Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação

Previsão de carga de curto prazo usando ensembles de previsores selecionados e evoluídos por Algoritmos Genéticos.

Autor: Marcos de Almeida Leone Filho
Orientador: Prof. Dr. Takaaki Ohishi
Co-orientadora: Profa. Dra. Rosângela Ballini

Dissertação de Mestrado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica. Área de concentração: **Energia Elétrica**.

Banca Examinadora

Fernando José Von Zuben, Prof. Dr. ..... DCA/FEEC/UNICAMP
Paulo Sérgio Franco Barbosa, Prof. Dr. ..... DRH/FEC/UNICAMP
Secundino Soares Filho, Prof. Dr. ..... DENSIS/FEEC/UNICAMP
Takaaki Ohishi, Prof. Dr. ..... DENSIS/FEEC/UNICAMP

Campinas, SP
Janeiro/2006

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

Leone Filho, Marcos de Almeida
L553p Previsão de carga de curto prazo usando ensembles de previsores selecionados e evoluídos por algoritmos genéticos. / Marcos de Almeida Leone Filho. --Campinas, SP: [s.n.], 2006.

Orientadores: Takaaki Ohishi, Rosângela Ballini
Dissertação (Mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Sistemas de energia elétrica – Distribuidor de carga. 2. Energia elétrica - Consumo. 3. Energia elétrica - Produção. 4. Teoria dos conjuntos. 5. Redes neurais (Computação). 6. Sistemas difusos. 7. Algoritmos genéticos. I. Ohishi, Takaaki. II. Ballini, Rosângela. III. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. IV. Título.

Título em Inglês: Short-term load forecasting using ensembles of selected and evolved predictors by genetic algorithms
Palavras-chave em Inglês: Electric load forecasting, Ensembles, Artificial neural networks, Neuro-fuzzy networks, Genetic algorithms
Área de concentração: Energia Elétrica
Titulação: Mestre em Engenharia Elétrica
Banca examinadora: Fernando Jose Von Zuben, Paulo Sérgio Franco Barbosa, Secundino Soares Filho
Data da defesa: 31/01/2006

Resumo

Neste trabalho é proposta uma metodologia para previsão de séries temporais de carga de energia elétrica de curto prazo. Esta metodologia vem sendo muito utilizada no contexto da previsão de séries temporais e do reconhecimento de padrões. Os autores que propuseram esta metodologia a chamaram de “Ensembles”. Este nome tenta explicar o é este modelo: uma combinação de partes que juntas formam um só modelo.

Neste sentido, este nome expressa com relativa clareza qual é o principal aspecto desta metodologia, que no caso específico deste trabalho, é o de fazer várias previsões de uma mesma série temporal utilizando diferentes ferramentas que sozinhas são suficientemente competentes para prever a série temporal em questão, e em seguida combinar as soluções para, deste modo, tentar obter uma solução melhor do que quando é usada somente uma ferramenta.

As ferramentas usadas para compor a previsão dos “Ensembles” finais são Redes Neurais Artificiais (RNAs) e Redes Neurais Nebulosas. Atualmente, estas redes são largamente utilizadas em problemas de previsão de séries temporais, principalmente quando o fator gerador destas séries é um sistema não-linear. Desta forma, isto as tornou candidatas potenciais para prever valores de uma série de cargas de energia elétrica, pois este tipo de série tem características essencialmente não-lineares. Sendo assim, foram utilizados quatro tipos de redes: RNAs MLPs, RNAs Recorrentes, RNAs de Base Radial e Redes Neurais Nebulosas tipo ANFIS.

Com os modelos básicos de redes foram, utilizados Algoritmos Genéticos para evoluir os parâmetros destas redes e, assim, chegar a uma população de redes suficientemente competentes para fazer as previsões da série de cargas. Na próxima etapa, com os resultados das previsões da população de redes evoluídas foi feita a seleção dos melhores agrupamentos destas redes evoluídas e, como este processo requer a avaliação de diferentes configurações de modelos, esta seleção é baseada em Algoritmos Genéticos.

Os resultados obtidos ao se utilizar “ensembles” mostraram que este modelo foi capaz de alcançar uma grande robustez na previsão, reduzindo os erros de previsão, suavizando os resultados de previsão e deixando o modelo menos suscetível a grandes erros quando surgem “outliers” no conjunto de dados.

Palavras-chave: Previsão de Carga de Energia Elétrica, Ensembles, Redes Neurais Artificiais, Redes Neurais Nebulosas, Algoritmos Genéticos.

Abstract

This work proposes a methodology for short-term electric power load forecasting. This methodology is being widely used under the context of time series prediction and pattern recognition. It was named “ensembles” by the authors who developed it. This name carries the meaning of an assemblage of parts considered as forming a whole.

Therefore, this name expresses rather clearly the main characteristic of this methodology, which under the framework of this study is to make several predictions of the same time series using various different tools in which every single one alone is sufficiently competent to predict the above mentioned time series. After that, the predictions are combined in order to achieve a better prediction compared to the one that is obtained if a single predictor is used.

The tools implemented to form the final “ensembles” prediction are Artificial Neural Networks (ANNs) and Neuro-fuzzy Networks. Nowadays, these networks are being widely used in time series predictions problems, mainly when the factor that generates these series is a non-linear system. Hence, this fact has elected them as potential candidates to predict future values of an electric power load series because this series has essentially non-linear characteristics. As a result, four types of networks were utilized in this work: MLPs ANNs, Recurrent ANNs, Radial Basis ANNs and ANFIS type Neuro-fuzzy networks.

So, with the basic networks models, Genetic Algorithms were applied to evolve the parameters of these networks and, as a consequence, a population of networks sufficiently capable of predicting future values of the load time series was built. On the next step, with the results obtained from the evolved population of networks, a selection of the most suitable results of the individual networks were made and, as soon as this process implies the evaluation of multiple different combinations of models, this methodology was based on Genetic Algorithms. Then, this selected networks were combined.

The results when using “ensembles” revealed that this model was able to reach a great robustness in prediction tasks. In that sense, it was possible to reduce the level of prediction error, to smooth the resulting predictions and to make the model more stable reducing the possibilities of presenting high levels of errors when the used data set contains “outliers”.

Keywords: Electric Load Forecasting, Ensembles, Artificial Neural Networks, Neuro-fuzzy Networks, Genetic Algorithms.

Aos meus pais, à minha família, ao meu amigo Murilo e à minha namorada Cláudia que sempre me apoiaram durante todo meu curso de mestrado.

Meu especial agradecimento: Ao professor Secundino que me "adotou" como seu aluno na Faculdade de Engenharia Elétrica, confiando na minha capacidade e me proporcionando anos de intenso aprendizado; Ao professor Takaaki que sempre esteve junto acompanhando o meu trabalho e me apoiou em todos os instantes; À professora Rosângela que sempre contribuiu com excelentes idéias para o desenvolvimento do meu trabalho de mestrado; Aos meus colegas do COSE que sempre me ajudaram em todas as minhas necessidades acadêmicas; À Claudia, minha namorada, que me ajudou fazendo a correção ortográfica e gramatical desta dissertação além de ter sempre me apoiado durante todo o meu mestrado; À FAPESP e ao CNPq pelo apoio financeiro.

Sumário

Lista de Figuras

Lista de Tabelas

Glossário

AG – Algoritmos Genéticos

ANFIS – Adaptive Neuro-Fuzzy Inference System (Sistema de Inferência Neural Nebulosa Adaptativa)

ANNs – Artificial Neural Networks (Redes Neurais Artificiais)

AR – Autoregressive Model (Modelo auto-regressivo)

ARIMA – Autoregressive Integrated Moving Average Model (Modelo auto-regressivo, integrado e de médias móveis)

ARMA – Autoregressive Moving Average Model (Modelo auto-regressivos e de médias móveis)

Backward – No sentido regressivo

Ensembles – Agrupamento e combinação de ferramentas especialistas.

Fitness – Medida de adaptação dos indivíduos em seu ambiente.

Forward – No sentido progressivo

GA – Genetic Algorithms (Algoritmos Genéticos)

MA – Moving Average Model (Modelo médias móveis)

MLPs – Multi-Layer Perceptrons

Neuro-Fuzzy – Neural Nebuloso

Preditor – Do inglês “predictor”. O mesmo que previsor.

RBF – Radial Basis Function

RNAs – Redes Neurais Artificiais

RNR – Redes Neurais Recorrentes

T-S – Sistemas de Inferência Nebulosa tipo Takagi-Sugeno

Lista de Símbolos

$C_d(t)$ - Carga média diária na semana t do histórico, em uma determinado dia de semana d, onde $d = 1,2,...,7$ que corresponde a $d = seg, ter, ..., dom$ (Cap. 3, Sessão 3.1).

$C_{i,j}$ - Coeficiente de correlação entre o i-ésimo e o j-ésimo previsor do ensemble (Cap. 2, Sessão 2.5).

a_t - Componente aleatória (ruído branco) de uma série temporal.

T_t - Componente de tendência de uma série temporal.

S_t - Componente sazonal de uma série temporal.

σ - Desvio Padrão.

E_i - Erro de generalização do i-ésimo previsor do ensemble (Cap. 2, Sessão 2.5).

$f(\bullet)$ - Função de ativação de neurônios em RNAs (Cap. 2, Sessão 2.3).

$h(\bullet)$ - Função de ativação de neurônios em RNAs (Cap. 2, Sessão 2.3).

$\varphi(\bullet)$ - Função de fitness (Cap. 2, Sessão 3.1).

$\mu_A(x)$ - Grau de pertinência de x em A.

t - Instante no tempo (unidade de tempo).

$\mathbf{W}$ - Matriz de pesos de uma rede neural artificial.

$\bar{Z}$ - Média amostral da série temporal Z.

$\phi(\bullet)$ - Operador Auto-Regressivo.

$\theta(\bullet)$ - Operador Médias-Móveis.

ω_i - Peso associado ao i-ésimo previsor do ensemble (Cap. 2, Sessão 2.5).

$w_{i,j}$ - Peso da sinapse que conecta o neurônio i ao j (Cap. 2, Sessão 2.3).

Z_t - Valores de uma série temporal em um dado instante t.

$Var[\cdot]$ - Variância Amostral.

σ^2 - Variância Amostral.

$\mathbf{b}$ - Vetor de bias de uma rede neural artificial.

Capítulo 1: Considerações Iniciais

1 Introdução

1.1 Motivação e Importância

Atualmente, a alta competitividade imposta pelo mercado faz com que uma empresa que deseje estar bem posicionada tenha bons mecanismos de tomada de decisão como diferencial competitivo. Neste sentido, o desenvolvimento de ferramentas computacionais capazes de melhorar os processos de tomada de decisão é fundamental.

No atual contexto científico-tecnológico há, hoje, uma revolução no que diz respeito à quantidade e à sofisticação das ferramentas que auxiliam a tomada de decisão e controle. Esta revolução tem dois aspectos básicos. O primeiro se preocupa com a quantidade de variáveis e possíveis decisões nos complexos sistemas atuais, sendo que uma das áreas do conhecimento que trata estes problemas é conhecida como "*Pesquisa Operacional*" ou "*Management Science*". O segundo aspecto se preocupa com as decisões que são mais difíceis de serem modeladas, pois muitas vezes nem se sabe quais são as variáveis que implicam o acontecimento de um determinado fenômeno. Técnicas como "Programação Heurística" e "Inteligência Artificial" são utilizadas nestes casos [41].

No que diz respeito ao processo de tomada de decisão, vale salientar que, em operação e gerenciamento de sistemas, uma decisão tomada em um determinado instante visa melhorar a eficiência de um determinado sistema. Assim, ele será tão mais eficiente quanto melhor esta decisão fizer com que o seu comportamento no futuro seja melhorado de alguma forma. No

entanto, na maioria dos sistemas, não se sabe como será o comportamento deles no futuro, daí a necessidade de se dispor de ferramentas capazes de aprender com o passado para tentar fazer previsões para o futuro, pois estes dados não são conhecidos e o conhecimento deles, mesmo que aproximado, serviria como excelente parâmetro no processo de tomada de decisão.

Neste trabalho será tratada a classe de problemas onde não se conhece a relação entre o resultado do processo e o fator ou fatores que geraram este resultado. Sendo assim, é necessária a utilização de métodos de solução de problemas conhecidos como *heurísticos*. Eles são assim conhecidos pois permitem o tratamento de uma ampla classe de problemas reais, mas sem a garantia de achar a solução ótima para o problema.

O objetivo deste trabalho é tentar fazer previsões de valores futuros de uma série de dados de cargas registrados por uma companhia distribuidora de energia elétrica. Este tipo de problema é conhecido na bibliografia como “Predição de Séries Temporais” ou “Previsão de Séries Temporais”. Quando as séries temporais estudadas são resultado de um fenômeno natural ou estocástico, a previsão de valores futuros destas séries normalmente é muito complicada e *nunca* se espera uma solução ótima, ou seja, com erro nulo de previsão.

O problema mais geral quando se faz previsão de séries temporais é que não existe uma forma fechada ou um modelo que pode ser aplicado a todos os tipos de séries temporais, assim não existe um modelo “campeão” para fazer previsão de séries temporais. Assim sendo, normalmente é desenvolvido e parametrizado um modelo dedicado, também conhecido como especialista, que é capaz de predizer valores futuros de uma determinada série temporal de dados. Mesmo assim, eventualmente, esta série de dados pode mudar o padrão de comportamento e fazer com que este modelo ou os seus parâmetros não sejam mais tão adequados. Ou ainda, um

dado modelo pode apreender bem um comportamento característico de uma série temporal, enquanto um outro modelo pode apreender melhor um outro comportamento desta mesma série.

Desta forma, quando uma série temporal apresenta características mais complexas são mais adequados modelos híbridos ou modulares, onde não será empregada apenas uma ferramenta dedicada, mas um conjunto de ferramentas cujos resultados são combinados de alguma maneira. Assim, além de existirem diversos modelos para previsão de séries temporais, individualmente eles podem ser parametrizados de muitas maneiras diferentes, ficando muito difícil explorar todas as possíveis combinações de modelos e parâmetros diferentes para se chegar a uma previsão otimizada, o que sugere a utilização de heurísticas, como os Algoritmos Genéticos, para tentar chegar a uma boa configuração dos parâmetros e combinação de previsores.

Neste contexto, como a série de cargas estudada neste trabalho tem comportamento afetado por vários fatores, dentre eles, o mercado, o preço de energia elétrica, a temperatura e o clima, é sugerida a utilização de uma ferramenta que seja suficientemente robusta para garantir uma previsão com boa precisão. Neste sentido, é proposta neste trabalho uma técnica que resolve várias vezes o problema usando modelos de previsão diferentes e, em seguida, combina os resultados obtidos por eles. Esta técnica é conhecida como "*Ensemble*" e vem sendo largamente utilizada em várias aplicações como em [1], [21], [23], [25], [27], [39], [44] e [45].

1.2 O problema da previsão da carga

Em um sistema de potência, a energia gerada por ele deve ser exatamente aquela demandada pelos consumidores, ou seja, o sistema deve produzir exatamente a mesma energia requerida pelos consumidores que estão ligados a ele (soma-se a isto também todas as perdas em todo o sistema) a cada instante de tempo, continuamente, pois a energia excedente pode causar

sobre-tensão, uma vez que não é possível armazená-la, e energia em escassez pode causar sub-tensão no sistema, ou até mesmo falta de energia.

Assim, nos processos de operação e controle dos sistemas de potência, o acompanhamento da carga é fundamental, já que para fornecer energia elétrica de boa qualidade, segura e economicamente, uma empresa de energia elétrica precisa dispor de mecanismos que possibilitem a resolução de vários problemas técnicos e operacionais.

Com o propósito de obter melhorias no planejamento e operação do sistema elétrico, novas ferramentas de otimização e controle são desenvolvidas e aplicadas, tendo resultados relevantes na redução de custos. Uma das ferramentas mais utilizadas é a Análise e Previsão de Séries Temporais. Esta ferramenta visa obter valores futuros de séries de vazões, consumo, demanda, entre outros, dado um histórico prévio destas séries.

Desta forma, estas séries já registradas são estudadas, analisadas e, como resultado, é previsto um valor futuro para uma determinada série, como por exemplo, uma série de dados de carga.

Neste trabalho será abordado o problema de predizer valores futuros de uma série de cargas de uma empresa de energia elétrica no curto prazo. Para se entender melhor os tipos de previsão mais comumente abordados, estes serão explanados a seguir.

1.3 Tipos de previsão de carga em sistemas de potência

- ***Previsão de curto prazo***

Este tipo de previsão estima qual será a carga de energia elétrica desde um horizonte de algumas horas à frente até alguns dias à frente. Ela é a responsável por orientar o planejamento da operação, transferência de energia e gerenciamento da demanda [38], ou seja, ela é fundamental para se saber com antecedência qual será a operação do sistema. A previsão de curto prazo é

essencial para uma operação mais segura e de maior qualidade. Além disto, uma boa previsão de curto prazo pode otimizar os recursos de produção, diminuindo, desta forma, os custos de produção de energia elétrica.

- ***Previsão de médio prazo***

A previsão de energia elétrica de médio prazo pretende estimar a demanda de energia elétrica para os próximos meses, podendo chegar até um ano à frente.

A previsão de médio prazo é necessária para a programação do suprimento de combustível, operações de manutenção e de planejamento do intercâmbio. Ela é essencialmente orientada à otimização dos recursos disponíveis. Ou seja, como a operação das hidrelétricas depende dos estados atuais dos reservatórios e das afluências que serão recebidas durante um determinado período, deseja-se que as hidrelétricas produzam o máximo possível para evitar a utilização das termoelétricas (custos marginais). Neste caso, tanto o mercado quanto as afluências futuras são desconhecidas e são substituídas por valores estimados.

- ***Previsão de longo prazo***

Este tipo de previsão é usado para prever a demanda de energia elétrica nos próximos anos, podendo alcançar até um horizonte de dez anos. A previsão de longo prazo desempenha um papel completamente diferente dos dois outros tipos de previsão citados anteriormente. Ela tem papel fundamental na estratégia e planejamento do sistema de energia elétrica, ou seja, a previsão de longo prazo é fundamental para servir como ponto de referência para orientar os investimentos no setor energético e decisões comerciais.

Além disso, no contexto da distribuição de energia elétrica, a previsão de longo prazo é essencial, uma vez que as decisões de compra ou venda de energia elétrica são tomadas na maioria dos casos com cinco anos de antecedência.

Vale destacar que o maior problema de fazer previsões de longo prazo reside no fato de que ela é dependente de outros valores que também devem ser estimados para o mesmo período de previsão, como a disponibilidade de geração, preços do petróleo, câmbio e taxa de juros.

1.4 Previsão de carga de curto prazo

A previsão da carga de curto prazo é uma importante ferramenta na elaboração do programa de operação dos próximos dias. Erros na previsão da carga podem significar não atendimento da demanda, o que conseqüentemente gera perda de eficiência, diminuição da confiabilidade do sistema e aumento dos custos.

Por conseguinte, o conhecimento do comportamento da carga futura é o primeiro pré-requisito para um planejamento seguro e econômico do sistema de energia elétrica. Aumentar a segurança e a economia da operação de sistemas são as motivações para a realização de previsões de cargas mais confiáveis nos sistemas elétricos de potência.

O presente trabalho enfoca a previsão da demanda de carga de curto prazo, em que o grande desafio é manter o equilíbrio entre a energia gerada e a consumida. Um desequilíbrio, tanto de escassez como de excesso de geração de carga, pode causar problemas de sub-tensão ou de sobre-tensão no sistema. As soluções possíveis para equilibrar essa escassez ou excesso de energia são as reprogramações das geradoras ou as transferências de energias através dos intercâmbios dos sistemas. Entretanto, tanto as reprogramações quanto o intercâmbio demandam

certo intervalo de tempo, daí a importância de se realizar uma boa previsão de carga com antecedência suficiente.

Capítulo 2: Embasamento Teórico

Neste capítulo, será primeiramente abordada a análise de séries temporais, evidenciando desde os seus conceitos básicos até a modelagem para a sua previsão de valores futuros. Em seguida, serão expostas algumas dentre as principais ferramentas utilizadas atualmente na previsão de séries temporais.

1 Análise de Séries Temporais

Os objetivos básicos da análise de séries temporais são:

- Modelagem do fenômeno sob consideração a partir do estudo das características inerentes da série temporal em questão;
- Obtenção de conclusões estatísticas sobre o fenômeno;
- Avaliação da adequação do modelo em termos de previsão.

A seguir, serão abordados aspectos gerais de análise de séries temporais, iniciando com conceitos básicos de processos estocásticos, obtenção de funções de autocovariança e autocorrelação e, finalmente, será abordada a modelagem de séries temporais mostrando os principais modelos de regressão utilizados tradicionalmente, até os modelos utilizados mais recentemente, como as Redes Neurais Artificiais.

1.1 Introdução

Dada uma série temporal $Z(t_i), i = 1,2,...,n$, onde n denota o número de observações desta série, é de possível interesse:

- Investigar o mecanismo gerador da série temporal. Um exemplo de investigação é analisar as amplitudes e freqüências dos perfis de onda e tentar descobrir como estas foram geradas. Assim, é possível, por exemplo, procurar periodicidade nos dados. Neste caso poderá ser feita uma *análise espectral* nos dados.
- Fazer previsões de valores futuros da série;
- Descrever o comportamento da série. Neste caso é possível usar ferramentas como a construção de gráficos, histogramas e diagramas de dispersão, a verificação de existência de tendências, ciclos e variações sazonais.

Em muitas aplicações práticas, há problemas que podem ser caracterizados ou modelados como um sistema em que uma série de dados entra e outra sai deste sistema. No entanto, normalmente, o que transforma ou como são transformados os dados de entrada nos dados de saída é desconhecido, sendo que o operador que transforma os dados de entrada nos dados de saída é conhecido como *função de transferência*. Este sistema é chamado de Sistema Dinâmico. Abaixo, é apresentado um esquema deste sistema. $X(\cdot)$ representa a série de entrada, $Z(\cdot)$ a série de saída e $v(\cdot)$ a função de transferência.

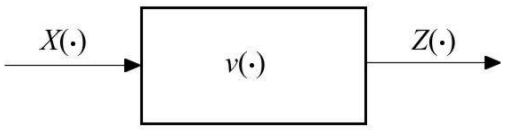


Figura 2.1 - Sistema Dinâmico

Caso $v(\cdot)$ seja um funcional linear, o sistema é conhecido como *Sistema Linear*. Caso contrário, o sistema será chamado de *Sistema Não Linear*. O sistema da Figura 2.1 poderia ser, por exemplo, modelado da seguinte maneira:

$$Z(t) = \sum_{\tau=0}^{\infty} v(\tau)\, X(t-\tau) \tag{2.1}$$

Conhecendo as séries de entrada e de saída, para resolver o problema de se obter $Z(\cdot)$ a partir de $X(\cdot)$ é preciso estimar ou escolher uma função $v(t)$ apropriada ao tipo de série.

Desta forma, a análise de uma série temporal deve ser feita de tal forma que o resultado seja um conjunto de propriedades estatísticas que caracterizem, conseqüentemente, o comportamento da série.

Quando se fala sobre análise de séries temporais, há, basicamente, dois tipos de análise que podem ser realizadas. Uma é a análise da série no domínio do tempo e outra no domínio da freqüência. Como, neste trabalho, serão estudadas séries resultantes de processos não determinísticos, será abordada, aqui, a análise de séries temporais no domínio do tempo, pois normalmente é esta a abordagem adotada pela maioria dos autores.

Como o próprio nome diz, uma série temporal do tipo $Z(t_i),\ i = 1,2,...,n$ se desenvolve ao longo do tempo e cada uma das i observações são coletadas em instantes diferentes, normalmente espaçados igualmente entre si. O mecanismo que gera estas observações é considerado um processo estocástico que pode ser descrito por leis probabilísticas, como será visto mais adiante.

A decomposição da série temporal em componentes de tendência, sazonalidade e aleatoriedade é a principal característica da análise clássica de séries temporais.

Considerando, então, as observações $Z(t_i),\ i = 1,2,...,n$ de uma série temporal, o modelo de decomposição consiste em escrever Z_t como uma soma de três componentes não-observáveis:

$$Z_t = T_t + S_t + a_t \tag{2.2}$$

Na equação (2.2) os termos T_t e S_t correspondem à tendência e sazonalidade, respectivamente, enquanto a_t é a componente aleatória de média zero e variância constante σ_a^2. Devido a estas características, usualmente a série $\{a_t\}$ é conhecida como um ruído branco.

Quando uma série temporal possui algum tipo de tendência, se diz que ela é não estacionária. O próximo tópico aborda o conceito de estacionaridade e mostra uma técnica largamente utilizada para remover as tendências de uma série temporal.

1.1.1 Estacionariedade

Normalmente, quando se utilizam modelos clássicos de previsão de séries temporais, é necessário que a série temporal seja estacionária. Assim sendo, como a maioria das séries temporais apresenta comportamento não estacionário, é necessário fazer uma transformação na série para que ela se torne estacionária.

Vale salientar que uma classe de modelos chamada ARIMA, que será abordada mais adiante, é capaz de descrever de maneira satisfatória séries estacionárias e não estacionárias desde que não tenham comportamento explosivo, e a classe de modelos baseados em Redes Neurais absolutamente não exigem que as séries temporais sejam estacionárias para poder descrevê-las com boa exatidão devido às suas características, que serão analisadas na seção 2.3.

Uma série temporal não estacionária apresenta, em geral, *tendências* que se alteram ao longo do tempo. Estas tendências podem ser lineares com inclinações positivas ou negativas, ou até mesmo não lineares. Nestes períodos onde as tendências permanecem as mesmas, é possível considerá-los como individualmente estacionários, apesar da série como um todo ser não estacionária.

O tipo mais comum de não estacionariedade é chamado de homogêneo. Neste caso, a série pode ser estacionária, flutuando ao redor de um nível, por certo tempo, depois mudar de nível e flutuar ao redor de outro nível, e assim por diante. A Figura 2.2 ilustra este tipo de não estacionariedade:

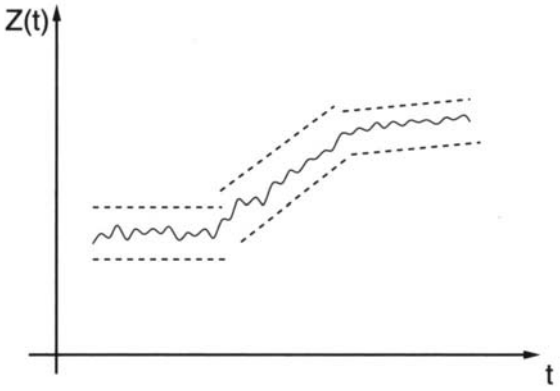


Figura 2.2 - Série não estacionária quanto às inclinações.

Tendo em vista o comportamento não estacionário da maioria das séries temporais e que a maioria dos procedimentos de análise estatística pressupõe que elas sejam estacionárias, será necessário transformar os dados originais das séries não estacionárias em séries estacionárias. A transformação mais comum consiste em tomar *diferenças sucessivas* [33] da série original, até se obter uma série estacionária. A primeira diferença de $Z(t)$ é definida como:

$$\Delta Z(t) = Z(t) - Z(t-1) \tag{2.3}$$

A segunda diferença de $Z(t)$ pode ser obtida da seguinte forma:

$$\Delta^2 Z(t) = \Delta\left(\Delta Z(t)\right) \tag{2.4}$$

$$\Delta^2 Z(t) = \Delta\left[Z(t) - Z(t-1)\right] \tag{2.5}$$

$$\Delta^2 Z(t) = \left(Z(t) - Z(t-1)\right) - \left(Z(t-1) - Z(t-2)\right) \tag{2.6}$$

$$\Delta^2 Z(t) = Z(t) - 2Z(t-1) + Z(t-2) \tag{2.7}$$

De uma forma geral, a n-ésima diferença de $Z(t)$ é:

$$\Delta^n Z(t) = \Delta\left[\Delta^{n-1} Z(t)\right] \tag{2.8}$$

Para exemplificar como funciona a transformação dos dados de uma série temporal para torná-la estacionária, foi utilizada a série de dados de carga registrados em uma companhia de energia elétrica do estado de São Paulo. Esta mesma série foi utilizada como estudo de caso neste trabalho. Os valores em cada instante *t* da série correspondem à carga horária média diária. A partir das equações (2.3) e (2.7) foram feitas as transformações de primeira e segunda ordem, como pode ser visto abaixo:

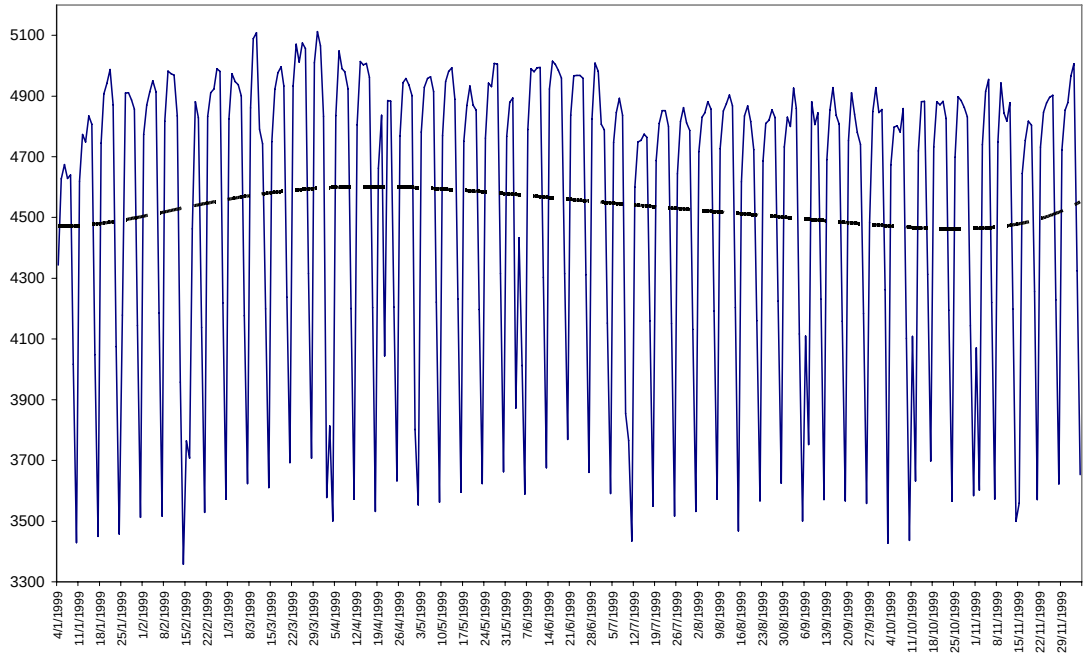


Figura 2.3 - Série Temporal de Carga Utilizada

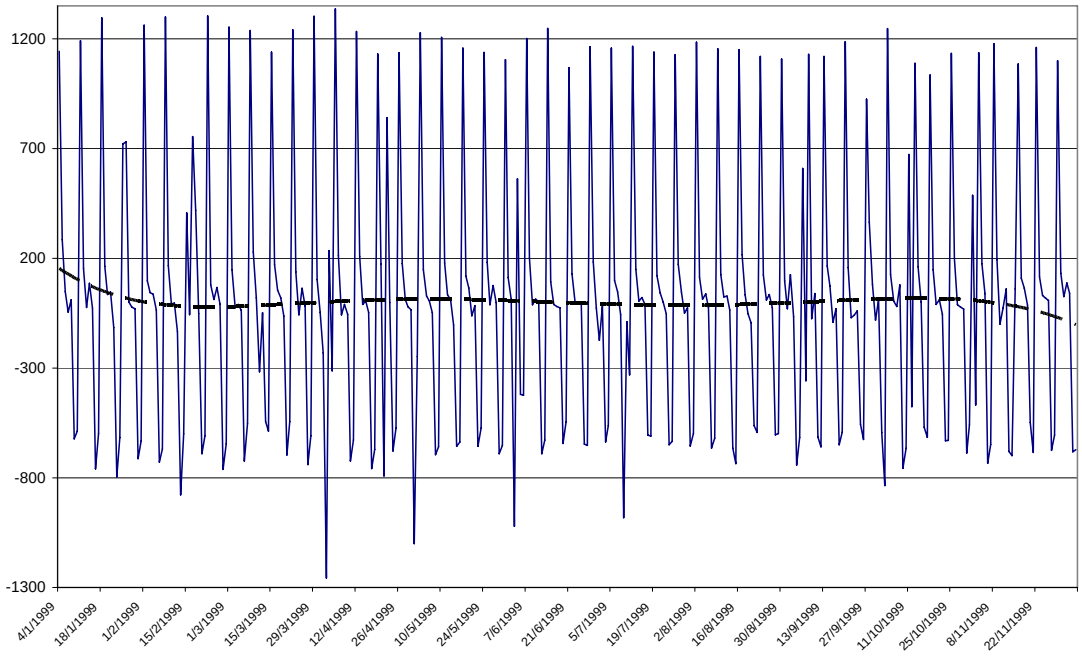


Figura 2.4 - Série de carga com transformação de primeira ordem

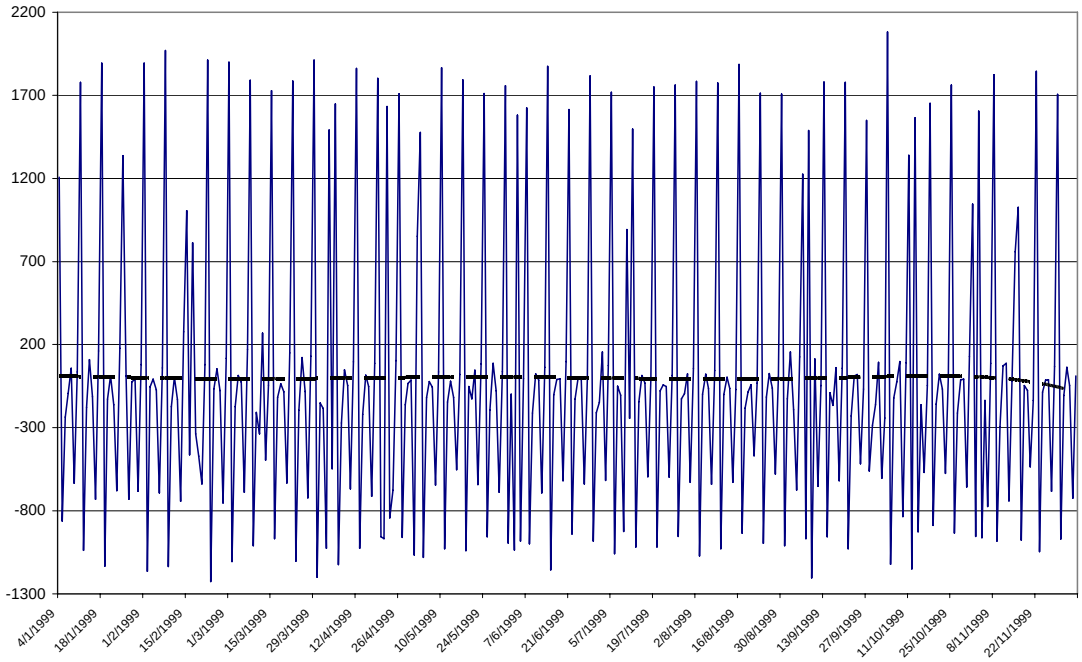


Figura 2.5 - Série de carga com transformação de segunda ordem.

Nos gráficos acima, as linhas tracejadas denotam as tendências de cada uma das séries temporais. Estas linhas foram obtidas através de um ajuste polinomial de ordem 6 (foi usada esta

ordem pois se desejava captar qualquer tendência bimestral dentro deste período de análise de 12 meses). Assim, é possível ver que a série da Figura 2.3 apresenta nitidamente um comportamento não estacionário. Com a transformação de primeira ordem, mostrada na Figura 2.4, percebe-se que a série fica predominantemente estacionária exceto nos trechos inicial e final. Finalmente, quando usada a transformação de segunda ordem, a série fica quase totalmente estacionária, como pode ser visto na Figura 2.5.

1.2 Sazonalidade

Como já foi visto, a sazonalidade de uma série temporal pode ser encarada como uma componente desta. No caso da série temporal estudada neste trabalho, é certo afirmar que ela foi gerada por um processo estocástico, assim, a sazonalidade também é dita como sendo estocástica.

Um dos métodos mais usados para calcular a componente sazonal de uma série temporal é o *método das médias móveis* [33]. Para explicar como é calculada a componente sazonal de uma série temporal, será tomada como exemplo a série estudada neste trabalho, usando o método das médias móveis.

Através da análise da Figura 2.3, fica evidente que a característica mais marcante nesta série é o perfil de "onda" que se repete de 7 em 7 dias, o que deixa claro que a sazonalidade desta série é semanal. Desta forma, para calcular a componente sazonal é necessário, inicialmente, calcular as médias das segundas ($d = 1$), terças ($d = 2$) e assim por diante para as n_d semanas:

$$\overline{Y_d} = \frac{1}{n_d}\sum_{i=1}^{n_d} Y_{id}\ ,\ d = 1, 2, ..., 7 \tag{2.9}$$

Em seguida, se calcula a média dos valores calculados acima:

$$\overline{Y} = \frac{1}{7}\sum_{d=1}^{7} \overline{Y_d} \tag{2.10}$$

Assim, é calculado um perfil constante sazonal:

$$\hat{S}_d = \overline{Y_d} - \overline{Y} \tag{2.11}$$

Finalmente, o modelo pode ser escrito como:

$$Z_t = T_t + \hat{S}_d + a_t \tag{2.12}$$

Para a série de cargas utilizada neste trabalho, foi calculada a componente sazonal. Quando ela é subtraída da série, restam ainda as componentes de tendência e o ruído branco, como pode ser visto na Figura 2.6.

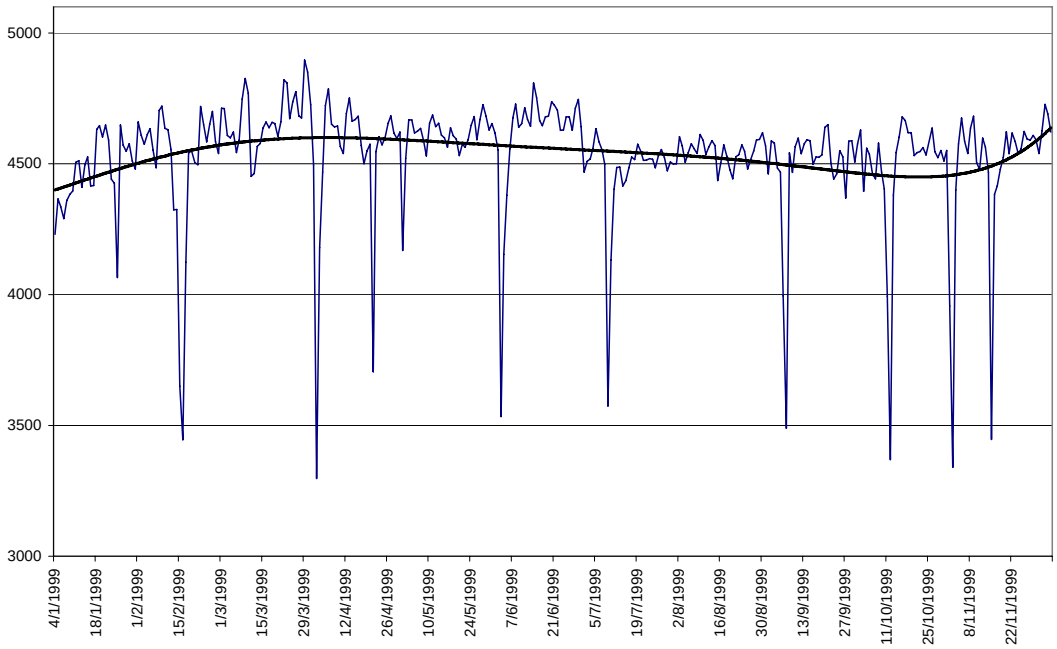


Figura 2.6 - Série de cargas sem a sazonalidade.

Em seguida, foi aplicada a transformação de segunda ordem na série da Figura 2.6 para torná-la estacionária. Com a remoção da componente de tendência, a série temporal resultante fica como ilustra a Figura 2.7.

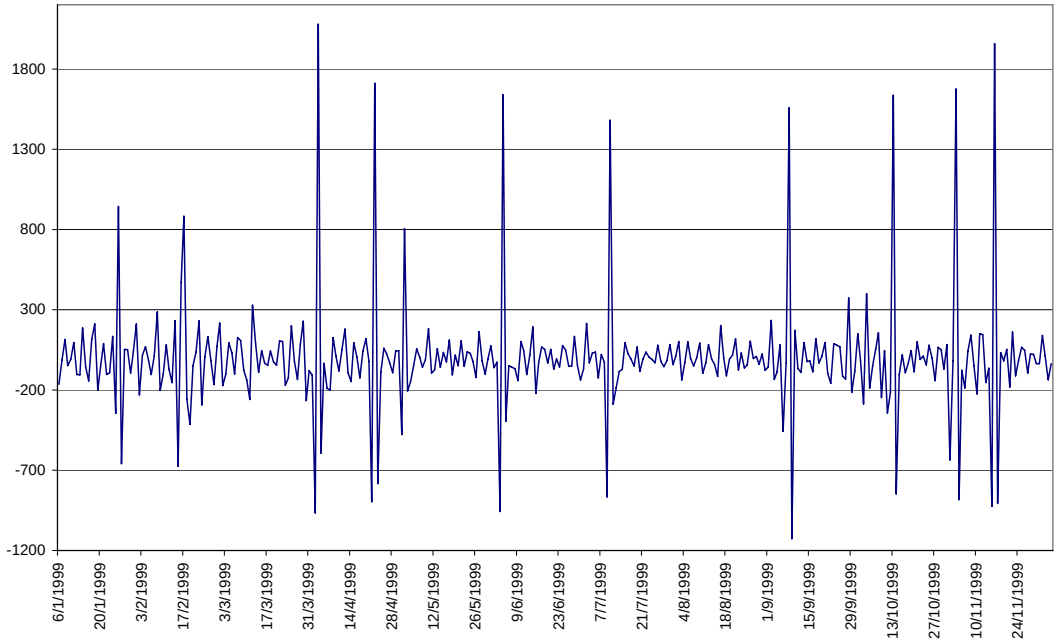


Figura 2.7 - Série de cargas estacionária e sem sazonalidade.

Não é necessário fazer nenhum tipo de cálculo para perceber que a série da Figura 2.7 não tem variância constante e, portanto, não se trata de um ruído branco, como era esperado. Este fato já sinaliza que a previsão de valores futuros para esta série temporal deverá apresentar erros elevados, principalmente devido aos picos (estes dados atípicos também são chamados de *outliers*) que aparecem em instantes aleatórios, desconfigurando o padrão de comportamento da série. O capítulo 2 irá abordar em detalhes técnicas de modelagem para previsão de valores futuros de séries sem tendência e sem sazonalidade.

1.3 Processos Estocásticos

Existem basicamente dois tipos de séries temporais. O primeiro deles é conseqüência de um fenômeno determinístico, ou seja, que pode ser modelado a partir da utilização de teoremas e axiomas que explicam o seu comportamento, ou seja, o modelo resultante conhece perfeitamente o futuro se é conhecido o estado do sistema em um dado instante de tempo. Assim, bastaria

calcular a fórmula que modela esta série para poder calcular valores futuros dela. O segundo é resultado de fenômenos regidos por leis probabilísticas, conhecidos como processos estocásticos. Este trabalho propõe um modelo para previsão de séries temporais resultante de processos estocásticos ou aleatórios. Assim, não serão abordadas aqui a análise ou previsão de séries temporais determinísticas.

Um processo estocástico ou função aleatória pode ser definido da seguinte forma:

Definição [1] *Seja um dado conjunto* T. *Um processo estocástico é uma família* $Z = \{Z(t), t \in T\}$, *tal que, para cada* $t \in T$, $Z(t)$ *é uma variável aleatória.*

Logo, uma série temporal a ser analisada é considerada como um resultado particular produzido por um mecanismo de probabilidade intrínseco de um processo estocástico. Em outras palavras, uma observação é uma realização de uma variável aleatória $Z(t)$ com função de densidade de probabilidade $p(Z(t))$. Da mesma forma, uma variável aleatória *N*-dimensional como função de densidade de probabilidade conjunta $p(Z(1), Z(2), ..., Z(N))$ pode descrever uma série temporal [3].

Como conceitos estatísticos importantes que qualificam uma série temporal têm-se a média (eq. (2.13)) e a variância (eq. (2.14)). Como será tratada aqui especificamente a análise de séries temporais discretas, estes conceitos são definidos da seguinte forma:

$$\overline{Z} = \frac{1}{N}\sum_{t=1}^{N} Z(t) \tag{2.13}$$

$$Var[Z] = \sigma_Z^2 = \frac{1}{N}\sum_{t=1}^{N}\left(Z(t) - \overline{Z}\right)^2 \tag{2.14}$$

As equações (2.13) e (2.14) também são conhecidas como média amostral e variância amostral, respectivamente, pois neste caso são usadas *N* amostras para calculá-las. A variância

amostral de Z também é usualmente denotada por σ_Z^2. A raiz quadrada da variância é conhecida como desvio padrão e denotada como σ_Z.

2 Modelagem para Séries Temporais

Além das análises que podem ser feitas com uma série temporal, conforme já foi descrito, também é essencial se analisar o processo que a gerou, tentando descobrir quais os fatores externos que podem influenciá-lo, o que ocorreu no processo que ocasionou o aparecimento dos dados detectados como *outliers* e, eventualmente, tentar descobrir alguma outra variável que possa explicar o comportamento da série.

Neste trabalho, como será tratado o problema de previsão de curto prazo, será previsto somente um ponto à frente na série com base no histórico existente, fazendo com que seja desnecessária a utilização de qualquer outro tipo de dado que não seja estritamente a série de cargas utilizada aqui.

2.1 Modelos Lineares Estacionários

2.1.1 Processo linear geral

Todos os modelos lineares estacionários que serão vistos aqui são casos particulares de um modelo de *filtro linear*. Os modelos lineares mostrados aqui, foram propostos por Box & Jenkins [3] e uma abordagem revisada pode ser encontrada em [33]. Este modelo supõe que a série temporal seja gerada através de um filtro linear (ou sistema linear), cuja entrada é o ruído branco. No caso específico do problema aqui tratado, seria a série temporal apresentada na Figura 2.7. O modelo de um filtro linear é ilustrado na Figura 2.8.

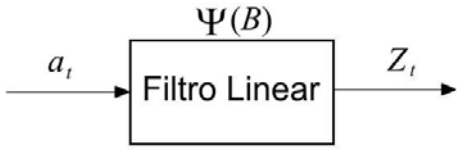


Figura 2.8 - Modelo de um Filtro Linear

Formalmente, as saídas são definidas em função das entradas da seguinte maneira:

$$Z_t = \mu + a_t + \psi_1 a_{t-1} + \psi_2 a_{t-2} + ... = \mu + \psi(B) a_t \tag{2.15}$$

Onde:

$$\psi(B) = 1 + \psi_1 B + \psi_2 B^2 + ... \tag{2.16}$$

A função $\psi(B)$ é denominada função de transferência do filtro e μ é um parâmetro determinando o nível da série. No caso específico de uma série estacionária, μ é a média amostral.

Chamando $\tilde{Z}_t = Z_t - \mu$, a partir da equação (2.15) se obtém:

$$\tilde{Z}_t = \psi(B) a_t \tag{2.17}$$

Ainda é possível escrever $\tilde{Z}_t$ em uma forma alternativa, como uma soma ponderada de valores passados $\tilde{Z}_{t-1}, \tilde{Z}_{t-2}, ...$ mais um ruído a_t:

$$\tilde{Z}_t = \pi_1 \tilde{Z}_{t-1} + \pi_2 \tilde{Z}_{t-2} + ... + a_t = \sum_{j=1}^{\infty} \pi_j \tilde{Z}_{t-j} + a_t \tag{2.18}$$

2.1.2 Modelos Auto-regressivos

Se na equação (2.18), $\pi_j = 0$ para $j > p$, se obtém um modelo auto-regressivo de ordem *p* (denominado também pela sigla AR(*p*)), que é modelado da seguinte forma:

$$\tilde{Z}_t = \phi_1 \tilde{Z}_{t-1} + \phi_2 \tilde{Z}_{t-2} + ... + \phi_p \tilde{Z}_{t-p} + a_t \tag{2.19}$$

Este modelo também é conhecido como *filtro de resposta ao impulso infinito* (IIR), uma vez que as saídas não são apenas uma transformação das entradas externas a_t, mas também uma dinâmica interna que envolve os valores $\tilde{Z}_{t-k}$, $k = 1,2,..., p$, significando que, na prática, este modelo é realimentado pelos próprios dados da série $\tilde{Z}_{t-k}$, $k = 1,2,..., p$.

2.1.3 Modelos de médias móveis

De maneira análoga à modelagem do filtro linear descrito pela equação (2.15), o modelo de médias móveis de ordem *q* é descrito da seguinte forma:

$$Z_t = \mu + a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2} - ... - \theta_q a_{t-q} \tag{2.20}$$

Essa equação descreve um filtro linear, também conhecido como filtro de convolução, uma vez que a nova série Z_t é descrita a partir da série a_t por um filtro de ordem *q* com coeficientes θ_i, $i = 1,2,..., q$. Essa relação é denominada *médias móveis* ou MA(*q*). Além deste nome, este filtro é também conhecido como *filtro de resposta ao impulso finita*, ou FIR, uma vez que este filtro opera em um laço aberto sem realimentação. A transformação realizada por ele é feita apenas com as entradas externas a_{t-i}, $i = 1,2,..., q$ que lhe são fornecidas. Isso garante com que a saída se anule após *q* passos, assim que o sinal de entrada se anule.

2.1.4 Modelos auto-regressivos e de médias móveis

Os modelos auto-regressivos são bastante populares em algumas áreas, como na descrição de séries econômicas, em que se pensa no valor de alguma variável em um determinado instante *t* como função dos valores defasados da mesma variável. No entanto, representar um processo, por mais simples que ele seja, utilizando um modelo de médias móveis não parece ser muito apropriado, pois ele não consegue representar de forma satisfatória a maioria dos processos reais

que, em geral, são bem complexos. Assim, para aumentar o número de série temporais que poderiam ser tratadas por um mesmo modelo sem aumentar muito o número de parâmetros, a inclusão de termos auto-regressivos e médias móveis seria, alternativamente, uma solução mais adequada.

Assim, são definidos os modelos auto-regressivos e de médias móveis ou ARMA(*p*,*q*), onde *p* é a ordem da porção auto-regressiva e *q* é a ordem da porção médias móveis, como descrito na equação (2.21).

$$\tilde{Z}_t = \phi_1 \tilde{Z}_{t-1} + \phi_2 \tilde{Z}_{t-2} + ... + \phi_p \tilde{Z}_{t-p} + a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2} - ... - \theta_q a_{t-q} \quad (2.21)$$

Denotando $\phi(B)$ e $\theta(B)$ como os operadores auto-regressivos e de médias móveis, respectivamente, a equação (2.21) admite a seguinte forma compacta:

$$\phi(B)\tilde{Z}_t = \theta(B)a_t \quad (2.22)$$

2.2 Modelos Lineares Não-Estacionários

Como foi visto, os métodos lineares estacionários requerem que a série temporal seja estacionária. Surge, portanto, a necessidade de se tratar as séries temporais considerando também a tendência nela contida. Neste sentido, são propostos os modelos ARIMA.

2.2.1 Modelos ARIMA

No caso da equação (2.22), se o termo $\tilde{Z}_t$ for substituído por um termo W_t, onde $W_t = \Delta^d \tilde{Z}_t$, é possível representar W_t por um modelo ARMA(*p*,*q*):

$$\phi(B)W_t = \theta(B)a_t \quad (2.23)$$

Como W_t é uma diferença de $\tilde{Z}_t$, então $\tilde{Z}_t$ é uma integral de W_t. Assim se diz que $\tilde{Z}_t$ segue um modelo auto-regressivo, *integrado*, de médias móveis, ou ARIMA:

$$\phi(B)\Delta^d \tilde{Z}_t = \theta(B)a_t \tag{2.24}$$

Nota-se que o modelo da equação (2.24) é de ordem (p,d,q) e denotado como ARIMA(*p*,*d*,*q*), uma vez que *p* e *q* são ordens de $\phi(B)$ e $\theta(B)$, como já foi explicado anteriormente.

É interessante perceber que os modelos ARIMA são modelos mais genéricos, de tal forma que eles englobam os outros modelos já descritos anteriormente, simplesmente zerando uma ou mais ordens é possível obter os outros modelos:

$$\text{ARIMA(p,0,0) = AR(p);} \tag{2.25}$$

$$\text{ARIMA(0,0,q) = MA(q);} \tag{2.26}$$

$$\text{ARIMA(p,0,q) = ARMA(p,q).} \tag{2.27}$$

2.3 Redes Neurais Artificiais (RNAs)

Uma RNA é uma ferramenta computacional resultante de pesquisas no campo da inteligência artificial que procuravam entender e modelar o comportamento do cérebro humano. A interação entre neurônios no cérebro humano, na realidade, ainda não é compreendida pela ciência, mas é identificada como comportamento inteligente. Com o propósito de desenvolver o mesmo tipo de estrutura para um modelo computacional de comportamento inteligente, neurologistas e pesquisadores da inteligência artificial propuseram uma rede altamente interconectada de nódulos (ou neurônios).

Em síntese, as RNAs consistem em elementos de processamento altamente interconectados denominados neurônios. O modelo de um neurônio é ilustrado na Figura 2.9.

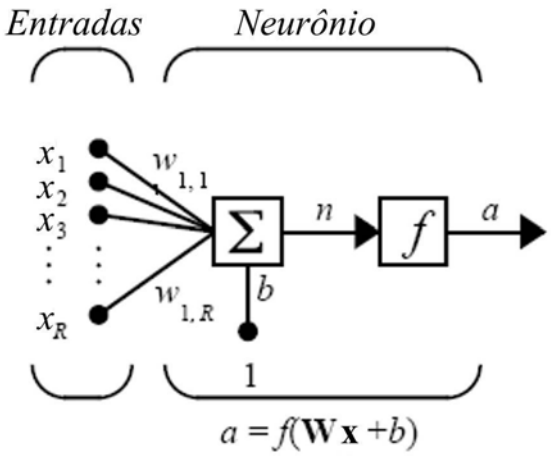


Figura 2.9 - Modelo de um neurônio.

Um neurônio pode ter um número R de entradas e uma saída a. A saída de um neurônio é determinada por uma função não-linear $f(\cdot)$, conhecida como *função de ativação*, e uma soma ponderada por pesos $w_{1,i}(i = 1,2,...,R)$ das entradas somada ao valor de um bias b. Assim, a saída a é facilmente calculada em função das entradas, conforme a equação (2.28):

$$a = f\left(\mathbf{W}\mathbf{x} + b\right) \tag{2.28}$$

Onde $\mathbf{x} \in \Re^{R\times 1}$ é o vetor dos dados de entrada e $\mathbf{W} \in \Re^{1\times R}$ é o vetor dos pesos que ponderam as entradas.

O modelo mais clássico de neurônio foi proposto por McCulloch e Pitts em 1943. O modelo é parecido com o da Figura 2.9. No entanto, a função de ativação por eles usada foi a tipo degrau em que não existiam pesos nas conexões ou bias. Contudo, este modelo foi esquecido por muito tempo, pois não encontrou grandes aplicações práticas.

Em 1957, Rosenblatt criou um tipo de neurônio denominado por ele como *Perceptron*. A função de ativação do Perceptron também era do tipo degrau, entretanto, possuindo pesos e bias. Rosenblatt também desenvolveu uma técnica para ajustar os pesos. A principal característica do

Perceptron é que ele tem convergência garantida quando os padrões de entrada forem linearmente separáveis, ou seja, ele pode ser usado como um classificador de padrões.

Com a evolução do projeto de redes neurais artificiais, foram acrescentados mais neurônios que se interconectam através sinapses ou conexões que aplicam pesos as sinais que por elas passam. Os pesos de uma rede neural são normalmente inicializados com zeros ou com valores randômicos próximos de zero. Para que a rede consiga reproduzir a saída desejada a partir dos dados de entrada, estes pesos são ajustados durante o período de treinamento da rede neural.

Durante o treinamento de uma RNA, o conjunto de pesos iniciais deve ser adaptado para a rede aprender o comportamento de um sistema. Os algoritmos particulares de treinamento (ou aprendizado) de uma RNA ajustam iterativamente os pesos das conexões entre os neurônios até que os pares desejados de informações de entrada(s) e saída(s) sejam obtidos e as relações de causa e efeito possam ser estabelecidas. Se as condições mudarem de tal modo que o desempenho do modelo não seja mais adequado, pode-se submeter a RNA a mais um treinamento sob estas novas condições de entrada(s) e saída(s) para corrigir seu desempenho. Assim sendo, pode-se submeter a RNA a uma atualização periódica, resultando num modelo auto-ajustável em linha.

As RNAs são especificadas pela topologia ou arquitetura, características dos neurônios, e regras de treinamento.

Quanto à arquitetura da rede, ela pode ser especificada em termos de números de entradas, números de camadas intermediárias, números de neurônios em cada camada intermediária, número de saídas e nas maneiras como são feitas as conexões entre os neurônios.

Quanto às características dos neurônios, a principal é a função de ativação que neles é utilizada. Existem vários tipos de funções de ativação diferentes, entre elas estão as funções sigmoidais, a função degrau e as funções de base radial.

Finalmente, quanto às regras de treinamento, vale dizer que existem três paradigmas básicos de treinamento: o supervisionado, o não supervisionado e o por reforço. Neste trabalho será abordado somente o aprendizado supervisionado. Assim sendo, dentro deste paradigma existem várias maneiras de ajustar os pesos de uma rede neural, sendo que uma das mais conhecidas é chamada de retro propagação do erro, expressão derivada do inglês "*backpropagation*". Na Figura 2.10, é mostrado um esquema de uma rede neural com duas camadas intermediárias, uma camada de saída com S saídas e neurônios com funções de ativação f^n, onde n é o índice que representa a camada.

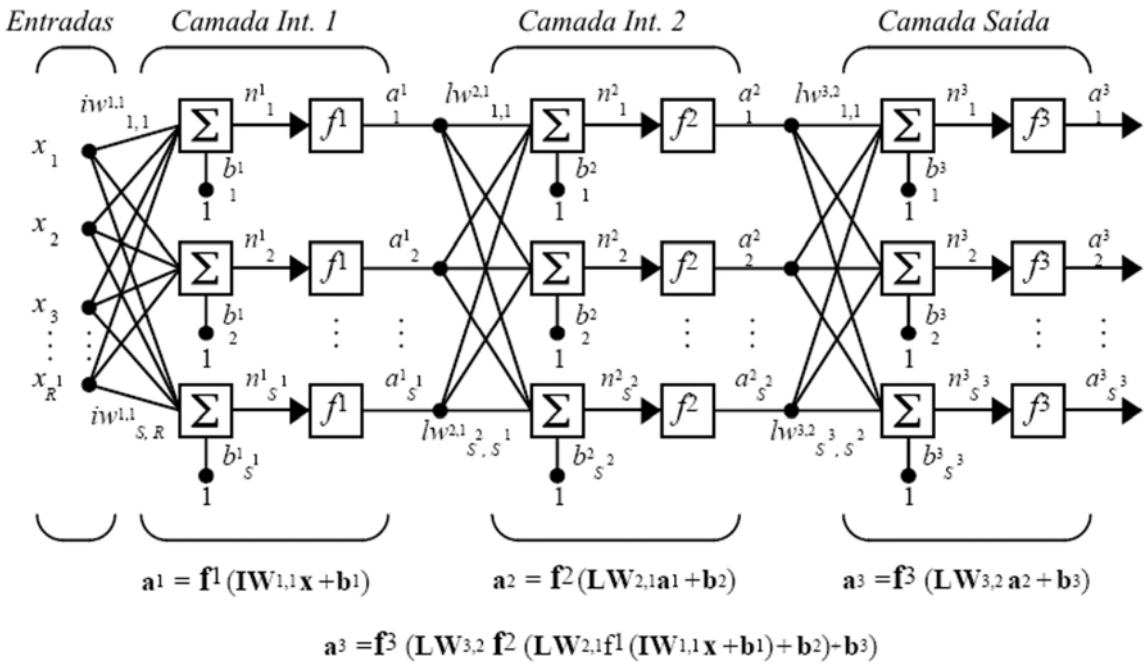


Figura 2.10 - Arquitetura de uma Rede Neural

As redes neurais podem ser utilizadas em vários campos de pesquisa, podendo ser aplicadas para resolver vários tipos de problemas diferentes. Entre as aplicações usuais das RNAs têm-se: reconhecimento e classificação de padrões, clustering ou agrupamento, previsão de séries temporais, aproximação de funções, otimização, processamento de sinais, análise de imagens e

controle de processos [50], requerendo para tanto outras arquiteturas além daquela apresentada na Figura 2.10.

2.3.1 Redes Neurais para Previsão de Séries Temporais

Como será visto, as Rede Neurais Artificiais utilizadas nas aplicações deste trabalho, com os seus pesos ajustados após uma fase adequada de treinamento deve se comportar como um *aproximador universal* mapeando os dados de entrada nos dados de saída desejados.

Durante a fase de treinamento, o objetivo é justamente fazer com que a rede seja capaz de fazer um mapeamento de tal forma que o erro no acerto das saídas quando alimentada por determinadas entradas seja mínimo. Assim, durante o treinamento são apresentadas entradas e saídas desejadas para que os pesos sejam ajustados. No entanto, após o treinamento a rede se comportará como uma função e precisará somente de dados de entrada para calcular as saídas. Quando se utiliza redes neurais para previsão de séries temporais, espera-se que as saídas calculadas pela rede estimem valores futuros de uma série. Mas para isto é preciso que a rede tenha conseguido abstrair características da série temporal, como sazonalidade e tendência.

Para uma RNA treinada, generalização significa estimar a saída desejada para valores de entrada não utilizados na determinação do modelo. Este processo requer aproximar valores de saída para pontos nos quais não existem exemplos.

Sob essa perspectiva de aproximação, o processo de aprendizagem é mal condicionado no sentido de que a informação contida nos dados não é suficiente para reconstruir unicamente o mapeamento em regiões onde dados não estão disponíveis. Neste caso, normalmente o *erro de generalização* da rede é alto.

Sendo assim, é preciso assumir a priori algumas características do mapeamento, como por exemplo, que ele é suave: pequenas variações em alguns parâmetros de entrada causarão

pequenas variações na saída. Técnicas que exploram as restrições de suavidade em problema de aproximação são conhecidas como técnicas de regularização [59]. O ensemble desenvolvido neste trabalho realiza implicitamente a regularização em seu mapeamento e esta característica ainda será melhor discutida mais adiante.

2.3.2 Redes Neurais Multicamadas

Com a publicação do livro "*Perceptrons*" [32], em sua primeira edição de 1969, foi proposta a rede perceptron com múltiplas camadas que se mostrou capaz de superar as limitações encontradas no perceptron simples. Esta rede foi chamada de "*Multi-Layer Percetron*", ou "*Perceptron multicamadas*" ou simplesmente MLP. Uma rede neural é considerada como sendo uma MLP se ela tem pelo menos uma camada intermediária.

A estrutura de uma rede MLP é similar à rede da Figura 2.10. No entanto, os neurônios têm função do tipo sigmoidal, como, por exemplo, a função logística e a função tangente hiperbólica. A diferença principal em relação ao modelo tradicional dos Perceptrons é que passaram a existir muitos neurônios e, conseqüentemente, mais possíveis conexões entre eles com pesos e bias a serem ajustados. Desta forma, foi necessário, também, o desenvolvimento de um algoritmo suficientemente eficiente para ajustar estes pesos.

Um algoritmo de *retropropagação do erro*, conhecido como *backpropagation*, se tornou muito popular após a publicação do livro "*Parallel Distributed Processing*" [60] de Rumelhart e colaboradores. Este algoritmo consiste basicamente de dois passos:

- *Propagação progressiva do sinal funcional*: durante este processo, todos os pesos da rede são mantidos fixos;

- *Retropropagação do erro*: durante este processo os pesos da rede são ajustados baseados no erro. O sinal de erro é propagado em sentido oposto ao de propagação do sinal funcional, por isso o nome de *retropropagação do erro*.

Como o objetivo da rede é conseguir mapear corretamente um conjunto de saídas desejadas, o melhor conjunto de pesos será aquele que minimizar o erro da saída obtida em relação à desejada. Assim, serão utilizadas informações sobre o gradiente do erro em relação aos pesos e limiares da rede, pois atualizando os pesos da rede na direção oposta ao vetor gradiente em relação aos pesos se minimiza o erro entre a saída da rede e a saída desejada.

Zekic [55] utiliza redes neurais MLP para fazer previsão de séries temporais de índices de estoques de mercado. Estes tipos de séries temporais são também conhecidas como Séries Temporais Financeiras e, normalmente, são de difícil modelagem e predição.

2.3.3 Redes Neurais Recorrentes

Além das redes neurais multicamadas, as redes neurais recorrentes representam uma das arquiteturas de RNA mais utilizadas em diversos tipos de aplicações. As RNAs recorrentes recebem este nome pois possuem pelo menos um laço realimentando a saída de neurônios para outros neurônios da rede.

As Redes Neurais Recorrentes (RNR) são capazes de representar uma grande variedade de processos dinâmicos, pois a presença de realimentação de informação permite a criação de representações internas e de dispositivos de memória capazes de processar e armazenar informações temporais e sinais seqüenciais. A presença de conexões recorrentes ou realimentação de informação pode conduzir a comportamentos complexos, mesmo com um número reduzido de parâmetros.

As redes de Elman [12] são possivelmente as redes neurais recorrentes mais conhecidas. A arquitetura básica de uma rede de Elman é apresentada na Figura 2.11.

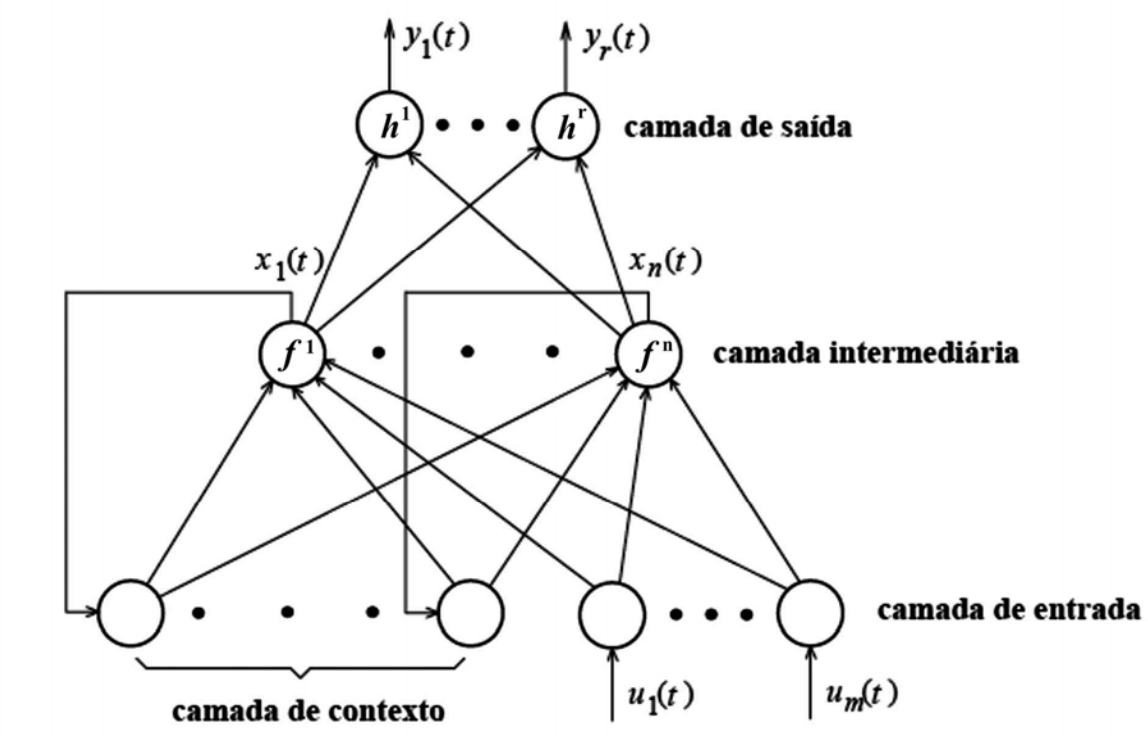


Figura 2.11 - Arquitetura básica de uma Rede de Elman

De acordo com a Figura 2.11, é possível deduzir os sinais de saída da camada intermediária $x_j(t),\ j=1,2,...,n$, em função dos sinais da camada de entrada e da camada de contexto e também deduzir os sinais de saída da camada de saída $y_k(t),\ k=1,2,...,r$ em função da camada intermediária conforme as equações (2.29) e (2.30), respectivamente:

$$x_j(t) = f^j\left(\sum_{i=1}^{m} w_{j,i}\, u_i(t)\right) + \sum_{l=1}^{n} w_{j,l}\, x_j(t-1),\ j=1,2,...,n \tag{2.29}$$

$$y_k(t) = h^k\left(\sum_{j=1}^{n} w_{k,j}\, x_j(t)\right),\ k=1,2,...,r \tag{2.30}$$

Usando a notação matricial e assumindo que $\mathbf{W}_{entrada} \in \Re^{n \times m}$ e $\mathbf{u}(t) \in \Re^{m \times 1}$ são a matriz de pesos e vetor de sinais da camada de entrada, $\mathbf{W}_{contexto} \in \Re^{n \times n}$ é a matriz de pesos da camada de contexto, $\mathbf{W}_{in} \in \Re^{r \times n}$ e $\mathbf{x}(t) \in \Re^{n \times 1}$ são a matriz de pesos e o vetor de sinais da camada intermediária e $\mathbf{y}(t) \in \Re^{r}$ é o vetor de sinais da camada de saída, as equações (2.29) e (2.30) ficam da seguinte forma:

$$\mathbf{x}(t) = \mathbf{f}\left(\mathbf{W}_{entrada}\,\mathbf{u}(t)\right) + \mathbf{W}_{contexto}\,\mathbf{x}(t-1) \tag{2.31}$$

$$\mathbf{y}(t) = \mathbf{h}\left(\mathbf{W}_{saida}\,\mathbf{x}(t)\right) \tag{2.32}$$

Das equações (2.31) e (2.32) se conclui que:

$$\mathbf{y}(t) = \mathbf{h}\left(\mathbf{W}_{saida}\left[\mathbf{f}\left(\mathbf{W}_{entrada}\,\mathbf{u}(t)\right) + \mathbf{W}_{contexto}\,\mathbf{x}(t-1)\right]\right) \tag{2.33}$$

Como se vê pela equação (2.33), as redes neurais recorrentes têm a capacidade de reutilizar os dados já processados, que são os pesos das conexões por onde passam estes sinais $\mathbf{x}(t-1)$ já processados, o que as tornam capazes de modelar processos dinâmicos. Por outro lado, redes neurais não-recorrentes podem ser interpretadas como poderosos operadores de transformação de representação, mas não são capazes de reutilizar a informação transformada, produzindo apenas mapeamentos estáticos.

Neste trabalho será utilizada uma arquitetura para as Redes Neurais de Elman, semelhante à representada na Figura 2.12:

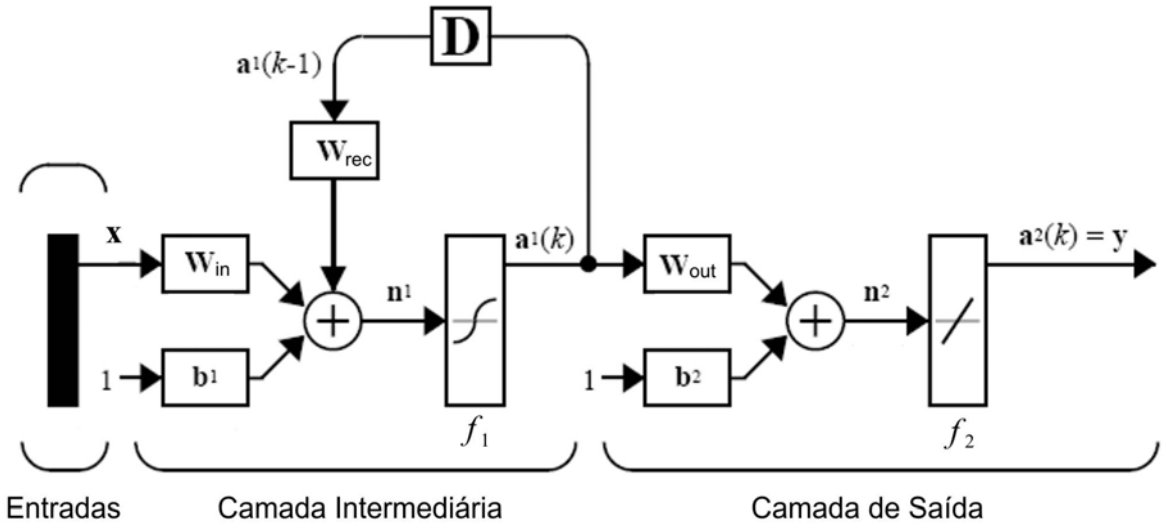


Figura 2.12 - Modelo da Rede Neural Recorrente utilizada.

Para a rede da Figura 2.12, também é possível deduzir as saídas em função das entradas. As equações (2.34) e (2.35) mostram o cálculo das saídas da camada intermediária em função das entradas e o cálculo das saídas em função das saídas da camada intermediária, respectivamente:

$$\mathbf{a}^1(k) = f_1\left(\mathbf{W}_{in}\mathbf{x}(k) + \mathbf{W}_{rec}\mathbf{a}^1(k-1) + \mathbf{b}^1\right) \tag{2.34}$$

$$\mathbf{a}^2(k) = \mathbf{y} = f_2\left(\mathbf{W}_{out}\mathbf{a}^1(k) + \mathbf{b}^2\right) \tag{2.35}$$

Das equações (2.34) e (2.35) é possível calcular a saída em função das entradas:

$$\mathbf{y} = f_2\left(\mathbf{W}_{out}\left[\, f_1\left(\mathbf{W}_{in}\mathbf{x}(k) + \mathbf{W}_{rec}\mathbf{a}^1(k-1) + \mathbf{b}^1\right)\right] + \mathbf{b}^2\right) \tag{2.36}$$

As funções de ativação $f_1(\cdot)$ e $f_2(\cdot)$ podem ser de vários tipos. Um exemplo comumente usado nas redes neurais de Elman é a tangente hiperbólica para $f_1(\cdot)$ e uma função puramente linear para $f_2(\cdot)$. Os símbolos que representam estas funções podem ser vistos nos neurônios da rede da Figura 2.12.

As redes neurais recorrentes podem ser treinadas da mesma maneira que as redes neurais MLP.

Como exemplos de aplicação de redes recorrentes em previsão, é possível citar o trabalho de Wan [49] que evidencia o comportamento dinâmico das redes recorrentes e mostra os benefícios de utilizá-las para previsão de séries temporais. Kuo, Principe e DeVries [26] utilizam uma rede neural recorrente para fazer previsão de séries temporais caóticas e, ao invés de prever um ponto à frente na série temporal, como é usual, eles fazem previsão de vários pontos à frente de uma só vez e conseguem obter um desempenho superior do que quando usada uma rede MLP convencional. Principe, Rathie e Kuo [37] usam uma rede neural recorrente para prever valores futuros de uma série temporal gerada por um sistema dinâmico e conseguem obter bons resultados.

2.3.4 Redes Neurais de Base Radial (RNAs-RBF)

As Redes Neurais RBF foram desenvolvidas baseadas na *teoria da aproximação*, onde residem também seus aspectos formais. O problema de descobrir um mapeamento de um espaço de entrada para um espaço de saída é essencialmente equivalente ao problema de sintetizar uma memória associativa que retorna uma saída aproximada quando apresentada aos dados de entrada e capaz de *generalizar* quando apresentada a novas entradas. Assim sendo, como as outras redes neurais, as Redes Neurais RBF têm o objetivo de servirem como uma função $f : \Re^m \to \Re^n$ que mapeia um espaço de entrada em um espaço de saída.

De acordo com a teoria da aproximação, quando se deseja, por exemplo, aproximar uma função contínua e multivariada $\{y(\mathbf{x}),\ \mathbf{x} \in \Re^{m\times 1}\}$ por uma função de aproximação $\{f(\mathbf{w},\mathbf{x}),\ \mathbf{w} \in \Re^{h\times 1},\ \mathbf{x} \in \Re^{m\times 1}\}$ dado um número fixo h de parâmetros $\mathbf{w}$, há, claramente, dois

aspectos importantes do modelo a serem definidos: a função $f(\cdot)$ e os parâmetros $\mathbf{w}$. Escolhida uma função $f(\cdot)$ específica, o problema se reduz a encontrar o conjunto de parâmetros $\mathbf{w}$ ótimo que fornece a melhor aproximação possível de $y(\cdot)$ para um determinado conjunto de dados de entrada.

Caso fosse utilizado um modelo $f(\mathbf{x})$ de regressão linear para aproximar uma função $y(\mathbf{x})$, este modelo assumiria a seguinte forma:

$$f(\mathbf{x}) = \sum_{j=1}^{m} w_j h_j(\mathbf{x}) \tag{2.37}$$

O modelo representado pela equação (2.37) é uma combinação linear de m funções fixas ($h_j(\mathbf{x}),\ j = 1,2,...,m$) geralmente denominadas *funções-base*, por analogia com o conceito de vetor gerado a partir da combinação linear de *vetores-base*. Se as funções-base e quaisquer parâmetros que elas possam ter são assumidos como fixos, e as funções-base $h_j(\mathbf{x}),\ j = 1,2,...,m$, forem não lineares, o modelo será não-linear mas *linear nos parâmetros* porque a flexibilidade ou habilidade de $f(\mathbf{x})$ assumir formas diferentes deriva apenas da liberdade de escolher diferentes valores para os coeficientes w_j, $j = 1,2,...,m,$ da combinação linear. Contudo, se além das funções-base serem não-lineares, os seus parâmetros também fossem ajustáveis, o modelos será não-linear e também *não-linear nos parâmetros*.

Em princípio, qualquer conjunto de funções poderia ser escolhido para formar uma base $\{h_j(\cdot),\ j = 1,2,...,m\}$. No entanto, existem conjuntos mais adequados para resolver problemas específicos e outros tipos de conjuntos mais flexíveis que apresentam desempenho adequado para uma ampla gama de problemas. Assim sendo, as funções de base radial (funções não-lineares), conhecidas também como RBF (do termo em inglês “Radial Basis Function”), surgem como uma

proposta de funções-base para serem utilizadas em qualquer modelo de regressão não-linear (linear ou não-linear nos parâmetros) e, particularmente, como função de ativação em qualquer tipo de rede neural multicamada.

Uma função de ativação de base radial é caracterizada por apresentar uma resposta que decresce (ou cresce) monotonicamente com a distância a um ponto central. O centro e a taxa de decrescimento (ou crescimento) em cada direção, que resultam na *dispersão* da função, são os parâmetros que podem ser ajustados nestas funções. Uma função de base radial monotonicamente decrescente típica é a função gaussiana, a sua função-base é dada na forma:

$$h_j(x) = \exp\left(-\frac{(x-c_j)^2}{r_j^2}\right),\ j = 1, 2, ..., m \tag{2.38}$$

A Figura 2.13 mostra uma função de base radial do tipo descrito pela equação (2.38).

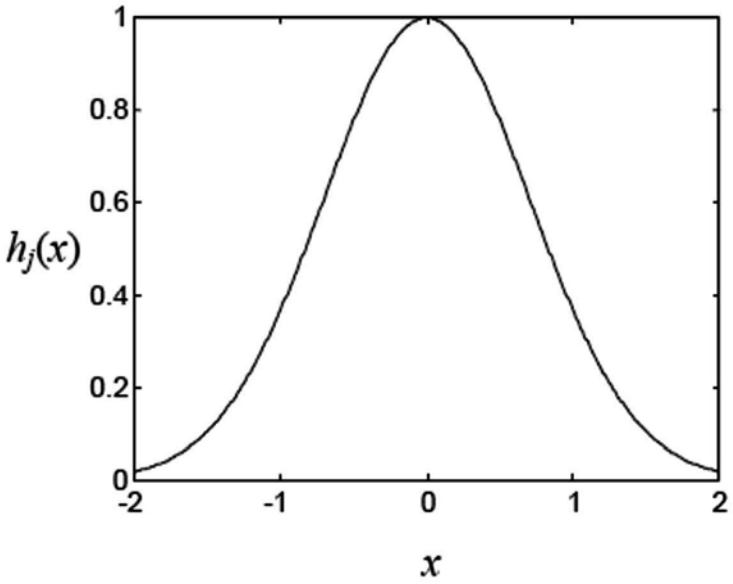


Figura 2.13 - Função de Base Radial tipo Gaussiana

A característica essencial de uma Rede Neural de Base Radial é que a camada de saída é sempre uma combinação linear dos sinais da camada oculta, sendo que neste tipo de rede neural se utiliza somente uma camada oculta (ou intermediária) onde todos os neurônios têm função de

ativação do tipo RBF, ou seja, nas redes neurais RBF somente os pesos na camada de saída são ajustáveis. Portanto, durante a fase de treinamento é possível ajustar os pesos na camada de saída, as dispersões e os centros das funções de base radial. Na Figura 2.14 é apresentada a estrutura básica de uma RNA-RBF.

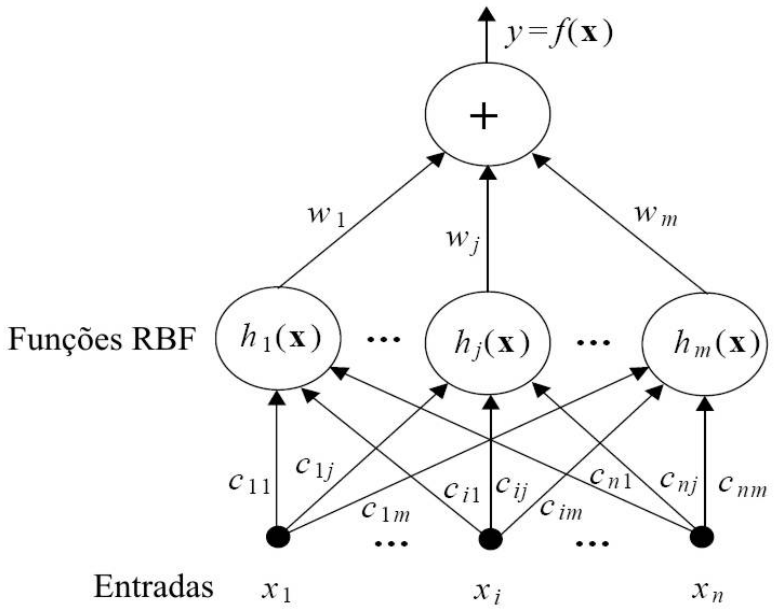


Figura 2.14 - Estrutura básica da RNA de Base Radial

De acordo com a Figura 2.14, a saída de uma RNA-RBF poderia ser calculada da seguinte maneira:

$$y = \sum_{j=1}^{m} w_j \exp\left(-\frac{\left(x_1 - c_{1,j}\right)}{\sigma_{1,j}} - \frac{\left(x_2 - c_{2,j}\right)}{\sigma_{2,j}} - ... - \frac{\left(x_i - c_{i,j}\right)}{\sigma_{i,j}} \right),\ i = 1,...,n \qquad (2.39)$$

Conforme explicitado pela equação (2.39), a conseqüência imediata do uso de funções de ativação de base radial está na forma como as entradas são processadas pelos neurônios da camada intermediária. Ao invés da ativação interna de cada neurônio da camada intermediária se dar pelo emprego do produto escalar entre o vetor de entradas e o vetor de pesos, como no caso

das redes MLP, ela é obtida a partir de uma norma ponderada da diferença entre ambos os vetores.

As RNAs-RBF permitem um procedimento rápido de atualização dos pesos. Este procedimento consiste em separar os padrões de entrada com hiperelipsóides e apresentar uma estrutura apta à construção de mapeamentos locais dos dados de entrada e saída. Existem várias propostas para treinamento das RNAs-RBF; um exemplo é o algoritmo de aprendizado proposto por Holcomb & Morari [19] que introduz incrementalmente funções de base radial para a camada intermediária, de modo que cada função seja aproximadamente ortogonal às demais. O algoritmo inicia com um único neurônio e outros neurônios são adicionados à rede quando necessário. A localização dos centros da função de base radial associada a cada neurônio é otimizada usando técnicas convencionais de otimização. Neste trabalho será utilizada uma técnica de treinamento similar a esta, conforme técnica desenvolvida por Chen, Cowan e Grant [5] que ajusta os centros das funções de base radial um a um até que uma rede adequada tenha sido construída. Este algoritmo tem a propriedade de que cada centro individualmente maximiza o incremento da variância em relação à saída desejada, evitando, desta forma, o mau condicionamento numérico em determinados tipos de problema.

O trabalho de Coelho e Canciglieri Júnior [7] é um exemplo de aplicação de redes neurais de base radial na predição de séries temporais. Eles utilizam séries temporais clássicas e chegam à conclusão que o modelo por eles proposto é atrativo para aplicações em identificação de sistemas complexos e mercado financeiro.

2.4 Sistemas Nebulosos

O conceito de conjuntos nebulosos foi introduzido por Zadeh [54] com o intuito de representar dados que não são precisos, ou seja, são nebulosos. Da mesma maneira que existe uma forte relação entre a lógica Booleana e a teoria de conjuntos clássica, há também uma forte relação entre a lógica nebulosa e a teoria de conjuntos nebulosos.

Na teoria clássica de conjuntos, um subconjunto *A* de *X* pode ser definido através de uma função característica χ_A como um mapeamento dos elementos de *X* para os elementos do conjunto $\{0,1\}$. Dada uma entrada *x*, quando este mapeamento resulta um valor zero, isto poderia significar, por exemplo, que o elemento *x* não pertence a *A*, e caso contrário, este elemento pertenceria a *A*. Portanto, a proposição "*x* está em *A*" pode ser determinada pelo par ordenado $(x, \chi_A(x))$. Assim, um determinado elemento *x* pertence a *A* se o segundo elemento do par ordenado é 1, e não pertence a *A* caso o segundo elemento seja igual a 0.

De maneira análoga, um subconjunto *nebuloso A* de *X* pode ser definido como um conjunto de pares ordenados, onde o primeiro elemento do par ordenado traz os elementos de *X* e o segundo elemento pode assumir valores no intervalo [0,1]. Novamente, se o segundo elemento do par ordenado assumir o valor zero, significa que *x* não pertence a *A* e se for um, *x* pertence a *A*. No entanto, agora é possível definir também valores entre 0 e 1 para um determinado elemento de *X*. Assim, se diz que um determinado elemento *x* de *X* pertence a *A* com um determinado *grau de pertinência* $\mu_A(x)$, e quando $\mu_A(x)=0$ e $\mu_A(x)=1$, *x* não pertence a *A* e pertence a *A*, respectivamente, e no caso $0<\mu_A(x)<1$, *x* pertence a *A* com um determinado grau de pertinência $\mu_A(x)$, ou seja, quando se qualifica um elemento *x* como pertencendo ou não a um determinado conjunto *A*, é possível atribuir um grau de pertinência de *x* ao conjunto *A*.

Enfim, com a teoria de conjuntos nebulosos, a proposição "*x* está em *A*" é determinada pelo par ordenado $(x, \mu_A(x))$ e o "*grau de verdade*" desta proposição é determinado pelo segundo elemento do par ordenado. Assim, $\mu_A(x)$ pode ser descrita como uma função que mapeia elementos de *A* em relação aos seus *graus de pertinência* do conjunto *A*. $\mu_A(x)$ também é conhecida como *função de pertinência* e é definido da seguinte forma:

Definição [2] *Seja X um conjunto não vazio. Um conjunto nebuloso A em X é caracterizado pela função de pertinência* $\mu_A : X \to [0,1]$ *e* $\mu_A(x)$ *é interpretado como o grau de pertinência do elemento* $x \in X$ *ao conjunto nebuloso A.*

Assim, *A* é determinado da seguinte forma:

$$A = \left\{ \left(x, \mu_A(x) \right) \mid x \in X \right\} \tag{2.40}$$

Contudo, normalmente a notação $A(x)$ é utilizada ao invés de $\mu_A(x)$. Ainda assim, quando se tem um conjunto discreto $X = \{x_1, x_2, ..., x_n\}$ e *A* é um conjunto nebuloso em *X*, é comumente usada a notação:

$$A = \mu_1 / x_1 + \mu_2 / x_2 + ... + \mu_n / x_n , \tag{2.41}$$

onde o termo $\mu_i / x_i , i = 1,2,..., n$, significa que μ_i é o grau de pertinência de x_i em *A* e o sinal de mais representa a união e não a soma.

Para exemplificar o emprego de uma função de pertinência, será tomado o problema de modelar os números reais próximos a um. Uma possível função de pertinência que seria capaz de modelar este problema é:

$$A(x) = \exp\left(-\beta \left(x - 1 \right)^2 \right), \tag{2.42}$$

onde $\beta \in \Re^+$.

A função de pertinência da equação (2.42) é ilustrada na Figura 2.15.

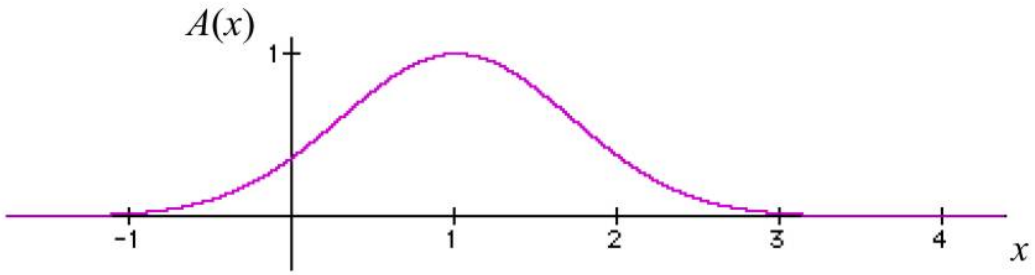


Figura 2.15 - Exemplo de Função de Pertinência (números reais próximos a 1).

2.4.1 Redes Neurais Nebulosas

Os sistemas híbridos que combinam lógica nebulosa, redes neurais, algoritmos genéticos e sistemas especialistas proporcionam métodos eficientes para resolver uma grande variedade de problemas. Cada uma dessas técnicas tem propriedades computacionais particulares (por exemplo, habilidade de aprender, explanação das decisões) que as tornam especialistas para certos problemas particulares e não para outros. Por exemplo, as redes neurais são boas em reconhecimento de padrões, não são capazes de explicar como alcançam suas decisões. Os sistemas da lógica nebulosa, que podem processar informação imprecisa, são bons em explicar suas decisões, mas não são capazes de adquirir automaticamente as regras que sustentam aquelas decisões. Estas limitações foram à motivação principal para a criação dos sistemas Híbridos Inteligentes, onde duas ou mais técnicas são combinadas de maneira que superem as limitações de técnicas individuais. Os sistemas híbridos são também importantes ao considerar a natureza variada de domínios da aplicação.

O uso de sistemas híbridos inteligentes está crescendo rapidamente com aplicações bem sucedidas em muitas áreas, incluindo controle de processos, engenharia, negócios e finanças, avaliação do crédito e diagnóstico médico. Um desses sistemas híbridos corresponde aos sistemas

neurais nebulosos, que combinam as técnicas de redes neurais e as técnicas de inferência nebulosas.

A lógica nebulosa fornece um mecanismo de inferência sobre a incerteza, e as redes neurais oferecem grandes vantagens computacionais, tais como a aprendizagem, adaptação, o paralelismo e a generalização.

O processo computacional para projetar sistemas neurais nebulosos começa com o desenvolvimento de um “neurônio nebuloso” baseado na compreensão de morfologias neurobiológicas, seguida por mecanismos de aprendizagem. Isto conduz às seguintes três etapas em um processo computacional neural nebuloso:

- Desenvolvimento de modelos neurais nebulosos motivados pelos neurônios biológicos;
- Modelos de conexões sinapticas que incorpora a nebulosidade na rede neural;
- Desenvolvimento de algoritmos de aprendizado.

Existem dois possíveis modelos de sistemas neurais nebulosos [15]. O primeiro é baseado em indicações lingüísticas, em que o bloco de relações nebulosas fornece um vetor da entrada a uma rede neural multicamadas. A rede neural pode ser adaptada (treinada) para fornecer saídas desejadas ou decisões. O modelo é o seguinte:

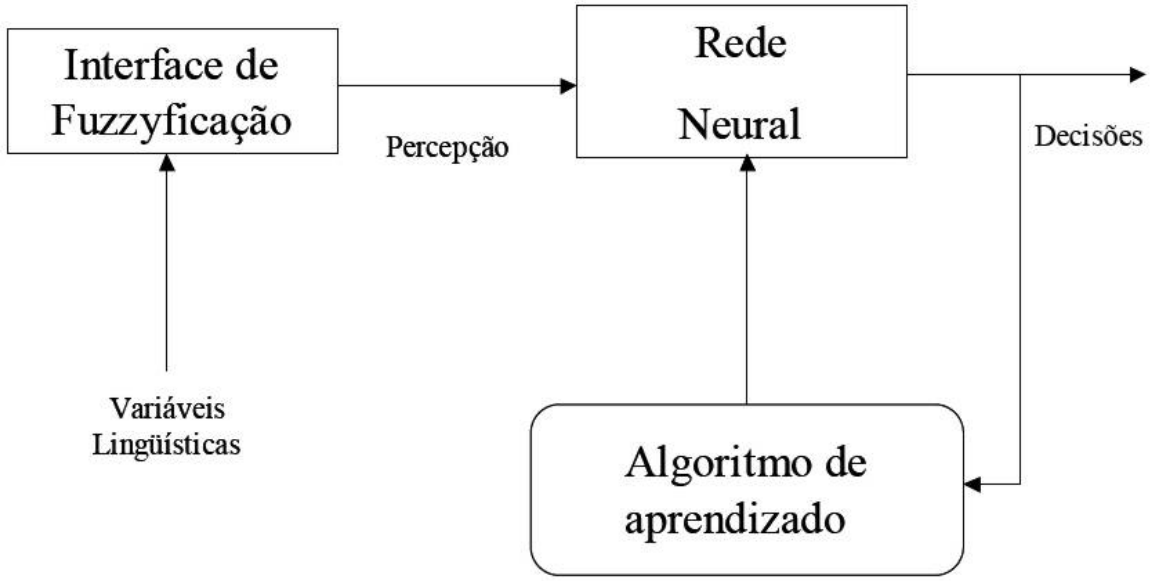


Figura 2.16 - Primeiro modelo de sistema neural nebuloso.

O segundo é uma rede multicamadas que dirige o mecanismo de inferência nebulosa:

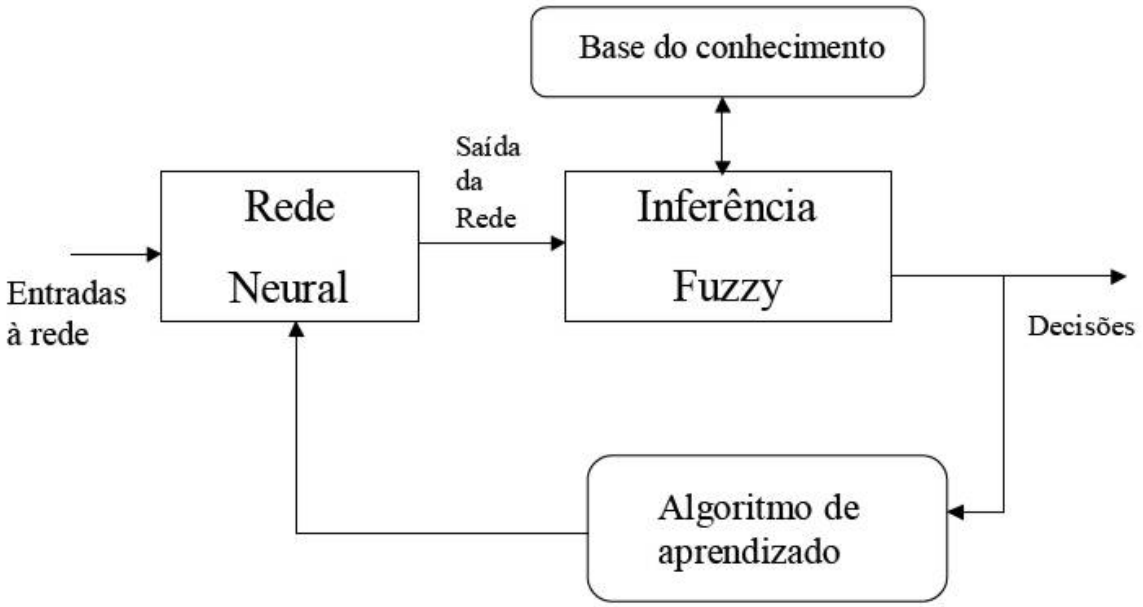


Figura 2.17 - Segundo modelo de sistema neural nebuloso.

As redes neurais são usadas para representar os sistemas de inferência nebulosa, os mesmos que são empregados como sistemas de tomada de decisões. Embora a lógica nebulosa possa

codificar o conhecimento com etiquetas lingüísticas, usualmente toma muito tempo definir e ajustar as funções de pertinência. As técnicas de aprendizado das redes neurais podem automatizar este processo e substancialmente reduzir o tempo e o custo de desenvolvimento ao melhorar o desempenho.

Teoricamente, as redes neurais e os sistemas nebulosos são equivalentes no que se diz respeito à capacidade de aprendizado, mas na prática cada um tem suas próprias vantagens e desvantagens.

Para redes neurais, o conhecimento é adquirido automaticamente através do treinamento da rede, mas o processo de aprendizagem é relativamente lento e a análise da rede treinada é difícil. Não é possível extrair o conhecimento estrutural (regras) da rede neural treinada, nem se pode integrar ao modelo nenhum conhecimento prévio sobre o problema a fim simplificar o procedimento de aprendizagem.

Por outro lado, os sistemas nebulosos são mais favoráveis, porque seu comportamento pode ser explicado com base em regras nebulosas, e assim seu desempenho pode ser ajustado mudando as regras. Mas, a aquisição de conhecimento é difícil, bem como no que diz respeito ao conjunto de variáveis de entrada, pois como necessita ser dividido em diversos intervalos, as aplicações de sistemas nebulosos ficam restritas aos problemas em que o conhecimento está disponível e o número de variáveis de entrada é pequeno. Para superar o problema da aquisição de conhecimento, as redes neurais são estendidas para extrair automaticamente regras nebulosas dos dados numéricos.

A seguir, apresenta-se um modelo neural nebuloso proposto por Jang [22]. Ele é conhecido como ANFIS e é baseado em sistemas de inferência nebulosa tipo Takagi-Sugeno [T-S] [42].

2.4.2 O modelo ANFIS

A sigla ANFIS deriva do termo em inglês "*adaptive neuro-fuzzy inference system*" ou *sistema adaptativo de inferência neural-nebulosa*. O modelo ANFIS foi proposto por Jang [22] e consiste em um sistema híbrido, o mesmo que é funcionalmente equivalente ao mecanismo de inferência [T-S]. A base de regras é a mesma usada por Takagi e Sugeno [42].

Um sistema de inferência nebulosa [T-S] tem a seguinte estrutura:

$$\text{Se } x \text{ é } A \text{ e } y \text{ é } B, \text{ então } z = f(x, y), \tag{2.43}$$

o que significa que o resultado da inferência será uma função das variáveis de entrada. Tomando como base este tipo de inferência, Jang [22] desenvolveu o modelo ANFIS, que utiliza uma função linear $z = f(x, y)$ como resposta do sistema de inferência. A seguir, é apresentada a estrutura do modelo ANFIS.

Considere um sistema de inferência nebulosa com duas entradas x e y e uma saída z representada por uma combinação linear das entradas. Para um sistema tipo [T-S] de primeira ordem, um conjunto de regras nebulosas SE-ENTÃO é o seguinte:

- Regra 1: Se x é A_1 e y é A_2, então $z_1 = a_1 x + b_1 y + c_1$
- Regra 2: Se x é B_1 e y é B_2, então $z_2 = a_2 x + b_2 y + c_2$

A Figura 2.18 ilustra este sistema de inferência nebulosa.

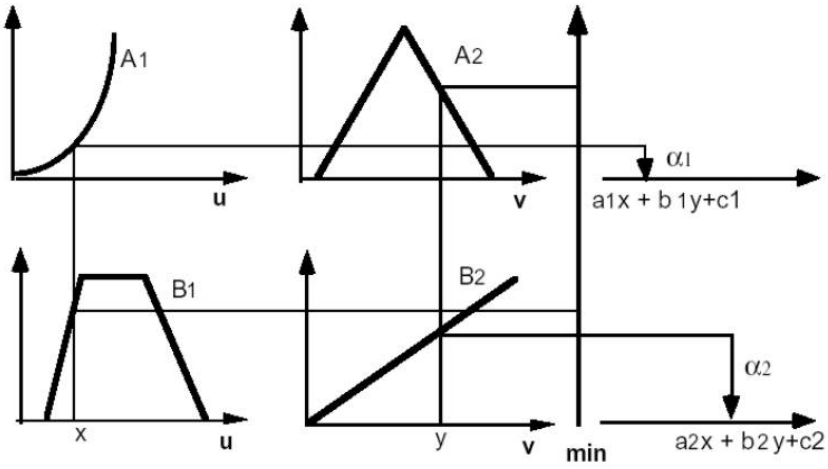


Figura 2.18 - Sistema de Inferência tipo [T-S].

Como o operador lógico "e" pode ser modelado por uma t-norma contínua, os níveis de ativação α_1 e α_2 das regras para um dado valor (x_0, y_0) são calculados como:

$$\alpha_1 = A_1(x_0) \wedge A_2(y_0) \tag{2.44}$$

$$\alpha_2 = B_1(x_0) \wedge B_2(y_0) \tag{2.45}$$

Como se sabe, as saídas individuais de cada regra são calculadas da seguinte forma:

$$z_1 = a_1 x_0 + b_1 y_0 + c_1 \tag{2.46}$$

$$z_2 = a_2 x_0 + b_2 y_0 + c_2 \tag{2.47}$$

A saída é obtida da seguinte maneira:

$$z_0 = \frac{\alpha_1 z_1 + \alpha_2 z_2}{\alpha_1 + \alpha_2} \Rightarrow z_0 = \beta_1 z_1 + \beta_2 z_2, \tag{2.48}$$

onde β_1 e β_2 são os valores normalizados de α_1 e α_2 em relação à soma $(\alpha_1 + \alpha_2)$. A rede neural híbrida que representa este tipo de inferência é uma rede adaptativa com cinco camadas:

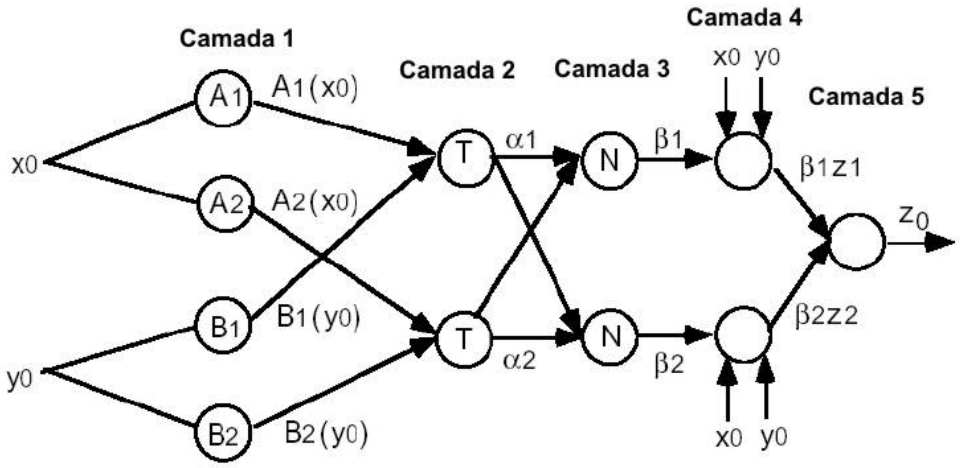


Figura 2.19 - Arquitetura ANFIS para inferência tipo [T-S].

Nesta arquitetura, todos os nós de uma mesma camada têm o mesmo papel. Cada uma das camadas representa uma parte do mecanismo de inferência:

- *Camada 1*: As entradas nesta camada correspondem às entradas *x* e *y*, e a saída dos nós é o grau de pertinência para o qual a variável de entrada satisfaz o termo lingüístico associado a este nó. Para representar o termo lingüístico, se usa uma função de pertinência. Por exemplo, pode-se usar uma função do tipo gaussiana:

$$A_i(x_0) = \exp\left[-\frac{1}{2}\left(\frac{x_0 - a_{i1}}{b_{i1}}\right)^2\right] \tag{2.49}$$

$$B_i(y_0) = \exp\left[-\frac{1}{2}\left(\frac{y_0 - a_{i2}}{b_{i2}}\right)^2\right] \tag{2.50}$$

Onde $\{a_{i1}, a_{i2}, b_{i1}, b_{i2}\}$ correspondem ao conjunto de parâmetros. Caso o valor desses parâmetros mude, o resultado das equações (2.49) e (2.50) mudará de acordo com eles. Em geral, qualquer função de pertinência, como a trapezoidal e a triangular, pode ser usada nos nós desta camada.

- *Camada 2*: Cada nó calcula o grau de ativação da regra associada. A saída dos neurônios é:

$$\alpha_1 = A_1(x_0) \wedge B_1(y_0) \tag{2.51}$$

$$\alpha_2 = A_2(x_0) \wedge B_2(y_0) \tag{2.52}$$

Ambos os nós estão representados com um “T” na Figura 2.19, pelo fato de que eles podem representar qualquer t-norma para modelar a operação lógica “e”. Os nós desta camada são conhecidos como nós de regras e funcionam como operadores de agregação.

- *Camada 3*: Cada nó nesta camada está representada por um “N” na Figura 2.19, para indicar a normalização dos graus de ativação. A saída do neurônio é o grau de ativação normalizado (com respeito à soma dos graus de ativação) da regra *i*.

$$\beta_1 = \frac{\alpha_1}{\alpha_1 + \alpha_2},\ \beta_2 = \frac{\alpha_2}{\alpha_1 + \alpha_2} \tag{2.53}$$

- *Camada 4*: A saída dos neurônios corresponde ao produto entre o grau de ativação normalizado multiplicado pela saída individual de cada regra:

$$\beta_1 z_1 = \beta_1 (a_1 x_0 + b_1 y_0 + c_1) \tag{2.54}$$

$$\beta_2 z_2 = \beta_2 (a_2 x_0 + b_2 y_0 + c_2) \tag{2.55}$$

- *Camada 5*: O único nó desta camada calcula a saída total do sistema como sendo a soma de todas as entradas individuais deste nó:

$$z_0 = \beta_1 z_1 + \beta_2 z_2 \tag{2.56}$$

2.4.2.1 Aprendizado no modelo ANFIS

Quando um conjunto de treinamento clássico do tipo $\{(x_k, y_k),\ k = 1,2,...,K\}$ é apresentado na camada de entrada da rede, então os parâmetros da rede neural híbrida (que determinam a

forma da função de pertinência) podem ser ajustados usando métodos de otimização não linear irrestrita. Um exemplo é o método do gradiente.

Na arquitetura ANFIS, a saída é expressa como uma combinação linear dos parâmetros do conseqüente. Assim a saída *z* é:

$$z = (\beta_1 x)a_1 + (\beta_1 y)b_1 + (\beta_1)c_1 + (\beta_2 x)a_2 + (\beta_2 y)b_2 + (\beta_2)c_2 \qquad (2.57)$$

Com o conhecimento das saídas, basta calcular o erro em relação às saídas desejadas e utilizar um algoritmo que corrija os parâmetros $\{a_1, b_1, c_1, a_2, b_2, c_2\}$ de tal forma que esta correção minimize este erro.

A regra de aprendizado híbrido combina o método do gradiente e o método de mínimos quadrados para atualizar os parâmetros no ANFIS. Para que a aprendizagem híbrida seja aplicada em grupo, cada época é composta de um passo *forward* e de um passo *backward*. No passo *forward*, os parâmetros das funções de pertinência são inicializados e um vetor de entradas e outro de saída desejada são apresentados. Calcula-se a saída da rede e, em seguida, o erro é calculado como a diferença entre a saída da rede e a saída desejada. No passo *backward*, os sinais do erro são propagados a partir da saída e em direção às entradas. O vetor gradiente é acumulado para cada dado de treinamento. No final do passo *backward* para todos os dados de treinamento, os parâmetros na camada de entrada (os parâmetros das funções de pertinência) são atualizados pelo método do gradiente.

Há também trabalhos como o de González, Rojas e Pomares [16] que utilizam computação evolutiva para acelerar a convergência do treinamento das redes neurais nebulosas e utilizam estas redes evoluídas com operadores genéticos para fazer previsão de séries temporais. Como um exemplo de aplicação ainda mais voltado para este trabalho, é possível citar a tese de doutorado de Ballini [2], onde é desenvolvido um previsor neural nebuloso para tratar o problema

da previsão de vazões naturais que afetam o sistema hidroelétrico de potência brasileiro. O trabalho de Ballini, da mesma maneira que este, serve de subsídio para orientar as tomadas de decisão no que diz respeito à operação e ao planejamento do sistema elétrico brasileiro, com o objetivo de minimizar o custo de produção de energia elétrica.

2.5 Os Ensembles

2.5.1 Introdução

Em 1990, Hansen e Salamon [18] mostraram pela primeira vez que a capacidade de generalização de um sistema neural poderia ser melhorado combinando várias redes neurais, como, por exemplo, treinando as redes individualmente e combinando os seus resultados de alguma maneira.

Portanto, ensembles ou agrupamento de regressores foi definido como uma forma de combinar soluções obtidas de múltiplos regressores. A vantagem de fazer este tipo de combinação vem do fato de que há uma tendência de se obter um resultado melhor do que aquele obtido individualmente pelo melhor dos previsores.

Em geral, um ensemble de redes neurais é construído em duas etapas. Na primeira, um determinado número de redes neurais é treinado e, na segunda etapa, os resultados das redes são combinados de alguma maneira.

Para o treinamento das redes neurais individualmente, existem diversas técnicas. Os métodos mais conhecidos são ***Bagging*** e ***Boosting***. Boosting foi proposto por Shapire [40] e aprimorado por Freund *et al.* [13], [14] e funciona gerando uma série de redes neurais nas quais os conjuntos de treinamento são determinados pela performance das redes neurais antecessoras, de tal forma que, quando ocorrem erros grandes em determinadas redes e em determinados

pontos de avaliação, será dada ênfase especial a estes pontos no treinamento das redes na etapa seguinte. Bagging foi proposto por Breiman [4] baseado na técnica de “bootstrap sampling” [11] e propõe a geração de vários conjuntos de treinamento diferentes a partir do conjunto de treinamento original. Em seguida, cada uma das redes é treinada individualmente com os conjuntos de treinamento gerados. Ainda existem várias outras metodologias de treinamento desenvolvidas por outros autores. Cherkauer [6] treina as redes neurais individualmente com diferentes números de neurônios na camada intermediária. Krogh & Vedelsby [25] utilizam validação cruzada para criar redes neurais individuais. Hampshire & Waibel [17] usam diferentes funções de ativação para treinar diferentemente as redes neurais. Maclin & Shavlik [29] inicializam as matrizes de pesos das redes de maneira diferente. Opitz & Shavlik [35] exploram os parâmetros das redes individuais utilizando um algoritmo genético. Yao & Liu [53] combinam todas as redes evoluídas utilizando um algoritmo evolutivo. Liu & Yao [28] usam aprendizado correlacionado negativamente para gerar uma população de redes neurais correlacionadas negativamente entre si.

Na segunda etapa, para combinar os resultados da população de regressores, há várias abordagens diferentes. As mais comuns são votação múltipla ou votação majoritária para problemas de classificação de padrões [18] e média aritmética [34] ou média ponderada [36] para problemas de regressão. Ainda existem várias outras abordagens para combinar múltiplos previsores. Ueda [45] explora a ponderação dos pesos das redes utilizando otimização linear para calcula-los. Jimenez [23] emprega pesos dinâmicos determinados em função da exatidão de cada uma das redes. Wolpert [51] utiliza sistemas de aprendizado, como redes neurais, para combinar as redes individuais.

2.5.2 Motivação para a utilização de ensembles

Tanto observações empíricas quanto em aplicações em determinados previsores específicos confirmam que, dado um algoritmo de aprendizado, é possível que ele seja melhor do que qualquer outro para prever determinado subconjunto de dados de entrada. No entanto, é praticamente impossível que um determinado previsor seja capaz de prever melhor do que qualquer outro previsor para todo o domínio de dados de entrada. Assim sendo, a utilização de múltiplos previsores tenta explorar o bom comportamento local de cada um dos previsores e, com isto, aumenta a precisão e a confiabilidade da previsão, uma vez que se um dos previsores errar muito em determinado subconjunto de dados de entrada, os outros previsores tendem a compensar o erro deste.

Contudo, existem razões mais profundas para um ensemble ter desempenho superior quando comparado a um especialista individualmente. Para exemplificar, pode-se usar um problema de classificação dicotômica desenvolvida por Dietterich [10]. Neste exemplo, se existir L hipóteses com erros menores que 0.5, então o ensemble resultante usando a técnica de votação majoritária tem um erro menor do que um só classificador sozinho, uma vez que os erros dos classificadores não são correlacionados ou são negativamente correlacionados. De fato, se forem considerados 21 classificadores, a probabilidade de erro individual de cada classificador for $p = 0,3$ e os erros forem independentes, o erro do ensemble por votação majoritária é dado pela área sob a distribuição binomial, onde mais de $L/2$ hipóteses são falsas:

$$P_{ensemble} = \sum_{(i=\lceil L/2 \rceil)}^{L} \binom{L}{i} p^i \left(1-p\right)^{L-i} \Rightarrow P_{ensemble} = 0,026 << p = 0,3 \qquad (2.58)$$

Este resultado tem sido estudado pelos matemáticos desde o século XVIII e, no contexto das ciências sociais, um exemplo é o trabalho do filósofo Condorcet [8], onde ele propõe o

Teorema do Júri de Condorcet provando que o julgamento de um júri é superior em comparação ao julgamento de um só indivíduo. Isto acontece, pois o julgamento de um só indivíduo, quando lhe é perguntado sim ou não sobre um tema, a probabilidade de erro dele é 0,5, ou seja, uma em duas possibilidades de julgamento (sim/não). No entanto, quando mais indivíduos são consultados, a probabilidade de erro diminui, como já foi constatado anteriormente através da utilização da equação (2.58), contudo, para o Teorema do Júri $p = 0,5$.

2.5.3 Selecionando os Previsores: Generalização e Correlação

Suponha que se deseje aproximar a função $f : \Re^m \to \Re^n$ usando um ensemble que combina N redes neurais individuais. Os mapeamentos de cada uma das redes neurais individuais são combinados usando média ponderada dos previsores, onde um peso $\omega_i (i = 1,2,..., N)$ é atribuído à uma rede individual f_i e satisfaz as equações (2.59) e (2.60):

$$0 \le \omega_i \le 1 \tag{2.59}$$

$$\sum_{i=1}^{N} \omega_i = 1 \tag{2.60}$$

Por conveniência, se assume que cada rede individualmente tem somente uma saída. Assim a função que se deseja aproximar pode ser expressa como $f : \Re^m \to \Re$. Entretanto, se deve notar que o desenvolvimento exposto a seguir pode ser facilmente generalizado para o caso de redes com mais de uma saída.

Supõe-se uma entrada $x \in \Re^m$ aleatoriamente escolhida de acordo com uma distribuição $p(x)$. A saída esperada de x é $d(x)$. Entretanto, a saída efetiva da i-ésima rede neural é $f_i(x)$. Então, a saída do ensemble pode ser escrita como:

$$\overline{f}(x) = \sum_{i=1}^{N} \omega_i f_i(x) \tag{2.61}$$

O erro de generalização $E_i(x)$ da i-ésima rede neural e o erro de generalização $E(x)$ do ensemble, dada uma entrada x, são respectivamente:

$$E_i(x) = \left(f_i(x) - d(x) \right)^2 \tag{2.62}$$

$$E(x) = \left(\overline{f}(x) - d(x) \right)^2 \tag{2.63}$$

Então, o erro de generalização E_i da i-ésima rede neural e o erro de generalização E do ensemble na distribuição $p(x)$ são respectivamente:

$$E_i = \int p(x) E_i(x) dx \tag{2.64}$$

$$E = \int p(x) E(x) dx \tag{2.65}$$

O erro médio de generalização das redes neurais que compõem o ensemble em x e o erro delas em $p(x)$ são respectivamente:

$$\overline{E}(x) = \sum_{i=1}^{N} \omega_i E_i(x) \tag{2.66}$$

$$\overline{E} = \int p(x) \overline{E}(x) dx \tag{2.67}$$

Agora se define o coeficiente de correlação entre a i-ésima e a j-ésima rede neural individual como:

$$C_{ij} = \int p(x) \left(f_i(x) - d(x) \right) \left(f_j(x) - d(x) \right) dx \tag{2.68}$$

Note que C_{ij} deve satisfazer as equações (2.69) e (2.70):

$$C_{ii} = E_i \tag{2.69}$$

$$C_{ij} = C_{ji} \tag{2.70}$$

Das equações (2.61) e (2.63), se obtém o seguinte:

$$E(x) = \left(\sum_{i=1}^{N} \omega_i f_i(x) - d(x) \right) \left(\sum_{j=1}^{N} \omega_j f_j(x) - d(x) \right) \tag{2.71}$$

Considerando as equações (2.71) e (2.68) é possível deduzir que:

$$E = \sum_{i=1}^{N} \sum_{j=1}^{N} \omega_i \omega_j C_{ij} \tag{2.72}$$

Assumindo que a combinação das redes seja feita utilizando média simples, então temos que $\omega_i = 1/N \ (i = 1,2,..., N)$ e da equação (2.72) é possível deduzir que:

$$E = \sum_{i=1}^{N} \sum_{j=1}^{N} C_{ij} / N^2 \tag{2.73}$$

Agora, supondo que uma k-ésima rede neural é retirada do ensemble, o erro de generalização do novo ensemble passa a ser:

$$E' = \sum_{\substack{i=1 \\ i \neq k}}^{N} \sum_{\substack{j=1 \\ j \neq k}}^{N} C_{ij} / (N-1)^2 \tag{2.74}$$

Considerando as equações (2.73) e (2.74), é possível obter o seguinte:

$$E - E' = \frac{2 \sum_{\substack{i=1 \\ i \neq k}}^{N} C_{ik} + E_k - (2N-1)E}{(N-1)^2} \tag{2.75}$$

Analisando a equação (2.75) é possível perceber que, para satisfazer a situação desejada, ou seja, onde o erro do ensemble com todas as redes é maior do que o erro de generalização do ensemble com uma rede a menos ($E > E'$), é necessário que o numerador da equação (2.75) seja positivo. Assim:

$$E < \frac{2\sum_{\substack{i=1 \\ i \neq k}}^{N} C_{ik} + E_k}{2N - 1} \tag{2.76}$$

Combinando as equações (2.76) e (2.73):

$$(2N-1)\sum_{i=1}^{N}\sum_{j=1}^{N} C_{ij} < 2N^2 \sum_{\substack{i=1 \\ i \neq k}}^{N} C_{ik} + N^2 E_k \tag{2.77}$$

Finalmente, é possível concluir que, em alguns casos, é preferível combinar um subconjunto das redes neurais individuais do que combinar todas. É interessante perceber que as redes individuais que podem ser retiradas do ensemble devem satisfazer a equação (2.77). No entanto, o esforço computacional exigido para identificar quais as redes neurais individuais que serão excluídas do ensemble é muito alto. Assim, será proposta neste trabalho uma técnica para selecionar o subconjunto de redes que irão participar do ensemble. Esta técnica é descrita em vários trabalhos já realizados, como em [52], [58] e [56], e se revelou eficiente e com custo computacional bem mais reduzido.

2.5.4 Estrutura e Características dos Ensembles

As características dos ensembles são listadas abaixo:

- É garantido, teoricamente, que o ensemble é melhor ou pelo menos tão bom quanto o melhor previsor da população [36];
- Usa eficientemente todos os dados do problema disponíveis, evitando “overfitting” [36];
- Implicitamente realiza regularização, pois “suaviza” o espaço funcional, evitando, desta forma, “overfitting” [36];
- Quando todos os previsores na população têm desempenho parecido e o erro que cada um apresenta é independente, o desempenho do ensemble fica evidente [46].

Para que o funcionamento de um ensemble seja bem compreendido, abaixo é apresentado um diagrama de um ensemble.

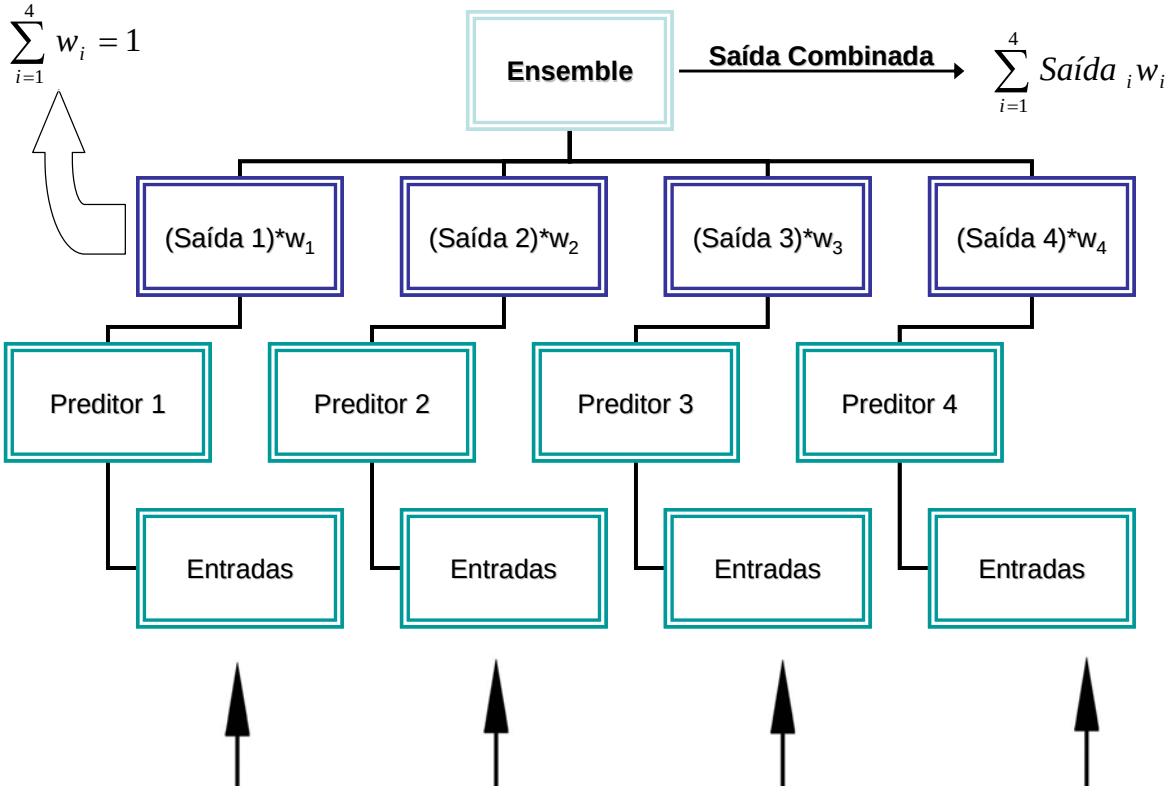


Figura 2.20 - Estrutura de um ensemble de previsores, considerando 4 componentes.

Como vemos, a Figura 2.20 representa um ensemble que combina as saídas de 4 previsores. O processo global ocorre da seguinte maneira:

- Cada previsor recebe os dados de entrada do problema (os dados podem não ser os mesmos para previsores distintos);
- Cada previsor faz um mapeamento $f : \Re^m \rightarrow \Re$ da saída a partir de m entradas;
- Cada saída de cada previsor é multiplicada por um peso, onde a somatória de todos os pesos impostos às saídas deve ser igual a 1;

- As saídas normalmente são selecionadas para formar o ensemble, de tal forma que nem todas as saídas dos previsores são combinadas, mas somente aquelas que melhoram o desempenho do ensemble.
- As saídas selecionadas e ponderadas são somadas para compor a solução do ensemble.

Como exemplos de aplicação de ensembles, se pode citar o trabalho de Lima, Coelho e Von Zuben [27] que utiliza ensembles de Máquinas de Vetores de Suporte para resolver problemas de regressão paramétrica. Assaad e Boné [1] usam ensembles de redes neurais recorrentes gerados pela técnica Boosting para fazer previsões de séries temporais. Inoue e Narihisa [21] propõem um ensemble de redes neurais cujas arquiteturas são evoluídas com a utilização de algoritmos genéticos e programação evolutiva. Eles concluem que o desempenho do ensemble melhora na medida em que são adicionadas mais redes neurais e que quanto maior o conjunto de treinamento, melhor a previsão de séries temporais caóticas. Rothermich [39] faz previsões de séries temporais financeiras utilizando redes neurais negativamente correlacionadas e obtém bons resultados. Toulson e Toulson [44] usam ensembles de redes neurais para prever o risco futuro de determinadas carteiras de investimento e isto os orienta a tomar decisões para selecionar determinados portfólios mais interessantes. Magalhães [30] usa uma rede neural com um neurônio para combinar múltiplas previsões de vazões naturais. O ensemble final utiliza os pesos das conexões sinápticas de sua rede neural para ponderar a utilização de cada previsor quando o ensemble é combinado.

3 Os Algoritmos Genéticos e os Ensembles

Como foi visto, os ensembles têm os seguintes aspectos:

- Trata-se de uma abordagem ***populacional***, ou seja, vários previsores são combinados para compor a solução final;
- Os melhores previsores são ***selecionados*** para compor o ensemble, sendo que esta ***seleção*** tem papel fundamental no bom desempenho do ensemble.

Somente analisando estes aspectos, fica evidente qual seria o papel dos algoritmos genéticos neste cenário: os algoritmos genéticos são ferramentas poderosas na evolução de ***populações*** de possíveis soluções de um problema através da ***seleção*** dos indivíduos mais bem preparados.

A aplicação de algoritmos genéticos na seleção de previsores em ensembles é recente e o primeiro trabalho publicado sobre esta técnica foi proposto por Zhou, Wu, Jiang e Chen [58]. Nesta publicação, eles nomeiam esta técnica de ***GASEN***, nome que derivou do próprio título do trabalho. Este trabalho chegou a ganhar prêmios e desencadeou uma série de outros trabalhos também executados pela mesma equipe chinesa. O próximo trabalho publicado [57] trata basicamente dos mesmos aspectos e em um outro trabalho [52] eles propõem construir vários ensembles e combinar os resultados deles, ou seja, é o ensemble dos ensembles. Esta nova técnica foi nomeada de ***e-GASEN***. Em trabalho mais recente [56] eles mostram que os ensembles gerados por ***GASEN*** podem ter mais capacidade de generalização do que aqueles gerados por ***Bagging*** e ***Boosting***.

Portanto, fica claro que esta ferramenta pode ter um desempenho muito bom para o problema tratado neste trabalho e, sendo assim, será utilizada com algumas modificações para tentar melhorar ainda mais o seu desempenho.

3.1 Os Algoritmos Genéticos

Algoritmos Genéticos são algoritmos estocásticos de busca, assim como "*simulated annealing*" (abordagem não-populacional) e "*threshold acceptance*", baseados em idéias evolutivas de genética e seleção natural. Eles combinam processos naturais que se baseiam na evolução, especialmente aqueles estabelecidos por Charles Darwin de sobrevivência do mais apto com troca de informação estruturada, porém randômica, para formar um algoritmo de busca com a habilidade de adaptação dos seres vivos [48].

A cada geração, um novo conjunto de aproximações é criado por um processo de seleção de indivíduos de acordo com seu nível de "fitness" (medida de adaptação do indivíduo no ambiente) no domínio do problema e os modificando com utilização de operadores emprestados da genética natural. Esse processo leva à evolução de populações de indivíduos que são mais adaptados ao seu meio ambiente do que os indivíduos dos quais foram criados, como na seleção natural. Ou seja, um indivíduo estará mais apto ao ambiente quando ele corresponder a uma solução mais eficaz para o problema.

Apesar de já estarem sendo desenvolvidos desde 1962, a primeira conquista significativa em Algoritmos Genéticos foi feita em 1975 com a publicação de "*Adaptation in Natural and Artificial System*" por John Holland, seus colegas e seus alunos na Universidade de Michigan. Desde então eles vêm sendo largamente estudados, experimentados e aplicados em vários campos da engenharia, da biologia e das ciências naturais.

Os Algoritmos Genéticos recebem como entrada uma população de indivíduos em representação genotípica (geração inicial) e uma função (fitness) que mede a adequação relativa de cada indivíduo. Os indivíduos são codificados como listas ordenadas (cromossomos) onde cada atributo equivale a um gene e seu valor a um alelo. O tamanho da lista está relacionado ao número de atributos necessários para descrever o indivíduo. À geração inicial, os Algoritmos Genéticos empregam operadores de crossover e mutação para gerar novos indivíduos. Ele usa vários critérios de seleção de modo a escolher os melhores indivíduos para a reprodução. O quão 'bom' é cada indivíduo é determinado pela função-objetivo.

O emprego mais comum dado aos Algoritmos Genéticos é em problemas de otimização, onde o problema a ser resolvido faz o papel do ambiente e cada indivíduo da população é associado a uma solução-candidata. Com o processo de evolução dos Algoritmos Genéticos, se espera que a cada geração obtenham-se soluções-candidatas mais e mais eficazes, contudo sem a garantia de se chegar à solução ótima no final do processo evolutivo. É importante enfatizar que algoritmo genético é uma alternativa de abordagem de problemas classificada como método fraco, concebido para resolver problemas genéricos em mundos não-lineares e não-estacionários e não garantem eficiência total na obtenção da solução. Geralmente garantem uma "boa aproximação" para a solução.

3.1.1 Comparação com outros métodos

Os Algoritmos Genéticos diferem dos métodos tradicionais porque buscam uma população de pontos em paralelo, não um único ponto, e não necessitam de informação de primeira ordem ou mesmo qualquer informação funcional a respeito do problema. Além disso, usam regras probabilísticas, não determinísticas e trabalham com uma codificação do conjunto de parâmetros e não diretamente com eles.

Métodos de otimização clássicos (como o do gradiente) enfatizam o aproveitamento das melhores soluções na busca de aprimoramentos (explotação), enquanto métodos de busca aleatória exploram o espaço de busca (exploração) ignorando o aproveitamento de regiões promissoras. O processo de evolução executado por um Algoritmo Genético faz uma busca em um espaço de soluções potenciais, apresentando um balanço notável entre explotação e exploração, equilíbrio que segundo Michalewicks [31] é essencial para uma boa performance [47].

Comparando com outros algoritmos estocásticos, os AGs mantêm uma população de soluções candidatas enquanto o "*simulated annealing*" processam um único ponto no espaço de busca.

3.1.2 Elementos dos Algoritmos Genéticos

3.1.2.1 Representação da população

O primeiro passo na implementação de um algoritmo genético é a geração de uma população inicial. Para os algoritmos genéticos canônicos, cada membro desta população é representado por uma lista binária (*bitstring*) de comprimento l.

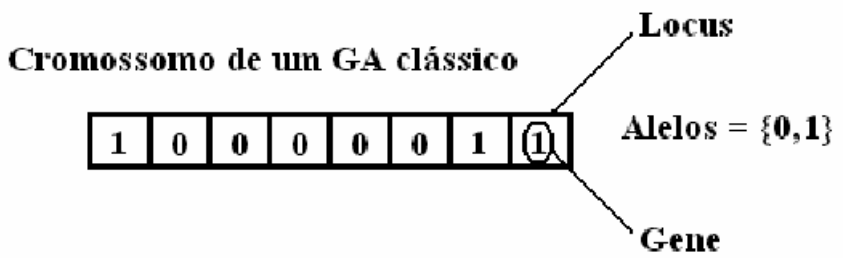


Figura 2.21 - Lista binária (bitstring) de comprimento l.

Cada seqüência é freqüentemente referida como um genótipo ou, alternativamente, como um cromossomo. Os bits são os alelos. A execução de um algoritmo genético pode ser dividida em dois estágios.

Inicia-se com a população corrente. Em seguida, selecionam-se indivíduos para formar uma população intermediária em uma etapa chamada seleção. Logo após, a recombinação (crossover) e a mutação são aplicadas à população intermediária gerando novos indivíduos que serão inseridos na população de acordo com seus respectivos graus de adaptação. É gerada, assim, uma nova população.

O processo que se estende desde o início da geração da população corrente até a geração de uma nova população constitui uma geração na execução do algoritmo genético.

3.1.2.2 A função de fitness

A avaliação dos indivíduos em um algoritmo evolucionário começa com a definição da *função-objetivo,* $f : A_x \to \Re$, sendo que o número de atributos de cada indivíduo é definido pela representação. A função-objetivo mede freqüentemente algum custo a ser minimizado ou lucro a ser maximizado. A definição da função-objetivo irá depender da aplicação.

Propriedades da função-objetivo:

- Deve refletir a medida relevante a ser otimizada;
- Deve exibir algumas regularidades sobre o espaço definido pela representação adotada;
- Deve fornecer informação suficiente para forçar a pressão seletiva do algoritmo evolucionário.

A função de fitness $\varphi : A_x \to \Re_m$ mapeia o valor da função-objetivo em um intervalo não-negativo. A função de fitness é freqüentemente uma composição da função-objetivo:

$$\varphi\left(a_i(t)\right) = g\left(f\left(a_i(t)\right)\right) \tag{2.78}$$

Sendo $A_x = a_i(t)$. Observe que tal mapeamento é necessário quando o objetivo é minimizar a função-objetivo, uma vez que altos valores de fitness correspondem a baixos valores para a função-objetivo.

A noção de avaliação e fitness é algumas vezes usada indistintamente. Entretanto, é importante distinguir entre a avaliação da função e a avaliação de fitness. A avaliação da função, ou avaliação da função-objetivo, fornece uma medida de performance com respeito a um conjunto particular de parâmetros. A função de fitness transforma essa medida de performance em uma alocação de oportunidades reprodutivas.

3.1.2.2.1 Atribuição de fitness baseada em Rank

Na atribuição de fitness baseada em rank, a população é ordenada de acordo com os valores da função objetivo. O valor de fitness atribuído para cada indivíduo depende apenas de sua posição na classificação e não propriamente do valor da função-objetivo.

O problema de estagnação, causado quando a pressão seletiva é muito pequena, e o problema da convergência prematura, causado quando a busca ocorre em uma área muito limitada, podem ser superados. Esta técnica impede que um único indivíduo gere muitos filhos, estabelecendo limites para a reprodução. A atribuição de fitness baseada em rank fornece um simples e efetivo caminho para controlar a pressão seletiva.

Considere N_{ind} o número de indivíduos da população, *Pos* a posição de um indivíduo nesta população (o indivíduo com menor fitness assume $Pos = 1$ e o com maior assume $Pos = N_{ind}$) e seja *SP* o valor da pressão seletiva. O valor de fitness de um indivíduo é calculado como:

3.1.2.2.2 Ranking-Linear

$$Fitness(Pos) = 2 - SP + 2(SP - 1)\frac{Pos - 1}{N_{ind} - 1} \tag{2.79}$$

O Ranking-Linear permite valores de *SP* no intervalo [1.0, 2.0].

3.1.2.2.3 Ranking-Não-Linear

Um novo método para o ranking é o Ranking-Não-Linear. Este tipo de ranking permite uma maior pressão seletiva em comparação ao ranking-linear.

$$Fitness(Pos) = \frac{N_{ind} X^{Pos-1}}{\sum_{i=1}^{N_{ind}} X^{i-1}}, \tag{2.80}$$

onde X é computado como a raiz do polinômio:

$$0 = (SP - 1) X^{N_{ind}-1} + (SP) X^{N_{ind}-2} + ... + (SP) X + SP \tag{2.81}$$

O Ranking-Não-Linear permite valores de *SP* no intervalo [1, $N_{ind} - 2$].

3.1.2.3 Seleção

A seleção é o processo de escolha de indivíduos para a reprodução em um algoritmo evolucionário.

Uma forma popular de seleção é a chamada seleção proporcional. Essa abordagem envolve a geração de indivíduos segundo a proporção dos fitness individuais. O processo de seleção pode ser convenientemente decomposto nas seguintes etapas:

- mapeamento da função-objetivo;
- criação de uma distribuição de probabilidade proporcional ao fitness.

3.1.2.3.1 Seleção por Roleta

Uma vez atribuídos os valores de fitness aos indivíduos, o próximo passo na seleção proporcional é criar uma distribuição de probabilidade tal que a probabilidade de seleção de um dado indivíduo para reprodução é proporcional ao seu valor de fitness, isto é:

$$\Pr_{prop}(i) = \frac{\varphi(i)}{\sum_{i=1}^{\mu} \varphi(i)} \tag{2.82}$$

O algoritmo utilizado na seleção é o famoso "*Roulette Wheel Sampling Algorithm*". O algoritmo pode ser interpretado como uma roleta cuja área é dividida de acordo com os valores relativos de fitness individuais.

N	Cromossomo	Fitness	Graus
1	0001100101010	6.0	180
2	0101001010101	3.0	90
3	1011110100101	1.5	45
4	1010010101001	1.5	45

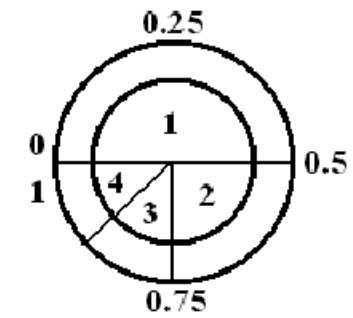


Figura 2.22 - Seleção via Roleta.

Os indivíduos com maiores valores de fitness "detêm" uma maior área na roleta, apresentando uma maior probabilidade de serem selecionados para reprodução, já que a distribuição de probabilidade associada às posições da roleta é uniforme.

3.1.2.3.2 Seleção Estocástica Uniforme

A "*Amostragem Estocástica*" fornece um bias nulo e um espalhamento mínimo. Os indivíduos são mapeados em seguimentos contínuos de uma linha, de tal forma que cada indivíduo corresponde a um segmento de tamanho proporcional ao seu fitness, da mesma forma que no algoritmo da roleta. Aqui, ponteiros igualmente espaçados são colocados sobre a linha. Considere N_{sel} o número de indivíduos a serem selecionados, então a distância entre os ponteiros são $1/N_{sel}$ e a posição do primeiro ponteiro é dada pelo número aleatório gerado na faixa [0, $1/N_{sel}$].

Para selecionar seis indivíduos, a distância entre os ponteiros é de 1/6 = 0.167. A Figura 2.23 ilustra o exemplo.

Amostrando um número aleatório (random number) 0.1 e somando distâncias incrementais 0.167:

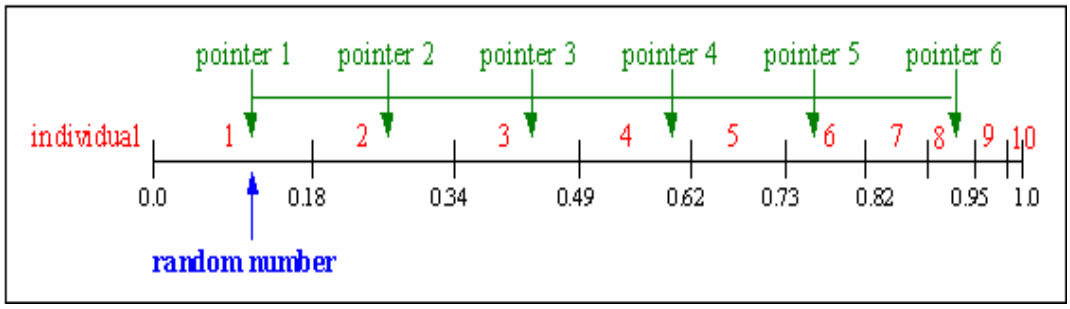


Figura 2.23 - Amostragem Estocástica Universal.

Assim, os indivíduos selecionados são: 1, 2, 3, 4, 6, 8.

3.1.2.3.3 Seleção elitista

A seleção elitista, na verdade, é um subtipo de seleção, pois pode ser usada tanto na seleção por roleta como na seleção estocástica. A seleção elitista faz com que o melhor indivíduo da população, em termos de fitness, sempre fique pareado com os outros indivíduos que foram

selecionados por qualquer uma das técnicas de seleção precedentes. Isto faz com que, na prática, este melhor indivíduo sempre se recombine com os outros indivíduos na etapa de recombinação (crossover).

3.1.2.4 Recombinação

Recombinação é o processo de produzir novos indivíduos pela combinação da informação contida em dois ou mais pais. Dependendo da representação adotada, diferentes métodos podem ser usados.

3.1.2.4.1 Recombinação Discreta – Todas as Representações

A recombinação discreta executa uma troca de valores de variáveis entre os indivíduos. Para cada posição, os dois pais contribuem para a geração da variável do filho na mesma posição correspondente. Isto é realizado por uma escolha randômica de igual probabilidade:

$$Var_i^0 = Var_i^{P1} \cdot a_i + Var_i^{P2}(1-a_i) \;\; i \in (1,2,...,N \text{ var}), \qquad a_i \in \{0,1\} \tag{2.83}$$

Este tipo de recombinação gera vértices do hipercubo definido pelos pais. A figura a seguir ilustra o efeito geométrico da recombinação discreta.

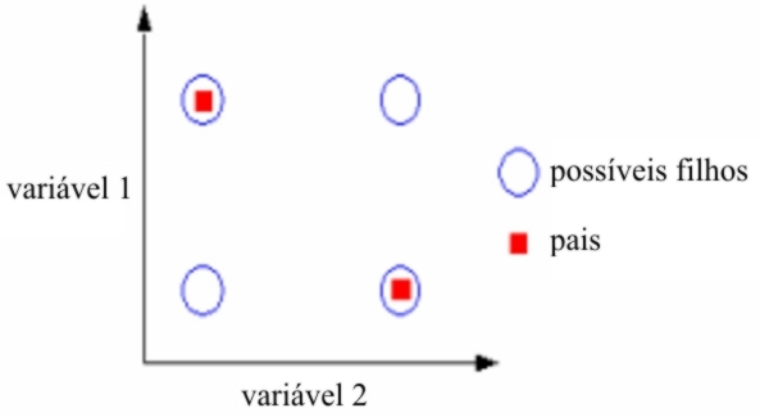


Figura 2.24 - Possíveis posições dos filhos depois da recombinação discreta.

Considere dois indivíduos com três variáveis.

Indivíduo 1: 12 25 5

Indivíduo 2: 123 4 34

Para cada posição dos pais é realizada uma escolha randômica com igual probabilidade.

Amostra 1: 2 2 1

Amostra 2: 1 2 1

Depois da recombinação dois novos indivíduos são gerados:

Filho 1: 123 4 5

Filho 2: 12 4 5

A recombinação discreta pode ser usada para qualquer tipo de variável (binária, inteira, real ou simbólica).

3.1.2.4.2 Crossover (Recombinação Binária)

Nos sistemas biológicos, a recombinação é um processo complexo que ocorre entre pares de cromossomos. Dois cromossomos são fisicamente alinhados ocorrendo ruptura em uma ou mais posições sobre cada cromossomo. Fragmentos de cromossomos homólogos são trocados antes da quebra ser reparada. Isto resulta na recombinação de material genético, que contribui para a variabilidade na população. Nos algoritmos evolucionários, este processo tem sido abstraído utilizando os operadores de crossover que trocam “substrings” entre cromossomos representados por “strings” lineares de símbolos.

No processo de recombinação de variáveis binárias, apenas partes dos indivíduos são trocadas. Dependendo do número de partes, os indivíduos são divididos antes da troca, através da determinação do número de pontos de crossover.

3.1.2.4.3 Crossover de um ponto

No crossover de um ponto, uma posição $k \in [1, N_{var} - 1]$, sendo N_{var} o número de variáveis dos indivíduos, é escolhida randomicamente.

Uma operação básica de crossover foi introduzida por Holland [20]. Trata-se de um procedimento de três passos.

Primeiramente, dois indivíduos são escolhidos aleatoriamente a partir de uma população de pais gerada pelo operador de seleção. Depois, uma ou mais posições da lista são escolhidas como pontos de quebra (pontos de crossover), indicando os segmentos de listas para a troca. Finalmente, segmentos são trocados entre os pais e, então, combinados para produzir dois indivíduos filhos. A proporção de pais submetidos ao crossover durante uma geração é controlada pela probabilidade de crossover $p_c \in [0,1]$, a qual determina a freqüência com que o crossover é invocado. Como exemplo ilustrativo de crossover de um ponto, considere dois indivíduos pais *x* e *y*, um crossover de um ponto é selecionado por escolha randômica. Dois novos indivíduos são gerados pela troca de fragmentos dos pais, a partir deste ponto.

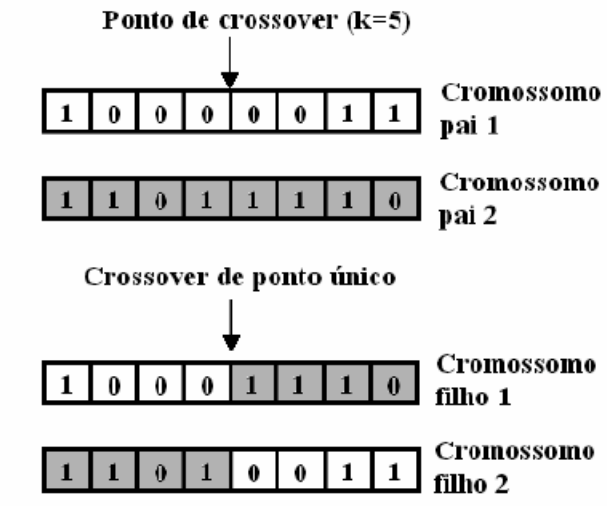


Figura 2.25 - Crossover de um ponto.

O procedimento geral proposto por Holland [20] define uma família de operadores que pode ser descrita como segue. Dado um espaço **I** de indivíduos, um operador de crossover é um mapeamento:

$$r : \mathbf{I} \times \mathbf{I} \xrightarrow{m} \mathbf{I} \times \mathbf{I},\ r(a,b) = r(c,d) \tag{2.84}$$

Sendo $m \in B^l$, B^l o espaço das bitstrings de comprimento *l*, e

$$c_i = \begin{cases} a_i \ se\ m_i = 0 \\ b_i \ se\ m_i = 1 \end{cases},\ d_i = \begin{cases} b_i \ se\ m_i = 0 \\ a_i \ se\ m_i = 1 \end{cases} \tag{2.85}$$

Esta é a descrição formal que caracteriza o crossover como um operador binário.

Em um crossover de dois pontos duas posições são escolhidas randomicamente e as variáveis trocadas entre os indivíduos estão entre estas posições.

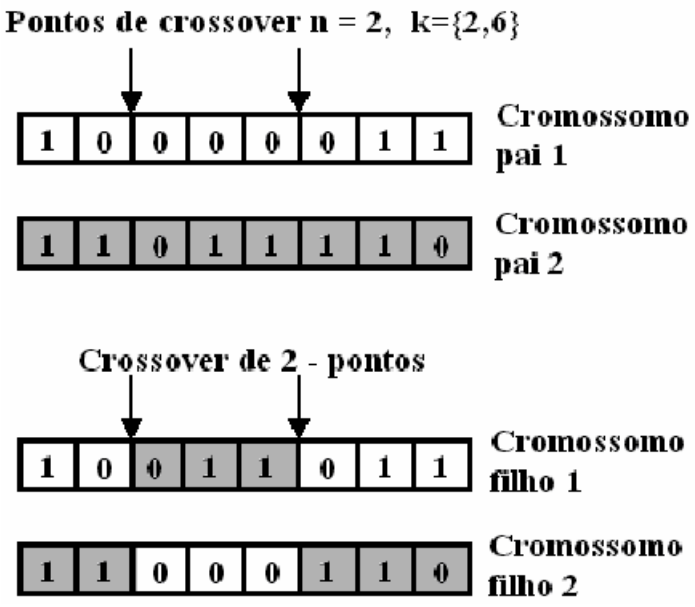


Figura 2.26 - Crossover de 2 pontos.

3.1.2.4.4 Crossover de n pontos

Este operador foi primeiro implementado por De Jong [9] e generaliza o conceito de crossover de ponto único, fazendo do número de pontos de quebra um parâmetro. No entanto, não há consenso sobre as vantagens e desvantagens de usar valores para *n* maiores ou iguais a três ($n \geq 3$).

Para o crossover multi-pontos, posições $k \in [1, N_{var} - 1]$, $i = 1,2,...,m$, são escolhidas randomicamente. As posições determinam os pontos de crossover e as variáveis entre pontos sucessivos são trocadas entre os dois pais, gerando dois novos filhos como ilustra a Figura 2.27.

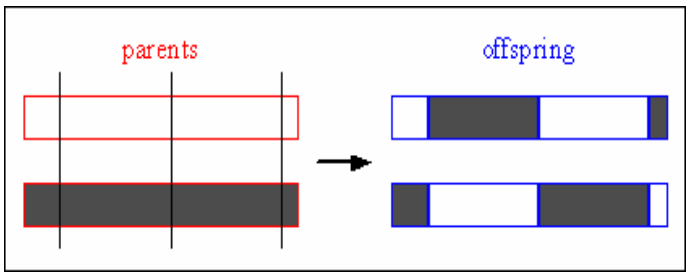


Figura 2.27 - Crossover Multi-ponto.

Considere dois indivíduos com 11 bits cada:

Indivíduo 1: 0 1 1 1 0 0 1 1 0 1 0

Indivíduo 2: 1 0 1 0 1 1 0 0 1 0 1

Escolhendo 3 pontos de crossover:

Pos. cross. (m=3): 2 6 10

Após o crossover, dois novos indivíduos são gerados:

Filho 1: 0 1| 1 0 1 1| 1 1 0 1| 1

Filho 2: 1 0| 1 1 0 0| 0 0 1 0| 0

A idéia por trás do crossover multi-ponto é que partes do cromossomo representando um indivíduo bem adaptado podem não necessariamente estar contida em cadeias binárias

adjacentes. Além disso, a característica combinatória do crossover multi-ponto parece encorajar a exploração do espaço de busca, não favorecendo a convergência prematura, efetuando uma busca mais ampla.

3.1.2.4.5 Crossover Uniforme

Para esse operador o número de pontos de crossover não é fixo. Nesse caso, a decisão de inserir um ponto de quebra é feita independentemente para cada posição da lista.

O crossover de um ponto e o crossover multi-ponto definem pontos de crossover como lugares entre lóci onde um indivíduo poder ser fragmentado. O crossover uniforme generaliza este esquema fazendo cada lócus um potencial ponto de crossover. Uma máscara de crossover de tamanho equivalente ao dos indivíduos é criada para determinar qual dos pais irá fornecer o bit para os filhos. Este método é idêntico à recombinação discreta. Neste caso, as variáveis são bits.

3.1.2.4.6 Crossover de Embaralhamento (Shuffle)

O crossover de embaralhamento é similar ao crossover uniforme. Um único ponto de crossover é selecionado. Mas antes, as variáveis são trocadas de posição no cromossomo. Elas são randomicamente embaralhadas em ambos os pais. Em seguida, efetua-se o crossover de um ponto. Logo após, a variáveis nos filhos são então desembaralhadas. Isto remove a polarização posicional.

3.1.2.4.7 Crossover com Substituição Reduzida (Reduced surrogate)

Este operador força o crossover a produzir sempre novos indivíduos possíveis. Este é implementado restringindo a localização de pontos de crossover de tal maneira que pontos de quebra acontecem somente onde os valores dos genes são diferentes.

3.1.2.5 Mutação

Na evolução natural, a mutação é um processo randômico em que uma parte do gene é alterada de alguma maneira para produzir uma nova estrutura genética. Em algoritmos genéticos, a mutação é aplicada randomicamente com baixa probabilidade, usualmente com uma variação entre 0,001 e 0,01 e modifica elementos cromossômicos. Normalmente é considerada como um operador de suporte. O papel da mutação é freqüentemente visto como um auxiliar na garantia de que a probabilidade de busca de qualquer seqüência fornecida nunca será zero, e auxiliando, desta forma, na recuperação o material genético bom perdido durante a ação de seleção ou crossover.

3.1.2.5.1 Mutação Binária

O efeito da mutação em uma seqüência binária é mostrado na figura abaixo, para um cromossomo de 10 bits representando um valor real decodificado em um intervalo [0,10] usando codificação *Standard* e *Gray* e um código de mutação no ponto 3 da seqüência binária. Aqui a mutação binária troca o valor do bit selecionado.

Como a mutação é geralmente aplicada uniformemente em toda a população da seqüência, é possível que a seqüência sofra mutações em mais de um ponto.

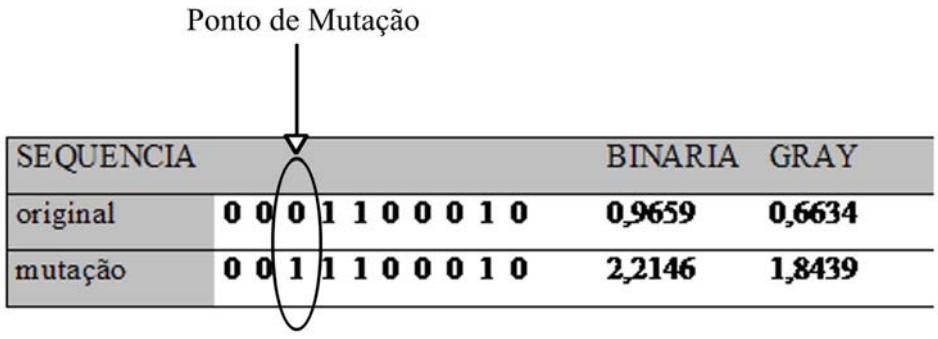


Figura 2.28 - Usando codificação Standard e Gray.

3.1.2.6 Reinserção

Depois da geração dos filhos, eles devem ser inseridos na população. Este processo é especialmente importante se o número de filhos gerados é menor que o tamanho da população. Outro caso é quando nem todos os filhos podem ser usados em cada geração ou, ainda, no caso em que se gera mais filhos do que o necessário. A reinserção é um processo que consiste em determinar quais indivíduos filhos devem ser inseridos na nova população e quais indivíduos devem ser substituídos. O algoritmo de seleção determina o esquema de reinserção:

- Reinserção global para toda a população baseada em algoritmo de seleção (roleta, amostragem estocástica universal, etc.);
- Reinserção local para seleção local.

Alguns estudos sobre técnicas de reinserção apresentados por vários autores mostram que a reinserção baseada em fitness não apresenta melhor desempenho. Desempenho não muito inferior foi obtido pela reinserção dos piores indivíduos. A técnica com melhor desempenho foi a reinserção dos filhos no lugar dos indivíduos mais velhos da população, ou seja, aqueles que permaneceram na população por várias gerações.

3.1.2.7 Critério de parada

O critério de parada do algoritmo comumente utilizado é o de estipular um número máximo de gerações e então testar a aceitabilidade da população e, no caso da resposta ser negativa, o algoritmo é reiniciado. Outro possível critério é o de fixar um determinado nível de diversidade da população, desta forma, os indivíduos da população seriam suficientemente parecidos e, conseqüentemente, a continuidade do processo pouco provavelmente gerará um indivíduo tão melhor que compense o esforço computacional.

Capítulo 3: Estudo de Caso

1 Estrutura do Ensemble Utilizado

O Ensemble utilizado neste trabalho tem características parecidas com as do ***GASEN*** [58]. Ele utiliza Redes Neurais e Neurais Nebulosas com arquiteturas diferentes de tal forma que os parâmetros mais básicos da arquitetura destas redes são também evoluídos com a utilização de algoritmos genéticos, como, por exemplo, o número de neurônios na camada intermediária de uma rede MLP. Para que fique claro qual foi a estrutura do ensemble utilizado, a Figura 3.1 ilustra este cenário.

Na Figura 3.1, de baixo para cima, é possível ver que os dados de entrada alimentam as entradas dos previsores básicos. Em seguida, são representadas as redes individuais, no caso, composta por três tipos de RNAs e um tipo de Rede Neural Nebulosa. Cada tipo de rede é composta, por sua vez, por diferentes modelos cujos conjuntos de parâmetros são resultados de uma evolução realizada com Algoritmos Genéticos. Por exemplo, no primeiro bloco (RNAs MLP) foram utilizadas vinte diferentes redes MLP, sendo que o que as diferencia é o conjunto de parâmetros de cada uma, como o tipo de função de ativação, número de neurônios na camada intermediária, entre outros. Finalmente, o ensemble é obtido pela combinação das saídas destes modelos.

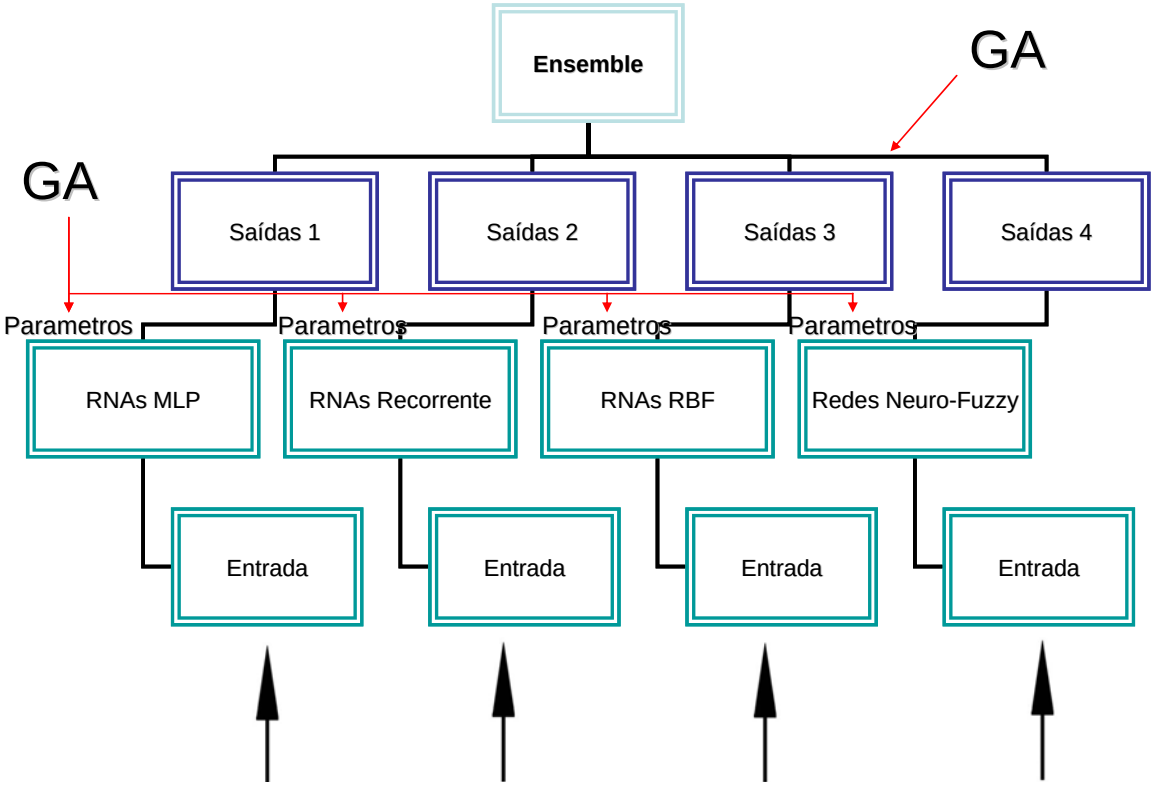


Figura 3.1 - Estrutura do Ensemble utilizado.

Como se vê no esquema da Figura 3.1, fica evidente que na primeira etapa, onde os previsores individuais serão criados, quatro tipos diferentes de modelos de previsão têm os seus parâmetros evoluídos em paralelo. Assim, após a evolução dos parâmetros, haverá quatro populações de previsores com características próprias, como já foram descritas anteriormente para cada um destes quatro modelos. Portanto, se espera um bom resultado do ensemble final uma vez que os modelos têm características, teoricamente, complementares, como já foi visto.

Na segunda etapa, serão feitas *N* previsões com os *N* previsores evoluídos na primeira etapa nos quatro modelos utilizados. No entanto, para melhorar o desempenho do ensemble, é empregado um algoritmo genético que selecionará um subconjunto de previsores e somente as previsões deste subconjunto irão compor o ensemble. A composição dos resultados deste subconjunto pode ser feita de diversas maneiras. Neste trabalho, foram feitos vários testes para

compor os resultados, o que se revelou mais eficiente e estável foi usando a média aritmética. Portanto, os resultados apresentados aqui para o ensemble utilizam a combinação dos previsores via média aritmética.

2 Dados utilizados

Como já foi visto, neste trabalho foi considerada uma série temporal de dados de carga de uma companhia de energia elétrica do estado de São Paulo. Foi utilizado um histórico de 100 semanas com início no dia 19 de janeiro de 1998 e fim no dia 19 de dezembro de 1999. Os dados do histórico se encontravam em base horária.

Assim sendo, foi utilizada uma metodologia para previsão desta série temporal que já foi proposta por Kadowaki, Ohishi, Soares e Ballini [24]. Esta metodologia procede da seguinte maneira:

- Com os dados de carga em base horária, é calculada uma carga média diária;
- É construída uma série temporal de cargas horárias médias diárias;
- Os dados de carga diária são separados por dia da semana. Assim serão construídas sete séries temporais, uma para cada dia de semana;
- Com esta série temporal de carga diária, é feita a previsão de valores futuros;
- A partir dos valores previstos de carga média diária, é ajustada uma curva de carga horária de acordo com o perfil do dia de semana correspondente ao dia previsto (esta etapa não será desenvolvida, pois o foco deste trabalho é avaliar o desempenho na previsão de séries temporais quando são utilizados “ensembles”).

Os dados de carga média por dia são plotados na Figura 3.2. No modelo proposto, os dados de entrada correspondem às séries temporais das segundas feiras, terças e assim por diante. Desta

forma, os dados que alimentam as entradas na fase de treinamento e são utilizados para avaliar o desempenho do modelo proposto correspondem às séries temporais ilustradas na Figura 3.3.

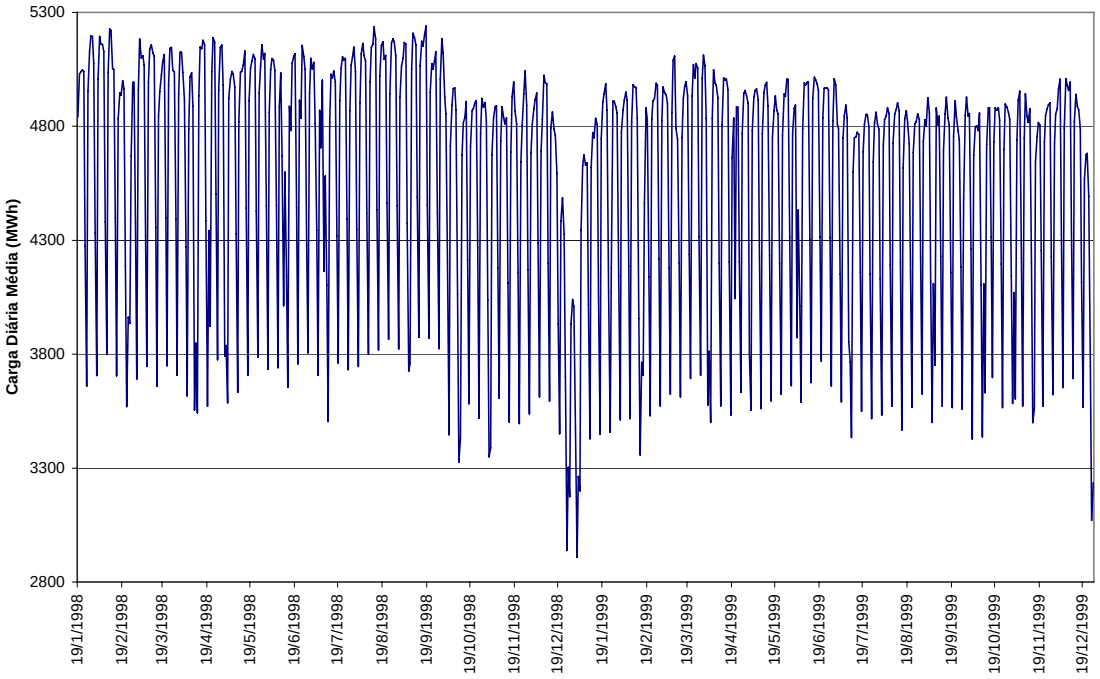


Figura 3.2 - Série Temporal em base diária.

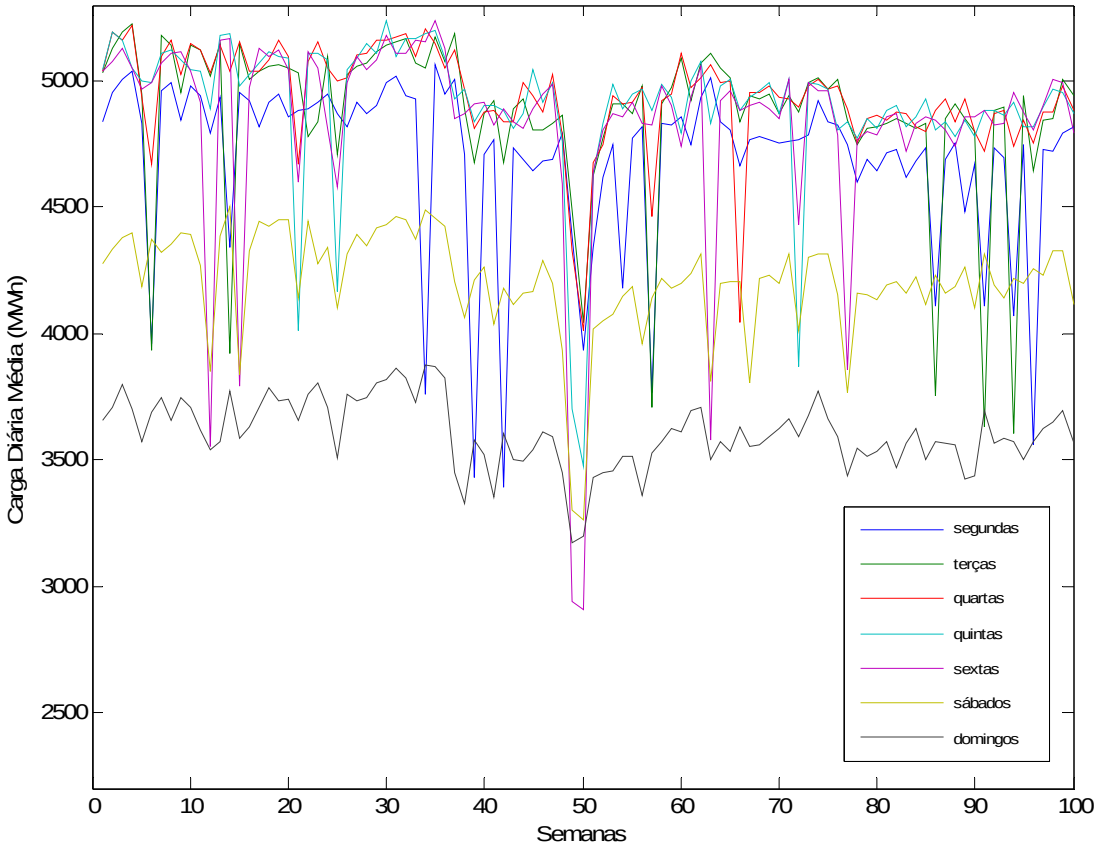


Figura 3.3 - Séries Temporais utilizadas para previsão.

A série temporal da Figura 3.3 foi dividida basicamente em dois trechos. O primeiro é o histórico de treinamento das redes, que corresponde ao trecho que vai do dia 19 de janeiro de 1998 até o dia 17 de janeiro de 1999 (semana 52). O segundo é o histórico que irá servir para avaliar a evolução dos previsores (semanas 53 a 68), definir a composição do ensemble (semanas 69 a 84) e realizar as avaliações de previsão com o ensemble definitivamente composto (semanas 85 a 100).

2.1 Tratamento dos dados

A análise das séries temporais da Figura 3.3 deixa claro que, principalmente as séries dos dias úteis têm variações muito grandes. Se utilizadas as séries sem nenhum tratamento, estas variações farão com que o treinamento dos previsores individuais fique muito complicado, pois estes dados atípicos podem fazer com que o algoritmo de treinamento se perca e que o modelo fique mal condicionado.

Assim sendo, foi implementado um algoritmo que detecta estes "outliers" e os substitui por valores médios dos três dias anteriores. A classificação dos "outliers" é feita quando um valor de carga é superior a 10% em relação à média dos mesmos três dias anteriores. Como resultado, os dados utilizados neste trabalho, após o tratamento, assumem a seguinte forma:

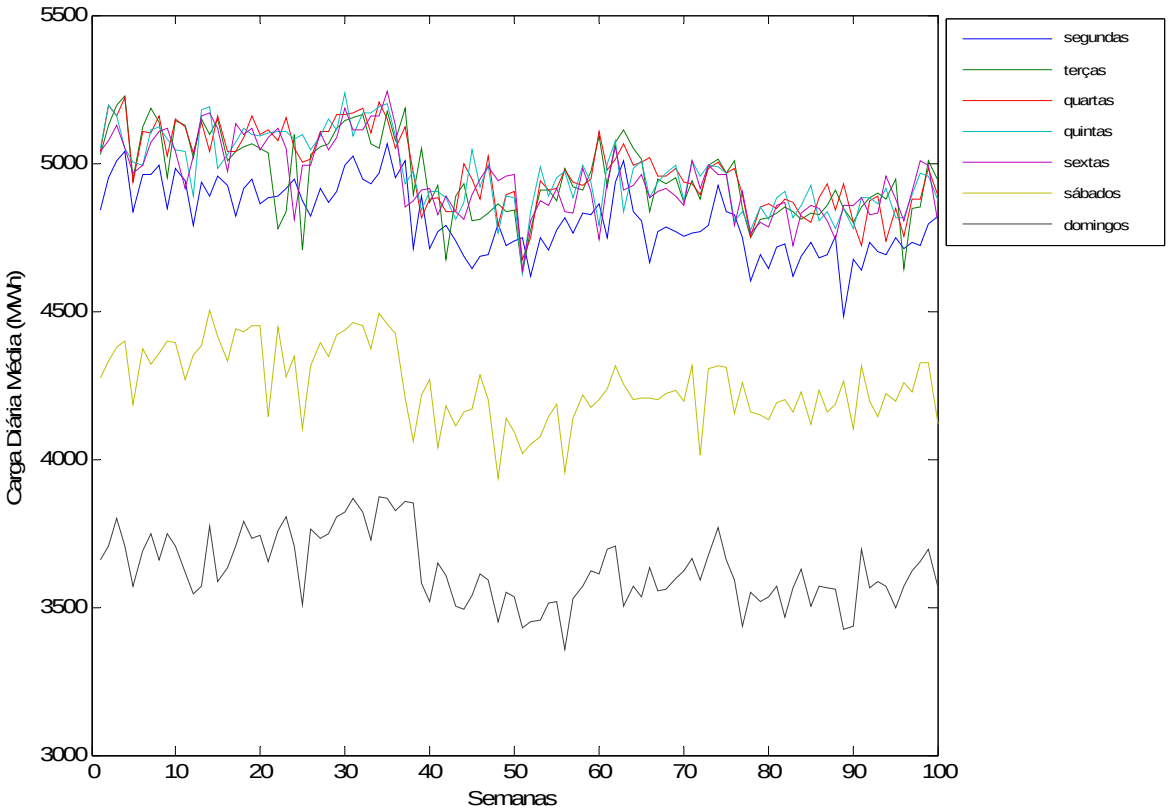


Figura 3.4 - Séries Temporais tratadas para previsão.

3 Implementação do modelo proposto

A implementação do Ensemble de previsores selecionados por GA segue a mesma estrutura daquela apresentada pelo diagrama da Figura 3.1.

Todos os previsores utilizados foram retirados dos Toolboxes de Redes Neurais e Lógica Nebulosa do MATLAB, e o algoritmo genético utilizado foi o Toolbox ***AGBIN*** [43] também para MATLAB.

Para que a implementação do ensemble utilizado neste trabalho seja melhor compreendido, será feita uma explicação passo a passo da implementação, partindo dos dados de entrada do ensemble em direção à solução final do ensemble, ou seja, seguindo o diagrama da Figura 3.1 de baixo para cima.

Sendo assim, como a primeira etapa, ou seja, a etapa da concepção dos dados a serem modelados e estudados, já foi exposta nos tópicos anteriores, agora serão detalhadas as próximas etapas e os resultados obtidos.

3.1 Primeira Fase: Criação dos previsores

Neste tópico, será descrito como foi feita a implementação do Ensemble utilizado para fazer a previsão de curto prazo um passo à frente. Como já foi visto, a primeira fase de construção do ensemble se constitui na criação de uma população de bons previsores.

3.1.1 Características dos previsores individuais

Antes de iniciar qualquer etapa relativa ao desenvolvimento do ensemble, é preciso assegurar que cada um dos previsores individuais terão condições de fazer boas previsões sozinhos. Para isto, foi preciso estudar o problema e determinar uma arquitetura básica em termos

de número de entradas, características das entradas, conjuntos de treinamento e abordagem de previsão.

Inicialmente, foi decidido que a melhor abordagem de previsão seria fazer previsões individuais para cada dia da semana. Assim sendo, em cada passo de treinamento das redes, a saída desejada corresponderia a um determinado dia de semana.

Tendo em vista que cada modelo individual seria treinado para cada dia da semana, se tentou utilizar diferentes configurações de entradas. Um dos testes realizados foi a utilização de uma só entrada defasada de uma semana, ou seja, entra, por exemplo um sábado e como saída desejada se espera o sábado da próxima semana. No entanto, esta arquitetura se revelou muito simples para que os modelos individuais pudessem ser ajustados satisfatoriamente.

A arquitetura de entradas de dados que se revelou mais eficiente foi a utilização de duas entradas de tal forma que, quando se pretende prever, por exemplo, o domingo da semana t são usados como entradas o sábado da semana anterior em uma entrada e o domingo da semana anterior em outra entrada. A Figura 3.5 ilustra esta situação:

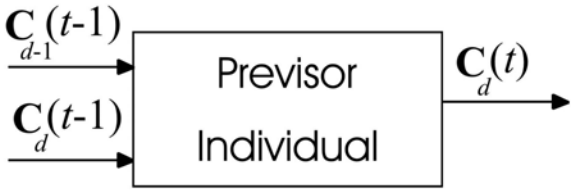


Figura 3.5 - Arquitetura de dados utilizada nos previsores individuais.

De acordo com a Figura 3.5, **C** é o vetor de cargas horárias médias por dia, $d = 1,2,...7$ representa os dias da semana, onde 1 é segunda feira, 2 é terça e assim por diante e $t = 1,2,...,100$ designa a semana utilizada do histórico.

3.1.2 Resultados nas primeiras avaliações dos previsores individuais

Vale ressaltar que, para evitar o mau condicionamento numérico ao utilizar os dados nas redes implementadas, os valores de cargas das séries temporais foram normalizados entre zero e um. Esta transformação foi feita de acordo com a equação (3.1).

$$\tilde{C}_d(t) = \frac{C_d(t)}{\max(\mathbf{C}_d)},\ d = 1,2,...,7,\ t = 1,2,...,100,\ \mathbf{C} \in \Re^{100} \tag{3.1}$$

De acordo com a equação (3.1), $\tilde{\mathbf{C}}_d \in \Re^{100}$ são as *d* séries temporais normalizadas entre zero e um, e $\mathbf{C}_d \in \Re^{100}$ as *d* séries originais. Percebe-se claramente que com a equação (3.1) se obtém as séries normalizadas entre zero e um e também pode ser usada para recuperar as séries normalizadas para os valores originais.

Com as séries normalizadas, foram feitos os primeiros testes nas redes individuais. Inicialmente, foi construída uma rede MLP com os seguintes parâmetros:

- Função de ativação na camada intermediária: Log-sigmoidal;
- Função de ativação na camada de saída: linear;
- Número de neurônios na camada intermediária: 8;
- Número de padrões (semanas) de treinamento: 20;
- Algoritmo de treinamento: método do gradiente com momento e taxa de aprendizado adaptativa.

Na Figura 3.6, se vê um exemplo de previsões feito para um período de 48 semanas para as quartas-feiras.

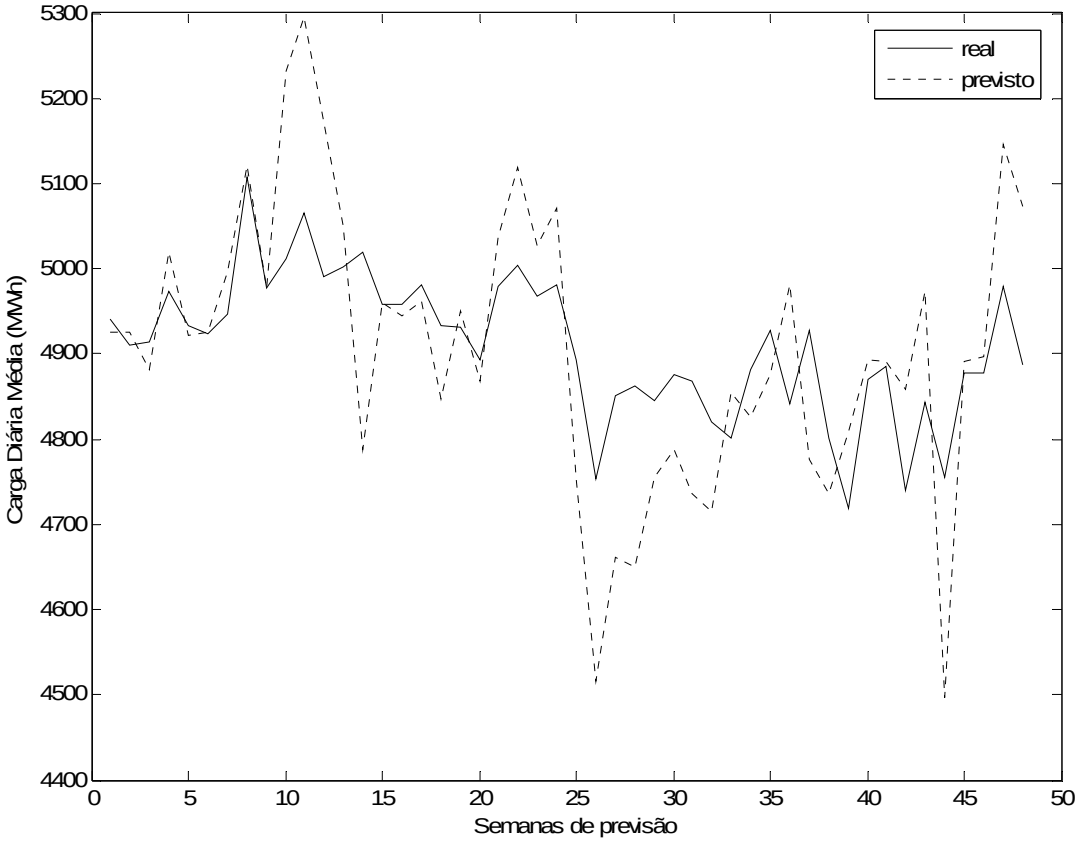


Figura 3.6 - Exemplo de previsão de uma MLP nas quartas feiras.

Neste teste com esta rede MLP, o erro médio percentual absoluto acumulado (somatório dos erros percentuais absolutos dia-a-dia) nos 48 meses de teste de previsão e nos sete dias de semana foi 1,94%.

Foi feito o mesmo teste com a rede neural recorrente. Os parâmetros utilizados foram os mesmos da rede MLP. Um exemplo de previsão para as quartas feiras é ilustrado na Figura 3.7.

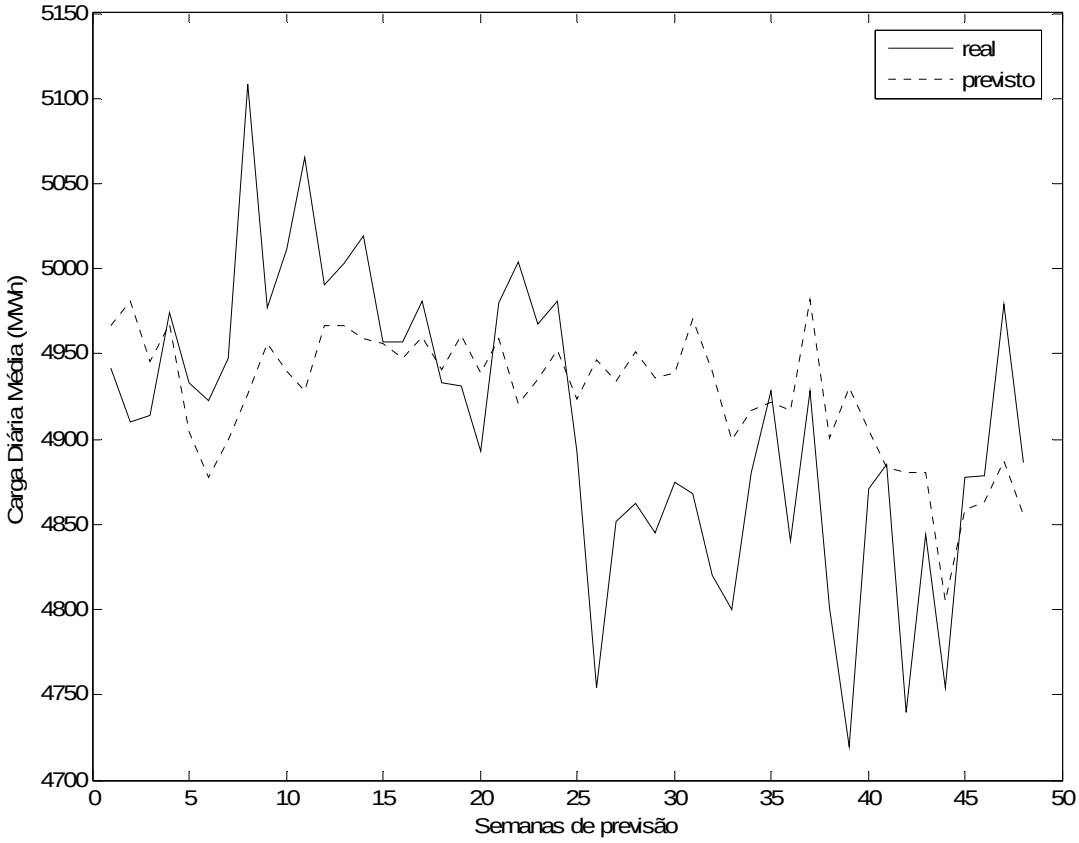


Figura 3.7 - Exemplo de previsão para uma rede recorrente nas quartas feiras.

Como se percebe, a rede neural recorrente erra mais do que a rede MLP. No entanto, a sua aproximação foi mais suave e, em muitas semanas, os erros foram opostos aos erros da rede MLP, motivando desde já a combinação dos dois resultados.

Com a utilização do modelo neural recorrente, o erro médio percentual absoluto acumulado nos 48 meses de teste de previsão e nos sete dias de semana foi 1,65%.

Foi feito também o teste com o modelo de rede neural de base radial. O resultado pode ser visto na Figura 3.8. Os parâmetros utilizados neste teste com a rede RBF foram:

- Valor do espalhamento da função de base radial: 10;
- Tipo de treinamento: adição de neurônios na camada intermediária.

- Padrões de treinamento: 20.

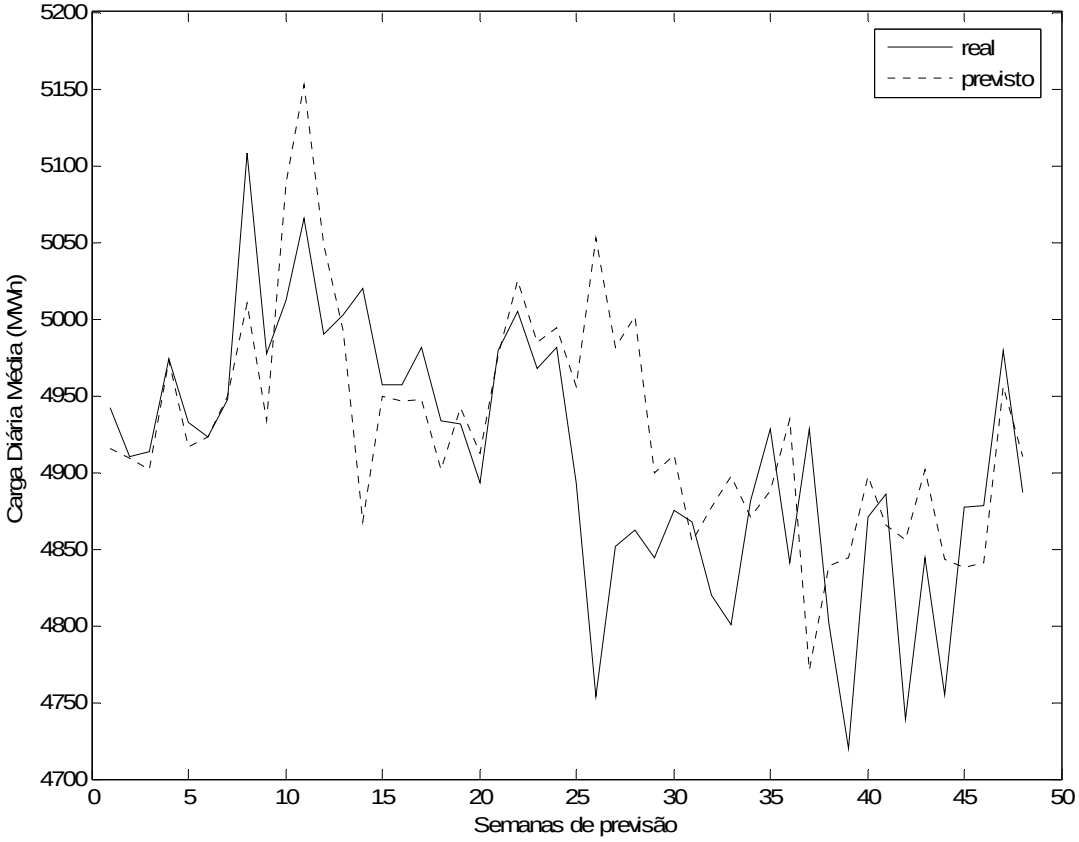


Figura 3.8 - Exemplo de previsão para uma rede RBF nas quartas feiras.

Quando foi utilizada a rede RBF, o erro médio percentual absoluto acumulado nos 48 meses de teste de previsão e nos sete dias de semana foi 1,29%.

Finalmente, foi testado o modelo neural nebuloso com a arquitetura ANFIS. O resultado para as quartas-feiras pode ser visto na Figura 3.9. O modelo ANFIS foi parametrizado da seguinte maneira:

- Tipo de função de pertinência: Função tipo sino generalizada;
- Espalhamento da função de pertinência: 10;
- Número de funções de pertinência por entrada: 10;

- Número de padrões de treinamento: 10;
- Número de épocas de treinamento: 500.

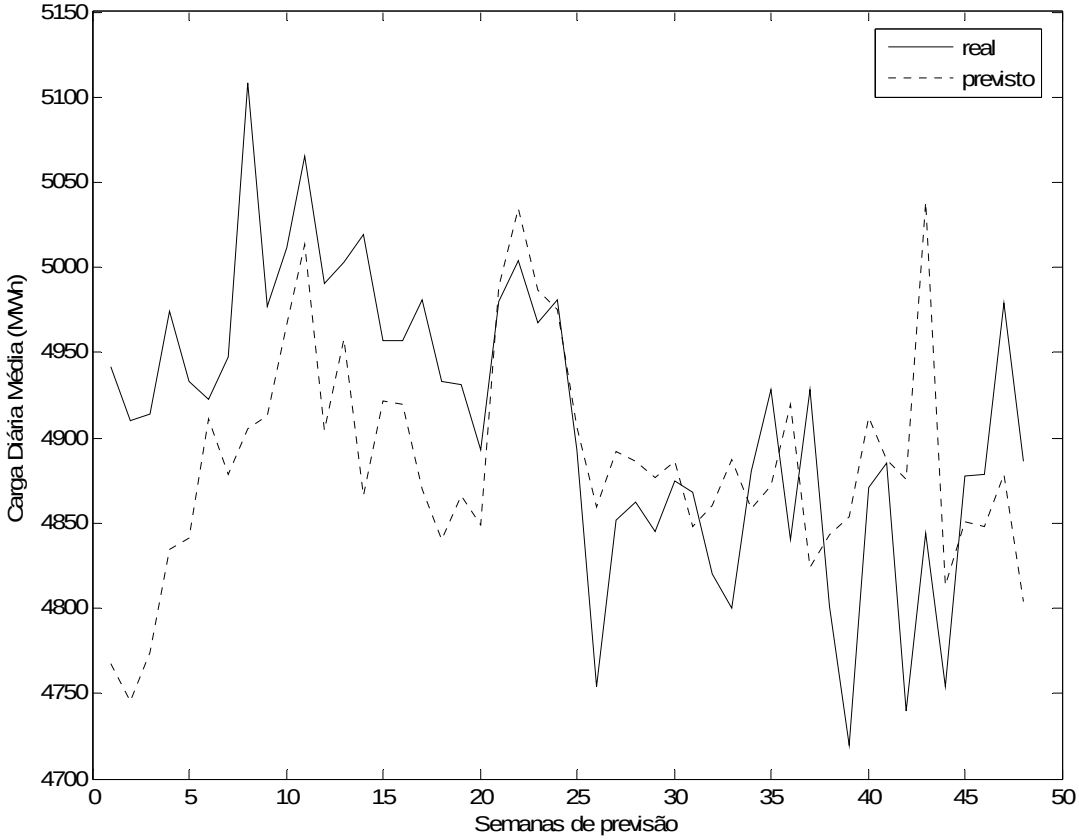


Figura 3.9 - Exemplo de previsão para o modelo ANFIS nas quartas feiras.

Ao utilizar a rede neural nebulosa, o erro médio percentual absoluto acumulado nos 48 meses de teste de previsão e nos sete dias de semana foi 2,45%.

3.1.3 Evolução dos parâmetros dos previsores

Após criada a configuração básica dos previsores, foi feita a evolução dos parâmetros dos previsores básicos para se chegar a uma população de bons previsores para resolver o problema de prever valores futuros da série temporal estudada neste trabalho.

O critério utilizado para avaliar os indivíduos foi o erro percentual absoluto médio no período de avaliação acumulado para as previsões dos sete dias da semana:

$$E^i = \sum_{d=1}^{7}\left(\sum_{t=53}^{68}\left\|\tilde{C}_{d,t}^{i}\right\|\right),\ i = 1,2,...,tamanho\left(População\right) \tag{3.2}$$

Contudo, como o algoritmo genético atua maximizando os valores da função objetivo (os valores de fitness) e o objetivo é minimizar o erro, foi necessária uma transformação dos valores de erros para valores de fitness de tal forma que quando se maximiza o fitness, se minimiza o erro. Assim:

$$fitness\left(i\right) = \frac{1}{E^i},\ i = 1,2,...,tamanho(População) \tag{3.3}$$

Portanto, de acordo com a equação (3.3), quanto menor o erro, maior o valor de fitness.

Desta forma, foram testadas diversas parametrizações para o algoritmo genético utilizado para evoluir as populações de previsores. A configuração que melhor se ajustou ao problema foi a seguinte:

- Número de indivíduos: 50;
- Porcentagem de indivíduos selecionados para a próxima geração: 40%;
- Taxa de Crossover: 80%;
- Tipo de recombinação: crossover de dois pontos (ponto duplo) com substituição reduzida (reduced surrogate);
- Probabilidade de mutação: 10%;
- Tipo de seleção: roleta;
- Subtipo de seleção: normal;
- Tipo de reinserção: fitness.

Foi feita a evolução dos parâmetros por apenas 50 gerações, pois este processo é muito caro computacionalmente, uma vez que, ao avaliar todo novo indivíduo na população é necessário fazer 16 previsões para cada um dos 7 dias de semana, ou seja, 112 previsões.

Uma vez decidido usar o Toolbox AGBIN [43] para evolução dos parâmetros dos previsores, foi decidido também utilizar a codificação binária para representação do cromossomo de cada um dos indivíduos (conjunto de parâmetros) de cada um dos quatro previsores básicos (veja Figura 3.).

Para fazer o mapeamento de valores binários dos cromossomos em parâmetros dos previsores, foi utilizado o ***Código de Gray*** por ser considerado o mais adequado para utilização em algoritmos genéticos com codificação binária. A seguir, são apresentados os resultados para os quatro previsores básicos.

3.1.3.1 Codificação para Evolução das Redes Neurais MLP e Recorrentes

Como as Redes Neurais MLP e Recorrentes são muito parecidas, foi utilizada a mesma codificação para a evolução dos parâmetros destas redes.

Os únicos parâmetros que foram pré-estabelecidos são o número de camadas intermediárias igual a um e o número de épocas de treinamento igual a 200. Todos os outros parâmetros foram evoluídos com Algoritmos Genéticos. Nas tabelas a seguir, podemos ver uma listagem destes parâmetros:

Tabela 3.1 - Algoritmos de Treinamento: Redes MLP e Recorrente

Algoritmos de Treinamento (16 opções: 4bits)	
Função do MATLAB	**Descrição:**
'trainb'	Treinamento em batelada com regras de apredizado para pesos e bias.
'trainbfg'	Backpropagation: BFGS quasi-Newton
'trainbr'	Regularização Bayesiana.
'trainc'	Atualização incremental em ordem cíclica.
'traincgb'	Backpropagation: gradiente conjugado Powell-Beale.
'traincgf'	Backpropagation: gradiente conjugado Fletcher-Powell.
'traincgp'	Backpropagation: gradiente conjugado Polak-Ribiere.
'traingd'	Backpropagation: método do gradiente.
'traingda'	Backpropagation: método do gradiente com taxa de aprendizado adaptativa.
'traingdm'	Backpropagation: método do gradiente com momento.
'traingdx'	Backpropagation: método do gradiente com momento e taxa de aprendizado adapatativa.
'trainoss'	Backpropagation: Levenberg-Marquardt.
'trainr'	Backpropagation: Secante de uma passo.
'trainrp'	Atualização incremental aleatória.
'trains'	Backpropagation: Residual (Rprop).
'trainscg'	Atualização incremental em ordem seqüencial.

Tabela 3.2 - Funções de Ativação e números de neurônios: Redes MLP e Recorrente.

Funções de Ativação da Camada Intermediária e de Saída (11x2 opções: 4x2bits=8bits)	
Função do MATLAB	**Descrição:**
'compet'	"Competitiva"
'hardlim'	Função Degrau
'hardlims'	Função Degrau Simétrica
'logsig'	Log-sigmoidal
'poslin'	Linear positiva
'purelin'	Linear simples
'satlin'	Linear saturada
'satlins'	Linear saturada simétrica
'softmax'	Softmax
'tansig'	Tangente hiperbólica sigmoidal
'tribas'	Base triangular
Número de Neurônios na camada intermediária: (16 opções: 4bits)	
De 2 a 17	

As tabelas acima mostram que, para representar este conjunto de parâmetros em um cromossomo de valores binários, são necessários 16 bits, pois são necessários 4 bits para codificar o algoritmo de treinamento utilizado, mais 4 bits para a função de ativação na camada intermediária, mais 4 bits para a função de ativação na camada de saída e, finalmente, mais 4 bits para o número de neurônios na camada intermediária.

Vale salientar que a decisão de utilizar valores de números de neurônios na camada intermediária que variam entre 2 e 17 se deve ao fato de que com a utilização de menos de dois neurônios na camada intermediária a rede ficaria com pouca capacidade de aprendizado e a utilização de mais de 17 neurônios na camada intermediária pode fazer com que o modelo fique excessivamente ajustado reduzindo sua capacidade de generalização.

A representação dos genes do cromossomo pode ser vista na Figura 3.10.

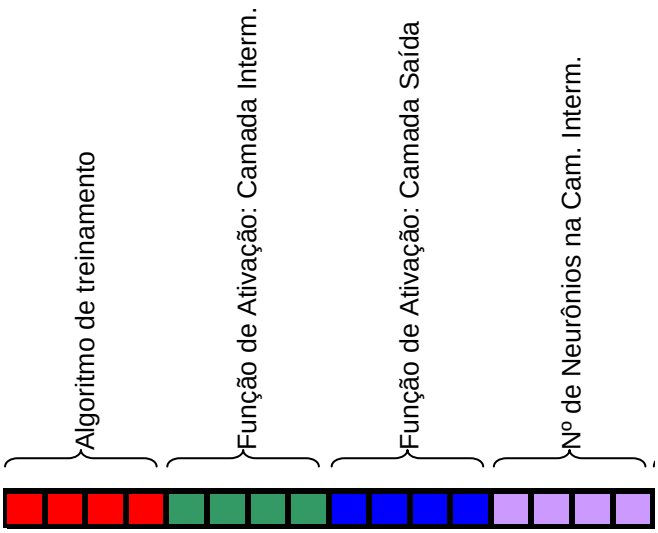


Figura 3.10 - Configuração do Cromossomo: Parâmetros das Redes MLP e Recorrente

3.1.3.2 Evolução das Redes Neurais MLP

Com a codificação genética adotada para evoluir os parâmetros das redes MLP, foi feita a evolução dos parâmetros por 50 gerações. A Figura 3.11 mostra como foi a evolução da população de redes MLP. Neste tipo de gráfico, cada uma das linhas representa respectivamente os valores de fitness do melhor indivíduo e da média dos valores de fitness de toda a população em cada uma das gerações.

Durante a evolução, foram avaliados 922 tipos diferentes de Redes MLP. O processo de evolução demorou 196 minutos para terminar.

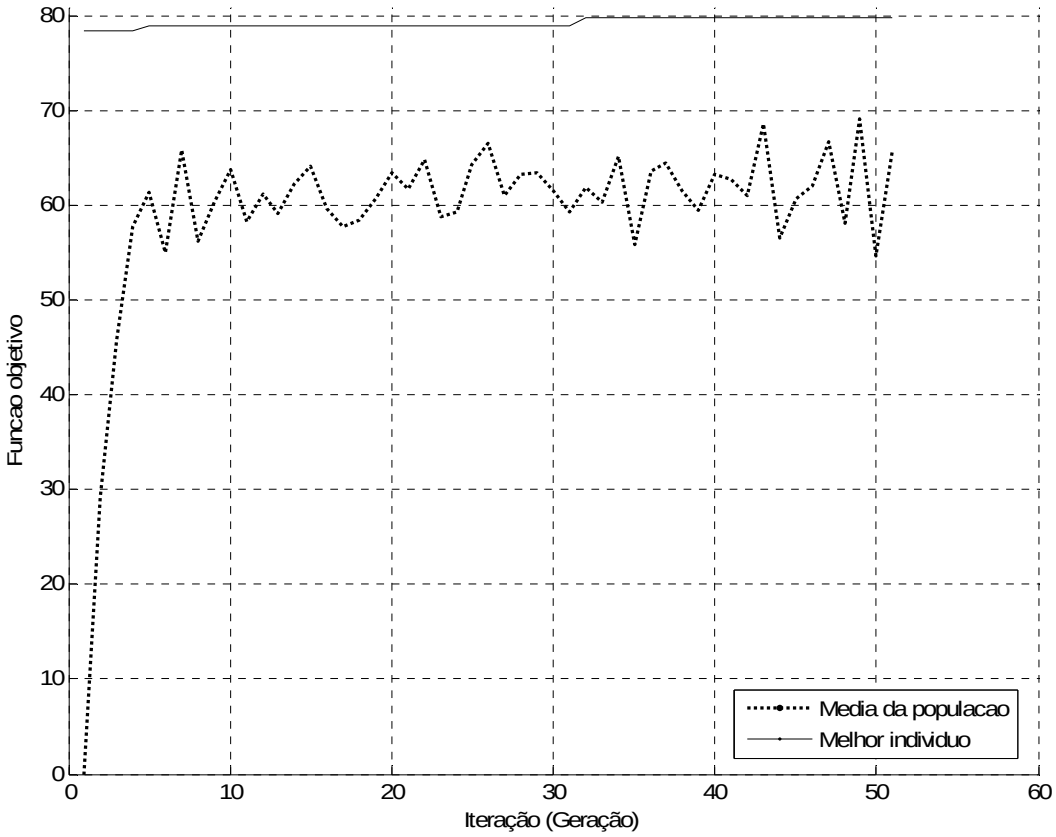


Figura 3.11 - Evolução da População para as Redes MLP.

Após esta evolução dos parâmetros das Redes MLP, foram selecionadas as 20 melhores para comporem o ensemble. A Tabela 3.3 mostras as características destas 20 melhores redes MLP. Nela é possível perceber que os indivíduos gerados têm baixos erros de previsão, o que os tornam competitivos quando forem futuramente combinados.

Tabela 3.3 - As 20 melhores Redes MLP.

					Fcn Ativação Camadas		Neurônios na Camada
Indivíduo	**Genótipo**	**fitness**	**erro médio**	**treinamento**	**Intermediária**	**Saída**	**Intermediária**
1	0 1 1 1 1 1 0 0 1 1 0 0 1 1 0 1	78,54787	1,27311%	'traincgb'	'satlins'	'satlins'	9
2	0 1 1 1 1 1 0 0 1 1 0 0 1 1 0 1	78,54787	1,27311%	'traincgb'	'satlins'	'satlins'	9
3	0 1 1 1 1 1 0 0 1 1 0 0 1 1 0 1	78,54787	1,27311%	'traincgb'	'satlins'	'satlins'	9
4	0 1 1 1 1 1 0 0 1 1 0 0 1 1 0 1	78,54787	1,27311%	'traincgb'	'satlins'	'satlins'	9
5	0 1 0 1 1 1 0 1 0 1 0 0 1 1 1 1	78,61295	1,27206%	'traincgf'	'softmax'	'satlin'	10
6	0 1 0 1 1 1 0 1 0 1 0 0 1 1 1 1	78,61295	1,27206%	'traincgf'	'softmax'	'satlin'	10
7	0 1 0 1 1 1 0 1 0 1 0 0 1 1 1 1	78,61295	1,27206%	'traincgf'	'softmax'	'satlin'	10
8	0 1 0 1 1 1 0 1 0 1 0 0 1 1 1 1	78,61295	1,27206%	'traincgf'	'softmax'	'satlin'	10
9	0 1 1 1 0 1 1 1 0 1 0 0 1 0 0 0	78,79887	1,26905%	'traincgb'	'poslin'	'satlin'	15
10	0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0	78,85847	1,26809%	'trainscg'	'softmax'	'satlin'	15
11	0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 1	78,91713	1,26715%	'trainscg'	'softmax'	'satlin'	14
12	0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 1	78,91713	1,26715%	'trainscg'	'softmax'	'satlin'	14
13	0 0 0 0 1 1 0 1 1 1 0 0 1 0 0 1	78,9399	1,26679%	'trainscg'	'softmax'	'satlins'	14
14	0 0 0 0 1 1 0 1 1 1 0 0 1 0 0 1	78,9399	1,26679%	'trainscg'	'softmax'	'satlins'	14
15	0 0 0 0 1 1 0 1 1 1 0 0 1 0 0 1	78,9399	1,26679%	'trainscg'	'softmax'	'satlins'	14
16	0 1 1 1 0 1 0 1 0 1 0 1 1 1 0 0	79,02085	1,26549%	'traincgb'	'purelin'	'purelin'	8
17	0 1 1 1 0 1 0 1 0 1 0 1 1 1 0 0	79,02085	1,26549%	'traincgb'	'purelin'	'purelin'	8
18	0 0 0 1 1 1 1 1 0 1 0 0 1 0 0 1	79,65631	1,25539%	'trainb'	'tansig'	'satlin'	14
19	0 0 0 1 1 1 1 1 0 1 0 0 1 0 0 1	79,65631	1,25539%	'trainb'	'tansig'	'satlin'	14
20	0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1	79,84411	1,25244%	'trainb'	'tansig'	'tansig'	14

3.1.3.3 Evolução das Redes Neurais Recorrentes

Foi feita também a evolução dos parâmetros por 50 gerações para as redes neurais recorrentes. A Figura 3.12 mostra como foi esta evolução.

Durante a evolução, foram avaliados 879 conjuntos de parâmetros diferentes para Redes MLP. O processo de evolução demorou 514 minutos para terminar.

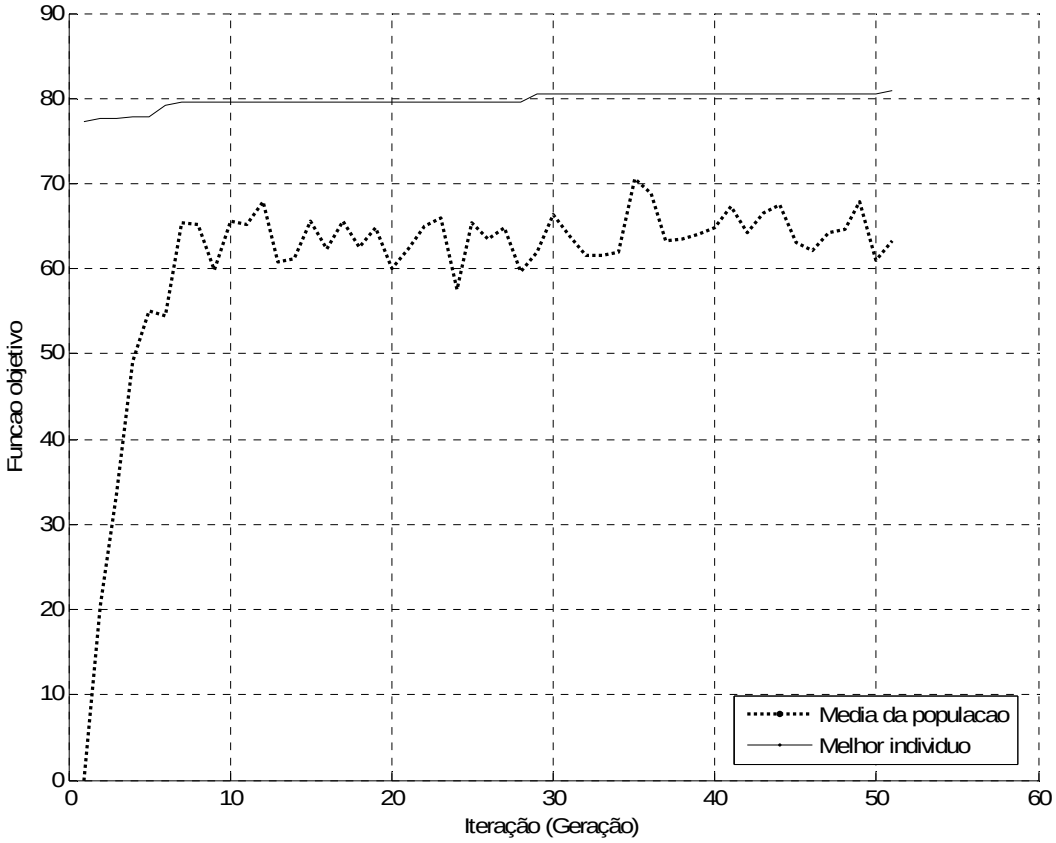


Figura 3.12 - Evolução da População para as Redes Recorrentes.

As 20 melhores redes neurais recorrentes que, na próxima etapa, irão compor o ensemble, têm as suas características listadas na Tabela 3.4.

Tabela 3.4 - As 20 melhores Redes Recorrentes.

					Fcn Ativação Camadas		Neurônios na Camada
Indivíduo	Genótipo	fitness	erro médio	treinamento	Intermediária	Saída	Intermediária
1	0 0 0 0 0 1 1 0 1 1 1 0 0 1 1 0	78,36754	1,27604%	'trainscg'	'logsig'	'tribas'	4
2	0 1 1 0 1 1 1 0 0 1 0 0 1 1 1 0	78,42642	1,27508%	'trainc'	'tribas'	'satlin'	11
3	0 1 1 1 0 1 0 0 1 1 1 0 0 1 1 0	78,45227	1,27466%	'traincgb'	'satlin'	'tribas'	4
4	0 0 0 0 0 1 1 0 1 1 1 0 0 0 0 0	78,4853	1,27412%	'trainscg'	'logsig'	'tribas'	16
5	0 0 0 1 0 1 1 0 0 1 1 1 1 0 0 0	78,53485	1,27332%	'trainb'	'logsig'	'poslin'	15
6	0 0 0 0 0 1 1 0 1 1 0 0 0 0 0 0	78,60984	1,27211%	'trainscg'	'logsig'	'satlins'	16
7	0 0 1 1 0 1 1 0 1 1 1 0 0 1 0 0	78,665	1,27121%	'trainbfg'	'logsig'	'tribas'	7
8	0 0 0 0 0 1 1 0 1 1 1 0 0 0 1 0	78,67895	1,27099%	'trainscg'	'logsig'	'tribas'	3
9	1 1 1 0 1 1 1 1 0 1 1 1 1 1 0 0	78,72794	1,27020%	'traingdx'	'tansig'	'poslin'	8
10	0 1 1 1 0 1 1 0 1 1 1 1 0 0 1 1	78,90299	1,26738%	'traincgb'	'logsig'	'tansig'	2
11	0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1	78,98491	1,26606%	'traincgb'	'logsig'	'logsig'	5
12	0 0 0 0 0 1 1 0 1 1 1 0 1 0 0 0	79,04538	1,26510%	'trainscg'	'logsig'	'tribas'	15
13	0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 0	79,07488	1,26462%	'trainscg'	'logsig'	'satlin'	15
14	0 0 0 0 0 1 1 0 0 1 1 0 0 1 1 0	79,11711	1,26395%	'trainscg'	'logsig'	'logsig'	4
15	0 0 1 1 1 1 1 0 1 1 1 0 1 0 1 1	79,11863	1,26392%	'trainbfg'	'tribas'	'tribas'	13
16	0 0 1 1 0 1 1 0 1 1 1 0 1 1 0 0	79,53863	1,25725%	'trainbfg'	'logsig'	'tribas'	8
17	0 1 1 1 0 1 1 0 1 1 1 0 1 1 0 1	79,64008	1,25565%	'traincgb'	'logsig'	'tribas'	9
18	0 0 0 0 0 1 1 0 1 1 1 0 1 1 1 0	79,69384	1,25480%	'trainscg'	'logsig'	'tribas'	11
19	0 0 0 0 0 1 1 0 1 1 0 0 1 0 0 0	79,7009	1,25469%	'trainscg'	'logsig'	'satlins'	15
20	0 0 1 1 0 1 1 0 1 1 1 0 0 0 1 0	80,12942	1,24798%	'trainbfg'	'logsig'	'tribas'	3

3.1.3.4 Codificação para Evolução das Redes RBF

Nas RNAs RBF, todos os parâmetros foram evoluídos com Algoritmos Genéticos. Nas tabelas abaixo podemos ver a listagem destes parâmetros:

Tabela 3.5 - Arquiteturas de treinamento: Redes RBF

Arquiteturas de treinamento para Redes Neurais RBF (3 opções: 2 bits)	
Função do MATLAB	**Descrição:**
newgrnn	Constrói uma rede neural de regressão generalizada.
newrb	Constrói uma rede neural de base radial simples.
newrbe	Constrói uma rede neural de base radial exata.

Tabela 3.6 - Espalhamento da Função de Base Radial: Redes RBF

Espalhamento (spread) da função de base radial: (256 opções: 8bits)
De 1 a 256

Como se vê, para representar o conjunto de parâmetros das RNAs RBF em um cromossomo binário, são necessários 10 bits. É importante salientar que os centros das funções de base radial são otimizados pelos algoritmos de treinamento utilizados, e, portanto, seus valores não serão comentados, pois não são relevantes para a comparação que será feita neste trabalho. Na Figura 3.13 podemos ver a representação do cromossomo para as RNAs RBF:

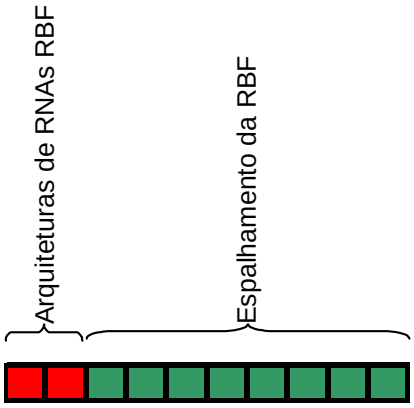


Figura 3.13 - Configuração do cromossomo: RNAs RBF

3.1.3.5 Evolução das Redes RBF

Como no caso das redes MLP e recorrentes, a evolução dos parâmetros por 50 gerações para as redes RBF. A Figura 3.14 mostra como foi esta evolução.

Durante a evolução, foram avaliados 403 tipos diferentes de Redes RBF. O processo de evolução demorou 163 minutos para terminar.

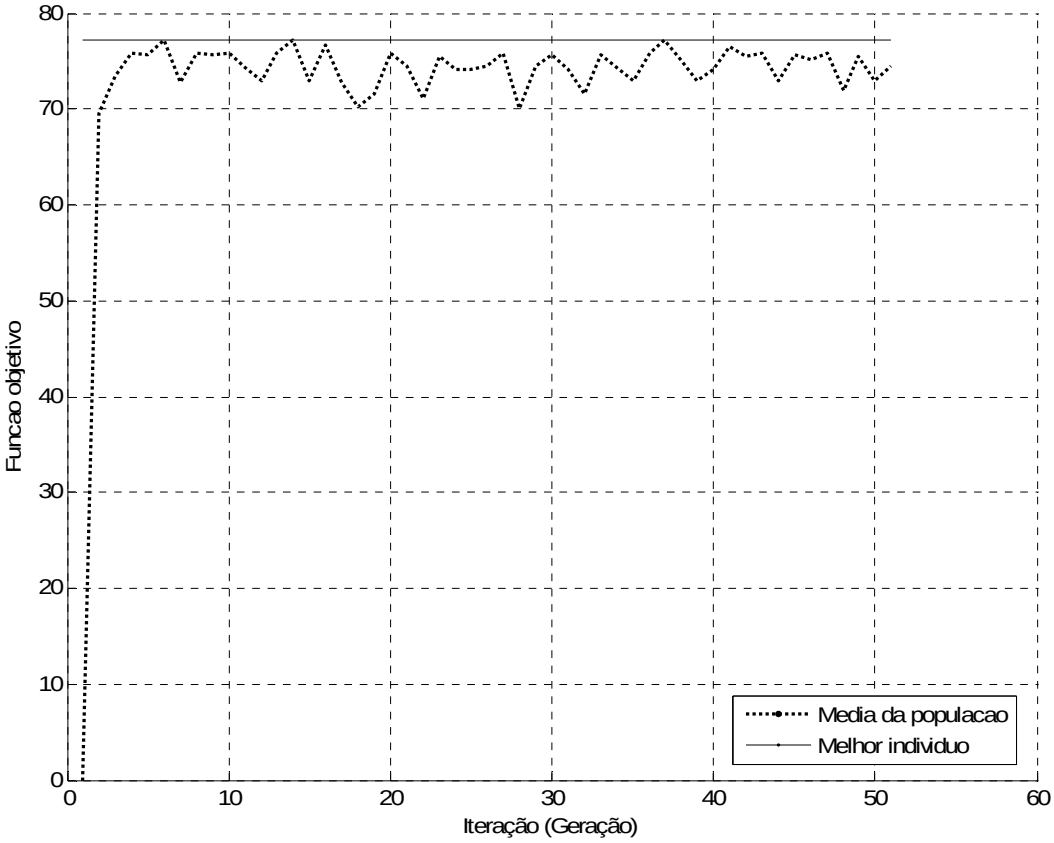


Figura 3.14 - Evolução da População para as redes RBF.

As 20 melhores redes RBF foram escolhidas para representarem a classe de redes RBF na composição do ensemble. Os seus genótipos e características estão listadas na Tabela 3.7.

Tabela 3.7 - As 20 melhores Redes RBF.

Indivíduo	Genótipo	fitness	erro médio	treinamento	Espalhamento RBF
1	1 1 1 0 1 1 0 0 0 1	75,70726	1,32088%	'newrbe'	222
2	1 1 1 0 0 1 1 0 0 0	75,70726	1,32088%	'newrbe'	239
3	1 1 1 0 0 1 0 0 1 1	75,70726	1,32088%	'newrbe'	226
4	1 1 1 0 0 0 0 1 1 0	75,70726	1,32088%	'newrbe'	251
5	1 1 1 0 0 0 1 1 1 1	75,70726	1,32088%	'newrbe'	245
6	1 1 1 0 1 0 1 1 1 0	75,70726	1,32088%	'newrbe'	203
7	1 1 1 0 0 0 0 0 1 0	75,70726	1,32088%	'newrbe'	252
8	1 1 1 0 0 1 1 1 1 0	75,70726	1,32088%	'newrbe'	235
9	1 1 1 0 0 1 0 0 1 0	75,70726	1,32088%	'newrbe'	227
10	1 1 1 0 0 1 0 0 0 0	75,70726	1,32088%	'newrbe'	224
11	1 1 0 1 1 0 1 1 1 1	75,834	1,31867%	'newrbe'	74
12	1 1 0 1 0 0 0 1 1 1	76,18649	1,31257%	'newrbe'	122
13	1 1 0 1 0 1 1 0 1 0	76,21517	1,31207%	'newrbe'	108
14	1 1 0 1 0 1 1 0 0 1	76,29297	1,31074%	'newrbe'	110
15	1 1 0 1 0 1 1 1 1 1	76,36802	1,30945%	'newrbe'	106
16	1 1 0 1 0 1 1 0 1 1	76,4657	1,30778%	'newrbe'	109
17	1 1 0 1 0 0 1 1 0 0	76,57174	1,30596%	'newrbe'	119
18	1 1 0 1 0 0 0 1 0 0	76,73733	1,30315%	'newrbe'	120
19	1 1 0 1 0 0 0 1 1 0	76,87571	1,30080%	'newrbe'	123
20	0 1 1 0 0 0 1 1 0 1	77,23834	1,29469%	'newrb'	246

3.1.3.6 Codificação para Evolução das Redes Neurais Nebulosas

Os seguintes parâmetros foram evoluídos utilizando Algoritmos Genéticos:

Tabela 3.8 - Funções de pertinência: RNAs Nebulosas.

Funções de Pertinência (8 opções: 3 bits)	
Função do MATLAB	**Descrição:**
'dsigmf'	Diferença de duas funções de pertinência sigmoidais
'gauss2mf'	Curva gaussiana de dois lados
'gaussmf'	Curva gaussiana
'gbellmf'	Curva tipo sino generalizada
'pimf'	Curva em formato de Pi
'psigmf'	Produto de duas funções de pertinência sigmoidais
'trapmf'	Função de pertinência trapezoidal
'trimf'	Função de pertinência triangular

Tabela 3.9 - Número de épocas de treinamento: RNAs Nebulosas.

Número de épocas de treinamento (128 opções: 7bits)
De 10 a 137

Tabela 3.10 - Número de funções de pertinência por entrada: RNAs Nebulosas.

Número de Funções de Pertinência por entrada (64 opções: 6bits)
De 2 a 65

Como é possível notar, neste caso será preciso 16 bits para que seja possível representar o conjunto de parâmetros descritos acima. Na Figura 3.15, pode-se ver o cromossomo que representa este conjunto de parâmetros.

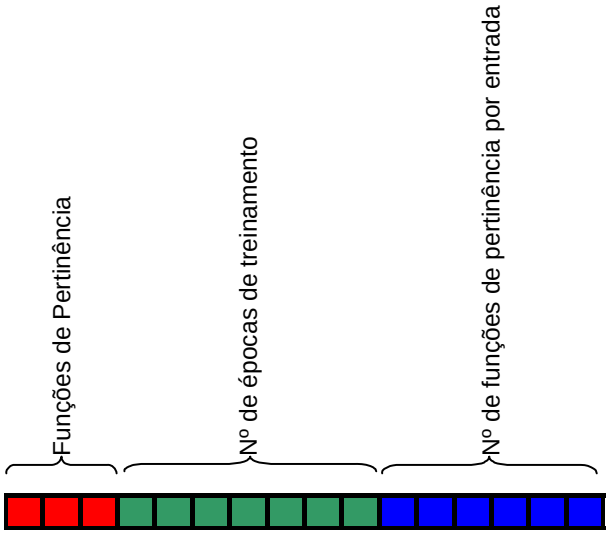


Figura 3.15 - Configuração do Cromossomo: Redes Neuro Fuzzy

3.1.3.7 Evolução das Redes Neurais Nebulosas

Após a evolução dos parâmetros das redes neurais nebulosas por 50 gerações se chegou a uma população de redes nebulosas evoluída. A Figura 3.16 mostra como foi esta evolução. Durante a evolução, foram avaliados 879 tipos diferentes de Redes Nebulosas. O processo de evolução demorou 207 minutos para terminar.

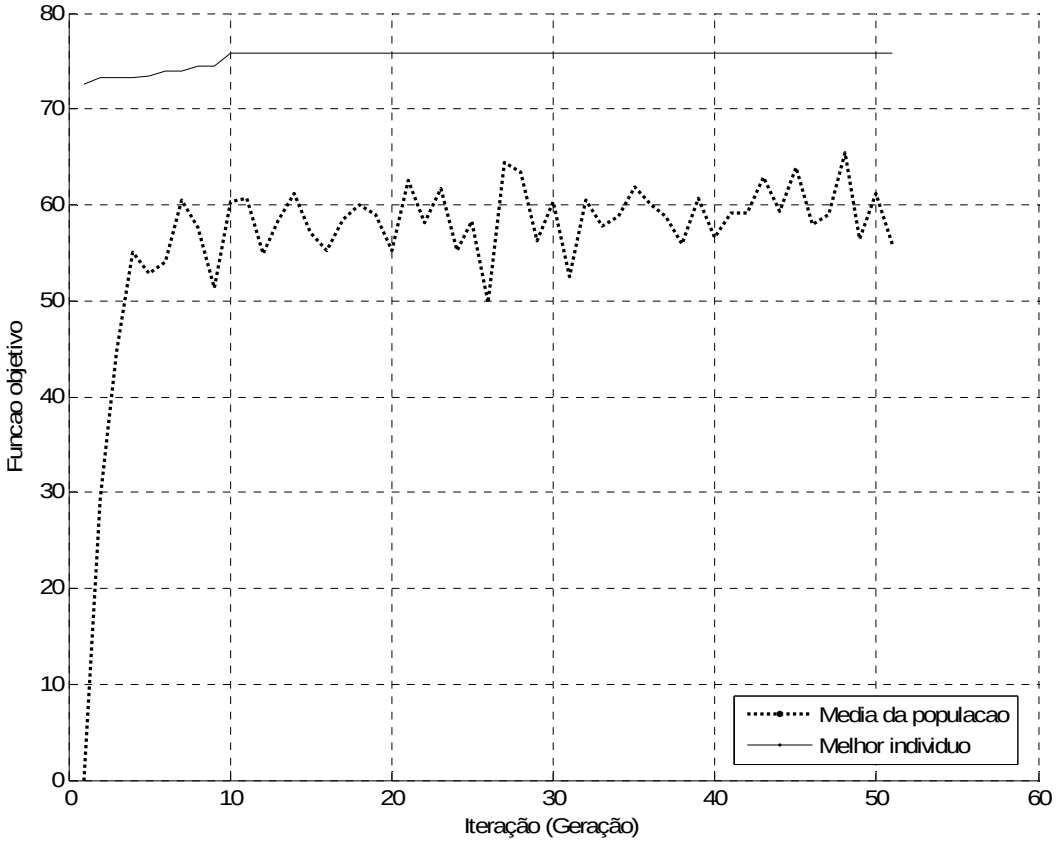


Figura 3.16 - Evolução da População para as Redes Nebulosas.

Assim sendo, foram selecionadas as 20 melhores parametrizações de Redes Neurais Nebulosas. Estes indivíduos e suas características estão listados na Tabela 3.11.

Tabela 3.11 - As 20 melhores Redes Nebulosas.

Indivíduo	Genótipo	fitness	erro médio	Fcn Pert.	No. Fcn Pert. p/ Entrada	Número de Épocas Treinamento
1	1 0 0 1 1 0 1 1 1 0 1 0 0 0 0 1	64,20158	1,55759%	'dsigmf'	2	105
2	0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 1	66,64269	1,50054%	'dsigmf'	2	80
3	0 0 0 1 0 0 0 1 1 0 1 0 0 0 0 1	60,20842	1,66090%	'dsigmf'	2	9
4	0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1	68,19807	1,46632%	'dsigmf'	2	16
5	0 0 0 1 0 0 0 0 0 1 0 1 0 0 1 1	72,16578	1,38570%	'dsigmf'	59	3
6	0 0 0 1 0 0 0 0 0 1 1 1 0 0 1 1	67,36955	1,48435%	'dsigmf'	59	2
7	0 0 0 1 1 0 0 1 0 0 0 0 0 0 0 1	68,48399	1,46020%	'dsigmf'	3	112
8	1 1 0 1 1 0 1 1 1 0 0 0 0 0 0 1	72,37969	1,38160%	'psigmf'	3	104
9	1 0 0 1 0 0 0 0 0 1 1 1 0 1 1 1	68,788	1,45374%	'dsigmf'	52	2
10	0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 1	69,38663	1,44120%	'dsigmf'	61	1
11	0 0 0 1 0 0 0 0 0 0 1 1 0 1 1 1	67,84773	1,47389%	'dsigmf'	52	1
12	0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1	62,57896	1,59798%	'dsigmf'	3	14
13	1 0 0 1 0 0 0 0 0 1 1 1 1 1 1 1	71,35155	1,40151%	'dsigmf'	42	2
14	1 0 0 1 0 0 0 0 0 0 1 1 0 1 0 1	68,92599	1,45083%	'dsigmf'	51	1
15	1 0 0 1 0 0 0 0 0 1 1 1 1 1 1 1	68,72586	1,45506%	'dsigmf'	43	2
16	1 0 0 1 0 0 0 0 0 0 1 1 1 0 1 1	73,01438	1,36959%	'dsigmf'	37	1
17	1 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1	64,34992	1,55400%	'dsigmf'	3	54
18	0 0 0 1 0 0 0 0 1 0 1 1 0 1 0 1	70,60739	1,41628%	'dsigmf'	50	6
19	1 0 0 1 0 0 0 0 0 0 1 1 0 1 1 1	66,29382	1,50844%	'dsigmf'	53	1
20	1 1 0 1 0 1 1 0 1 0 0 0 0 0 0 1	73,55843	1,35946%	'psigmf'	3	39

3.2 Seleção dos Previsores para compor o Ensemble

Após a evolução dos parâmetros dos previsores, serão utilizados os 20 melhores conjuntos de parâmetros de cada um dos quatro modelos básicos de previsão. Sendo assim, será usado um total de 80 aproximações para a série temporal de cargas.

Neste contexto, o algoritmo genético será responsável por selecionar as aproximações que irão compor o ensemble. Desta forma, o cromossomo de cada indivíduo (ensemble candidato a ser a solução do Algoritmo Genético) vai ter 80 bits onde cada um dos bits vai indicar se a aproximação do previsor em questão vai fazer parte do ensemble (valor do bit igual a 1) ou não (valor do bit igual a 0). Na Figura 3.17, é ilustrada a representação do cromossomo.

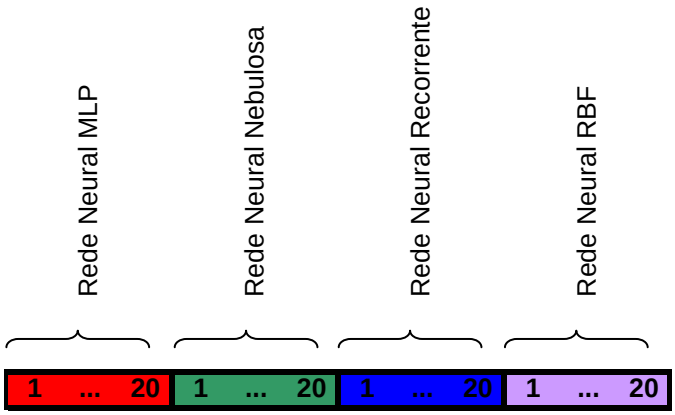


Figura 3.17 - Configuração do Cromosso dos Ensembles

No entanto, diferentemente da evolução dos previsores, onde o critério adotado para selecionar os melhores era o erro médio absoluto acumulado nos sete dias de semana durante o período que vai da semana 53 até a 68, o critério adotado na evolução dos ensembles é o erro médio de previsão no período entre as semanas 69 e 84 para cada dia de semana individualmente. Assim, serão evoluídos sete ensembles. Um para cada dia da semana.

Portanto, para poder comparar o desempenho dos ensembles com o desempenho dos previsores individuais, é necessário tomar como referência o desempenho de cada um dos melhores previsores dos quatro modelos básicos de previsão em cada um dos sete dias da semana, apresentado na Tabela 3.12:

Tabela 3.12 - Erros por dia de semana dos melhores previsores.

	segundas	terças	quartas	quintas	sextas	sábados	domingos
Melhor MLP	1,4386%	0,8206%	0,4130%	0,9640%	0,6447%	1,3999%	1,3999%
Melhor RNR	1,4792%	0,9397%	0,5990%	1,0830%	0,8607%	1,2721%	1,6153%
Melhor RBF	1,9936%	0,7733%	1,1165%	0,9831%	0,7246%	1,2136%	1,5024%
Melhor ANFIS	1,2778%	1,1460%	0,4509%	0,7993%	0,8934%	1,2466%	1,6194%

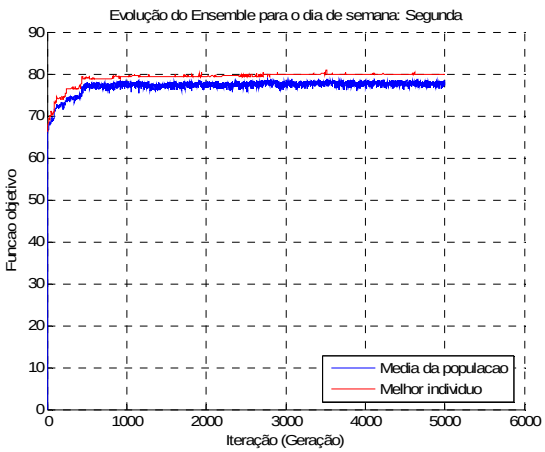
Não é difícil perceber na Tabela 3.12 que as segundas, sábados e domingos têm erros de previsão quase duas vezes maiores do que os outros dias de semana. Estes dias de semana mais difíceis de prever tendem a ter suas previsões melhoradas após a composição final dos ensembles.

Para a evolução de cada um dos ensembles, foram utilizados os seguintes parâmetros no Algoritmo Genético:

- Porcentagem de indivíduos selecionados para a próxima geração: 40%;
- Número de indivíduos: 50;
- Taxa de Crossover: 80%;
- Tipo de recombinação: crossover de dois pontos (ponto duplo) com substituição reduzida (reduced surrogate);
- Probabilidade de mutação: 4%;
- Tipo de seleção: roleta;
- Subtipo de seleção: normal;
- Tipo de reinserção: fitness;
- Número de gerações: 5000.

Na composição da solução de cada ensemble da população de 50 ensembles a cada geração do Algoritmo Genético, os previsores que irão compor o ensemble terão as suas soluções combinadas usando média simples, pois outros métodos de combinação não se revelaram muito mais eficientes do que este.

As evoluções dos ensembles para cada dia da semana são ilustradas nas figuras a seguir.



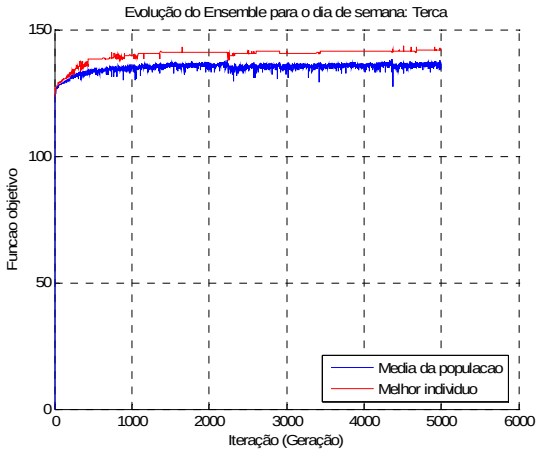
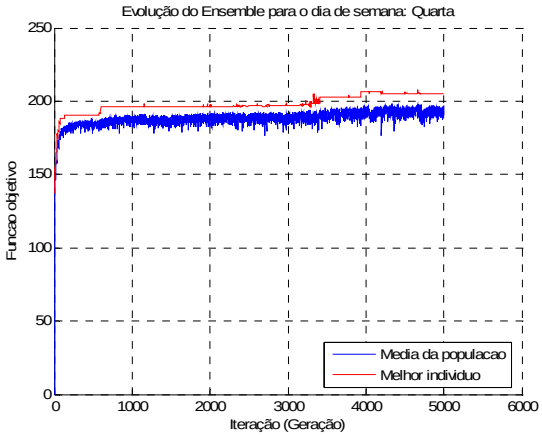


Figura 3.18 - Evolução dos ensembles para segundas e terças.



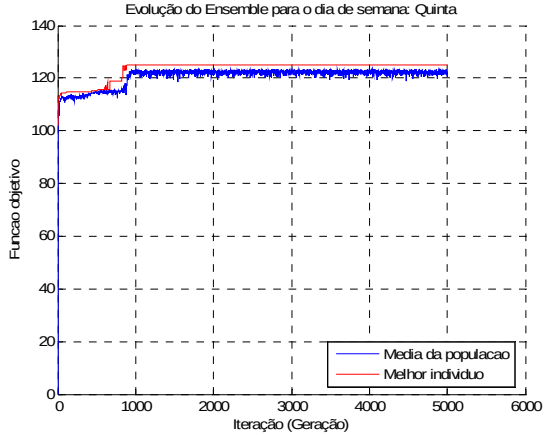
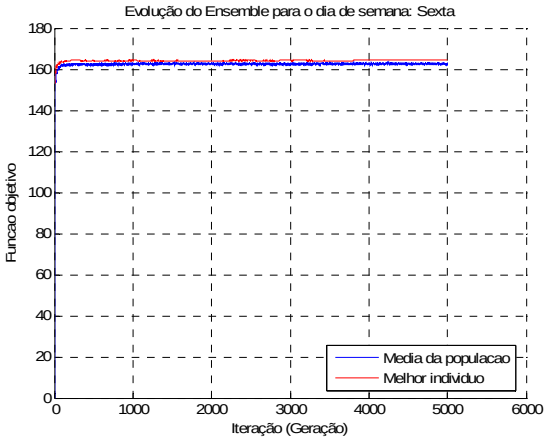


Figura 3.19 - Evolução dos Ensembles para quartas e quintas.



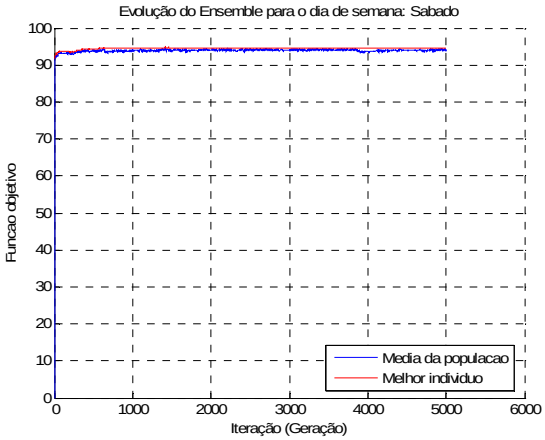


Figura 3.20 - Evolução dos Ensembles para sextas e sábados

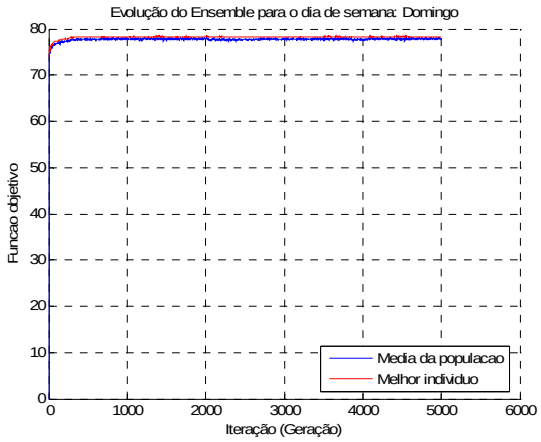


Figura 3.21 - Evolução dos Ensembles para os domingos.

Como se vê nas figuras 3.18, 3.19, 3.20 e 3.21, os melhores ensembles foram formados para as terças, quartas, quintas e sextas. Os outros dias tiveram valores de fitness menores quando comparado com o desempenho dos melhores previsores (Tabela 3.12), como pode ser visto na tabela abaixo:

Tabela 3.13 - Desempenho comparado do Ensemble após sua evolução.

	segundas	terças	quartas	quintas	sextas	sábados	domingos
Melhor MLP	1,4386%	0,8206%	**0,4130%**	0,9640%	**0,6447%**	1,3999%	**1,3999%**
Melhor RNR	1,4792%	0,9397%	0,5990%	1,0830%	0,8607%	1,2721%	1,6153%
Melhor RBF	1,9936%	**0,7733%**	1,1165%	0,9831%	0,7246%	**1,2136%**	1,5024%
Melhor ANFIS	**1,2778%**	1,1460%	0,4509%	**0,7993%**	0,8934%	1,2466%	1,6194%
Ensemble	**1,2495%**	**0,7040%**	**0,4877%**	**0,7989%**	**0,6086%**	**1,0563%**	**1,2774%**
Ganho Desempenho	**2,2148%**	**8,9540%**	**-18,0800%**	**0,0401%**	**5,5899%**	**12,9577%**	**8,7462%**

A Tabela 3.13 faz uma comparação com os resultados da Tabela 3.12. É possível perceber que os melhores desempenhos dos melhores previsores (destacados em negrito) são, na maioria dos casos, inferiores ao desempenho do ensemble. Vale destacar que há duas principais características nas previsões dos ensembles. Primeiro que ele melhora sensivelmente as previsões nos dias da semana que os previsores individuais são mal sucedidos. Segundo que, em todos os dias da semana, ele melhora a robustez da previsão, evitando oscilações bruscas.

3.3 Teste de previsão usando o ensemble gerado

Finalmente se chegou a sete modelos de previsão para serem utilizados na previsão desta série temporal de cargas, ou seja, um modelo para cada dia da semana. Na Tabela 3.14 se vê quais foram os previsores elegidos para compor cada um dos ensembles.

Tabela 3.14 - Modelos de previsão adotados.

	Redes MLP																				Redes Recorrentes																			
segunda	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0
terça	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
quarta	0	0	0	1	0	0	1	1	1	1	1	0	1	1	0	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	1
quinta	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0
sexta	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0
sábado	1	1	0	1	1	0	1	1	1	1	0	0	1	1	0	1	1	0	1	1	0	1	0	0	1	1	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0
domingo	0	0	0	0	1	1	0	0	1	1	1	0	1	1	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	Redes RBF																				Redes Nebulosas																			
segunda	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
terça	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
quarta	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0
quinta	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
sexta	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	1	0
sábado	0	0	1	1	1	0	1	0	0	0	0	1	1	1	0	0	0	1	1	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
domingo	1	1	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1	0	1	0	1	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0

É muito curioso observar na Tabela 3.14 o fato de alguns tipos de redes terem sido priorizadas para compor o ensemble em determinados dias de semana e em outros não. As Redes Neurais RBF, por exemplo, foram muito utilizadas nas previsões das quintas e domingos. No entanto, praticamente não foram utilizadas nas previsões das segundas, terças e quartas. Outro exemplo nítido é o caso das Redes Neurais MLP. Elas foram praticamente elegidas para prever as segundas-feiras, enquanto as outras redes praticamente não participaram das previsões das segundas. Mais curioso ainda é perceber que o melhor conjunto de previsores para as segundas-feiras foram as redes neurais nebulosas e ainda assim a combinação de Redes MLP gerou um resultado melhor do que quando usado o melhor previsor da população (vide Tabela 3.13).

Todos estes fatos levam à conclusão certa de que a combinação é a melhor estratégia quando se deseja errar menos, aumentar a confiabilidade do modelo e conseguir robustez.

Assim sendo, finalmente foram usados os sete ensembles para prever valores futuros da série de cargas até agora não tocados: o intervalo que vai da semana 85 a semana 100.

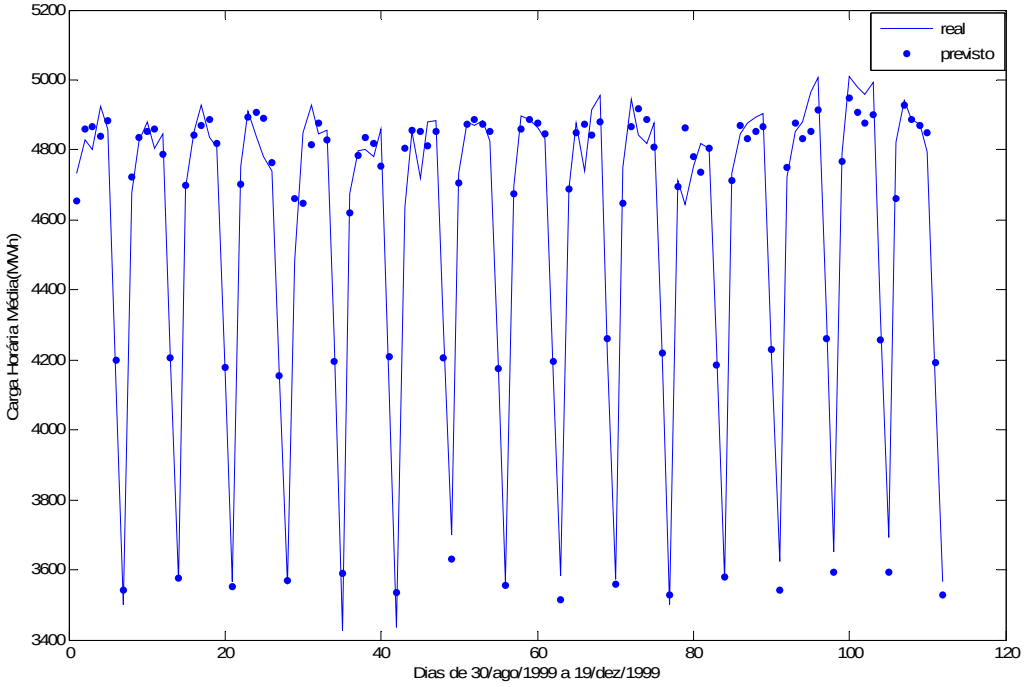


Figura 3.22 - Previsão diária do Ensemble (da semana 85 a 100)

A Figura 3.22 mostra que o ensemble conseguiu prever os valores futuros da série de cargas com uma boa precisão e sem apresentar erros grosseiros em nenhum ponto de previsão. No entanto, ainda é necessária uma comparação de previsão com aqueles melhores previsores obtidos na evolução dos previsores individuais. Os previsores que serão usados nas comparações serão os mesmos usados nas comparações da Tabela 3.13. No entanto, agora eles farão a previsão do período que vai da semana 85 a 100.

A comparação no período de previsão é descrita na Tabela 3.15.

Tabela 3.15 - Comparação das previsões do ensemble com as previsões dos melhores previsores.

	Erro do ensemble	Erro do melhor ANFIS	Erro da melhor MLP	Erro da melhor RBF	Erro da melhor RNR
segunda	1,308602%	1,493730%	1,354941%	1,611217%	**1,154030%**
terça	1,016545%	1,101555%	**0,962446%**	1,191683%	1,051593%
quarta	1,186056%	1,322113%	**1,167349%**	**1,119413%**	**1,090042%**
quinta	1,180181%	**1,020084%**	1,192088%	1,386649%	**1,172826%**
sexta	0,953057%	1,150908%	0,984195%	**0,922179%**	0,976900%
sábado	1,267415%	1,280592%	**1,131369%**	1,382765%	**1,172218%**
domingo	1,428569%	1,421638%	**1,410873%**	1,463115%	1,431786%
ACUMULADO	1,191489%	1,255803%	**1,171894%**	1,296717%	**1,149913%**

É perceptível que o ensemble continua fornecendo boas previsões e continua robusto uma vez que foi mais bem sucedido do que a maioria das previsões das melhores redes. No entanto, ainda se vê que ele não conseguiu superar (por pouco) a melhor rede recorrente e a melhor MLP. Este fato se deve, provavelmente, ao fato de que os previsores estão muito bem preparados para prever este tipo de série temporal. Contudo, vale salientar que há uma grande chance destes melhores previsores terem sido mais bem sucedidos do que o ensemble somente neste período de previsão. Assim, possivelmente em outro trecho de previsão eventualmente a melhor rede RBF seja, por exemplo, mais bem sucedida. Entretanto, o desempenho do ensemble pode ser assegurado como melhor do que a maioria das previsões, daí a robustez do modelo do ensemble.

3.4 Teste de previsão usando ensemble com todos os previsores

Ainda assim, foi feito um teste utilizando todas as previsões dos 80 melhores previsores, ao invés de utilizar os previsores selecionados por Algoritmos Genéticos. A Tabela 3.16 mostra uma comparação dos resultados quando se utilizam os ensembles dos previsores selecionados por Algoritmos Genéticos e quando se usa todos os previsores evoluídos para compor o ensemble.

Tabela 3.16 - Comparação dos Ensembles dos Selecionados X Todos

	Ensemble Selecionados	Ensemble Todos
segunda	1,308602%	1,325303%
terça	1,016545%	1,013127%
quarta	1,186056%	1,146162%
quinta	1,180181%	1,218930%
sexta	0,953057%	0,976844%
sábado	1,267415%	1,268040%
domingo	1,428569%	1,463459%
ACUMULADO	1,191489%	1,201695%

É possível perceber que, como era esperado (veja tópico 2.5.3), o ensemble dos previsores selecionados foi mais bem sucedido, em média, do que o ensemble de todos os previsores.

Capítulo 4: Conclusões

O problema de previsão de séries temporais um passo à frente de séries de cargas de energia elétrica, abordado neste trabalho, é extremamente importante no processo de tomada de decisão, quando se deseja otimizar o fornecimento de energia elétrica e a utilização dos recursos disponíveis para uma distribuição de energia elétrica mais segura.

Tradicionalmente, os modelos clássicos de previsão de séries temporais têm sido utilizados para fazer este tipo de previsão. No entanto, se sabe que a série de cargas está sujeita a fatores cujo comportamento é não-linear. Este fato faz com que os modelos clássicos sejam ineficientes na previsão deste tipo de série temporal, pois eles pertencem à classe de modelos lineares para previsão de séries temporais.

Neste contexto, as redes neurais artificiais têm sido muito utilizadas atualmente para prever séries temporais com características não-lineares. No entanto, ainda assim, para que as redes neurais tenham um bom desempenho para fazer determinado tipo de previsão, seus parâmetros e sua arquitetura devem ser exaustivamente estudados até que se consiga atingir um bom resultado. Este estudo será muito dependente do conhecimento que o usuário que está fazendo as previsões tem sobre a teoria de Redes Neurais Artificiais e de suas experiências prévias.

Mesmo assim, dado que seja possível construir um previsor especialista que seja capaz de prever determinado fenômeno com boa eficácia, ainda é comum que este previsor especialista comece a registrar altos níveis de erro no caso da série temporal começar a mudar o seu comportamento com o tempo.

É com o intuito de resolver estes problemas que foi proposta a maioria dos modelos de previsão híbridos e modulares, como é o caso dos “ensembles”.

Neste trabalho, foi desenvolvido um modelo de “ensembles” onde inicialmente é criada uma população de previsores básicos cujos parâmetros são evoluídos por Algoritmos Genéticos. Estes previsores básicos são três tipos de RNAs e um tipo de Rede Neural Nebulosa. Em seguida, os resultados dos previsores evoluídos são selecionados de maneira a otimizar o resultado final dos “ensembles” quando estes resultados selecionados forem combinados. Para selecionar os previsores que irão compor os “ensembles” finais, são novamente utilizados Algoritmos Genéticos.

Como foi demonstrado neste trabalho, a utilização de ensembles para a previsão de séries temporais se revelou uma técnica robusta e capaz de ter um desempenho superior quando comparado ao desempenho de um previsor individualmente.

Os testes realizados mostraram que, na maioria dos casos, a previsão do ensemble resultou erros menores àqueles obtidos pelos melhores previsores de cada uma das redes básicas utilizadas. Assim, como havia uma expectativa de que o ensemble conseguisse superar os melhores previsores em todos os casos, é necessário salientar que os casos nos quais os ensembles são superados pelos melhores previsores individuais ocorrem somente em determinados períodos de previsão. Sendo assim, quando os ensembles são testados em outro período de previsão, eventualmente outros previsores individuais conseguirão superar o desempenho dos ensembles e aqueles que conseguiram superar os ensembles em outro período de previsão não serão tão eficientes como eram.

Neste sentido, quando vários períodos de previsão são considerados, o desempenho do ensemble pode ser considerado superior, pois se fosse sempre usado o melhor previsor de um determinado período, ele não seria sempre melhor que o ensemble em outros períodos.

Portanto, foi possível concluir que os ensembles conferem ao modelo uma estabilidade de desempenho quando ele é utilizado em períodos diferentes com diferentes características de comportamento da série temporal, o que revela ainda outra característica essencial do modelo de previsão utilizando ensembles: a flexibilidade de utilização.

Referências

[1] ASSAAD, M.; BONÉ, R. – "**Improving Time Series Prediction By Recurrent Neural Network Ensembles**" em Université François-Rabelais, Tours, 2003.

[2] BALLINI, R. – "**Análise e previsão de vazões utilizando modelos de séries temporais, redes neurais e redes neurais nebulosas**", Tese de Doutorado, Unicamp, Campinas, São Paulo, Brasil, 2000, 169 p.

[3] BOX, G.; JENKINS, G.; REINSEL, G. C. – "**Time Series Analysis, Forecasting and Control**", 3ª ed., Holden Day, Oakland, California, EUA, 1994.

[4] BREIMAN, L. – "**Bagging predictors**", Machine Learning 24, (1996). p. 123-140.

[5] CHEN, S.; COWAN, C.F.N.; GRANT, P. M. – "**Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks**" em IEEE Transactions on Neural Networks, vol. 2, no. 2, 1991, p. 302-309.

[6] CHERKAUER, K. J. – "**Human expert level performance on a scientific image analysis task by a system using combined artificial neural networks**", em Proc. 13th AAAI Workshop on Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms, Portland, OR, 1996. p. 15-21.

[7] COELHO, L. S.; CANCIGLIERI JÚNIOR, O. – "**Rede neural de base radial aplicada em previsão de séries temporais: algoritmo e aplicação.**", em XX Encontro Nacional de Engenharia de Produção, XX ENEGEP, São Paulo, SP, 2000.

[8] CONDORCET, N. C. – "**Essai sur l' application de l' analyse à la probabilité des decisions rendues à la pluralité des voix**", Imprimerie Royale, Paris, 1785.

[9] DE JONG, K.A. – "**An analysis of the behavior of a class of genetic adaptive systems**", Ph.D. Dissertation, University of Michigan, Ann Arbor, 1975.

[10] DIETTERICH, T. G. – "**Ensemble methods in machine learning. In J. Kittler and F. Roli, editors, Multiple Classifier Systems**", em First International Workshop, MCS 2000, vol. 1857 of Lecture Notes in Computer Science, 2000, p. 1-15.

[11] EFRON, B.; TIBSHIRANI, R. – "**An Introduction to the Bootstrap.**", Chapman & Hall, New York, 1993.

[12] ELMAN, J. L. – "**Finding structure in time**" Cognitive Science, vol. 14, 1990, p. 179-211.

[13] FREUND, Y. – "**Boosting a weak algorithm by majority**", em Information and Computation 121, 1995. p. 256-285.

[14] FREUND, Y.; SCHAPIRE, R. E. – "**A decision-theoretic generalization of on-line learning and an application to boosting**", em Journal of Computer and System Sciences 55, 1997. p. 119-139.

[15] FULLÉR, R. – "**Neural Fuzzy Systems**", Åbo Akademy University, Finlândia, 1995.

[16] GONZÁLEZ, J; ROJAS, I.; POMARES, H. – "**Evolutive Identification of Fuzzy Systems for Time-Series Prediction**", em Lecture Notes in Computer Science, vol. 2439, 2002, p. 517-526.

[17] HAMPSHIRE, J.; WAIBEL, A. – "**A novel objective function for improved phoneme recognition using time-delay neural networks**", em IEEE Trans. Neural Networks 1, 1990. p. 216-228.

[18] HANSEN, L. K.; SALAMON, P. – "**Neural network ensembles**", em IEEE Trans. Pattern Analysis and Machine Intelligence 12, nº 10, 1990, p. 993-1001.

[19] HOLCOMB, T. R.; MORARI, M. – "**PLS/Neural Networks**", em Comput. Chem. Engineering, vol. 16, no. 4, 1992, p. 393-411.

[20] HOLLAND, J.H. – "**Adaptation in Natural and Artificial Systems**", University of Michigan Press, 1975.

[21] INOUE, H.; NARIHISA, H. – "**Ensemble Self-Generating Neural Networks for Chaotic Time Series Prediction**" em Proc. of IPMU2000 (8th Information Processing and Management of Uncertainty in Knowledge-Based Systems Conference), 2000, p. 1524-1531.

[22] JANG, J. S. R. – "**ANFIS: Adaptive-network-based fuzzy inference system**", IEEE Transactions on Systems, Man, and Cybernetics, No.23, 1993, p. 665-685.

[23] JIMENEZ, D. – "**Dynamically weighted ensemble neural networks for classification**", em Proc. IEEE Int. Joint Conf. on Neural Networks, vol.1, Anchorage, AK, 1998. p. 753-756.

[24] KADOWAKI, M.; OHISHI, T.; SOARES, S.; BALLINI, R. – "**Short-term load forecasting using nerofuzzy network model**", em XIV Congresso Brasileiro de Automática - CBA 2002, Vol. 1, Natal, RN, BRASIL, 2002, p.365-370.

[25] KROGH, A.; VEDELSBY, J. – "**Neural network ensembles, cross validation, and active learning**", em Advances in Neural Information Processing Systems 7, eds. G. Tesauro, D. S. Touretzky, and T. K. Leen, MIT Press, Cambridge, MA, 1995. p. 231-238.

[26] KUO, J.; PRINCIPE, J. C.; DEVRIES, B. – "**Prediction of chaotic time series using recurrent neural networks**", em Intl. Work. on Neural Networks for Signal Processing, 1992, p. 436-443.

[27] LIMA, C. A. M.; COELHO, A. L. V.; VON ZUBEN, F. J. – "**Ensembles of Support Vector Machines for Regression Problems**" em Procs. INNS-IEEE International Joint Conference on Neural Networks (IJCNN), vol. 3, 2002, p. 2381-2386.

[28] LIU, Y.; YAO, X. – "**Ensemble learning via negative correlation**", em Neural Networks 12, 1999. p. 1399-1404.

[29] MACLIN, R.; SHAVLIK, J. W. – "**Combining the predictions of multiple classifiers: Using competitive learning to initialize neural networks**", em Proc. 14th Int. Joint Conf. on Artificial Intelligence, Montreal, Canada, 1995. p. 524-530.

[30] MAGALHÃES, M. H. – "**Redes Neurais, Metodologias de Agrupamento e Combinação de Previsores Aplicados à Previsão de Vazões Naturais**", Dissertação de Mestrado, Unicamp, Campinas, São Paulo, Brasil, 2004, 107 p.

[31] MICHALEWICZ, Z. – "**Genetic algorithms + Data Structures = Evolution Programs**", 3rd edition, Springer-Verlag, 1996.

[32] MINSKY, M. L.; PAPERT, S. A. – "**Perceptrons: Introduction to Computational Geometry**", Expanded edition, MIT Press, 1988.

[33] MORETTIN, P. A.; TOLOI, C. M. – "**Análise de Séries Temporais**", 1ª ed., Edgard Blücher Ltda, São Paulo, Brasil, 2004, 535 p.

[34] OPITZ, D. W.; SHAVLIK, J. W. – "**Actively searching for an effective neural network ensemble**", em Connection Science 8, 1996. p. 337-353.

[35] OPITZ, D. W.; SHAVLIK, J. W. – "**Generating accurate and diverse members of a neural network ensemble**", em Advances in Neural Information Processing Systems 8, eds. D. S. Touretzky, M. Mozer, and M. Hasselmo, MIT Press, Cambridge, MA, 1996. p. 535-541.

[36] PERRONE, M.P.; COOPER, L.N. – "**When networks disagree: ensemble methods for hybrid neural networks."** em Artificial Neural Networks for Speech and Vision, 1993. p. 126–142.

[37] PRINCIPE, J. C;, RATHIE, A.; KUO, J. – "**Prediction of chaotic time series with neural networks, the issue of dynamic modeling**", em Intl. Journal of Bifurcation and Chaos, Vol. 2, No. 4, 1992, p. 989-996.

[38] RAHMAN, S.; HAZIM, O. – "**A generalized knowledge-based short-term load-forecasting technique**", IEEE Transactions on Power Systems 8(2), 1993.

[39] ROTHERMICH, J. – "**Financial Time-Series Prediction using Negatively Correlated Neural Network Ensembl**es" Mini-projeto em The University of Birmingham School of Computer Science, 2002.

[40] SCHAPIRE, R. E. – "**The strength of weak learnability**", em Machine Learning 5, 1990. p. 197-227.

[41] SIMON, H. A. – "**The new science of management decision**", 3ª ed. revisada, Prentice Hall Inc., 1977, 175 p.

[42] TAKAGI, T. & SUGENO, M. "**Fuzzy Identification of Systems and Its Applications to Modeling and Control**" em IEEE Transactions on Systems, Man and Cybernetics, vol. SMC-15, 1985, p. 116-132.

[43] THOMAZ, A.; FERREIRA, B. R.; BARBOSA, L. M. J.; LEONE FILHO, M. A; SILVA, M. A.; QUISPE, N. R. P – "**Toolbox AGBIN**", Mini-projeto da disciplina "IA707 - Computação Evolutiva", Unicamp, Campinas, SP, Brasil, 2004.

[44] TOULSON, D. L.; TOULSON, S. P. – "**Use of Neural Network Ensembles for Portfolio Selection and Risk Management**" em NeuroCOLT Technical Report Series, NC-TR-96-046, 1996.

[45] UEDA, N. – "**Optimal linear combination of neural networks for improving classification performance**", em IEEE Trans. Pattern Analysis and Machine Intelligence 22, 2000. p. 207-215.

[46] VALENTINI, G.; MASULLI, F. – "**Ensembles of learning machines.**" Em Series Lecture Notes in Computer Sciences, Springer-Verlag, vol. 2486, 2002. p. 3-19.

[47] VON ZUBEN, F.J. – "**Computação Evolutiva: Uma Abordagem Pragmática**", material de apoio da disciplina "IA707 - Computação Evolutiva", Unicamp, Campinas, SP, Brasil, 2004.

[48] WALL, M. – "**Overview of genetic algorithms**", Massachusetts Institute of Technology

[49] WAN, E. A. – "**Modeling Nonlinear Dynamics with Neural Networks: Examples in Time Series Prediction**", em Proc. SPIE Vol. 2204, Fifth Workshop on Neural Networks: Academic/Industrial/NASA/Defense. An International Conference on Computational Intelligence: Neural Networks, Fuzzy Systems, Evolutionary Programming and Virtual Reality, 1993, p. 327.

[50] WIDROW, B.; RUMELHART, D. E.; LEHR, M. A. – "**Neural networks: applications in industry, business and science**", Communications of the ACM, vol. 37, no. 3, 1994, p. 93-105.

[51] WOLPERT, D. H. – "**Stacked generalization**", em Neural Networks 5, (1992). p. 241-259.

[52] WU, J. X.; ZHOU, Z. H.; CHEN, Z. Q. – "**Ensemble of GA based selective neural network ensembles**." em Proceedings of the 8th International Conference on Neural Information Processing (ICONIP'01), vol.3, Shanghai, China, 2001. p. 1477-1482.

[53] YAO, X.; LIU, Y. – "**Making use of population information in evolutionary artificial neural networks**", em IEEE Trans. Systems, Man and Cybernetics - Part B: Cybernetics 28, 1998. p. 417-425.

[54] ZADEH, L. A. – "**Fuzzy Sets**" em Information and Control, 8, 1965, p. 338-353.

[55] ZEKIC, M. – "**Neural Network Applications in Stock Market Predictions – A Methodology Analysis**", em Proceedings of the 9th International Conference on Information and Intelligent Systems '98, 1998, p. 255-263.

[56] ZHOU, Z. H.; WU, J. X.; TANG W.; CHEN, Z. Q. – "**Selectively ensembling neural classifiers.**" em Proceedings of the International Joint Conference on Neural Networks (IJCNN'02), vol.2, Honolulu, HI, 2002. p. 1411-1415

[57] ZHOU, Z. H.; WU, J. X.; TANG W.; CHEN, Z. Q. – "**Combining regression estimators: GA-based selective neural network ensemble.**" em International Journal of Computational Intelligence and Applications, 2001. p. 341-356

[58] ZHOU, Z.H.; WU, J.X.; JIANG, Y.; CHEN, S. F. – "**Genetic algorithm based selective neural network ensemble.**" em Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI'01), vol.2, Seattle, WA, 2001, pág. 797-802.

[59] GIROSI, F., JONES, M., POGGIO, T. – "**Regularization Theory and Neural Networks Architectures.**" em Neural Computation, vol. 7, no. 2, 1995, pág. 219-269.

[60] RUMELHART, D. E., MCCLELLAND, J. L. AND THE PDP RESEARCH GROUP – "**Parallel Distributed Processing: Explorations in the Microstructure of Cognition**", Volume 1: Foundations, The MIT Press, 1986.