádrupla não são independentes: o grafo de conectividade pode ser dado por *C* (*V*, *E*), onde *V* são as unidades (neurônios) e *E* as conexões (pesos). O *espaço de estados S* corresponde à coleção de funções que podem ser implementadas por uma dada classe de redes neurais. Cada estado $s \in S$ pode ser representado por $s = C(V, E)$.

Critério de Terminação da Busca

Critérios muito utilizados para a interrupção do procedimento de busca são: aguardar até que o erro de treinamento atinja um determinado limiar, ou até que o erro não sofra decrescimento significativo após ser adicionado um determinado número de unidades intermediárias. As vantagens destes critérios são a simplicidade e a facilidade de observação do erro de treinamento. Por outro lado, eles requerem a definição de vários parâmetros por parte do usuário.

Estratégia de Busca

Esta é a parte mais crítica dos métodos construtivos [14]. A estratégia de busca determina como mover-se de um estado para outro no espaço de estados, até que a busca seja finalizada. De maneira equivalente, determina como o grafo de conectividade *C* evolui durante a busca. Seja s_1 (V_1, E_1) o *estado atual* e s_2 (V_2, E_2) o *próximo estado*. Nos algoritmos construtivos, as seguintes propriedades são quase sempre satisfeitas:

$$\text{I.} V_1 \subseteq V_2; \qquad \text{II.} E_1 \subset E_2.$$

Ou seja, unidades e conexões existentes no estado atual devem ser preservadas no próximo estado, e deve haver algumas conexões adicionais no estado seguinte. Em princípio, é possível que haja mudança apenas em *E* (com $V_1 = V_2$) para alguns estados sucessivos. Isto pode levar a várias arquiteturas de redes distintas com desempenho similares, e portanto este esquema não é muito utilizado na prática. Portanto, geralmente tem-se na prática $V_1 \subset V_2$, com $|V_2| = |V_1| + 1$.

Transição de Estados

O *mapeamento de transição de estados* $\Delta: S \rightarrow S$, é um componente chave para os algoritmos construtivos. Os algoritmos construtivos, guiados pelas propriedades I e II acima, devem restringir as transições de estado possíveis, definindo um mapeamento adequado.

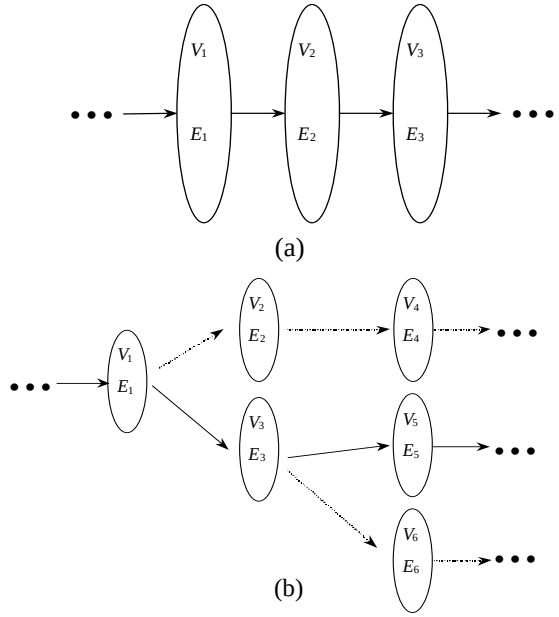


Figura 1: Mapeamentos de transição de estados. (a) Monovariável. (b) Multivariável.

Δ pode ser mono ou multivariável. No caso mais simples em que ele é monovariável, existe apenas um estado seguinte a ser explorado. A passagem de estados é representada por uma cadeia simples de estados (ver. Figura 1(a)). Um problema possível é a perda de flexibilidade.

Em um mapeamento Δ multivariável, geralmente existem vários próximos estados possíveis para determinado estado atual (ver. Figura 1(b)). Cada um deles é chamado de candidato; sendo que candidatos diferentes devem possuir diferentes quantidades de unidades escondidas, além de poderem apresentar arquiteturas distintas. A existência de múltiplos candidatos permite assim analisar várias arquiteturas de rede distintas, de forma que a escolhida seja a mais adaptada ao problema.

3. Algoritmos Construtivos

Nesta seção, serão apresentados os algoritmos construtivos que foram implementados para efeito comparativo: *projection pursuit learning* (PPL), *cascade-correlation híbrido* (CASCOR) e A*.

3.1 Projection Pursuit Learning (PPL)

Algoritmos nesta classe são inspirados em técnicas estatísticas baseadas nos modelos de regressão por busca de projeção (*projection pursuit regression* – PPR) propostos em [9].

Geralmente, a função de ativação é uma composição aditiva de *n* funções não-lineares (sem perda de generalidade, assume-se aqui uma única saída):

$$\hat{g}(\mathbf{x}) = \sum_{j=1}^{n} f_j(\mathbf{a}_j^T \mathbf{x}), \qquad (1)$$

onde **x** é o vetor de entrada, $\mathbf{a}_j$ é o vetor de projeção e f_j é a função de ativação do *j*-ésimo neurônio, também chamado de *smoother*. Existe uma similaridade evidente

entre o PPR e o PPL, o qual pode ser visto como uma rede neural com uma única camada intermediária. Considerando uma arquitetura PPL, a expressão para a função de aproximação é dada por:

$$\hat{g}_p = w_{0p} + \sum_{j=1}^{l}\left[w_{jp} f_j\left(\sum_{i=1}^{k} v_{ij}^T x_i + v_{0j} \right)\right], \qquad (2)$$

onde l e k são o número de unidades intermediárias e número de entradas, respectivamente; v_{ij} ($i\neq 0$) é o peso conectando a i-ésima entrada à j-ésima unidade intermediária, w_{jp} ($j\neq 0$) é o peso conectando a j-ésima unidade intermediária à p-ésima saída, e w_{0p} e v_0 são os limiares. As diferenças estão na presença de limiares e no escalonamento das funções de ativação f_j (j =1,..., l).

No PPL, o mapeamento de transição de estados é monovariável. O número de unidades intermediárias é aumentado de um a cada transição, e a nova unidade sempre é adicionada à mesma camada (ver Figura 2 para o caso de uma única saída).

Ao invés de utilizar função de ativação sigmoidal fixa e efetuar o treinamento envolvendo todas as unidades, estes algoritmos utilizam unidades intermediárias com função de ativação mais flexíveis e treinam uma unidade de cada vez. É feita a utilização do procedimento de retro-ajuste (*back-fitting*), que faz um ajuste cíclico das conexões ligadas a cada unidade já instalada, enquanto mantém fixos os parâmetros (pesos e outros parâmetros que definem as funções de ativação das unidades intermediárias) das outras unidades, até que não haja variação significativa no desempenho.

Função de Ativação das Unidades Intermediárias

Uma característica deste algoritmo é a utilização de funções de ativação mais flexíveis. No PPL, a função de ativação é não-paramétrica, como, por exemplo *splines*, ou paramétrica, por exemplo polinômios de Hermite. Para as redes que utilizam os polinômios de Hermite, cada unidade intermediária é representada como sendo uma combinação linear dada por:

$$f(z) = \sum_{r=1}^{R} c_r h_r(z), \qquad (3)$$

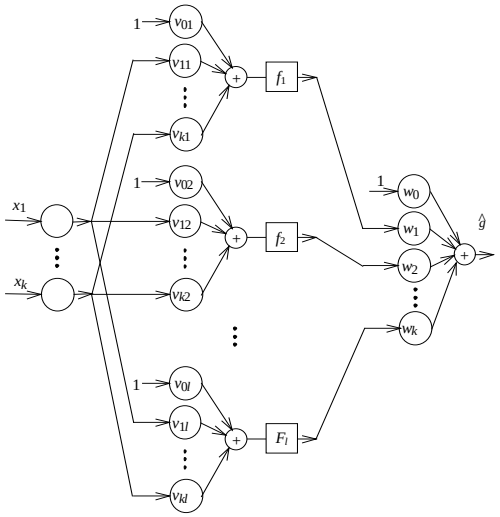


Figura 2: Arquitetura típica de um modelo PPL.

onde $h_r(z)$ é um polinômio ortonormal de Hermite e R é um parâmetro definido pelo usuário, chamado de *ordem* da função de ativação. Estas funções (polinômios de Hermite) permitem a interpolação suave e o cálculo rápido e preciso das derivadas. O parâmetro R controla o grau de flexibilidade da respectiva função de ativação.

3.2 Cascade Correlation Híbrido (CASCOR)

Ao contrário do algoritmo anterior, a arquitetura *cascade-correlation* proposta em [8], constrói redes com múltiplas camadas intermediárias. Este arranjo estrutural permite o desenvolvimento de associações funcionais poderosas, mesmo com unidades intermediárias simples.

Esquema de Conectividade

Quando uma nova unidade intermediária está para ser criada, além de estabelecer a conexão com cada uma das entradas e saídas originais da rede, também é feita uma conexão entre a nova unidade e as unidades intermediárias preexistentes. Cada nova unidade adiciona uma nova camada com um único neurônio à rede, gerando uma arquitetura em cascata (Figura 3). Geralmente as unidades intermediárias são simples, mas esta estrutura em cascata pode ser implementada com unidades intermediárias mais complexas.

Treinamento

Somente as conexões ligadas às novas unidades intermediárias são atualizadas até a convergência. As demais unidades em camadas anteriores são mantidas inalteradas. Primeiro os pesos ligados à nova unidade são ajustados (treinamento da entrada) através da otimização da função objetivo, enquanto todas as outras conexões são mantidas fixas. Geralmente existe um conjunto de candidatos a unidades intermediárias, cada um utilizando uma condição inicial aleatória diferente. Em seguida, as conexões ligadas a esta unidade são fixadas e os pesos que conectam as unidades intermediárias às saídas são atualizados (treinamento da saída). Se a saída da rede é linear, o treinamento da saída torna-se um problema linear e o conjunto de pesos da saída é convenientemente determinado utilizando-se a solução fechada dada pelo método dos quadrados mínimos.

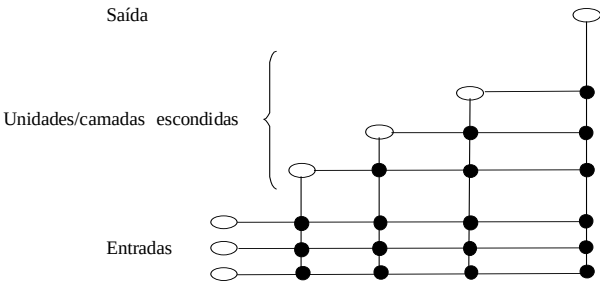


Figura 3: Arquitetura Cascade Correlation (CASCOR). No diagrama, elipses vazias representam unidades de entrada, intermediárias e de saída, enquanto os pontos pretos referem-se a conexões entre unidades.

As novas unidades intermediárias maximizam a covariância entre o erro residual e a ativação das unidades intermediárias:

$$S_{cascor} = \sum_{o} \left| \sum_{p} (J_{po} - \bar{J}_o)(H_p - \bar{H}) \right| \qquad (4)$$

onde p faz a varredura sobre todos os padrões de treinamento, o trata de todas as unidades de saída, H_p é a ativação da nova unidade intermediária para o padrão p, J_{po} é o erro residual da saída o para o padrão p, antes da adição da nova unidade intermediária, e $\bar{H}$ e $\bar{J}_o$ são os valores médios correspondentes para todos os padrões.

Funções de Ativação Distintas

O algoritmo original proposto em [8] foi modificado para permitir a utilização de mais de uma função de ativação como candidatos a compor a nova unidade a ser introduzida em cascata [6]. Isso significa que unidades com funções de ativação diferentes podem competir simultaneamente. O algoritmo CASCOR híbrido seleciona o melhor candidato, permitindo definir qual função de ativação, a partir de um conjunto finito, é mais adequada considerando-se o estágio atual do problema.

As Tabelas de 1 a 3, apresentam possíveis conjuntos de funções de ativação que foram utilizados na implementação do CASCOR.

Tabela 1: Funções do tipo senoidal.

Ativação	Expressão	Intervalo
Sen	$f(x) = \text{sen}(x)$	[-1.0, 1.0]
Cos	$f(x) = \cos(x)$	[-1.0, 1.0]

Tabela 2: Funções do tipo gaussiana.

Ativação	Expressão	Intervalo
Gaussiana	$f(x) = e^{-(x^2/2)}$	(0.0, 1.0]
Inverso da Gaussiana	$f(x) = -e^{-(x^2/2)}$	[-1.0, 0.0)

Tabela 3: Função do tipo sigmoidal.

Ativação	Expressão	Intervalo
Tangente Hiperbólica	$f(x) = \tanh(x)$	(-1.0, 1.0)

3.3 Algoritmo A*

A construção de um grafo de estruturas de redes neurais e a determinação de valores de avaliação para estes grafos são os problemas abordados pelo algoritmo A*. A aplicação deste algoritmo garante, teoricamente, a otimalidade da solução e da busca [5].

O algoritmo A* trabalha um grafo cujos arcos são denominados custos. Um subconjunto do conjunto de nós é o conjunto alvo, cujos elementos satisfazem algum critério de sucesso. A idéia básica do algoritmo A* é que, dado um nó n_i do grafo, o custo do caminho ótimo, partindo deste nó ao nó alvo n_G^*, seja descrito por uma função $h^*(n_i)$. Nem o caminho ótimo, nem os custos associados a ele são conhecidos. Por outro lado, se o custo associado ao caminho ótimo puder ser estimado por uma função, como $h(n_i)$, que avalia alguma heurística conhecida sobre o problema de busca, então a função de avaliação pode ser dada por:

$$f(n_i) = g(n_i) + h(n_i), \qquad (5)$$

onde $g(n_i)$ representa os custos associados ao melhor caminho conhecido, partindo do nó n_0 ao nó n_i , e $h(n_i)$ corresponde a uma estimativa dos custos do melhor caminho do nó n_i ao elemento mais próximo do conjunto alvo n_G^* (função heurística).

Esta função quantifica quão promissora é a expansão do nó n_i, e utiliza procedimentos de validação cruzada (VC) para sua avaliação, onde o conjunto de dados é dividido em treinamento, validação e teste [17]. O algoritmo A* simplesmente busca a estratégia de sempre expandir o nó mais promissor.

Aplicação do Algoritmo* A* *para a Solução do Problema de Otimização da Estrutura de uma Rede Neural

Considere o grafo de rede dado anteriormente por $C(V, E)$, onde V corresponde aos nós (unidades) e E corresponde às conexões (pesos). Partindo desse grafo, é possível construir um conjunto $\boldsymbol{A} = \{C\}$ infinito de estruturas de rede. Para aplicar um algoritmo de busca a este conjunto, é preciso definir relações entre seus elementos.

A aplicação de um algoritmo de busca é equivalente a definir um operador de expansão $\Gamma(C)$: $\boldsymbol{A} \rightarrow 2^{\boldsymbol{A}}$ que mapeia qualquer estrutura $C \in \boldsymbol{A}$ em um conjunto de sucessores (que é subconjunto de $\boldsymbol{A}$). Cada elemento $C \in \boldsymbol{A}$ pode ser visto como um nó do grafo cujos arcos são determinados por $\Gamma(C)$ (ver Figura 4). Para maiores detalhes sobre o operador de expansão, consulte Doering *et al.* [5].

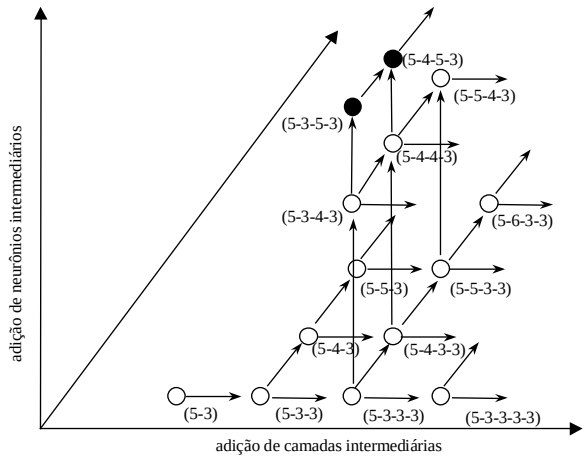


Figura 4: Aplicação sucessiva do operador de expansão $\Gamma(C)$ a uma estrutura inicial com cinco entradas, três saídas e nenhuma unidade intermediária.

Um critério de avaliação que determina o conjunto alvo ***G*** é:

$$\boldsymbol{G} = \{ C \in \boldsymbol{A} \mid \eta(C) \leq \eta_0 \}. \quad (6)$$

onde $\eta(C)$ é o valor estimado do erro de generalização aplicado a C, e η_0 o critério de parada escolhido. Assim, o conjunto alvo ***G*** é composto por todas as estruturas de rede que satisfazem o critério de avaliação.

Para aplicar o algoritmo A* a um valor não negativo apropriadamente escolhido de η_0 é necessário definir uma função custo e uma função heurística.

Função custo: a cada operador de expansão $C' \in \Gamma(C)$ é especificado um vetor de função de custo, composto por dois componentes:

- um representa a complexidade da arquitetura gerada em relação ao número de unidades intermediárias e
- o outro representa a complexidade da estrutura em relação à quantidade de camadas intermediárias.

Função heurística: a cada estrutura de rede C pode ser especificado um vetor de parâmetros θ^* que minimiza a esperança de uma função de custo. A função heurística constitui uma combinação convexa do erro para o conjunto de treinamento e do erro para o conjunto de avaliação [5].

4. Resultados Experimentais

Para comparar o desempenho dos três métodos construtivos descritos neste artigo (PPL, CASCOR e A*), serão abordados seis problemas distintos. Seja N o número de amostras disponíveis para treinamento, e $MSE = 0.01$ o critério de parada dos algoritmos para todos os problemas:

- problema de paridade 2 (XOR): $N = 4$;
- $\sin(x).\cos(2x)$: $N = 25$;
- ESP: problema de mundo real [1,2]; $N = 75$, $MSE = 0.01$;
- IRIS: este *benchmark* é parte do banco de dados para aprendizagem de máquina [19]; $N = 150$;
- GLASS: este *benchmark* é parte do banco de dados para aprendizagem de máquina [19]; $N = 214$ and
- SOJA: problema de mundo real [1,2], $N = 116$.

A ordem dos polinômios de Hermite utilizados pelo algoritmo PPL foi 5 para os problemas XOR e $\text{sen}(x)\times\cos(2x)$, 15 para o problema GLASS e 10 para os demais. O algoritmo A* possui funções de ativação do tipo tangente hiperbólica para os neurônios intermediários e de saída, enquanto o PPL e o CASCOR possuem saídas lineares e ativações intermediárias definidas durante o processo de treinamento. As camadas intermediárias do CASCOR são treinadas com o algoritmo *quickprop* [7] e a saída linear é calculada diretamente pelo método dos quadrados mínimos. O conjunto fechado de funções de ativação que podem ser utilizadas nas camadas intermediária do algoritmo CASCOR é composto pelas seguintes funções: tangente hiperbólica, $\text{sen}(x)$, $\cos(x)$, gaussiana e inverso da gaussiana (ver Seção 3.2).

O principal resultado a ser apresentado será o número de unidades e camadas intermediárias geradas pelos algoritmos para solucionar os problemas, onde *net* $[n_i\text{-}n_{hj}\text{-}n_o]$ representa a arquitetura da rede ao final do treinamento (com n_i igual ao número de entradas, n_{hj} a quantidade de unidades intermediárias na camada j e n_o o número de saídas da rede).

A Tabela 4 apresenta os resultados obtidos pelos métodos construtivos.

A Figura 5 apresenta alguns exemplos de funções de ativação construídas pelos polinômios de Hermite dos modelos baseados em busca de projeção (PPL) para o problema ESP, ilustrando a flexibilidade destas funções construídas iterativamente.

5. Conclusões

Os resultados apresentados na Tabela 4 mostram que os algoritmos A* e CASCOR nem sempre necessitam adicionar neurônios ou unidades intermediárias para resolverem os problemas propostos, dados os critérios de parada escolhidos.

Verificou-se que, embora o A* permita a criação de redes com múltiplas camadas e múltiplos neurônios por camada, sempre uma rede com uma única camada intermediária foi escolhida. Este fato reforça o teorema da aproximação universal, aplicável às redes neurais artificiais com uma camada intermediária [4].

Outro aspecto importante, é o fato de que o CASCOR sempre escolheu (para os problemas abordados) como função de ativação as funções seno ou coseno, embora outras funções estivessem disponíveis.

O algoritmo PPL apresentou resultados bons para problemas que consideram uma única saída e poucas entradas, e resultado pobre para um dos problemas cuja rede possui mais de uma saída (IRIS). Esta é uma das dificuldades do método e vários estudos têm sido desenvolvidos no sentido de amenizar os seus efeitos [3].

Referências

[1] **Castro, L.N., Von Zuben, F.J. & Martins, W.,** "Hybrid and Constructive Learning Applied to a Prediction Problem in Agriculture". *Proceedings of the IEEE IJCNN'98*, Alaska/USA, vol. 3, pp. 1932-1926, maio de 1998a.

[2] **Castro, L.N., & Von Zuben, F.J.,** "A Hybrid Paradigm for Weight Initialization in Supervised Feedforward Neural Network Learning", *Proceedings of the ICS'98 - Workshop on Artificial Intelligence*, Taiwan/China, pp. 30-37, dezembro de 1998b.

[3] **Breiman, L. & Friedman, J. H.**, "Predicting Multivariate Responses in Multiple Linear Regression", *Journal of the Royal Statistical Society B*, 59(1):3-54, 1997.

[4] **Cybenko, G.**, "Approximation by Superpositions of a Sigmoidal Function", *Mathematics of Control Signals and Systems*, vol. 2, pp. 303-314, 1989.

[5] **Doering, A., Galicki, M. & Witte, H.**, "Structure Optimization of Neural Networks with the A*-Algorithm", *IEEE Trans. on Neural Networks*, vol. 8, n° 6, pp. 1434-1445, 1997.

[6] **Ensley, D. & Nelson, D.E.,** "Extrapolation of Mackey-Glass data using Cascade Correlation", *SIMULATION*, 58:5, pp. 333-339, 1992.

[7] **Fahlman, S.E.,** "An Empirical Study of Learning Speed in Back-Propagation Networks", *Technical Report*, September 1988.

[8] **Fahlman, S.E., & Lebiere, C.**, "The Cascade Correlation Learning Architecture", *Advances in Neural Information Processing Systems 2* (*NIPS 2*), pp. 524-532, 1990

[9] **Friedman, J.H., & Stuetzle, W.**, "Projection Pursuit Regression", *Journal of the American Statistical Association*, vol. 76, n° 376, pp. 817-823, 1981.

[10] **Geman, S., Bienenstock, E. & Doursat R.**, "Neural Networks and the Bias/Variance Dilemma", *Neural Computation*, vol. 4, pp. 1-58, 1992.

[11] **Girosi F., Jones M. & Poggio T.**, "Regularization Theory and Neural Networks Architectures", *Neural Computation*, vol. 7, pp.219-269, 1995.

[12] **Hwang, J. N., Lay, S. R., Maechler M., Martin, R. D. & Schimert J.**, "Regression Modeling in Back-Propagation and Projection Pursuit Learning", *IEEE Trans. On Neural Networks*, vol. 5, n° 3, pp. 342-353, 1994.

[13] **Kwok T.Y., & Yeung. D.Y.**, "Use of a Bias Term in Projection Pursuit Learning Improves Approximation and Convergence Properties", *IEEE Trans. on Neural Networks*, vol. 7, n° 5, pp. 1168-1183, 1996.

[14] **Kwok T.Y., & Yeung. D.Y.**, "Constructive Algorithms for Structure Learning in Feedforward Neural Networks for Regression Problems", *IEEE Trans. on Neural Networks*, vol. 8, n° 3, pp. 630-645, 1997.

[15] **McKeown J.J., Stella F. & Hall G.**, "Some Numerical Aspects of the Training Problem for Feed-Forward Neural Nets", *Neural Networks*, vol. 10, n° 8, pp.1455-1463, 1997.

[16] **Prechelt, L.,** "Automatic Early Stopping Using Cross Validation: Quantifying the Criteria", *Neural Networks*, vol. 11, n° 4, pp. 761-767, 1998.

[17] **Reed, R.**, "Pruning Algorithms – A Survey", *IEEE Trans. on Neural Networks*, vol. 4, n° 5, pp. 740-747, 1993.

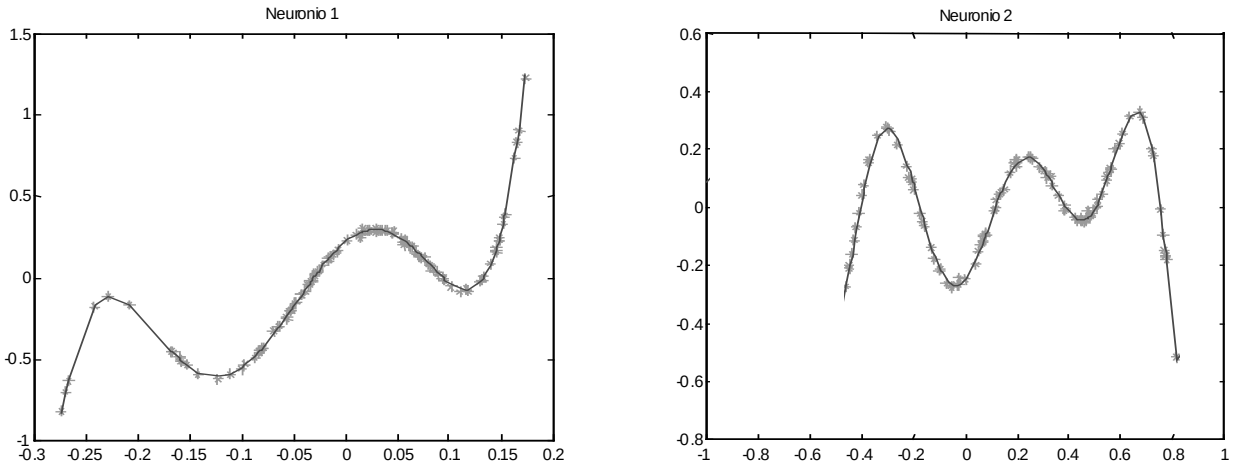
[18] **Thimm, G., & Fiesler, E.**, "Evaluating Pruning Methods", *Proceedings of the International Symposium on Artificial Neural Networks*, Taiwan, 1995

[19] ____ Neural Networks Benchmarks: URL: ftp://ftp.ics.uci.edu/pub/machine-leraning-databases.

Apêndice

Tabela 4: Comparação de arquiteturas geradas por cada um dos três métodos construtivos apresentados.

	Arquitetura resultante		
	PPL	A*	CASCOR
XOR	[2, **1**, 1]	[2, **2**, 1]	[2, **1**, 1]
$\sin(x).\cos(2x)$	[1, **3**, 1]	[1, **6**, 1]	[1, **2**, 1]
ESP	[2, **3**, 5]	[2, **0**, 5]	[2, **0**, 5]
IRIS	[4, **30**, 3]	[4, **4**, 3]	[4, **9**, 3]
GLASS	[9, **28**, 3]	[9, **24**, 3]	[9, **28**, 3]
SOJA	[36, **4**, 1]	[36, **3**, 1]	[36, **3**, 1]



(a) (b)

Figura 5: Exemplos de funções de ativação construídas pelo PPL para o problema ESP. A ordem dos polinômios de Hermite é 10. (a) Neurônio 1. (b) Neurônio 2.