UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO INDUSTRIAL

Contribuições para o Aprendizado por Busca de Projeção

Autor: Leonardo de Moraes Holschuh

Orientador: Prof. Dr. Fernando José Von Zuben

Co-Orientador: Prof. Dr. Clodoaldo Aparecido de Moraes Lima

Dissertação de Mestrado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica.

Área de Concentração: Engenharia de Computação

Comissão Examinadora:

Fernando José Von Zuben – DCA/FEEC/Unicamp – Presidente

Eduardo Raul Hruschka – ICMS/USP (São Carlos/SP) – Membro Externo

Romis R. de Faissol Attux – DCA/FEEC/Unicamp – Membro Interno

Campinas – São Paulo – Brasil
Novembro de 2008

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

H741c

Holschuh, Leonardo de Moraes
Contribuições para o aprendizado por busca de projeção / Leonardo de Moraes Holschuh. --Campinas, SP: [s.n.], 2008.

Orientadores: Fernando José Von Zuben, Clodoaldo Aparecido de Moraes Lima.
Dissertação de Mestrado - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Redes neurais (Computação). 2. Modelos não-lineares (Estatística). 3. Projeção. 4. Algoritmos genéticos. 5. Inteligência artificial. I. Von Zuben, Fernando José. II. Lima, Clodoaldo Aparecido de Moraes. III. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. IV. Título.

Título em Inglês: Contributions to projection pursuit learning
Palavras-chave em Inglês: Neural networks, Models non-linear (statistical), Projection, Genetic algorithms, Artificial intelligence
Área de concentração: Engenharia de Computação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora: Eduardo Raul Hruschka, Romis R. de Faissol Attux
Data da defesa: 14/11/2008
Programa de Pós Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE MESTRADO

Candidato: Leonardo de Moraes Holschuh

Data da Defesa: 14 de novembro de 2008

Título da Tese: "Contribuições para o Aprendizado por Busca de Projeção"

Prof. Dr. Fernando José Von Zuben (Presidente): Fernando José Von Zuben

Prof. Dr. Eduardo Raul Hruschka: Eduardo R. Hruschka

Prof. Dr. Romis Ribeiro de Faissol Attux: Romis

Agradecimentos

A toda minha família, pelo apoio e carinho.

Ao meu orientador, Prof. Dr. Fernando José Von Zuben, por me guiar neste processo e pela compreensão nos momentos difíceis.

Ao meu co-orientador, Clodoaldo Aparecido de Moraes Lima, pelas dicas e sugestões no decorrer dos estudos.

A todo o pessoal do LBiC, pela companhia na jornada.

Aos professores e funcionários da FEEC, cujos esforços têm me acompanhado desde a graduação.

Ao CNPq, pelo apoio financeiro.

Dedico este trabalho aos meus pais, Maria da Conceição e Heinz Johann, e a Patricia, que me deram apoio e incentivo, sempre que necessário.

Resumo

A obtenção de modelos parcimoniosos é uma necessidade em vários problemas de engenharia, como no caso de projeto de sistemas embarcados. Algoritmos construtivos para treinamento supervisionado têm apresentado efetividade como metodologias de projeto de redes neurais artificiais (RNAs) parcimoniosas, de boa acurácia e capacidade de generalização, embora requeiram mais recursos computacionais durante o processo de síntese da RNA. O aprendizado por busca de projeção está entre os métodos construtivos mais utilizados, mas ainda apresenta algumas limitações, sendo que aqui se procura tratar três delas: a inicialização da direção de projeção, o emprego de entrada de polarização junto aos neurônios da camada intermediária e a seleção de variáveis visando a redução no número de entradas, ou seja, na dimensão do vetor de projeção. Utilizou-se uma técnica de seleção de variáveis denominada *wrapper*, cuja implementação envolveu o emprego de um algoritmo genético, e realizaram-se experimentos de análise de desempenho no contexto de predição de séries temporais, indicando que as três propostas sugeridas trazem contribuições para o processo de aprendizado construtivo.

Palavras-chave: aprendizado construtivo por busca de projeção, redes neurais artificiais, seleção de variáveis, algoritmo genético, predição de séries temporais.

Abstract

The production of parsimonious models is a common demand on a wide variety of engineering problems, as in the design of embedded systems. Constructive algorithms for supervised learning have shown to be effective methodologies for the synthesis of parsimonious artificial neural networks, with high levels of accuracy and generalization capability, though requiring more computational resources during the training phase. Even being one of the most frequently adopted constructive learning methods, the projection pursuit learning algorithm still presents some limitations, and three of them will be treated here: the initialization of the direction of projection, the use of a bias term at the input of the hidden-layer neurons, and the selection of input variables as a form of reducing the number of inputs, i.e. the dimension of the vector of projection. The variable selection technique adopted here is denoted wrapper, and a genetic algorithm was considered as the search engine. The performance analysis has been carried out by experiments involving time series prediction, indicating that the three propositions suggested to deal with limitation of projection pursuit learning contribute favorably to the process of constructive learning.

Keywords: projection pursuit for constructive learning, artificial neural networks, variable selection, genetic algorithm, time series prediction.

Sumário

Lista de Figuras

Lista de Tabelas

Trabalho Publicado pelo Autor

HOLSCHUH, L.M., LIMA, C.A.M., VON ZUBEN, F.J. “A Wrapper for Projection Pursuit Learning”, *Proceedings of the International Joint Conference on Neural Networks*. Orlando. United States of America. pp. 2892-2897. August 2007. (ISBN: 1-4244-1380-X)

Capítulo 1

Introdução e Motivação

As redes neurais artificiais (RNAs) estão entre os mais bem sucedidos modelos de aprendizado a partir de dados amostrados (CHERKASSKY & MULIER, 2007). No contexto de aprendizado supervisionado, em que existem valores desejados para a saída da rede neural relacionados aos estímulos de entrada, as RNAs podem ser considerados modelos de aprendizado semi-paramétricos, pois estão simultaneamente presentes estruturas paramétricas e não-paramétricas. Quando se define *a priori* a arquitetura da rede neural (neurônios, conexões e funções de atuação), predominam estruturas paramétricas, enquanto que há um domínio de estruturas não-paramétricas quando vários aspectos da arquitetura são definidos durante o processo de treinamento (CAPOBIANCO, 1996).

De fato, uma das questões de mais difícil solução é a determinação da arquitetura mais indicada para a rede neural, visando a solução de um dado problema. Mesmo restrito ao contexto de aprendizado supervisionado, a literatura identifica um grande número de diferentes propostas de arquiteturas para as RNAs, algumas inspiradas em estruturas e funcionalidades do cérebro e outras motivadas por propriedades matemáticas e de aproximação de funções (ARBIB, 1995).

A busca por qual arquitetura fornecerá melhores resultados tem sido tratada na literatura como um processo não-sistemático, baseado no teste de diferentes configurações de rede, e necessariamente dependente das propriedades do problema a ser resolvido, assim como das mais diferentes restrições de projeto. Estas restrições freqüentemente incluem limitações de poder de processamento e espaço disponível de memória para se implementar a RNA, principalmente quando se empregam sistemas embarcados (GONÇALVES *et al*., 1998), tornando arquiteturas parcimoniosas altamente desejadas.

Mesmo nos restringindo a arquiteturas tradicionais e de capacidade de aproximação universal comprovada, como redes *feedforward* com uma camada intermediária de neurônios (HORNIK *et al*., 1989; HORNIK *et al*., 1994), as quais são casos particulares de

perceptrons de múltiplas camadas (MLP, do inglês *Multilayer Perceptron*), ainda permanece o problema não-trivial de determinar o número de neurônios da camada intermediária. Caso se queira ir ainda mais além, cabe definir as funções de ativação apropriadas para neurônios da camada intermediária e de saída.

O problema da determinação do número de neurônios de uma camada intermediária (ou mesmo do número de camadas) é ainda agravado pelo fato do erro de aproximação de RNAs treinadas por algoritmos de gradiente descendente, os quais empregam a técnica de retropropagação (*backpropagation*, na literatura em inglês) (CHAUVIN & RUMELHART, 1995), não necessariamente cair monotonicamente com o aumento da complexidade da rede (GONÇALVES *et al*., 1998). Podem ocorrer casos em que RNAs mais flexíveis (e, por isso, em tese mais capazes de aproximar uma função arbitrária desconhecida) apresentarem, após treinadas, erros quadráticos maiores que os de RNAs menos flexíveis. Isto normalmente se deve à interferência de mínimos locais durante o treinamento. Redes excessivamente flexíveis também facilitam a ocorrência de sobre-ajuste (*overfitting*, na literatura em inglês) junto aos dados amostrais, resultando em baixa capacidade de generalização, embora este efeito possa ser reduzido com o emprego de técnicas que interrompam o treinamento antes que o sobre-ajuste torne-se significativo (DE CASTRO, 1999; LOUGHREY & CUNNINGHAM, 2004, 2005; REUNANEN, 2003).

Neste contexto, a utilização de técnicas e algoritmos que determinem automaticamente a arquitetura de rede utilizada torna-se bastante desejável, motivando o estudo dos métodos construtivos de treinamento de RNAs (KWOK & YEUNG, 1996, 1997; DE CASTRO *et al.*, 1999).

Neste trabalho, focamos nossa atenção no Aprendizado por Busca de Projeção (PPL, do inglês *Projection Pursuit Learning*) (HWANG *et al*., 1994; VON ZUBEN & NETTO, 1995, 1997; VON ZUBEN, 1996; KWOK & YEUNG, 1996, 1997; DE CASTRO *et al.*, 1999; GONÇALVES *et al.*, 1998; MELEIRO *et al*., 2008), um algoritmo construtivo de treinamento de RNAs que tem demonstrado grande flexibilidade e efetividade na obtenção de RNAs de boa acurácia e capacidade de generalização, além de manter uma estrutura de rede bastante parcimoniosa. Embora as arquiteturas resultantes estejam restritas a uma camada intermediária, o número de neurônios dessa camada intermediária, o formato da função de

ativação de cada um desses neurônios e os pesos das conexões sinápticas devem ser definidos durante o aprendizado.

A expressão Aprendizado por Busca de Projeção se sustenta no fato de que os pesos sinápticos entre as entradas e cada neurônio da camada intermediária podem ser interpretados como constituintes de um vetor de projeção, visto que ocorre um produto interno entre o vetor de estímulos de entrada e esse vetor de pesos. Sob esta interpretação, encontrar esses pesos sinápticos equivale a identificar uma direção de projeção "interessante" para o vetor de estímulos de entrada (HUBER, 1985).

A existência de tamanha flexibilidade para a rede neural do PPL traz algumas desvantagens, dentre as quais destaca-se o fato de o PPL ter sua eficiência perceptivelmente afetada pela presença de um grande número de variáveis (estímulos) de entrada, comum em muitas aplicações. Quando em grande número, as variáveis de entrada não são, necessariamente, todas relevantes ao problema em questão, podendo, até mesmo, algumas serem redundantes.

Além disso, a implementação do PPL previamente disponível para o grupo de pesquisa encontrava-se "datada", sendo passível de melhorias e atualizações, principalmente levando em consideração os recursos computacionais atualmente disponíveis para o grupo, que permitem a utilização de ferramentas mais ambiciosas em termos de custo de processamento e de memória.

1.1 Objetivos

Os estudos, implementações e simulações realizados durante o desenvolvimento da pesquisa tiveram por objetivo principal:

- a familiarização com os mecanismos envolvidos no aprendizado construtivo, bem como a definição do escopo de aplicação dos mesmos;
- implementação de melhorias junto ao algoritmo de Aprendizado por Busca de Projeção (*Projection Pursuit Learning*), visando obter um código mais otimizado e uma maior flexibilidade da ferramenta computacional resultante, a fim de fornecer

ao grupo de pesquisa uma ferramenta prática e atualizada para a utilização em futuros experimentos;

- realização de aplicações práticas e comparações de desempenho com abordagens alternativas presentes na literatura, visando melhor situar o PPL frente a modelos alternativos.

Conforme será apresentado nos capítulos seguintes, estes três objetivos foram cumpridos com êxito, e os resultados obtidos ainda apontam várias perspectivas futuras da pesquisa envolvendo o PPL.

1.2 Organização do texto

O texto desta dissertação encontra-se organizado da seguinte forma:

No Capítulo 2, o aprendizado construtivo e o algoritmo de Aprendizado por Busca de Projeção são apresentados em detalhe, incluindo uma breve discussão sobre a base estatística da busca de projeção (o algoritmo de Regressão por Busca de Projeção – PPR) e sobre as limitações do PPL.

O Capítulo 3 discute conceitos vinculados à pesquisa em Seleção de Variáveis, cujas técnicas serão empregadas no tratamento de limitações do PPL. Também inclui uma breve explanação sobre algoritmos genéticos e seu emprego na metodologia *wrapper* de seleção.

As contribuições implementadas para tratar as limitações do algoritmo de PPL são relatadas no Capítulo 4, e o Capítulo 5 apresenta os resultados e comparações experimentais. Já o Capítulo 6 apresenta as conclusões e considerações finais, bem como as perspectivas futuras do trabalho.

Por fim, o Apêndice A apresenta os conjuntos de dados utilizados nos experimentos e simulações do Capítulo 5, e o Apêndice B introduz o algoritmo SAND (*Simulated ANnealing for Diversity*, na literatura em inglês).

Capítulo 2

Aprendizado por Busca de Projeção

2.1 Aprendizado construtivo

É no contexto da necessidade de determinar a melhor arquitetura neural que a classe de algoritmos de treinamento construtivo (KWOK & YEUNG, 1996, 1997; DE CASTRO *et al.*, 1999) apresenta vantagens consideráveis: ao testar se a modificação da estrutura da rede, geralmente de natureza incremental, resultará em melhoria na capacidade de aproximação, os métodos construtivos estabelecem que o aumento da complexidade da RNA não causa perda de qualidade do resultado final, promovendo a monotonicidade da relação entre número de neurônios e queda do erro de treinamento (GONÇALVES *et al*., 1998).

O tipo de arquitetura de rede apresentada no final do treinamento depende do método construtivo específico, mas, em geral, a RNA é inicializada e treinada com a configuração neural mais simples possível (quase sempre um único neurônio na única camada intermediária). Caso a rede resultante não satisfaça os requisitos de aproximação e generalização desejados, aumenta-se a complexidade da arquitetura (normalmente introduzindo-se mais um neurônio na camada intermediária, embora adicionar conexões entre neurônios já existentes também possa ser feito) e retreina-se a rede. O processo continua, aumentando-se incrementalmente a complexidade da RNA e retreinando-a até que os requisitos sejam alcançados, ou se torne impossível ou muito custoso melhorar a resposta da rede, ou esta extrapole os limites de complexidade aceitáveis. Em alguns métodos, toda a rede é retreinada após a introdução de um novo elemento na estrutura. Em outros, apenas os novos componentes sofrem treinamento.

Os métodos construtivos também apresentam vantagens quando comparados a outra classe de métodos de determinação automática de arquitetura: os chamados métodos de poda (*pruning methods* na literatura em língua inglesa) (RUSSELL, 1993; KWOK & YEUNG, 1996). Nos métodos de poda, a RNA é inicializada com um número elevado de neurônios,

os quais vão sendo aos poucos removidos (podados) da arquitetura, na medida em que apresentarem baixa participação na resposta da rede.

Vantagens dos métodos construtivos sobre os de poda incluem:

- métodos construtivos trabalharem desde o início com redes mais simples, reduzindo o custo computacional de treinamentos e re-treinamentos;

- eles possuem um ponto de início claro e não arbitrário (um neurônio na camada intermediária), enquanto os de poda requerem que um valor elevado para o número inicial de neurônios seja arbitrariamente determinado;

- são mais propensos a fornecerem arquiteturas parcimoniosas, pois ao se inicializar a RNA com a menor dimensão possível e ir gradualmente aumentando o número de neurônios conforme necessário, tende-se a obter RNAs com o menor número possível de neurônios que satisfaça os critérios de acurácia e de capacidade de generalização estabelecidos, enquanto os métodos de poda, sendo inicializados com um valor elevado de neurônios intermediários, não podem garantir que a arquitetura final seja a menor possível, uma vez que mínimos locais podem interromper o processo de poda prematuramente.

2.2 Aprendizado por busca de projeção

Aprendizado por Busca de Projeção (do inglês *Projection Pursuit Learning*), ou PPL, é um algoritmo construtivo de treinamento de RNAs: a partir de um único neurônio artificial na camada intermediária, a rede vai sendo treinada pelo ajuste dos pesos sinápticos da camada intermediária e de saída, pela determinação de funções de ativação apropriadas para cada neurônio da camada intermediária, bem como pela eventual introdução de novos neurônios artificiais nesta única camada intermediária.

O resultado é uma RNA do tipo *feedforward* com uma arquitetura semelhante à de um *perceptron* multicamadas (MLP, do inglês *Multi-Layer Perceptron*) (HORNIK *et al.*, 1989, 1994; CHAUVIN & RUMELHART, 1995): formada por uma camada de neurônios de

entrada, uma camada intermediária, e uma de saída; porém com funções de ativação específicas para cada neurônio da camada intermediária e definição automática do número de neurônios nesta camada, o qual tende a ser comparativamente menor que aquele necessário para uma rede MLP (HWANG, *et al.*, 1994; VON ZUBEN, 1996; GONÇALVES *et al.*, 1998).

A metodologia empregada pelo PPL é derivada de uma técnica estatística de regressão conhecida por *Projection Pursuit Regression* (PPR), que por sua vez é uma aplicação específica em Busca de Projeção (*Projection Pursuit*- PP), e une as vantagens do aprendizado construtivo (parcimônia nas estruturas neurais) com as das técnicas de PP (vulnerabilidade reduzida à "maldição da dimensionalidade" e interpretabilidade dos resultados) (FRIEDMAN & STUETZLE, 1981; HUBER, 1985).

2.3 Busca de projeção

As técnicas de PP começaram a ser formalizadas entre meados dos anos 1970 e 1980 (HUBER, 1985), embora alguns trabalhos anteriores já se utilizassem de conceitos que viriam a ser considerados centrais à área.

O objetivo inicial das técnicas de PP foi servir de auxílio, ou substituto, à observação visual humana na busca por estruturas e relações em conjuntos de dados estatísticos (*Exploratory Data Analysis* na literatura em inglês) (FRIEDMAN & TUKEY, 1974; HUBER, 1985). Este auxílio é particularmente necessário no caso de conjuntos de dados com mais de três dimensões, que são impossíveis de serem representados graficamente de forma direta. E mesmo no caso de dados tridimensionais, pode-se estar limitado a representações gráficas em duas dimensões, as quais podem obscurecer ou ressaltar determinadas estruturas dependendo da projeção utilizada (HUBER, 1985).

Desta forma, PP consiste na automatização de um processo de redução da dimensionalidade dos dados que busca ressaltar estruturas presentes nos mesmos, principalmente as potencialmente "interessantes". O algoritmo é análogo à tarefa realizada por um pesquisador humano que rotaciona os eixos de um gráfico tridimensional projetado em uma tela de computador, buscando obter uma melhor visualização dos dados. O gráfico

observado é uma projeção linear bidimensional e o pesquisador ajusta a projeção na medida em que esta lhe revela ou não uma informação de interesse.

Expressamos uma projeção linear de $\Re^m$ para $\Re^r$ de uma variável aleatória $\boldsymbol{X}$, na forma:

$$\boldsymbol{Z} = \boldsymbol{A}\boldsymbol{X}, \quad \boldsymbol{X} \in \Re^m, \ \boldsymbol{Z} \in \Re^r, \ \boldsymbol{A}_{r\times m}, \ \text{rank}(\boldsymbol{A}) = r, \ \text{e } 0 < r < m. \tag{2.1}$$

Definimos PP como a busca por um mapeamento linear, representado pela matriz $\boldsymbol{A}^*$, que maximize (ou minimize) uma determinada função objetivo $I(\boldsymbol{Z})$. Esta função é normalmente denominada Índice de Projeção.

Para que a projeção produzida pelo mapeamento linear seja "útil", é necessário que o Índice de Projeção forneça uma medida de quão "interessante" é uma dada projeção. Os primeiros índices foram determinados heuristicamente, mas passaram a preponderar índices baseados em medidas estatísticas, como as de desvio padrão ou de entropia relativa (HUBER, 1985).

A projeção buscada pode ser de qualquer número de dimensões, desde que $r < m$, mas, por motivos de visualização gráfica, dificilmente lida-se com $r > 3$; em muitas das aplicações de PP (incluindo PPL), lida-se apenas com a projeções unidimensionais ($r = 1$), o que reduz a expressão (2.1) a:

$$\boldsymbol{Z} = \boldsymbol{a}^T \boldsymbol{X}, \ \boldsymbol{X} \in \Re^m, \ \boldsymbol{Z} \in \Re, \ \boldsymbol{a} \in \Re^m. \tag{2.2}$$

Uma vez encontrada uma ou mais projeções interessantes através de PP, o próximo passo dependerá do objetivo da análise: identificar, localizar ou isolar *clusters*, ou obter uma descrição parcimoniosa dos dados, dentre outras possibilidades.

2.4 Regressão por busca de projeção

Em problemas de regressão, é dado um par de variáveis aleatórias $\{\boldsymbol{X},\boldsymbol{Y}\}$, $\boldsymbol{X} \in \Re^m$, $\boldsymbol{Y} \in \Re$, e temos que estimar a esperança condicional de $\boldsymbol{Y}$ dado $\boldsymbol{X}$ com base em um conjunto amostral $\{(\boldsymbol{x}_i,y_i): i = 1, 2, ..., N\}$, ou seja:

$$f(\boldsymbol{x}) = E(\boldsymbol{Y} \mid \boldsymbol{X} = \boldsymbol{x}). \tag{2.3}$$

Caso seja considerada uma forma funcional específica para a superfície de regressão $f(\bullet)$, é possível utilizar um método paramétrico adequado e reduzir significativamente a dificuldade do problema. No entanto, situações nas quais isto é possível com graus de certeza razoáveis são bastante raras, e erros na determinação do modelo podem levar a conclusões equivocadas.

Métodos não-paramétricos, por não suporem formas funcionais específicas, embora possam incluir algum grau de restrição sobre a superfície de regressão (razão pela qual eles também recebem a denominação de modelos semi-paramétricos), são menos propensos a polarização no processo de modelagem. No entanto, são computacionalmente mais custosos e, justamente por fazerem menos suposições, necessitam extrair dos dados amostrais uma quantidade maior de informação.

Isto faz com que eles se tornem particularmente vulneráveis ao que se convencionou chamar de "maldição da dimensionalidade": na medida em que o número de dimensões aumenta, a quantidade de dados amostrais necessários para se manter a densidade de informação cresce exponencialmente. Com isso, para grande número de dimensões, ou o conjunto de dados se torna intratavelmente grande, ou somos forçados a lidar com conjuntos que trazem uma quantidade de informação reduzida (FRIEDMAN & STUETZLE, 1981; HUBER, 1985), voltando a ter dificuldades com polarizações no processo de modelagem.

Os métodos de regressão não-paramétricos mais bem sucedidos baseiam-se no refinamento sucessivo do modelo gerado para descrever a superfície de regressão. Modelos cada vez mais complexos são construídos a partir do modelo anterior de forma a obter-se uma melhor aproximação. Devido à "maldição da dimensionalidade", estes métodos serão bem sucedidos na medida em que a superfície de regressão possa ser aproximada por modelos com baixa complexidade (uma vez que a informação disponível pode ser insuficiente para selecionar adequadamente modelos mais complexos).

A qualidade dos dados amostrais não se relaciona apenas com a dimensão do espaço (número de entradas), mas também com o nível de ruído presente no processo de amostragem, o que aumenta os desafios de uma regressão não-paramétrica.

Uma forma de lidar com a esparsidade inerente do espaço multidimensional é trabalhar em projeções de dimensões reduzidas deste espaço, preferencialmente projeções unidimensionais. Assim, em PPR (proposta originalmente apresentada em FRIEDMAN & STUETZLE, 1981), a função $y = f(\boldsymbol{x})$ é aproximada por uma soma de funções de expansão ortogonal, também conhecidas como funções telha (do inglês *ridge functions*), denotadas por g_j, j=1, ...,n:

$$f(\boldsymbol{x}) \approx \sum_{j=1}^{n} g_j\left(\boldsymbol{a}_j^T \boldsymbol{x}\right) \tag{2.4}$$

A função unidimensional g_j é uma representação suave (do inglês *smoother*) da projeção utilizada, e os vetores de projeção $\boldsymbol{a}_j$ são encontrados de forma a minimizar a soma dos quadrados dos resíduos da aproximação:

$$\boldsymbol{a}_j = \arg\min_{\boldsymbol{a}_j} \sum_{i=1}^{N} \left(y_i - g_j(\boldsymbol{a}_j^T \boldsymbol{x}_i)\right)^2 \tag{2.5}$$

Podemos dizer então que $I(\boldsymbol{a}_j) = \sum_{i=1}^{N} \left(y_i - g_j(\boldsymbol{a}_j^T \boldsymbol{x}_i)\right)^2$ é o Índice de Projeção utilizado no PPR.

Nota-se que funções unidimensionais da forma $g(z)$, $z \in \Re$, quando têm seu argumento na forma de um produto escalar, ou seja, $z = \boldsymbol{a}^T\boldsymbol{x}$, $\boldsymbol{a} \in \Re^m$ e $\boldsymbol{x} \in \Re^m$, variam em uma única direção (aquela definida pelo vetor de projeção $\boldsymbol{a}$), mas apresentam valor constante nas $m - 1$ direções ortogonais a esta. Por isto são denominadas funções de expansão ortogonal.

À medida que uma projeção e um modelo suave para esta são encontrados, a contribuição destes para a aproximação de $f(\boldsymbol{x})$ é removida do modelo, de forma que o próximo passo $n + 1$ explorará apenas as estruturas ainda não representadas. Os resíduos $\boldsymbol{d}_j \in \Re^N$ ($j = 1, ..., n$) são dados por $\boldsymbol{d}_j = [d_{j1}\, d_{j2} \ldots d_{jN}]^T$, onde:

$$d_{ji} = y_i - \sum_{\substack{k=1 \\ k \neq j}}^{n} g_k(\boldsymbol{a}_k^T \boldsymbol{x}_i), i = 1, ..., N \tag{2.6}$$

Novos passos são executados até que o grau de aproximação desejado seja atingido, ou então não seja mais possível refinar o modelo.

Assim, PPR consiste em uma técnica de regressão não-paramétrica que modela uma superfície de regressão como a soma de funções suaves de expansão ortogonal ao vetor de projeção correspondente.

Apesar de a lógica geral do algoritmo ser simples, sua implementação requer vários cuidados, principalmente no método utilizado para se obter as funções de expansão ortogonal. Sobre-ajuste em uma das primeiras funções unidimensionais pode invalidar aproximações seguintes, prejudicando o modelo de regressão como um todo (HUBER, 1985).

Algumas referências sobre PPR também conceituam uma fase de retro-ajuste (*backfitting*) que antecede a busca de um novo termo da aproximação (FRIEDMAN & STUETZLE, 1981; HUBER, 1985). Durante a fase de *backfitting*, tanto a função de expansão ortogonal quanto a projeção dos termos previamente obtidos são reajustadas, levando agora em consideração a mais recente projeção encontrada. Isto aumenta o custo computacional do algoritmo, e, por isso, não costumava ser implementada, mas permite a obtenção de uma melhor aproximação, uma vez que combinações ótimas para m termos não continuam necessariamente ótimas para $m + 1$ termos.

Podemos perceber esta capacidade de melhoria se atentarmos para o fato de que, ao introduzirmos um novo termo no somatório de funções de expansão ortogonal, podemos atualizar o resíduo utilizado para definir o Índice de Projeção de um termo anterior incluindo a participação da nova função. Pode, portanto, ser possível encontrar um novo par $(\boldsymbol{a}_j^*, g_j^*)$ diferente do atual e de melhor Índice de Projeção.

2.5 PPR *vs.* PPL

Uma RNA tipo *feedforward*, com uma única camada intermediária e função de ativação de saída linear, aproxima uma função desconhecida através da composição aditiva das funções de ativação dos neurônios artificiais. Esta composição é expressa na forma:

$$\hat{y}_k = w_{0k} + \sum_{j=1}^{n}\left[w_{jk}\, f_j\left(v_{oj} + \sum_{l=1}^{m} v_{lj}\, x_l \right)\right], \tag{2.7}$$

onde $\hat{y}_k$ é a estimativa do k-ésimo componente do vetor de saída $\boldsymbol{y} = [y_1 \ldots y_r]^T$; w_{ok} é a polarização do k-ésimo neurônio de saída; n é o número de neurônios na camada intermediária; w_{jk} é o peso sináptico conectando o j-ésimo neurônio da camada intermediária ao k-ésimo neurônio de saída; f_j: $\Re \rightarrow \Re$ é a função de ativação do j-ésimo neurônio da camada intermediária; m é o número de entradas da rede; v_{0j} é a polarização do j-ésimo neurônio da camada intermediária; v_{lj} é o peso sináptico conectando a l-ésima entrada ao j-ésimo neurônio da camada intermediária; e x_l é a l-ésima entrada da rede.

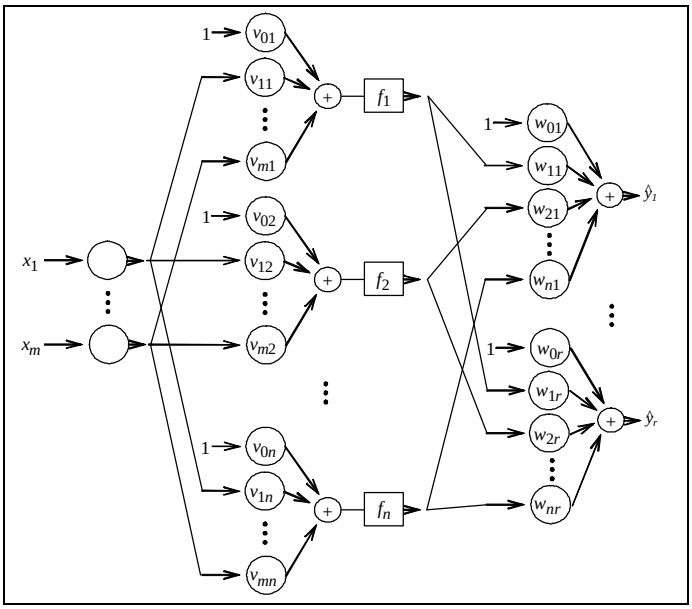

Fig. 2.1 – Rede neural artificial tipo *feedforward* com uma camada intermediária

Comparando (2.7) com (2.4), percebemos que o algoritmo de PPR é análogo a uma RNA sem entradas de polarização e pesos sinápticos de saída iguais a um. Cada vetor de projeção linear $\boldsymbol{a}_j$ equivale ao vetor $\boldsymbol{v}_j = [v_{1j} \ldots v_{mj}]^T$ dos pesos sinápticos da camada intermediária e cada função de expansão ortogonal g_j a uma função de ativação f_j.

Assim, é possível nos aproveitarmos das técnicas de PPR para a síntese construtiva de RNAs com funções de ativação independentes para cada neurônio da camada intermediária.

De fato, há apenas três diferenças significativas entre os modelos gerados por técnicas de PPR e pelo algoritmo de PPL: no PPL há adição ponderada da participação de cada projeção na aproximação da função desejada; tem-se a presença de entradas de polarização; e existe a possibilidade de múltiplas saídas ($r > 1$).

2.6 O algoritmo de aprendizado por busca de projeção

No aprendizado por busca de projeção (PPL, do inglês *Projection Pursuit* Learning), a RNA é inicializada com um único neurônio na camada intermediária e todo o processo de treinamento se baseia em amostras de entrada-saída na forma: $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{N} \equiv (\boldsymbol{X}, \boldsymbol{Y})$. Este neurônio é treinado juntamente com os neurônios de saída, produzindo a função de ativação (f_1), os pesos sinápticos intermediários ($\boldsymbol{v}_1$), bem como os pesos de saída ($\boldsymbol{w}_{(1)}$). A notação $\boldsymbol{w}_{(j)}$ indica o vetor-linha formado pelos pesos sinápticos de saída associados ao neurônio da camada intermediária j, ou seja $\boldsymbol{w}_{(j)} = [w_{j0}, w_{j1}, ..., w_{jn}]$. Caso a aproximação das saídas desejadas $\boldsymbol{y}_i$. $i = 1, ...,N$, ainda seja insatisfatória, um novo neurônio é introduzido na camada intermediária, sendo que os pesos correspondentes e sua função de ativação são definidos da mesma forma descrita acima, fixando-se todos os parâmetros associados ao neurônio 1.

Após um novo neurônio ser adicionado e treinado, realiza-se uma fase de retro-ajuste (*backfitting*), na qual todos os neurônios da rede são retreinados em seqüência e ciclicamente. Isto é feito porque o algoritmo de treinamento dos neurônios conduz a valores adequados para f_j, $\boldsymbol{v}_j$ e $\boldsymbol{w}_{(j)}$ para uma configuração específica de rede, e, uma vez que esta configuração é alterada pela inclusão de mais um neurônio, freqüentemente torna-se possível buscar melhores valores para os parâmetros já obtidos (VON ZUBEN & NETTO, 1997).

Caso, após o retro-ajuste, a aproximação ainda esteja insatisfatória, um novo neurônio é adicionado, iniciando um novo ciclo de treinamento e retro-ajuste. O processo continua até o erro de aproximação da rede atingir valores aceitáveis, não ser mais possível obter melhoras significativas no erro com a adição de um novo neurônio, ou a rede atingir um número máximo de neurônios previamente estabelecido.

2.6.1 O PPL passo-a-passo

Nas versões iniciais do PPL, todos os parâmetros da rede associados a um dado neurônio (f_j, $\boldsymbol{v}_j$ e $\boldsymbol{w}_{(j)}$) eram determinados iterativamente. A cada ciclo, um parâmetro tinha seu valor atualizado por vez, fixando-se os outros dois parâmetros enquanto o primeiro era otimizado. O valor atualizado deste parâmetro era então fixado e utilizado na fase de otimização do próximo parâmetro. O ciclo seguia até os valores dos três parâmetros convergirem (FRIEDMAN & STUETZLE, 1981; HWANG, *et al.*, 1994).

No entanto, VON ZUBEN & NETTO (1997) demonstraram que é possível utilizar propriedades algébricas para obter uma solução fechada para $\boldsymbol{w}_{(j)}$, de forma que apenas f_j e $\boldsymbol{v}_j$ sejam obtidos de forma iterativa, ocorrendo em seguida o cálculo de $\boldsymbol{w}_{(j)}$.

O procedimento de treinamento de cada neurônio adicionado, seguindo a proposta de por VON ZUBEN & NETTO (1997), dá-se da seguinte forma:

Sejam $\sigma_{i\,0} = 1$, $i = 1, ..., N$, a entrada de polarização do neurônio de saída l;

$\sigma_{i\,j} = f_j(\boldsymbol{v}_j^T \mathbf{x}_i)$, $j = 1, ..., n$ e $i = 1, ..., N$, o sinal j de entrada para o neurônio de saída l;

$\boldsymbol{\sigma}_0$ e $\boldsymbol{\sigma}_j$, $j = 1, ..., n$, os vetores coluna definidos pelos elementos $\sigma_{i\,0}$ e $\sigma_{i\,j}$;

$\boldsymbol{Y}_{N\times r}$, a matriz contendo os valores das r saídas e N instâncias amostrais;

$\boldsymbol{D}_n = \boldsymbol{Y} - \sum_{j=0}^{n-1} \boldsymbol{\sigma}_j \boldsymbol{w}_{(j)}$, $n > 0$, a matriz do resíduo de uma rede com $n - 1$ neurônios; e

$\boldsymbol{u}_n$ o componente principal da matriz de correlação $\mathbf{D}_n\mathbf{D}_n^T$.

Um neurônio n tem os valores ótimos seus pesos de entrada ($\boldsymbol{v}_n^*$) e função de ativação (f_n^*) dados por um processo iterativo: começando por um $\boldsymbol{v}_n$ fixo, acha-se um f_n ótimo, cujo valor é fixado e utilizado para se obter um valor ótimo para $\boldsymbol{v}_n$. O processo segue até haver convergência, a qual é garantida pelos resultados em JONES (1987).

A inicialização de $\boldsymbol{v}_n$ pode seguir várias estratégias, mas, em geral, criam-se vários vetores candidatos à direção de projeção inicial e testam-se os mesmos, utilizando-se o índice mais promissor.

Já f_n pode ser determinada utilizando-se um entre vários métodos de regressão unidimensional (desde que se disponha de mecanismos para controlar a suavidade da função). Dentro da literatura de PPL, tanto métodos paramétricos (como as funções de Hermite) (HWANG, *et al.*, 1994; MELEIRO *et al.*, 2008) quanto não-paramétricos (como *splines*) (VON ZUBEN & NETTO, 1997) são utilizados, e f_n^* é dada pela expressão:

$$f_n^* = \arg\min_{f_n} \sum_{i=1}^{N} \left(f_n(\boldsymbol{v}_n^T \boldsymbol{x}_i) - u_{ni} \right)^2 \tag{2.8}$$

onde $\boldsymbol{x}_i \in \Re^{(m+1)}$ é o vetor coluna correspondente à *i*-ésima linha de $\boldsymbol{X}$ acrescido da entrada de polarização, e u_{ni} é o *i*-ésimo componente de $\boldsymbol{u}_n$.

O novo valor de $\boldsymbol{v}_n$ é dado pela solução do problema:

$$\boldsymbol{v}_n^* = \arg\max_{\boldsymbol{v}_n} \frac{(\boldsymbol{\sigma}_n^T \boldsymbol{u}_n)^2}{\boldsymbol{\sigma}_n^T \boldsymbol{\sigma}_n} \tag{2.9}$$

Após a convergência de um processo cíclico envolvendo cálculos seqüenciais de f_n^* e $\boldsymbol{v}_n^*$, obtém-se:

$$\boldsymbol{\sigma}_n^* = f_n^* \left(\boldsymbol{v}_n^{*T} \boldsymbol{x} \right) \tag{2.10}$$

Em seguida, calcula-se $\boldsymbol{w}_{(n)}^*$ pela expressão:

$$\boldsymbol{w}_{(n)}^* = \left(\boldsymbol{\sigma}_n^{*T} \boldsymbol{\sigma}_n^* \right)^{-1} \boldsymbol{\sigma}_n^{*T} \boldsymbol{D}_n, \quad n > 0 \tag{2.11}$$

E $\boldsymbol{w}_{(0)}^*$ é dado por:

$$\boldsymbol{w}_{(0)}^* = \min_{\boldsymbol{w}_{(0)}} \left\| \boldsymbol{\sigma}_0 \boldsymbol{w}_{(0)} - \boldsymbol{Y} \right\| \tag{2.12}$$

2.6.2 Obtenção da função de ativação unidimensional

Para se determinar a forma de f_j, foi utilizado neste trabalho um modelo baseado na combinação linear de funções ortonormais de Hermite (HWANG, *et al.*, 1994; MELEIRO *et al.*, 2008). A escolha destas funções se deve à facilidade de se calcular os valores e as derivadas dos polinômios de Hermite (HWANG *et al.*, 1994).

Dado o conjunto de dados amostrais $\{(z_i, u_{ji})\}_{i=1}^{N}$, onde z_i é a projeção da *i*-ésima instância das entradas ($\boldsymbol{x}_i$) na direção $\boldsymbol{v}_j$ e u_{ji} é a saída desejada do neurônio intermediário *j* para o vetor de entrada $\boldsymbol{x}_i$, f_j pode ser expressa como:

$$f_j(z_i) = \sum_{p=0}^{P} c_{jp} h_p(z_i) \tag{2.13}$$

onde *P* é a ordem máxima das funções de Hermite utilizadas, c_{jp} são os coeficientes de regressão, e $h_p(z)$ são as funções ortonormais de Hermite, as quais são dadas pela expressão abaixo:

$$h_p(z_i) = \frac{1}{\sqrt{p!2^p}} H_p(z_i)\phi(z_i), \quad -\infty < z_i < \infty, \tag{2.14}$$

Sendo a função de ponderação (*weight function*) $\phi(z_i)$ dada por:

$$\phi(z_i) = \pi^{-(1/4)} e^{-(z_i/2)}, \quad -\infty < z_i < \infty, \tag{2.15}$$

e os polinômios de Hermite $H_p(z_i)$ sendo determinados recursivamente da seguinte forma:

$$\begin{aligned} &H_0(z_i) = 1, \\ &H_1(z_i) = 2z_i, \\ &H_{p+1}(z_i) = 2\big(z_i H_p(z_i) - pH_{p-1}(z_i)\big), \quad p \geq 1 \end{aligned} \tag{2.16}$$

Definindo:

$$\boldsymbol{H} = \begin{bmatrix} h_0(z_1) & h_1(z_1) & \cdots & h_p(z_1) \\ h_0(z_2) & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ h_0(z_N) & h_1(z_N) & \cdots & h_p(z_N) \end{bmatrix}, \; \boldsymbol{c}_j = \begin{bmatrix} c_{j1} \\ c_{j2} \\ \vdots \\ c_{jp} \end{bmatrix} \text{ e } u_j = \begin{bmatrix} u_{j1} \\ u_{j2} \\ \vdots \\ u_{jN} \end{bmatrix}, \quad (2.17)$$

temos que a solução de quadrados mínimos para:

$$\boldsymbol{c}_j^* = \arg\min_{\boldsymbol{c}_j} \left\| u_j - \boldsymbol{H}\boldsymbol{c}_j \right\|, \quad (2.18)$$

produz, supondo $P \leq N$:

$$\boldsymbol{c}_j^* = \left(\boldsymbol{H}^T \boldsymbol{H}\right)^{-1} \boldsymbol{H}^T u_j \quad (2.19)$$

2.6.3 Obtenção das direções de projeção

Para se determinar uma direção ótima de projeção, usa-se a expressão (2.9). Uma vez que os valores dos polinômios de Hermite e de suas derivadas podem ser obtidos analiticamente, sem a necessidade de procedimentos numéricos, podemos calcular facilmente as derivadas de primeira e segunda ordem da função-objetivo em (2.9).

A derivada primeira de $H_p(z_i)$ com relação a z_i é:

$$\frac{dH_p(z_i)}{dz_i} = 2pH_{p-1}(z_i) \quad (2.20)$$

Já as derivadas de h_p e f_j com relação a z_i podem ser calculadas analiticamente (VON ZUBEN & NETTO, 1995), e são dadas por:

$$\frac{dh_p(z_i)}{dz_i} = \sqrt{2p}h_{p-1}(z_i) - z_i h_p(z_i) \quad (2.21)$$

$$\frac{d^2 h_p(z_i)}{dz_i^{\,2}} = 2\sqrt{p(p-1)}h_{p-2}(z_i) - 2\sqrt{2p}z_i h_{p-1}(z_i) + (z_i^2 - 1)h_p(z_i) \quad (2.22)$$

$$f_j'(z_i) = \frac{df_j(z_i)}{dz_i} = \sum_{p=1}^{P} c_p \frac{dh_p(z_i)}{dz_i} \tag{2.23}$$

$$f_j''(z_i) = \frac{d^2 f_j(z_i)}{dz_i^2} = \sum_{p=1}^{P} c_p \frac{d^2 h_p(z_i)}{dz_i^2} \tag{2.24}$$

E com estes termos, pode-se implementar o método de Newton (LUENBERGER, 2003) para a obtenção de $\boldsymbol{v}_j^*$.

2.6.4 Retro-Ajuste

No retro-ajuste, cada neurônio j (j = 1, ... ,n) tem, em seqüência e ciclicamente, seus parâmetros retreinados de forma a solucionar o problema de otimização

$$\min_{f_j, \boldsymbol{v}_j, \boldsymbol{w}_{(j)}} \left\| \boldsymbol{\sigma}_j \boldsymbol{w}_{(j)} - \boldsymbol{D}_n^{[j]} \right\|, \tag{2.25}$$

com,

$$\boldsymbol{D}_n^{[j]} = \boldsymbol{Y} - \sum_{\substack{i=0 \\ i \neq j}}^{n} \boldsymbol{\sigma}_i \boldsymbol{w}_{(i)} \tag{2.26}$$

O termo $\boldsymbol{D}_n^{[j]}$ corresponde ao resíduo da rede, excluindo-se a participação do neurônio j, e a convergência do processo é também garantida pelos resultados em JONES (1987).

2.7 Exemplo ilustrativo

Para ilustrar o funcionamento do algoritmo PPL, treinamos uma RNA para aproximar a função $f^{(5)}(x_1,x_2)$ (ver Apêndice A) apresentada abaixo (veja Figura 2.2):

$$f^{(5)}(x_1, x_2) = 1.9\left[1.35 + e^{x_1 - x_2} \operatorname{sen}\left(13(x_1 - 0.6)^2\right) \cdot \operatorname{sen}\left(7x_2\right)\right] \tag{2.27}$$

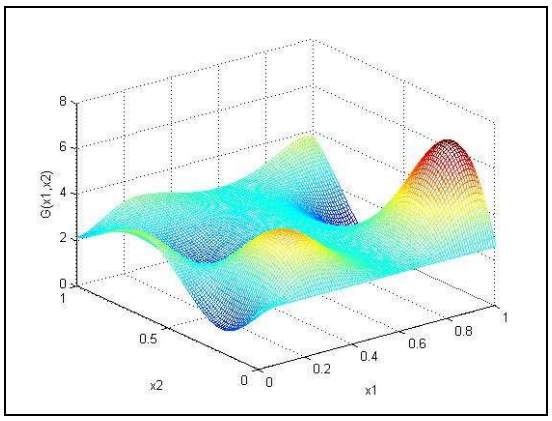

Figura 2.2 – Mapeamento $\Re^2 \rightarrow \Re$ a ser aproximado

Esta função foi escolhida como exemplo por ser muito utilizada na literatura e considerada bastante desafiadora para modelos de aproximação que realizam composição aditiva de funções-base, como é o caso no PPL. Por ser uma função de apenas duas variáveis, é possível representar graficamente as funções de expansão ortogonal produzidas pelos neurônios da camada intermediária.

As funções de ativação dos neurônios intermediários foram determinadas utilizando-se funções de Hermite com polinômios de ordem zero a onze. O número máximo de neurônios permitidos foi seis, e a função $f^{(5)}(x_1,x_2)$ foi amostrada gerando-se aleatoriamente 500 instâncias do par $(x_1,x_2) \in [0,1]^2$ e calculando-se o valor de $f^{(5)}$ para cada instância.

Nas Figuras 2.3 a 2.29, são apresentadas as funções de ativação e direções de projeção de cada neurônio após cada etapa de retro-ajuste. Nos gráficos das funções de ativação, os pontos em verde representam os resíduos de cada instância dos dados amostrais considerando a participação de todos os outros neurônios da rede. Tanto as funções como os resíduos são apresentados em função da projeção unidimensional determinada pelos pesos sinápticos do neurônio. Os valores presentes no topo dos gráficos das funções de ativação, ao lado do número do neurônio, representam cem vezes o valor do cosseno entre a resposta da rede e o vetor de resíduos usado na aproximação, ou seja, quanto mais distante de 100 for este valor, pior a aproximação.

Comparando as funções de ativação e resíduos de cada neurônio à medida que novos neurônios são introduzidos (do neurônio 1, por exemplo: Figuras 2.4, 2.6, 2.9, 2.13,

2.18 e 2.24), podemos observar não apenas a forma como a fase retro-ajuste afeta a função, mas também como o resíduo "visto" por um neurônio artificial é afetado pela introdução de novos neurônios. É justamente esta alteração no resíduo da rede que permite que a participação de cada neurônio seja aumentada pelo retro-ajuste, re-otimizando os seus parâmetros para o novo cenário.

Por fim, as formas das funções de projeção, escalonadas pelo peso de saída correspondente, em função das variáveis de entrada originais (x_1 e x_2) são mostradas nas Figuras 2.30 a 2.35.

É interessante notar que o quinto e o sexto neurônio apresentam participação bastante reduzida (veja Figuras 2.34 e 2.35), atuando quase que apenas nos "cantos" da aproximação de $f^{(5)}(x_1,x_2)$.

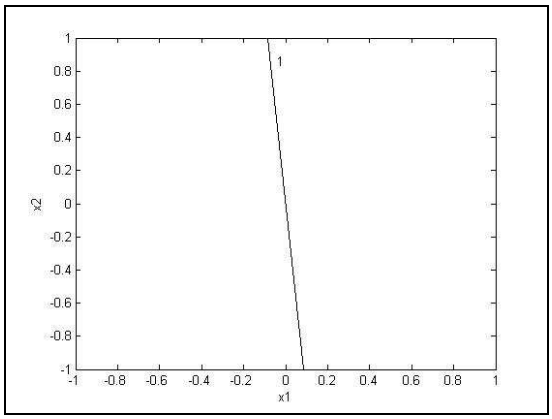

Figura 2.3 – Direção de projeção do primeiro neurônio

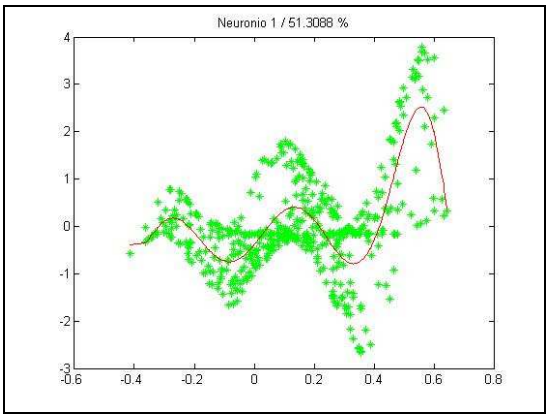

Figura 2.4 – Neurônio 1 após seu treinamento inicial

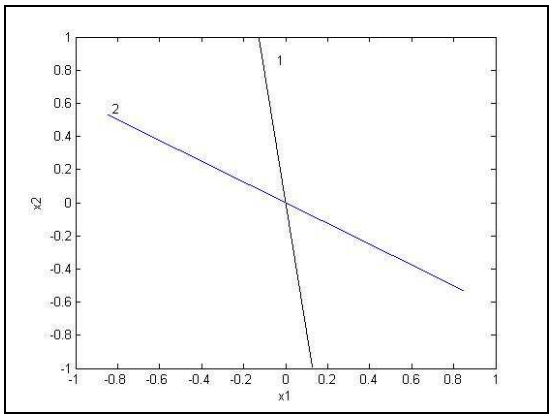

Figura 2.5 – Direções de projeção dos primeiros dois neurônios

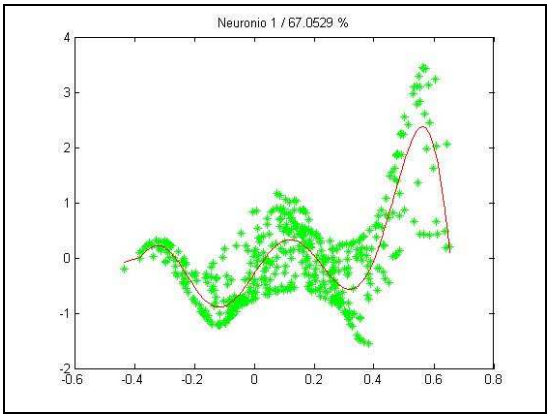

Figura 2.6 – Neurônio 1 após introdução do neurônio 2 e retro-ajuste

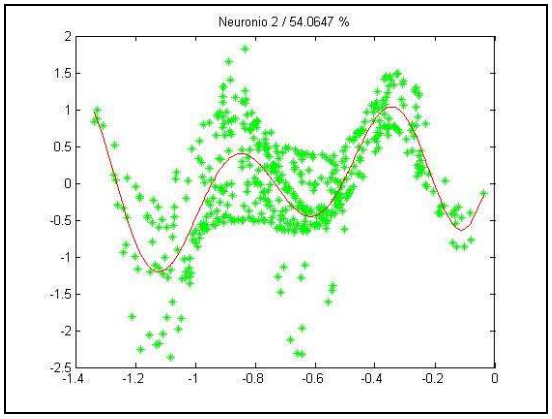

Figura 2.7 – Neurônio 2 após sua introdução e retro-ajuste

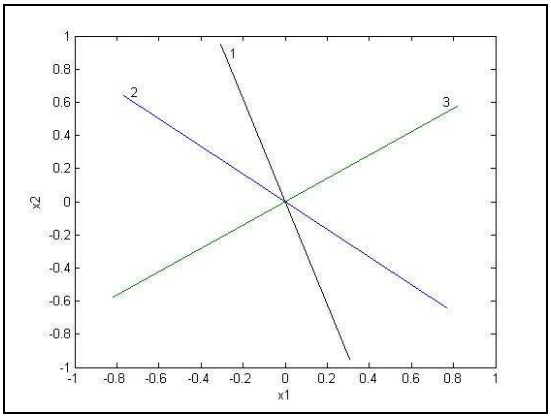

Figura 2.8 – Direções de projeção dos primeiros três neurônios

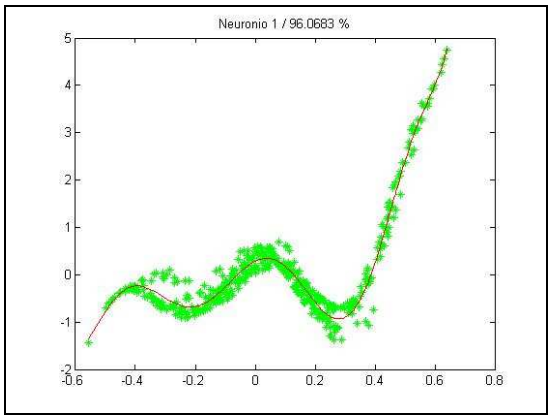

Figura 2.9 – Neurônio 1 após introdução do neurônio 3 e retro-ajuste

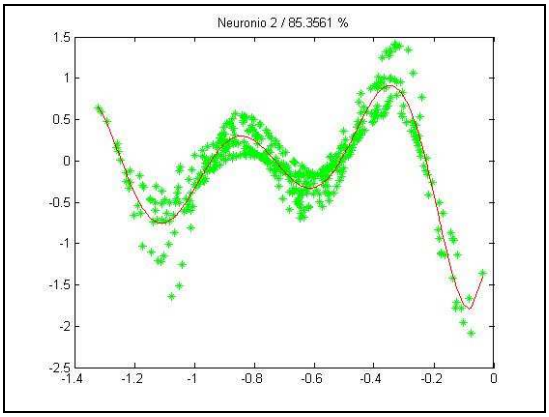

Figura 2.10 – Neurônio 2 após introdução do neurônio 3 e retro-ajuste

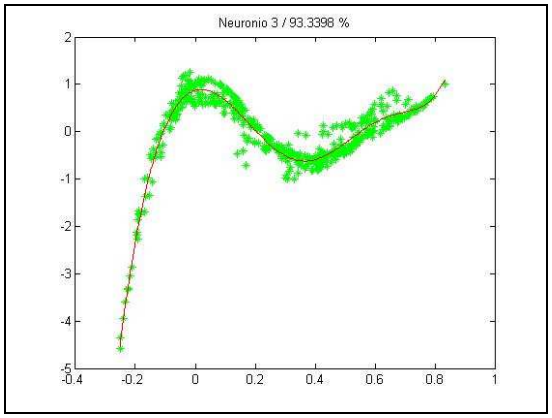

Figura 2.11 – Neurônio 3 após sua introdução e retro-ajuste

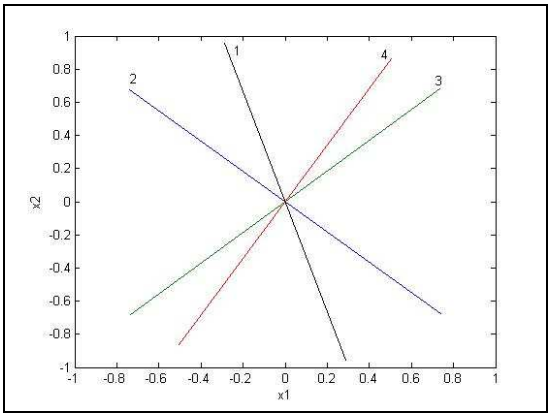

Figura 2.12 – Direções de projeção dos primeiros quatro neurônios

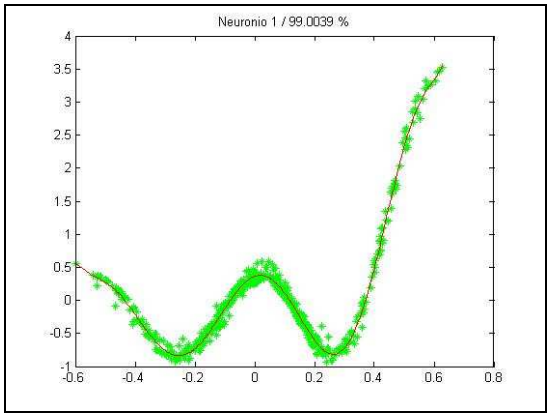

Figura 2.13 – Neurônio 1 após introdução do neurônio 4 e retro-ajuste

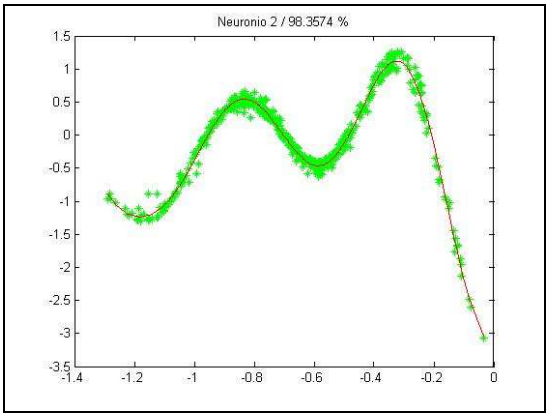

Figura 2.14 – Neurônio 2 após introdução do neurônio 4 e retro-ajuste

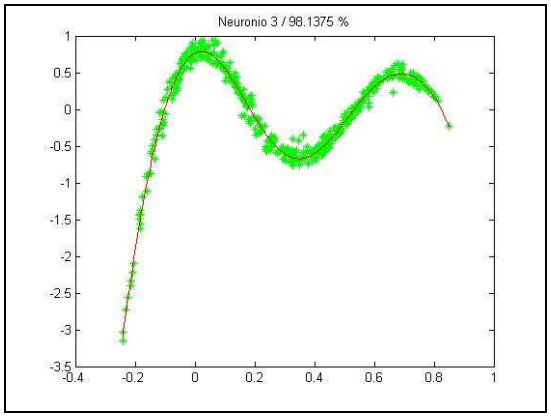

Figura 2.15 – Neurônio 3 após introdução do neurônio 4 e retro-ajuste

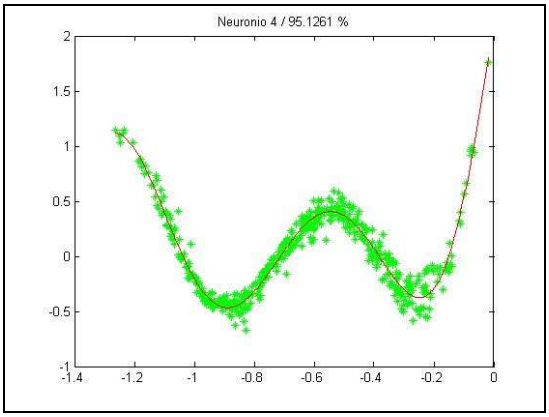

Figura 2.16 – Neurônio 4 após sua introdução e retro-ajuste

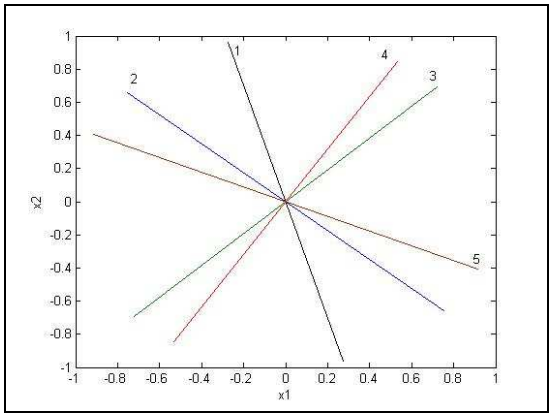

Figura 2.17 – Direções de projeção dos primeiros cinco neurônios

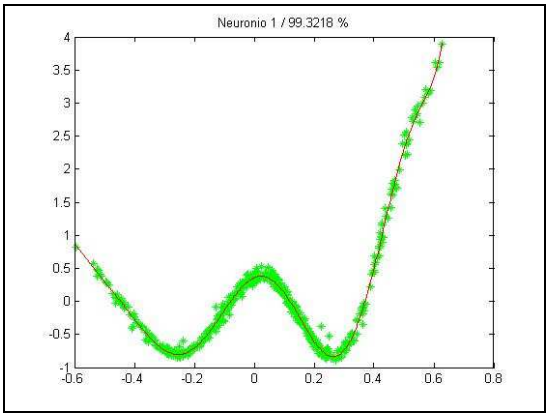

Figura 2.18 – Neurônio 1 após introdução do neurônio 5 e retro-ajuste

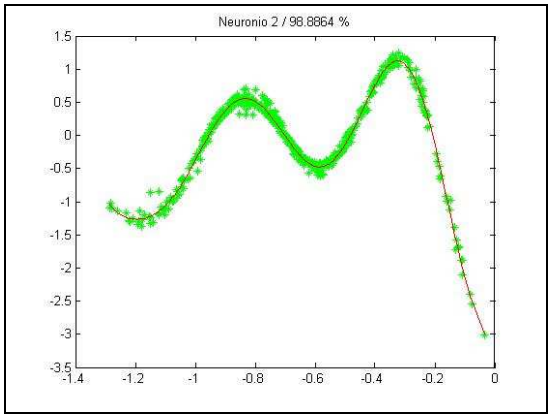

Figura 2.19 – Neurônio 2 após introdução do neurônio 5 e retro-ajuste

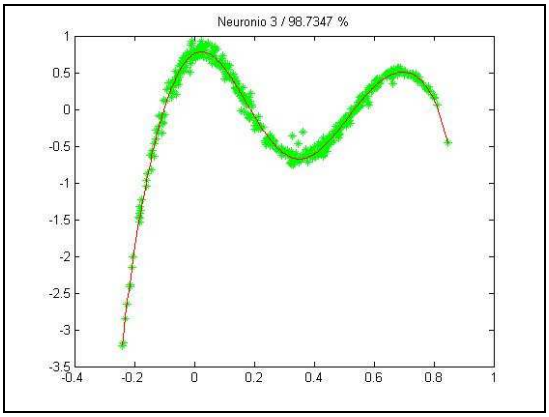

Figura 2.20 – Neurônio 3 após introdução do neurônio 5 e retro-ajuste

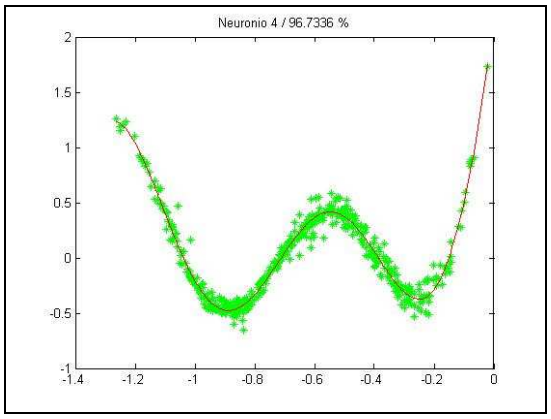

Figura 2.21 – Neurônio 4 após introdução do neurônio 5 e retro-ajuste

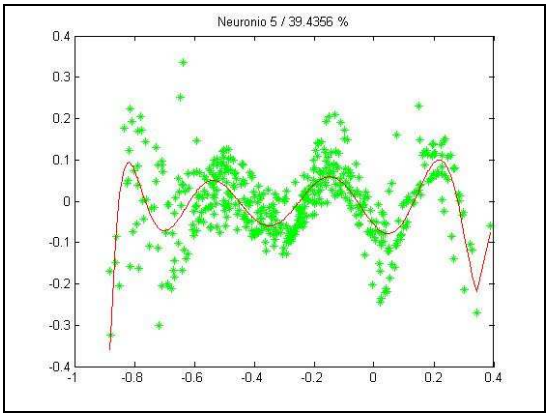

Figura 2.22 – Neurônio 5 após sua introdução e retro-ajuste

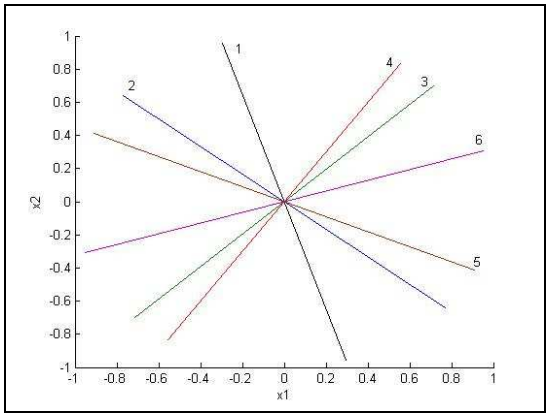

Figura 2.23 – Direção de Projeção para os seis neurônios

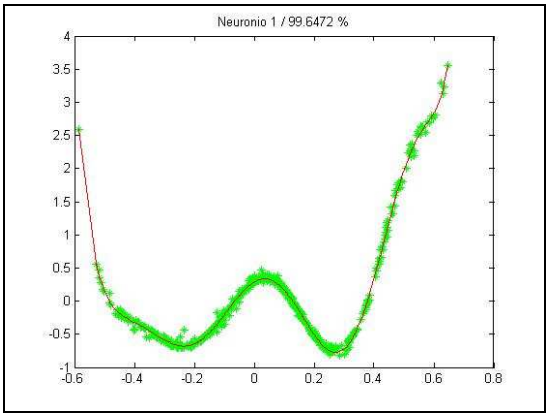

Figura 2.24 – Neurônio 1 após introdução do neurônio 6 e retro-ajuste

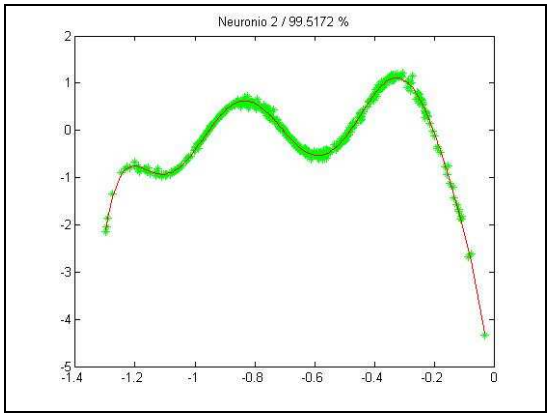

Figura 2.25 – Neurônio 2 após introdução do neurônio 6 e retro-ajuste

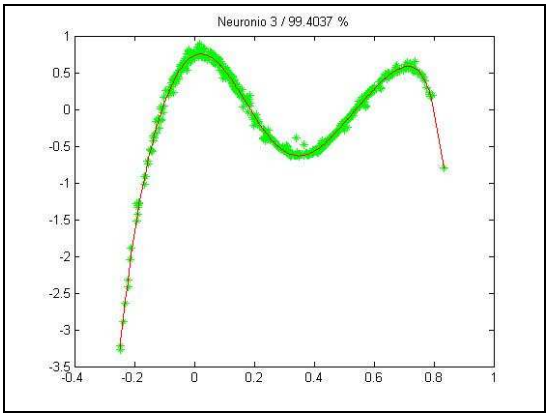

Figura 2.26 – Neurônio 3 após introdução do neurônio 6 e retro-ajuste

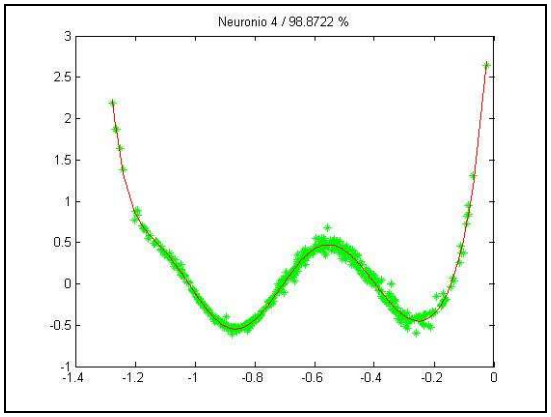

Figura 2.27 – Neurônio 4 após introdução do neurônio 6 e retro-ajuste

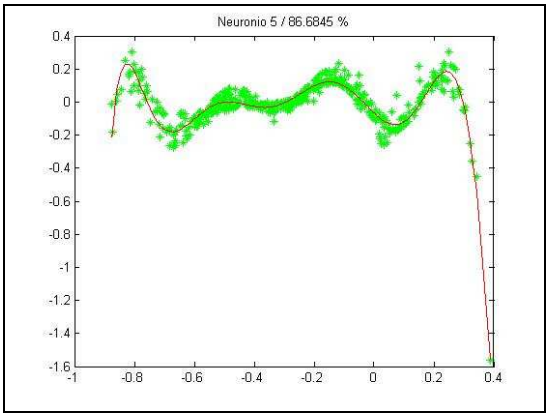

Figura 2.28 - Neurônio 5 após introdução do neurônio 6 e retro-ajuste

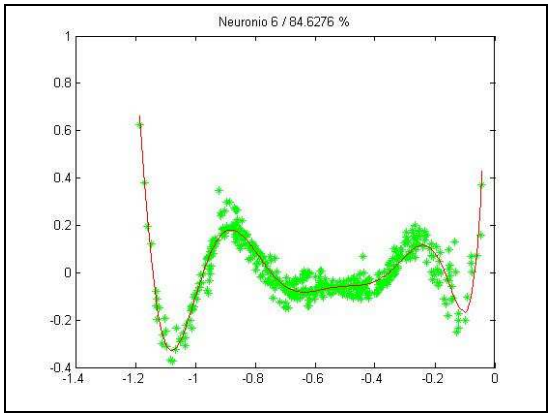

Figura 2.29 - Neurônio 6 após sua introdução e retro-ajuste

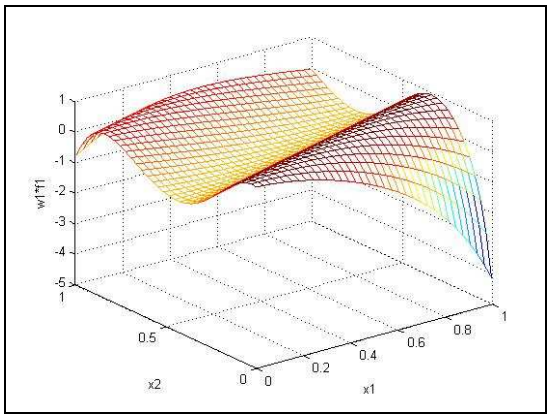

Figura 2.30 - $w_1 f_1$ em função de x_1 e x_2

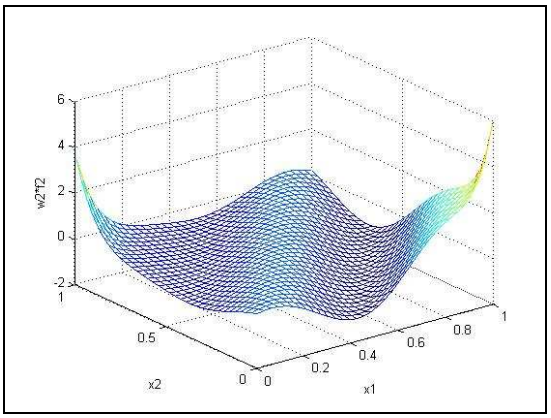

Figura 2.31 – $w_2 f_2$ em função de x_1 e x_2

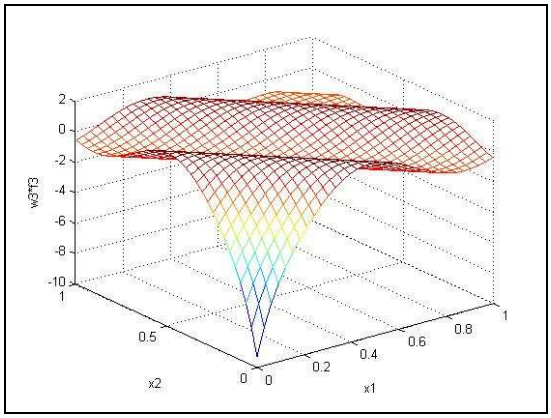

Figura 2.32 – $w_3 f_3$ em função de x_1 e x_2

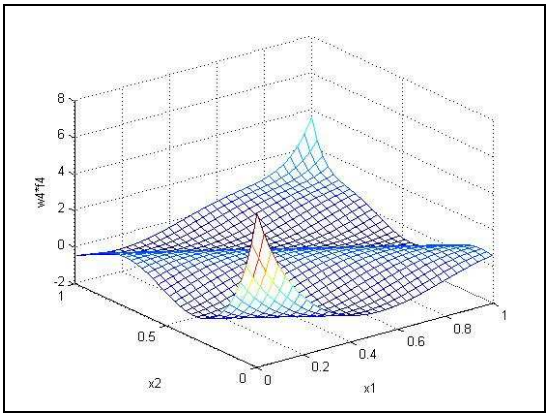

Figura 2.33 – $w_4 f_4$ em função de x_1 e x_2

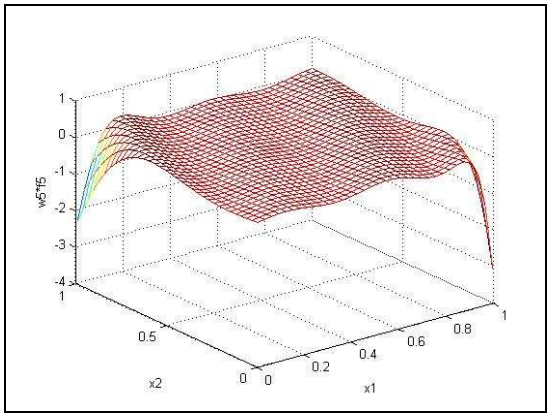

Figura 2.34 – $w_5 f_5$ em função de x_1 e x_2

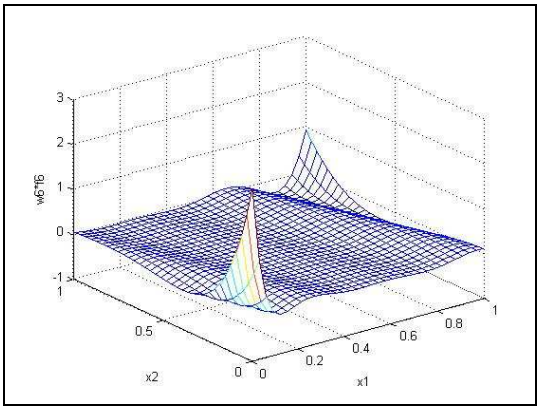

Figura 2.35 – $w_6 f_6$ em função de x_1 e x_2

2.8 Limitações do aprendizado por busca de projeção

O algoritmo PPL é bastante eficaz no treinamento de redes neurais tipo *feedforward* com estruturas parcimoniosas, sendo menos vulnerável aos efeitos da "maldição da dimensionalidade" que métodos de regressão e de treinamento de RNAs tradicionais, pois executa suas regressões em espaços unidimensionais, após definir uma direção de projeção para os dados originais.

Porém, o processo de busca de projeção ainda é sensível à dimensão do vetor de entrada. À medida que o número de entradas cresce, nota-se maior dificuldade em encontrar direções de projeção unidimensionais que preservem estruturas representativas dos dados amostrais, o que pode resultar em progressos lentos para o erro de aproximação, com a inserção de novos neurônios.

Este efeito é mais pronunciado em casos em que algumas das entradas sejam excessivamente ruidosas ou pouco relevantes para a determinação da(s) saída(s), pois forçam o algoritmo a encontrar sistematicamente projeções nas quais elas sejam pouco representativas.

Outra limitação do PPL é quanto ao treinamento de RNAs de múltiplas saídas: a busca por projeções que destaquem simultaneamente estruturas interessantes em todas as saídas é um processo custoso que raramente rende frutos, sendo quase sempre mais vantajoso o treinamento de redes separadas para cada saída (VON ZUBEN, 1996).

Por último, o PPL em suas versões iniciais (HWANG *et al.*, 1994) não possuía entrada de polarização nos neurônios da camada intermediária. Isto limitava o poder de aproximação destes neurônios, como mostrado por KWOK & YEUNG (1996). A versão do algoritmo descrita anteriormente inclui entradas de polarização.

Além disto, o PPL apresenta maior dificuldade na aproximação de alguns tipos de funções, como as harmônicas (MAECHLER *et al.*, 1990), as quais seriam melhor representadas por produtos de funções de expansão ortogonal e não pela soma delas.

Todas essas limitações serão tratadas ao longo dos demais capítulos, a menos da possibilidade de composição multiplicativa, a qual foi tratada em IYODA (2000).

Capítulo 3

Seleção de Variáveis

O termo Seleção de Variáveis (SV) se refere ao processo de identificar e isolar quais variáveis presentes em um banco de dados são mais expressivas na análise e solução de um determinado problema (KOHAVI & JOHN, 1997, 1998; GUYON & ELISSEEFF, 2003), em termos de grau de relevância e de não-redundância.

A utilização de métodos de SV é muito freqüente em diversas áreas da computação, como classificação de textos, mineração de dados e síntese de modelos (GUYON & ELISSEEFF, 2003), que possuem em comum a necessidade de lidar com conjuntos de dados amostrais bastante complexos, freqüentemente envolvendo número bastante elevado de variáveis independentes, bem como de instâncias.

As técnicas de SV permitem reduzir o número de variáveis a serem empregadas, permitindo melhor visualização e interpretação dos resultados, ou mesmo viabilizando o uso dos dados para algumas aplicações específicas. A seleção também pode resultar em modelos de maior acurácia na sua capacidade de representação. Algoritmos de treinamento de classificadores ou regressores apresentam degradação da acurácia do modelo gerado na presença de variáveis irrelevantes ou de variáveis correlacionadas (KOHAVI & JOHN, 1997, 1998).

Desta forma, SV também constitui uma maneira de se atenuar a “maldição da dimensionalidade” (HUBER, 1985), cuja estratégia empregada difere das técnicas de PP por reduzir o número de dimensões de fato, em vez de apenas trabalhar em projeções de menor dimensão. E, da mesma forma que uma determinada projeção dos dados pode falhar em realçar estruturas de interesse dos dados, uma seleção ruim pode remover informações importantes do conjunto de dados ou manter variáveis pouco informativas.

3.1 Variáveis *vs.* características

Na literatura de SV, às vezes (embora não universalmente) faz-se diferença entre os conceitos de *variável* e de *característica* (em inglês: *variables* e *features*). GUYON & ELISSEEFF (2003) definem v*ariáveis* como sendo os parâmetros de entrada originais de um conjunto de dados e *características* como sendo parâmetros de entrada sintéticos, construídos a partir das variáveis originais. Na literatura, variáveis e características podem também receber a denominação de atributos.

Tanto variáveis quanto características podem tomar parte em um mesmo processo de seleção (os algoritmos de seleção não fazem diferença entre os dois tipos de entradas) e existem alguns algoritmos que incluem tentativas de criar novas características como parte integrante da estratégia de seleção. Este procedimento é conhecido como Construção de Características (CC, do inglês *Feature Construction*) e, por vezes, pode resultar em conjuntos com um número de entradas superior ao de variáveis originais. Quando o número de entradas selecionadas é limitado de tal forma que haja necessariamente uma redução da dimensão dos dados (independentemente da adoção ou não de novas características), o processo é denominado Extração de Características (EC, do inglês: *Feature Extraction*) (GUYON & ELISSEEFF, 2003).

Como o funcionamento dos algoritmos de seleção é indiferente a um parâmetro ser uma variável ou uma característica, a distinção entre os dois freqüentemente não é importante. Por isso, adota-se neste trabalho a nomenclatura geral de variável para todas as entradas fornecidas a um modelo (classificador ou regressor), fazendo-se a distinção apenas se relevante.

3.2 Tipos de relevância

O papel de um algoritmo de SV é selecionar variáveis que sejam relevantes para a solução de um problema, permitindo a obtenção de classificadores ou regressores com performances superiores. JOHN *et al.* (1994) e KOHAVI & JOHN (1997, 1998) dividem as

variáveis em três classes, conforme o efeito de sua remoção de um classificador ou regressor ótimo (baseado na distribuição de probabilidade verdadeira do problema):

- **Fortemente Relevantes**: quando a remoção isolada da variável causa degradação de performance;
- **Fracamente Relevantes**: quando a variável não é fortemente relevante, mas existe um subconjunto de variáveis para o qual sua remoção causa degradação de performance. Por exemplo: duas variáveis, x_1 e x_2, relevantes mas redundantes em um problema de classificação, de modo que a remoção de x_1 de um subconjunto que não contenha x_2 cause degradação de performance;
- **Irrelevante**: quando a variável não é nem fortemente, nem fracamente relevante, ou seja, pode ser removida sem degradar a performance, independentemente de quais outras entradas estejam sendo incluídas.

O conceito de relevância serve de guia para o desenvolvimento e estudo dos algoritmos de seleção. Porém, tendo em vista que a distribuição de probabilidade dificilmente é conhecida em problemas reais de engenharia, a relevância de uma variável é sempre dada por uma estimativa.

3.2.1 Relevância *vs.* otimalidade

Um subconjunto de variáveis é dito ótimo quando o uso de qualquer outro não resultaria em uma melhor performance final (KOHAVI & JOHN, 1997, 1998). Diferentes modelos de classificação ou regressão apresentam diferentes reações aos graus de correlação e irrelevância das variáveis apresentadas, e mesmo ao número total de variáveis. Por isso, um subconjunto de variáveis que seja ótimo para um determinado modelo, não necessariamente o será para um outro.

É bom notar também que relevância de uma variável não implica em sua presença no subconjunto ótimo (embora seja o caso mais comum) (KOHAVI & JOHN, 1997, 1998). Como exemplo de variáveis relevantes que podem não pertencer ao subconjunto ótimo

podemos citar variáveis fracamente relevantes (que podem estar sendo substituídas por outras redundantes que contenham a mesma informação) e variáveis muito ruidosas (que mesmo relevantes poderiam atrapalhar a performance ao dificultar estimar a distribuição de probabilidade real). Como exemplo de variáveis irrelevantes mas que pertenceriam a um subconjunto ótimo, podemos citar as entradas fixas de valor '1' em uma rede MLP sem entrada de polarização (KOHAVI & JOHN, 1997). Outro exemplo em que otimalidade e relevância não são equivalentes são os casos em que a inclusão de duas ou mais variáveis fracamente relevantes, redundantes entre si, permite que o preditor reduza o ruído e obtenha uma melhor performance.

Assim, parece ser vantajoso para o algoritmo de SV considerar o tipo de classificador ou regressor adotado, caso se deseje atingir a melhor performance possível. Além disso, o subconjunto ótimo não é necessariamente único, sendo possível existirem subconjuntos diferentes que permitam obter performances equivalentes. Também se supõe aqui que se está fazendo o melhor uso possível das variáveis selecionadas, quando da síntese do classificador ou regressor, o que nem sempre é verdadeiro, já que um determinado algoritmo pode falhar em estabelecer corretamente a relação entre uma entrada e a saída do modelo sendo aproximado.

3.3 Ranqueamento *vs.* seleção de subconjuntos

Alguns algoritmos de seleção estabelecem, como forma principal ou auxiliar de seleção, um *ranking* das variáveis. Estes algoritmos de "ranqueamento" usam a carga informativa individual de cada variável como critério de relevância, gerando uma lista ordenada. Cabe ao usuário escolher quais utilizará.

Um exemplo deste tipo de algoritmo é o RELIEF (KIRA & RENDEL, 1992), o qual determina a relevância das variáveis com base na diferença entre cada instância e as suas duas instâncias mais próximas de mesma classe e de classe oposta (em sua versão original, este algoritmo estava voltado para problemas de classificação binária, embora existam atualmente versões para problemas com maior número de classes; KONENKO, 1994).

Os aspectos positivos deste tipo de algoritmo incluem simplicidade e escalabilidade, pois requerem apenas a computação de um valor para cada variável e a ordenação destes

valores (GUYON & ELISSEEFF, 2003), além de fornecerem uma informação mais completa do que apenas se a variável é considerada relevante ou não. Porém, uma vez que as variáveis são avaliadas, independentemente do contexto fornecido pelas outras, pode resultar no final uma escolha de variáveis redundantes entre si ou então podem acabar sendo negligenciadas variáveis que possuem bom poder preditivo quando associadas a outras.

Já os algoritmos de seleção de subconjuntos avaliam as variáveis dentro de um contexto (ou seja, como parte de um subconjunto específico), o que permite evitar mais facilmente a inclusão ou exclusão desnecessária de variáveis. Porém, esta estratégia resulta em algoritmos mais complexos computacionalmente e menos escaláveis.

3.4 Paradigmas de seleção de variáveis

A literatura divide os algoritmos de SV em três paradigmas (ver Figura 3.1), ou metodologias, principais: *wrappers*, filtros e métodos embutidos (ou embarcados, do inglês *embedded*) (YU & LIU, 2004). Esses paradigmas podem também ser combinados, produzindo métodos híbridos.

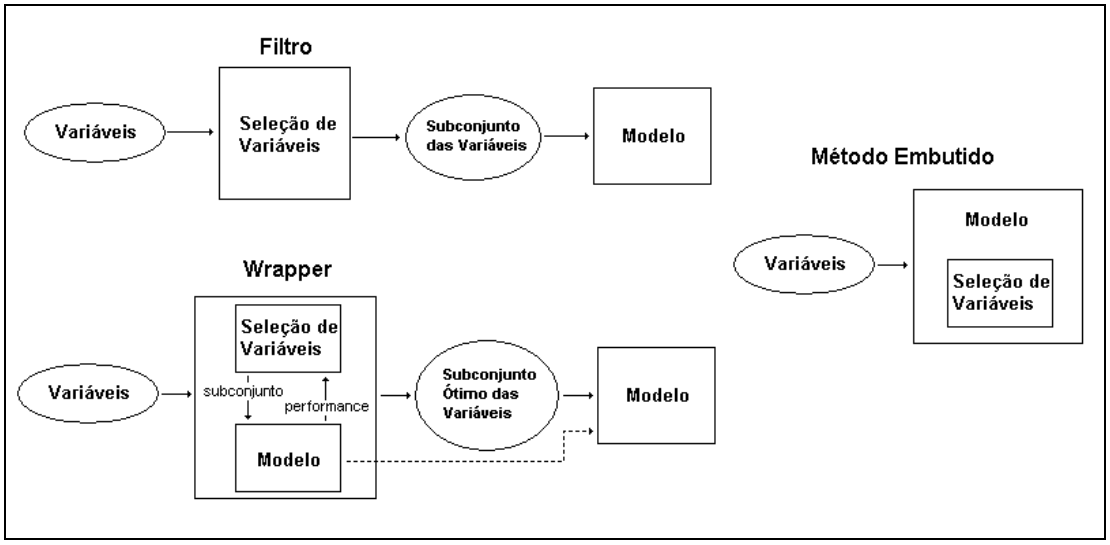

Figura 3.1 – Os três paradigmas de seleção de variáveis: Filtro, *Wrapper* e Método Embutido

A diferenciação entre *wrappers* e filtros (bem como a proposta da própria metodologia dos *wrappers*) foi feita pela primeira vez por JOHN *et al.* (1994). Posteriormente, incluiu-se a classe dos métodos embutidos entre as famílias de algoritmos de SV (GUYON & ELISSEEFF, 2003). Já os híbridos referem-se a métodos que combinam características de mais de um dos outros paradigmas e não serão tratados em mais detalhes.

3.4.1 Filtros

Muitos algoritmos de SV operam unicamente com base nas propriedades estatísticas intrínsecas dos dados disponíveis (como a correlação entre as variáveis de entrada e a saída ou entre as entradas apenas) na tentativa de determinar a relevância e/ou redundância de cada variável. Isto é feito independentemente das propriedades do classificador ou regressor que será empregado.

Mesmo sem considerar o impacto do uso efetivo desse subconjunto de variáveis selecionadas, a seleção é realizada com a expectativa de melhorar a performance e facilitar o processo de treinamento do classificador ou regressor, ao fornecer um subconjunto de dados reduzido (reduzindo a dimensionalidade do problema e simplificando a tarefa do algoritmo de aprendizagem).

Como a seleção é realizada em uma etapa anterior ao treinamento do preditor (ao qual não é em momento algum apresentado o conjunto completo de variáveis) ela é considerada um passo de pré-processamento dos dados e pode ser vista como uma etapa de "filtragem" do banco de dados. Por isso a denominação *filtro* (JOHN *et al*., 1994; KOHAVI & JOHN, 1997, 1998).

Em métodos de seleção que utilizam filtros, não há qualquer forma de realimentação entre o algoritmo de seleção e o algoritmo de aprendizagem. Em outras palavras, o processo de treinamento não guia a seleção, que ocorre independentemente do uso que será feito do subconjunto de variáveis selecionado. No entanto, é possível escolher ou projetar filtros adequados às características conhecidas de um classificador ou regressor específico (GUYON & ELISSEEFF, 2003).

Os algoritmos de filtro são muito utilizados atualmente (e historicamente) devido aos custos computacionais comparativamente baixos, sendo o paradigma mais comum na

estatística (KOHAVI & JOHN, 1998). Além disto, os resultados obtidos são mais “genéricos” que os fornecidos pelas outras metodologias, uma vez que não são vinculados a um classificador ou regressor específico.

A natureza genérica dos resultados possui aspectos positivos e negativos: resultados mais genéricos podem ser utilizados satisfatoriamente com uma gama maior de problemas e algoritmos de treinamento, mas a ausência de uma calibração específica tem por conseqüência falta de garantia de bons resultados finais para um dado classificador ou regressor.

Desta forma, filtros são mais indicados quando existe um interesse maior na obtenção de um banco de dados simplificado; quando os recursos computacionais disponíveis são limitados em comparação com a dimensão do problema de interesse; e quando a literatura registra um bom histórico de aplicação de um determinado algoritmo de filtro para um determinado classificador ou regressor.

3.4.2 *Wrappers*

Os wrappers diferenciam-se dos filtros por utilizarem, como parte do processo de seleção, o próprio classificador ou regressor para o qual as variáveis estão sendo selecionadas — para cada subconjunto em avaliação, um classificador ou regressor é treinado e sua performance é utilizada como medida da qualidade do subconjunto.

Esta estrutura permite que os *wrappers* busquem, não variáveis relevantes, mas um subconjunto de variáveis que seja o mais indicado para determinada aplicação. Para obter este subconjunto de alta performance, é necessário realizar o treinamento de pelo menos um classificador ou regressor para cada subconjunto de variáveis considerado, resultando em um elevado esforço computacional. Além disto, por serem específicos para determinada aplicação e para um classificador ou regressor específico, os resultados fornecidos por um *wrapper* não possuem garantias bom desempenho caso sejam aproveitados para uma aplicação diferente ou por outro tipo de classificador ou regressor, sendo, portanto, menos genéricos que os resultados de um algoritmo do tipo filtro.

O nome vem do fato do algoritmo de seleção “envolver” (em inglês: *wrap*) o modelo (classificador ou regressor) como parte do processo de seleção de variáveis.

Um outro ponto desfavorável dos *wrappers* é a tendência a sobre-ajuste, principalmente quando o número de amostras é pequeno (KOHAVI & JOHN, 1997, 1998; REUNANEN, 2003; RADTKE *et al.*, 2006) ou buscas mais intensas são realizadas (REUNANEN, 2003; LOUGHREY & CUNNINGHAM, 2004). Isto se deve ao fato de que poucas amostras podem não permitir generalizar bem os dados e quanto maior o número de subconjuntos de variáveis testados, maior a probabilidade de considerar um que cause sobre-ajuste, prejudicando o processo de seleção. Para evitar isto, é comum dividir os dados amostrais em um conjunto de treinamento (para o modelo) e um conjunto independente de validação (para a avaliação do subconjunto), ou mesmo a utilização de técnicas como a validação-cruzada (KOHAVI & JOHN,1997, 1998) e interrupção prematura (LOUGHREY & CUNNINGHAM, 2004).

Apesar das desvantagens (custo computacional elevado, pouca abrangência dos resultados e a necessidade de se controlar o sobre-ajuste), o uso dos *wrappers* justifica-se devido ao seu objetivo primário: o de obter um subconjunto de variáveis que seja o mais indicado para o modelo considerado. Também é interessante notar que, além de selecionar o subconjunto de variáveis, o método de *wrappers* permite obter o próprio classificador ou regressor, já treinado durante o processo de seleção.

O uso de *wrappers* é normalmente indicado quando o tipo de modelo a ser empregado na solução do problema já está bem definido e a obtenção do melhor classificador ou regressor possível é mais importante que a obtenção de um banco de dados simplificado. Evidentemente, a viabilidade da implementação prática está associada a existência de recursos computacionais suficientes à disposição.

3.4.3 Métodos embutidos

Os métodos embutidos envolvem a SV realizada como parte integrante de algum algoritmo de aprendizado de máquina. Nesses algoritmos, o processo de seleção não é claramente diferenciável do treinamento do modelo e, assim como no caso dos *wrappers*, os resultados obtidos são calibrados em relação a um classificador ou regressor específico.

Esta classe de algoritmos é comparativamente pouco discutida na literatura geral de SV; os algoritmos têm poucas propriedades em comum (GUYON & ELISSEEFF, 2003) e, por isso, são melhor explorados pela literatura específica de aprendizado de máquina nas quais

estão inseridos. Um exemplo seria o emprego de critérios de relevância na síntese de máquinas de vetores-suporte, fazendo com que o próprio processo de obtenção dos vetores-suporte incorpore a etapa de seleção de variáveis (RAKOTOMAMONJY, 2003), outro exemplo seria o algoritmo C4.5 para árvores de decisão (QUINLAN, 1993).

3.5 Estrutura geral de um *wrapper*

Um algoritmo do tipo *wrapper* é constituído por dois elementos principais: uma estratégia de busca do subconjunto de variáveis que leva o classificador ou regressor ao melhor desempenho e uma função de avaliação da qualidade dos subconjuntos candidatos (com base no desempenho do classificador ou regressor associado).

3.5.1 Estratégia de busca

A estratégia de busca envolve a escolha de uma representação adequada para os possíveis subconjuntos de variáveis (definindo assim o espaço de busca e associando um ponto do espaço de busca a cada subconjunto); dos operadores responsáveis por gerar os novos candidatos a serem avaliados (novos pontos no espaço de busca); uma condição inicial e um critério de interrupção da busca.

Uma representação bastante simples e direta consiste na codificação binária, contendo um *bit* para cada variável ("zero" indicando a exclusão da variável no subconjunto e "um" indicando sua inclusão). Esta representação garante que cada subconjunto possível seja representado no espaço de busca e que esta representação seja única, além de permitir definir operadores simples para gerar novos subconjuntos a partir dos existentes.

As quatro formas mais simples de se explorar o espaço de busca são a *busca exaustiva*, a *seleção adiante*, a *retro-eliminação* (do inglês: *exhaustive search*, *forward selection* e *backward elimination*, respectivamente) e a busca bidirecional (KOHAVI & JOHN, 1997; GUYON & ELISSEEFF, 2003).

Na busca exaustiva, todos os pontos do espaço de busca são testados e comparados de forma a se encontrar o melhor subconjunto de variáveis. Estratégias baseadas em busca

exaustiva são freqüentemente evitadas devido ao custo elevado para se realizar todas as avaliações necessárias, particularmente em casos com um número elevado de variáveis (GUYON & ELISSEEFF, 2003), já que o número de possíveis combinações de variáveis é da ordem de 2^n, com n sendo o número de variáveis.

Já na seleção adiante (*forward*), a exploração do espaço de estados é realizada adicionando-se novas variáveis a um subconjunto selecionado em um passo anterior. De forma geral, inicia-se a busca a partir do conjunto vazio e utiliza-se um operador simples que gera os "estados filhos" adicionando uma das variáveis ausentes no "estado pai". Cada "estado filho" é avaliado e o melhor avaliado é selecionado como o "pai" do próximo passo. Um critério de interrupção comum neste caso é a falha em encontrar um "estado filho" de melhor avaliação que o "pai" (KOHAVI & JOHN, 1997, 1998). Trata-se de um método guloso, que sempre adiciona a variável que mais contribui para o incremento de performance do subconjunto atual, razão pela qual o melhor subconjunto pode não ser obtido no final.

Na retro-eliminação, faz-se o inverso: explora-se o espaço de busca pela remoção de variáveis de um "estado pai". O estado inicial mais comum é o equivalente ao subconjunto que contém todas as variáveis, e o operador simples associado gera os "estados filhos" eliminando uma das variáveis presentes no "estado pai". O "estado filho" melhor avaliado torna-se o "estado pai" do passo seguinte. Como critério de interrupção, costuma-se adotar também a falha em encontrar um "estado filho" de melhor avaliação que o "estado pai" (KOHAVI & JOHN, 1997, 1998). Também temos aqui um método guloso que sofre da mesma deficiência da seleção adiante: o melhor subconjunto pode não ser obtido no final.

A estratégia de seleção adiante é computacionalmente menos custosa, pois evita o treinamento excessivo de modelos com muitas entradas (que são de treinamento mais custoso que os com poucas entradas), já que inicia a busca por subconjuntos pequenos. No entanto, a retro-eliminação permite que as variáveis sejam avaliadas desde o início em conjunto com todas as outras, o que pode evitar que certas combinações de variáveis que contribuem para o bom desempenho do classificador ou regressor sejam desfeitas (GUYON & ELISSEEFF, 2003; KOHAVI & JOHN, 1997, 1998).

A diferença entre as duas estratégias pode ser facilmente visualizada a partir de um exemplo simples: dado um conjunto composto por seis variáveis (x_1, x_2, x_3, x_4, x_5, x_6), das

quais apenas x_1, x_2 e x_3 são úteis para uma determinada aplicação e as demais são prejudiciais; e ainda supondo que a contribuição individual de x_1 é maior que a de x_2, que é maior que a de x_3, que x_4 é menos prejudicial do que x_5, que é menos prejudicial do que x_6, e que a performance do modelo aumenta monotonicamente conforme aumenta a soma das contribuições individuais de cada variável (e cai equivalentemente com a presença das variáveis prejudiciais). A busca se dará para as duas estratégias (interrompendo a busca caso nenhuma melhora seja obtida em um passo) conforme apresentado na Tabela 3.1 abaixo.

Tabela 3.1 – Exemplo de busca pelo subconjunto de variáveis que leva ao melhor desempenho

Passo	**Seleção Adiante**		**Retro-Eliminação**	
	Estado Pai Atual	**Estados Filhos Avaliados**	***Estado Pai Atual***	***Estados Filhos Avaliados***
1	[000000]	[100000] [010000] [001000] [000100] [000010] [000001]	[111111]	[011111] [101111] [110111] [111011] [111101] [111110]
2	[100000]	[110000] [101000] [100100] [100010] [100001]	[111110]	[011110] [101110] [110110] [111010] [111100]
3	[110000]	[111000] [110100] [110010] [110001]	[111100]	[011100] [101100] [110100] [111000]
4	[111000]	[111100] [111010] [111001]	[111000]	[011000] [101000] [110000]
final	[111000]	Busca terminada	[111000]	Busca terminada

As duas estratégias avaliariam um total de 18 estados, sendo 15 estados dotados de um máximo de três variáveis e apenas três com quatro variáveis para a seleção adiante e 15 contendo um mínimo de três variáveis e apenas três com duas para a retro-eliminação (ver também a Figura 3.2).

O exemplo, porém, supõe ausência de mínimos locais no espaço de estados, os quais causariam a interrupção prematura da busca. Se x_2 e x_3 sozinhas atrapalhassem o preditor, mas juntas aumentassem sua acurácia, a busca poderia ser interrompida entre os passos 2 e 3 na seleção adiante, por exemplo.

Além disso, o número de subconjuntos potencialmente necessários de se explorar no espaço de busca para se encontrar a melhor combinação de variáveis cresce muito

rapidamente com o aumento do número de variáveis, tornando estas estratégias de busca pouco factíveis.

Formas de se reduzir a vulnerabilidade a mínimos locais e aumentar a eficiência da exploração do espaço de busca, tanto da seleção adiante quanto da retro-eliminação, incluem diferentes critérios de interrupção ($k > 1$ passos sem melhorias de avaliação, por exemplo) e o uso de operadores mais complexos (como os operadores compostos descritos em KOHAVI & JOHN, 1997), e mesmo a combinação de operadores de adição e de eliminação de variáveis (REUNANEN, 2003), dando origem a um processo de busca bidirecional.

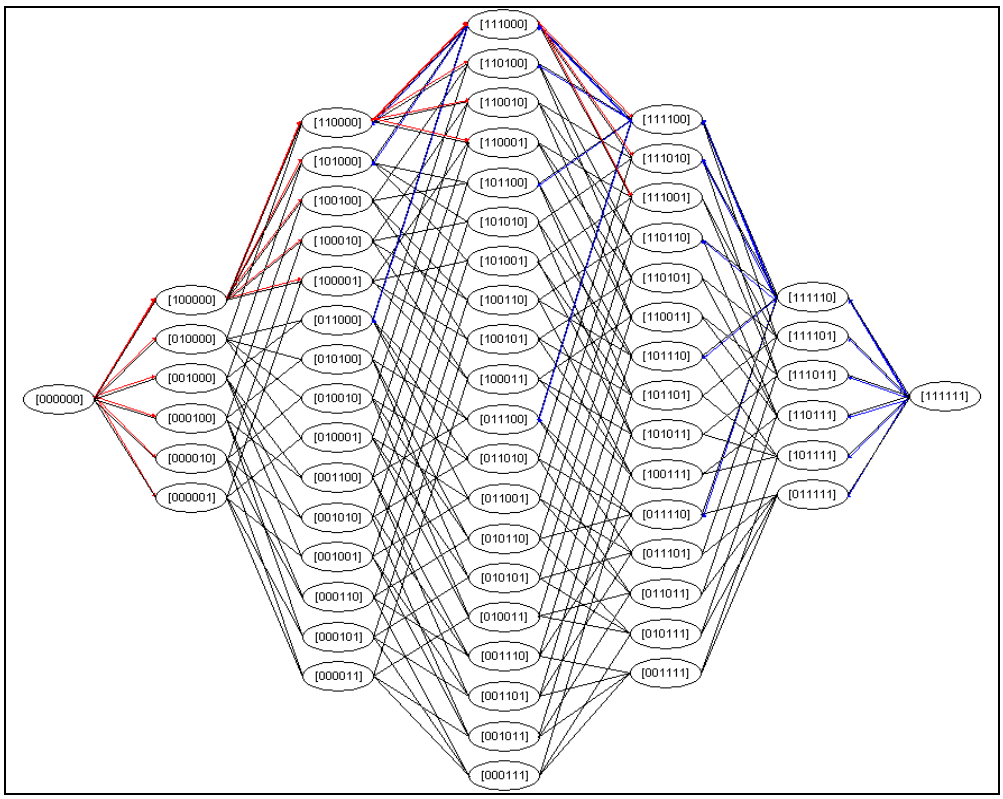

Figura 3.2 – Espaço de busca com codificação binária para um conjunto de seis variáveis e operadores de adição e eliminação simples. Setas em vermelho indicam os subconjuntos avaliados pela seleção adiante, no exemplo associado à Tabela 3.1. Setas em azul indicam os subconjuntos avaliados pela retro-eliminação.

Outro método de se explorar o espaço de busca é empregando alguma meta-heurística para favorecer a busca em regiões promissoras. Este tipo de busca é freqüentemente implementado utilizando-se métodos de computação evolutiva (BÄCK *et*

al., 2000a, 2000b). A exploração evolutiva torna-se particularmente atraente para buscas junto a problemas em que há um grande número de variáveis originais com interações complexas, pois apresenta baixa vulnerabilidade a mínimos locais e é capaz de explorar o espaço de busca de forma bastante eficiente (BÄCK *et al.*, 2000a, 2000b).

Algoritmos genéticos, em particular, são bastante adequados para esta aplicação, pois mesmo as formas clássicas do algoritmo e dos operadores genéticos (crossover e mutação) permitem a implementação de um *wrapper*, visto que a codificação binária pode ser adotada prontamente.

3.5.2 Função de avaliação

Uma vez que o objetivo de um *wrapper* é obter um modelo (classificador ou regressor) com o melhor desempenho possível, utiliza-se como função de avaliação do subconjunto de variáveis candidatas alguma estimativa da acurácia do modelo que emprega este subconjunto.

Como mencionado anteriormente, ao se definir uma função de avaliação da qualidade dos subconjuntos de variáveis, é interessante tomar medidas que minimizem a possibilidade ou o grau de sobre-ajuste. Para isto evita-se utilizar critérios como o erro junto ao conjunto de dados de treinamento (KOHAVI & JOHN, 1997, 1998).

O uso de um conjunto de dados de validação, separado do de treinamento, para avaliar a performance do modelo é altamente recomendável. Outras medidas, como procedimentos de validação cruzada ou interrupção prematura do treinamento, também podem ser utilizadas.

Apesar do termo principal da função de avaliação ser uma medida da performance do modelo, é comum adicionar-se um termo que penalize subconjuntos com um número grande de variáveis, de forma a polarizar a busca na direção de subconjuntos menores ou apenas “desempatar” em favor dos subconjuntos menores (KOHAVI & JOHN, 1997, 1998; GUYON & ELISSEEFF, 2003).

3.6 Wrappers evolutivos

Como mencionado, a implementação de um algoritmo genético (AG) como mecanismo de busca de um *wrapper* é um processo bastante direto, tendo sido o método escolhido para o tratamento de entradas excessivas para o PPL.

Um AG consiste em uma meta-heurística populacional inspirada pela teoria neo-darwiniana da evolução das espécies (BÄCK *et al.*, 2000a, 2000b). Em um AG, conceitos como hereditariedade, adaptação ao meio, reprodução, mutação e seleção natural são modelados de modo simplificado, de forma a permitir a "evolução" de possíveis soluções de um problema em busca de uma solução ótima.

Para tanto, associamos, a cada solução explorada, um *indivíduo*. Esta associação é feita codificando-se as características da solução em um vetor de atributos referido como *cromossomo*, que representa a carga "genética" do indivíduo. É necessário que a codificação utilizada para o *cromossomo* seja capaz de representar todas as possíveis soluções do problema.

A forma mais simples de codificação utilizada em AGs é a codificação binária, na qual cada *bit* ou seqüência de *bits* equivale a um gene e é responsável por indicar a presença, ausência, tipo, ou mesmo o valor de um determinado atributo da solução. Nota-se que este tipo de codificação é completamente compatível com a representação de estados discutida para os *wrappers*.

Determinante para a qualidade dos resultados obtidos pelo AG é a escolha de uma função de *fitness* apropriada para o problema. A função de *fitness* é responsável por "medir" o grau de "adaptação" de cada indivíduo ao "ambiente", associando um valor numérico a cada cromossomo. O "grau de adaptação" (*fitness*) é uma medida da qualidade da solução e é a função a ser otimizada durante a evolução.

A determinação da melhor função de *fitness* para um problema pode, em alguns casos, consistir em um problema à parte, requerendo ajustes e testes sucessivos do AG. Porém, no contexto de AGs para implementação de *wrappers*, a escolha é direta, consistindo da estimativa da acurácia do preditor empregada (erro de validação, validação cruzada, etc.).

Na metáfora neo-darwiniana, para podermos realizar um processo evolutivo, é necessário definir uma população composta por vários indivíduos (ou seja, um conjunto de soluções diferentes) e um modelo de reprodução que dê preferência aos genes dos indivíduos mais adaptados para a transmissão às "gerações futuras".

No contexto de AGs, entende-se por "reprodução" a geração de novas soluções a partir da combinação de soluções anteriores. Para isto, a literatura oferece uma variedade de operadores e de métodos de seleção de quais indivíduos vão se reproduzir em determinada geração (BÄCK *et al.*, 2000a, 2000b). O operador de reprodução mais simples é o *crossover* (ou recombinação) de um ponto e o método de seleção mais utilizado é a "roleta".

Na seleção por roleta, um ou mais pares de indivíduos são escolhidos para a reprodução estabelecendo-se, para cada um, uma probabilidade de seleção proporcional ao valor de seu *fitness*.

O operador de *crossover* de um ponto sorteia um local no qual os *cromossomos* que formam um par de reprodução serão divididos, dando origem a duas parcelas cada. A parcela anterior do primeiro cromossomo é então unida à parcela posterior do segundo e *vice-versa*, gerando dois novos cromossomos (ver Figura 3.3).

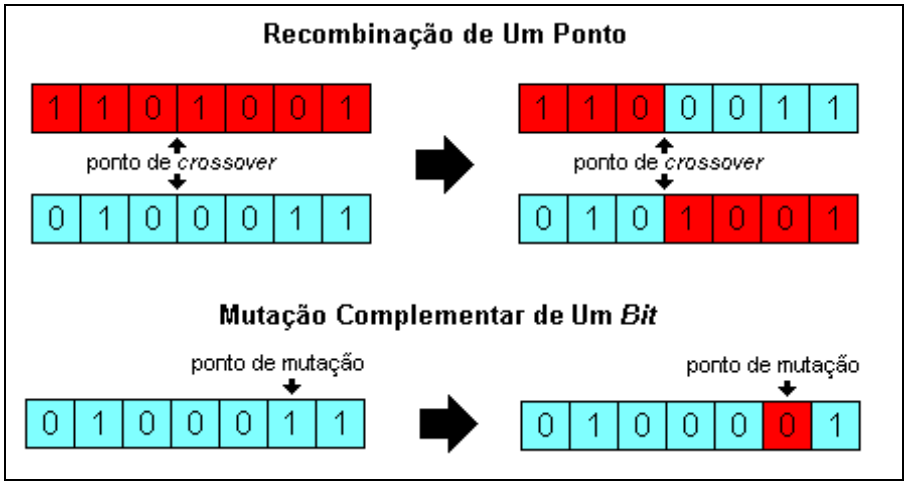

Figura 3.3 – Representação gráfica do *crossover* de um ponto e da mutação complementar de um *bit*

De forma geral, a aplicação, ou não, do operador de *crossover* (ou de outro operador de reprodução) em um par de cromossomos selecionados para se reproduzirem depende de uma probabilidade específica definida pelo usuário. Esta taxa de recombinação permite que

alguns pares sejam “copiados” para a geração seguinte sem sofrerem alteração genética, e é tipicamente um valor elevado (acima de 80%; BÄCK *et al.*, 2000a, 2000b).

O *crossover* e outros possíveis operadores de reprodução são os mecanismos de um AG responsáveis pela exploração do espaço de busca, permitindo que o algoritmo mude seu foco para outras regiões promissoras (que combinem as características de boas soluções já encontradas). Entretanto as novas soluções geradas pela reprodução normalmente se encontram em regiões distantes das ocupadas pelos “pais” (principalmente em espaços de busca definidos por um número grande de atributos), de modo que estes operadores não são suficientes para garantir que a vizinhança de uma boa solução seja analisada.

O papel de gerar novas soluções próximas às anteriores cabe a uma classe de operadores chamada de operadores de mutação, que podem introduzir pequenas alterações genotípicas nos cromossomos dos indivíduos da população. O operador de mutação mais simples para a codificação binária é a mutação complementar de um *bit* na qual um *bit* do cromossomo é sorteado e tem seu valor alterado (“um” vira “zero” e “zero” vira “um”; ver Figura 3.3). A mutação pode também conduzir a soluções distantes da solução corrente e pode até ser direcionada, ou seja, privilegiando a busca em certas regiões avaliadas como mais promissoras.

A mutação também tem o efeito de potencialmente introduzir na população características que não estavam presentes na geração anterior, aumentando a diversidade.

Assim como a recombinação, a mutação tem sua ocorrência controlada por uma taxa determinada pelo usuário, mas, diferentemente da taxa de recombinação, a taxa de mutação costuma ter valores reduzidos (BÄCK *et al.*, 2000a, 2000b).

Dessa forma, a partir de uma população inicial de soluções para um problema, um AG calcula o *fitness* de cada indivíduo; seleciona um determinado número destes indivíduos para sofrerem reprodução, a qual pode resultar, ou não, em novas soluções; e checa a ocorrência de mutação, definindo deste modo uma nova geração de soluções, que tem seu *fitness* calculado, reiniciando o ciclo.

O processo evolutivo do AG continua por gerações sucessivas até cumprir-se o critério de parada, o qual pode tomar várias formas: número máximo de gerações, número de gerações sem melhoria, meta de *fitness* alcançada, entre outros (BÄCK *et al.*, 2000a, 2000b). O indivíduo de melhor *fitness* da geração final é então fornecido como resposta.

A estrutura descrita acima descreve a forma mais básica de um AG e é de fácil implementação. Deve-se, no entanto, atentar ao fato de que boas soluções são substituídas, a cada geração, por outras que não são necessariamente superiores. Assim, embora se observe um aumento gradual do *fitness* médio da população com o passar das gerações (o qual não é necessariamente monotônico), é bem possível que a melhor solução gerada durante a execução do GA não seja preservada para ser retornada como solução final (BÄCK *et al.*, 2000a, 2000b).

A segunda limitação consiste em uma redução gradual da variabilidade entre os indivíduos da população. Uma vez que um indivíduo de *fitness* particularmente grande seja gerado, este passará a dominar a seleção por roleta, e seus sucessores (que provavelmente também terão *fitness* elevado) acabarão por dominar a população. A princípio isto pode soar vantajoso, pois indica uma convergência da população para uma região de ótimo. Porém, esta região pode representar apenas um ótimo local e, estando a diversidade muito reduzida, o algoritmo falhará em mudar seu foco para novas regiões, uma vez que tanto a reprodução quanto a mutação gerarão indivíduos na vizinhança do ótimo local.

A primeira limitação costuma ser vencida por uma estratégia conhecida como *elitismo* (BÄCK *et al.*, 2000a, 2000b), que consiste em garantir que o melhor indivíduo de cada geração seja preservado, sem alterações, na próxima geração. Já a segunda pode ser controlada ajustando-se os parâmetros do AG (taxa de mutação, taxa de recombinação, número de indivíduos na população) de forma a se obter uma convergência mais lenta, ou mesmo adotando-se mecanismos específicos de manutenção de diversidade (um exemplo simples seria substituir uma parcela da população por indivíduos gerados aleatoriamente).

Atenção também precisa ser dada à possibilidade de gerar soluções não factíveis (como subconjuntos vazios de variáveis). Nestes casos várias estratégias são possíveis, a mais simples sendo simplesmente atribuir um valor arbitrariamente ruim de *fitness* a tais indivíduos.

Capítulo 4

Contribuições

Embora o algoritmo de PPL tenha se mostrado eficiente no treinamento de RNAs parcimoniosas, de capacidade de predição acurada e boa generalização (HWANG *et al*., 1994; KWOK & YEUNG (1996); MELEIRO *et al*., 2008), algumas limitações ainda restringem sua aplicação em determinadas situações.

Nos contextos atualmente relevantes para o nosso grupo de pesquisa, quatro características do PPL apresentam-se particularmente limitantes, duas das quais são específicas da implementação que tínhamos disponível e outras duas inerentes ao algoritmo de PPL.

A principal limitação inerente ao método é a vulnerabilidade da performance do PPL ao número de entradas. À medida que cresce a dimensão do vetor de entradas, aumenta a dificuldade de se encontrar vetores de projeção que preservem estruturas dos dados suficientes para garantir boas aproximações. A segunda limitação dessa natureza refere-se à performance durante o treinamento pelo PPL em problemas de múltiplas saídas, a qual sofre uma degradação perceptível.

Já as limitações específicas da nossa implementação são: um limite de no máximo sete entradas para a RNA; treinamento ocorre sempre a partir do mesmo estado inicial (indesejável caso este estado leve a mínimos locais ruins) e a ausência de entradas de polarização nos neurônios artificiais da camada intermediária.

Para tratar essas limitações, atuamos no algoritmo em três frentes: incluir na implementação as entradas de polarização; substituir o método de inicialização dos pesos sinápticos dos neurônios da camada intermediária (responsável pelo limite do número de entradas e pela determinação do estado inicial do PPL); e associar um procedimento de seleção de variáveis ao PPL, de forma a se obter RNAs de melhor performance. O problema relativo a múltiplas saídas permanece em aberto.

4.1 Múltiplas saídas

Ao se apresentar mais de uma saída ao PPL, força-se o algoritmo a buscar projeções que apresentem informação relevante simultaneamente para todas as saídas. Isto por si só já resulta em um aumento da dificuldade na busca de projeção e, por conseqüência, em um aumento no esforço computacional. Porém, na prática, estas projeções mutuamente informativas só existirão caso as saídas sejam altamente correlacionadas, já que estas se expressam como diferentes combinações lineares das respostas dos neurônios intermediários.

Esta condição também é necessária para que as funções de ativação dos neurônios da camada intermediária tenham participação elevada na aproximação simultânea de várias saídas.

Como esta é uma situação rara no universo dos problemas de engenharia nos quais RNA são mais bem empregadas (mapeamentos não-lineares desconhecidos), observa-se que o PPL costuma necessitar de um número elevado de neurônios para realizar o mapeamento simultâneo das múltiplas saídas, e, devido à dificuldade na busca de projeção, não raramente tem dificuldades em obter boas aproximações.

Algumas alternativas para flexibilizar a relação entre as saídas no PPL e permitir um melhor mapeamento múltiplo foram exploradas dentro do grupo, porém, ainda sem resultados promissores. Sendo assim, o treinamento de uma RNA em separado por saída mantém-se como a solução mais adequada.

4.2 Entradas de polarização

Na versão original do algoritmo de PPL, proposta por HWANG *et al*. (1994), as RNAs geradas eram desprovidas de entradas de polarização nos neurônios artificiais. Isto ocorre porque não havia termos equivalentes no PPR e, uma vez que as propriedades de aproximação e de convergência do método eram conhecidas pelos estatísticos e comprovadas, julgou-se que elas se manteriam no PPL. A saída das RNAs geradas pelo PPL assumiam então, a seguinte forma:

$$\hat{y}_k = \sum_{j=1}^{n}\left[w_{jk} f_j\left(\sum_{l=1}^{m} v_{lj} x_l \right)\right] \tag{4.1}$$

Esta crença foi seguida por ocasião da programação da implementação do PPL utilizado pelo nosso grupo de pesquisa.

No entanto KWOK & YEUNG (1996) mostraram, teórica e experimentalmente, que ao substituir os *smoother* não-paramétricos do PPR por funções de ativação paramétricas (funções de Hermite, especificamente) no PPL, o algoritmo de HWANG *et al*. (1994) não conservava a propriedade de aproximação universal para valores finitos da ordem P das funções de Hermite. KWOK & YEUNG (1996) demonstraram ainda que a adição de um termo de polarização aos neurônios da rede garante a aproximação universal, e ainda aumenta a razão de convergência em relação ao número de neurônios e melhora a capacidade de generalização da RNA resultante.

Assim, o código da implementação foi modificado inserindo um termo de polarização v_{0j}, de modo que a saída da rede assumiu a forma:

$$\hat{y}_k = \sum_{j=1}^{n}\left[w_{jk} f_j\left(v_{0j} + \sum_{l=1}^{m} v_{lj} x_l \right)\right], \tag{4.2}$$

garantindo a propriedade de aproximação universal. Uma outra diferença introduzida por KWOK & YEUNG (1996) foi a possibilidade da inclusão do termo de ordem zero na determinação de $f_j(z_i)$ (equação 2.13), o qual não era utilizado por HWANG *et al*. (1994).

É importante, porém, observar que a implementação do PPL utilizada pelo grupo de pesquisa tem por base o algoritmo de VON ZUBEN (1996), não o de HWANG *et al*. (1994), e difere deste (e do de KWOK & YEUNG, 1996) por apresentar entradas de polarização nos neurônios da camada de saída (entre outros aspectos).

4.3 Inicialização dos pesos sinápticos dos neurônios da camada intermediária

Na implementação do PPL já disponível junto ao grupo de pesquisa, a inicialização do vetor de pesos dos neurônios da camada intermediária é feita “montando-se” um grupo de diferentes vetores direcionais. Cada vetor é utilizado para treinar um neurônio artificial e o neurônio que apresentar melhor índice de projeção (ou seja, que melhor aproximar o resíduo da RNA) é escolhido para ser introduzido na rede.

Estes vetores são referidos por *vetores candidatos à direção inicial de projeção*, e os neurônios gerados a partir deles são treinados utilizando-se parâmetros mais relaxados (fornecendo um treinamento parcial apenas). Uma vez escolhido um destes neurônios, o mesmo é submetido a um novo treinamento com parâmetros mais exigentes.

A geração de múltiplos vetores candidatos à direção inicial de projeção justifica-se devido ao fato de a superfície definida pelo erro de aproximação de uma RNA conter tipicamente vários mínimos locais. Várias direções candidatas permitem explorar mais regiões da superfície de erro e reduzir a chance do treinamento do neurônio convergir para um mínimo local ruim.

O treinamento parcial de neurônios a partir das direções candidatas evita que direções de projeção de pior índice inicial, mas que convergiriam para um bom mínimo, sejam preteridas em favor de direções de melhor índice inicial, mas que convergiriam para mínimos locais não tão bons.

Os vetores candidatos são “montados” de forma a se tentar cobrir o espaço multidimensional, definido pelo número de entradas, uniformemente. Esta cobertura uniforme consiste em uma forma de se tentar evitar que boas direções sejam ignoradas porque as direções candidatas se concentraram em regiões piores da superfície de erro.

O processo de “montagem” ocorre da seguinte forma: dado um determinado valor para o parâmetro *passo* (definido pelo usuário, mas com valor *default* de 0,4), o algoritmo percorre cada dimensão dos vetores candidatos atribuindo aos componentes dos vetores valores espaçados pelo tamanho do *passo*. Cada valor atribuído ao termo de uma dimensão define um vetor, e cada vetor já definido terá o termo da dimensão seguinte excursionado, gerando novos vetores a partir dos anteriores. Cada dimensão é percorrida dentro de

intervalos definidos de forma a garantir, ao final do processo, módulo unitário para todos os vetores "montados".

Considerando *m* entradas para a RNA e iniciando com os termos correspondentes à primeira dimensão e, supondo um valor de 0,4 para o *passo*, percorre-se o intervalo [–1; 1], gerando os valores (–1; –0,6; –0,2; 0,2; 0,6; 1).

Para cada um destes valores, percorre-se o termo da segunda dimensão, agora dentro do intervalo $\left[-\sqrt{\left(1-\left(v_{n1}^{k}\right)^{2}\right)};+\sqrt{\left(1-\left(v_{n1}^{k}\right)^{2}\right)}\right]$, com v_{n1}^{k} sendo o valor do primeiro termo para o *k*-ésimo vetor candidato à direção inicial de projeção do neurônio *n*.

Geram-se, desta forma, os vetores parciais: $[-1\ 0]^{T}$, $[-0{,}6\ -0{,}8]^{T}$, $[-0{,}6\ -0{,}4]^{T}$, $[-0{,}6\ 0]^{T}$, $[-0{,}6\ 0{,}4]^{T}$, $[-0{,}6\ 0{,}8]^{T}$, $[-0{,}2\ -0{,}9798]^{T}$, $[-0{,}2\ -0{,}5798]^{T}$, $[-0{,}2\ -0{,}1798]^{T}$, $[-0{,}2\ 0{,}2202]^{T}$, $[-0{,}2\ 0{,}6202]^{T}$, $[0{,}2\ -0{,}9798]^{T}$, $[0{,}2\ -0{,}5798]^{T}$, $[0{,}2\ -0{,}1798]^{T}$, $[0{,}2\ 0{,}2202]^{T}$, $[0{,}2\ 0{,}6202]^{T}$, $[0{,}6\ -0{,}8]^{T}$, $[0{,}6\ -0{,}4]^{T}$, $[0{,}6\ 0]^{T}$, $[0{,}6\ 0{,}4]^{T}$, $[0{,}6\ 0{,}8]^{T}$, $[1\ 0]^{T}$.

O processo segue adicionando-se a terceira dimensão a cada vetor parcial, e excursionando o valor desta no intervalo $\left[-\sqrt{\left(1-\sum_{l=1}^{2}\left(v_{nl}^{k}\right)^{2}\right)};+\sqrt{\left(1-\sum_{l=1}^{2}\left(v_{nl}^{k}\right)^{2}\right)}\right]$, e assim por diante, até a última dimensão, cujos termos recebem os valores $-\sqrt{\left(1-\sum_{l=1}^{m}\left(v_{nl}^{k}\right)^{2}\right)}$ e $+\sqrt{\left(1-\sum_{l=1}^{m}\left(v_{nl}^{k}\right)^{2}\right)}$, com *m* igual ao número de entradas da RNA. As Figuras 4.1-4.6 apresentam conjuntos de direções candidatas para duas e três dimensões geradas por este método (para um *passo* de 0,2 e também de 0,4 e de 0,6).

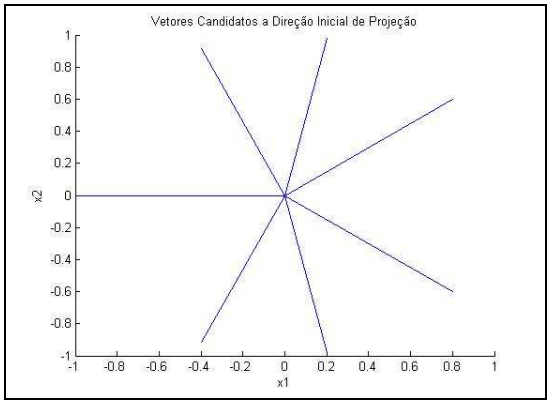

Figura 4.1 – Duas dimensões. *Passo* = 0,6

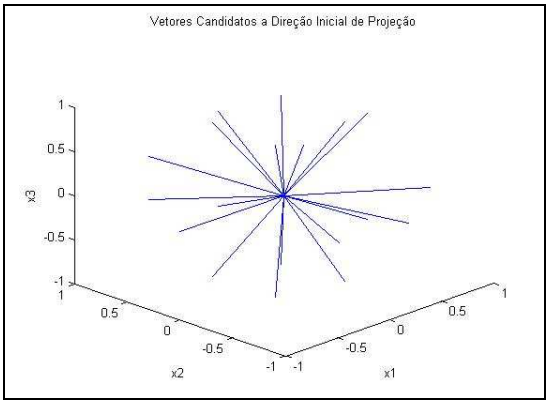

Figura 4.2 – Três dimensões. *Passo* = 0,6

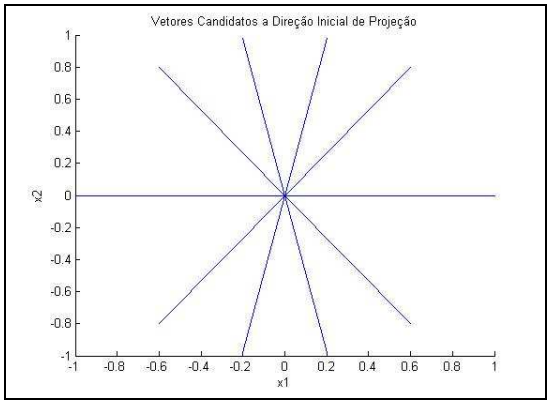

Figura 4.3 – Duas dimensões. *Passo* = 0,4

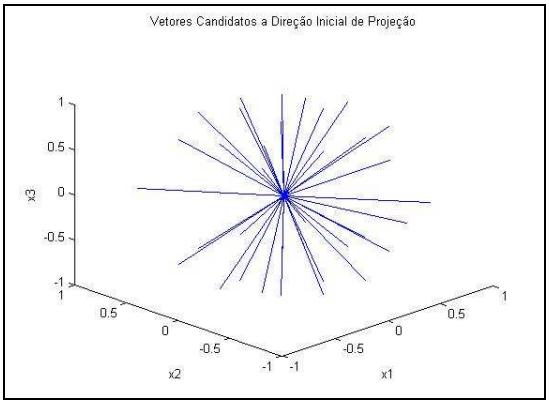

Figura 4.4 – Três dimensões. *Passo* = 0,4

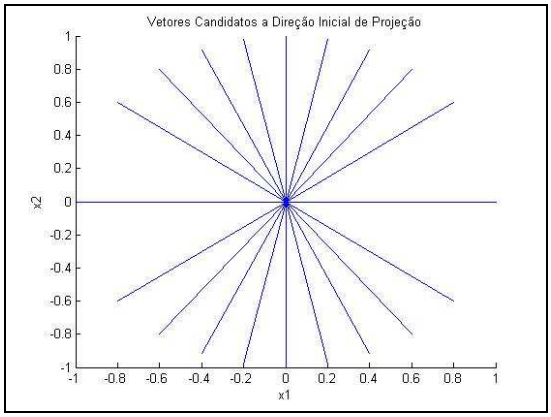

Figura 4.5 – Duas dimensões. *Passo* = 0,2

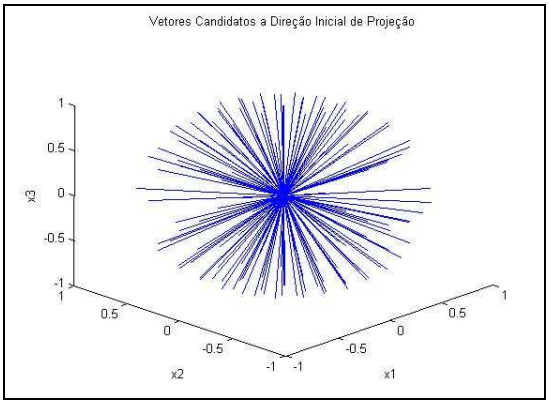

Figura 4.6 – Três dimensões. *Passo* = 0,2

Desta forma, o número de direções candidatas geradas depende do valor utilizado para o *passo* e do número de entradas, e o conjunto de direções gerado é sempre o mesmo para um dado par {*passo*, *m*} (assim o PPL inicia o treinamento sempre do mesmo estado inicial quando é utilizado este método de inicialização dos pesos intermediários). O motivo de o algoritmo estar limitado a apenas sete entradas deve-se ao fato de que o número total de direções candidatas cresce rapidamente com o aumento de *m*, e mais rápido ainda com a redução no *passo*, como pode ser visto nas Figuras 4.7 e 4.8.

Este método de inicialização dos pesos sinápticos, apesar de ser de execução rápida e resultar em resultados satisfatórios no final do processo de treinamento (tendo sido utilizado em VON ZUBEN (1996), LIMA (2000), MELEIRO *et al.* (2008), resulta em três características que podem dar margem a melhorias:

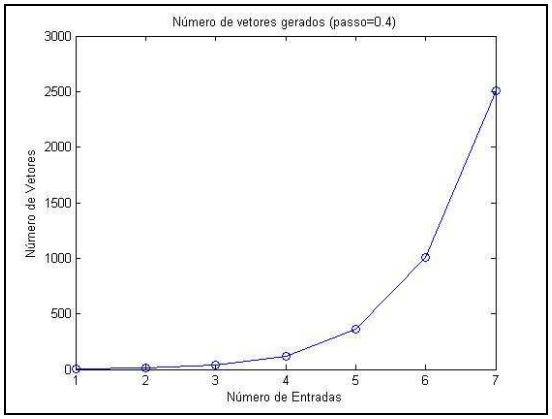

Figura 4.7 – Número de direções vs. número de entradas (*m*)

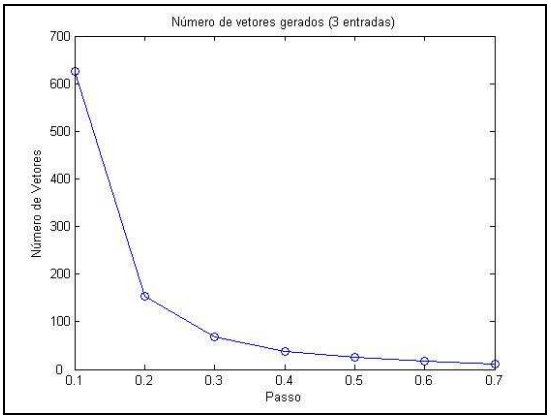

Figura 4.8 – Número de direções vs. valor do *passo*

- A primeira é o fato de as soluções candidatas geradas terem uma distribuição apenas razoavelmente uniforme no espaço, sendo possível obter ainda uma maior uniformidade.

- A segunda é o fato de a inicialização gerar sempre o mesmo estado inicial do PPL e, portanto, sempre resultar na mesma RNA. Caso este estado inicial leve a um mínimo local ruim, é impossível obter redes melhores sem alterar a inicialização.

- E a terceira é o limite do número de dimensões, que restringe o uso do PPL a problemas com menos de dez variáveis de entrada, já que a extensão do algoritmo para valores maiores é muito pouco prática.

Estas características foram toleradas na época desta implementação devido ao custo computacional já elevado do PPL, que tornava difícil justificar maiores esforços na inicialização de cada neurônio, principalmente no tocante a problemas com muitas variáveis, já que o tempo de treinamento do PPL aumenta conforme se adicionam novas entradas (LIMA, 2000). No entanto, com o aumento do poder computacional disponível hoje, torna-se factível a utilização do PPL para treinar redes com um número maior de entradas.

4.3.1 Métodos alternativos de inicialização dos pesos sinápticos da camada intermediária

Para superar as limitações apresentadas pelo método de "montagem" de direções candidatas para a inicialização do vetor de pesos sinápticos dos neurônios da camada intermediária, duas variações para a geração dos vetores candidatos foram implementadas e testadas.

A primeira variação consiste na simples geração aleatória de um conjunto de direções candidatas. A segunda, na geração aleatória seguida pela aplicação de um algoritmo gerador de diversidade (o SAND, ver apêndice A). Ambos os métodos evitam as limitações da inicialização com o mesmo estado em todas as execuções e também não restringem o número de entradas. Porém, espera-se que a inicialização puramente aleatória force o PPL a trabalhar com um conjunto de vetores candidatos distribuídos menos uniformemente no espaço, enquanto o uso do SAND causará um aumento do custo computacional da inicialização.

4.4 Seleção de variáveis de entrada para o aprendizado por busca de projeção

Como afirmado anteriormente, a presença de um número elevado de entradas prejudica a performance do treinamento realizado pelo algoritmo de PPL. Mesmo a simples inclusão de uma única entrada irrelevante tem o potencial de prejudicar o treinamento. Para ilustrar este fato, utilizaremos a mesma função $f^{(5)}(x_1,x_2)$ já empregada no do Capítulo 2 (equação 2.27, ver também Apêndice A).

Utilizando-se o PPL com entradas de polarização, foram treinadas cinco RNAs a partir de um conjunto de dados de treinamento de 350 instâncias, com $(x_1, x_2) \in [0,1]^2$, máximo de seis neurônios e função de Hermite com ordem doze; as redes foram testadas utilizando um conjunto de teste independente de 150 instâncias. A seguir, foram treinadas e testadas mais cinco RNAs. Foram utilizados os mesmos parâmetros e conjunto de dados, mas incluindo uma terceira entrada $x_3 \in [0,1]$ aleatória e irrelevante para a determinação da

saída. Os resultados estão na Tabela 4.1 e mostram claramente uma queda na performance do treinamento.

Tabela 4.1 – Comparação do PPL com e sem entrada irrelevante

Erro Quadrático Médio de Teste	**Sem entrada irrelevante**	**Com entrada irrelevante**
Média	0.79381613339921	0.88768058635894
Desvio padrão	0.07869185874716	0.16301581509327

Embora um aumento no esforço necessário para se obter uma boa rede ao final do treinamento seja esperado quando são adicionadas entradas inúteis ou pouco úteis (já que aumentam a dimensão dos vetores de projeção necessários), a existência de projeções capazes de "filtrar" o ruído representado pelas mesmas (projeções ortogonais às entradas inúteis) deveria garantir que o PPL fosse capaz de treinar redes com a mesma acurácia e capacidade de generalização das treinadas utilizando-se apenas entradas úteis. A dificuldade mostrada pelo PPL em atingir estas projeções leva-nos a crer que a adição de entradas leva a uma proliferação de mínimos locais na superfície de regressão, prejudicando o treinamento e justificando a seleção das entradas na obtenção de RNA melhores.

4.4.1 *Wrapper* para seleção de entradas no aprendizado por busca de projeção

Visando coerência com a motivação principal do algoritmo de PPL, a obtenção de RNAs parcimoniosas e com boa acurácia e capacidade de generalização, adotou-se a metodologia de *wrappers* para realizar a SV a ser associada ao aprendizado construtivo do PPL, uma vez que um *wrapper* permitirá selecionar um subconjunto de variáveis úteis que garantam uma boa performance da RNA final. Além disso, o uso de um *wrapper* torna trivial o ajuste e reaproveitamento do algoritmo para problemas de natureza variada (já que o treinamento da RNA é tratado como uma "caixa preta" pelo *wrapper*).

Uma vez que a presença de entradas extras, por si só, tende a degradar a performance do resultado alcançado pelo PPL, optou-se por não incluir um termo de penalização para subconjuntos com muitas entradas. Escolheu-se também utilizar uma

estratégia de otimização baseada em Algoritmos Genéticos (AG) e codificação binária do espaço de busca. Os operadores de reprodução e mutação utilizados são: *crossover* de um ponto e mutação complementar de um *bit*. A função de *fitness* é inverso do erro quadrático médio (EQM) da RNA treinada sobre um conjunto de dados de validação (não utilizados no treinamento); é utilizada a seleção por roleta, a população tem tamanho fixo e os indivíduos de melhor e pior *fitness* de cada geração são preservados na geração posterior. O critério de parada utilizado é o de número de gerações

Ao algoritmo final resultante, damos o nome ***WIS-PPL*** (do inglês: *Wrapper for Input Selection in Projection Pursuit Learning*), cujo pseudo-código é apresentado abaixo:

1. *Inicializar a primeira Geração de subconjuntos de entradas candidatos,*
2. *Para Geração = 1 até MaxGen,*
    - 2.1. *Para Subconjunto = 1 até o último,*
        - 2.1.1. *Treinar RNA usando PPL e Subconjunto,*
        - 2.1.2. *Calcular* Fitness *baseado no erro de validação,*
    - 2.2. *Fim do* Para
    - 2.3. *Determinando próxima Geração:*
        - 2.3.1. *Copiar Subconjuntos com melhor e pior* Fitness *para a próxima Geração,*
        - 2.3.2. *Para as vagas restantes na população:*
            - 2.3.2.1. *Rodar roleta para escolher um par,*
            - 2.3.2.2. *Checar* crossover*,*
            - 2.3.2.3. *Checar mutação,*
        - 2.3.3. *Fim do* Para
3. *Fim do* Para

Capítulo 5

Resultados Experimentais

Neste capítulo, são apresentados os resultados e comparações experimentais realizados visando avaliar as modificações incluídas no algoritmo de PPL e também o algoritmo WIS-PPL.

Os resultados estão divididos em três grupos: o primeiro inclui aqueles relativos ao efeito da inclusão da entrada de polarização nos neurônios da camada intermediária; o segundo, os pertinentes à modificação do método de inicialização dos neurônios da camada intermediária; e o terceiro, os resultados obtidos para o algoritmo WIS-PPL, incluindo uma análise do impacto do ajuste dos parâmetros do algoritmo genético no resultado final, comparações entre os resultados do WIS-PPL e do PPL sem seleção de variáveis, de *perceptrons* multicamadas e de alguns outros algoritmos, e um estudo sobre o grau de sobre-ajuste observado no algoritmo.

5.1 Influência da entrada de polarização

Como apresentado no Capítulo 4, KWOK & YEUNG (1996) demonstraram que o algoritmo do PPL não tinha garantia de aproximação universal devido à ausência de entradas de polarização na camada intermediária. No entanto, o algoritmo de VON ZUBEN (1996), base da implementação disponível no nosso grupo de pesquisa, apresenta entradas de polarização na camada de saída, não estando, portanto, coberto pela análise feita em KWOK & YEUNG (1996).

Para observar a influência da introdução de entradas de polarização na camada intermediária no algoritmo de VON ZUBEN (1996) e confirmar o aumento da flexibilidade do PPL, foi realizado um experimento simples, descrito a seguir.

Utilizou-se uma RNA arbitrária, com entradas de polarização nas camadas intermediárias e de saída, para gerar um conjunto de dados de treinamento de 500 instâncias. Depois, utilizando-se um esquema de validação cruzada de dez divisões, obteve-se a média e desvio padrão do erro quadrático médio (EQM) para uma série de RNAs treinadas utilizando-se o PPL com e sem entrada de polarização na camada intermediária.

Os parâmetros da RNA utilizada para gerar os dados de treinamento são apresentados na Tabela 5.1. Foram treinadas RNAs atribuindo-se ao número máximo de neurônio na camada intermediária (*nMAX*) os valores 2, 3, 4 e 6 e a ordem do polinômio de Hermite (*P*) recebeu os valores 2, 3 e 5, de forma a se obter modelos com diferentes graus de flexibilidade. As médias e desvios padrões dos EQMs apresentados em cada configuração de parâmetros são apresentados na Tabela 5.2.

Tabela 5.1 – Parâmetros da RNA utilizada na geração do conjunto de dados de treinamento

Número de entradas: 2 **Neurônios da camada intermediária:** 3 **Ordem do polinômio de Hermite:** 2	$\boldsymbol{v} = \begin{bmatrix} 0{,}5 & 0{,}5 & 0{,}5 \\ 0{,}5 & 0{,}5 & 0{,}5 \\ 0{,}5 & 0{,}5 & 0{,}5 \end{bmatrix}$	$\boldsymbol{w} = \begin{bmatrix} 0{,}5 & 0{,}5 \\ 0{,}5 & 0{,}5 \\ 0{,}5 & 0{,}5 \end{bmatrix}$	$\boldsymbol{c} = \begin{bmatrix} 0{,}5 & 0{,}5 & 0{,}5 \\ 0{,}5 & 0{,}5 & 0{,}5 \\ 0{,}5 & 0{,}5 & 0{,}5 \end{bmatrix}$
As primeiras colunas das matrizes v e w, correspondem aos pesos das entradas de polarização da camada intermediária e da camada de saída, respectivamente. As colunas da matriz $\mathbf{c}$ *correspondem aos vetores* $\mathbf{c}_j$ *de cada neurônio da camada intermediária.*			

Tabela 5.2 – EQM de validação das RNAs treinadas para aproximar os dados de treinamento gerados pela RNA da Tabela 5.1

nMAX	**Polarização na camada intermediária**	***P* = 2**		***P* = 3**		***P* = 5**	
		Média	Desvio padrão	Média	Desvio padrão	Média	Desvio padrão
2	Não	**0.0011890**	0.0008111	**0.0003932**	0.0002295	**0.0001094**	0.0000964
	Sim	**0.0003908**	0.0004309	**0.0002068**	0.0002289	**0.0000316**	0.0000224
3	Não	**0.0007578**	0.0006529	**0.0003307**	0.0002450	**0.0000444**	0.0000262
	Sim	**0.0001070**	0.0000952	**0.0000449**	0.0000373	**0.0000286**	0.0000328
4	Não	**0.0006226**	0.0004927	**0.0002189**	0.0001536	**0.0000386**	0.0000367
	Sim	**0.0002193**	0.0002643	**0.0004789**	0.0013489	**0.0000077**	0.0000093
6	Não	**0.0005961**	0.0005010	**0.0002511**	0.0002966	**0.0000400**	0.0000628
	Sim	**0.0002058**	0.0005593	**0.0000237**	0.00004637	**0.0000200**	0.0000445

Observamos uma tendência clara da arquitetura com entrada de polarização na camada intermediária de obter erros menores, principalmente nos casos em que a arquitetura não é flexível o suficiente ($nMAX < 3$ e $P < 3$), justificando a inclusão destas junto ao algoritmo de VON ZUBEN (1996).

5.2 Inicialização dos pesos sinápticos da camada intermediária

Para determinar os méritos relativos entre os três métodos estudados para a inicialização dos pesos sinápticos dos neurônios da camada intermediária, uma série de comparações experimentais foi realizada. Estes resultados são apresentados na Tabela 5.3. Foram considerados o método original de inicialização (de montagem dos vetores candidatos), o método aleatório e o método utilizando o SAND.

Inicialmente, foram treinadas várias RNAs como preditores para a série temporal *England Monthly Temperature* (ver Apêndice A), sendo o número de entradas variado de um a seis (máximo permitido pela implementação corrente do método de montagem, levando em conta a presença da entrada de polarização na camada intermediária de neurônios). As entradas apresentadas foram os valores da série com atrasos consecutivos (ou seja, t – 1, t – 2, ..., t – 6). Utilizou-se um máximo de três neurônios, ordem da função de Hermite igual a dois, e entrada de polarização na camada intermediária.

Para todos os métodos de inicialização aleatórios, foram treinadas dez redes para cada caso de número de entradas e calculada a média e o desvio padrão dos valores analisados. Para o método de montagem, foi utilizado um *passo* de 0,4. O valor do Índice de Projeção (IP) de cada neurônio da camada intermediária após o seu treinamento inicial, mas anterior a qualquer etapa de retro-ajuste, bem como os IPs para os neurônios após o fim do treinamento das redes (após todos os retro-ajustes) são apresentados na Tabela 5.3.

O IP de um neurônio j da camada intermediária é proporcional ao quadrado do cosseno do ângulo entre o vetor de saídas do neurônio, $\boldsymbol{\sigma}_j \in \Re^N$, e a saída desejada do neurônio, $\boldsymbol{u}_j \in \Re^N$, ou seja:

$$IP = \frac{(\boldsymbol{\sigma}_j^T \boldsymbol{u}_j)^2}{\boldsymbol{\sigma}_j^T \boldsymbol{\sigma}_j} \tag{5.1}$$

Este índice IP tem valor máximo em um, o qual implica que o neurônio foi capaz de reproduzir exatamente a saída desejada. Como o desempenho do neurônio é discretamente influenciado pela qualidade da direção de projeção, justifica-se o uso de IP para avaliar o desempenho dos métodos de inicialização dos pesos sinápticos da camada intermediária, pois são eles que definem a direção de projeção de cada neurônio.

O número de vetores candidatos à direção inicial de projeção (parâmetro *b*) utilizado para o método aleatório e também para o SAND foi inferior ao número gerado pelo método de montagem, como pode ser visto na segunda e quarta colunas da Tabela 5.3.

Alguns padrões chamam atenção especial na Tabela 5.3. Primeiramente, os valores dos IPs apresentam uma variação mínima de método para método, de forma que neste quesito não é possível apresentar um método que seja superior. Isto ocorre para todos os neurônios, antes e depois do retro-ajuste. Porém, isto acontece apesar de ter-se reduzido drasticamente o número de candidatos gerados pelos métodos aleatórios em comparação com o método de montagem, indicando que se vinha trabalhando com um número desnecessariamente elevado de candidatos (considerando a implementação computacional já existente).

Outro padrão perceptível na Tabela 5.3 é a mínima diferença apresentada entre os métodos aleatórios simples e SAND, tanto na média como no desvio padrão. Esperava-se que a utilização do SAND gerasse resultados mais estáveis (menor desvio padrão), uma vez que este promove uma distribuição mais equilibrada por parte dos vetores candidatos à direção inicial de projeção, mas isto não foi observado. O motivo provável é o ajuste que a direção inicial sofre antes de selecionar o neurônio de melhor desempenho, o que acaba compensando os efeitos da melhor distribuição dos vetores candidatos, resultando em diferenças muito pequenas nos resultados dos dois métodos.

Para explorar um pouco melhor o efeito do número de direções candidatas na qualidade final dos neurônios, geramos um novo grupo de redes neurais, variando o número de entradas e o número de vetores candidatos. Aproveitando que não foi observada diferença significativa para o primeiro neurônio no caso de haver ou não um segundo e um terceiro neurônio (justificável, uma vez que cada neurônio é treinado isolado dos outros, mesmo durante o retro-ajuste), as redes foram geradas com um único neurônio na camada intermediária (demais parâmetros iguais aos utilizados para os resultados da Tabela 5.3).

Tabela 5.3 – Comparação dos métodos de inicialização: valores do índice de projeção dos neurônios intermediários.

***England Monthly Temperature* – Índice de Projeção**							
Após introdução e treinamento do neurônio, mas antes do retro-ajuste							
Número de Entradas (*m*)	Número de Direções (*b*)	**Montagem (*passo*=0.4)**	Número de Direções (*b*)	***SAND***		**Aleatória pura**	
		Valor		*Média*	*Desvio Padrão*	*Média*	*Desvio Padrão*
		1º neurônio					
1	10	0.6598573	3	0.6598572	0.0000006	0.6598468	0.0000181
2	37	0.7936029	5	0.7930774	0.0002288	0.7928487	0.0003487
3	120	0.8540729	10	0.8538162	0.0001912	0.8537685	0.0001871
4	366	0.8776799	20	0.8774475	0.0001189	0.8773585	0.0001743
5	1010	0.8900984	40	0.8898193	0.0000939	0.8898649	0.0001471
6	2504	0.8949489	80	0.8947461	0.0001285	0.8946465	0.0001256
		2º neurônio					
1	10	0.0000335	3	0.0000219	0.0000109	0.0000081	0.0000133
2	37	0.0931882	5	0.0933043	0.0002008	0.0930918	0.0002289
3	120	0.0860614	10	0.0858637	0.0002492	0.0859085	0.0002122
4	366	0.0598084	20	0.0586809	0.0022777	0.0583613	0.0022170
5	1010	0.0509781	40	0.0501612	0.0002412	0.0486945	0.0028135
6	2504	0.0530714	80	0.0528221	0.0002038	0.0526522	0.0002651
		3º neurônio					
1	10	0.0000178	3	0.0000127	0.0000053	0.00001309	0.0000113
2	37	0.0304361	5	0.0261110	0.0048903	0.02436790	0.0044058
3	120	0.0234430	10	0.0227571	0.0020844	0.0243550	0.0008817
4	366	0.0185517	20	0.0190736	0.0007751	0.0196817	0.0010863
5	1010	0.0277699	40	0.0217029	0.0041985	0.0222048	0.0040827
6	2504	0.0364079	80	0.0299274	0.0057616	0.0276957	0.0062746
Após o treinamento completo da rede, incluindo retro-ajuste							
		1º neurônio					
1	10	0.6597456	3	0.6600584	0.0002901	0.6602388	0.0006624
2	37	0.8125156	5	0.7991801	0.0139538	0.7971025	0.0104868
3	120	0.8587299	10	0.8579871	0.0020628	0.8594765	0.0013260
4	366	0.8732637	20	0.8778102	0.0029569	0.8749920	0.0042073
5	1010	0.8877254	40	0.8859489	0.0007279	0.8892166	0.0037861
6	2504	0.8986213	80	0.8971095	0.0025569	0.8942728	0.0033119
		2º neurônio					
1	10	0.0002725	3	0.0015102	0.0028310	0.0013826	0.0028681
2	37	0.2309929	5	0.2114153	0.0253305	0.2100229	0.0228196
3	120	0.3413829	10	0.2857298	0.0584263	0.3308075	0.0350711
4	366	0.2171301	20	0.3191866	0.0270984	0.2763907	0.0563295
5	1010	0.1684507	40	0.1684156	0.0009743	0.1824641	0.0200742
6	2504	0.1232599	80	0.1325434	0.0325631	0.1538727	0.0429550
		3º neurônio					
1	10	0.0000533	3	0.0000472	0.0000276	0.0000422	0.0000381
2	37	0.0866716	5	0.1394021	0.0746974	0.1127975	0.0773655
3	120	0.1032916	10	0.0725978	0.0380418	0.1014223	0.0221486
4	366	0.0209677	20	0.1036998	0.0202708	0.0768151	0.0390251
5	1010	0.0332417	40	0.0468868	0.0095158	0.0608561	0.0349198
6	2504	0.0440512	80	0.0434269	0.0152885	0.0658631	0.0254047

Foram treinadas dez redes para cada situação. Os resultados obtidos estão nas Tabelas 5.4 (médias) e 5.5 (desvio padrão).

Tabela 5.4 – Efeito do número de candidatos nas inicializações aleatórias: médias dos IP

England Monthly Temperature – *Médias dos Índices de Projeção*							
Número de Direções (*b*)	**1 Entrada**	**2 Entradas**	**3 Entradas**	**4 Entradas**	**5 Entradas**	**6 Entradas**	**7 Entradas**
Método Aleatório com SAND							
1	0.6598484	0.7917036	0.8531162	0.8760012	0.8892947	0.8942566	0.8949023
3	0.6598569	0.7929769	0.8537267	0.8770562	0.8894890	0.8943166	0.8952311
5	0.6598576	0.7930499	0.8535852	0.8771336	0.8896281	0.8943484	0.8953435
10	0.6598577	0.7933547	0.8538421	0.8773052	0.8896544	0.8944579	0.8953947
20	0.6598577	0.7934930	0.8538749	0.8774507	0.8897249	0.8945126	0.8955620
40	0.6598577	0.7935734	0.8540381	0.8775041	0.8898502	0.8946268	0.8954909
80	0.6598577	0.7935905	0.8540170	0.8775081	0.8899715	0.8946507	0.8956685
150	0.6598577	0.7936538	0.8540857	0.8775922	0.8900462	0.8947399	0.8956697
200	0.6598577	0.7936533	0.8541470	0.8775423	0.8900641	0.8947624	0.8957175
Método Aleatório Simples							
1	0.6598387	0.7922038	0.8518749	0.8766019	0.8888731	0.8935545	0.8951748
3	0.6598500	0.7928269	0.8534408	0.8771456	0.8895258	0.8942790	0.8953105
5	0.6598560	0.7932139	0.8536166	0.8770965	0.8895254	0.8943178	0.8953246
10	0.6598576	0.7931883	0.8537739	0.8773548	0.8895603	0.8944619	0.8954729
20	0.6598577	0.7932846	0.8537987	0.8773337	0.8895842	0.8944387	0.8954862
40	0.6598577	0.7935122	0.8540481	0.8774078	0.8898704	0.8945167	0.8955893
80	0.6598577	0.7935976	0.8541129	0.8775359	0.8899463	0.8946644	0.8955602
150	0.6598577	0.7936239	0.8541264	0.8776164	0.8900014	0.8947613	0.8956608
200	0.6598577	0.7936315	0.8541371	0.8776024	0.8900009	0.8947509	0.8957184

Tabela 5.5 – Efeito do número de candidatos nas inicializações aleatórias: desvio padrão dos IP

***England Monthly Temperature* – Desvio Padrão dos Índices de Projeção**							
Número de Direções (*b*)	**1 Entrada**	**2 Entradas**	**3 Entradas**	**4 Entradas**	**5 Entradas**	**6 Entradas**	**7 Entradas**
Método Aleatório com SAND							
1	1.0788955×10^{-5}	0.0014019	0.000668	0.000279	0.0002868	0.0001215	0.0012793
3	0.0961343×10^{-5}	0.0004210	0.000187	0.000296	0.0000840	0.0001119	0.0001543
5	0.0184113×10^{-5}	0.0003562	0.000309	0.000242	0.0001519	0.0001402	0.0000917
10	0.0085349×10^{-5}	0.0001911	0.000219	0.000143	0.0001547	0.0001105	0.0000697
20	0.0006016×10^{-5}	0.0000859	0.000125	0.000196	0.0001737	0.0001316	0.0001091
40	0.0001727×10^{-5}	0.0000923	0.000099	0.000084	0.0001136	0.0001409	0.0001000
80	0.0000113×10^{-5}	0.0000349	0.000053	0.000066	0.0001001	0.0001206	0.0001065
150	0.0000069×10^{-5}	0.0000223	0.000076	0.000063	0.0001266	0.0000745	0.0000871
200	0.0000148×10^{-5}	0.0000150	0.000069	0.000093	0.0000721	0.0000899	0.0000745
Método Aleatório Simples							
1	2.3979994×10^{-5}	0.0012127	0.0038139	0.000724	0.0016005	0.0014880	0.0001369
3	1.7569206×10^{-5}	0.0002793	0.0001837	0.000282	0.0001344	0.0000301	0.0001199
5	0.3550660×10^{-5}	0.0003571	0.0002673	0.000197	0.0001281	0.0000832	0.0000595
10	0.0274883×10^{-5}	0.0002699	0.0002232	0.000206	0.0001548	0.0001005	0.0002067
20	0.0025934×10^{-5}	0.0002217	0.0001866	0.000141	0.0001139	0.0001412	0.0001707
40	0.0016821×10^{-5}	0.0001058	0.0001006	0.000124	0.0001013	0.0001039	0.0001555
80	0.0000737×10^{-5}	0.0000529	0.0000573	0.000074	0.0001386	0.0001536	0.0000925
150	0.0000303×10^{-5}	0.0000326	0.0000766	0.000043	0.0000582	0.0000549	0.0001028
200	0.0000402×10^{-5}	0.0000426	0.0000681	0.000059	0.0000767	0.0000691	0.0000541

Percebe-se que as médias dos valores do IP estabilizam-se para valores comparativamente reduzidos de direções candidatas. Este mesmo comportamento é observado para outras funções; na Figura 5.1 podemos ver as curvas para a média dos valores dos IPs de RNAs de um neurônio na camada intermediária treinadas para aproximar as cinco funções bidimensionais de HWANG *et al*. (1994) (ver Apêndice A). Foram treinadas dez redes para cada função e cada método de inicialização, usando ordem de Hermite igual a 2. Portanto, podemos considerar adequada a utilização de um número baixo (entre 10 e 20) de direções na inicialização dos pesos dos neurônios da camada intermediária.

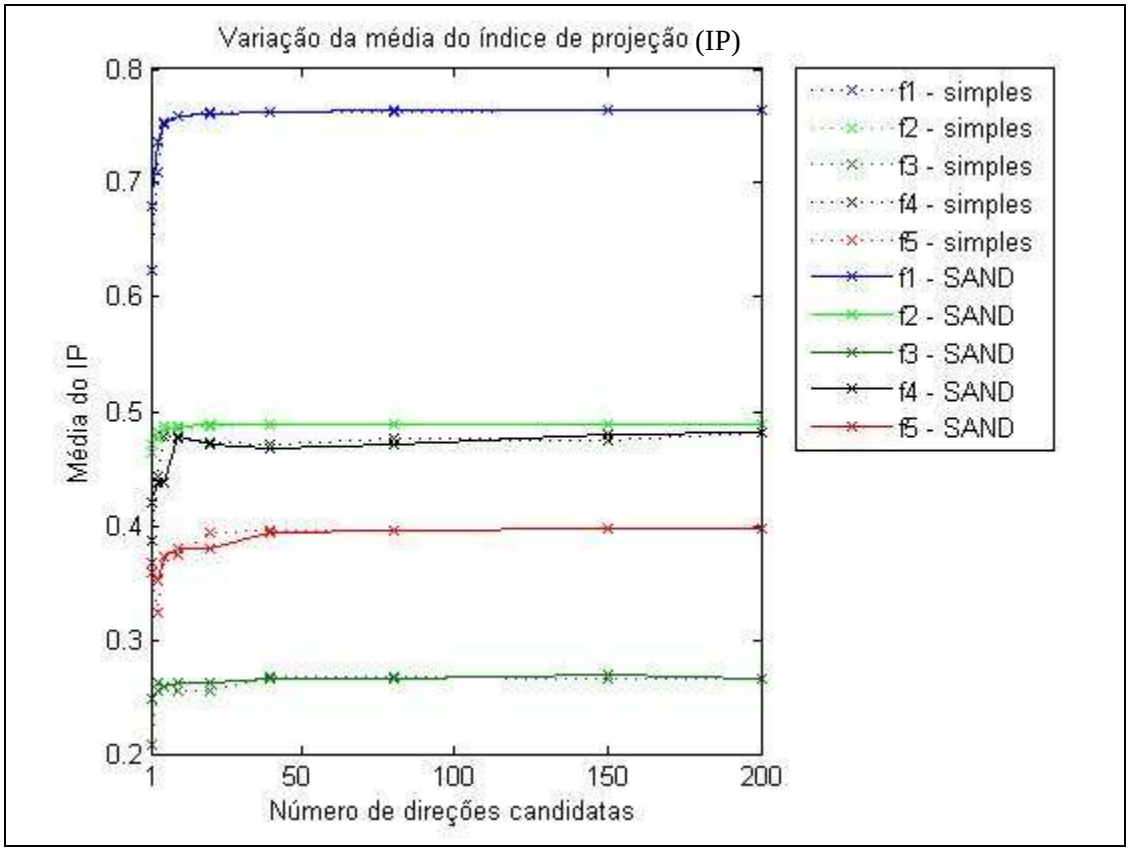

Figura 5.1 – Média dos valores dos índices de projeção (IP) para RNAs treinadas para aproximar as funções bidimensionais: $f^{(1)}$ (função de interação simples), $f^{(2)}$ (função radial), $f^{(3)}$ (função harmônica), $f^{(4)}$ (função aditiva) e $f^{(5)}$ (função de interação complicada), utilizando os métodos de inicialização dos pesos da camada intermediária aleatório simples e aleatório com SAND.

Os diferentes valores de IP para as cinco funções são esperados, ao indicar o quanto da tarefa de aproximação é passível de ser executada por um único neurônio na camada intermediária, em cada caso.

5.3 Seleção de variáveis de entrada para o aprendizado por busca de projeção

Para averiguar o impacto do procedimento de seleção de entradas na qualidade das RNAs geradas pelo PPL, foram realizados ensaios de execução do WIS-PPL (*Wrapper for Input Selection in Projection Pursuit Learning*) visando observar se ocorreria convergência para um subconjunto reduzido de variáveis, o grau de sobre-ajuste apresentado pela RNA final obtida e, por último, comparações entre as redes neurais geradas e outros métodos da literatura, incluindo o PPL sem seleção.

5.3.1 Convergência e análise paramétrica do WIS-PPL

Com o intuito de observar o efeito das taxas de mutação e de recombinação, bem como do número de indivíduos na população na qualidade do resultado final, o WIS-PPL foi executado para selecionar RNAs que aproximassem as funções bidimensionais $f^{(1)}$ e $f^{(2)}$ (função de interação simples e função radial,respectivamente; ver Apêndice A), utilizando ordem máxima para as funções de Hermite igual a dois e redes com máximo de três neurônios na camada intermediária.

Os parâmetros do algoritmo genético foram variados conforme apresentado na Tabela 5.6 e o treinamento via PPL foi realizado com um conjunto de 200 instâncias; a validação (utilizada para a determinação do *fitness* de cada modelo) e o teste (utilizado apenas para a comparação final dos resultados) foram feitos com conjuntos independentes de 50 instâncias cada. Para as duas funções, foi fornecido um conjunto de seis entradas $\{\mathbf{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5$ e $\boldsymbol{x}_6\}$, sendo as duas primeiras, $\boldsymbol{x}_1$ e $\boldsymbol{x}_2$, as entras originais das funções; $\boldsymbol{x}_3$ e $\boldsymbol{x}_4$ são entradas aleatórias irrelevantes; e $\boldsymbol{x}_5$ e $\boldsymbol{x}_6$ são versões ruidosas de $\mathbf{x}_1$ e $\boldsymbol{x}_2$. Cada combinação de parâmetros foi treinada cinco vezes para cada função. A Tabela 5.6 apresenta os resultados obtidos para dez séries de execuções do WIS-PPL.

O critério de comparação utilizado foi o erro quadrático médio normalizado (NMSE, do inglês *Normalized Mean Square Error*):

$$NMSE = \frac{1}{\sigma^2 N} \sum_{i=1}^{N} (x_i - \hat{x}_i)^2, \tag{5.2}$$

onde x_i é o valor da i-ésima amostra (i = 1, ..., N); $\hat{x}_i$ é a estimativa do valor da i-ésima amostra e σ^2 é a variância das amostras.

Os valores do NMSE de teste obtidos para os modelos selecionados em cada conjunto de execuções do WIS-PPL foram comparados e o melhor valor é apresentado sublinhado na Tabela 5.6. Média e desvio padrão dos erros apresentados para cada combinação de parâmetros também são fornecidos.

Tabela 5.6 – NMSE de teste das RNAs evoluídas para aproximar $f^{(1)}$ e $f^{(2)}$ com algumas variações dos parâmetros do algoritmo genético

Execução	N. indivíduos = 6 Prob. Recom. = 90% Prob. Mutação = 40%	N. indivíduos = 10 Prob. Recom.= 90% Prob. Mutação = 20%	N. indivíduos = 10 Prob. Recom. = 90% Prob. Mutação = 40%	N. indivíduos = 10 Prob. Recom. = 70% Prob. Mutação = 40%	N. indivíduos = 14 Prob. Recom. = 90% Prob. Mutação = 40%
Função de interação simples $f^{(1)}$					
1	0.0000246	0.0000116	0.0000194	0.0000217	0.0000252
2	0.0000106	0.0000144	0.0000506	0.0000276	0.0000224
3	0.0000318	0.0000347	0.0000147	0.0000159	0.0000182
4	0.0000341	0.0000081	0.0000195	0.0000306	0.0000144
5	0.0000266	0.0000271	0.0000142	0.0000185	0.0002380
6	0.0000289	0.0000251	0.0000159	0.0000081	0.0000080
7	0.0000273	0.0000123	0.0000159	0.0000053	0.0000021
8	0.0000136	0.0000452	0.0000211	0.0000378	0.0000139
9	0.0000169	0.0000237	0.0000326	0.0000129	0.0000153
10	0.0000236	0.0000199	0.0000220	0.0000081	0.0000012
Média	***0.0000238***	***0.0000222***	***0.0000226***	***0.0000186***	***0.0000144***
Desvio	***0.0000077***	***0.0000115***	***0.0000112***	***0.0000108***	***0.0000085***
	Função radial $f^{(2)}$				
1	0.0055452	0.0090161	0.0045863	0.0055332	0.0059472
2	0.0106595	0.0054591	0.0054779	0.0068259	0.0042959
3	0.0050966	0.0086277	0.0056762	0.0080458	0.0068528
4	0.0063518	0.0062799	0.0057841	0.0085062	0.0102374
5	0.0056063	0.0053069	0.0058385	0.0074560	0.0069458
6	0.0051022	0.0060551	0.0052179	0.0055193	0.0052562
7	0.0060591	0.0053494	0.0051626	0.0108734	0.0052171
8	0.0113631	0.0058852	0.0050880	0.0053437	0.0052811
9	0.0053284	0.0053250	0.0052859	0.0054828	0.0053781
10	0.0047399	0.0059671	0.0052387	0.0057442	0.0055667
Média	**0.0065852**	**0.0063272**	**0.0053356**	**0.0069331**	**0.0060978**
Desvio	**0.0023851**	**0.0013613**	**0.0003755**	**0.0018112**	**0.0016548**

Observamos que a combinação {14 indivíduos por geração, probabilidade de recombinação de 90%, probabilidade de mutação de 40%} forneceu para $f^{(1)}$ o melhor NMSE em um número maior de execuções, além de apresentar a menor média e um desvio padrão bem reduzido; e que a combinação {10 indivíduos por geração, probabilidade de recombinação de 90%, probabilidade de mutação de 40%} forneceu para $f^{(2)}$ os melhores resultados.

Percebe-se também que o impacto do ajuste dos parâmetros dentro dos intervalos testados não foi muito significativo, de forma que a combinação {10 indivíduos por geração, probabilidade de recombinação de 90%, probabilidade de mutação de 40%} (a que resulta em um menor número de treinamentos por geração entre as duas combinações "vencedoras") foi adotada como padrão nos experimentos seguintes.

5.3.2 Comparações do WIS-PPL com PPL e com MLP

Para observar se o processo de seleção de entradas realizado pelo WIS-PPL está resultando na obtenção de redes parcimoniosas, com ganho de acurácia e de capacidade de generalização, o algoritmo foi executado 10 vezes para selecionar um subconjunto de entradas e treinar um preditor para a série temporal *England Monthly Temperature* (ver Apêndice A).

A série foi dividida em 2.731 instâncias de treinamento, 296 instâncias de validação e outras 296 instâncias de teste, com doze entradas candidatas a serem selecionadas e uma saída. As entradas candidatas correspondem aos atrasos consecutivos da saída (t – 1, t – 2, ..., t – 12) de forma a incluírem o período de um ano completo. Utilizou-se o NMSE como critério de comparação.

Os parâmetros utilizados foram nMAX = 3, P = 4, taxa de recombinação de 90%, taxa de mutação de 40%, número de gerações igual a 50, e uma população de dez indivíduos por geração.

Inicialmente, uma bateria de treinamentos preliminares foi realizada para ajustar os limiares de treinamento. O limiar que controla o ciclo de treinamento de cada neurônio (limiar2) e o limiar que controla o retro-ajuste (limiar3) foram variados nas combinações apresentadas na Tabela 5.7. Foram observados a média e o desvio padrão do NMSE sobre o

conjunto de validação, calculados sobre dez treinamentos independentes, para observar o efeito dos limiares no treinamento. No caso destes testes preliminares envolvendo os limiares de treinamento, as RNAs foram treinadas fornecendo as doze entradas para o PPL, sem seleção.

Tabela 5.7 – Efeito do ajuste dos limiares de treinamento no NMSE de validação para série *England Monthly Temperature*

	Limiar2 = 10^{-4} Limiar3 = 10^{-4}	**Limiar2 = 10^{-3} Limiar3 =10^{-4}**	**Limiar2 = 10^{-3} Limiar3 = 10^{-3}**	**Limiar2 = 10^{-2} Limiar3 = 10^{-3}**	**Limiar2 = 10^{-2} Limiar3 = 10^{-2}**
Média do NMSE de Validação	0.0912870	0.0844449	0.0846800	0.08632698	0.0847848
Desvio Padrão do NMSE de Validação	0.0033729	0.0034537	0.0019894	0.0011404	0.0030180

A combinação escolhida para as simulações seguintes foi limiar2 = 10^{-3}, limiar3 = 10^{-3}, devido à melhor combinação de média e desvio padrão obtidos.

O WIS-PPL foi executado dez vezes e as redes obtidas foram comparadas com RNAs geradas pelo PPL sem seleção, por *perceptrons* multicamadas (MLP) utilizando todas as entradas e por MPLs utilizando apenas as entradas selecionadas pelo WIS-PPL. Os MLPs foram treinados com o mesmo número de neurônios na camada intermediária que as RNAs geradas pelo PPL (três) utilizando um algoritmo de treinamento de segunda ordem por dez mil iterações. Os NMSE obtidos sobre o conjunto de dados de teste estão na Tabela 5.8. O PPL sem seleção de entradas (ou seja, utilizando as doze entradas) e os MLPs foram treinados por dez vezes.

A comparação da média do NMSE de treinamento obtido pelo WIS-PPL com a média obtida pelo PPL sem seleção de entradas aponta para um ganho de acurácia após o processo de seleção. Também observamos que o PPL foi capaz de obter uma melhor aproximação que as redes MLPs para uma mesma dimensão de rede.

Tabela 5.8 – Resultados do WIS-PPL para a série *England Monthly Temperature,* comparado com MLP

Execução WIS-PPL	**cromossomo**	***NMSE de teste***
1	111000101111	0.1026830
2	111110111011	0.0937182
3	101011110111	0.0919974
4	101010100011	0.0937254
5	110110000011	0.0963105
6	101111100011	0.0939698
7	100011100111	0.0913234
8	111111101110	0.0957601
9	110110100110	0.0936591
10	101111100111	0.0947112
Média		**0.0947858**
Desvio Padrão		**0.0031554**
PPL, 12 entradas	**Média**	0.0970146
	Desvio Padrão	0.0039989
MLP, 12 entradas; 10000 iterações	**Média**	0.0978385
	Desvio Padrão	0.0000129
MLP, entradas selecionadas na execução 7; 10000 iterações	**Média**	0.0970649
	Desvio Padrão	0.0006349

5.3.3 Comparações utilizando a série temporal *Yearly Sunspot Number*

Para estas comparações, o WIS-PPL foi executado dez vezes com taxa de mutação de 0,4, taxa de recombinação de 0,9, com população de dez indivíduos por geração durante 50 gerações. Utilizou-se o PPL com um máximo de três neurônios na camada intermediária, ordem de Hermite igual a dois, camada intermediária sem entradas de polarização.

Os resultados obtidos são comparados com os apresentados em MARRA & MORABITO (2006), que apresentam comparação de performance entre diversos modelos de preditores para a série *Sunspot*. Os vários modelos são: Auto-Regressivo (AR), *Weight Elimination Feedforward Network* (Wnet), *Dynamical Recurrent Neural Network* (DRNN), *Soft Weight Sharing Network* (SSNet), *Scale Neural Network* (ScaleNet), o método de predição por comitê de Wan (COMM), redes recorrentes treinadas com o algoritmo *Backpropagation Through Time* (BPTT), BBTT construtivo (CBPTT), *Violation Guided Back Propagation* (VGBP), e redes recorrentes de Elman com dados pré-processados (DDEN) (MARRA & MORABITO, 2006).

Para realizar estas comparações, foram utilizados os dados dos anos de 1700 a 1994, normalizados de forma a apresentarem média zero e desvio padrão unitário. Estes foram

divididos em um conjunto de aprendizado, indo de 1700 a 1920 e três conjuntos de teste: de 1921 a 1955 (Test1), de 1956 a 1979 (Test2), e de 1980 a 1994 (Test3).

Os dados de aprendizado foram organizados em instâncias compostas de 11 entradas (de forma a permitir que um período completo da série esteja disponível como possível entrada) e uma saída, e divididos aleatoriamente em um conjunto de treinamento de 262 instâncias e um conjunto de validação de 22 instâncias. O critério de comparação novamente é o erro quadrático médio normalizado.

A Tabela 5.9 apresenta os resultados para dez execuções. Os cromossomos são a codificação binária dos subconjuntos selecionados com cada *bit* representando, da esquerda para a direita, os atrasos de (t – 1) a (t –11). Também são apresentados os resultados obtidos treinando-se uma RNA usando o PPL com todas as 11 possíveis entradas.

Notamos que a média dos NMSEs obtidos pelo WIS-PPL é melhor em dois casos que a obtida pelo PPL sem seleção, configurando a ocorrência de ganho no processo de seleção. Porém houve uma variação muito grande no cromossomo selecionado indicando que algumas das entradas podem carregar informação semelhante ou mesmo que o PPL esteja obtendo sucesso em cancelar efeitos ruidosos destas entradas.

Tabela 5.9 –Erros de teste para dez execuções do WIS-PPL e PPL sem seleção de entradas

RNA	**Cromossomo selecionado**	**NMSE**		
		Test1	**Test2**	**Test3**
Execução 1	11011000001	0.058	0.074	0.048
Execução 2	11010110001	0.062	0.035	0.060
Execução 3	11011110001	0.048	0.071	0.058
Execução 4	11011010001	0.046	0.060	0.035
Execução 5	11011000000	0.057	0.082	0.045
Execução 6	11001101100	0.044	0.075	0.062
Execução 7	11111001001	0.043	0.057	0.056
Execução 8	11000001001	0.060	0.084	0.086
Execução 9	11000001001	0.056	0.091	0.053
Execução 10	11001111010	0.053	0.038	0.044
Média		***0.053***	***0.067***	***0.055***
Desvio Padrão		***0.007***	***0.019***	***0.014***
PPL com 11 entradas	11111111111	**0.073**	**0.048**	**0.063**

Como uma forma de lidar com esta variação, foi determinado um “cromossomo de compromisso” a partir do voto majoritário dos dez cromossomos selecionados pelo WIS-PPL (cada entrada é mantida neste cromossomo de compromisso desde que tenha sido

selecionada por pelo menos metade das execuções). Foram treinadas cinco redes a partir deste subconjunto. Os resultados estão nas Tableas 5.10 e 5.11.

Tabela 5.10 – Cromossomo de voto majoritário

Entrada	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
Número de Seleções	10	10	1	6	7	4	4	5	1	1	7
Voto majoritário	1	1	0	1	1	0	0	1	0	0	1

Tabela 5.11 – Erros de teste para o subconjunto de entradas do voto majoritário

Voto Majoritário (cinco treinamentos independentes)	**NMSE**		
	Test1	**Test2**	**Test3**
Média	*0.047*	*0.066*	*0.051*
Desvio Padrão	*0.006*	*0.009*	*0.005*

Os erros apresentados são inferiores à média das execuções do WIS-PPL, o que confirma a viabilidade da utilização do voto majoritário.

Na Tabela 5.12, apresentamos a comparação entre os resultados do WIS-PPL e os vários métodos apresentados em MARRA & MORABITO (2006).

Tabela 5.12 – Comparação entre preditores para a série *Sunspot*

Modelo	**NMSE**		
	Test1	**Test2**	**Test3**
AR(12)	0.427	0.966	0.238
Wnet	0.086	0.350	0.219
SSNet	0.077	N/A	N/A
DRNN	0.091	0.273	N/A
COMM	0.065	0.240	0.148
ScaleNet	0.057	0.130	N/A
BPTT	0.084	0.300	N/A
CBPTT	0.092	0.251	N/A
VGBP	0.033	0.052	0.033
DDEN	0.043	0.080	0.028
PPL com 11 entradas	*0.073*	*0.048*	*0.063*
WIS-PPL, voto majoritário	***0.047***	***0.066***	***0.051***
WIS-PPL, melhor	**0.046**	**0.060**	**0.035**

Observamos que os resultados obtidos pelo WIS-PPL são plenamente competitivos com os obtidos por outros modelos de preditores, sendo ultrapassado claramente apenas pelo VGBP e estando muito próximo do DDEN. Mesmo o PPL sem seleção de entradas compara-se positivamente contra a maioria dos outros modelos.

As Figuras 5.2 e 5.3 apresentam o comportamento dos preditores treinados pela Execução 4 e pelo PPL sem seleção (11 entradas) respectivamente, ilustrando o ganho de acurácia observado pelo WIS-PPL,

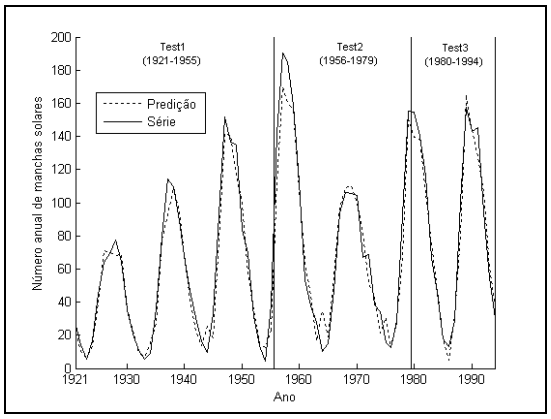

Figura 5.2 – Preditor treinado na execução 4 do WIS-PPL

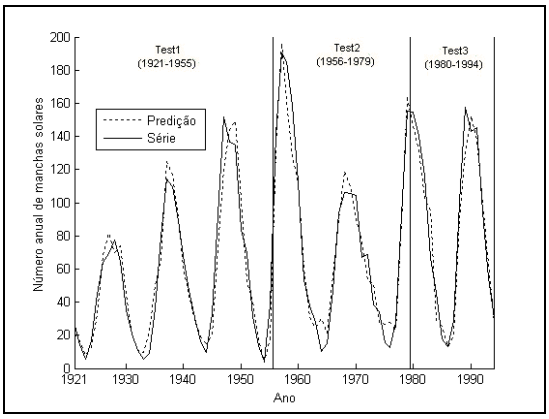

Figura 5.3 – Preditor treinado pelo PPL sem seleção de entrada

5.3.4 Análise de sobre-ajuste

Conforme discutido no Capítulo 3, o emprego de *wrappers* para seleção de variáveis freqüentemente apresenta problemas relacionados a sobre-ajuste aos dados de treinamento. A utilização, no algoritmo genético, de um conjunto de dados independente dos de treinamento para a determinação do *fitness* de cada subconjunto (dados de validação) reduz a tendência de sobre-ajuste, mas não a elimina. Por isso, é importante utilizar um terceiro conjunto de dados (dados de teste) como critério de comparação entre os modelos gerados pelo *wrapper* e outros modelos, como foi realizado nos experimentos acima, pois isto permite uma avaliação não viciada da acurácia e capacidade de generalização das RNAs obtidas.

Para observar qual o grau de sobre-ajuste apresentado pelo WIS-PPL, foram preparados gráficos com a evolução dos NMSEs de treinamento, validação e teste. Alguns destes gráficos podem ser observados nas Figuras 5.4 a 5.13, abaixo. Tipicamente, foi observado um grau reduzido de sobre-ajuste, como nos casos das Figuras 5.4, 5.6, 5.8 e 5.11. Porém, casos de sobre-ajuste significativo (Figuras 5.9, 5.10, e 5.13) e sem sobre-ajuste (Figuras 5.5, 5.7 e 5.12) também foram encontrados. A ocorrência de sobre-ajuste é

percebida observando o comportamento do erro de teste: o aumento deste quando da diminuição do erro de validação (que determina o *fitness*) indica perda de capacidade de generalização.

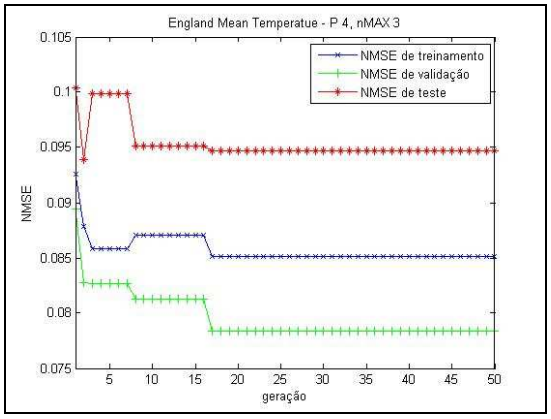

Figura 5.4 – Série *England Mean Temperature*, P = 4, nMAX = 3

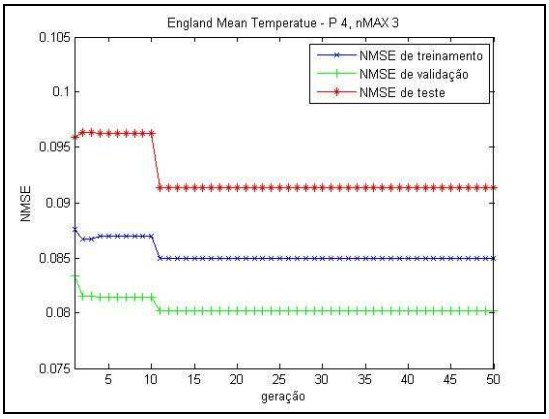

Figura 5.5 – Série *England Mean Temperature*, P = 4, nMAX = 3

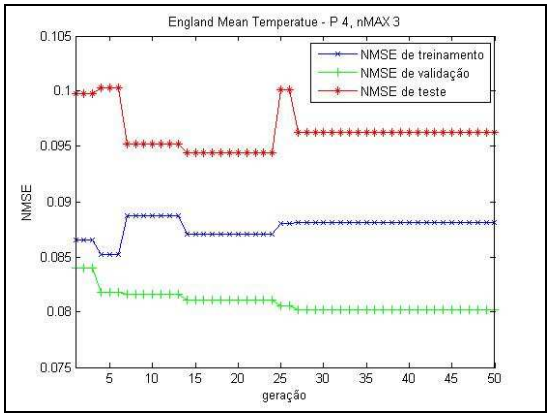

Figura 5.6 – Série *England Mean Temperature*, P = 4, nMAX = 3

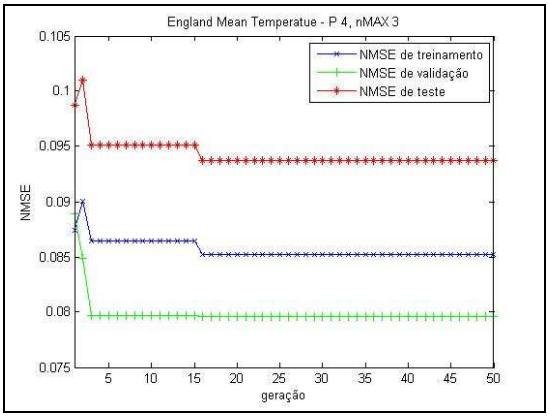

Figura 5.7 – Série *England Mean Temperature*, P = 4, nMAX = 3

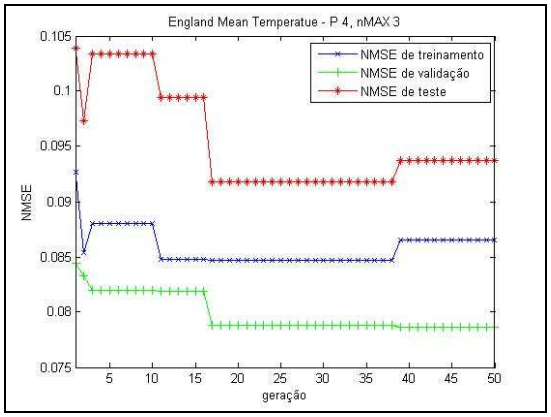

Figura 5.8 – Série *England Mean Temperature*, P = 4, nMAX = 3

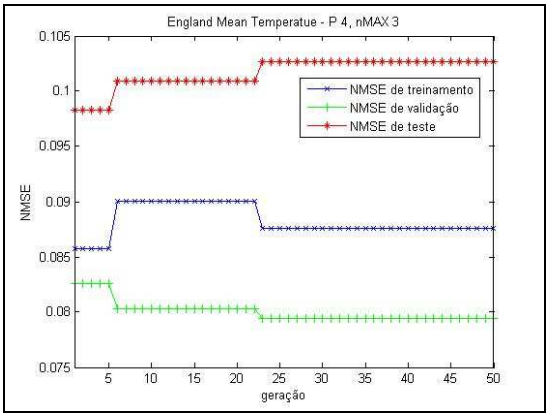

Figura 5.9 – Série *England Mean Temperature*, P = 4, nMAX = 3

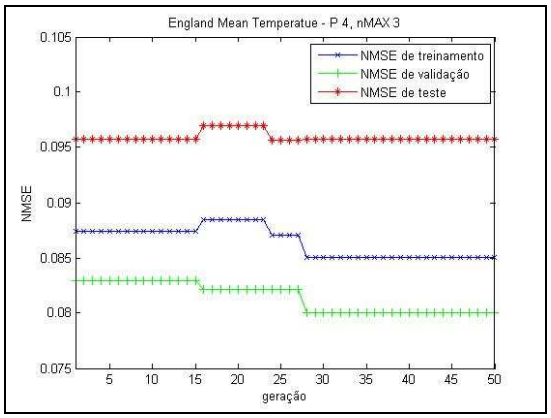

Figura 5.10 – Série *England Mean Temperature*, P = 4, nMAX = 3

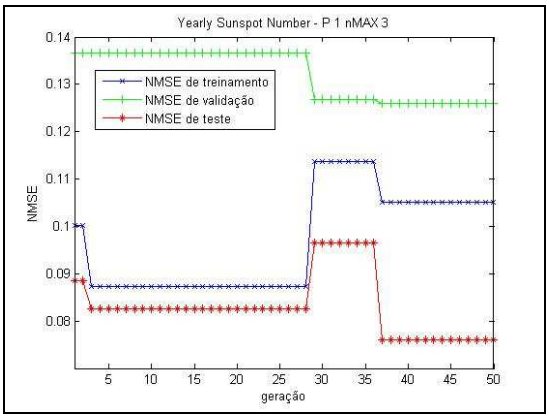

Figura 5.11 – Série *Yearly Sunspot Number*, P = 1, nMAX =3

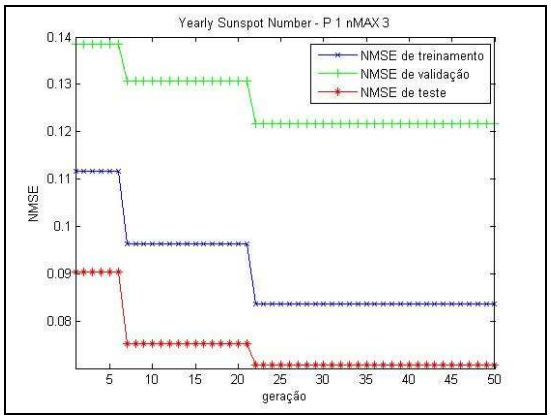

Figura 5.12 – Série *Yearly Sunspot Number*, P = 1, nMAX =3

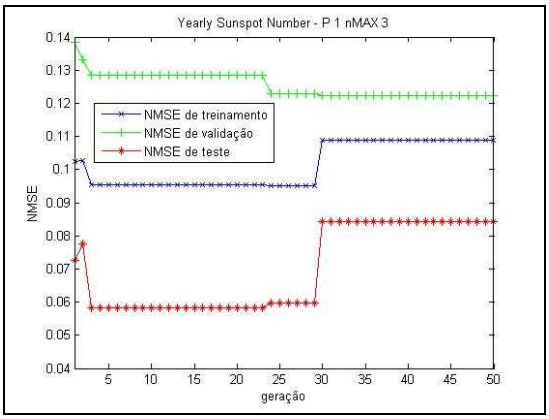

Figura 5.13 – Série *Yearly Sunspot Number*, P = 1, nMAX =3

Capítulo 6

Conclusão

Conforme já apresentado, metodologias para projeto de RNAs parcimoniosas que possam ser implementadas em sistemas com reduzido poder de processamento e limitado espaço de memória, sem sacrificar acurácia ou generalização, são, em muitos casos, uma necessidade atual para viabilizar economicamente o uso de RNAs como soluções para problemas de diversas áreas da engenharia. Este fato justifica a utilização de algoritmos de treinamento mais custosos e o emprego de técnicas auxiliares, como as de seleção de variáveis, no desenvolvimento de RNAs, desde que o custo de projeto adicional resulte em economias de implementação.

O tempo de aprendizado do algoritmo WIS-PPL é consideravelmente alto, devido à necessidade de se treinar múltiplas redes para cada problema, mas a possibilidade de obter modelos de predição parcimoniosos e resultados melhores do que os que seriam obtidos sem a seleção de variáveis apontam favoravelmente para o emprego deste na síntese de modelos via RNAs.

Os resultados obtidos para a série *Yearly Sunspot Number* indicam que o PPL é bastante competitivo em aplicações de síntese de preditores de séries temporais com uma janela de atrasos moderada, e o uso de um *wrapper* associado aumentou ainda mais esta competitividade. Estes resultados vêm se unir a outros anteriores obtidos pelo nosso grupo de pesquisa (VON ZUBEN & NETTO, 1995, 1997; VON ZUBEN, 1996; GONÇALVES *et al.*, 1998; DE CASTRO *et al.*, 1999; IYODA, 2000; LIMA, 2000; MELEIRO *et al.*, 2008) e por outros grupos (MAECHLER *et al.*, 1990; HWANG *et al.*, 1994; KWOK & YEUNG, 1996, 1997) que mostram o poder e flexibilidade do aprendizado por busca de projeção como ferramenta computacional.

No tocante às melhorias diretas introduzidas na implementação do algoritmo de PPL disponível junto ao grupo de pesquisa, a substituição do método de inicialização resultou em um algoritmo mais ágil, devido à redução do número de vetores candidatos à direção

inicial de projeção, implicando em menos tempo gasto no treinamento parcial destes vetores, o qual é realizado anteriormente à escolha da melhor direção. Acrescenta-se a isso o fato de não haver mais limitações arbitrárias no número máximo de entradas permitidas. Já a introdução das entradas de polarização na camada intermediária resultou em um aumento da flexibilidade dos neurônios artificiais do PPL.

Assim, podemos concluir que os objetivos principais propostos para este trabalho, conforme apresentados no Capítulo 1, foram satisfatoriamente cumpridos. No entanto várias frentes ainda permanecem aberta a novas investigações, dando margem a futuros aperfeiçoamentos tanto no WIS-PPL, como no próprio PPL.

6.1 Perspectivas futuras

Entre as principais limitações do PPL que foram levantadas ao longo do texto (ver Capítulo 1), a única na qual ainda não conseguimos nenhum avanço de fato é a baixa eficiência no treinamento de RNAs com múltiplas saídas. Esta é uma frente que permanece desafiadora e cuja solução (ou apenas amenização) pode levar a várias novas possibilidades de aplicação do PPL. É evidente que, na ausência de um melhor tratamento para múltiplas saídas, a solução mais indicada por hora é recorrer a uma RNA para cada saída.

Um aspecto não abordado nestes estudos foi a possibilidade de implementar uma etapa de poda de neurônios no PPL. A etapa de poda afetaria neurônios cuja participação tenha se tornado muito reduzida durante o retro-ajuste, potencialmente possibilitando RNAs ainda mais parcimoniosas. Também poderia ser promissor incorporar no PPL alguns dos novos resultados derivados de técnicas não-paramétricas de aproximação de funções, como modelagem hierárquica.

No tocante às alterações introduzidas em nossa implementação do PPL, alguns pontos ainda merecem estudos posteriores. Era esperado que a utilização do SAND para equilibrar a distribuição das direções candidatas na inicialização dos pesos sinápticos da camada intermediária tivesse resultado em vantagem sobre a geração aleatória simples. Isto não foi observado, sendo a explicação mais provável o efeito do treinamento parcial sofrido pelos neurônios após a proposta das direções candidatas e antes da escolha de uma delas. Um estudo mais aprofundado deste efeito poderia talvez resultar em métodos mais

eficientes de inicialização. Outra possibilidade é realizar estudos que avaliem melhor a capacidade de generalização da RNA gerada com relação à ordem dos polinômios de Hermite ou *splines* suavizantes utilizados.

Quanto às entradas de polarização, os experimentos mostram ganho de flexibilidade. Porém, ficou em aberto um estudo comparativo das diferenças de capacidade de aproximação universal e convergência entre o algoritmo de KWOK & YEUNG (1996) e o de VON ZUBEN (1996).

Outra perspectiva promissora para o PPL seria adaptar as metodologias construtivas para ambientes cujas propriedades estatísticas sejam variantes no tempo, como no caso de *data streams*. Uma possibilidade de adaptação seria, por exemplo, manter as funções de expansão ortogonais obtidas em um treinamento inicial como "bases" estáticas, mas utilizar composições dinâmicas das mesmas, permitindo que a rede se adapte ao momento sem a necessidade de recorrer ao custoso processo de busca de projeção em tempo real.

O WIS-PPL também, e principalmente, fornece perspectivas de melhorias futuras. A substituição do GA simples utilizado por outros mecanismos de busca pode talvez resultar em uma convergência mais rápida para o subconjunto mais adequado de entradas. É possível também introduzir outros elementos da arquitetura, como a ordem das funções de Hermite, no processo evolutivo. Alternativas, como a validação cruzada, podem ser introduzidas no cálculo do *fitness* de cada subconjunto, ajudando a reduzir a chance de sobre-ajuste. A possibilidade de sobre-ajuste, juntamente com a demanda por recursos computacionais, apresentam-se como vulnerabilidades do algoritmo.

O emprego do WIS-PPL em outros tipos de problemas, como problemas de classificação de padrões, pode abrir outras perspectivas de estudo. Um estudo mais aprofundado das circunstâncias que levam à alta variabilidade nos subconjuntos de entradas selecionados em execuções diferentes do algoritmo para um mesmo problema pode resultar em uma compreensão mais profunda de como utilizar melhor as propriedades inatas de seleção do PPL.

Por fim, outros paradigmas de seleção de variáveis poderiam ser testados junto ao PPL, seja introduzindo uma etapa explícita de seleção de variáveis (poda de entradas) no algoritmo, seja baseando a avaliação dos subconjuntos no retreinamento de uma única RNA, ao invés de considerar o treinamento de múltiplas redes, ou até mesmo a

identificação e emprego de filtros apropriados como um passo de pré-processamento dos dados, aliviando a tarefa posterior do *wrapper*.

Apêndice A

Funções Utilizadas nos Experimentos

A.1 Problemas de regressão bidimensionais

HWANG *et al.* (1994) apresentam um conjunto de cinco funções bidimensionais envolvendo diferentes tipos de interação das variáveis de entrada, visando avaliar a performance do PPL. KWOK & YEUNG (1996) utiliza o mesmo conjunto de funções em algumas simulações do PPL com polarização da camada intermediária. Estas funções também são utilizadas em alguns testes e comparações apresentados nesta dissertação e são:

- função de interação simples:

$$f^{(1)}(x_1, x_2) = 24.234\left[(x_1 - 0.4)(x_2 - 0.6) + 0.36\right] \tag{A.1}$$

- função radial:

$$f^{(2)}(x_1, x_2) = 24.234\left[r^2(0.75 - r^2)\right], \quad r^2 = (x_1 - 0.5)^2 + (x_2 - 0.5)^2 \tag{A.2}$$

- função harmônica:

$$f^{(3)}(x_1, x_2) = 42.659\left[\frac{2 + x_1}{20} + \mathrm{Re}\left(z^5\right)\right], \quad z = x_1 + jx_2 - 0.5(1 + j) \tag{A.3}$$

- função aditiva:

$$f^{(4)}(x_1, x_2) = 1.3356\left[1.5(1 - x_1) + e^{2x_1 - 1}\,\mathrm{sen}\left(3\pi(x_1 - 0.6)^2\right) + e^{3(x_2 - 0.5)}\,\mathrm{sen}\left(4\pi(x_2 - 0.9)^2\right)\right] \tag{A.4}$$

- função de interação complicada:

$$f^{(5)}(x_1, x_2) = 1.9\left[1.35 + e^{x_1 - x_2}\,\mathrm{sen}\left(13(x_1 - 0.6)^2\right)\cdot \mathrm{sen}\left(7x_2\right)\right] \qquad \text{(A.5)}$$

Em todas as funções, $(x_1,x_2) \in [0,1]^2$. Para alguns testes, definiu-se também uma variável aleatória $x_3 \in [0,1]$ (com distribuição uniforme) e irrelevante para os problemas, bem como uma variável $x_4 = 0{,}7x_1 + 0{,}3\,x_3$.

A.2 Yearly Sunspot Number

A série temporal *Yearly Sunspot Number* é uma série histórica, tendo seus valores registrados desde o ano de 1700 d.C até o presente. Ela registra o número de manchas negras observadas na superfície do Sol (manchas solares) a cada ano. Trata-se de uma série periódica, com o período aproximado de 11 anos (MARRA & MORABITO, 2006), muito utilizada na literatura de preditores de séries temporais.

A série até o ano de 1994 pode ser observada na Figura A.1 abaixo. Os valores para a série foram obtidos de SIDC-TEAM (2008). Foram utilizados os valores dos anos de 1700 a 1994 (para fins de comparação com a literatura) normalizados com média zero e desvio padrão unitário.

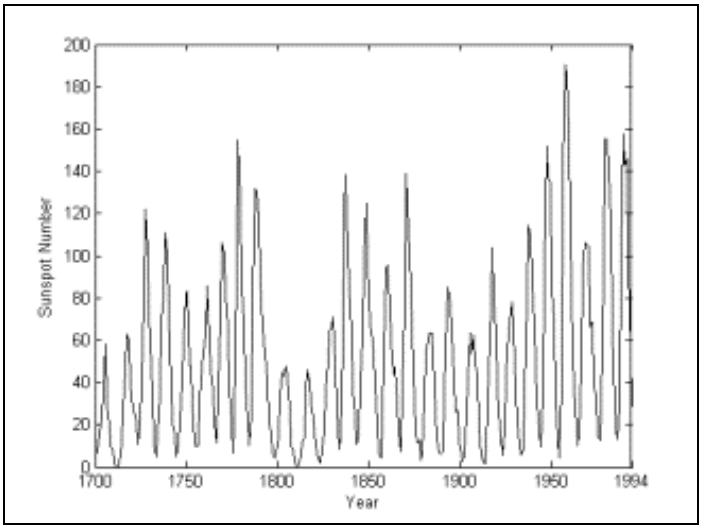

Figura A.1 – Série temporal *Yearly Sunspot Number* (1700–1994)

A.3 England Monthly Temperature

A série temporal *England Monthly Temperature* (HIPEL & MCLEOD, 1994**)** registra a temperatura média mensal na Inglaterra entre os anos de 1723 e 1970. A série foi normalizada para média zero e desvio padrão unitário.

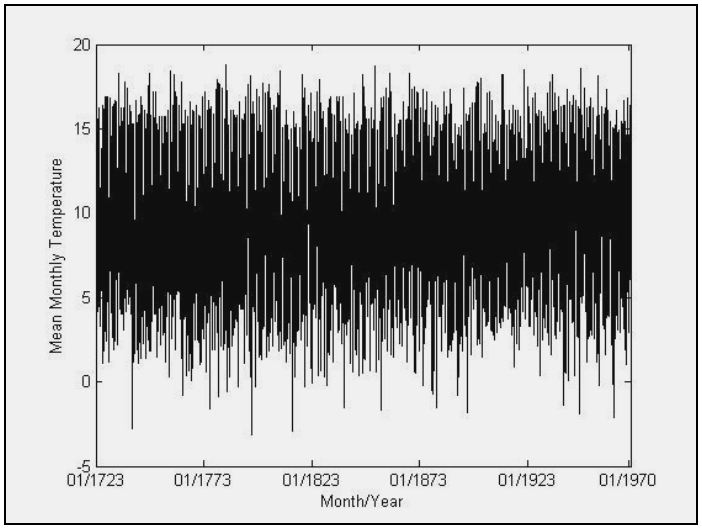

Figura A.2– A série temporal *England Monthly Temperature* (1723–1970)

Apêndice B

Cozimento Simulado para Diversidade

Simulated Annealing for Diversity (SAND; DE CASTRO, 2001), ou Cozimento Simulado para Diversidade, é uma implementação da meta-heurística conhecida por *Simulated Annealing* (Recozimento Simulado) com o objetivo de gerar um conjunto de vetores uniformemente distribuídos no espaço.

B.1 Cozimento simulado

Cozimento (do inglês *Annealing*) refere-se ao processo metalúrgico de síntese de ligas metálicas através do seu aquecimento e posterior resfriamento controlado: ao se aquecer o metal, a mobilidade dos átomos é aumentada, permitindo que estes se desloquem aleatoriamente para estados de energia superiores, e ao se resfriar o metal, os átomos vão perdendo a mobilidade. Como o resfriamento utilizado é lento e gradual, os átomos têm uma probabilidade maior de assumir configurações de baixa energia interna, resultando em cristais maiores e com propriedades desejadas, como maior resistência a forças de torção.

O *simulated annealing* (SA) funciona por meio de um processo análogo ao processo físico do recozimento, constituindo-se em uma meta-heurística probabilística de minimização global. A cada iteração, o SA busca um novo candidato a mínimo entre os vizinhos do estado atual, causando um pequeno deslocamento aleatório no sistema e calculando a *energia* (o valor da função a ser minimizada) do novo estado. Se esta energia for inferior à do estado atual, o deslocamento é automaticamente aceito e o estado do sistema é atualizado para o novo; mas se a energia for maior, a aceitação é tratada probabilisticamente.

A probabilidade de aceitação do novo estado depende de sua energia, da energia do estado atual, e de um parâmetro global do algoritmo chamado *temperatura*, que diminui com o tempo e é dada pela expressão:

$$P(\Delta E) = \exp\left(\frac{-\Delta E}{T}\right), \tag{B.1}$$

sendo E a energia do estado, *e* ΔE a diferença das energias entre o estado atual e o novo e T a "temperatura" do sistema.

Observa-se que a probabilidade diminui para valores maiores de ΔE e menores de T, fazendo com que deslocamentos que piorem muito a qualidade do sistema sejam menos prováveis, mas ainda permitindo que se escape de mínimos locais; e com que transições para estados de maior energia fiquem cada vez mais improváveis com o número de iterações (uma vez que a temperatura vai sendo gradualmente reduzida). Em particular, quando T chega a zero, a probabilidade de transição para estados de energia superior é nula e o algoritmo de *simulated annealing* se reduz a um método guloso de minimização (a tomada de decisão é sempre no sentido de melhorar localmente a função objetivo).

A forma como se dará a redução de T com o tempo é dada pelo plano de cozimento (do inglês *annealing schedule*) e pode seguir várias estratégias (incluindo às vezes etapas de elevação da temperatura) e influencia a qualidade da solução final encontrada. É importante que, a cada temperatura, o algoritmo seja executado por iterações suficientes para que o sistema possa explorar devidamente aquela condição de operação. Critérios de parada comuns incluem valores mínimos de energia ou de temperatura.

B.2 SAND

O SAND consiste de uma implementação específica do algoritmo de SA com o objetivo de induzir diversidade em uma dada "população". A energia E do sistema é dada pela similaridade entre os vetores que compõem a população, quanto maior a similaridade, maior é a energia do sistema. A temperatura T é reduzida cada vez que o sistema falha em ter sua energia alterada durante um número δ de iterações; a taxa α de mutação sofrida pela população também é reduzida a cada etapa de perturbação para uma dada temperatura. A redução de T e α é dada pela taxa β. O pseudo-código abaixo resume o algoritmo SAND:

1. *P = população inicial de N vetores,*
2. *E = energia(P),*
3. P_{Tmp} *= perturbação(P),* $\alpha = \beta.\alpha$*,*
4. E_{Tmp} *= energia(*P_{Tmp}*)*

se $\Delta E < 0$*, perturbação aceita:* $P = P_{Tmp}$*, retorna ao Passo* 3*,*

se $\Delta E > 0$*, cheque se perturbação é aceita (Equação* x.1*),*

se $\Delta E = 0$*, cont = cont* +1*,*

se cont > δ*, cont = 0,* $T = \beta.T$*,* α *= valor inicial, retorne ao Passo* 3

caso contrário, retorne ao Passo 3*,*

A energia do sistema é dada pelo comprimento do vetor direcional médio da população:

$$E = \left(\bar{I}^T \cdot \bar{I}\right)^{1/2}, \tag{B.2}$$

sendo:

$$\bar{I} = \frac{1}{N}\sum_{i=1}^{N} I_i \ , \tag{B.3}$$

e I_i os vetores direcionais unitários. Como $\bar{I}$ é a resultante dos N vetores, então, quando E se aproximar de zero, temos que os N vetores estão distribuídos de forma equilibrada.

O critério de parada é baseado no índice I_{SC}:

$$I_{SC}(\%) = 100 \cdot (1 - E) \tag{B.4}$$

Quando o I_{SC} atingir o valor pretendido (normalmente algo próximo de 100%), o algoritmo é interrompido.

B.2.1 Exemplos de geração de diversidade pelo SAND

As figuras abaixo mostram a atuação do SAND sobre um conjunto de 5 vetores bidimensionais (Figura B.1) e um conjunto de dez vetores tridimensionais (Figura B.2).

Nelas podemos observar como a distribuição vai se uniformizando com o tempo. Os gráficos B.1(f) e B.2(f) mostram a evolução do índice I_{SC}(%). Percebe-se como a energia do

sistema oscila com amplitudes cada vez menores (à medida que a temperatura é reduzida) até se estabilizar próxima de zero (I_{SC}(%) próximo de 100).

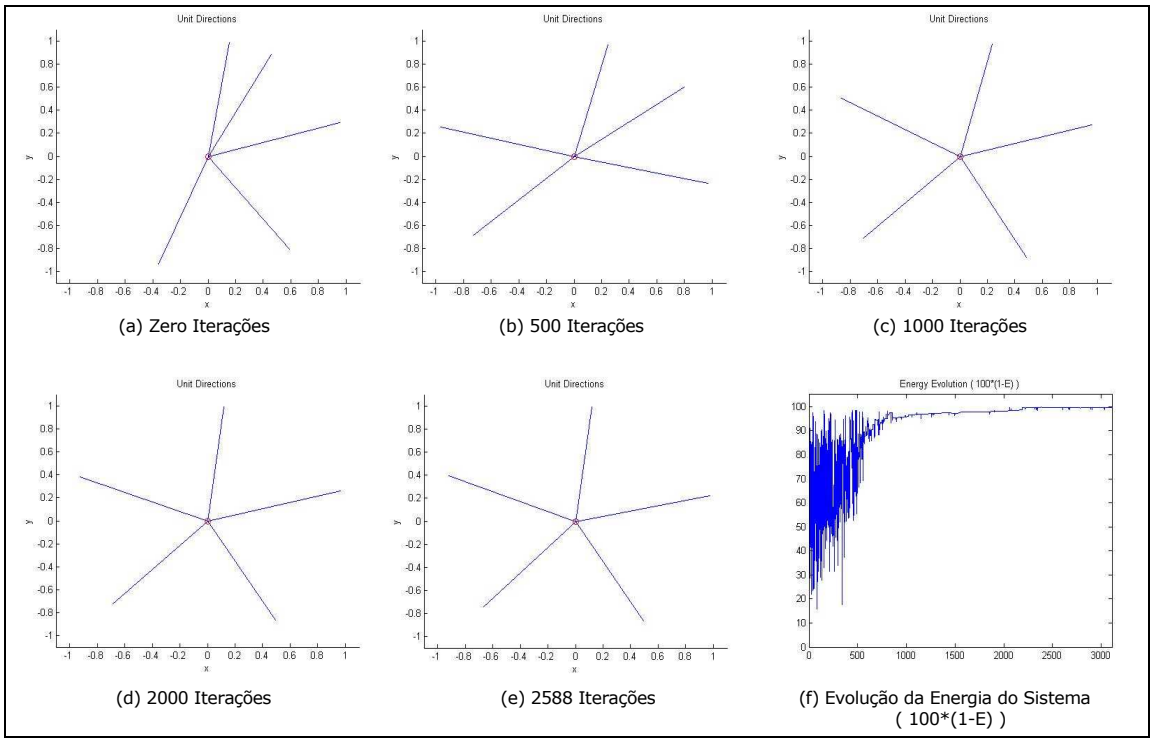

Figura B.1 – SAND atuando sobre um conjunto de cinco vetores bidimensionais

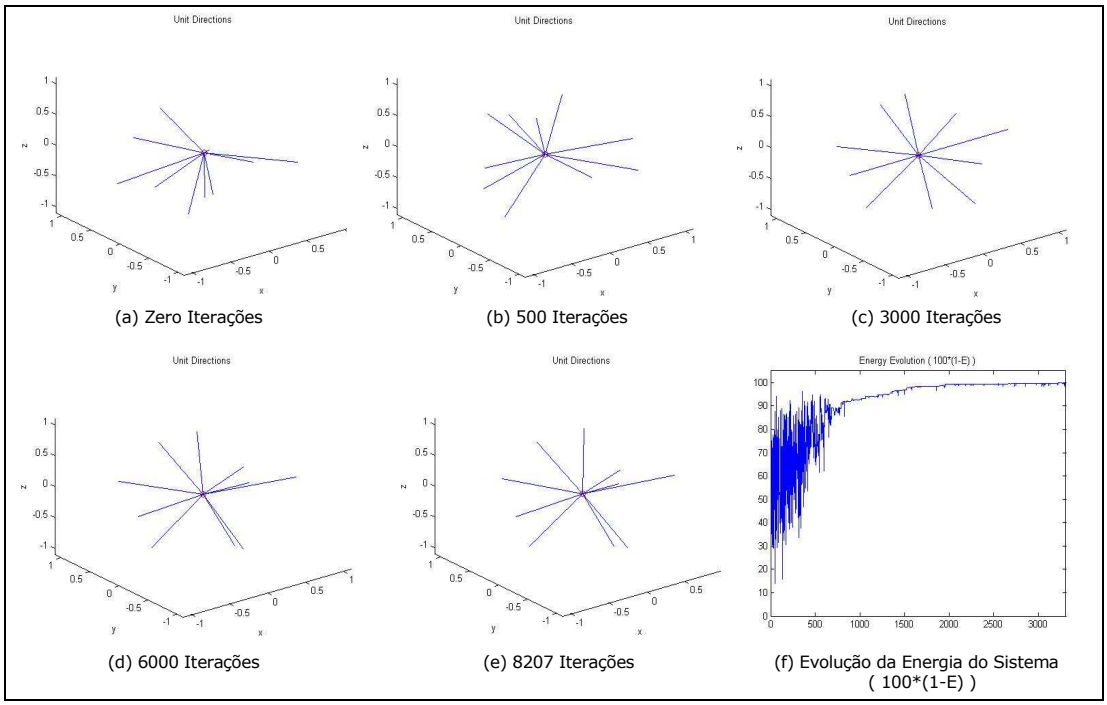

Figura B.2 – SAND atuando sobre um conjunto de dez vetores tridimensionais

Referências Bibliográficas

ARBIB, M.A. (1995). Handbook of Brain Theory and Neural Networks, The MIT Press.

BÄCK, T; FOGEL, D.B. & MICHALEWICZ, Z. (editors) (2000a). Evolutionary Computation 1: Basic Algorithms and Operators. Bristol, IOP Publishing.

BÄCK, T.; FOGEL, D.B. & MICHALEWICZ, Z. (editors) (2000b). Evolutionary Computation 2: Advanced Algorithms and Operators. Bristol, IOP Publishing.

CAPOBIANCO, E. (1996). Statistical Aspects of Semiparametric Neural Networks, Technical Report, Department of Statistical Sciences, University of Padua.

CHAUVIN, Y. & RUMELHART, D.E. (eds.) (1995). Backpropagation: Theory, Architectures, and Applications (Developments in Connectionist Theory), Lawrence Erlbaum.

CHERKASSKY, V. & MULIER, F.M. (2007). Learning from Data: Concepts, Theory, and Methods, 2nd. edition, Wiley-IEEE Press.

DE CASTRO, L.N. (2001). Engenharia Imunológica: Desenvolvimento e Aplicação de Ferramentas Computacionais Inspiradas em Sistemas Imunológicos Artificiais, Tese de Doutorado, Universidade Estadual de Campinas.

DE CASTRO, L.N.; IYODA, E.M.; PINHEIRO, E. & VON ZUBEN, F.J. (1999). Redes Neurais Construtivas: Uma Abordagem Comparativa, *Proceedings of the IV Brazilian Conference on Neural Networks*, pp.102-107.

FRIEDMAN, J. H. & STUETZLE, W. (1981). Projection Pursuit Regression, *Journal of the American Statistical Association*, vol. 76, pp. 817-823.

FRIEDMAN, J.H. & TUKEY, J.W. (1974). A Projection Algorithm for Exploratory Data Analysis, *IEEE Transactions on Computers*, vol. c-23, pp. 881-890.

GONÇALVES, R.; VON ZUBEN, F.J. & GOMIDE, F. (1998). Using Constructive Learning in Embedded Systems Engineering, *Proceedings of the IEEE International Joint Conference on Neural Networks*, vol. 1, pp. 251-255.

GUYON, I. & ELISSEEFF, A. (2003). An Introduction to Variable and Feature Selection, *Journal of Machine Learning Research*, vol. 3, pp. 1157-1182.

HIPEL K.W. & MCLEOD, A.I. (1994**).** *Hipel-McLeod Time Series Datasets Collection*: http://www.stats.uwo.ca/faculty/aim/epubs/datasets/default.htm.

HORNIK, K.; STINCHCOMBE, M. & WHITE, H (1989). Multilayer feedforward networks are universal approximators, *Neural Networks*, vol. 2 , no. 5, pp. 359-366.

HORNIK, K.; STINCHCOMBE, M.; WHITE, H. & AUER, P. (1994). Degree of Approximation Results for Feedforward Networks Approximating Unknown Mappings and Their Derivatives, *Neural Computation*, vol. 6, no. 6, pp. 1262-1275.

HUBER, P.J. (1985). Projection Pursuit (with Discussions), *The Annals of Statistics*, vol.3, no. 2, pp.435-525.

HWANG, J.N.; LAY, S.R.; MAECHLER, M.; MARTIN, R.D. & SCHIMERT, J. (1994). Regression Modeling in Back-Propagation and Projection Pursuit Learning, *IEEE Transactions on Neural Networks*, vol. 5, no. 3, 342-353.

IYODA, E.M. (2000). Inteligência Computacional no Projeto Automático de Redes Neurais Híbridas e Redes Neurofuzzy Heterogêneas, Dissertação de Mestrado, Universidade Estadual de Campinas.

JOHN, G.H.; KOHAVI, K. & PFLEGER, K. (1994). Irrelevant Features and the Subset Selection Problem, *Machine Learning: Proceedings of the Eleventh International Conference*, pp. 121-129, Morgan Kaufman Publishers.

JONES, L.K. (1987). On a Conjecture of Huber concerning the Convergence of Projection Pursuit Regression, *The Annals of Statistics*, vol. 15, no. 2, pp. 880-882.

KIRA, K. & RENDELL, L.A. (1992). A practical approach to feature selection, *International Conference on Machine Learning*, pp. 368-377.

KOHAVI, R. & JOHN, G.H. (1997). Wrappers for Feature Subset Selection, *Artificial Intelligence Journal special issue on relevance*, vol. 97, no. 1-2, pp.273-324.

KOHAVI, R. & JOHN, G.H. (1998). The Wrapper Approach, in *Feature Selection for Knowledge Discovery and Data Mining*, LIU, H. AND MOTODA, H. (eds.), Kluwer Academic Publishers.

KONENKO, I. (1994). Estimating Attributes: Analysis and Extensions of RELIEF, *European Conference on Machin e Learning*, pp. 171-182.

KWOK, T.-Y. & YEUNG, D.-Y. (1996). Use of Bias Term in Projection Pursuit Learning Improves Approximation and Convergence, *IEEE Trans. on Neural Networks*, vol. 7, no. 5, pp.1168-1183.

KWOK, T.-Y. & YEUNG, D.-Y. (1997) Constructive Algorithms for Structure Learning in Feedforward Neural Networks for Regression Problems, *IEEE Trans. on Neural Networks*, vol. 8, no. 3, pp.630-645.

LIMA, C.A.M. (2000). Emprego de Teoria de Agentes no Desenvolvimento de Dispositivos Neurocomputacionais Híbridos e Aplicação ao Controle e Identificação de Sistemas Dinâmicos, Dissertação de Mestrado, Universidade Estadual de Campinas.

LOUGHREY, J. & CUNNINGHAM, P. (2004). Overfitting in Wrapper-Based Feature Subset Selection: The harder you try the worse it gets, *Proceedings of International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pp. 33-43.

LOUGHREY, J. & CUNNINGHAM, P. (2005). Using Early-Stopping to Avoid Overfitting in Wrapper-Based Feature Subset Selection Employing Stochastic Search, *Department of Computer Science, Trinity College, Tech. Rep. TCD-CS-2005-37*.

LUENBERGER, D.G. (2003). Linear and Nonlinear Programming, 2nd Edition, Springer.

MAECHLER, M.; MARTIN, D.; SCHIMERT, J.; CSOPPENSZKY, M. & HWANG, J.N. (1990). Projection Pursuit Learning Networks for Regression, *Proceedings of the 2nd International IEEE Conference on Tools for Artificial Intelligence*, pp. 350-358.

MARRA, S. & MORABITO, F.C. (2006). A new technique for solar activity forecasting using recurrent Elman networks, *International Journal of Computational Intelligence*, vol. 3, no. 1, pp. 8-13.

MELEIRO, L.A.C.; VON ZUBEN, F.J. & MACIEL FILHO, R. (2008). Constructive Learning Neural Network Applied to Identification and Control of a Fuel-Ethanol Fermentation Process, *Engineering Applications of Artificial Intelligence*, to appear.

QUINLAN, J.R. (1993). C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers Inc.

RADTKE, P.V.W.; WON, T. & SABOURIN, R. (2006), An Evaluation of Over-Fit Control Strategies for Multi-Objective Evolutionary Optimization, *Proceedings of the IEEE International Joint Conference on Neural Networks*, pp. 3327-3334.

RAKOTOMAMONJY, A. (2003). Variable Selection Using SVM-based Criteria, *Journal of Machine Learning Research*, no. 3, pp.1357-1370.

REUNANEN, J. (2003). Overfitting in Making Comparisons Between Variable Selection Methods, *Journal of Machine Learning Research*, no. 3, pp. 1371-1382.

RUSSELL, R. (1993). Pruning Algorithm – A Survey, *IEEE Trans. on Neural Networks*, vol. 4, no. 5, pp. 740-747.

SIDC-TEAM (2008). World Data Center for the Sunspot Index, Royal Observatory of Belgium, *Yearly Report on the International Sunspot Number*, online catalogue of the sunspot index: http://www.sidc.be/sunspot-data/.

VON ZUBEN, F.J. (1996). Modelos Paramétricos e Não-Paramétricos de Redes Neurais Artificiais e Aplicações, Tese de Doutorado,Universidade Estadual de Campinas.

VON ZUBEN, F.J. & NETTO, M.L.A. (1995). Unit-Growing Learning Optimizing the Solvability Condition Applied to Constructive Learning, *Proceedings of the IEEE International Conference on Neural Networks*, vol. 2, pp. 795-800.

VON ZUBEN, F.J. & NETTO, M.L.A. (1997). Projection Pursuit and the Solvability Condition Applied to Constructive Learning, *Proceedings of the IEEE International Conference on Neural Networks*, vol. 2, pp. 1062-1067.

YU, L. & LIU, H. (2004). Efficient Feature Selection via Analysis of Relevance and Redundancy, *The Journal of Machine Learning Research*, vol. 5, pp. 1205-1224.

Índice Remissivo de Referências Bibliográficas