PREVISÃO DE ARRECADAÇÃO DO ICMS ATRAVÉS DE REDES NEURAIS NO BRASIL

JUAN CAMILO SANTANA CONTRERAS

Orientador: Prof. Dr. Francisco Cribari Neto

Área de Concentração: Estatística Aplicada

Dissertação submetida como requerimento parcial para obtenção do grau de Mestre em Estatística pela Universidade Federal de Pernambuco

Recife, dezembro de 2005

Universidade Federal de Pernambuco
Mestrado em Estatística

02 de dezembro de 2005

(data)

Nós recomendamos que a dissertação de mestrado de autoria de

Juan Camilo Santana Contreras

intitulada

Previsão da Arrecadação do ICMS, através de Redes Neurais no Brasil

seja aceita como cumprimento parcial dos requerimentos para o grau de Mestre em Estatística.

Klaus Leite Pinto Vasconcellos

Coordenador da Pós-Graduação em Estatística

Prof. Klaus L. P. Vasconcellos
Coordenador da
Pós-graduação em
Estatística, UFPE

Banca Examinadora:

Francisco Cribari Neto

Francisco Cribari Neto — orientador

Valderio Anselmo Reisen (UFES)

Klaus Leite P. Vasconcellos

Klaus Leite Pinto Vasconcellos

Este documento será anexado à versão final da dissertação.

A Sara

AGRADECIMENTOS

A Deus, a força vital e inspiradora que me fez possível superar todas as dificuldades que se apresentaram ao longo de minha vida.

À minha mãe, cujo amor e carinho estiveram perto de mim durante momentos difíceis apesar da distância.

Ao Dr. Francisco Cribari Neto pelas sugestões, orientação incondicional e comprometida ao longo desta dissertação.

A meus colegas colombianos pela amizade.

Ao professor Dr. Luis Alberto López da Universidade Nacional de Colômbia pelos seus conselhos.

Aos colegas brasileiros da Universidade Federal de Pernambuco pelo recebimento, ajuda e atenção incondicional, ao longo dos dois últimos anos.

A Valéria, pela sua presteza em resolver problemas relacionados com a Pós-Graduação.

Aos participantes da banca examinadora, pelas sugestões.

Ao `CNPq`, pelo apoio financieiro.

RESUMO

A presente dissertação se centra na temática de produção de previsões de valores futuros de arrecadações tributárias. Em particular, busca-se avaliar a utilidade de métodos de previsão baseados em redes neurais. Os resultados obtidos para arrecadações do *ICMS* nacional e de três estados (Pernambuco, Rio de Janeiro e São Paulo) mostram que previsões obtidas por redes neurais podem ser mais precisas do que aquelas fornecidas por metodologias de previsão mais tradicionais como o alisamento exponencial e o método de Box e Jenkins. Os resultados revelam ainda que combinações de previsões que incluem redes neurais tendem a alcançar maior precisão do que combinações que não incluem redes neurais.

ABSTRACT

The chief goal of this thesis is to evaluate the usefulness of neural network methods in predicting sale taxes revenues in Brazil (nationwide and for three different states). The results show that neural networks based forecasts can be considerably more accurate than forecasts obtained by using exponential smoothing and *SARIMA* methods. It is also shown that the inclusion of neural networks forecasts in the combination of individual forecasts leads to higher forecasting accuracy.

Índice

CAPÍTULO 1

INTRODUÇÃO

As três esferas de governo – união, estados e municípios – possuem competência impositiva para cobrar tributos. Os tributos de maior destaque incidem sobre a renda, sobre o consumo e sobre o patrimônio. A arrecadação do *ICMS*, Imposto sobre Operações Relativas à Circulação de Mercadorias e sobre Prestações de Serviços de Transporte Interestadual e Intermunicipal e de Comunicação, compete principalmente aos estados e é um dos impostos mais importantes, juntamente com o *IR* (Imposto de Renda) e com o *IPI* (Imposto sobre Produtos Industrializados). O conhecimento sobre o padrão evolutivo desses tributos, dentro do contexto da atividade econômica brasileira, é útil para o planejamento e para a tomada de decisões por parte de governantes. Neste sentido, a geração de previsões sobre arrecadação de impostos desempenha papel importante dentro da política econômica.

Recentemente, estratégias de modelagem alternativas às usualmente adotadas têm sido consideradas quando o objetivo reside na previsão de valores futuros de séries econométricas. Em particular, modelos não-paramétricos como redes neurais têm

recebido cada vez mais atenção por parte de usuários; veja Zhang, Patuwo & Hu (1998) e Medeiros, Teräsvirta & Rech (2002).

Na presente dissertação, uma estratégia alternativa de previsão utilizando redes neurais será considerada com vistas à elaboração de prognósticos relativos à arrecadação do *ICMS* agregado a nível nacional, como também relativos às arrecadações do *ICMS* em São Paulo, Rio de Janeiro e Pernambuco. Metodologias tradicionais, que possuem boa capacidade preditiva, tais como algoritmos de alisamento exponencial, método de Box e Jenkins e combinação de previsões, serão estudadas conjuntamente com redes neurais. A análise de cada uma das séries será feita com base no período que se estende de julho de 1994 a dezembro de 2004. Em primeira análise, o interesse recairá sobre a previsão de dezembro de 2004 e, em segunda análise, o período de julho a dezembro de 2004 servirá como base para avaliação de previsões. Na primeira análise, previsões relativas ao período que se estende de janeiro a março de 2005 também serão produzidas e avaliadas.

A plataforma computacional a ser utilizada é o software `R`, baseado na linguagem de programação `S`, veja Cribari-Neto & Zarkos (1999) e Venables & Ripley (2002); este programa pode ser obtido da página `http://www.r-project.org` para diferentes sistemas operacionais. Adicionalmente, será utilizado o programa `TRAMO-SEATS`, cuja sigla corresponde a "Time Series Regression with ARIMA Noise, Missing Observations and Outliers" e "Signal Extraction in ARIMA Time Series", desenvolvido por Víctor Gómez e Agustín Maravall, veja Goméz & Maravall (1996). O software `TRAMO-SEATS` é utilizado por muitas instituições internacionais, bancos, universidades, centros de pesquisa e companhias privadas devido à sua flexibilidade no que tange à interpolação de valores perdidos, identificação e correção por outliers e extração de componentes não-observáveis. Através da página `http://www.bde.es` do Banco de Espanha a versão para o sistema operacional Windows pode ser obtida gratuitamente. Outros programas econométricos gratuitos como por exemplo, o `GRETL`, possui um módulo asssociado ao `TRAMO-SEATS`. O software `GRETL` pode ser obtido gratuitamente para

diferentes sistemas operacionais através da página `http://gretl.sourceforge.net`.

A dissertação encontra-se organizada em capítulos, como descrito a seguir. No capítulo 2, conceitos gerais sobre séries temporais, como processo estocástico, estacionariedade, sazonalidade, função de autocorrelação e de autocorrelação parcial serão introduzidos. Adicionalmente, serão descritas as metodologias de alisamento exponencial, Box e Jenkins e de combinação de previsões, como também serão discutidas a descomposição das séries em componentes não-observáveis e a identificação e correção de outliers. No capítulo 3, a rede neural perceptron multicamadas será descrita, sendo ela uma das arquiteturas de redes neurais mais amplamente utilizadas, por sua capacidade para mapear qualquer função contínua a um valor real, veja Hornik, Stinchcombe & White (1989). As principais características do perceptron multicamadas serão discutidas; igualmente, descrições sobre o algoritmo de aprendizagem e conceitos relacionados com o parâmetro de aprendizado, momento, camadas ocultas, neurônios, função de ativação e *epochs* serão apresentadas. O capítulo 4 apresenta uma análise descritiva preliminar sobre cada uma das séries e os resultados de previsão obtidos através de cada uma das metodologias e análises já referenciadas. Por último, o capítulo 5 contém um resumo das conclusões.

CAPÍTULO 2

ESTRATÉGIAS USUAIS DE PREVISÃO

Neste capítulo é feita uma apresentação dos conceitos introdutórios relacionados à previsão e à modelagem de séries temporais.

2.1 Conceitos Básicos

Muitas informações das características econômicas e sociais, tanto de indivíduos quanto de empresas e países, são freqüentemente coletadas com fins de análise para posteriormente servir de base para atividades de planejamento e tomada de decisões. Uma série temporal é a coleção de valores assumidos por uma variável aleatória ao longo de tempo. Para entender o que é uma série temporal dentro do contexto de processos estocásticos, faz-se necessário definir o que sejam tais processos.

Definição 1. Um processo estocástico é uma família de variáveis aleatórias definidas em um mesmo espaço de probabilidade, associada a um conjunto índice de números reais, tal que a cada elemento do conjunto corresponda uma e só uma

variável aleatória. Denotamos um processo estocástico por $\{Z(t), t \in T\}$, em que T é o conjunto índice e $Z(t)$ é a variável aleatória relacionada com o elemento t de T.

Quando T é um intervalo de números reais, fechado ou aberto, o processo estocástico é dito ser contínuo; quando T é um conjunto finito ou enumerável, o processo é dito ser discreto.

Uma série temporal é uma sucessão de observações geradas por um processo estocástico cujo índice é relacionado ao tempo. Assim como existem processos estocásticos discretos e contínuos, existem também séries temporais discretas e contínuas. Em particular, se as observações de uma série temporal discreta são realizadas nos instantes $t_1, t_2, \ldots, t_N$, o respectivo processo estocástico é denotado por $\{Z(t_1), Z(t_2), \ldots, Z(t_N)\}$ ou simplesmente por $\{Z_1, Z_2, \ldots, Z_N\}$.

Em quase todas análises estatísticas, exceto na análise de séries temporais é comum supor que as observações provêm de variáveis aleatórias independentes, de forma tal que com o único conhecimento das funções de densidade individuais é possível obter a função de densidade conjunta. Em contraste, no caso de séries temporais há toda uma estrutura de correlação entre as observações; assim, não é possível se obter a função de densidade conjunta tão diretamente.

Um conceito importante, relacionado à estabilidade dos dois primeiros momentos de um processo estocástico é o de estacionariedade (fraca).

Definição 2. Um processo estocástico $\{Z(t), t \in T\}$ é dito ser *fracamente estacionário* ou estacionário de segunda ordem (ou em sentido amplo) se

- $E\{Z(t)\} = \mu$, constante para todo $t \in T$.
- $\gamma_k = \gamma(t, t+k) = \text{Cov}(Z(t), Z(t+k))$ é uma função apenas de $|k|$.

Alternativamente, um processo é dito ser estritamente estacionário ou fortemente estacionário se a função de densidade conjunta de um conjunto qualquer de variáveis é invariante no tempo; veja Box, Jenkins & Reinsel (1994). Por conseguinte, estacionariedade estrita implica estacionariedade fraca (quando os dois primeiros momentos da

série existem), porém o inverso não é válido, exceto quando a distribuição for definida pelos dois primeiros momentos, como por exemplo a distribuição normal multivariada.

Sobre estacionariedade, a função de autocovariancias possui as seguintes propriedades:

i. $\gamma_0 > 0$;

ii. $\gamma_k = \gamma_{-k},\ \forall k$;

iii. $|\gamma_k| \leq \gamma_0,\ \forall k$.

Para evitar a dependência sobre unidades de medida, é preferível trabalhar com autocorrelações. A k-ésima autocorrelação é definida como

$$\rho_k = \frac{\gamma_k}{\gamma_0},\ k = 0, \pm 1, \pm 2, \ldots\ .$$

Um outro instrumento é utilizado para facilitar a escolha de um modelo que descreva o processo estocástico estacionário: a função de autocorrelação parcial. A k-ésima autocorrelação parcial (ϕ_{kk}) é obtida através de

$$\boldsymbol{P}_k \boldsymbol{\phi}_k = \boldsymbol{\rho}_{(k)},$$

em que $\boldsymbol{P}_k$ é a matriz de autocorrelações de ordem k, $\boldsymbol{\phi}_k = (\phi_{k1}, \phi_{k2}, \ldots, \phi_{kk})^\top$ e $\boldsymbol{\rho}_{(k)} = (\rho_1, \rho_2, \ldots, \rho_k)^\top$.

2.2 Alisamento Exponencial

Uma estratégia de previsão amplamente utilizada devido à sua simplicidade e eficiência computacional baseia-se no uso de algoritmos de alisamento exponencial. Neste contexto, objetiva-se extrair da série algum tipo de padrão que constituirá o elemento principal para explicar o comportamento histórico e para prever o futuro da série. Para detalhes sobre algoritmos de alisamento exponencial, veja Morettin & Toloi (2004) e Montgomery & Johnson (1976).

2.2.1 Alisamento exponencial simples

Seja a série temporal $Z_1, Z_2, \ldots, Z_N$. O método de alisamento exponencial toma o nível da série como

$$\bar{Z}_t = \alpha Z_t + (1-\alpha)\bar{Z}_{t-1}, \quad \bar{Z}_0 = Z_1, \quad t = 1, \ldots, N, \tag{2.1}$$

ou

$$\bar{Z}_t = \alpha \sum_{k=0}^{t-1} (1-\alpha)^k Z_{t-k} + (1-\alpha)^t \bar{Z}_0, \quad t = 1, \ldots, N. \tag{2.2}$$

Aqui, $\bar{Z}_t$ é denominado valor exponencialmente suavizado e $0 < \alpha < 1$ é a constante de suavização, usualmente desconhecida.

A previsão de Z_{t+h}, sendo h o horizonte de previsão, é dada pelo último valor exponencialmente suavizado, isto é,

$$\hat{Z}_t(h) = \bar{Z}_t, \quad \forall h > 0, \tag{2.3}$$

$$\hat{Z}_t(h) = \alpha Z_t + (1-\alpha)\hat{Z}_{t-1}(h+1), \quad \forall h > 0, \tag{2.4}$$

Quando o valor de α é próximo de um, pesos altos são dados às observações mais recentes da série na previsão.

O método de alisamento exponencial é flexível e de fácil entendimento. Uma desvantagem contudo é a dificultade em determinar o valor mais apropriado da constante de suavização.

2.2.2 Alisamento exponencial de Holt

Este método tem a vantagem de suavizar não só o nível da série mas também sua tendência. Os valores do nível e da tendência da série, no instante t, são dados por

$$\bar{Z}_t = \alpha Z_t + (1-\alpha)(\bar{Z}_{t-1} + \hat{T}_{t-1}), \quad 0 < \alpha < 1, \quad t = 2, \ldots, N, \tag{2.5}$$

$$\hat{T}_t = \beta(\bar{Z}_t - \bar{Z}_{t-1}) + (1-\beta)\hat{T}_{t-1}, \quad 0 < \beta < 1, \quad t = 2, \ldots, N, \tag{2.6}$$

respectivamente, com α e β sendo constantes de suavização.

A previsão de Z_{t+h} no instante t é dada por

$$\hat{Z}_t(h) = \bar{Z}_t + h\hat{T}_t, \ \forall h > 0, \tag{2.7}$$

isto é, a previsão é feita adicionando-se ao nível estimado ($\bar{Z}_t$) a tendência multiplicada pelo número de passos à frente que se deseja prever (h).

2.2.3 Alisamento exponencial de Holt-Winters

Dependendo das características da série, dois métodos são utilizados a partir de três equações de suavização diferentes, associadas a cada uma das componentes da série: nível, tendência e sazonalidade.

a. Série Sazonal Multiplicativa. Considere uma série sazonal com período s. A variante mais usual do método Holt-Winters considera o fator sazonal F_t como sendo multiplicativo, enquanto a tendência T_t permanece aditiva, μ_t representa o nível médio da série e a_t o erro no instante t, isto é,

$$Z_t = \mu_t F_t + T_t + a_t, \ t = 1, 2, \ldots, N. \tag{2.8}$$

As três equações de suavização são dadas por

$$\bar{Z}_t = \alpha \left(\frac{Z_t}{\hat{F}_{t-s}} \right) + (1-\alpha)(\bar{Z}_{t-1} + \hat{T}_{t-1}), \quad 0 < \alpha < 1, \tag{2.9}$$

$$\hat{T}_t = \beta(\bar{Z}_t - \bar{Z}_{t-1}) + (1-\beta)\hat{T}_{t-1}, \quad 0 < \beta < 1, \tag{2.10}$$

$$\hat{F}_t = \gamma \left(\frac{Z_t}{\bar{Z}_t} \right) + (1-\gamma)\hat{F}_{t-s}, \quad 0 < \gamma < 1, \tag{2.11}$$

$t = s+1, \ldots, N$, representando as estimativas do nível, da tendência e do fator sazonal, respectivamente; α, β e γ são as constantes de suavização. Os valores previstos da série são dados pelas equações

$$\begin{aligned} \hat{Z}_t(h) &= (\bar{Z}_t + h\hat{T}_t)\hat{F}_{t+h-s}, \quad t = 1, \ldots, s, \\ \hat{Z}_t(h) &= (\bar{Z}_t + h\hat{T}_t)\hat{F}_{t+h-2s}, \quad t = s+1, \ldots, 2s, \\ &\vdots \quad \vdots \end{aligned} \tag{2.12}$$

em que $\bar{Z}_t$, $\hat{T}_t$ e $\hat{F}_t$ são obtidas das equações (2.9), (2.10) e (2.11), respectivamente. Esquemas de atualização das previsões quando é obtida uma nova observação podem ser encontrados em Morettin & Toloi (2004). Os valores iniciais das equações de recorrência são calculados por meio das seguintes equações:

$$\hat{F}_j = \frac{Z_j}{(\frac{1}{s})\sum_{k=1}^{s} Z_k}, \quad j = 1, 2, \ldots, s,$$

$$\bar{Z}_s = \frac{1}{s}\sum_{k=1}^{s} Z_k, \quad \hat{T}_s = 0.$$

b. Série Sazonal Aditiva. O procedimento anterior pode ser modificado para lidar com situações onde o fator sazonal é aditivo, i.e.,

$$Z_t = \mu_t + T_t + F_t + a_t, \ t = 1, 2, \ldots, N. \tag{2.13}$$

As estimativas do fator sazonal, nível e tendência da série são dadas por

$$\bar{Z}_t = \alpha(Z_t - \hat{F}_{t-s}) + (1-\alpha)(\bar{Z}_{t-1} + \hat{T}_{t-1}), \quad 0 < \alpha < 1, \tag{2.14}$$

$$\hat{T}_t = \beta(\bar{Z}_t - \bar{Z}_{t-1}) + (1-\beta)\hat{T}_{t-1}, \quad 0 < \beta < 1, \tag{2.15}$$

$$\hat{F}_t = \gamma(Z_t - \bar{Z}_t) + (1-\gamma)\hat{F}_{t-s}, \quad 0 < \gamma < 1, \tag{2.16}$$

α, β e γ sendo as constantes de suavização. Neste caso, as equações de previsão são dadas por

$$\begin{aligned} \hat{Z}_t(h) &= \bar{Z}_t + h\hat{T}_t + \hat{F}_{t+h-s}, \quad t = 1, \ldots, s, \\ \hat{Z}_t(h) &= \bar{Z}_t + h\hat{T}_t + \hat{F}_{t+h-2s}, \quad t = s+1, \ldots, 2s, \\ &\vdots \quad \vdots \end{aligned} \tag{2.17}$$

com $\bar{Z}_t$, $\hat{F}_t$ e $\hat{T}_t$ definidas com antes. Igualmente, atualizações das previsões dada a entrada de uma nova observação podem ser calculadas como descrito em Morettin & Toloi (2004).

Para esses dois algoritmos as desvantagems são as mesmas que para o de alisamento exponencial simples e de Holt: a dificultade de determinar os valores mais apropriados das constantes de suavização. Da mesma forma, o estudo de propriedades estatísticas, tais como média e variância das previsões é difícil. Em geral, a determinação das constantes é feita minimizando a soma de quadrados dos erros de previsão um passo à frente; veja Granger & Newbold (1977).

2.3 Metodologia de Box e Jenkins

A metodologia proposta em Box & Jenkins (1976) e, mais recentemente, em Box et al. (1994) apresenta a possibilidade de modelagem de séries temporais não–estacionárias através de diferenças sucessivas. Se $\{Z_t\}$ é o processo original, que não é estacionário devido a uma tendência não-determinística, é muitas vezes possível obter um processo estacionário $\{W_t\}$ tomando

$$W_t = \Delta^d Z_t \quad d \geq 1,$$

em que Δ^d representa o operador diferença, definido como

$$\Delta^d = (1 - B)^d,$$

B sendo o operador de retardo, i.e., para algum k inteiro maior que zero, $B^k Z_t = Z_{t-k}$. Quando isso ocorre, dizemos que Z_t é integrável de ordem d, denotado $Z_t \sim I(d)$.

A estratégia adotada na construção do modelo é baseada em um processo iterativo que está determinado pelos seguintes passos: identificação, estimação e validação.

2.3.1 Modelo ARIMA

Se W_t é uma série temporal estacionária de média zero, um modelo que representa $\{W_t\}$ é dado por

$$\phi(B)W_t = \theta(B)a_t, \tag{2.18}$$

ou

$$\phi(B)\Delta^d Z_t = \theta(B)a_t, \tag{2.19}$$

em que $\phi(B)$ e $\theta(B)$ são dados por

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p, \tag{2.20}$$

$$\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \cdots - \theta_q B^q, \tag{2.21}$$

p e q sendo constantes a serem determinadas. O supostos usualmente considerados são que as equações $\phi(z) = 0$ e $\theta(z) = 0$ devem ter todas as raízes (não comuns) fora do círculo unitário para cumprir as condições de estacionariedade e invertibilidade, respectivamente, e $\{a_t\}$ deve ser um ruído branco de média zero e variância σ_a^2. O modelo (2.19) é chamado auto-regressivo integrado de médias móveis de ordem (p, d, q), sendo denotado por ARIMA(p, d, q).

Para o processo de identificação do modelo, p e q são determinados através da análise da função de autocorrelação e da função de autocorrelação parcial; uma vez feito isso, os parâmetros $\{\phi_i\}$, $\{\theta_j\}$ e σ_a^2 são estimados utilizando algum algoritmo iterativo; veja Box et al. (1994). Neste processo é possível que mais de um modelo seja identificado, havendo vários procedimentos na literatura para a escolha do melhor modelo, baseados na função penalizadora; veja Morettin & Toloi (2004). Os mais utilizados são o AIC (Critério de Informação de Akaike) e o BIC (Critério de Informação Bayesiano), dados por

$$AIC(p, q) = N\ln(\hat{\sigma}_a^2) + 2(p + q), \tag{2.22}$$

$$BIC(p, q) = \ln(\hat{\sigma}_a^2) + (p + q)\frac{\ln N}{N}, \tag{2.23}$$

respectivamente, em que N é o número de observações da série temporal e $\hat{\sigma}_a^2$ é a variância residual estimada do modelo ARIMA(p, d, q). O modelo escolhido será aquele que minimize (2.22) ou (2.23), a depender do critério escolhido.

O último passo na determinação do modelo é a análise de diagnóstico. Neste caso, os resíduos são utilizados para se determinar se o modelo ajustado representa suficientemente bem os dados. Se o modelo for adequado, espera-se que os resíduos sejam aproximadamente não correlacionados. Utilizando as autocorrelações dos resíduos, $\hat{r}_k$, com variância aproximada $1/n$, onde $n = N - d$, é possível obter intervalos de confiança assintóticos; veja Morettin & Toloi (2004). Adicionalmente, a estatística de Ljung e Box (Ljung & Box, 1978) pode ser utilizada para testar a hipótese de não-autocorrelação. Se o modelo for apropriado, a estatística

$$Q(K) = n(n+2)\sum_{j=1}^{K}\frac{\hat{r}_j^2}{(n-j)}.$$

terá, aproximadamente, distribuição χ^2 com $K-p-q$ graus de liberdade. A hipótese de ruído branco será rejeitada para valores grandes de $Q(K)$. Igualmente, pode-se testar a normalidade dos erros utilizando a estatística de Bowman-Shenton; veja Doornik & Hansen (1994).

O interesse, após os três passos da construção do modelo terem sido completados, reside na previsão de $Z_{t+h}, h \geq 1$, supondo que dispomos das observações $\ldots, Z_{t-2}, Z_{t-1}, Z_t$, até o instante t. Esta previsão de origem t e horizonte h será denotada por $\hat{Z}_t(h)$. A previsão de erro quadrático médio mínimo é a esperança condicional de Z_{t+h}, dadas as observações passadas da série. A previsão de $\hat{Z}_t(h)$ é dada por

$$\hat{Z}_{t+h} = \sum_{j=0}^{\infty}\varphi_{_{h+j}}a_{_{t+h-j}},$$

em que $\varphi(B) = 1 + \varphi_1 B + \varphi_2 B^2 + \ldots$ e $\{\varphi_i\}$ é obtido da equação $\phi(B)\varphi(B) = \theta(B)$. O erro de previsão é $e_t(h) = a_{t+h} + \varphi_1 a_{t+h-1} + \cdots + \varphi_{h-1}a_{t+1}$, com variância $\mathrm{Var}(e_t(h)) = \sigma_a^2\sum_{j=0}^{h-1}\varphi_j^2$. Adicionalmente, previsões atualizadas podem ser

obtidas quando uma nova observação é coletada; para uma discussão sobre o tema, veja Morettin & Toloi (2004).

2.3.2 Modelo SARIMA

Quando um padrão dinâmico de sazonalidade estocástica está presente na série, este pode ser modelado com uma pequena modificação sobre o modelo (2.19), da forma seguinte:

$$\phi(B)\Phi(B^s)\Delta^d\Delta_s^D Z_t = \theta(B)\Theta(B^s)a_t, \tag{2.24}$$

em que o processo $W_t = \Delta^d\Delta_s^D Z_t$ é estacionário, enquanto que d, D e s representam o número de diferenças simples e sazonais feitas sobre Z_t e o período da sazonalidade, respectivamente. Além disso, $\Delta_s = (1 - B^s)$, $\Delta_s^D = (1 - B^s)^D$ e os polinômios associados à componente sazonal em (2.24) são dados por

$$\Phi(B^s) = 1 - \Phi_1 B^s - \cdots - \Phi_P B^{sP}, \tag{2.25}$$

$$\Theta(B^s) = 1 - \Theta_1 B^s - \cdots - \Theta_Q B^{sQ}. \tag{2.26}$$

O modelo (2.24) é chamado modelo auto-regressivo integrado de médias móveis sazonal e denotado por SARIMA$(p, d, q) \times (P, D, Q)_s$.

Os demais polinômios são os definidos em (2.20) e (2.21). Não há, em princípio, nenhuma dificuldade adicional na identificação, estimação e verificação de modelos sazonais. É necessário tomar a diferença da série com respeito a Δ e Δ_s, a fim de produzir estacionariedade. Em seguida, inspecionamos as funções de autocorrelação e autocorrelação parcial amostrais para determinar os valores de p e q nos 'lags' simples, e para obter P e Q nos 'lags' sazonais, obtendo um modelo tentativo. Em seguida, estimamos os parâmetros identificados por máxima verossimilhança. Finalmente, para verificar se o modelo proposto é adequado, utilizamos o teste de Ljung-Box.

2.3.3 Correção por outliers

Os valores de uma série temporal podem ser afetados por eventos externos e atípicos, que podem chegar a ter efeito notável sobre a identificação e previsão do modelo, motivo pelo qual é muitas vezes preciso utilizar uma metodologia que permita identificá-los. As observações que são inconsistentes com o comportamento geral da série histórica são denominadas *valores atípicos* ou *outliers*. Para controlar o efeito que essas observações espúrias possam exercer sobre o modelo utiliza-se a análise de intervenção; para uma discussão mais detalhada sobre o tema veja Guerrero (1991) e Morettin & Toloi (2004). Neste ponto, há que se fazer a distinção entre variáveis de intervenção e outliers; veja Box & Tiao (1975). Quando há informação a priori acerca de um evento especial que originou observações anormais, tal como a data e seu provável efeito, este deve ser capturado através da análise de intervenção. Outliers, em contraste, representam anomalias nas observações para as quais não há informação a priori sobre datas de ocorrência ou sobre os padrões dinâmicos de seus efeitos. A correção automática de outliers requer conhecimento a priori sobre o padrão dinâmico que será considerado e um procedimento sistemático de detecção.

O objetivo principal da correção de outliers é modificar os dados para que não haja informações distoantes. Quatro tipos de outliers têm sido considerados na literatura estatística, veja Ljung & Box (1986), Chen & Liu (1993), Peña (2001): atípico aditivo (AO - additive outlier), atípico de innovação (IO-innovation outlier), mudança de nível (LS - level shift) e mudança transitória (TC - transitory change). Assumindo que a série observada tem k outliers, seu efeito combinado pode ser expresso como

$$Z_t^* = \sum_{j=1}^{k} \xi_j(B)\omega_j I_t^{(\tau_j)} + Z_t,$$

em que Z_t^* é a série contaminada, Z_t a série que segue, supondo por simplicidade na notação, o modelo (2.19), ω_j é o impacto inicial do outlier no tempo $t = \tau_j$, $I_t^{(\tau_j)}$ é uma variável dummy que é igual a 1 se $t = \tau_j$ e igual a 0 caso contrário, e $\xi_j(B)$ determina a dinâmica do outlier que acontece em $t = \tau_j$, de acordo com o seguinte

esquema:

$$\text{AO} : \xi_j(B) = 1, \tag{2.27}$$

$$\text{LS} : \xi_j(B) = 1/(1-B), \tag{2.28}$$

$$\text{TC} : \xi_j(B) = 1/(1-\delta B),\ 0 < \delta < 1, \tag{2.29}$$

$$\text{IO} : \xi_j(B) = \hat{\theta}(B)/\hat{\phi}(B). \tag{2.30}$$

Estes quatro tipos de outliers afetam a série observada de maneiras diferentes: os efeitos gerados pelos tipos AO, LS e TC não dependem do modelo ARIMA ajustado, o que não ocorre com o tipo IO, já que esse utiliza a representação em médias móveis do modelo ARIMA ajustado para gerar o efeito sobre a série observada; veja Kaiser & Maravall (2001). AO e TC geram um efeito transitório sobre a série, enquanto que LS tem efeito permanente. Todavía, os efeitos gerados por AO, TC e LS são limitados, diferentemente do que acontece com IO. Chen & Liu (1993) sugerem uma aproximação para detecção e correção automática de outliers que conduz a procedimentos confiáveis e eficientes; veja Gómez & Maravall (2001).

2.3.4 Extração das componentes não-observáveis

Suponha que a série de interesse foi modificada através de um esquema de pré-processamento de forma a minorar os efeitos exercidos pelos outliers e que há interesse em componentes não-observáveis, e.g., tendência e sazonalidade. A suposição preliminar é que o sinal de interesse pode ser extraído de Z_t de forma aditiva:

$$Z_t = s_t + n_t,$$

em que n_t representa a componente de não-sinais e s_t é o sinal de interesse; se o sinal for a série ajustada sazonalmente, então n_t será a componente sazonal, ou se o sinal é a tendência, um ruído branco ou componente transitório pode ser incluído em n_t.

A descomposição também pode ser multiplicativa: $Z_t = s_t n_t$; porém, tomando-se a transformação logaritmo, volta-se ao caso aditivo. Além disso, as duas componentes,

s_t e n_t, seguem os seguintes processos estocásticos lineares:

$$\phi_s(B)s_t = \theta_s(B)a_{st}, \tag{2.31}$$

$$\phi_n(B)n_t = \theta_n(B)a_{nt}. \tag{2.32}$$

Com relação a (2.31) e (2.32), as seguintes suposições serão feitas:

- Os processos a_{st} e a_{nt} são mutuamente independentes, ruídos brancos de média zero e variâncias σ_s^2 e σ_n^2, respectivamente.
- Os polinômios $\phi_s(B)$ e $\phi_n(B)$ são primos, isto é, o maior grau associado a um polinômio deve ser primo com o maior grau do outro polinômio.
- Os polinômios $\theta_s(B)$ e $\theta_n(B)$ não têm raízes unitárias, simultaneamente. Esta suposição estabelece uma condição suficiente de invertibilidade sobre a série Z_t.

Assumindo o modelo ARIMA definido por simplicidade em (2.19) para a série observada Z_t, os polinômios das partes auto-regressiva e de médias móveis (invertível) podem ser expressos em função de (2.31) e (2.32) como

$$\phi(B) = \phi_s(B)\phi_n(B),$$

$$\theta(B)a_t = \phi_n(B)\theta_s(B)a_{st} + \phi_s(B)\theta_n(B)a_{nt},$$

Tendo, então, o conjunto de observações $Z_1, Z_2, \ldots, Z_t$, a idéia geral é obter estimadores de mínimo erro quadrático (`MMSE`) e previsões de s_t e n_t. Além disso, é possível obter erros-padrão das previsões e dos estimadores; uma discussão sobre o tema encontra-se em Maravall & Kaiser (2000).

2.3.5 Combinação de previsões

Tem sido sugerido na literatura que combinações lineares de duas ou mais previsões, proporcionadas por diferentes metodologias, podem apresentar melhores resultados em comparação com as previsões individuais, veja Barnard (1963) e Hendry & Clements (2004).

Se Z_t é a série a ser predita no instante t e $f_1, f_2, \ldots, f_k$ são as previsões individuais através de k diferentes metodologias, então a combinação das previsões no prognóstico da série no tempo t será dada por

$$\sum_{j=1}^{k} \psi_j f_j,$$

em que $0 \leq \psi_j \leq 1$ e $\sum_{j=1}^{k} \psi_j = 1$. Cada ψ_j será definido como sendo inversamente proporcional ao erro de previsão da metodologia correspondente. Cumpre ressaltar, todavia, que há propostas diferentes para escolha desses pesos; veja Newbold & Granger (1974).

CAPÍTULO 3

REDES NEURAIS ARTIFICIAIS

3.1 Fundamentos Biológicos das Redes Neurais

O elemento estrutural e funcional mais importante no sistema de comunicação neural é a célula nervosa ou neurônio, através da qual são transmitidas informações. As informações são transmitidas entre os distintos neurônios através de propagações, formando redes, nas quais se produzem e se armazenam informações. Além disso, uma parte dos neurônios mantém relação com os receptores de informação, através dos quais chegam comunicações procedentes do exterior ou interior do organismo até as redes neurais. Outra parte conduz as informações, na forma de ordens, para os sistemas que as executam. Uma das prolongações dos neurônios chamada *axônio* é encarregada da condução de impulsos e a uma distância mais ou menos grande da origem, ramifica-se e forma os botões terminais que ficam em contato com outros neurônios ou com células executoras, mas sem chegar a se fundir com elas. Essa zona de contato é chamada de *sinapse*.

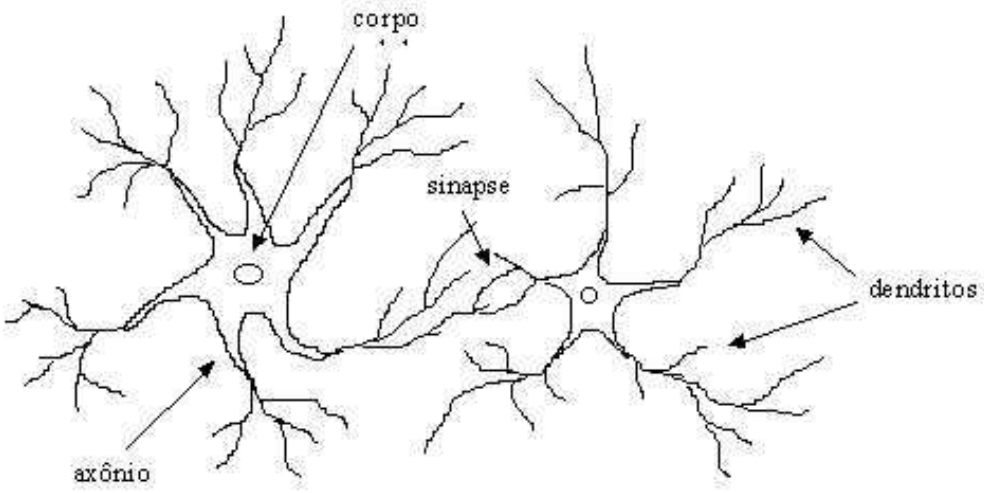

Figura 3.1: Descrição de uma célula nervosa típica.

A missão dos neurônios abrange cinco funções parciais:

- Os neurônios captam a informação que chega a eles na forma de impulsos procedentes de outros neurônios ou receptores.
- Integram-se em um código especial próprio da célula.
- Transmitem a informação codificada na forma de freqüência de impulsos através dos seus axônios.
- Nos seus terminais, transmitem os impulsos aos neurônios subseqüentes ou às células executoras.

Um diagrama de uma célula nervosa típica é apresentado na figura 3.1 em que o neurônio consta de um corpo celular e de um núcleo, semelhante ao resto das células do organismo, porém conta com alguns elementos específicos. Em primeiro lugar, está o axônio, que é uma ramificação da saída do neurônio. Através dele propagam-se uma série de impulsos eletro-químicos. Adicionalmente, o neurônio conta com um grande número de ramificações de entrada, os dendritos, que propagam o sinal até

o seu interior. O funcionamento é como segue. A sinapse pega a informação eletro-química procedente das células vizinhas às que a célula em questão está conetada; esta informação chega ao núcleo, onde é processada até gerar uma resposta que é propagada pelo axônio. Posteriormente, o sinal propagado pelo axônio ramifica-se e chega aos dendritos de outras células através das *sinapses*. As sinapses são os elementos de união entre axônio e dendritos. O funcionamento, em geral, será o de uma grande malha que propaga sinais eletro-químicos de umas células a outras.

A conectividade entre as células do cérebro é muito elevada. Calcula-se que em cada cérebro existem em torno de 100 bilhões de neurônios, conectados cada um deles com aproximadamente outros 10.000, i.e., cada atividade neural afeta outros 10.000 neurônios, formando assim, uma rede de tamanho enorme. Um dispositivo de tal complexidade não pode ser duplicado com a tecnologia atual. Nosso comportamento, inteligente ou não, segue um esquema deste tipo. São estes mecanismos que os modelos artificiais que têm aparecido ao longo da história das redes neurais tentam incorporar.

3.2 Modelo Computacional

A grande diferença entre um programa baseado em redes neurais e os programas computacionais convencionais é que aqueles *modificam* em certa medida a informação de entrada para obter uma saída ou resposta. Não se deve simplesmente aplicar cegamente um algoritmo; de fato, o processo de manipulação da informação recebida depende das diferentes características, tanto estruturais como funcionais da rede. Existem diversos modelos de redes neurais que seguem filosofias de desenhos, regras de aprendizado e funções de construção das respostas muito diferentes. Uma primeira classificação é feita em função do percurso da informação dentro da rede, sendo conhecidas as redes alimentadas para frente (*feedforward*) e as que possuem retropropagação (*retropropagation*).

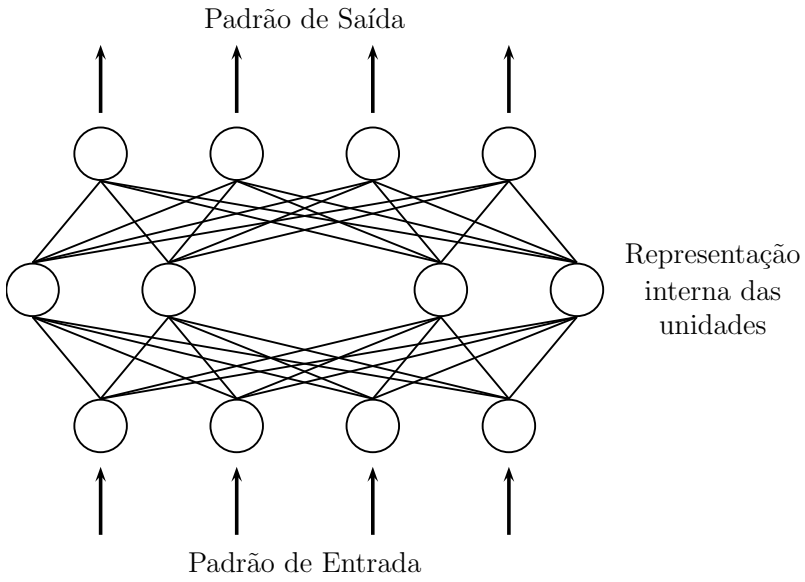

Figura 3.2: Estrutura geral de uma rede neural.

3.2.1 Estrutura básica da rede

A maneira como os neurônios se conectam é denominada padrão de conectividade ou arquitetura da rede. A estrutura básica de interconexão entre os neurônios é a de rede multicamada, apresentada na figura 3.2. Trata-se de uma estrutura típica de implementação do paradigma conhecido como *retropropagação* que será descrito mais adiante. No primeiro nível há as células de entrada; estas unidades recebem os valores de uns padrões representados como vetores, que são utilizados como entradas da rede. A seguir, há uma série de camadas intermediárias, ditas ocultas, cujas unidades respondem a características particulares dos padrões da entrada. Pode haver uma ou mais camadas ocultas. O último nível ou camada é a saída, que corresponde à resposta de interesse. Cada interconexão entre unidades do processo atua como uma rota de comunicação: através destas, viajam valores numéricos de uma célula a outra, utilizando os pesos das conexões. Estes pesos são ajustados na fase de aprendizagem, com a finalidade de produzir uma rede de neurônios artificial final.

O funcionamento da rede é simples. Para cada vetor de entrada, a introducão na rede se dá através da cópia de cada valor desse vetor na célula de entrada correspondente. Cada célula de entrada na rede, uma vez recebida toda a informação, processa e propaga a mesma através de cada uma das conexões entre as células, até que se chegue às células de saída da rede.

3.2.2 Aprendizado

A aprendizagem de uma rede neural artificial consiste na determinação dos valores exatos dos pesos para todas as conexões, que capacitam a rede para a resolução eficente do problema. O processo geral de apredizagem consiste em introduzir um a um todos os exemplos do conjunto de aprendizagem e modificar os pesos das conexões seguindo um determinado esquema de aprendizado. Uma vez introduzidos todos os exemplos, checa-se se houve convergência; se não, repete-se o processo e todos os exemplos do conjunto voltam a ser introduzidos. A modificação dos pesos pode ser feita depois da introdução de cada exemplo do conjunto ou uma vez introduzidos todos eles.

O critério de convergência depende do tipo da rede utilizada ou do tipo de problema a ser resolvido. A finalização do período de aprendizado pode ser determinada:

- Mediante um número fixo de ciclos. Determina-se a priori quantas vezes será introduzido todo o conjunto de treinamento e, uma vez superado esse número, interrompe-se o processo e obtém-se a rede final.

- Quando o erro cai abaixo de uma quantidade pré-estabelecida. Neste caso, é definida uma função de erro, ao nível do padrão individual ou ao nível da totalidade do conjunto de treinamento.

- Quando a modificação dos pesos se torna irrelevante. Alguns modelos definem um esquema de aprendizado que faz com que as conexões (pesos) sejam modificadas cada vez com menor intensidade. Se o processo continuar, chegará o

momento em que já não haverá mais mudanças nos valores dos pesos de nenhuma conexão; é neste momento que se diz que a rede convergiu.

Segundo o esquema de aprendizado e o problema a ser resolvido, podem-se distinguir três tipos de esquemas de aprendizado:

- Aprendizado supervisionado. Neste esquema, os dados do conjunto de aprendizado têm dois tipos de atributos: os dados propriamente ditos e alguma informação relativa à solução do problema. Cada vez que um exemplo é introduzido e processado para se obter uma saída, tal saída é comparada com a saída que deveria ter sido produzida, que está disponível dentro do conjunto de dados de aprendizagem. A diferença entre as duas influirá na alteração dos pesos. Desta forma, as mudanças nos pesos serão feitas proporcionalmente à diferença entre eles. Para este tipo de aprendizagem diz-se que há um professor externo encarregado de determinar se a rede está comportando-se da forma adequada com relação à comparação entre as saídas produzidas e as esperadas, atuando, em conseqüência, na mudança dos pesos.

- Aprendizado não supervisionado. Neste esquema, os dados do conjunto de aprendizagem só têm informação dos exemplos, e não há nada que permita guiar o processo de aprendizagem; isto é, não existe professor externo que determine a aprendizagem. A rede só modificará os valores dos pesos a partir da informação interna. Quando se utiliza aprendizagem não supervisionada, a rede tenta determinar as características mais significativas, regularidades e redundâncias, dos dados do conjunto de treinamento.

- Aprendizado por reforço. O processo é semelhante ao caso supervisionado, porém o interesse não é que o sistema dê boas notícias sobre o conjunto de aprendizado, que já é conhecido, mas sobre os dados que possam surgir no futuro e cujas saídas são desconhecidas. O que acontece em muitos casos é

que um ajuste muito bom do conjunto de aprendizagem leva a previsões ruins. Neste caso, diz-se que a rede sofreu de *sobreajuste* dos dados de treinamento e que a sua capacidade de generalização é reduzida. O erro produzido pelos dados de treinamento não é uma boa medida da capacidade de predição ou generalização da rede. Para determinar se a rede produz saídas adequadas, divide-se o conjunto de aprendizagem em dois conjuntos, chamados de treinamento e validação. O conjunto de treinamento é utilizado para se aprender os valores dos pesos, i.e., para o ajuste da rede. A diferença é que, em vez de se medir o erro no conjunto de treinamento, utiliza-se a medição do erro relativo do conjunto de validação. Desta forma, para medir a eficácia da rede em resolver o problema, são utilizados dados que não são parte do conjunto de treinamento. Se o erro sobre o conjunto de validação for pequeno, ficará garantida a capacidade de generalização da rede.

3.3 Modelo Perceptron Multicamadas

Os primeiros estudos sobre redes neurais artificiais datam dos anos 50, sobretudo com o surgimento do modelo perceptron simples ou de camada única. Este modelo foi concebido como um sistema capaz de realizar tarefas de classificação de forma automática, porém com limitações na sua arquitetura e no mecanismo de aprendizagem; veja Isasi & Galván (2004). As primeiras tentativas devidas a Minsky & Papert (1969) mostraram que uma combinação de vários perceptrons simples (inclusão de neurônios ocultos) poderia resultar em uma solução adequada para se tratar certos problemas não-lineares; no entanto, os autores não apresentaram uma solução ao problema de adaptar os pesos da camada de entrada à camada oculta, pois a regra de aprendizagem do perceptron simples não poderia ser aplicada neste caso. A idéia de combinar vários perceptrons serviu de base para estudos posteriores, e foram os resultados obtidos por Rumelhart, Hilton & Williams (1986*b*) que apresentaram

uma maneira de retropropagar os erros medidos na saída da rede para os neurônios ocultos, dando lugar à chamada *Regra Delta Generalizada* que não é mais que uma generalização da regra Delta, veja Widrow (1960).

Cybenko (1989) e Hornik et al. (1989) entre outros, mostraram que o perceptron multicamadas é um aproximador universal, além de ser na atualidade uma das arquiteturas mais utilizadas na solução de problemas, devido a seu fácil uso e a sua vasta aplicabilidade. Estas redes têm sido aplicadas com sucesso em uma grande variedade de áreas como reconhecimento da fala: Cohen, Franco, Morgan, Rumelhart & Abrash (1993); modelagem de sistemas dinâmicos: Narendra & Parthasaranty (1990); previsão de séries temporais: Wieggend, Huberman & Rumelhart (1990).

A habilidade do perceptron multicamadas para aprender a partir de um conjunto de exemplos, como aproximar funções não-lineares, filtrar ruído nos dados, etc. faz com que seja um modelo adequado para abordar problemas reais, sem que isto indique que sejam os melhores aproximadores universais. Cada uma das classes de aproximadores tem as suas próprias características, não havendo tipicamente superioridade universal de nenhuma delas. Serão as considerações práticas de cada problema que determinarão a escolha do método a ser utilizado.

3.3.1 Arquitetura do perceptron multicamadas

A arquitetura do perceptron multicamadas é caracterizada pelo fato de seus neurônios serem agrupados em camadas de diferentes níveis. Cada uma destas camadas é formada por um conjunto de neurônios. Há três tipos de camadas diferentes: a camada de entrada, as camadas ocultas e a camada de saída, como se observa na figura 3.3. Os neurônios da camada de entrada não atuam como neurônios no sentido estrito da palavra, uma vez que se encarregam unicamente de receber sinais ou padrões que vêm do exterior e propagam tais sinais a todos os neurônios da camada seguinte. A última camada atua como saída da rede, proporcionando ao exterior a resposta da rede para cada um dos padrões de entrada. Os neurônios das camadas ocultas reali-

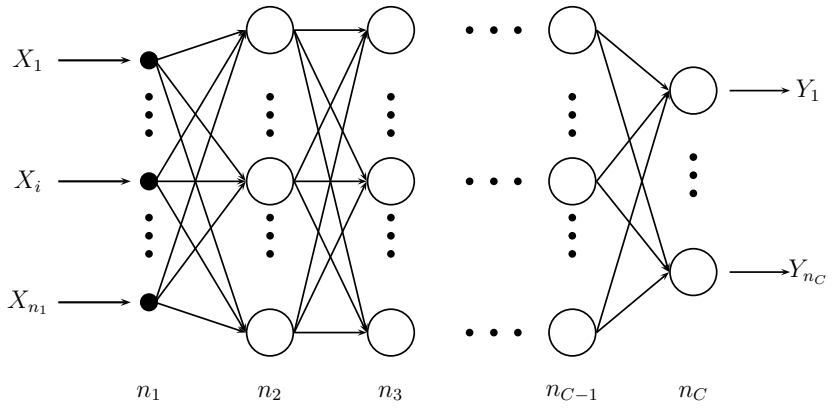

Figura 3.3: Rede neural *feedforward*.

zam um processamento não-linear dos padrões recebidos. Como pode ser observado na figura 3.3, as conexões do perceptron multicamadas estão sempre dirigidas para frente, i.e., os neurônios de uma camada conectam-se com os neurônios da camada seguinte, por isso recebem o nome de redes alimentadas para frente ou redes *feedforward*. Às conexões é associado um número real chamado peso da conexão e aos neurônios da rede um umbral, que no caso do perceptron multicamadas é tratado como uma conexão adicional ao neurônio.

3.3.2 Propagação dos padrões da entrada

O perceptron multicamadas define uma relação entre as variáveis de entrada e as variáveis de saída da rede. Esta relação é obtida propagando para frente os valores das variáveis de entrada. Com este fim, cada neurônio da rede processa a informação recebida por suas entradas e produz uma resposta ou ativação, que se propaga através das conexões correspondentes para os neurônios da camada seguinte. Apresentaremos a seguir as expressões dadas por Isasi & Galván (2004) para o cálculo das ativações dos neurônios da rede.

Considere um perceptron multicamadas com C camadas ($C-2$ camadas ocultas) e n_c neurônios na camada c, para $c = 1, 2, \ldots, C$. Seja $W^c = \left(w_{ij}^c\right)$ a matriz de pesos associada às conexões da camada c à camada $c+1$, para $c = 1, 2, \ldots, C-1$, em que w_{ij}^c representa o peso da conexão do neurônio i da camada c ao neurônio j da camada $c+1$; além disso, seja $U^c = (u_i^c)$ o vetor de umbrais dos neurônios da camada c para $c = 2, \ldots, C$. Denote-se como a_i^c a ativação do neurônio i da camada c; estas ativações são calculadas da seguinte forma:

- Ativação dos neurônios da camada de entrada (a_i^1). Os neurônios da camada de entrada encarregam-se de transmitir para a rede os sinais recebidos do exterior. Portanto,

$$a_i^1 = x_i, \tag{3.1}$$

  para $i = 1, 2, \ldots, n_1$, em que $X = (x_1, x_2, \ldots, x_{n_1})$ representa o vetor ou padrão de entrada à rede.

- Ativação dos neurônios da camada oculta c (a_i^c). Os neurônios ocultos da rede processam a informação recebida, aplicando a função de ativação f à soma dos produtos das ativações que recebem pelos seus correspondentes pesos, isto é:

$$a_i^c = f(\sum_{j=1}^{n_{c-1}} w_{ji}^{c-1} a_j^{c-1} + u_i^c), \tag{3.2}$$

  para $i = 1, 2, \ldots, n_c$ e $c = 2, 3, \ldots, C-1$.

- Ativação dos neurônios da camada de saída (a_i^C). Como no caso anterior, a ativação destes neurônios é dada pela função de ativação f aplicada à soma dos produtos das entradas que recebem pelos seus pesos correspondentes:

$$y_i = a_i^C = f(\sum_{j=1}^{n_{C-1}} w_{ji}^{C-1} a_j^{C-1} + u_i^C), \tag{3.3}$$

  para $i = 1, 2, \ldots, n_C$, em que $Y = (y_1, y_2, \ldots, y_{n_C})$ é o vetor de saída da rede.

Tabela 3.1: Funções de ativação.

função	f(x)
Linear	x
Tanh15	$tanh(1.5x)$
Sigmoidal	$(1 + e^{-x})^{-1}$
Seno	$sen(x)$
Gaussiana	$exp(-x^2)$
Treshold	$I_{\{x<\theta\}}, \quad \|\theta\| \geqslant 0$
Logística simétrica	$2/(1 + exp(-x)) - 1$
Complemento Gaussiana	$1 - exp(-x^2)$

A função f é chamada *função de ativação*. Para o perceptron multicamadas, as funções de ativação mais utilizadas são a sigmóide e a tangente hiperbólica; no entanto, várias outras funções de ativação são utilizadas, como é descrito em Gately (1996); veja também a tabela 3.1. O propósito da função de ativação ou transferência é não permitir saídas com valores muito grandes os quais podem *paralisar* (veja seção 3.3.6) a rede neural e inibir o treinamento. Como estabelecido por Klimasauskas (1994), se a rede é desenhada para aprender um comportamento em média, a função de ativação que deve ser utilizada é a sigmoide, porém, se a aprendizagem envolve desvios da média a melhor função é a tangente hiperbólica. Funções como a sigmóide e a tangente hiperbólica são freqüentemente utilizadas em séries temporais, já que são não-lineares e continuamente diferenciáveis, o que é uma propriedade desejável para o treinamento da rede. Geralmente a função de ativação no perceptron multicamadas é comum a todos os neurônios da rede e é escolhida pelo pesquisador, escolha que é feita unicamente com base nos valores de ativação que se deseja que os neurônios alcancem; a escolha de uma ou outra função de ativação não influi na capacidade da rede para resolver o problema; veja Isasi & Galván (2004). Em algumas situações e dependendo da natureza do problema, os neurônios de saída se distinguem dos demais neurônios da rede utilizando outro tipo de função de ativação; neste caso, a mais utilizada é a função linear.

3.3.3 Algoritmo de retropropagação

O algoritmo de aprendizado é o mecanismo mediante o qual se vão adaptando e modificando todos os parâmetros da rede. No caso do perceptron multicamadas, trata-se de um algoritmo de aprendizado supervisionado, i.e., a modificação dos parâmetros é feita de forma que a saída da rede seja a mais próxima da saída proporcionada pelo pesquisador ou saída desejada.

O problema de aprendizagem da rede é um problema de minimização da seguinte forma:

$$\min_{\Theta} E,$$

sendo Θ o conjunto de parâmetros da rede (pesos e umbrais) e E uma função do erro que avalia a diferença entre a saída da rede e a saída desejada. Na maioria dos casos, a função de erro é definida como

$$E = \frac{1}{N}\sum_{n=1}^{N} e(n), \tag{3.4}$$

em que N é o número de observações ou padrões e $e(n)$ é o erro cometido pela rede para o n-ésimo padrão, que é dado por

$$e(n) = \frac{1}{2}\sum_{i=1}^{n_C} \left(s_i(n) - y_i(n)\right)^2, \tag{3.5}$$

sendo $Y_n = (y_1(n), y_2(n), \ldots, y_{n_C}(n))$ e $S_n = (s_1(n), s_2(n), \ldots, s_{n_C}(n))$ os vetores de saída da rede e saída desejada para o n-ésimo padrão, respectivamente.

Desta forma, se Θ^* é um mínimo da função (3.4), i.e., o ponto onde o erro é o menor possível e a saída da rede é próxima da desejada, obtém-se o fim do processo de aprendizado. A presença de funções de ativação não-lineares faz com que a resposta da rede seja não-linear com relação aos parâmetros a serem ajustados, acarretando que a minimização de (3.4) é um problema de otimização não-linear; veja Nocedal & Wright (1999). Para o perceptron multicamadas, o método mais utilizado é o *steepest descent* sobre a função E; porém, têm sido desenvolvidos métodos de busca aleatória

para localizar o mínimo da função E, veja Solis & Wets (1981), e métodos baseados em técnicas evolutivas, veja Montana & Davis (1989).

Ainda que o aprendizado da rede seja feito para minimizar o erro total (3.4), o procedimento mais utilizado baseia-se no método do gradiente estocástico ou treinamento baseado em padrões (*Adaptative training*), veja Pham & Xing (1995), que consiste em uma minimização sucessiva dos erros de cada padrão $e(n)$, ao invés da minimização do erro total E. Portanto, aplicando o método *steepest descent*, cada parâmetro θ ($\theta \in \Theta$) da rede é modificado para cada padrão de entrada n de acordo com a seguinte lei de aprendizado:

$$\theta(n) = \theta(n-1) - \eta \frac{\partial e(n)}{\partial \theta}, \tag{3.6}$$

em que $e(n)$ é como definido em (3.5) e η é a taxa de aprendizado que influi na magnitude de deslocamento sobre a superfície do erro. Dado que os neurônios da rede são agrupados em camadas de distintos níveis, é possível aplicar o método do gradiente de forma eficiente resultando no conhecido algoritmo de retropropagação ou *regra delta generalizada*; veja Rumelhart et al. (1986*b*).

3.3.4 A regra delta generalizada

Os resultados obtidos por Isasi & Galván (2004) no desenvolvimento matemático da regra delta generalizada são apresentados a seguir; dois casos são considerados na obtenção dos pesos e umbrais da rede.

i. **Pesos da camada oculta $C-1$ à camada de saída C e umbrais da camada de saída**.

Pesos:

$$w_{ji}^{C-1}(n) = w_{ji}^{C-1}(n-1) + \eta \delta_i^C(n) a_j^{C-1}(n). \tag{3.7}$$

Umbrais:

$$u_i^C(n) = u_i^C(n-1) + \eta\delta_i^C(n), \tag{3.8}$$

em que

$$\delta_i^C(n) = (s_i(n) - y_i(n))f'(\sum_{j=1}^{n_{C-1}} w_{ji}^{C-1}a_j^{C-1} + u_i^C) \tag{3.9}$$

e $f'(\cdot)$ é a primeira derivada da função de ativação, $j = 1, 2, \ldots, n_{C-1}$, $i = 1, 2, \ldots, n_C$.

ii. **Pesos da camada c à camada $c+1$ e umbrais dos neurônios da camada $c+1$ para $c = 1, 2, \ldots, C-2$.**

Pesos:

$$w_{kj}^c(n) = w_{kj}^c(n-1) + \eta\delta_j^{c+1}(n)a_k^c(n). \tag{3.10}$$

Umbrais:

$$u_j^{c+1}(n) = u_j^{c+1}(n-1) + \eta\delta_j^{c+1}(n), \tag{3.11}$$

em que

$$\delta_j^{c+1}(n) = f'(\sum_{k=1}^{n_c} w_{kj}^c a_k^c + u_j^{c+1})\delta_i^{c+2}(n)w_{ji}^{c+1} \tag{3.12}$$

e $f'(\cdot)$ é a primeira derivada da função de ativação, $k = 1, 2, \ldots, n_c$, $j = 1, 2, \ldots, n_{c+1}$ e $c = 1, 2, \ldots, C-2$. Quando $c = C-2$, a expressão $\delta_i^{c+2}(n)w_{ji}^{c+1}$ na equação (3.12) muda para $\sum_{i=1}^{n_C} \delta_i^C(n)w_{ji}^{C-1}$, com $\delta_i^C(n)$ sendo definida em (3.9).

A mudança em um peso, como pode ser observado em (3.6), é proporcional ao gradiente do erro, com a proporcionalidade sendo dada pelo parâmetro η chamado taxa

de aprendizado. Este parâmetro é encarregado de controlar o deslocamento dos pesos da rede na superfície do erro seguindo a direção negativa do gradiente. Determina, portanto, a magnitude de tal deslocamento influindo na velocidade de convergência do algoritmo. Valores altos da taxa de aprendizado, em princípio, poderiam favorecer uma convergência mais rápida, pois permitem avançar rapidamente na superfície do erro. No entanto, taxas de aprendizado altas podem ter conseqüências negativas sobre a aprendizagem, fazendo com que o método pule um mínimo ou oscile ao redor do mínimo. Valores pequenos das taxas de aprendizagem podem evitar estes problemas, ainda que possivelmente acarretem convergência mais lenta do algoritmo de aprendizagem, pois a magnitude de deslocamento na superfície do erro é menor.

Um método simples para evitar a instabilidade no algoritmo de aprendizagem devido à taxa de aprendizado é modificar (3.6) através da inclusão de um segundo termo, chamado *momento*, obtendo desta forma a seguinte lei:

$$\theta(n) = \theta(n-1) - \eta\frac{\partial e(n)}{\partial \theta} + \alpha\Delta\theta(n-1), \tag{3.13}$$

em que $\Delta\theta(n-1) = \theta(n-1) - \theta(n-2)$ é a mudança do parâmetro θ da iteração $n-2$ para a iteração $n-1$. Aqui, α é um número positivo ponderando a quantidade anterior. Esta regra foi proposta por Rumelhart, Hilton & Williams (1986*a*), e preserva as propriedades da regra definida em (3.6), no sentido que modifica os parâmetros da rede para minimizar a função do erro (3.4). O novo termo, $\alpha\Delta\theta(n-1)$, incorpora ao método alguma inércia, fazendo com que a modificação atual do parâmetro dependa só da direção da modificação anterior, podendo evitar oscilações. Fazendo cálculos sucessivos sobre $\Delta\theta(n-1)$, Isasi & Galván (2004) apresentam uma expressão mais geral de (3.13):

$$\theta(n) = \theta(n-1) - \eta\sum_{t=0}^{n}\alpha^{n-t}\frac{\partial e(t)}{\partial \theta}\ . \tag{3.14}$$

Pode-se observar através de (3.14) que a mudança corrente de um parâmetro é determinada pela soma dos gradientes do erro, para todas as iterações anteriores. Portanto, quando as derivadas parciais dos erros com relação aos pesos têm sinais

opostos em iterações consecutivas, a soma pode ajudar a reduzir o efeito das derivadas parciais nas mudanças de sinais, com a finalidade de procurar uma trajetória mais suave para o peso, o que torna o método mais estável e faz com que ele apresente menos oscilações. Além disso, se as derivadas parciais dos erros com relação aos pesos têm o mesmo sinal em iterações consecutivas, a utilização do parâmetro de momento proporciona uma mudança maior no peso, acelerando a convergência do algoritmo.

Haykin (1994) chegou a algumas conclusões com relação aos valores de η e α que devem ser adotados no treinamento da rede neural. Após utilizar diferentes combinações dos parâmetros para treinar uma rede retropropagada com dois neurônios na camada oculta, ele constatou que a aprendizagem da rede foi mais eficiente utilizando os valores 0.1 e 0.5, respectivamente. Para avaliar o desempenho da rede nestes experimentos, foram considerados o erro final e o número de iterações necessárias para o treinamento. Os valores citados foram os que geraram os melhores resultados. Além disso, o autor também concluiu que quando η tende a 0, o uso de α próximo a 1 produz um aumento na velocidade de convergência. Por outro lado, quando η tende a 1, deve-se utilizar α tendendo a 0 para assegurar a estabilidade da aprendizagem. Foi constatado ainda que o uso das constantes $\eta = \{0.5, 0.9\}$ e $\alpha = \{0.9\}$ causou oscilações e erros finais elevados, que são efeitos indesejáveis.

3.3.5 Processo de aprendizagem do perceptron multicamadas

Seja $\{(X(n), S(n))\}$ o conjunto de padrões que representam o problema a resolver, em que $X(n) = \big(x_{_1}(n), \ldots, x_{n_{_1}}(n)\big)$ são os padrões de entrada à rede, $S(n) = (s_{_1}(n), \ldots, s_{n_{_1}}(n))$ são as saídas desejadas para tais entradas e N é o número de padrões disponíveis. Freqüentemente os padrões de entrada e saída são escalonados mediante uma transformação linear nos intervalos $[0, 1]$ e $[-1, 1]$, dependendo da função de ativação utilizada. A transformação linear e média/desvio padrão são as mais utilizadas em redes neurais, veja Kaastra & Boyd (1996).

O processo de aprendizagem do perceptron multicamadas é composto pelos seguintes passos:

1. Os pesos e umbrais são inicializados ao acaso no intervalo $(0, 1)$.

2. O n-ésimo padrão $(X(n), S(n))$ é utilizado no treinamento da rede, em que o vetor $X(n)$ é propagado até a saída da rede utilizando as equações (3.1), (3.2) e (3.3), obtendo-se a reposta $Y(n)$.

3. Avalia-se o erro quadrático médio para o n-ésimo padrão utilizando a equação (3.5).

4. Utiliza-se a regra delta generalizada para modificar os pesos e umbrais da rede. Para tanto, são utilizadas as equações (3.7) a (3.12).

5. São repetidos os passos 2, 3, 4 para todos os padrões de treinamento completando assim uma iteração ou ciclo de aprendizado. Este ciclo é geralmente chamado de *epoch*.

6. Avalia-se o erro total E cometido pela rede, utilizando os padrões de treinamento.

7. São repetidos os passos 2, 3, 4, 5 e 6 até se obter o menor erro de treinamento, para o qual são realizados m *epochs*.

O processo de aprendizado do perceptron multicamadas deve ser finalizado quando $\frac{\partial E}{\partial w} \approx 0$, momento no qual os parâmetros da rede não mudam de forma perceptível entre iterações consecutivas. Este processo de aprendizagem é o mais utilizado, ainda que exista uma variante conhecida como processo *batch*, veja Pham & Xing (1995). Este processo é diferente do *Adaptative training* (veja seção 3.3.3), pois nele os parâmetros da rede só são modificados quando todos os padrões de treinamento são apresentados à rede e não para cada padrão de treinamento isoladamente.

3.3.6 Capacidade de generalização

Na hora de avaliar o comportamento da rede e em particular do perceptron multicamadas, não importa apenas saber se a rede aprendeu com sucesso os padrões utilizados durante a aprendizagem, mas também conhecer o comportamento da rede frente a padrões que não foram utilizados durante o treinamento. De nada adianta dispor de uma rede que aprendeu corretamente os padrões de treinamento, mas que não responde adequadamente a novos padrões. É necessario que durante o processo de aprendizado a rede extraia as características mais importantes dos padrões de entrada, para que depois possa responder corretamente a padrões diferentes.

Portanto, quando se realiza o processo de aprendizado da rede é muito importante, até mesmo imprescindível, avaliar sua capacidade de generalização. Para isso, é necessário dispor de dois conjuntos de padrões, um chamado *conjunto de treinamento*, que treina e modifica os pesos e umbrais da rede, e outro chamado *conjunto de validação* ou *teste*, que mede a capacidade da rede para responder corretamente a padrões que não foram ingressados durante o treinamento. Quando a rede aproxima corretamente os padrões de aprendizagem mas não responde bem aos padrões de validação, diz-se que houve sub-aprendizagem da rede, o que pode ocorrer devido a vários fatores como o uso de um número excessivo de neurônios ou camadas ocultas, veja Isasi & Galván (2004).

Além disso, o algoritmo de retropropagação do perceptron multicamadas possui também uma série de deficiências, que são mencionadas a seguir e descritas mais detalhadamente em Isasi & Galván (2004):

- Mínimos locais. A superfície que define o erro E e, por sua vez, os parâmetros da rede, é complexa e cheia de vales e colinas. Devido à utilização do método do gradiente para encontrar o mínimo desta função, pode-se correr o risco de que o processo de minimização finalize em um mínimo local, veja Minsky & Papert (1988).

- Paralisia. O fenômeno da paralisia ou saturação no perceptron multicamadas ocorre quando a entrada total de um neurônio da rede assume valores muito altos, tanto positivos como negativos, através da função de ativação. Quando este fenômeno acontece, os parâmetros permanecem constantes e, como conseqüência, a soma dos erros locais permanece constante por um período longo de tempo, veja Lee & Kim (1991). Ainda que esta situação possa ser confundida com a presença de um mínimo local, devido ao erro constante, aqui pode ocorrer que, depois de algum tempo, o erro volte a decrescer.

3.3.7 Previsão de séries temporais

As redes neurais têm sido amplamente utilizadas nos últimos anos no contexto de previsão de séries temporais. Isto se deve às seguintes características das redes neurais:

- Sua capacidade para aproximar e capturar relações a partir de um conjunto de exemplos, sem a necessidade de se ter informações adicionais sobre a distribuição dos dados.
- A capacidade das redes neurais para construir relações não-lineares.
- A capacidade das redes neurais para construir relações a partir de informações incompletas ou com ruído.
- Os modelos baseados em redes neurais são fáceis de serem construídos e usados.

Seja $\{Z_t\}_{t=1,\ldots,N}$ uma série temporal cujo comportamento pode ser descrito por um modelo não-linear de regressão. Estes modelos caracterizam-se por captar o comportamento temporal da série no instante $t+1$ como uma função não-linear de $r+1$ valores anteriores da série temporal, i.e.,

$$Z_{t+1} = F(Z_t, Z_{t-1}, \ldots, Z_{t-r}) + \epsilon_t, \tag{3.15}$$

em que r é conhecido, ϵ_t um erro e $F(\cdot)$ é uma função não-linear desconhecida, que deve ser estimada ou aproximada. Dentro do contexto das redes neurais, geralmente não é feita nenhuma suposição sobre os erros do modelo, exceto quando se deseja construir intervalos de confiança, veja Chryssolouris, Lee & Ramsey (1996).

A construção de modelos não-lineares de regressão envolve a determinação da função $F(\cdot)$, a partir de um conjunto de dados disponíveis e de técnicas de aproximação, entre as quais estão as redes neurais. Já que o interesse reside na previsão de valores futuros, dois esquemas podem ser considerados:

- Previsão um passo à frente. Este procedimento consiste na previsão da série no instante imediatamente seguinte ao instante t, utilizando os dados disponíveis até t, i.e., faz-se a previsão de Z_{t+1} utilizando as observações $Z_t, Z_{t-1}, \ldots, Z_{t-r}$, $t = r+1, \ldots, N$. O vetor de entrada da rede será dado por $(Z_t, Z_{t-1}, \ldots, Z_{t-r})$ e a rede neural aproximará a função F por $\tilde{F}$, fornecendo o seguinte modelo de previsão:

$$\tilde{Z}_{t+1} = \tilde{F}(Z_t, Z_{t-1}, \ldots, Z_{t-r}).$$

- Previsão múltiplos passos à frente. Este procedimento consiste na previsão do valor da série no instante de tempo $t+h+1$, para o qual é utilizada a equação (3.15) de forma recorrente, i.e., a previsão é dada por

$$\tilde{Z}_{t+h+1} = \tilde{F}(Z_{t+h}, Z_{t+h-1}, \ldots, Z_{t+h-r}).$$

  No entanto, até o instante t nem toda a informação da entrada à rede está disponível, pois os valores da série $Z_{t+h}, Z_{t+h-1}, \ldots, Z_{t+1}$, para $h > 1$, não são conhecidos. Para contornar este problema, utilizam-se como entradas ao modelo neural os valores preditos pela rede neural nos instantes anteriores de tempo, em lugar dos valores da série. Assim, o modelo (3.15) pode ser utilizado na previsão em múltiplos passos, alimentando para trás a entrada da rede em procura de novas previsões até o instante $t+h+1$.

CAPÍTULO 4

AVALIAÇÃO RELATIVA DE PREVISÕES DE ARRECADAÇÃO TRIBUTÁRIA

Neste capítulo serão apresentadas as séries temporais escolhidas para a modelagem e previsão com redes neurais. Os resultados obtidos servirão como parâmetro de comparação com as previsões proporcionadas pelas metodologias de alisamento exponencial e de Box e Jenkins, com o objetivo de avaliar o poder preditivo das redes neurais na modelagem e previsão de séries temporais.

4.1 Considerações Gerais

A análise de modelagem e previsão apresentada a seguir basear-se-á em séries temporais mensais de arrecadação dos seguintes tributos:

a. *ICMS*. Imposto sobre Circulação de Mercadorias e sobre Prestações de Serviços de Transporte Interestadual e Intermunicipal e de Comunicação brasileiro.

b. *ICMSPE*. Imposto sobre Circulação de Mercadorias e sobre Prestações de Serviços de Transporte Interestadual e Intermunicipal e de Comunicação do estado de Pernambuco.

c. *ICMSRJ*. Imposto sobre Circulação de Mercadorias e sobre Prestações de Serviços de Transporte Interestadual e Intermunicipal e de Comunicação do estado de Rio de Janeiro.

d. *ICMSSP*. Imposto sobre Circulação de Mercadorias e sobre Prestações de Serviços de Transporte Interestadual e Intermunicipal e de Comunicação do estado de São Paulo.

As abreviações *ICMS*, *ICMSPE*, *ICMSRJ* e *ICMSSP* serão utilizadas para fazer referência a cada série. Os dados foram obtidos do *IPEADATA* através da página `www.ipeadata.gov.br`, com atualizações em 1 de julho de 2005. As séries a serem analisadas são deflacionadas pelo Índice de Preços ao Consumidor (*IPC*), ficando a preços de novembro de 2004. Os valores são expressos em milhões de reais. O período entre julho de 1994 a dezembro de 2004 é utilizado na modelagem e previsão de cada uma das séries temporais. Os logaritmos naturais das séries deflacionadas são utilizados na análise.

Em uma primeira análise (análise I), utilizamos o período de julho de 1994 a novembro de 2004 para a modelagem de cada série, na obtenção da previsão do mês de dezembro de 2004; o período de janeiro a março de 2005 também sendo prognosticado. O interesse principal é obter uma boa previsão do valor arrecadado do mês de dezembro de 2004 e observar, posteriormente, o desempenho na previsão do período de janeiro a março de 2005. Depois, em uma segunda análise (análise II), o período entre julho de 1994 e junho de 2004 é utilizado na modelagem com vistas à previsão das observações entre julho e dezembro de 2004. Em cada uma das análises, deseja-se observar como cada modelo captura a estrutura das séries, e as reflete na previsão de cada um dos períodos já definidos.

Duas formas de medição do erro de previsão são adotadas para estabelecer as comparações das previsões realizadas por cada metodologia:

- **Erro percentual absoluto médio (MAPE)**. Se $Z_1, Z_2, \ldots, Z_t$ são as observações da série temporal, e $Z_{t+1}, Z_{t+2}, \ldots, Z_{t+h}$ as previsões até o instante $t+h$, então
$$MAPE(\%) = \frac{1}{h}\sum_{k=1}^{h}\frac{|e_{t+k}|}{Z_{t+k}} \times 100,$$
em que e_{t+k} é o erro de previsão de Z_{t+k}, o valor da série no instante $t+k$.

- **Desvio absoluto da mediana (MAD)**. Aqui,
$$MAD = med(|e_{t+1} - med_e|, |e_{t+2} - med_e|, \ldots, |e_{t+h} - med_e|),$$
em que e_{t+k} é o erro de previsão no instante $t+k$, $k = 1, 2, \ldots, h$, e med_e a mediana dos erros de previsão. Esta medida foi sugerida por Dijk (1999) e é robusta a outliers.

4.2 Análise Descritiva

As séries de arrecadação a preços correntes são exibidas na figura 4.1, para o período de julho de 1994 a março de 2005. É possível observar uma tendência crescente na arrecadação, comum a todas as séries. A tabela 4.1 apresenta estatísticas descritivas.

Tabela 4.1: estatísticas - preços correntes (milhões de reais).

série	média	desv. padrão	min	max
ICMS	6838.5270	2550.1840	2459.9010	12150.9370
ICMSPE	182.5333	71.3622	61.6200	374.4500
ICMSRJ	667.2427	253.5677	237.0480	1668.4330
ICMSSP	2472.9000	816.7896	954.2540	4250.5450

Neste período, o *ICMS* arrecado foi em média de 6.84 bilhões de reais com desvio padrão de 2.55 bilhões de reais; a arrecadação mínima registrada foi de 2.46 bilhões

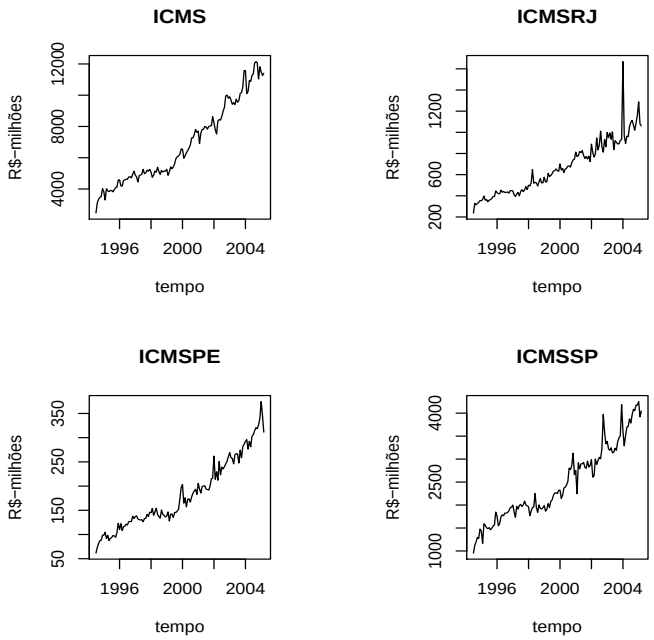

Figura 4.1: Séries de arrecadação a preços correntes: jul/94-mar/05.

no mês de julho de 1994 e a arrecadação máxima sendo de 12.15 bilhões no mês de setembro de 2004. A variação percentual média da arrecadação do *ICMS* durante meses consecutivos foi 1.35%. Para o *ICMSPE*, arrecadou-se em média 182.53 milhões de reais, com desvio padrão de 71.36 milhões de reais; a arrecadação mínima 61.62 milhões de reais no mês de julho de 1994, a arrecadação máxima sendo 374.45 milhões de reais no mês de janeiro de 2005. A variação percentual média da arrecadação durante meses consecutivos para o *ICMSPE* foi de 1.54%. Para o *ICMSRJ*, a média de arrecadação foi de 667.24 milhões de reais, com desvio padrão de 253.56 milhões de reais. A arrecadação mínima foi de 237.04 milhões de reais no mês de julho de 1994, enquanto a máxima foi de 1.67 bilhões de reais no mês de janeiro de 2004. A variação percentual média de arrecadação do *ICMSRJ* foi de 1.75%. Por último, o *ICMSSP*, com média de 2.47 bilhões de reais e desvio padrão de 816.79 milhões de reais, registrou um valor mínimo de 954.25 milhões de reais no mês de julho de 1994

e máximo de 4.25 bilhões de reais no mês de janeiro de 2005. A variação percentual média da arrecadação do *ICMSSP* foi de 1.44%.

As quatro séries de arrecadação deflacionadas, para o período entre julho de 1994 e março de 2005 são apresentadas na figura 4.2. Além disso, na tabela 4.2, estatísticas descritivas são exibidas para cada uma das séries.

Pode-se observar na tabela 4.2 que a média da série *ICMS* foi de 9.53 bilhões de reais, com desvio padrão de 1.32 bilhões de reais. O menor valor observado foi 6.76 bilhões de reais no mês de julho de 1994 e o maior valor observado foi 12.23 bilhões de reais, registrado no mês de dezembro de 2003. Para o *ICMSPE* a média foi 253.29 milhões de reais, com desvio padrão de 39.16 milhões de reais. O menor valor observado no *ICMSPE* foi 169.33 milhões de reais, no mês de julho de 1994, e o maior valor foi 368.97 milhões de reais no mês de janeiro de 2005. Para o *ICMSRJ* a média foi 930.53 milhões de reais com desvio padrão de 150.82 milhões de reais; o valor mínimo foi 651.41 milhões de reais no mês de julho de 1994 e valor máximo de 1.74 bilhões de reais, no mês de janeiro de 2004. Por último, a série do *ICMSSP* apresenta média de 3.48 bilhões de reais, com desvio padrão de 377.68 milhões de reais. O valor mínimo observado do *ICMSSP* aconteceu no mês de julho de 1994, com valor de 2.63 bilhões de reais, enquanto o máximo aconteceu no mês de outubro de 2002, com valor de 4.80 bilhões de reais. As taxas em média, de crescimento mês a mês são 0.53%, 0.71%, 0.92% e 0.62% para o *ICMS*, *ICMSPE*, *ICMSRJ* e *ICMSSP*, respectivamente.

Tabela 4.2: estatísticas - preços Nov/04 (milhões de reais).

Série	Média	Dsv. padrão	Min	Max
ICMS	9532.1210	1319.5900	6759.8500	12227.9210
ICMSPE	253.2867	39.1590	169.3328	368.9712
ICMSRJ	930.5325	150.8157	651.4119	1743.1352
ICMSSP	3483.2900	377.6796	2622.3060	4803.8310

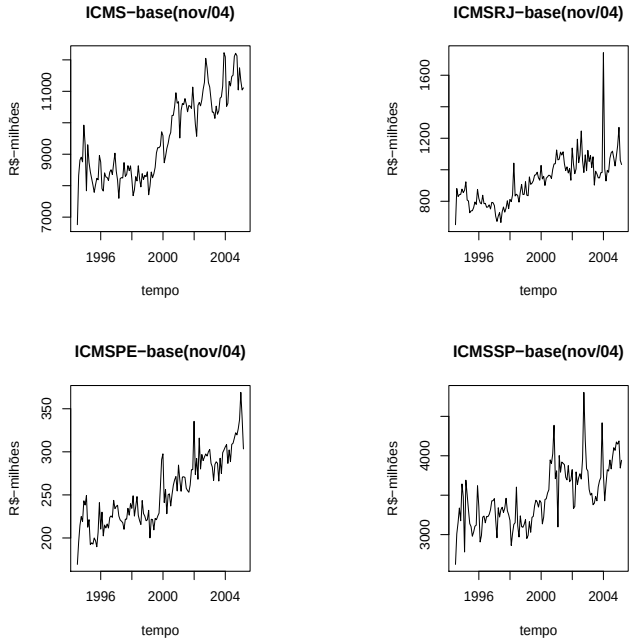

Figura 4.2: Séries de arrecadação a preços de nov/04: jul/94-mar/05.

Para cada uma das séries na figura 4.2, é possível observar fortes indícios de não estacionariedade. Tendências estocásticas podem ser observadas, e.g., no *ICMSPE*; no *ICMS*, uma tendência crescente inicia em 1999, depois de um período em ausência de tendência; O *ICMSRJ*, de 1997 até o final de 2002 exibe tendência crescente, seguindo-se um período de estabilidade, truncado pela ocorrência de uma alta arrecadação em janeiro de 2004, veja tabela 4.2. Por sua vez, o *ICMSSP* apresenta uma tendência crescente, porém não fortemente marcada; além disso, mudanças fortes na arrecadação do *ICMSSP* podem ser observadas, começando em 2001 e depois na metade do ano 2002 e em finais de 2003. Estruturas sazonais são notórias, e.g., para o *ICMS*, o *ICMSPE* e o *ICMSSP* como pode-se observar na figura 4.2; porém, para o *ICMSRJ* a existência de tal estrutura sazonal não é clara a partir da visualização da figura 4.2.

4.3 Modelagem e Previsão: Análise I

4.3.1 Alisamento exponencial

Os algoritmos aditivo e multiplicativo descritos na seção 2.2 serão utilizados para obter, inicialmente, previsões referentes ao período que se estende de dezembro de 2004 a março de 2005. As escolhas dos valores das constantes de suavização foram feitas através da minimização da soma de quadrados dos erros de previsão um passo à frente. A tabela 4.3 apresenta os valores otimizados das constantes para cada uma das séries na escala logarítmica.

Tabela 4.3: Parâmetros de suavização de alisamento exponencial.

	HW- aditivo			HW- multiplicativo		
Séries	α	β	γ	α	β	γ
ICMS	0.7537	0.0000	0.6930	0.7459	0.0000	0.6979
ICMSPE	0.3358	0.0006	0.4274	0.3353	0.0006	0.4316
ICMSRJ	0.3791	0.0000	0.5311	0.3815	0.0000	0.5347
ICMSSP	0.2748	0.0006	0.4289	0.2743	0.0008	0.4361

Nota-se da tabela 4.3 que, para todas as séries os valores das constantes de suavização são semelhantes para os modelos aditivo e multiplicativo. Isto é conseqüência de utilizar a transformação logarítmica; caso a especificação verdadeira seja multiplicativa, passa-se a ter uma especificação aditiva quando esta transformação é aplicada. Além disso, a constante de tendência determinada por β assume valores próximos de zero, indicando um crescimento lento dos valores de cada série ao longo do tempo. A presença de valores zero em β, como em *ICMS* e *ICMSRJ*, indicam auscencia no padrão de crescimento ou decrescimento dos valores assumidos pela tendência, durante períodos consecutivos de tempo.

A semelhança entre as constantes de suavização obtidas para os modelos aditivo e multiplicativo implica também previsões parecidas. Desta forma, será escolhido o modelo aditivo na geração das previsões. As tabelas 4.4 a 4.7 contêm os resultados

Tabela 4.4: Previsão do *ICMS* através de alisamento exponencial.

data	real	prev-o	er(%)	prev-m	er(%)
dez/04	11746.0785	11787.8512	0.3556	11787.8512	0.3556
jan/05	11331.6471	11696.4975	3.2197	11727.8862	3.4967
fev/05	11042.4777	10579.6193	−4.1916	10864.4875	−1.6119
mar/05	11109.9315	10987.8426	−1.0989	10925.2808	−1.6620

Tabela 4.5: Previsão do *ICMSPE* através de alisamento exponencial.

data	real	prev-o	er(%)	prev-m	er(%)
dez/04	337.2400	335.2494	−0.5903	335.2494	−0.5903
jan/05	368.9698	351.0205	−4.8647	350.3227	−5.0538
fev/05	339.0253	338.9405	−0.0250	332.6446	−1.8821
mar/05	303.3947	327.4105	7.9157	321.2998	5.9016

Tabela 4.6: Previsão do *ICMSRJ* através de alisamento exponencial.

data	real	prev-o	er(%)	prev-m	er(%)
dez/04	1155.0683	1050.6925	−9.0363	1050.6925	−9.0363
jan/05	1269.0384	1425.5667	12.3344	1375.2829	8.3720
fev/05	1060.0536	1017.9486	−3.9720	1026.3198	−3.1823
mar/05	1033.2296	1026.1146	−0.6886	1018.7735	−1.3991

Tabela 4.7: Previsão do *ICMSSP* através de alisamento exponencial.

data	real	prev-o	er(%)	prev-m	er(%)
dez/04	4148.0546	4157.0656	0.2172	4157.0656	0.2172
jan/05	4188.3608	3912.0027	−6.5982	3914.3115	−6.5431
fev/05	3844.5941	3598.9043	−6.3905	3534.1027	−8.0761
mar/05	3942.9386	3837.4936	−2.6743	3700.5425	−6.1476

da previsão um passo à frente (prev-o) e quatro ($h = 4$) passos à frente (prev-m), com seus respectivos erros relativos para cada uma das séries. Os resultados estão expressos em milhões de reais.

As previsões para o *ICMS* na tabela 4.4 para o meses de dezembro de 2004 e março de 2005 são razoavelmente precisas. Neste caso, os erros relativos absolutos foram menores que 5%. Já para o *ICMSPE*, tabela 4.5, as previsões dos meses de janeiro e março de 2005, exibem os erros relativos mais altos. Na tabela 4.6, os erros relativos de previsão dos meses de dezembro de 2004 e janeiro de 2005 para o *ICMSRJ* são consideravelmente altos, com notável subestimação e sobre-estimação respectivamente; porém, os demais erros relativos absolutos são menores que 4%. Para o *ICMSSP*, tabela 4.7, as previsões de janeiro e fevereiro de 2005 possuem, respectivamente, erros relativos de previsão um e quatro passos à frente menores que 9% aproximadamente, subestimando os verdadeiros valores; no entanto, uma boa previsão do mês de dezembro de 2004 é obtida.

Como é exibido na tabela 4.8, o *MAPE* e o *MAD* são calculados para as previsões um passo (prev-o) e quatro passos (prev-m) à frente, para cada uma das séries. É possível observar através desta tabela que os erros de previsão quatro passos à frente proporcionados pelo alisamento exponencial, em geral são menores do que os obtidos para um passo à frente.

Tabela 4.8: Medidas de erro das previsões de alisamento exponencial.

Série	*MAPE(%)*		*MAD*	
	prev-o	prev-m	prev-o	prev-m
ICMS	2.2165	1.7816	243.4697	113.2117
ICMSPE	3.3489	3.3569	8.9322	8.3283
ICMSRJ	6.5078	5.4974	48.6304	44.9598
ICMSSP	3.9701	5.2460	85.4566	34.0477

As séries corrigidas pelos outliers, veja seção 4.3.2, foram também utilizadas para se fazer a previsão de dezembro de 2004 até março de 2005; no entanto, os resultados não melhoram muito com relação aos exibidos nas tabelas 4.4 a 4.7.

4.3.2 Método de Box e Jenkins

A modelagem através da metodologia de Box e Jenkins consiste de três etapas como foi discutido na seção 2.3. Para o processo de identificação do modelo, será utilizado o criterio bayesiano de informação `BIC`, i.e., tendo um conjunto de possíveis modelos para uma determinada série, será escolhido aquele que apresenta o menor `BIC`; veja Mills (1999). O programa *TRAMO-SEATS* é utilizado para a identificação, estimação e validação dos modelos obtidos para cada uma das séries. Através da figura 4.2, como foi descrito na seção 4.2, as séries, em geral, são não-estacionárias e exibem alguns valores que poderiam corresponder a observações atípicas. Correção por outliers desta forma é importante na análise de séries temporais, como descrito por Box & Tiao (1975), Chen & Liu (1993), Kaiser & Maravall (2001), Maravall & Kaiser (2000), Peña (2001), entre outros. O programa *TRAMO-SEATS* é utilizado no processo de correção de outliers.

As figuras 4.3 a 4.6 exibem as funções amostrais de autocorrelação (`ACF`) e autocorrelação parcial (`PACF`), como também a série logaritmada no período de julho a novembro de 2004. Nesses gráficos o valor de 1.0 corresponde à defasagem de ordem 12, 2.0 à defasagem de ordem 24 e assim por diante.

Observando a figura 4.3, nota-se a presença de tendência e sazonalidade ($s = 12$) para o *ICMS*. Para o *ICMSPE*, figura 4.4, também observa-se a existência de tendência e sazonalidade com periodicidade $s = 12$. Já para o *ICMSRJ*, figura 4.5, o decaimento das autocorrelações à medida que aumenta a defasagem determina uma clara tendência, porém não fica evidente a presença de sazonalidade. A figura 4.6 sugere a existência de tendência e sazonalidade com periodicidade $s = 12$ para o *ICMSSP*.

O teste aumentado de Dickey-Fuller (`ADF`) é utilizado para testar a hipótese de raiz unitária (não-estacionariedade). Os resultados dos testes para as diferentes séries diferenciadas são apresentados na tabela 4.9. Os resultados indicam que as diferenças utilizadas forneceram séries estacionárias.

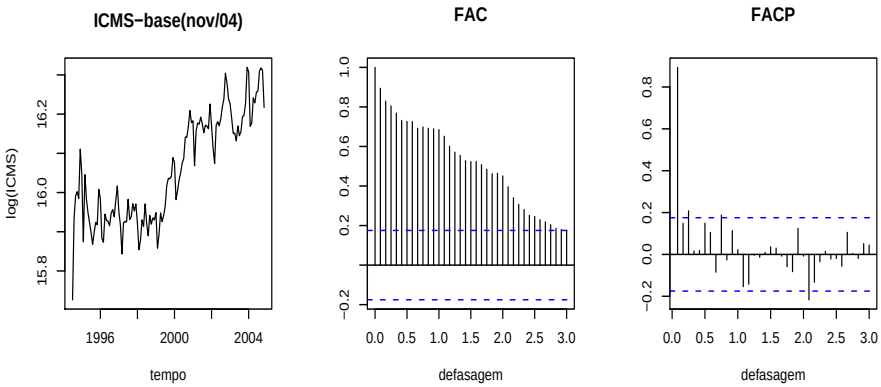

Figura 4.3: Série logaritmo do *ICMS*, ACF e PACF.

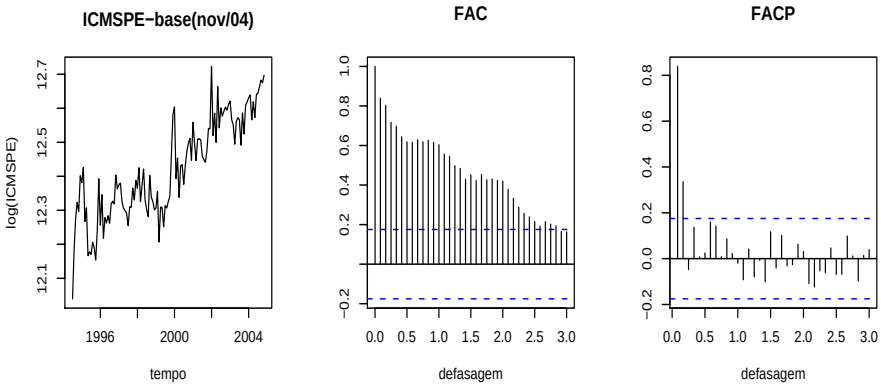

Figura 4.4: Série logaritmo do *ICMSPE*, ACF e PACF.

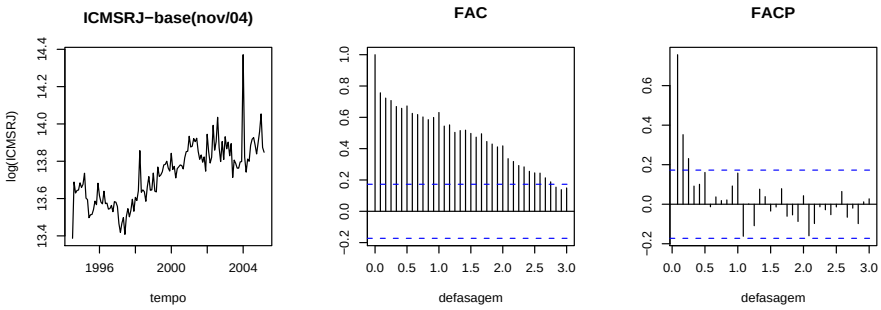

Figura 4.5: Série logaritmo do *ICMSRJ*, ACF e PACF.

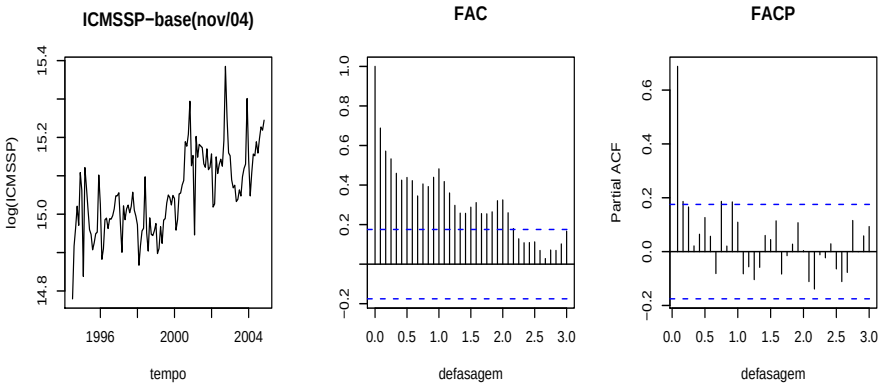

Figura 4.6: Série logaritmo do *ICMSSP*, ACF e PACF.

Tabela 4.9: Testes de estacionariedade.

série	defasagem	ADF	p valor
ICMS	$\Delta\Delta_{12}$	-5.1569	< 0.01
ICMSPE	$\Delta\Delta_{12}$	-5.3105	< 0.01
ICMSRJ	Δ	-6.4618	< 0.01
ICMSSP	$\Delta\Delta_{12}$	-6.0731	< 0.01

Determinadas as constantes d e D para cada série, diferentes modelos são considerados variando os valores de p, q, P e Q, na etapa de identificação, com o objetivo de escolher o modelo que proporcione o menor `BIC`. O programa *TRAMO-SEATS* permite variar os parâmetros de p e q de 0 até 3, enquanto que P e Q somente assumem os valores 0 e 1. Para cada um dos modelos analisados, identificação e correção de outliers são efetuadas.

Para a série do *ICMS*, o modelo $SARIMA(0,1,1) \times (0,1,1)_{12}$ apresenta o menor criterio de informação, `BIC` $= -6.6462$, do conjunto de modelos considerados. No que tange à série do *ICMSPE*, o modelo $SARIMA(1,1,0) \times (0,1,1)_{12}$ é escolhido com `BIC` $= -5.6916$; para *ICMSRJ*, o modelo selecionado é o $ARIMA(0,1,1)$, com `BIC` $= -5.5901$, uma vez que outros possíveis modelos tentativos ($ARIMA(0,1,3)$, $ARIMA(2,1,0)$ e $ARIMA(0,1,2)$), que exibem menores `BIC`, apresentam algumas estimativas estatisticamente não significativas. Além disso, as previsões do modelo $ARIMA(0,1,1)$, exibem menores erros de previsão que as dos outros modelos citados. Por último, para o *ICMSSP* o modelo $SARIMA(0,1,1) \times (0,1,1)_{12}$ com `BIC` $= -6.0643$ é escolhido.

Para os modelos escolhidos, o procedimento de detecção de outliers é feito usando o programa *TRAMO-SEATS*, segundo um esquema semelhante ao apresentado por Chen & Liu (1993). *TRAMO-SEATS* detecta três tipos de outliers: `AO`, `TS` e `LS`. Uma vez identificados os parâmetros do modelo *SARIMA* e os associados aos outliers, eles são estimados conjuntamente por máxima verosimilhança exata.

Tabela 4.10: Modelos escolhidos e estatísticas de diagnóstico.

Série	*ICMS*	*ICMSPE*	*ICMSRJ*	*ICMSSP*
Modelo $(p,d,q)\times(P,D,Q)_s$	SARIMA $(0,1,1)\times(0,1,1)_{12}$	SARIMA $(1,1,0)\times(0,1,1)_{12}$	ARIMA $(0,1,1)$	SARIMA $(0,1,1)\times(0,1,1)_{12}$
SE(Res)	0.0337	0.0544	0.0575	0.0409
BIC	−6.6462	−5.6916	−5.5901	−6.0643
Q-Val	9.5410	19.4100	21.5900	14.1300
N-test	2.2400	2.4400	1.8600	6.6900
SK(t)	−1.2000	0.3770	0.4110	−2.3000
KUR(t)	0.8960	1.5200	1.3000	1.1900
Q2	14.5300	28.1200	17.6600	25.7600
RUNS	0.1920	−0.7700	−0.1800	0.0000

Quantiles: $\chi^2_2(1\%) = 9.21$, $\chi^2_2(5\%) = 6.00$, $\chi^2_{22}(5\%) = 33.92$, $\chi^2_{23}(5\%) = 35.17$, $\chi^2_{24}(5\%) = 36.41$

A tabela 4.10 exibe os modelos selecionados para cada série, como também medidas de diagnóstico sobre os resíduos, em que `SE(Res)` é o erro-padrão dos resíduos; $\texttt{Q} - \texttt{Val}$ é a estatística de Ljung-Box para testar correlação residual, calculada sobre 24 autocorrelações amostrais (em todos os casos, a distribuição assintótica χ^2 é utilizada, com 22, 22, 23 e 22 graus de liberdade (g.l.), respectivamente); $\texttt{N} - \texttt{test}$ é a estatística de teste de Bowman-Shenton para normalidade (em todos os casos, a distribuição assintótica $\chi^2(2 \text{ g.l.})$ é utilizada); `SK(t)` é a estatística usada para testar se a assimetria é zero contra se é diferente de zero; `KUR(t)` é a estatística usada para testar se o excesso de curtose é zero contra se é diferente de zero; `Q2` é a estatística de McLeod & Li (1983) para testar a linearidade do processo (em todos os casos, a distribuição assintótica $\chi^2(24 \text{ g.l.})$ é utilizada); por último, `RUNS` é a estatística usada para testar a hipótese nula de aleatoriedade nos sinais dos resíduos. Todos os testes são realizados ao nível de significancia de 5%.

Através da tabela 4.10 é possível observar que para todos os modelos não há evidência contra a hipótese de ausência de autocorrelação dos resíduos, como também contra a hipótese de normalidade dos resíduos, com exceção dos resíduos do modelo do *ICMSSP*; no entanto, ao nível de significância de 1%, a hipótese de normalidade

pode ser não rejeitada. Os testes sobre a curtose e a assimetria não conduzem à rejeição das hipóteses nulas. O teste de Mcleod e Li sugere que a descrição das séries através de modelos lineares é aceitável.

Finalmente, a equação do modelo estimado para a série do logaritmo do *ICMS*, denotada por Z_t, é dada por

$$Z_t = \underset{(-6.01)}{-0.2125} I_t^{(Jul/94)} + \underset{(4.55)}{0.1337} I_t^{(Mar/95)} + X_t,$$

em que $I_t^{(\cdot)}$ é definido como na seção 2.3.3, entre parênteses o valor da t que gera o teste de significância e

$$(1-B)(1-B^{12})X_t = (1 + \underset{(-4.2)}{0.3652}B)(1 + \underset{(-157.7)}{0.9976}B^{12})a_t,$$

$$a_t \sim N(0, (0.0337)^2).$$

Para a série do logaritmo do *ICMSPE*, denotada por Z_t, o modelo estimado é dado por

$$Z_t = \underset{(-3.47)}{-0.1453} I_t^{(Out/95)} + \frac{0.2011}{1 - 0.7B} I_t^{(dez/99)} + X_t,$$

em que

$$(1 - \underset{(10.12)}{0.6776}B)(1-B)(1-B^{12})X_t = (1 + \underset{(-7.6)}{0.7136}B^{12})a_t,$$

$$a_t \sim N(0, (0.0543)^2).$$

Para a série do logaritmo do *ICMSRJ*, denotada por Z_t, o modelo estimado é dado por

$$Z_t = \underset{(11.43)}{0.5734} I_t^{(jan/04)} - \underset{(-4.82)}{0.2770} I_t^{(jul/94)} + \underset{(4.65)}{0.2332} I_t^{(abr/98)} + X_t,$$

em que

$$(1 - B)X_t = \underset{(-6.9)}{(1 + 0.5241B)} a_t,$$

$$a_t \sim N(0, (0.0574)^2).$$

A equação do modelo estimado para a série do logaritmo do *ICMSSP*, denotada por Z_t, é dada por

$$\begin{aligned} Z_t = {} & \underset{(5.44)}{0.1749} I_t^{(out/02)} + \underset{(5.05)}{0.1736} I_t^{(dez/03)} \\ & + \underset{(4.48)}{0.1424} I_t^{(nov/00)} + \underset{(4.58)}{0.1456} I_t^{(jun/98)} \\ & - \underset{(-4.00)}{0.1640} I_t^{(jul/94)} + \frac{\underset{(6.18)}{0.2332}}{1 - 0.7B} I_t^{(mar/95)} \\ & + \frac{\underset{(5.26)}{0.1845}}{1 - 0.7B} I_t^{(mar/01)} + \frac{\underset{(3.35)}{0.1171}}{1 - 0.7B} I_t^{(ago/00)} + X_t, \end{aligned}$$

em que

$$(1 - B)(1 - B^{12})X_t = \underset{(-4.4)}{(1 + 0.4082B)} \underset{(-6.7)}{(1 + 0.6845B^{12})} a_t,$$

$$a_t \sim N(0, (0.0409)^2).$$

A figura 4.7 apresenta as séries originais conjuntamente com as séries corrigidas, entre julho de 1994 e novembro de 2004. Pode-se observar que os valores que aparentavam ser outliers com a simples observação da figura 4.2 são detectados e corrigidos por cada um dos diferentes modelos definidos para cada série.

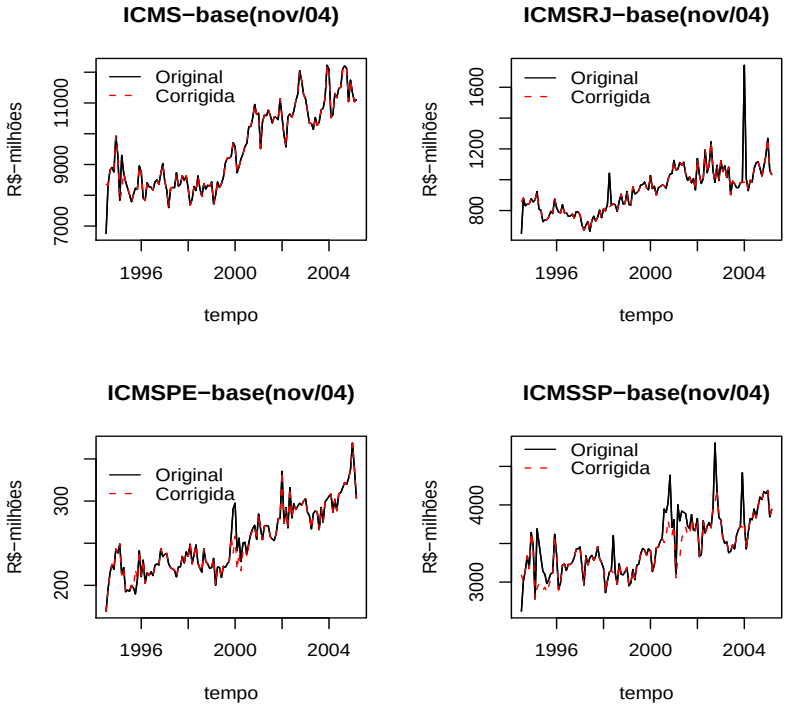

Figura 4.7: Séries corrigidas por outliers.

As previsões, expressas em milhões de reais, são exibidas para cada série, nas tabelas 4.11 a 4.14. Note que, para alguns valores objeto da previsão, mudanças consideráveis tiveram lugar, ocasionando que as previsões não as capturaram rapidamente, como é o caso da previsão um passo à frente do mês de março de 2005 do *ICMSPE*, com um erro relativo de 10.65%. O mesmo acontece com as previsões um e quatro passos à frente do *ICMSRJ* do mês de janeiro de 2005, com erros relativos de -12.57% e -15.70%, respectivamente. A previsão um passo à frente de fevereiro de 2005, do *ICMSRJ*, igualmente exibe um erro relativo de 11.57%. A previsão de valores futuros de *ICMSSP*, exibe, em geral, erros relativos menores que 4%.

Tabela 4.11: Previsão do *ICMS* através do modelo $SARIMA(0,1,1) \times (0,1,1)_{12}$.

data	real	prev-o	er(%)	prev-m	er(%)
dez/04	11746.0785	11944.7212	1.6911	11944.7212	1.6911
jan/05	11331.6471	11475.3062	1.2678	11626.5287	2.6023
fev/05	11042.4777	10422.6825	−5.6128	10617.3499	−3.8499
mar/05	11109.9315	10870.2037	−2.1578	10676.9738	−3.8970

Tabela 4.12: Previsão do *ICMSPE* através do modelo $SARIMA(1,1,0) \times (0,1,1)_{12}$.

data	real	prev-o	er(%)	prev-m	er(%)
dez/04	337.2400	331.0418	−1.8379	331.0418	−1.8379
jan/05	368.9698	345.6078	−6.3317	343.5198	−6.8976
fev/05	339.0253	340.9343	0.5631	328.9957	−2.9584
mar/05	303.3947	335.6959	10.6466	316.7601	4.4053

Tabela 4.13: Previsão do *ICMSRJ* através do modelo $ARIMA(0,1,1)$.

data	real	prev-o	er(%)	prev-m	er(%)
dez/04	1155.0683	1069.7120	−7.3897	1069.7120	−7.3897
jan/05	1269.0384	1109.5077	−12.5710	1069.7120	−15.7069
fev/05	1060.0536	1182.7455	11.5741	1069.7120	0.9111
mar/05	1033.2296	1122.6910	8.6584	1069.7120	3.5309

Tabela 4.14: Previsão do *ICMSSP* através do modelo $SARIMA(0,1,1) \times (0,1,1)_{12}$.

data	real	prev-o	er(%)	prev-m	er(%)
dez/04	4148.0546	4095.5026	−1.2669	4095.5026	−1.2669
jan/05	4188.3608	4166.2004	−0.5291	4135.0087	−1.2738
fev/05	3844.5941	3757.5323	−2.2645	3717.6414	−3.3021
mar/05	3942.9386	3915.2706	−0.7017	3821.6774	−3.0754

Por último, a tabela 4.15 exibe as medidas de erro de previsão, *MAPE* e *MAD*, para cada uma das séries na escala original. As previsões de *ICMS*, *ICMSPE* e *ICMSSP* apresentam valores do *MAPE* menores que 5% na previsão um e quatro passos à frente, enquanto a previsão de *ICMSRJ* exibe *MAPE* na previsão um passo à frente de 10.05%. O *MAD* das previsões um passo à frente de *ICMS* e de *ICMSSP* são menores que o *MAD* determinado pelas previsões quatro passos à frente; no entanto, para o *ICMSRJ* e o *ICMSPE*, acontece o caso contrário.

Tabela 4.15: Medidas de erro das previsões *SARIMA*: um e quatro passos à frente.

	MAPE(%)		*MAD*	
Série	prev-o	prev-m	prev-o	prev-m
ICMS	2.6824	3.0101	219.1853	315.8002
ICMSPE	4.8448	4.0248	12.6355	9.6259
ICMSRJ	10.0483	6.8847	104.0241	60.9193
ICMSSP	1.1906	2.2296	15.1958	34.3546

4.3.3 Redes neurais artificiais

A arquitetura de redes neurais mais amplamente utilizada é a perceptron multicamadas, já que é considerada um aproximador universal, enquanto o algoritmo de treinamento clássico é baseado na minimização do erro quadrático médio (*MSE*). No entanto, sabe-se que tal esquema de treinamento não é adequado para conjuntos de dados na presença de outliers, veja Chuang, Sue & Hsiao (2000), Liano (1996) e Pernía-Espinoza, Ordieres, de Pisón & Marcos (2005). Duas alternativas são consideradas na literatura para o controle de outliers: a primeira consiste em utilizar métodos robustos que preservam a confiabilidade, ainda que alguma quantidade de dados esteja contaminada; e a segunda consiste em identificar e corrigir os outliers, antes de modelar o conjunto de dados. Já que o perceptron multicamadas não é uma técnica robusta a outliers, o segundo caminho é o mais viável. O programa *TRAMO-SEATS* pode ser utilizado para a identificação e correção de outliers. A figura 4.7 exibe as séries corrigidas por outliers.

Para a construção de uma rede neural é preciso definir os neurônios de entrada e de saída, o número de camadas ocultas, o número de neurônios por camada, os parâmetros de aprendizagem, o momento e o número de *epochs* a se realizar; veja seção 3.3.4 e 3.3.5. Os neurônios de entrada à rede são determinados segundo a proposta de Varfis & Versino (1990). A idéia é incluir, como variáveis de entrada, variáveis defasadas da resposta de interesse, como também componentes de tendência ou sazonalidade que ajudem a rede a capturar a estrutura da série. Para a construção das nossas redes neurais, só a componente da tendência será extraída das respectivas séries, utilizando a descomposição do modelo *ARIMA*, como é comentado na seção 2.3.4. O número de variáveis defasadas a serem consideradas dependerá do análise sobre a respectiva série.

As camadas ocultas determinam a capacidade de generalização da rede. Em teoria, as redes neurais com uma camada oculta e um número suficiente de neurônios podem aproximar qualquer função contínua. Na prática, as redes neurais com uma ou duas camadas ocultas são amplamente utilizadas e têm um bom desempenho. É assim que, seguindo a proposta de Kaastra & Boyd (1996), recomenda-se que todas as redes neurais utilizem preferivelmente uma ou duas camadas ocultas, já que, tanto na teoria como na experimentação, as redes neurais com mais de três camadas ocultas não alcançam melhorias. Com este ponto em mente, serão consideradas no máximo duas camadas ocultas. Como comentado por Kaastra & Boyd (1996), a seleção do melhor número de neurônios ocultos envolve experimentação; a regra que é sempre utilizada para determinar esse número é selecionar a rede que tenha o melhor desempenho sobre o conjunto do teste (*testing set*) com o mínimo número de neurônios. Desta forma, em nosso caso, um número máximo de 6 neurônios ocultos será suficiente para obter resultados aceitáveis, já que um aumento deste número, somado ao número de camadas ocultas, pode aumentar o número de parâmetros a serem estimados, conduzindo a um sobre-ajuste.

A função de transferência comumente utilizada é a tangente hiperbólica, veja Klimasauskas (1994), porém o pacote *AMORE* utiliza uma função hiperbólica mais geral, que pode ajudar ao algoritmo de retropropagação a aprender mais rapidamente, veja Haykin (1994). Os parâmetros de aprendizagem e momento a serem considerados são $\{0.01, 0.1\}$ e $\{0.1, 0.5\}$, respectivamente; veja Haykin (1994). O número de *epochs* ou iterações de treinamento a ser utilizado em geral será de 10000, já que um aumento deste incrementa igualmente o tempo de estimação da rede neural; Kaastra & Boyd (1996) mencionam que epochs de 50000 até 191400 têm sido utilizadas na prática, com tempos de treinamento de até 60 horas. No entanto, o aumento no número de iterações de treinamento ou *epochs* não melhora necesariamente os resultados, já que o treinamento é afetado por muitos outros parâmetros, tal como a escolha da constante de aprendizagem e momento, as variáveis de entrada e a função de transferência.

Na modelagem através de redes neurais de cada uma das séries, o procedimento é como descrito a seguir. Primeiro, cada série transformada é re-escalada no intervalo $[-1, 1]$, antes de incluir variáveis defasadas como neurônios de entrada à rede. Diferentes conjuntos de variáveis defasadas são considerados como entrada à rede, com a finalidade de se obter um bom ajuste, e também a melhor previsão possível. Segundo, como comentado anteriormente, 2 camadas ocultas serão consideradas, 2 parâmetros de aprendizagem, 2 parâmetros de momento e um máximo de 6 neurônios por camada oculta. A função de transferência, como descrito antes, é a função hiperbólica anti-simétrica; veja Haykin (1994). Três classes de arquitetura de redes neurais são definidas:

a.) Rede 1. Somente uma camada oculta é considerada, com um número de até 6 neurônios.

b.) Rede 2. Duas camadas ocultas são consideradas, com igual número de neurônios em cada, de até 6 neurônios.

c.) Rede 3. Duas camadas ocultas são consideradas, com um neurônio adicional na segunda camada, com cada camada tendo até 6 neurônios.

Segundo as características das redes anteriores, 24 redes neurais são simuladas para cada arquitetura com o objetivo de obter as melhores previsões um passo e múltiplos passos à frente. 10000 *epochs* são consideradas na simulação de cada rede neural. Para o melhor ajuste da rede, um neurônio de entrada, descrevendo a tendência da respectiva série, será ingressado na rede. Para as previsões de valores futuros, igualmente precisaremos de previsões desta tendência. O pacote `AMORE` do programa `R` será utilizado na identificação e estimação dos parâmetros da rede neural; veja apêndice `A`.

A figura 4.8 exibe as tendências extraídas das séries de interesse, para o período entre julho de 1994 e novembro 2004, utilizando a descomposição do modelo *ARIMA*. Segundo estas figuras, até meados de 1998 parece não haver tendência em cada uma das séries, com exceção de *ICMSRJ*; porém, a partir da metade de 1998, o nível de arrecadação parece aumentar como o tempo.

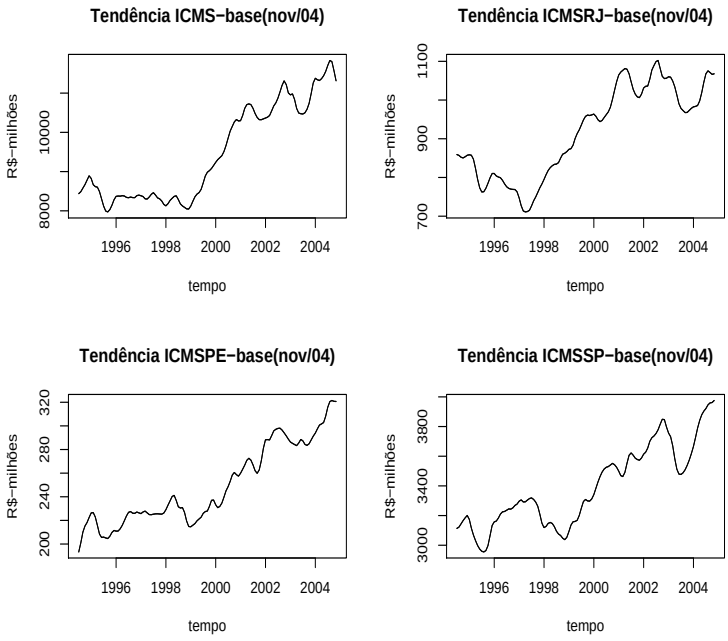

Figura 4.8: Tendências das séries de arrecadação.

Para esta primeira análise, o interesse recai sobre a previsão da arrecadação do mês de dezembro de 2004 fazendo uso do conjunto de treinamento (jul/94-nov/04). Ainda que, dezembro de 2004 seja a prioridade nesta primeira parte, as previsões de janeiro até março de 2005 também serão avaliadas. As tabelas 4.16 a 4.19 apresentam os resultados das redes com melhor desempenho, procurando-se obter o menor *MAPE* de previsão para dezembro de 2004. Os resultados são expressos em milhões de reais. A fim de designar a rede neural identificada, será utilizada a notação de Souza & Zandonade (1993), dada por $ANN(n_1, n_2, \ldots, n_C)$, em que $n_c,\ c = 1, 2, \ldots, C$ é o número de neurônios na camada c.

Tabela 4.16: Previsão do *ICMS* através de *ANN*(16,5,5,1).

data	real	prev-o	er(%)	prev-m	er(%)
dez/04	11746.0785	11737.6244	−0.0720	11737.6244	−0.0720
jan/05	11331.6471	11740.5592	3.6086	11605.7358	2.4188
fev/05	11042.4777	10700.7030	−3.0951	10648.5043	−3.5678
mar/05	11109.9315	10679.7502	−3.8720	10706.4830	−3.6314

Tabela 4.17: Previsão do *ICMSPE* através de *ANN*(16,2,3,1).

data	real	prev-o	er(%)	prev-m	er(%)
dez/04	337.2400	335.1488	−0.6201	335.1488	−0.6201
jan/05	368.9698	338.1856	−8.3433	339.0965	−8.0964
fev/05	339.0253	322.8264	−4.7781	336.8389	−0.6449
mar/05	303.3947	317.6133	4.6865	318.9310	5.1208

Tabela 4.18: Previsão do *ICMSRJ* através de *ANN*(4,2,3,1).

data	real	prev-o	er(%)	prev-m	er(%)
dez/04	1155.0683	1081.1869	−6.3963	1081.1869	−6.3963
jan/05	1269.0384	1050.0517	−17.2561	1069.1452	−15.7515
fev/05	1060.0536	1012.4665	−4.4891	1075.2352	1.4322
mar/05	1033.2296	1043.5511	0.9990	1075.7945	4.1196

Tabela 4.19: Previsão do *ICMSSP* através de *ANN*(14,1,1,1).

data	real	prev-o	er(%)	prev-m	er(%)
dez/04	4148.0546	4146.8933	−0.0280	4146.8933	−0.0280
jan/05	4188.3608	4152.2047	−0.8633	4152.3708	−0.8593
fev/05	3844.5995	3740.9869	−2.6950	3746.3403	−2.5558
mar/05	3942.9511	3803.5295	−3.5360	3819.4233	−3.1329

Para o *ICMS*, tem-se uma rede neural com 16 neurônios de entrada, definida pelas três primeiras defasagens da variável, a tendência e doze variáveis dummy que identificam cada mês do ano. A rede da classe 2, com 5 neurônios, $\eta = 0.01$ e $\alpha = 0.1$ é a que fornece a melhor previsão. Pode-se observar que o erro relativo da previsão do mês de dezembro de 2004 é 0.07%, enquanto que as demais previsões registram erros relativos mais elevados. Para o *ICMSPE*, a rede com 16 neurônios de entrada é definida pelas defasagens de primeira, segunda e décima-segunda ordem, pela tendência e por doze variáveis dummy identificando cada mês do ano. A rede da classe 3, com 2 neurônios, $\eta = 0.01$ e $\alpha = 0.5$ é a que melhor prevé o mês de dezembro, apresentando um erro relativo de 0.62%, o menor valor obtido entre o conjunto de redes neurais simuladas. Além disso, as outras previsões exibem erros relativos maiores na previsão um passo à frente que na previsão múltiplos passos à frente, com exceção do mês de março de 2005.

Como visto anteriormente, não há sazonalidade evidente na série *ICMSRJ*; portanto, as variáveis dummy não serão incluídas na análise. A rede neural com 4 neurônios de entrada, definida pelas primeiras três defasagens da série *ICMSRJ* e pela tendência, exibe os melhores resultados. Esta rede corresponde à classe 3, com 2 neurônios, $\eta = 0.01$ e $\alpha = 0.5$. A previsão do mês de dezembro apresenta erro relativo de 6.39%, o menor obtido sobre o conjunto total de redes neurais consideradas. A previsão do mês de janeiro de 2005 parece não ser bem capturada pelo modelo de redes neurais. A rede neural do *ICMSSP* é identificada com 14 neurônios de entrada, determinados pela primeira defasagem, pela tendência e por doze variáveis dummy

como antes; ela pertence à rede da classe 2, com um neurônio, $\eta = 0.01$ e $\alpha = 0.5$. O erro relativo de previsão para dezembro de 2004 é de 0.02%. As demais previsões exibem erros relativos menores que 4%. A tabela 4.20 exibe as medidas de erro da previsão, *MAPE* e *MAD*, para as previsões obtidas por redes neurais.

Tabela 4.20: Medidas de erro das previsões de *ANN*: um e quatro passos à frente.

	MAPE(%)		*MAD*	
Série	prev-o	prev-m	prev-o	prev-m
ICMS	2.6619	2.4225	210.8636	197.4972
ICMSPE	4.6070	3.6205	14.3465	8.8613
ICMSRJ	7.2851	6.9249	42.1014	58.2231
ICMSSP	1.7806	1.6440	51.2257	43.7689

4.3.4 Análise comparativa das previsões

As previsões obtidas por alisamento exponencial, pelo método de Box e Jenkins e por redes neurais encontram-se apresentadas nas tabelas 4.4 a 4.8, 4.11 a 4.15 e 4.16 a 4.20, respectivamente. Em particular, a previsão do mês de dezembro de 2004 proporcionada pelas redes neurais exibe o menor erro relativo de previsão.

Além disso, a medida de ajuste calculada sobre os resíduos do modelo associada às redes neurais é menor que a proporcionada pelo alisamento exponencial e pelo método de Box e Jenkins. As tabelas 4.21 e 4.22 exibem estes resultados.

Tendo escolhido as redes neuronais sobre este fato (mimimizar o erro de previsão do mês de dezembro de 2004), as previsões dos outros meses (janeiro até

Tabela 4.21: *er(%)* da previsão de dezembro de 2004.

Série	*SARIMA*	*HW*	*ANN*
ICMS	1.6911	0.3556	−0.0720
ICMSRJ	−7.3897	−9.0363	−6.3963
ICMSPE	−1.8379	−0.5903	−0.6201
ICMSSP	−1.2669	0.2172	−0.0280

Tabela 4.22: Medidas de ajuste: *HW*, *SARIMA* e *ANN*.

Modelo	Medida	*ICMS*	*ICMSRJ*	*ICMSPE*	*ICMSSP*
HW	*MAPE*	2.8962	5.3497	4.3486	4.4110
	MAD	188.2510	36.5261	8.4939	107.1198
SARIMA	*MAPE*	2.5951	4.3923	4.2231	3.1867
	MAD	195.2077	40.8097	8.0884	80.3654
ANN	*MAPE*	1.8314	4.2029	2.6837	1.7626
	MAD	119.5066	24.5098	5.3346	46.1128

março de 2005) são melhor descritas, segundo o *MAPE*, pelo alisamento exponencial na previsão um e múltiplos passos à frente; no entanto, o *MAPE* das previsões um passo à frente do *ICMSSP* sobre o modelo *SARIMA* é menor que o proveniente de outros métodos. As previsões múltiplos passos, obtidas pela rede neural para o *ICMSSP*, são melhores que as proporcionadas pelos outros dois métodos. Por outro lado, o *MAD* exibido pelas previsões múltiplos passos à frente do alisamento exponencial para todas as séries é menor que o proporcionado pelos outros métodos. As previsões da rede neural apresentam os menores *MAD* para o *ICMS* e o *ICMSRJ* na previsão um passo à frente. O *MAD* das previsões um passo à frente proporcionadas pelo modelo *SARIMA* são menores para o *ICMSSP*. Os *MAD* para as previsões um e quatro passos à frente, determinadas pelo alisamento exponencial do *ICMSPE* são as menores.

4.3.5 Combinação de previsões

Uma combinação de previsões elaborada cuidadosamente pode aproximar melhor os verdadeiros valores de interesse do que previsões individuais, veja Hendry & Clements (2004). Com essa motivação, a média ponderada das previsões individuais proporcionadas pelo alisamento exponencial, pela método de Box e Jenkins e por redes neurais será utilizada conjuntamente para se obter uma previsão melhorada. Os pesos de ponderação serão determinados pelos erros de previsão, com a condição que

a soma dos respectivos pesos seja um. As tabelas 4.23 a 4.26 exibem as previsões um e quatro passos à frente do *ICMS*, *ICMSPE*, *ICMSRJ* e *ICMSSP*, considerando a combinação das previsões dos três métodos, como também as previsões dois a dois. As previsões para cada mês, resultado das combinações, são apresentadas no apêndice B.

Tabela 4.23: Combinação de previsões *HW-SARIMA-ANN*.

Série	*MAPE(%)*		*MAD*	
	prev-o	prev-m	prev-o	prev-m
ICMS	1.7714	1.7988	179.7358	121.9652
ICMSPE	3.1962	3.1784	10.4816	9.0475
ICMSRJ	3.5286	2.1943	21.4649	6.5727
ICMSSP	1.0777	1.9176	17.7254	45.8953

Tabela 4.24: Combinação de previsões *HW-SARIMA*.

Série	*MAPE(%)*		*MAD*	
	prev-o	prev-m	prev-o	prev-m
ICMS	1.9899	1.8212	159.8329	169.9068
ICMSPE	3.7570	3.3844	9.9388	9.3197
ICMSRJ	2.7829	3.1063	13.1213	23.0903
ICMSSP	1.0741	2.3293	19.9653	45.9153

Tabela 4.25: Combinação de previsões *HW-ANN*.

Série	*MAPE(%)*		*MAD*	
	prev-o	prev-m	prev-o	prev-m
ICMS	2.0587	1.6923	189.0106	108.0336
ICMSPE	2.9754	3.1856	9.3999	9.5904
ICMSRJ	3.4875	2.9548	37.0300	48.1423
ICMSSP	1.8061	1.9480	42.4313	53.9989

Tabela 4.26: Combinação de previsões *ANN-SARIMA*.

Série	*MAPE(%)*		*MAD*	
	prev-o	prev-m	prev-o	prev-m
ICMS	1.9600	2.5064	199.2611	204.5418
ICMSPE	3.4888	3.4014	9.8322	8.4154
ICMSRJ	6.1263	6.8481	45.0786	58.9273
ICMSSP	0.9752	1.7391	15.3577	40.4738

Das tabelas anteriores é possível destacar a notável melhoria das previsões do *ICMSRJ* a partir da combinação de previsões. Neste caso, na previsão um passo à frente, os melhores resultados são exibidos pela combinação *HW-SARIMA-ANN*, como também na previsão quatro passos à frente, com relação às outras combinações. Em comparação com as previsões individuais, a combinação *HW-SARIMA-ANN* melhora as previsões ainda mais, apresentando menores valores do *MAPE*, para todas as séries, na previsão um e quatro passos à frente. A combinação *HW-SARIMA* exibe resultados favoráveis unicamente para o *ICMSRJ*.

4.4 Modelagem e Previsão: Análise II

Nesta segunda parte, as previsões referentes ao período que se estende de julho a dezembro de 2004 são obtidas através de alisamento exponencial, da metodologia de Box e Jenkins e via redes neurais.

4.4.1 Alisamento exponencial

As constantes de suavização são determinadas minimizando-se a soma de quadrados dos erros na previsão um passo à frente. As constantes de suavização dos modelos aditivo e multiplicativo são apresentadas na tabela 4.27. Os *MAPE* das previsões através dos métodos multiplicativo e aditivo são semelhantes. O método aditivo será o considerado nos resultados. As tabelas 4.28 a 4.31 exibem as previsões um e seis

passos à frente do alisamento exponencial, enquanto a tabela 4.32 fornece as medidas de precisão, *MAPE* e *MAD*, sobre as previsões de cada série. Os valores encontram-se expressos em milhões de reais.

Tabela 4.27: Parâmetros de suavização de alisamento exponencial.

Série	HW- aditivo			HW- multiplicativo		
	α	β	γ	α	β	γ
ICMS	0.7887	0.0000	0.5899	0.7859	0.0000	0.6015
ICMSPE	0.3199	0.0005	0.4204	0.3221	0.0006	0.4278
ICMSRJ	0.3725	0.0000	0.5033	0.3752	0.0000	0.5064
ICMSSP	0.2554	0.0007	0.4284	0.2520	0.0009	0.4256

Tabela 4.28: Previsão do *ICMS* através de alisamento exponencial.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	11512.3622	11177.4621	−2.9090	11177.4621	−2.9090
ago/04	12110.8247	11263.5229	−6.9962	11004.3367	−9.1364
set/04	12208.2788	11992.4759	−1.7677	11064.9167	−9.3655
out/04	12127.8133	12331.6539	1.6808	11219.0075	−7.4936
nov/04	11043.4190	12269.5360	11.1027	11310.2505	2.4162
dez/04	11746.1057	11850.0187	0.8847	11869.3499	1.0492

Tabela 4.29: Previsão do *ICMSPE* através de alisamento exponencial.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	309.9951	295.0647	−4.8163	295.0647	−4.8163
ago/04	315.8774	292.3216	−7.4573	287.7393	−8.9079
set/04	322.1099	313.9754	−2.5254	301.4773	−6.4055
out/04	319.6628	312.4438	−2.2583	297.5537	−6.9164
nov/04	326.7480	330.5456	1.1623	312.4907	−4.3634
dez/04	337.2409	334.9914	−0.6670	317.8580	−5.7475

Tabela 4.30: Previsão do *ICMSRJ* através de alisamento exponencial.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	1107.9730	1037.8274	−6.3310	1037.8274	−6.3310
ago/04	1117.9948	1072.0036	−4.1137	1046.2051	−6.4213
set/04	1074.0076	1035.4327	−3.5917	994.8328	−7.3719
out/04	1023.8763	1040.1755	1.5919	985.8604	−3.7129
nov/04	1087.5180	1083.4273	−0.3761	1032.9094	−5.0214
dez/04	1155.0671	1051.4808	−8.9680	1001.0500	−13.3340

Tabela 4.31: Previsão do *ICMSSP* através de alisamento exponencial.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	3832.8642	3899.5433	1.7397	3899.5433	1.7397
ago/04	3979.9547	3854.4543	−3.1533	3871.4900	−2.7253
set/04	4102.8901	4022.7246	−1.9539	4007.5474	−2.3238
out/04	4068.6564	4267.4627	4.8863	4229.9892	3.9653
nov/04	4175.0360	4056.4144	−2.8412	4070.0669	−2.5142
dez/04	4148.0694	4167.9297	0.4788	4151.2913	0.0777

Note na tabela 4.32 que as previsões um passo à frente apresentam os menores *MAPE* para o *ICMS*, o *ICMSPE* e o *ICMSRJ*, enquanto que o *MAPE* é menor nas previsões múltiplos passos para o *ICMSSP*. Com relação ao *MAD*, os menores desvios ocorrem nas previsões múltiplos passos das séries *ICMSPE*, *ICMSRJ* e *ICMSSP*; no entanto, a previsão um passo à frente para o *ICMS* exibe o menor desvio.

Tabela 4.32: Medidas de erro das previsões de alisamiento exponencial: um e seis passos à frente.

	MAPE(%)		*MAD*	
Série	prev-o	prev-m	prev-o	prev-m
ICMS	4.2235	5.3950	269.3703	503.0721
ICMSPE	3.1478	6.1928	6.3404	3.5893
ICMSRJ	4.1621	7.0321	33.0275	12.2830
ICMSSP	2.5089	2.2243	91.9084	60.6565

4.4.2 Método de Box e Jenkins

As séries deflacionadas são utilizadas para ajustar o modelo *SARIMA* e obter as previsões um e seis passos à frente. As séries são transformadas através do logaritmo natural para evitar possíveis flutuações. As figuras 4.9 a 4.12 exibem as funções de autocorrelação (`FAC`) e autocorrelação parcial (`FACP`) amostrais das séries no período de julho de 1994 a junho de 2004. Nesses gráficos o valor de 1.0 corresponde à defasagem de ordem 12, 2.0 à defasagem de ordem 24 e assim por diante.

As funções de autocorrelação das séries decaem rápido, exibindo em alguns casos, estruturas sazonais como é o caso do *ICMS*, em que as defasagens de ordem 12 e 24 destacam dois bicos de autocorrelação; o mesmo acontece com o *ICMSPE* e o *ICMSSP*. Já para o *ICMSRJ*, parece não haver algum comportamento na `FAC` que permita determinar a existência de alguma estrutura sazonal. O teste aumentado de Dickey-Fuller (`ADF`) é utilizado para testar a não-estacionariedade das séries sobre estas defasagens. A tabela 4.33 contém os respectivos resultados.

Tabela 4.33: Testes de estacionariedade.

série	defasagem	ADF	p valor
ICMS	$\Delta\Delta^{12}$	-5.1960	< 0.01
ICMSPE	$\Delta\Delta^{12}$	-5.1581	< 0.01
ICMSRJ	Δ	-7.1768	< 0.01
ICMSSP	$\Delta\Delta^{12}$	-5.8141	< 0.01

A partir da observação das figuras 4.9 a 4.12, consideramos a existência de estruturas sazonais nas séries *ICMS*, *ICMSPE* e *ICMSSP*. O modelo escolhido para o *ICMS* é o $SARIMA(2,1,0) \times (0,1,1)_{12}$ o qual apresenta o menor critério de informação, `BIC`$= -6.6831$, sobre o conjunto de modelos considerados no processo de seleção. O modelo identificado para o *ICMSPE* é o $SARIMA(1,1,0) \times (0,1,1)_{12}$ com `BIC`$=-5.6648$; o *MAPE* das previsões proporcionadas por este modelo são as melhores em comparação às dadas por outros modelos tentativos. Para o *ICMSSP* o modelo

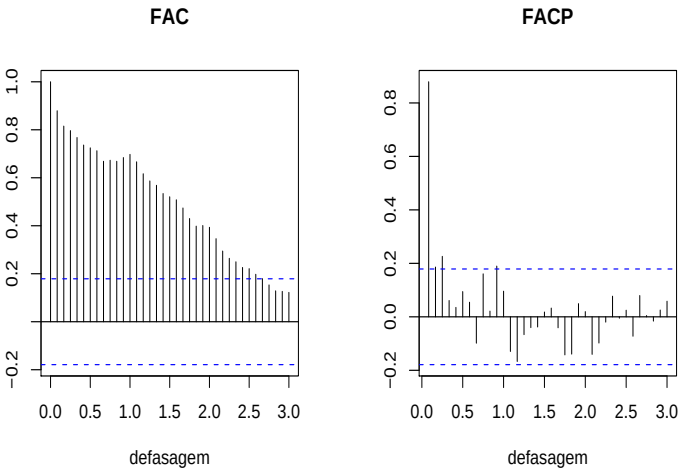

Figura 4.9: ACF e PACF: Série *ICMS*.

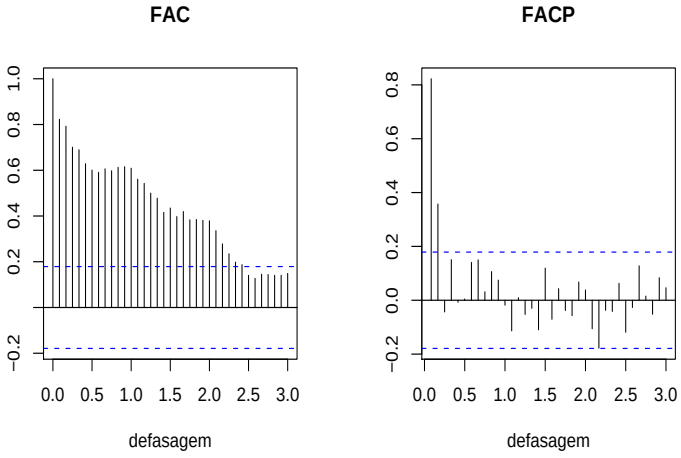

Figura 4.10: ACF e PACF: Série *ICMSPE*.

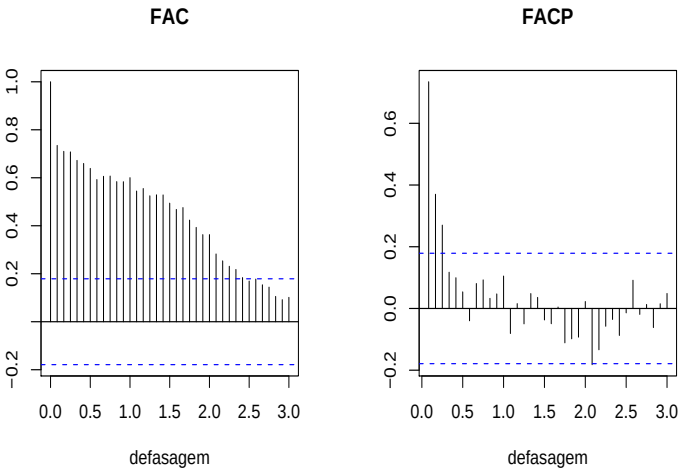

Figura 4.11: ACF e PACF: Série *ICMSRJ*.

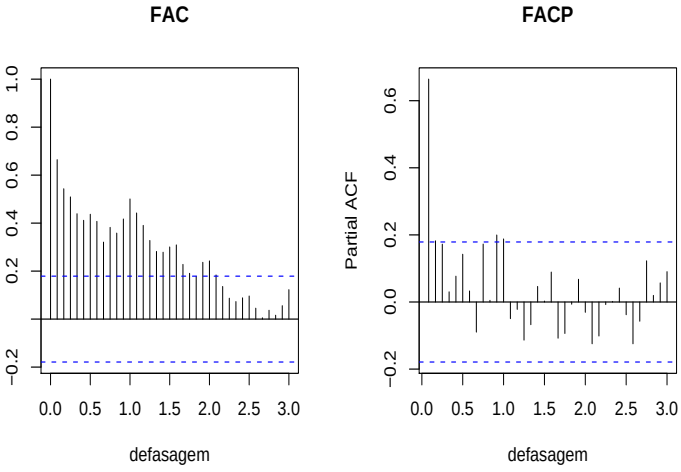

Figura 4.12: ACF e PACF: Série *ICMSSP*.

Tabela 4.34: Modelos escolhidos e estatísticas de diagnóstico.

Série	*ICMS*	*ICMSPE*	*ICMSRJ*	*ICMSSP*
Modelo $(p,d,q)\times(P,D,Q)_s$	SARIMA $(2,1,0)\times(0,1,1)_{12}$	SARIMA $(1,1,0)\times(0,1,1)_{12}$	ARIMA $(0,1,2)$	SARIMA $(0,1,1)\times(0,1,1)_{12}$
SE(Res)	0.0325	0.0550	0.0552	0.0415
BIC	-6.6832	-5.6649	-5.6041	-6.0261
Q-Val	10.1300	18.8500	19.4200	13.4000
N-test	0.0210	2.3400	3.1200	6.6400
SK(t)	-0.1300	0.4510	0.5970	-2.3200
KUR(t)	-0.0500	1.4600	1.6600	1.1300
Q2	23.0600	28.2400	13.7900	25.7600
RUNS	0.9810	-0.3900	-0.5600	-0.6100

Quantiles: $\chi^2_2(1\%) = 9.21$, $\chi^2_2(5\%) = 6.00$, $\chi^2_{21}(5\%) = 32.67$, $\chi^2_{22}(5\%) = 33.92$, $\chi^2_{24}(5\%) = 36.41$

$SARIMA(0,1,1)\times(0,1,1)_{12}$ é identificado com BIC$= -6.0260$. Por último, o melhor modelo identificado para o *ICMSRJ* é dado pelo $ARIMA(0,1,2)$ com BIC$= -5.6041$.

Identificação, correção de outliers e estimação dos parâmetros do modelo *SARIMA* para cada uma das séries é conduzida utilizando o programa *TRAMO-SEATS*. A tabela 4.34 exibe uma análise de diagnóstico para cada um dos modelos.

A tabela 4.34 exibe os modelos selecionadas para cada série, como também medidas de diagnóstico sobre os resíduos de cada um deles, em que SE(Res) é o erro padrão dos resíduos; $\mathtt{Q}-\mathtt{Val}$ é a estatística de Ljung-Box para testar correlação residual, calculada sobre 24 autocorrelações (em todos os casos, a distribuição assintótica χ^2 é utilizada com 21, 22, 22 e 22 g.l., respectivamente); $\mathtt{N}-\mathtt{test}$ é a estatística de teste de Bowman-Shenton para normalidade (em todos os casos, a distribuição assintótica χ^2(2 g.l.) é utilizada); SK(t) é a estatística usada para testar se a assimetria é zero contra se é diferente de zero; KUR(t) é a estatística usada para testar se o excesso de curtose é zero contra se é diferente de zero; Q2 é a estatística de McLeod & Li (1983) para testar a linearidade do processo (em todos os casos, a distribuição assintótica χ^2(24 g.l.) é utilizada); por último, RUNS é a estatística usada para testar a hipótese nula sobre aleatoriedade nos sinais dos resíduos. Todos os testes são realizados ao

nível de significância de 5%. Pode-se concluir que cada conjunto de resíduos é um ruído branco normal, exceto que a hipótese de normalidade dos resíduos do modelo identificado da série *ICMSSP* não é rejeitada ao nível de significância de 1%.

A equação do modelo estimado da série do logaritmo do *ICMS*, denotada por Z_t, é dada por

$$Z_t = \underset{(-6.27)}{-0.2145} I_t^{(Jul/94)} + \underset{(4.82)}{0.1343} I_t^{(Mar/95)} + X_t,$$

em que

$$\underset{(4.68)}{(1 - 0.3300B - 0.2098B^2)}\underset{(4.61)}{(1 - B)}(1 - B^{12})X_t = \underset{(-141.69)}{(1 + 0.9973B^{12})}a_t,$$

$$a_t \sim N(0, (0.0325)^2).$$

O modelo estimado da série do logaritmo do *ICMSPE*, denotada por Z_t, é dada por

$$Z_t = \underset{(4.82)}{-0.1470} I_t^{(Out/95)} + \frac{\underset{(-3.49)}{0.2016}}{1 - 0.7B} I_t^{(dez/99)} + X_t,$$

em que

$$\underset{(10.20)}{(1 - 0.6942B)}(1 - B)(1 - B^{12})X_t = \underset{(7.21)}{(1 + 0.7131B^{12})}a_t,$$

$$a_t \sim N(0, (0.055)^2).$$

O modelo estimado da série do logaritmo do *ICMSRJ*, denotada por Z_t, é dada por

$$Z_t = \underset{(12.22)}{0.5834} I_t^{(jan/04)} \underset{(-4.92)}{-0.2715} I_t^{(jul/94)} + \underset{(4.93)}{0.2356} I_t^{(abr/98)} + \underset{(3.45)}{0.1647} I_t^{(ago/02)} + X_t,$$

em que

$$(1-B)X_t = (1 + 0.5718B - 0.1108B^2)a_t,$$

$$(-7.00) \quad (\ 81.41)$$

$$a_t \sim N(0, (0.0552)^2).$$

A equação do modelo estimado da série do logaritmo do *ICMSSP*, denotada por Z_t, é dada por

$$Z_t = \underset{(5.08)}{0.1676I_t^{(out/02)}} + \underset{(5.04)}{0.1746I_t^{(dez/03)}}$$

$$+\underset{(4.47)}{0.1432I_t^{(nov/00)}} + \underset{(4.55)}{0.1455I_t^{(jun/98)}}$$

$$\underset{(-3.95)}{-0.1640I_t^{(jul/94)}} + \underset{(6.15)}{\frac{0.2360}{1-0.7B}I_t^{(mar/95)}}$$

$$+\underset{(5.25)}{\frac{0.1869}{1-0.7B}I_t^{(mar/01)}} + \underset{(3.36)}{\frac{0.1196}{1-0.7B}I_t^{(ago/00)}} + X_t,$$

em que

$$(1-B)(1-B^{12})X_t = (1 + \underset{(-4.02)}{0.3870B})(1 + \underset{(-6.44)}{0.6804B^{12}})a_t,$$

$$a_t \sim N(0, (0.0415)^2).$$

As previsões para o período entre julho e dezembro de 2004 são exibidas nas tabelas 4.35 a 4.38. Os resultados estão expressos em milhões de reais.

Tabela 4.35: Previsão do *ICMS* através do modelo $SARIMA(2,1,0) \times (0,1,1)_{12}$.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	11512.3622	11420.2654	−0.8000	11420.2654	−0.8000
ago/04	12110.8247	11463.0571	−5.3487	11412.2740	−5.7680
set/04	12208.2788	12099.9878	−0.8870	11639.3249	−4.6604
out/04	12127.8133	12199.5424	0.5914	11718.7420	−3.3730
nov/04	11043.4190	12301.5644	11.3927	11821.1399	7.0424
dez/04	11746.1057	11884.6237	1.1793	12302.3394	4.7355

Tabela 4.36: Previsão do *ICMSPE* através do modelo $SARIMA(1,1,0) \times (0,1,1)_{12}$.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	309.9951	288.4278	−6.9573	288.4278	−6.9573
ago/04	315.8774	302.7755	−4.1478	296.1733	−6.2379
set/04	322.1099	318.7486	−1.0435	297.3009	−7.7020
out/04	319.6628	326.2858	2.0719	304.4615	−4.7554
nov/04	326.7480	333.9749	2.2118	311.2339	−4.7480
dez/04	337.2409	331.1087	−1.8183	315.3694	−6.4854

Tabela 4.37: Previsão do *ICMSRJ* através do modelo $ARIMA(0,1,2)$.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	1107.9730	1020.3946	−7.9044	1020.3946	−7.9044
ago/04	1117.9948	1067.5694	−4.5103	1030.6498	−7.8127
set/04	1074.0076	1098.8558	2.3136	1030.6498	−4.0370
out/04	1023.8763	1093.7306	6.8225	1030.6498	0.6616
nov/04	1087.5180	1060.5668	−2.4782	1030.6498	−5.2292
dez/04	1155.0671	1064.2065	−7.8663	1030.6498	−10.7714

Tabela 4.38: Previsão do *ICMSSP* através do modelo $SARIMA(0,1,1) \times (0,1,1)_{12}$.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	3832.8642	3969.2988	3.5596	3969.2988	3.5596
ago/04	3979.9547	3823.0535	−3.9423	3905.9047	−1.8606
set/04	4102.8901	4082.0791	−0.5072	4068.9682	−0.8268
out/04	4068.6564	4206.9017	3.3978	4180.3269	2.7447
nov/04	4175.0360	4108.1814	−1.6013	4166.5545	−0.2031
dez/04	4148.0694	4092.4608	−1.3406	4109.8620	−0.9211

A tabela 4.39 contém medidas de precisão sobre as previsões dos modelos identificados para cada série. Notamos da tabela 4.39 que as previsões um passo à frente apresentam o menor *MAPE* para o *ICMS*, o *ICMSPE* e o *ICMSRJ*, enquanto que para o *ICMSSP* as previsões seis passos à frente exibem o menor *MAPE*. Com relação ao *MAD* os menores desvios são obtidos pelas previsões um passo à frente do *ICMS* e múltiplos passos, do *ICMSPE*, *ICMSRJ* e *ICMSSP*.

Tabela 4.39: Medidas de erro das previsões *SARIMA*: um e seis passos à frente.

	MAPE(%)		*MAD*	
Série	prev-o	prev-m	prev-o	prev-m
ICMS	3.3665	4.3965	123.4046	383.1682
ICMSPE	3.0418	6.1477	9.8625	2.7045
ICMSRJ	5.3159	6.0694	50.5311	22.1103
ICMSSP	2.3915	1.6860	73.6681	34.9270

4.4.3 Redes neurais artificiais

O processo de identificação das redes neurais é feito de forma semelhante ao considerado na análise I. Neste caso, o conjunto de treinamento, para cada série, será o período de julho de 1994 a junho de 2004 e o conjunto de teste será o período de julho a dezembro de 2004. As três classes de arquiteturas de redes neurais são: Rede 1, Rede 2 e Rede 3, como definidas na seção 4.3.3. Os valores dos parâmetros de aprendizagem e momento dados por $\{0.01, 0.1\}$ e $\{0.1, 0.5\}$, respectivamente, serão utilizados. Além disso, 10000 epochs serão consideradas no treinamento de cada rede neural.

Procura-se obter redes neurais que exibam as melhores medidas de precisão: *MAPE* e *MAD*, sobre as previsões do conjunto de teste; além disso, um bom ajuste sobre o conjunto de treinamento é necessário para evitar o sobre-ajuste. A proposta de Varfis & Versino (1990) é utilizada novamente na identificação das redes neurais. Neste caso, a tendência de cada uma das séries é extraída do modelo *SARIMA* para

ser um neurônio de entrada. As redes identificadas inicialmente através deste processo apresentam menores medidas de erro na previsão múltiplos passos à frente em comparação com as proporcionadas pela metodologia de Box e Jenkins e com alisamento exponencial. Este é o caso da rede $ANN(15,1,1)$ com $\eta = 0.1$ e $\alpha = 0.1$ da série *ICMS*, em que as primeiras duas defasagens são utilizadas, juntamente com a tendência e com 12 variáveis dummy, identificando cada mês do ano. Para o *ICMSPE*, a rede $ANN(16,1,1)$ com $\eta = 0.1$, $\alpha = 0.5$ e as três primeiras defasagens, exibe as menores medidas de erro nas previsões múltiplos passos, em comparação às dadas pelas outras duas metodologias. Para o *ICMSRJ*, a rede $ANN(3,1,1,1)$ com as duas primeiras defasagens, $\eta = 0.01$ e $\alpha = 0.5$, apresenta menores medidas de erro na previsão múltiplos passos; quanto ao *ICMSSP*, a rede $ANN(15,1,1,1)$, com $\eta = 0.01$, $\alpha = 0.5$ e as duas primeiras defasagens apresenta um ligeiro aumento no *MAPE* da previsão múltiplos passos, mas uma diminuição considerável na medida do erro das previsões um passo à frente, em comparação com as outras metodologias.

A tabela 4.40 exibe *MAPE* e *MAD* para cada uma das redes descritas anteriormente, podendo ser comparadas com as respectivas medidas referenciadas nas tabelas 4.32 e 4.39. Igualmente, as tabelas 4.41 a 4.44 contém as previsões para cada série. Os resultados são expressos em milhões de reais. É possível observar que o *MAPE* das previsões múltiplos passos à frente das redes neurais são menores que os dados por *SARIMA* e por alisamento exponencial de Holt-Winters. Através do *MAD* notamos que os desvios obtidos com as redes neurais são menores para o *ICMS* na previsão múltiplos passos à frente em comparação com as outras metodologias; quanto à rede neural do *ICMSPE* menores desvios na previsão um e múltiplos passos à frente são obtidos; para a rede neural do *ICMSRJ* e do *ICMSSP*, isto acontece na previsão um passo à frente. O melhor desempenho das previsões múltiplos passos à frente através das redes neurais é coerente com as conclusões obtidas por Fernandes (1995).

Tabela 4.40: Medidas de erro das previsões através de ANN com tendência.

Série	*MAPE(%)*		*MAD*	
	prev-o	prev-m	prev-o	prev-m
ICMS	4.2351	3.9608	361.2909	367.1399
ICMSPE	6.2879	5.5972	3.4639	2.5865
ICMSRJ	7.4286	6.0430	29.7292	22.8573
ICMSSP	1.6654	1.7574	49.3134	53.8354

Tabela 4.41: Previsão do *ICMS* através de *ANN*(15,1,1).

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	11512.3622	11449.0807	−0.5497	11449.0807	−0.5497
ago/04	12110.8247	11424.6059	−5.6662	11434.3209	−5.5859
set/04	12208.2788	11474.8702	−6.0075	11577.4523	−5.1672
out/04	12127.8133	11494.0492	−5.2257	11618.0444	−4.2033
nov/04	11043.4190	11619.5549	5.2170	11722.2582	6.1470
dez/04	11746.1057	12068.5083	2.7448	11994.1549	2.1118

Tabela 4.42: Previsão do *ICMSPE* através de *ANN*(16,1,1).

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	309.9951	291.2275	−6.0542	291.2275	−6.0542
ago/04	315.8774	292.8014	−7.3054	297.2860	−5.8856
set/04	322.1099	298.6507	−7.2830	301.3356	−6.4494
out/04	319.6628	305.1321	−4.5456	308.1586	−3.5988
nov/04	326.7480	310.3016	−5.0334	311.1468	−4.7747
dez/04	337.2409	311.9287	−7.5057	314.2393	−6.8205

Tabela 4.43: Previsão do *ICMSRJ* através de *ANN*(3,1,1,1).

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	1107.9730	1019.3747	−7.9964	1019.3747	−7.9964
ago/04	1117.9948	1001.7610	−10.3966	1032.4653	−7.6503
set/04	1074.0076	996.3262	−7.2329	1031.1240	−3.9929
out/04	1023.8763	1010.4638	−1.3100	1030.7528	0.6716
nov/04	1087.5180	1030.7425	−5.2206	1030.9796	−5.1988
dez/04	1155.0671	1011.6670	−12.4149	1030.9178	−10.7482

Tabela 4.44: Previsão do *ICMSSP* através de *ANN*(15,1,1,1).

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	3832.8642	3955.1141	3.1895	3955.1141	3.1895
ago/04	3979.9547	3916.1516	−1.6031	3901.2205	−1.9783
set/04	4102.8901	4020.3117	−2.0127	4017.6190	−2.0783
out/04	4068.6564	4084.7047	0.3944	4100.6252	0.7857
nov/04	4175.0360	4067.8697	−2.5668	4072.0210	−2.4674
dez/04	4148.0694	4138.6905	−0.2261	4146.1883	−0.0453

Um outro conjunto de redes foi simulado. A tendência é excluída da entrada da rede para observar se é possível melhorar as previsões. Só as doze variáveis dummy, dependendo da existência de sazonalidade, e as séries defasadas são consideradas como neurônios de entrada à rede. A idéia é deixar as defasagens da série construirem a tendência. Serão escolhidas as redes com o melhor ajuste possível e os menores erros de previsão sobre o conjunto de teste. Para o *ICMS*, a rede *ANN*(14,4,5,1) com $\eta = 0.01$ e $\alpha = 0.1$ é identificada, apresentando os menores *MAPE* para as previsões um passo e múltiplos passos. Neste caso, as duas primeiras defasagens são utilizadas. Para o *ICMSPE*, a rede *ANN*(15,2,1) com as três primeiras defasagens, $\eta = 0.01$ e $\alpha = 0.1$ exibe o menor *MAPE* na previsão múltiplos passos à frente, porém com um valor do *MAPE* um pouco maior na previsão um passo à frente. Para o *ICMSRJ* a rede neural identificada é *ANN*(1,1,1) com $\eta = 0.01$ e $\alpha = 0.1$, com a primeira defasagem da série sendo a entrada da rede. Com esta rede acontece algo semelhante ao verificado para a rede do *ICMSPE* com relação ao *MAPE*. Finalmente, a rede identificada para o *ICMSSP* é *ANN*(15,1,1,1) dada por $\eta = 0.01$ e $\alpha = 0.1$, com as duas primeiras defasagens e a tendência. Deixou-se a tendência nesta última rede, uma vez que nenhuma outra rede sem tendência apresentou melhores resultados na previsão. Esta rede exibe um valor do *MAPE* levemente maior na previsão múltiplos passos à frente em comparação com as obtidas pelas outras metodologias. A tabela 4.45 exibe as medidas de precisão relativas às previsões um passo e múltiplos passos à frente, de onde são obtidas as conclusões comentadas anteriormente. Note também

Tabela 4.45: Medidas de erro das previsões através de ANN sem tendência.

Série	*MAPE(%)*		*MAD*	
	prev-o	prev-m	prev-o	prev-m
ICMS	3.1836	4.3451	215.8590	392.4455
ICMSPE	3.5231	5.6878	143.8273	290.1802
ICMSRJ	4.4673	5.8166	100.5846	245.3943
ICMSSP	1.6654	1.7574	116.2060	267.7710

que o *MAD* das previsões de redes neurais sem tendência são maiores do que obtidas através das redes neurais com tendência. As previsões obtidas das redes neurais estão exibidas nas tabelas 4.46 a 4.49. Todos os resultados são expressos em milhões de reais.

Tabela 4.46: Previsão do *ICMS* através de *ANN*(14,4,5,1).

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	11512.3622	11177.9092	−2.9052	11177.9092	−2.9052
ago/04	12110.8247	11262.1714	−7.0074	11104.8222	−8.3066
set/04	12208.2788	12118.9391	−0.7318	11519.3640	−5.6430
out/04	12127.8133	12225.0783	0.8020	11846.1088	−2.3228
nov/04	11043.4190	11685.5080	5.8142	11714.4069	6.0759
dez/04	11746.1057	11529.8514	−1.8411	11842.0818	0.8171

Tabela 4.47: Previsão do *ICMSPE* através de *ANN*(15,2,1).

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	309.9951	279.9034	−9.7072	279.9034	−9.7072
ago/04	315.8774	308.0755	−2.4699	299.9947	−5.0281
set/04	322.1099	311.9724	−3.1472	302.1957	−6.1824
out/04	319.6628	319.7933	0.0408	311.1250	−2.6709
nov/04	326.7480	329.3776	0.8048	313.7526	−3.9772
dez/04	337.2409	320.4848	−4.9686	315.1141	−6.5611

Tabela 4.48: Previsão do *ICMSRJ* através de *ANN*(1,1,1).

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	1107.9730	1050.4824	−5.1888	1050.4824	−5.1888
ago/04	1117.9948	1071.0607	−4.1981	1037.0182	−7.2430
set/04	1074.0076	1076.6662	0.2475	1028.5494	−4.2326
out/04	1023.8763	1051.3336	2.6817	1023.1432	−0.0716
nov/04	1087.5180	1020.1192	−6.1975	1019.6398	−6.2416
dez/04	1155.0671	1059.3118	−8.2900	1017.3584	−11.9221

Tabela 4.49: Previsão do *ICMSSP* através de *ANN*(15,1,1,1).

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	3832.8642	3955.1141	3.1895	3955.1141	3.1895
ago/04	3979.9547	3916.1516	−1.6031	3901.2205	−1.9783
set/04	4102.8901	4020.3117	−2.0127	4017.6190	−2.0783
out/04	4068.6564	4084.7047	0.3944	4100.6252	0.7857
nov/04	4175.0360	4067.8697	−2.5668	4072.0210	−2.4674
dez/04	4148.0694	4138.6905	−0.2261	4146.1883	−0.0453

4.4.4 Análise comparativa das previsões

Na segunda análise, examinou-se a capacidade das redes neurais para fazer previsões um e múltiplos passos à frente. Utilizando as redes neurais e a tendência como neurônio de entrada, é possível obter redes capazes de fornecer boas previsões múltiplos passos à frente, com relação ao *MAPE* e *MAD*, como aconteceu com as previsões múltiplos passos do *ICMS*, *ICMSPE* e *ICMSRJ*, veja tabela 4.40; no que tange as previsões múltiplos passos do *ICMSSP*, o modelo *SARIMA* exibe os melhores resultados. Nas previsões um passo à frente, as redes neurais são as melhores na previsão do *ICMSSP*; enquanto o modelo *SARIMA* exibe os melhores resultados na previsão um passo à frente para o *ICMS* e o *ICMSPE*; o alisamento exponencial de Holt-Winters (*HW*) apresenta os melhores resultados na previsão um passo à frente para o *ICMSRJ*; veja as tabelas 4.32, 4.39 e 4.40 para fazer as comparações. Com relação ao *MAD*, as redes neurais apresentam, em geral, os menores desvios, tanto

na previsão de um e quanto na de múltiplos passos à frente.

Considerando as redes neurais sem o neurônio de tendência, as previsões múltiplos passos à frente seguem sendo melhores, em comparação com as proporcionadas pelas outras duas metodologias; veja tabelas 4.32, 4.39 e 4.45. No caso do *ICMS*, através das redes neurais, os melhores resultados são obtidos na previsão um e múltiplos passos à frente. O *ICMSSP* também é previsto de forma acurada usando-se redes neurais quando o horizonte de previsão é de um passo; também, as redes neurais conseguem proporcionar resultados aceitáveis na previsão múltiplos passos do *ICMSRJ* e *ICMSPE*. Quando a tendência é retirada da rede neural, os valores do *MAD* sobre os erros de previsão tornam-se mais elevados do que os fornecidos pelas outras metodologias.

4.4.5 Combinação de previsões

Semelhantemente ao que foi realizado na análise I, as previsões proporcionadas por cada método serão combinadas com vistas a se produzir uma melhor previsão de cada série temporal. As tabelas 4.50 a 4.52 exibem as previsões resultantes da combinação de cada um dos dois métodos com a rede neural tendo o neurônio de tendência como entrada (ANN_{ct}); as tabelas 4.53 a 4.55 consideram as mesmas combinações, mas com a rede neural sem o neurônio de tendência como entrada (ANN_{st}). A tabela 4.56 contém os resultados da combinação entre o modelo *SARIMA* e o alisamento exponencial (HW). As previsões para cada mês, resultado das combinações utilizando a rede neural com tendência são apresentadas no apêndice `C`, ao passo que as previsões com a rede neural sem tendência são exibidos no apêndice `D`.

As previsões resultantes de utilizar a rede neural com o neurônio de tendência e os outros dois métodos simultaneamente, mostram-se as mais precisas. Através desta combinação, os valores *MAPE* das previsões um e seis passos à frente para cada série são menores do que aqueles fornecidos por cada um dos métodos individualmente, com a exceção da previsão seis passos à frente do *ICMSPE*, que é superada pela com-

Tabela 4.50: Combinação de previsões *HW-SARIMA-ANN*$_{ct}$.

Série	*MAPE(%)*		*MAD*	
	prev-o	prev-m	prev-o	prev-m
ICMS	2.7433	3.5245	134.7323	431.5789
ICMSPE	2.4658	5.7267	4.8750	2.2824
ICMSRJ	3.6037	6.0000	29.4758	16.2487
ICMSSP	1.2964	1.1329	52.2552	44.3115

Tabela 4.51: Combinação de previsões *HW-ANN*$_{ct}$.

Série	*MAPE(%)*		*MAD*	
	prev-o	prev-m	prev-o	prev-m
ICMS	2.8840	3.7493	191.1808	433.0099
ICMSPE	3.3320	5.5961	6.9937	2.6287
ICMSRJ	4.4897	6.0113	36.4464	13.1767
ICMSSP	1.5312	1.6510	49.2839	52.0715

Tabela 4.52: Combinação de previsões *ANN*$_{ct}$*-SARIMA*

Série	*MAPE(%)*		*MAD*	
	prev-o	prev-m	prev-o	prev-m
ICMS	2.5589	3.9952	144.4007	381.4217
ICMSPE	2.7727	5.8144	8.6705	2.8779
ICMSRJ	4.6655	6.0558	38.8153	22.4821
ICMSSP	1.4083	1.2446	47.3433	39.4720

binação *HW-ANN*$_{ct}$. No caso do *MAD*, as previsões individuais apresentam menores desvios em comparação com a combinação entre eles, com exceção de alguns poucos casos. Os melhores resultados na previsão, segundo o *MAPE*, são os proporcionados pela combinação *HW-SARIMA-ANN*$_{ct}$.

Quando a rede neural sem o neurônio de tendência é utilizada, as combinações das previsões proporcionadas pelas metodologias *HW-SARIMA-ANN*$_{st}$, *HW-ANN*$_{st}$ e *SARIMA-ANN*$_{st}$ exibem os menores valores do *MAPE* na previsão um passo à frente, em comparação com as previsões dadas por cada metodologia individualmente; no entanto, os valores do *MAPE* proporcionados pela combinação *HW-SARIMA-ANN*$_{st}$, no caso do *ICMSPE* e *ICMSSP*, ainda são menores na previsão um passo à frente que os fornecidos pelas combinações *HW-ANN*$_{st}$ e *SARIMA-ANN*$_{st}$. No caso das previsões seis passos à frente, a combinação *HW-SARIMA-ANN*$_{st}$ apresenta as melhores previsões para o *ICMS* e o *ICMSSP*, segundo o *MAPE*; a combinação *HW-ANN*$_{st}$ proporciona as melhores previsões seis passos à frente para o *ICMSPE* e a combinação *SARIMA-ANN*$_{st}$, para o *ICMSRJ*. Com relação ao *MAD*, os menores desvios, na previsão um e seis passos à frente, são distribuídos entre a combinação *HW-ANN*$_{st}$ e *SARIMA-ANN*$_{st}$. Com relação à combinação *HW-SARIMA*, resultados destacáveis não são obtidos em comparação com a combinação das outras metodologias; só alguns resultados são melhores quando comparados com as previsões obtidas através de cada método individualmente.

Tabela 4.53: Combinação de previsões *HW-SARIMA-ANN*$_{st}$.

	MAPE(%)		*MAD*	
Série	prev-o	prev-m	prev-o	prev-m
ICMS	2.8790	3.5081	106.9346	399.7609
ICMSPE	2.1194	5.4827	5.1981	2.1534
ICMSRJ	3.5770	5.8405	35.2078	14.9543
ICMSSP	1.2964	1.1329	52.2552	44.3115

Tabela 4.54: Combinação de previsões *HW-ANN*$_{st}$.

Série	*MAPE(%)*		*MAD*	
	prev-o	prev-m	prev-o	prev-m
ICMS	3.1805	4.1367	225.8533	332.5872
ICMSPE	2.2031	5.2522	4.7437	2.0328
ICMSRJ	3.4847	5.9275	32.5197	11.0077
ICMSSP	1.5312	1.6510	49.2839	52.0715

Tabela 4.55: Combinação de previsões *ANN*$_{st}$*-SARIMA*.

Série	*MAPE(%)*		*MAD*	
	prev-o	prev-m	prev-o	prev-m
ICMS	2.6059	3.7801	94.9516	363.4399
ICMSPE	2.5421	5.6936	4.6612	3.6189
ICMSRJ	4.1524	5.7736	34.7307	19.7810
ICMSSP	1.4083	1.2446	47.3433	39.4720

Tabela 4.56: Combinação de previsões *HW-SARIMA*.

Série	*MAPE(%)*		*MAD*	
	prev-o	prev-m	prev-o	prev-m
ICMS	3.4858	3.5771	123.1461	413.2327
ICMSPE	2.3309	5.9371	5.8020	2.6298
ICMSRJ	3.7421	6.0277	38.9619	13.2202
ICMSSP	2.0344	1.4416	79.7310	59.8182

CAPÍTULO 5

CONCLUSÃO

Três metodologias foram avaliadas com vistas à previsão da arrecadação do Imposto sobre Circulação de Mercadorias e sobre Prestações de Serviços de Transporte Interestadual e Intermunicipal e de Comunicação (*ICMS*) do Brasil (agregado) e dos seguintes estados: Rio de Janeiro, Pernambuco e São Paulo. As metodologias utilizadas nesta dissertação foram o algoritmo de alisamento exponencial de Holt-Winters, os modelos *SARIMA* e as as redes neurais. Combinações de previsões individuais também foram consideradas. A identificação e a correção por outliers é tipicamente de considerável importância, antes da modelagem de um conjunto de dados, como tem sido descrito na literatura estatística. As séries de arrecadação são submetidas a este processo e as previsões produzidas foram analisadas. Os principais resultados obtidos ao longo de duas análises encontram-se resumidos a seguir.

1. Na análise I, foi possível encontrar redes neurais com maior precisão na previsão um passo à frente do que nas outras duas metodologias, como no caso da previsão de dezembro de 2004. Além disso, o ajuste do conjunto de treinamento

relativo à série de arrecadação verdadeira mostrou-se melhor que o fornecido pelos outros métodos. No entanto, quando as previsões de dezembro de 2004 a março de 2005 foram analisadas conjuntamente, o algoritmo de alisamento exponencial de Holt-Winters apresentou os melhores resultados na previsão um e múltiplos passos à frente segundo o *MAPE* e o *MAD*, com exceção do *ICMSSP*. Combinando as previsões do tipo *HW-SARIMA-ANN* foi possível melhorar as previsões individuais, especialmente no caso do *ICMSRJ*, segundo o *MAPE*. Através da combinação do *HW-SARIMA*, a previsão um passo à frente do *ICMSRJ* proporcionou, em geral, os menores *MAPE* e *MAD*. No caso do *ICMS*, *ICMSPE* e *ICMSSP*, a combinação das redes neurais com qualquer dos outros métodos se mostrou eficaz. Considerando os melhores resultados de previsão, através das metodologias individuais e da combinação de previsões, as previsões obtidas para a série *ICMS* apresentaram *MAPE* menor que 2% na previsão um e múltiplos passos à frente; as previsões obtidas para a série *ICMSPE* apresentaram *MAPE* menor que 3% na previsão um passo à frente e 4% na previsão múltiplos passos à frente; as previsões para a série *ICMSRJ* exibiram *MAPE* menor que 3% na previsão um e múltiplos passos à frente; as previsões obtidas para a série *ICMSSP* exibiram *MAPE* menor que 1% na previsão um passo à frente e 2% na previsão múltiplos passos à frente.

2. Na análise II, os resultados individuais de previsão por alisamento exponencial, por *SARIMA* e por redes neurais com o neurônio de tendência revelaram o melhor desempenho das redes neurais nas previsões múltiplos passos à frente, segundo o *MAPE*. Igualmente, pertenceram às redes neurais a maior parte de valores mínimos do *MAD*, na previsão um e múltiplos passos à frente. Neste caso, o algoritmo de alisamento exponencial deixou de ter desempenho tão destacado como na análise anterior. No que tange a combinações de previsões, a dada por *HW-SARIMA-ANN*, em geral, apresentou os melhores resultados na previsão um e múltiplos passos à frente, segundo o *MAPE*, para o *ICMSRJ*

e para o *ICMSSP*. Considerando os melhores resultados de prognóstico, através das metodologias individuais e de combinação de previsões, as previsões obtidas para o *ICMS* exibiram *MAPE* menor que 3% na previsão um passo à frente e 4% na previsão múltiplos passos à frente; as previsões obtidas para o *ICMSPE* exibiram *MAPE* menor que 3% na previsão um passo à frente e 6% na previsão múltiplos passos à frente; as previsões para o *ICMSRJ* exibiram *MAPE* menor que 4% na previsão um passo à frente e 6% na previsão múltiplos passos à frente; as previsões obtidas para o *ICMSSP* exibiram *MAPE* menor que 2% na previsão um passo e múltiplos passos à frente.

3. Na análise II, os resultados individuais de previsão por alisamento exponencial, por *SARIMA* e por redes neurais sem o neurônio de tendência revelaram que redes neurais tipicamente fornecem as melhores previsões múltiplos passos à frente, através do *MAPE*. No que tange à combinação de previsões, a combinação *HW-SARIMA-ANN* em geral apresentou os melhores resultados na previsão um passo à frente do *ICMSPE* e *ICMSSP* e na previsão múltiplos passos à frente do *ICMS* e *ICMSSP*. Com relação ao *MAD*, as combinações *HW-ANN* e *SARIMA-ANN* exibiram os menores desvios, em geral, na previsão um e múltiplos passos à frente. Adicionalmente, as previsões um passo à frente obtidas combinando-se métodos tradicionais com redes neurais forneceram os menores *MAPE*. Considerando os melhores resultados de prognóstico, através das metodologias individuais e da combinação de previsões, as previsões para o *ICMS* exibiram *MAPE* menor que 3% na previsão um passo à frente e 4% na previsão múltiplos passos à frente; as previsões para o *ICMSPE* exibiram *MAPE* menor que 3% na previsão um passo à frente e 6% na previsão múltiplos passos à frente; as previsões para o *ICMSRJ* exibiram *MAPE* menor que 4% na previsão um passo à frente e 6% na previsão múltiplos passos à frente, enquanto que as previsões do *ICMSSP* apresentaram *MAPE* menor que 2% na previsão um e múltiplos passos à frente.

As redes neurais têm um campo de aplicabilidade muito amplo. Os resultados contidos na presente dissertação ilustram a utilidade desse conjunto de métodos no que tange a previsão de séries econômicas. Os resultados sugerem que metodologias baseadas em redes neurais devem ser consideradas por aqueles que produzem rotineiramente previsões de receitas tributárias. Futuros trabalhos podem ser direcionados à utilização de redes recorrentes, redes híbridas e com poda de conexões entre neurônios, como também às novas propostas recentemente feitas na literatura, veja Ghiassi & Saidane (2005) e Ghiassi, Saidane & Zimbra (2005). Igualmente, a análise multivariada de séries temporais é uma janela aberta para estudos futuros.

APÊNDICE A

Programa para estimação e previsão com redes neurais

O pacote `AMORE` (`A MORE` flexible Neural Network) do `R` é utilizado para obter as estimativas dos parâmetros das redes neurais e as previsões um e múltiplos passos à frente para cada uma das séries. A seguir será apresentado o código fonte que descreve este processo para a série do *ICMS*, na análise II; as previsões das demais séries são obtidas de forma semelhante.

```
############################################################
###                                                      ###
### Objetivo :  Previsão da série ICMS                   ###
###             através de redes neurais                 ###
###             com arquitetura "feed-forward",          ###
###             algoritmo de ensino "Backpropagation"    ###
###                                                      ###
### Autor    :  Juan Camilo Santana                      ###
###                                                      ###
############################################################
```

```
#####################################
###### Importação da informação ######
#####################################

rm(list=ls(all=TRUE))

base<-read.table("C:\\DATA1.txt",header=T)
set.seed(161534)

h<-6     ## previsões

restrict<-function(z){
  zmin<-min(z)
  zmax<-max(z)
  z1<- (z - zmin) / (zmax - zmin)
  zr<- (2*z1)-1
  return(zr)
}

#############################
#### Transformação Série ####
#############################

icms.ts = ts(base$icms, start=c(1994,7), end=c(2004,12),
             frequency=12)
icms.r.log <- icms.ts

###################################
#######    Rede Neural   ##########
###################################

 icms.ten <-ts(base$icmstrend, start=c(1994,7),end=c(2004,12),
               frequency=12)

 r<-length(icms.r.log)

 jan<-matrix(0,r,1)
 fev<-matrix(0,r,1)
 mar<-matrix(0,r,1)
 abr<-matrix(0,r,1)
 mai<-matrix(0,r,1)
 jun<-matrix(0,r,1)
```

```
jul<-matrix(0,r,1)
ago<-matrix(0,r,1)
set<-matrix(0,r,1)
out<-matrix(0,r,1)
nov<-matrix(0,r,1)
dez<-matrix(0,r,1)

i<-1
while(i <=121){ jul[i]<- 1.0; i<-i+12}
i<-2
while(i <=122){ ago[i]<- 1.0; i<-i+12}
i<-3
while(i <=123){ set[i]<- 1.0; i<-i+12}
i<-4
while(i <=124){ out[i]<- 1.0; i<-i+12}
i<-5
while(i <=125){ nov[i]<- 1.0; i<-i+12}
i<-6
while(i <=126){ dez[i]<- 1.0; i<-i+12}
i<-7
while(i <=115){ jan[i]<- 1.0; i<-i+12}
i<-8
while(i <=116){ fev[i]<- 1.0; i<-i+12}
i<-9
while(i <=117){ mar[i]<- 1.0; i<-i+12}
i<-10
while(i <=118){ abr[i]<- 1.0; i<-i+12}
i<-11
while(i <=119){ mai[i]<- 1.0; i<-i+12}
i<-12
while(i <=120){ jun[i]<- 1.0; i<-i+12}

x1<-matrix(0,r,1)
x1[1]<-0
for(i in 2:r) {
  x1[i] =  icms.r.log[i-1]
}
x2<-matrix(0,r,1)
x2[1:2]<-0
for(i in 3:r) {
  x2[i] =  icms.r.log[i-2]
}
```

```
x3<-matrix(0,r,1)
x3[1:3]<-0
for(i in 4:r) {
  x3[i] =  icms.r.log[i-3]
}
x4<-matrix(0,r,1)
x4[1:4]<-0
for(i in 5:r) {
  x4[i] =  icms.r.log[i-4]
}
x5<-matrix(0,r,1)
x5[1:5]<-0
for(i in 6:r) {
  x5[i] =  icms.r.log[i-5]
}
x6<-matrix(0,r,1)
x6[1:6]<-0
for(i in 7:r) {
  x6[i] =  icms.r.log[i-6]
}
x7<-matrix(0,r,1)
x7[1:7]<-0
for(i in 8:r) {
  x7[i] =  icms.r.log[i-7]
}
x8<-matrix(0,r,1)
x8[1:8]<-0
for(i in 9:r) {
  x8[i] =  icms.r.log[i-8]
}
x9<-matrix(0,r,1)
x9[1:9]<-0
for(i in 10:r) {
  x9[i] =  icms.r.log[i-9]
}
x10<-matrix(0,r,1)
x10[1:10]<-0
for(i in 11:r) {
  x10[i] =  icms.r.log[i-10]
}
```

```
x11<-matrix(0,r,1)
x11[1:11]<-0
for(i in 12:r) {
  x11[i] =  icms.r.log[i-11]
}
x12<-matrix(0,r,1)
x12[1:12]<-0
for(i in 13:r) {
  x12[i] =  icms.r.log[i-12]
}

x_1<-ts(x1, start=c(1994,7), end=c(2004,12),frequency=12)
x_2<-ts(x2, start=c(1994,7), end=c(2004,12),frequency=12)
x_3<-ts(x3, start=c(1994,7), end=c(2004,12),frequency=12)
x_4<-ts(x4, start=c(1994,7), end=c(2004,12),frequency=12)
x_5<-ts(x5, start=c(1994,7), end=c(2004,12),frequency=12)
x_6<-ts(x6, start=c(1994,7), end=c(2004,12),frequency=12)
x_7<-ts(x7, start=c(1994,7), end=c(2004,12),frequency=12)
x_8<-ts(x8, start=c(1994,7), end=c(2004,12),frequency=12)
x_9<-ts(x9, start=c(1994,7), end=c(2004,12),frequency=12)
x_10<-ts(x10, start=c(1994,7), end=c(2004,12),frequency=12)
x_11<-ts(x11, start=c(1994,7), end=c(2004,12),frequency=12)
x_12<-ts(x12, start=c(1994,7), end=c(2004,12),frequency=12)

#########################################################################
## bgeral contém a seguir a informação completa sobre as variáveis ##
#########################################################################

bgeral<-cbind(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9,x_10,
              x_11,x_12, icms.ten, jan, fev, mar, abr, mai,
              jun, jul, ago, set, out, nov, dez, icms.r.log)

#####################################
#### Incluindo variáveis mensais ####
#####################################

library(AMORE)

t.end<-length(icms.ts)-h
rang1<-1:t.end
```

```
x_1 <-bgeral[,"x_1"]
x_2 <-bgeral[,"x_2"]
x_3 <-bgeral[,"x_3"]
x_4 <-bgeral[,"x_4"]
x_5 <-bgeral[,"x_5"]
x_6 <-bgeral[,"x_6"]
x_7 <-bgeral[,"x_7"]
x_8 <-bgeral[,"x_8"]
x_9 <-bgeral[,"x_9"]
x_10 <-bgeral[,"x_10"]
x_11 <-bgeral[,"x_11"]
x_12 <-bgeral[,"x_12"]


jan<- bgeral[,"jan"]
fev<- bgeral[,"fev"]
mar<- bgeral[,"mar"]
abr<- bgeral[,"abr"]
mai<- bgeral[,"mai"]
jun<- bgeral[,"jun"]
jul<- bgeral[,"jul"]
ago<- bgeral[,"ago"]
set<- bgeral[,"set"]
out<- bgeral[,"out"]
nov<- bgeral[,"nov"]
dez<- bgeral[,"dez"]

icms.ten1<-icms.ten[rang1]
icms.ten1<-restrict(icms.ten1)

x_1<-restrict(x_1[2:t.end ])
a1<-x_1
x_1<-matrix(0,t.end)
x_1[1]<-0
for(i in 2:t.end){ x_1[i]<-a1[i-1] }

x_2<-restrict(x_2[3:t.end ])
a2<-x_2
x_2<-matrix(0,t.end)
x_2[1:2]<-0
for(i in 3:t.end){ x_2[i]<-a2[i-2] }
```

```
x_3<-restrict(x_3[4:t.end ])
a3<-x_3
x_3<-matrix(0,t.end)
x_3[1:3]<-0
for(i in 4:t.end){ x_3[i]<-a3[i-3] }

x_4<-restrict(x_4[5:t.end ])
a4<-x_4
x_4<-matrix(0,t.end)
x_4[1:4]<-0
for(i in 5:t.end){ x_4[i]<-a4[i-4] }

x_5<-restrict(x_5[6:t.end ])
a5<-x_5
x_5<-matrix(0,t.end)
x_5[1:5]<-0
for(i in 6:t.end){ x_5[i]<-a5[i-5] }

x_6<-restrict(x_6[7:t.end ])
a6<-x_6
x_6<-matrix(0,t.end)
x_6[1:6]<-0
for(i in 7:t.end){ x_6[i]<-a6[i-6] }

x_7<-restrict(x_7[8:t.end ])
a7<-x_7
x_7<-matrix(0,t.end)
x_7[1:7]<-0
for(i in 8:t.end){ x_7[i]<-a7[i-7] }

x_8<-restrict(x_8[9:t.end ])
a8<-x_8
x_8<-matrix(0,t.end)
x_8[1:8]<-0
for(i in 9:t.end){ x_8[i]<-a8[i-8] }

x_9<-restrict(x_9[10:t.end ])
a9<-x_9
x_9<-matrix(0,t.end)
x_9[1:9]<-0
for(i in 10:t.end){ x_9[i]<-a9[i-9] }
```

```
x_10<-restrict(x_10[11:t.end ])
a10<-x_10
x_10<-matrix(0,t.end)
x_10[1:10]<-0
for(i in 11:t.end){ x_10[i]<-a10[i-10] }

x_11<-restrict(x_11[12:t.end ])
a11<-x_11
x_11<-matrix(0,t.end)
x_11[1:11]<-0
for(i in 12:t.end){ x_11[i]<-a11[i-11] }

x_12<-restrict(x_12[13:t.end ])
a12<-x_12
x_12<-matrix(0,t.end)
x_12[1:12]<-0
for(i in 13:t.end){ x_12[i]<-a12[i-12] }

x_1_1<-x_1
x_2_1<-x_2
x_3_1<-x_3
x_4_1<-x_4
x_5_1<-x_5
x_6_1<-x_6
x_7_1<-x_7
x_8_1<-x_8
x_9_1<-x_9
x_10_1<-x_10
x_11_1<-x_11
x_12_1<-x_12

x_1<-x_1_1[rang1]
x_2<-x_2_1[rang1]
x_3<-x_3_1[rang1]
x_4<-x_4_1[rang1]
x_5<-x_5_1[rang1]
x_6<-x_6_1[rang1]
x_7<-x_7_1[rang1]
x_8<-x_8_1[rang1]
x_9<-x_9_1[rang1]
x_10<-x_10_1[rang1]
x_11<-x_11_1[rang1]
```

```
x_12<-x_12_1[rang1]


icms.ten<-icms.ten1
icms.r.log<- bgeral[rang1,"icms.r.log"]


jan<-bgeral[rang1,"jan"]
fev<-bgeral[rang1,"fev"]
mar<-bgeral[rang1,"mar"]
abr<-bgeral[rang1,"abr"]
mai<-bgeral[rang1,"mai"]
jun<-bgeral[rang1,"jun"]
jul<-bgeral[rang1,"jul"]
ago<-bgeral[rang1,"ago"]
set<-bgeral[rang1,"set"]
out<-bgeral[rang1,"out"]
nov<-bgeral[rang1,"nov"]
dez<-bgeral[rang1,"dez"]


r1<-length(icms.r.log)

#####################################################
## Processo de estimação dos pesos da rede neural ##
#####################################################

var<-12+3

x<-matrix(cbind(x_1,x_2,icms.ten,jan,fev,mar,abr,mai,jun,
          jul,ago,set,out,nov,dez), r1, var)

y<-matrix(cbind( icms.r.log),r1,1)

 crt<-c(0.01,0.1)          # taxa de aprendizado
 cm<-c(0.1,0.5)            # taxa momentum (momento)
 epoc<-10000               # número de epochs
 neu<-1:6                  # número neurônios
```

```
#####################################################
## Arquitetura de redes neurais. Os pesos são     ##
## inicializados ao acaso                          ##
#####################################################

 y.net.1.1  <- newff(n.neurons= c(var,neu[1],1),
               learning.rate.global=crt[1],
               momentum.global=cm[1] , error.criterium="LMS",
               hidden.layer="tansig",output.layer="purelin",
               method="ADAPTgdwm")
 y.net.2.1  <- newff(n.neurons= c(var,neu[1],1),
               learning.rate.global=crt[1],
               momentum.global=cm[2] , error.criterium="LMS",
               hidden.layer="tansig",output.layer="purelin",
               method="ADAPTgdwm")
 y.net.3.1  <- newff(n.neurons= c(var,neu[1],1) ,
               learning.rate.global=crt[2],
               momentum.global=cm[1] , error.criterium="LMS",
               hidden.layer="tansig",output.layer="purelin",
               method="ADAPTgdwm")
 y.net.4.1  <- newff(n.neurons= c(var,neu[1],1) ,
               learning.rate.global=crt[2],
               momentum.global=cm[2] , error.criterium="LMS",
               hidden.layer="tansig",output.layer="purelin",
               method="ADAPTgdwm")
 y.net.5.1  <- newff(n.neurons= c(var,neu[2],1) ,
               learning.rate.global=crt[1],
               momentum.global=cm[1] , error.criterium="LMS",
               hidden.layer="tansig",output.layer="purelin",
               method="ADAPTgdwm")
 y.net.6.1  <- newff(n.neurons= c(var,neu[2],1) ,
               learning.rate.global=crt[1],
               momentum.global=cm[2] , error.criterium="LMS",
               hidden.layer="tansig",output.layer="purelin",
               method="ADAPTgdwm")
 y.net.7.1  <- newff(n.neurons= c(var,neu[2],1) ,
               learning.rate.global=crt[2],
               momentum.global=cm[1] , error.criterium="LMS",
               hidden.layer="tansig",output.layer="purelin",
               method="ADAPTgdwm")
 y.net.8.1  <- newff(n.neurons= c(var,neu[2],1) ,
               learning.rate.global=crt[2],
```

```
              momentum.global=cm[2] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.9.1  <- newff(n.neurons= c(var,neu[3],1) ,
              learning.rate.global=crt[1],
              momentum.global=cm[1] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.10.1 <- newff(n.neurons= c(var,neu[3],1) ,
              learning.rate.global=crt[1],
              momentum.global=cm[2] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.11.1 <- newff(n.neurons= c(var,neu[3],1) ,
              learning.rate.global=crt[2],
              momentum.global=cm[1] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.12.1 <- newff(n.neurons= c(var,neu[3],1) ,
              learning.rate.global=crt[2],
              momentum.global=cm[2] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.13.1 <- newff(n.neurons= c(var,neu[4],1) ,
              learning.rate.global=crt[1],
              momentum.global=cm[1] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.14.1 <- newff(n.neurons= c(var,neu[4],1) ,
              learning.rate.global=crt[1],
              momentum.global=cm[2] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.15.1 <- newff(n.neurons= c(var,neu[4],1) ,
              learning.rate.global=crt[2],
              momentum.global=cm[1] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.16.1 <- newff(n.neurons= c(var,neu[5],1) ,
              learning.rate.global=crt[2],
              momentum.global=cm[2] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
```

```
              method="ADAPTgdwm")
y.net.17.1 <- newff(n.neurons= c(var,neu[5],1) ,
              learning.rate.global=crt[1],
              momentum.global=cm[1] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.18.1 <- newff(n.neurons= c(var,neu[5],1) ,
              learning.rate.global=crt[1],
              momentum.global=cm[2] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.19.1 <- newff(n.neurons= c(var,neu[5],1) ,
              learning.rate.global=crt[2],
              momentum.global=cm[1] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.20.1 <- newff(n.neurons= c(var,neu[5],1) ,
              learning.rate.global=crt[2],
              momentum.global=cm[2] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.21.1 <- newff(n.neurons= c(var,neu[6],1) ,
              learning.rate.global=crt[1],
              momentum.global=cm[1] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.22.1 <- newff(n.neurons= c(var,neu[6],1) ,
              learning.rate.global=crt[1],
              momentum.global=cm[2] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.23.1 <- newff(n.neurons= c(var,neu[6],1) ,
              learning.rate.global=crt[2],
              momentum.global=cm[1] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.24.1 <- newff(n.neurons= c(var,neu[6],1) ,
              learning.rate.global=crt[2],
              momentum.global=cm[2] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
```

```
## net.2

  y.net.1.2 <- newff(n.neurons= c(var,neu[1],neu[1],1) ,
               learning.rate.global=crt[1],
               momentum.global=cm[1] , error.criterium="LMS",
               hidden.layer="tansig",output.layer="purelin",
               method="ADAPTgdwm")
  y.net.2.2 <- newff(n.neurons= c(var,neu[1],neu[1],1) ,
               learning.rate.global=crt[1],
               momentum.global=cm[2] , error.criterium="LMS",
               hidden.layer="tansig",output.layer="purelin",
               method="ADAPTgdwm")
  y.net.3.2 <- newff(n.neurons= c(var,neu[1],neu[1],1) ,
               learning.rate.global=crt[2],
               momentum.global=cm[1] , error.criterium="LMS",
               hidden.layer="tansig",output.layer="purelin",
               method="ADAPTgdwm")
  y.net.4.2 <- newff(n.neurons= c(var,neu[1],neu[1],1) ,
               learning.rate.global=crt[2],
               momentum.global=cm[2] , error.criterium="LMS",
               hidden.layer="tansig",output.layer="purelin",
               method="ADAPTgdwm")
  y.net.5.2 <- newff(n.neurons= c(var,neu[2],neu[2],1) ,
               learning.rate.global=crt[1],
               momentum.global=cm[1] , error.criterium="LMS",
               hidden.layer="tansig",output.layer="purelin",
               method="ADAPTgdwm")
  y.net.6.2 <- newff(n.neurons= c(var,neu[2],neu[2],1) ,
               learning.rate.global=crt[1],
               momentum.global=cm[2] , error.criterium="LMS",
               hidden.layer="tansig",output.layer="purelin",
               method="ADAPTgdwm")
  y.net.7.2 <- newff(n.neurons= c(var,neu[2],neu[2],1),
               learning.rate.global=crt[2],
               momentum.global=cm[1] , error.criterium="LMS",
               hidden.layer="tansig",output.layer="purelin",
               method="ADAPTgdwm")
  y.net.8.2 <- newff(n.neurons= c(var,neu[2],neu[2],1) ,
               learning.rate.global=crt[2],
               momentum.global=cm[2] , error.criterium="LMS",
               hidden.layer="tansig",output.layer="purelin",
               method="ADAPTgdwm")
```

```
y.net.9.2 <- newff(n.neurons= c(var,neu[3],neu[3],1) ,
              learning.rate.global=crt[1],
              momentum.global=cm[1] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.10.2 <- newff(n.neurons= c(var,neu[3],neu[3],1) ,
              learning.rate.global=crt[1],
              momentum.global=cm[2] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.11.2 <- newff(n.neurons= c(var,neu[3],neu[3],1) ,
              learning.rate.global=crt[2],
              momentum.global=cm[1] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.12.2 <- newff(n.neurons= c(var,neu[3],neu[3],1) ,
              learning.rate.global=crt[2],
              momentum.global=cm[2] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.13.2 <- newff(n.neurons= c(var,neu[4],neu[4],1) ,
              learning.rate.global=crt[1],
              momentum.global=cm[1] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.14.2 <- newff(n.neurons= c(var,neu[4],neu[4],1) ,
              learning.rate.global=crt[1],
              momentum.global=cm[2] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.15.2 <- newff(n.neurons= c(var,neu[4],neu[4],1) ,
              learning.rate.global=crt[2],
              momentum.global=cm[1] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.16.2 <- newff(n.neurons= c(var,neu[4],neu[4],1) ,
              learning.rate.global=crt[2],
              momentum.global=cm[2] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.17.2 <- newff(n.neurons= c(var,neu[5],neu[5],1) ,
              learning.rate.global=crt[1],
```

```
                momentum.global=cm[1] , error.criterium="LMS",
                hidden.layer="tansig",output.layer="purelin",
                method="ADAPTgdwm")
  y.net.18.2 <- newff(n.neurons= c(var,neu[5],neu[5],1) ,
                learning.rate.global=crt[1],
                momentum.global=cm[2] , error.criterium="LMS",
                hidden.layer="tansig",output.layer="purelin",
                method="ADAPTgdwm")
  y.net.19.2 <- newff(n.neurons= c(var,neu[5],neu[5],1) ,
                learning.rate.global=crt[2],
                momentum.global=cm[1] , error.criterium="LMS",
                hidden.layer="tansig",output.layer="purelin",
                method="ADAPTgdwm")
  y.net.20.2 <- newff(n.neurons= c(var,neu[5],neu[5],1) ,
                learning.rate.global=crt[2],
                momentum.global=cm[2] , error.criterium="LMS",
                hidden.layer="tansig",output.layer="purelin",
                method="ADAPTgdwm")
  y.net.21.2 <- newff(n.neurons= c(var,neu[6],neu[6],1) ,
                learning.rate.global=crt[1],
                momentum.global=cm[1] , error.criterium="LMS",
                hidden.layer="tansig",output.layer="purelin",
                method="ADAPTgdwm")
  y.net.22.2 <- newff(n.neurons= c(var,neu[6],neu[6],1) ,
                learning.rate.global=crt[1],
                momentum.global=cm[2] , error.criterium="LMS",
                hidden.layer="tansig",output.layer="purelin",
                method="ADAPTgdwm")
  y.net.23.2 <- newff(n.neurons= c(var,neu[6],neu[6],1),
                learning.rate.global=crt[2],
                momentum.global=cm[1] , error.criterium="LMS",
                hidden.layer="tansig",output.layer="purelin",
                method="ADAPTgdwm")
  y.net.24.2 <- newff(n.neurons=c(var,neu[6],neu[6],1) ,
                learning.rate.global=crt[2],
                momentum.global=cm[2] , error.criterium="LMS",
                hidden.layer="tansig",output.layer="purelin",
                method="ADAPTgdwm")
## net 3

  y.net.1.3  <- newff(n.neurons= c(var,neu[1],neu[1]+1,1) ,
                learning.rate.global=crt[1],
```

```
              momentum.global=cm[1] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.2.3  <- newff(n.neurons= c(var,neu[1],neu[1]+1,1) ,
              learning.rate.global=crt[1],
              momentum.global=cm[2] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.3.3  <- newff(n.neurons= c(var,neu[1],neu[1]+1,1) ,
              learning.rate.global=crt[2],
              momentum.global=cm[1] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.4.3  <- newff(n.neurons= c(var,neu[1],neu[1]+1,1) ,
              learning.rate.global=crt[2],
              momentum.global=cm[2] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.5.3  <- newff(n.neurons= c(var,neu[2],neu[2]+1,1) ,
              learning.rate.global=crt[1],
              momentum.global=cm[1] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.6.3  <- newff(n.neurons= c(var,neu[2],neu[2]+1,1),
              learning.rate.global=crt[1],
              momentum.global=cm[2] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.7.3  <- newff(n.neurons= c(var,neu[2],neu[2]+1,1),
              learning.rate.global=crt[2],
              momentum.global=cm[1] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.8.3  <- newff(n.neurons= c(var,neu[2],neu[2]+1,1) ,
              learning.rate.global=crt[2],
              momentum.global=cm[2] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.9.3  <- newff(n.neurons= c(var,neu[3],neu[3]+1,1) ,
              learning.rate.global=crt[1],
              momentum.global=cm[1] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
```

```
            method="ADAPTgdwm")
y.net.10.3 <- newff(n.neurons= c(var,neu[3],neu[3]+1,1) ,
            learning.rate.global=crt[1],
            momentum.global=cm[2] , error.criterium="LMS",
            hidden.layer="tansig",output.layer="purelin",
            method="ADAPTgdwm")
y.net.11.3 <- newff(n.neurons=c(var,neu[3],neu[3]+1,1) ,
            learning.rate.global=crt[2],
            momentum.global=cm[1] , error.criterium="LMS",
            hidden.layer="tansig",output.layer="purelin",
            method="ADAPTgdwm")
y.net.12.3 <- newff(n.neurons=c(var,neu[3],neu[3]+1,1) ,
            learning.rate.global=crt[2],
            momentum.global=cm[2] , error.criterium="LMS",
            hidden.layer="tansig",output.layer="purelin",
            method="ADAPTgdwm")
y.net.13.3 <- newff(n.neurons= c(var,neu[4],neu[4]+1,1),
            learning.rate.global=crt[1],
            momentum.global=cm[1] , error.criterium="LMS",
            hidden.layer="tansig",output.layer="purelin",
            method="ADAPTgdwm")
y.net.14.3 <- newff(n.neurons= c(var,neu[4],neu[4]+1,1) ,
            learning.rate.global=crt[1],
            momentum.global=cm[2] , error.criterium="LMS",
            hidden.layer="tansig",output.layer="purelin",
            method="ADAPTgdwm")
y.net.15.3 <- newff(n.neurons= c(var,neu[4],neu[4]+1,1) ,
            learning.rate.global=crt[2],
            momentum.global=cm[1] , error.criterium="LMS",
            hidden.layer="tansig",output.layer="purelin",
            method="ADAPTgdwm")
y.net.16.3 <- newff(n.neurons= c(var,neu[4],neu[4]+1,1) ,
            learning.rate.global=crt[2],
            momentum.global=cm[2] , error.criterium="LMS",
            hidden.layer="tansig",output.layer="purelin",
            method="ADAPTgdwm")
y.net.17.3 <- newff(n.neurons= c(var,neu[5],neu[5]+1,1) ,
            learning.rate.global=crt[1],
            momentum.global=cm[1] , error.criterium="LMS",
            hidden.layer="tansig",output.layer="purelin",
            method="ADAPTgdwm")
y.net.18.3 <- newff(n.neurons= c(var,neu[5],neu[5]+1,1),
```

```
              learning.rate.global=crt[1],
              momentum.global=cm[2] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.19.3 <- newff(n.neurons= c(var,neu[5],neu[5]+1,1) ,
              learning.rate.global=crt[2],
              momentum.global=cm[1] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.20.3 <- newff(n.neurons=c(var,neu[5],neu[5]+1,1) ,
              learning.rate.global=crt[2],
              momentum.global=cm[2] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.21.3 <- newff(n.neurons= c(var,neu[6],neu[6]+1,1) ,
              learning.rate.global=crt[1],
              momentum.global=cm[1] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.22.3 <- newff(n.neurons= c(var,neu[6],neu[6]+1,1) ,
              learning.rate.global=crt[1],
              momentum.global=cm[2] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.23.3 <- newff(n.neurons= c(var,neu[6],neu[6]+1,1),
              learning.rate.global=crt[2],
              momentum.global=cm[1] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")
y.net.24.3 <- newff(n.neurons=c(var,neu[6],neu[6]+1,1) ,
              learning.rate.global=crt[2],
              momentum.global=cm[2] , error.criterium="LMS",
              hidden.layer="tansig",output.layer="purelin",
              method="ADAPTgdwm")

net.1.1   <- train(y.net.13.3, x, y,
              n.epochs=epoc,
              error.criterium="LMS",
              Stao=NA, report=F, show.step=1000)
```

```
## predição da rede neural sobre o conjunto do treinamento

  net.sim.1.1  <-  sim.MLPnet(net.1.1, x)

## Medidas de erro no ajuste (training)

  e.1.1     <- exp(y) - exp(net.sim.1.1)
  tMSE.1.1  <- mean(e.1.1^2);
  tMAE.1.1  <- mean(abs(e.1.1));
  tMAPE.1.1 <- mean( abs(e.1.1/exp(y)))*100;
  tME.1.1   <- mean(e.1.1);
  tMAD.1.1  <- median( abs( e.1.1 - median(e.1.1)) )

#########################################################
### Previsão da série com o conjunto de validação ###
#########################################################

 rang2<- (t.end+1):(t.end+h)

 x_1<-(bgeral[rang2,"x_1"]-min(bgeral[(2:t.end),"x_1"]))/
      (max(bgeral[(2:t.end),"x_1"])-min(bgeral[(2:t.end),"x_1"]))
 x_1<-(2*x_1) - 1

 x_2<-(bgeral[rang2,"x_2"]-min(bgeral[(3:t.end),"x_2"]))/
      (max(bgeral[(3:t.end),"x_2"])-min(bgeral[(3:t.end),"x_2"]))
 x_2<-(2*x_2) - 1

 x_3<-(bgeral[rang2,"x_3"]-min(bgeral[(4:t.end),"x_3"]))/
       (max(bgeral[(4:t.end),"x_3"])-min(bgeral[(4:t.end),"x_3"]))
 x_3<-(2*x_3) - 1

 x_4<-(bgeral[rang2,"x_4"]-min(bgeral[(5:t.end),"x_4"]))/
       (max(bgeral[(5:t.end),"x_4"])-min(bgeral[(5:t.end),"x_4"]))
 x_4<-(2*x_4) - 1

 x_5<-(bgeral[rang2,"x_5"]-min(bgeral[(6:t.end),"x_5"]))/
      (max(bgeral[(6:t.end),"x_5"])-min(bgeral[(6:t.end),"x_5"]))
 x_5<-(2*x_5) - 1

 x_6<-(bgeral[rang2,"x_6"]-min(bgeral[(7:t.end),"x_6"]))/
      (max(bgeral[(7:t.end),"x_6"])-min(bgeral[(7:t.end),"x_6"]))
 x_6<-(2*x_6) - 1
```

```
x_7<-(bgeral[rang2,"x_7"]-min(bgeral[(8:t.end),"x_7"]))/
     (max(bgeral[(8:t.end),"x_7"])-min(bgeral[(8:t.end),"x_7"]))
x_7<-(2*x_7) - 1

x_8<-(bgeral[rang2,"x_8"]-min(bgeral[(9:t.end),"x_8"]))/
     (max(bgeral[(9:t.end),"x_8"])-min(bgeral[(9:t.end),"x_8"]))
x_8<-(2*x_8) - 1

x_9<-(bgeral[rang2,"x_9"]-min(bgeral[(10:t.end),"x_9"]))/
      (max(bgeral[(10:t.end),"x_9"])-min(bgeral[(10:t.end),"x_9"]))
x_9<-(2*x_9) - 1

x_10<-(bgeral[rang2,"x_10"]-min(bgeral[(11:t.end),"x_10"]))/
     (max(bgeral[(11:t.end),"x_10"])-min(bgeral[(11:t.end),"x_10"]))
x_10<-(2*x_10) - 1

x_11<-(bgeral[rang2,"x_11"]-min(bgeral[(12:t.end),"x_11"]))/
     (max(bgeral[(12:t.end),"x_11"])-min(bgeral[(12:t.end),"x_11"]))
x_11<-(2*x_11) - 1

x_12<-(bgeral[rang2,"x_12"]-min(bgeral[(13:t.end),"x_12"]))/
     (max(bgeral[(13:t.end),"x_12"])-min(bgeral[(13:t.end),"x_12"]))
x_12<-(2*x_12) - 1


jan<-bgeral[rang2,"jan"]
fev<-bgeral[rang2,"fev"]
mar<-bgeral[rang2,"mar"]
abr<-bgeral[rang2,"abr"]
mai<-bgeral[rang2,"mai"]
jun<-bgeral[rang2,"jun"]
jul<-bgeral[rang2,"jul"]
ago<-bgeral[rang2,"ago"]
set<-bgeral[rang2,"set"]
out<-bgeral[rang2,"out"]
nov<-bgeral[rang2,"nov"]
dez<-bgeral[rang2,"dez"]


icms.ten<- (bgeral[rang2,"icms.ten"]-min(bgeral[rang1,"icms.ten"]))/
(max(bgeral[rang1,"icms.ten"])-min(bgeral[rang1,"icms.ten"]))
```

```
 icms.ten<-(2*icms.ten)-1
 icms.r.log<- bgeral[rang2,"icms.r.log"]


 r2<-length(icms.r.log)


 x1<-matrix(cbind(x_1, x_2, icms.ten, jan,
     fev, mar, abr, mai, jun, jul, ago, set,
     out, nov, dez), r2, var)


## Previsão um passo à frente

 net.val.sim.1.1  <-   sim.MLPnet(net.1.1, x1)

### medidas de erro na previsão

  e.1.1     <- exp(icms.r.log) - exp(net.val.sim.1.1)
  vMSE.1.1  <- mean(e.1.1^2);
  vMAE.1.1  <- mean(abs(e.1.1));
  vMAPE.1.1 <- mean( abs(e.1.1/exp(icms.r.log))) * 100;
  vME.1.1   <- mean(e.1.1);
  vMAD.1.1  <- median( abs( e.1.1 - median(e.1.1)) )
  er.1.1    <- ((exp(net.val.sim.1.1) - exp(icms.r.log))/
                exp(icms.r.log) )*100


## previsão múltiplos passos à frente

 minten<-min(bgeral[(1:120),"icms.ten"])
 maxten<-max(bgeral[(1:120),"icms.ten"])
 minicms<-min(bgeral[(1:120),"icms.r.log"])
 maxicms<-max(bgeral[(1:120),"icms.r.log"])
 minx1<-min(bgeral[(2:120),"x_1"])
 maxx1<-max(bgeral[(2:120),"x_1"])
 minx2<-min(bgeral[(3:120),"x_2"])
 maxx2<-max(bgeral[(3:120),"x_2"])
 r<-120

 prev<-matrix(0,6)
```

```
prevmul<-function(l){
  for(k in 2:6){
      vt<-matrix(0,12)
      vt[k+6]<-1
      ten<-base$icmstrend[r+k]
      ten<- 2*((ten-minten)/(maxten-minten)) - 1
      predx1<- ifelse(k>1,prev[k-1],bgeral[(r+k-1),"icms.r.log"])
      predx2<- ifelse(k>2,prev[k-2],bgeral[(r+k-2),"icms.r.log"])
      predx1<- 2*((predx1-minx1)/(maxx1-minx1)) - 1
      predx2<- 2*((predx2-minx2)/(maxx2-minx2)) - 1
      vc<-matrix(c(predx1,predx2,ten,vt),1)
      prev[k]<- sim.MLPnet(l, vc)
  }
  return(prev)
}

 prev<-matrix(0,6);prev[1]<-net.val.sim.1.1[1]
 prev<-prevmul(net.1.1);prev
 R1<-mean(abs((exp(prevmul(net.1.1))-exp(icms.r.log)))/
     exp(icms.r.log))*100
 R1
 mean(abs(er))
```

APÊNDICE B

Combinação de previsões da análise I

As previsões do *ICMS*, *ICMSPE*, *ICMSRJ* e *ICMSSP* resultantes da combinação de previsões individuais, na análise I, são apresentadas a seguir. A entrada *data* corresponde à data da observação sendo prevista; `real` é o verdadeiro valor da arrecadação; `prev-o` é a previsão um passo à frente; `prev-m` é a previsão quatro passos à frente e `er` é o erro relativo. Os resultados estão expressos em milhões de reais.

Tabela B.1: Previsão do *ICMS* através de *HW-SARIMA-ANN*.

data	real	prev-o	er(%)	prev-m	er(%)
dez/04	11746.0785	11739.9570	−0.0521	11739.9570	−0.0521
jan/05	11331.6471	11527.7379	1.7305	11638.3562	2.7067
fev/05	11042.4777	10619.2897	−3.8324	10801.1235	−2.1857
mar/05	11109.9315	10946.5507	−1.4706	10859.8796	−2.2507

Tabela B.2: Previsão do *ICMSPE* através de *HW-SARIMA-ANN*.

data	real	prev-o	er(%)	prev-m	er(%)
dez/04	337.2400	334.9881	−0.6677	334.9881	−0.6677
jan/05	368.9698	347.1047	−5.9260	346.1564	−6.1830
fev/05	339.0253	338.9440	−0.0240	336.0962	−0.8640
mar/05	303.3947	322.1060	6.1673	318.5606	4.9987

Tabela B.3: Previsão do *ICMSRJ* através de *HW-SARIMA-ANN*.

data	real	prev-o	er(%)	prev-m	er(%)
dez/04	1155.0683	1070.5765	−7.3149	1070.5765	−7.3149
jan/05	1269.0384	1224.9986	−3.4703	1264.6625	−0.3448
fev/05	1060.0536	1025.9019	−3.2217	1068.8230	0.8273
mar/05	1033.2296	1032.1196	−0.1074	1030.2287	−0.2904

Tabela B.4: Previsão do *ICMSSP* através de *HW-SARIMA-ANN*.

data	real	prev-o	er(%)	prev-m	er(%)
dez/04	4148.0546	4147.0347	−0.0246	4147.0347	−0.0246
jan/05	4188.3608	4161.2137	−0.6482	4144.2159	−1.0540
fev/05	3844.5941	3740.2795	−2.7133	3723.7135	−3.1442
mar/05	3942.9386	3906.4678	−0.9250	3807.0031	−3.4476

Tabela B.5: Previsão do *ICMS* através de *HW-ANN*.

data	real	prev-o	er(%)	prev-m	er(%)
dez/04	11746.0785	11739.6007	−0.0551	11739.6007	−0.0551
jan/05	11331.6471	11716.0274	3.3921	11645.2675	2.7677
fev/05	11042.4777	10657.9787	−3.4820	10827.8764	−1.9434
mar/05	11109.9315	10964.8765	−1.3056	10887.3866	−2.0031

Tabela B.6: Previsão do *ICMSPE* através de *HW-ANN*.

data	real	prev-o	er(%)	prev-m	er(%)
dez/04	337.2400	335.2016	−0.6044	335.2016	−0.6044
jan/05	368.9698	347.7641	−5.7473	347.1750	−5.9069
fev/05	339.0253	338.9401	−0.0251	336.3982	−0.7749
mar/05	303.3947	320.1562	5.5246	319.9484	5.4562

Tabela B.7: Previsão do *ICMSRJ* através de *HW-ANN*.

data	real	prev-o	er(%)	prev-m	er(%)
dez/04	1155.0683	1071.0081	−7.2775	1071.0081	−7.2775
jan/05	1269.0384	1298.5862	2.3284	1307.8492	3.0583
fev/05	1060.0536	1015.5414	−4.1990	1066.9966	0.6550
mar/05	1033.2296	1031.7312	−0.1450	1024.6704	−0.8284

Tabela B.8: Previsão do *ICMSSP* através de *HW-ANN*.

data	real	prev-o	er(%)	prev-m	er(%)
dez/04	4148.0546	4147.0594	−0.0240	4147.0594	−0.0240
jan/05	4188.3608	4148.1624	−0.9598	4148.3346	−0.9557
fev/05	3844.5941	3719.5331	−3.2529	3727.0198	−3.0582
mar/05	3942.9386	3825.1352	−2.9877	3794.9145	−3.7542

Tabela B.9: Previsão do *ICMS* através de *SARIMA-ANN*.

data	real	prev-o	er(%)	prev-m	er(%)
dez/04	11746.0785	11737.9988	−0.0688	11737.9988	−0.0688
jan/05	11331.6471	11504.4484	1.5249	11615.3734	2.5038
fev/05	11042.4777	10635.8759	−3.6822	10634.1103	−3.6981
mar/05	11109.9315	10825.0734	−2.5640	10692.7682	−3.7549

Tabela B.10: Previsão do *ICMSPE* através de *SARIMA-ANN*.

data	real	prev-o	er(%)	prev-m	er(%)
dez/04	337.2400	334.7291	−0.7445	334.7291	−0.7445
jan/05	368.9698	342.8954	−7.0668	341.6596	−7.4017
fev/05	339.0253	340.6862	0.4899	336.4831	−0.7499
mar/05	303.3947	320.5484	5.6539	317.6834	4.7096

Tabela B.11: Previsão do *ICMSRJ* através de *SARIMA-ANN*.

data	real	prev-o	er(%)	prev-m	er(%)
dez/04	1155.0683	1076.2721	−6.8218	1076.2721	−6.8218
jan/05	1269.0384	1088.8940	−14.1953	1069.4294	−15.7292
fev/05	1060.0536	1034.7327	−2.3886	1071.3034	1.0612
mar/05	1033.2296	1044.5907	1.0996	1072.2880	3.7802

Tabela B.12: Previsão do *ICMSSP* através de *SARIMA-ANN*.

data	real	prev-o	er(%)	prev-m	er(%)
dez/04	4148.0546	4146.8682	−0.0286	4146.8682	−0.0286
jan/05	4188.3608	4162.3785	−0.6203	4146.9410	−0.9889
fev/05	3844.5941	3750.6850	−2.4426	3735.5889	−2.8353
mar/05	3942.9386	3911.0368	−0.8091	3820.5712	−3.1035

Tabela B.13: Previsão do *ICMS* através de *HW-SARIMA*.

data	real	prev-o	er(%)	prev-m	er(%)
dez/04	11746.0785	11794.4945	0.4122	11794.4945	0.4122
jan/05	11331.6471	11504.9961	1.5298	11662.6557	2.9211
fev/05	11042.4777	10523.4316	−4.7004	10827.6282	−1.9457
mar/05	11109.9315	10963.6148	−1.3170	10887.0668	−2.0060

Tabela B.14: Previsão do *ICMSPE* através de *HW-SARIMA*.

data	real	prev-o	er(%)	prev-m	er(%)
dez/04	337.2400	334.8560	−0.7069	334.8560	−0.7069
jan/05	368.9698	349.0114	−5.4092	347.9463	−5.6979
fev/05	339.0253	338.9445	−0.0238	331.5933	−2.1922
mar/05	303.3947	330.3601	8.8879	318.3845	4.9407

Tabela B.15: Previsão do *ICMSRJ* através de *HW-SARIMA*.

data	real	prev-o	er(%)	prev-m	er(%)
dez/04	1155.0683	1062.0899	−8.0496	1062.0899	−8.0496
jan/05	1269.0384	1270.5393	0.1183	1307.6756	3.0446
fev/05	1060.0536	1035.3119	−2.3340	1066.4244	0.6010
mar/05	1033.2296	1026.7216	−0.6299	1025.6861	−0.7301

Tabela B.16: Previsão do *ICMSSP* através de *HW-SARIMA*.

data	real	prev-o	er(%)	prev-m	er(%)
dez/04	4148.0546	4155.3072	0.1748	4155.3072	0.1748
jan/05	4188.3608	4164.5763	−0.5679	4126.9496	−1.4662
fev/05	3844.5941	3739.8358	−2.7248	3691.3524	−3.9859
mar/05	3942.9386	3910.2606	−0.8288	3797.4303	−3.6904

APÊNDICE C

Combinação de previsões da análise II, com redes neurais utilizando o neurônio de tendência

As previsões do *ICMS*, *ICMSPE*, *ICMSRJ* e *ICMSSP* resultantes da combinação de previsões individuais, na análise II, são apresentadas a seguir. Aqui, diferentemente do apêndice anterior, `prev-m` indica as previsões seis passos à frente.

Tabela C.1: Previsão do *ICMS* através de *HW-SARIMA-ANN*$_{ct}$.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	11512.3622	11433.6264	−0.6839	11433.6264	−0.6839
ago/04	12110.8247	11402.1070	−5.8519	11355.8463	−6.2339
set/04	12208.2788	12068.0305	−1.1488	11545.8952	−5.4257
out/04	12127.8133	12205.9839	0.6446	11628.7940	−4.1147
nov/04	11043.4190	11819.8540	7.0307	11407.5575	3.2973
dez/04	11746.1057	11875.3220	1.1001	11909.5261	1.3913

Tabela C.2: Previsão do *ICMSPE* através de *HW-SARIMA-ANN*$_{ct}$.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	309.9951	292.4090	−5.6730	292.4090	−5.6730
ago/04	315.8774	298.8231	−5.3990	295.0691	−6.5875
set/04	322.1099	317.7181	−1.3635	300.3465	−6.7565
out/04	319.6628	318.4567	−0.3773	305.4525	−4.4454
nov/04	326.7480	330.4460	1.1318	311.6758	−4.6128
dez/04	337.2409	334.3746	−0.8499	316.0451	−6.2851

Tabela C.3: Previsão do *ICMSRJ* através de *HW-SARIMA-ANN*$_{ct}$.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	1107.9730	1027.7980	−7.2362	1027.7980	−7.2362
ago/04	1117.9948	1064.6179	−4.7743	1037.7229	−7.1800
set/04	1074.0076	1074.5967	0.0548	1026.2330	−4.4483
out/04	1023.8763	1023.9929	0.0114	1029.9895	0.5971
nov/04	1087.5180	1082.6491	−0.4477	1031.5490	−5.1465
dez/04	1155.0671	1049.9799	−9.0979	1023.4810	−11.3921

Tabela C.4: Previsão do *ICMSSP* através de *HW-SARIMA-ANN*$_{ct}$.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	3832.8642	3921.1487	2.3034	3921.1487	2.3034
ago/04	3979.9547	3894.1397	−2.1562	3897.3182	−2.0763
set/04	4102.8901	4075.0732	−0.6780	4056.5921	−1.1284
out/04	4068.6564	4087.4867	0.4628	4110.9813	1.0403
nov/04	4175.0360	4089.3559	−2.0522	4165.3005	−0.2332
dez/04	4148.0694	4142.8503	−0.1258	4147.4178	−0.0157

Tabela C.5: Previsão do *ICMS* através de *HW-ANN*$_{ct}$.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	11512.3622	11439.7171	−0.6310	11439.7171	−0.6310
ago/04	12110.8247	11360.8001	−6.1930	11317.3242	−6.5520
set/04	12208.2788	11951.2320	−2.1055	11457.8434	−6.1469
out/04	12127.8133	12253.1280	1.0333	11522.5423	−4.9908
nov/04	11043.4190	11737.1108	6.2815	11365.3885	2.9155
dez/04	11746.1057	11870.5800	1.0597	11894.0598	1.2596

Tabela C.6: Previsão do *ICMSPE* através de $HW\text{-}ANN_{ct}$.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	309.9951	293.5775	−5.2961	293.5775	−5.2961
ago/04	315.8774	292.5665	−7.3797	294.3849	−6.8041
set/04	322.1099	312.3306	−3.0360	301.4070	−6.4273
out/04	319.6628	310.9964	−2.7111	305.8991	−4.3057
nov/04	326.7480	329.5209	0.8486	311.8791	−4.5506
dez/04	337.2409	334.8106	−0.7206	316.3554	−6.1931

Tabela C.7: Previsão do *ICMSRJ* através de $HW\text{-}ANN_{ct}$.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	1107.9730	1030.7174	−6.9727	1030.7174	−6.9727
ago/04	1117.9948	1062.4950	−4.9642	1040.5261	−6.9293
set/04	1074.0076	1027.6970	−4.3119	1022.8923	−4.7593
out/04	1023.8763	1022.4599	−0.1383	1029.3305	0.5327
nov/04	1087.5180	1083.1552	−0.4012	1031.9780	−5.1070
dez/04	1155.0671	1037.8293	−10.1499	1019.1543	−11.7667

Tabela C.8: Previsão do *ICMSSP* através de $HW\text{-}ANN_{ct}$.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	3832.8642	3912.2849	2.0721	3912.2849	2.0721
ago/04	3979.9547	3903.4804	−1.9215	3890.9608	−2.2361
set/04	4102.8901	4021.5539	−1.9824	4013.1431	−2.1874
out/04	4068.6564	4085.8879	0.4235	4105.5128	0.9059
nov/04	4175.0360	4062.7217	−2.6901	4071.0623	−2.4904
dez/04	4148.0694	4144.0223	−0.0976	4147.4856	−0.0141

Tabela C.9: Previsão do *ICMS* através de $SARIMA\text{-}ANN_{ct}$.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	11512.3622	11439.8393	−0.6300	11439.8393	−0.6300
ago/04	12110.8247	11444.9389	−5.4983	11423.6509	−5.6740
set/04	12208.2788	12086.6499	−0.9963	11611.5709	−4.8877
out/04	12127.8133	12190.6196	0.5179	11679.2979	−3.6982
nov/04	11043.4190	11737.7781	6.2875	11765.0170	6.5342
dez/04	11746.1057	11913.2782	1.4232	12045.2762	2.5470

Tabela C.10: Previsão do *ICMSPE* através de *SARIMA-ANN*$_{ct}$.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	309.9951	290.0210	−6.4434	290.0210	−6.4434
ago/04	315.8774	300.3441	−4.9175	296.7620	−6.0515
set/04	322.1099	318.3443	−1.1691	299.6726	−6.9657
out/04	319.6628	322.6471	0.9336	306.8123	−4.0200
nov/04	326.7480	330.1436	1.0392	311.1906	−4.7613
dez/04	337.2409	330.0454	−2.1336	314.8328	−6.6445

Tabela C.11: Previsão do *ICMSRJ* através de *SARIMA-ANN*$_{ct}$.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	1107.9730	1019.8906	−7.9499	1019.8906	−7.9499
ago/04	1117.9948	1057.1456	−5.4427	1031.5766	−7.7298
set/04	1074.0076	1089.3388	1.4275	1030.8895	−4.0147
out/04	1023.8763	1013.4244	−1.0208	1030.7005	0.6665
nov/04	1087.5180	1055.0821	−2.9826	1030.8157	−5.2139
dez/04	1155.0671	1049.1559	−9.1693	1030.7841	−10.7598

Tabela C.12: Previsão do *ICMSSP* através de *SARIMA-ANN*$_{ct}$.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	3832.8642	3961.4309	3.3543	3961.4309	3.3543
ago/04	3979.9547	3902.9414	−1.9350	3903.7061	−1.9158
set/04	4102.8901	4078.3904	−0.5971	4061.9522	−0.9978
out/04	4068.6564	4086.3295	0.4344	4106.6623	0.9341
nov/04	4175.0360	4096.8881	−1.8718	4165.9180	−0.2184
dez/04	4148.0694	4137.4119	−0.2569	4146.1005	−0.0475

Tabela C.13: Previsão do *ICMS* através de *HW-SARIMA*.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	11512.3622	11403.1946	−0.9483	11403.1946	−0.9483
ago/04	12110.8247	11389.4541	−5.9564	11296.0188	−6.7279
set/04	12208.2788	12078.3611	−1.0642	11525.3197	−5.5942
out/04	12127.8133	12214.0987	0.7115	11634.5500	−4.0672
nov/04	11043.4190	12285.1373	11.2440	11364.0556	2.9034
dez/04	11746.1057	11862.4802	0.9908	11889.6119	1.2217

Tabela C.14: Previsão do *ICMSPE* através de *HW-SARIMA*.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	309.9951	292.9145	−5.5100	292.9145	−5.5100
ago/04	315.8774	300.3056	−4.9297	293.3983	−7.1164
set/04	322.1099	318.0525	−1.2597	299.7697	−6.9356
out/04	319.6628	319.9597	0.0929	302.2442	−5.4491
nov/04	326.7480	331.2877	1.3893	311.9153	−4.5395
dez/04	337.2409	334.5308	−0.8036	316.7633	−6.0721

Tabela C.15: Previsão do *ICMSRJ* através de *HW-SARIMA*.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	1107.9730	1031.0145	−6.9459	1031.0145	−6.9459
ago/04	1117.9948	1069.9900	−4.2938	1039.9336	−6.9823
set/04	1074.0076	1080.2567	0.5818	1022.3867	−4.8064
out/04	1023.8763	1042.9407	1.8620	1029.2716	0.5270
nov/04	1087.5180	1082.9125	−0.4235	1031.8253	−5.1211
dez/04	1155.0671	1058.6729	−8.3453	1018.9614	−11.7834

Tabela C.16: Previsão do *ICMSSP* através de *HW-SARIMA*.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	3832.8642	3912.9923	2.0906	3912.9923	2.0906
ago/04	3979.9547	3842.2028	−3.4611	3894.9638	−2.1355
set/04	4102.8901	4078.3316	−0.5986	4062.0667	−0.9950
out/04	4068.6564	4226.6410	3.8830	4196.4133	3.1400
nov/04	4175.0360	4095.7020	−1.9002	4165.9287	−0.2181
dez/04	4148.0694	4159.3925	0.2730	4150.9988	0.0706

APÊNDICE D

Combinação de previsões da análise II, com redes neurais sem o neurônio de tendência

As previsões do *ICMS*, *ICMSPE*, *ICMSRJ* e *ICMSSP* resultantes da combinação de previsões individuais na análise II, são apresentadas a seguir. Como no apêndice anterior, `prev-m` indica as previsões seis passos à frente.

Tabela D.1: Previsão do *ICMS* através de *HW-SARIMA-ANN*$_{st}$.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	11512.3622	11388.3589	−1.0771	11388.3589	−1.0771
ago/04	12110.8247	11355.2347	−6.2390	11247.0014	−7.1327
set/04	12208.2788	12100.2714	−0.8847	11523.2146	−5.6115
out/04	12127.8133	12217.6793	0.7410	11769.2739	−2.9563
nov/04	11043.4190	11894.4204	7.7060	11407.4814	3.2966
dez/04	11746.1057	11819.6618	0.6262	11860.5632	0.9744

Tabela D.2: Previsão do *ICMSPE* através de *HW-SARIMA-ANN*$_{st}$.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	309.9951	291.0581	−6.1088	291.0581	−6.1088
ago/04	315.8774	305.6118	−3.2499	296.7495	−6.0555
set/04	322.1099	317.5305	−1.4217	300.7115	−6.6432
out/04	319.6628	319.7934	0.0408	308.3079	−3.5521
nov/04	326.7480	330.1026	1.0267	312.6408	−4.3175
dez/04	337.2409	334.3112	−0.8687	316.2675	−6.2191

Tabela D.3: Previsão do *ICMSRJ* através de *HW-SARIMA-ANN*$_{st}$.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	1107.9730	1040.2732	−6.1102	1040.2732	−6.1102
ago/04	1117.9948	1070.3582	−4.2609	1039.0027	−7.0655
set/04	1074.0076	1076.7235	0.2529	1024.9239	−4.5701
out/04	1023.8763	1045.0429	2.0673	1023.2164	−0.0645
nov/04	1087.5180	1082.6872	−0.4442	1028.7557	−5.4033
dez/04	1155.0671	1058.8884	−8.3267	1018.4314	−11.8292

Tabela D.4: Previsão do *ICMSSP* através de *HW-SARIMA-ANN*$_{st}$.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	3832.8642	3921.1487	2.3034	3921.1487	2.3034
ago/04	3979.9547	3894.1397	−2.1562	3897.3182	−2.0763
set/04	4102.8901	4075.0732	−0.6780	4056.5921	−1.1284
out/04	4068.6564	4087.4867	0.4628	4110.9813	1.0403
nov/04	4175.0360	4089.3559	−2.0522	4165.3005	−0.2332
dez/04	4148.0694	4142.8503	−0.1258	4147.4178	−0.0157

Tabela D.5: Previsão do *ICMS* através de *HW-ANN*$_{st}$.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	11512.3622	11177.6860	−2.9071	11177.6860	−2.9071
ago/04	12110.8247	11262.8482	−7.0018	11059.3485	−8.6821
set/04	12208.2788	12100.4362	−0.8834	11398.3220	−6.6345
out/04	12127.8133	12244.8436	0.9650	11791.1370	−2.7761
nov/04	11043.4190	11811.2007	6.9524	11365.4370	2.9159
dez/04	11746.1057	11789.9612	0.3734	11852.3758	0.9047

Tabela D.6: Previsão do *ICMSPE* através de *HW-ANN*$_{st}$.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	309.9951	292.0697	−5.7825	292.0697	−5.7825
ago/04	315.8774	306.5181	−2.9630	297.0334	−5.9656
set/04	322.1099	313.1909	−2.7689	301.8492	−6.2900
out/04	319.6628	319.7909	0.0401	309.3638	−3.2218
nov/04	326.7480	329.7561	0.9206	313.1799	−4.1525
dez/04	337.2409	334.7345	−0.7432	316.6666	−6.1008

Tabela D.7: Previsão do *ICMSRJ* através de *HW-ANN*$_{st}$.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	1107.9730	1045.3974	−5.6478	1045.3974	−5.6478
ago/04	1117.9948	1071.5417	−4.1550	1042.1621	−6.7829
set/04	1074.0076	1076.4713	0.2294	1020.1903	−5.0109
out/04	1023.8763	1043.0829	1.8759	1023.1293	−0.0730
nov/04	1087.5180	1083.1949	−0.3975	1027.6955	−5.5008
dez/04	1155.0671	1055.7034	−8.6024	1010.1130	−12.5494

Tabela D.8: Previsão do *ICMSSP* através de *HW-ANN*$_{st}$.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	3832.8642	3912.2849	2.0721	3912.2849	2.0721
ago/04	3979.9547	3903.4804	−1.9215	3890.9608	−2.2361
set/04	4102.8901	4021.5539	−1.9824	4013.1431	−2.1874
out/04	4068.6564	4085.8879	0.4235	4105.5128	0.9059
nov/04	4175.0360	4062.7217	−2.6901	4071.0623	−2.4904
dez/04	4148.0694	4144.0223	−0.0976	4147.4856	−0.0141

Tabela D.9: Previsão do *ICMS* através de *SARIMA-ANN*$_{st}$.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	11512.3622	11403.1837	−0.9484	11403.1837	−0.9484
ago/04	12110.8247	11389.1046	−5.9593	11312.2561	−6.5938
set/04	12208.2788	12111.2642	−0.7947	11590.6818	−5.0588
out/04	12127.8133	12208.5379	0.6656	11805.1374	−2.6606
nov/04	11043.4190	11812.8063	6.9669	11759.9523	6.4883
dez/04	11746.1057	11781.4125	0.3006	11855.3885	0.9304

Tabela D.10: Previsão do *ICMSPE* através de $SARIMA\text{-}ANN_{st}$.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	309.9951	285.5349	−7.8905	285.5349	−7.8905
ago/04	315.8774	306.6881	−2.9092	298.4897	−5.5046
set/04	322.1099	318.0774	−1.2519	300.2777	−6.7779
out/04	319.6628	319.7958	0.0416	309.5271	−3.1708
nov/04	326.7480	329.9151	0.9693	312.7140	−4.2950
dez/04	337.2409	329.8539	−2.1904	315.2432	−6.5228

Tabela D.11: Previsão do *ICMSRJ* através de $SARIMA\text{-}ANN_{st}$.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	1107.9730	1041.4214	−6.0066	1041.4214	−6.0066
ago/04	1117.9948	1069.4401	−4.3430	1034.0746	−7.5063
set/04	1074.0076	1076.9174	0.2709	1029.6492	−4.1302
out/04	1023.8763	1057.0073	3.2358	1023.2301	−0.0631
nov/04	1087.5180	1054.9907	−2.9910	1026.1090	−5.6467
dez/04	1155.0671	1061.8874	−8.0670	1024.6763	−11.2886

Tabela D.12: Previsão do *ICMSSP* através de $SARIMA\text{-}ANN_{st}$.

data	real	prev-o	er(%)	prev-m	er(%)
jul/04	3832.8642	3961.4309	3.3543	3961.4309	3.3543
ago/04	3979.9547	3902.9414	−1.9350	3903.7061	−1.9158
set/04	4102.8901	4078.3904	−0.5971	4061.9522	−0.9978
out/04	4068.6564	4086.3295	0.4344	4106.6623	0.9341
nov/04	4175.0360	4096.8881	−1.8718	4165.9180	−0.2184
dez/04	4148.0694	4137.4119	−0.2569	4146.1005	−0.0475

Referências Bibliográficas

Barnard, G. A. (1963), New Methods of Quality Control, *Journal of the Royal Statistical Society A* **126**, 255–259.

Box, G. E. P. & Jenkins, G. M. (1976), *Time Series Analysis: Forecasting and Control*, San Francisco: Holden-Day.

Box, G. E. P. & Tiao, G. C. (1975), Intervention Analysis with Applications to Economic and Environmental Problems, *Journal of the American Statistical Association* **70**, 71–79.

Box, G. E. P., Jenkins, G. M. & Reinsel, G. (1994), *Time Series Analysis: Forecasting and Control*, 3rd ed., Englewood Cliffts: Prentice Hall.

Chen, C. & Liu, L. (1993), Joint Estimation of Model Parameters and Outliers Effects in Time Series, *Journal of the American Statistical Association* **88**, 284–297.

Chryssolouris, G., Lee, M. & Ramsey, A. (1996), Confidence Interval Prediction for Neural Networks Models, *IEEE Transactions on Neural Networks* **7**, 229–232.

Chuang, C. C., Sue, S. F. & Hsiao, C. C. (2000), The Annealing Robust Backpropagation (ARBP) Learning Algorithm, *IEEE Transaction on Neural Networks* **11**, 1067–1077.

Cohen, M., Franco, H., Morgan, N., Rumelhart, D. & Abrash, V. (1993), *Advances in Neural Information Processing Systems*, Morgan Kaufmann, capítulo Context-dependent multiple distribution phonetic modeling with MLPs, pp. 649–657.

Cribari-Neto, F. & Zarkos, S. G. (1999), `R`: Yet Another Econometric Programming Enviroment, *Journal of Applied Econometrics* **14**, 319–329.

Cybenko, M. (1989), Aproximation by Superposition of a Sigmoidal Function, *Mathematics of Control, Signals and Systems* **2**, 303–314.

Dijk, D. V. (1999), Smooth Transition Models: Extentions and Outlier Robust Inference, PhD thesis, Tinbergen Institute. Rotterdam, The Netherlands. `www.few.eur.nl/few/people/djvandijk/thesis`.

Doornik, J. A. & Hansen, H. (1994), A Omnibus Test for Univariate and Multivariate Normality, `www.nuff.ox.ac.uk/Users/Doorknik/`.

Fernandes, L. L. (1995), Utilização de Redes Neurais na Análise e Previsão de Séries Temporais. Dissertação de Mestrado. Universidade Federal do Rio Grande do Sul - Pós-Graduação da computação.

Gately, E. (1996), *Neural Networks for Financial Forecasting*, New York: John Wiley and Sons.

Ghiassi, M. & Saidane, H. (2005), A Dynamic Arquitecture for Artificial Neural Networks, *Neurocomputing* **63**, 397–413.

Ghiassi, M., Saidane, H. & Zimbra, D. K. (2005), A Dynamic Artificial Neural Network Model for Forecasting Time Series Events, *International Journal of Forecasting* **21**, 341–362.

Goméz, V. & Maravall, A. (1996), Programs TRAMO (Time series Regression with Arima noise, Missing observations, and Outliers) and SEATS (Signal Extraction in Arima Time Series). Instructions for the User, Technical Report, Working Paper 9628. Servicio de Estudios–Banco de España.

Gómez, V. & Maravall, A. (2001), *Automatic Modelling Methods for Univariate Series*, New York: John Wiley and Sons, capítulo 7, em Peña D., Tiao G.C. & Tsay R.S, A Course in Advanced Time Series Analysis.

Granger, C. W. & Newbold, P. (1977), *Forecasting Economic Time Series*, New York: Academic Press.

Guerrero, V. (1991), *Análisis de Series de Tiempo Económicas*, Universidad Autónoma Metropolitana de México D.F.

Haykin, S. (1994), *Neural Networks*, New York: McMillan College Publishing Company.

Hendry, D. F. & Clements, M. P. (2004), Pooling of Forecast, *Econometrics Journal* **7**, 1–31.

Hornik, K., Stinchcombe, M. & White, H. (1989), Multilayer Feedforward Networks and Universal Approximations, *Neural Networks* **2**, 359–366.

Isasi, P. & Galván, I. (2004), *Redes Neuronales Artificiales – Un Enfoque Práctico*, Madrid: Pearson-Prentice Hall.

Kaastra, I. & Boyd, M. (1996), Design a Neural Network for Forecasting Financial and Economic Time Series, *Neurocomputing* **10**, 215–236.

Kaiser, R. & Maravall, A. (2001), Seasonal Outliers in Time Series, `www.bde.es/servicio/software/trabajos.htm`.

Klimasauskas, C. C. (1994), Applying Neural Networks, *em* 'Neural Networks in Finance and Investing: Using Artificial Intelligence to Improve Real World Performance', R.R. Trippi and E. Turban, pp. 64–65.

Lee, Y. O. & Kim, M. (1991), The Effect of Initial Weights on Premature Saturation in Back-Propagation Learning, *em* 'International Conference of Neural Networks', Vol. 1, pp. 765–770.

Liano, K. (1996), Robust Error Measure for Supervised Neural Networks Learning with Outliers, *IEEE Transactions on Neural Networks* **7**, 246–250.

Ljung, G. M. & Box, G. E. P. (1978), On a Measure of Lack of Fit in Time Series Models, *Biometrika* **65**, 297–303.

Ljung, G. M. & Box, G. E. P. (1986), Time Series Model Specification in Presence of Outliers, *Journal of the American Statistical Association* **81**, 132–141.

Maravall, A. & Kaiser, R. (2000), Notes on Time Series Analysis, ARIMA models and Signal Extraction, `www.bde.es/servicio/software/trabajos.htm`.

McLeod, A. I. & Li, W. K. (1983), Diagnostic Checking ARMA Time Series Models Using Squared-Residuals Autocorrelation, *Journal of the Time Series Analysis* **4**, 269–273.

Medeiros, M., Teräsvirta, T. & Rech, G. (2002), Building Neural Networks Models for Time Series: A Statistical Approach, Technical Report 461, Departament of Economics, Catholic University of Rio de Janeiro.

Mills, T. (1999), *The Econometric Modelling of Financial Time Series*, 2nd ed., Cambridge University Press.

Minsky, M. & Papert, S. (1969), *Perceptrons: An Introduction to Computational Geometry*, Cambridge: The MIT Press.

Minsky, M. & Papert, S. (1988), *Perceptrons*, Cambridge: The MIT Press. Expanded Edition.

Montana, D. & Davis, L. (1989), Training Feedforward Neural Networks Using Genetic Algorithms, *em* 'International Joint Conference on Artificial Intelligence', Morgan Kaufmann.

Montgomery, D. C. & Johnson, L. A. (1976), *Forecasting and Time Series Analysis*, New York: McGrawn Hill.

Morettin, P. A. & Toloi, C. M. (2004), *Análise de Séries Temporais*, ABE - Projeto Fisher. São Paulo: Edgard Blücher.

Narendra, K. & Parthasaranty, K. (1990), Identification and Control of Dynamical Systems Using Neural Networks, *IEEE Transactions on Neural Networks* **1**, 4–27.

Newbold, P. & Granger, J. (1974), Experience with Forecasting Univariate Time Series and the Combination of Forecasts, *Journal of the Royal Statistical Society* **137**, 131–146.

Nocedal, J. & Wright, S. (1999), *Numerical Optimization*, New York: Springer-Verlag.

Peña, D. (2001), *Outliers Influencial Observations and Missing data*, New York: John Wiley and Sons, capítulo 6, em Peña D., Tiao G.C. & Tsay R.S, A Course in Advanced Time series Analysis.

Pernía-Espinoza, A., Ordieres, J., de Pisón, F. & Marcos, A. (2005), TAO-Robust Backpropagation Learning Algorithm, *Neural Networks* **18**, 191–204.

Pham, D. T. & Xing, L. (1995), *Neural Networks for Identification, Prediction and Control*, New York: Springer-Verlag.

Rumelhart, D., Hilton, G. & Williams, R. (1986*a*), Learning Representations by Backpropagating Errors, *Nature* **323**, 533–536.

Rumelhart, D., Hilton, G. & Williams, R. (1986*b*), *Parallel Distributed Processing*, Cambridge: The MIT Press, capítulo Learning representations by backpropagating errors.

Solis, F. & Wets, J. (1981), Minimization by Random Search Techniques, *Mathematics of Operations Research* **6**, 19–30.

Souza, R. C. & Zandonade, E. (1993), Forecasting Via Neural Networks: Comparative Study, Technical Report, Department of Electrical Engineering, Catholic University of Rio de Janeiro.

Varfis, A. & Versino, C. (1990), *Univariate Economic Time Series Forecasting*, Cambridge University Press.

Venables, W. N. & Ripley, B. D. (2002), *Modern Applied Statistics with* `S`, 4th ed., New York: Springer-Verlag.

Widrow, B. (1960), An Adaptative Adaline Neuron Using Chemical Memistors, Technical Report 1553-2, Standford Electronics Laboratory.

Wieggend, A., Huberman, B. & Rumelhart, D. (1990), Predicting the Future: a Connectionist Approach, Technical Report, PARC.

Zhang, G., Patuwo, B. & Hu, Y. (1998), Forecasting with Artificial Neural Networks: The State of Art, *International Journal of Forecasting* **14**, 35–62.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo