

Redes neurais artificiais MLP: Previsão de séries temporais

Cleibson Aparecido de Almeida (UAb), contato@cleibsonalmeida.blog.br

Elaine Cristina da Silva Schilipack (UFPR), elaineschilipack@gmail.com

Jorge Luiz Farias (CENIPA), fariasjlf@fab.mil.br

Nivaldo Aparecido Minervi (PRF), nivaldominervi@yahoo.com.br

Resumo:

O presente trabalho apresenta a aplicação de redes neurais artificiais para modelar de séries temporais. Para isso foi utilizado um conjunto de dados contendo 168 observações referente a produção mensal de leite durante os anos de 1962 até 1975, com sazonalidade de 12 meses. Os resultados foram obtidos após uma simulação computacional e comparação entre 2160 diferentes estruturas de redes perceptron de múltiplas camadas $NNA(p, P, k)_{12}$. Foi adotado o pacote *forecast* para realizar o treinamento e previsão da rede. Com isso, o modelo $NNA(6, 6, 57)_{12}$ foi escolhido para prever os valores da produção de leite por apresentar o menor valor de R^2 . O texto apresenta os conceitos teóricos na Introdução, em seguida é apresentada a Aplicação e por fim são feitas as Considerações Finais. Por fim, são apresentados a listagem dos autores citados e o script utilizado na modelagem.

Palavras-chave: Séries Temporais, Redes Neurais Artificiais, MLP.

Artificial neural networks MLP: Time series prediction

Abstract:

The present paper shows the application of artificial neural networks to model time series. For this purpose, a data set containing 168 observations regarding monthly milk production was used during the years 1962 to 1975, with a seasonality of 12 months. The results were obtained after a computational simulation and comparison between 2160 different structures of perceptron networks of multiple layers $NNA(p, P, k)_{12}$. The forecast package was adopted to perform training and forecasting of the network. Thus, the $NNA(6, 6, 57)_{12}$ model was chosen to predict milk production values by having the smallest value of R^2 . The text presents the theoretical concepts in the Introduction, then the Application is presented and finally the Final Considerations are made. Finally, the list of authors cited and the script used in the modeling are presented.

Keywords: Time Series, Artificial Neural Networks, MLP.

1 Introdução

As redes neurais artificiais (RNA) são sistemas paralelamente distribuídos compostos por unidades de processamento simples, conhecidos como neurônios artificiais, que realizam cálculos por meio de determinadas funções matemáticas (normalmente não-lineares). Tais unidades de processamento são dispostas em uma ou mais camadas e interligadas por um determinado número de conexões, geralmente unidirecionais. Essas conexões, na maioria dos modelos de redes neurais, estão associadas a pesos, os quais armazenam o conhecimento adquirido pelo modelo durante uma fase de treinamento / aprendizagem e servem para ponderar a entrada recebida por cada neurônio da rede (BRAGA; CARVALHO; LUDERMIR, 2007).

Segundo Khashei e Bijari (1943), as RNA são estruturas flexíveis e aproximadores universais que podem ser utilizados para a previsão de séries temporais com excelente grau de acurácia. Entretanto, como

qualquer técnica de matemática computacional, elas apresentam algumas desvantagens, principalmente quanto à performance computacional e também na dificuldade de definição das quantidades de neurônios que irão compor as camadas internas rede, neste caso quase sempre de forma empírica como se fosse uma caixa preta (ZHAO, 2009).

1.1 Topologia de Redes Neurais

A definição do tipo e estrutura de uma RNA que resolva determinado problema é realizada após a observação de alguns fatores. Esses fatores podem ser: a) complexidade do problema a ser resolvido; b) representatividade dos dados; c) dimensionalidade do espaço de entrada; d) características dinâmicas ou estáticas do problema; e) conhecimento a priori do problema (BRAGA; CARVALHO; LUDERMIR, 2007).

De acordo com Braga, Carvalho e Ludermir (2007), os principais tipos de redes neurais são: a) Perceptron; b) Adaline; c) Memórias matriciais; d) Perceptron de múltiplas camadas (MLP); e) Rede de base radial (RBF); f) Redes auto organizáveis; g) Redes neurais sem pesos e h) Redes híbridas. Este trabalho foca na rede perceptron de múltiplas camadas.

Quanto a estrutura, cada tipo de rede listada anteriormente pode assumir diferentes arquiteturas. Por exemplo: a rede perceptron de múltiplas camadas pode assumir uma arquitetura a) 1-3-4-1 ou b) 2-5-2, ou seja, no primeiro caso (a) trata-se de uma estrutura com 1 camada de entrada, 2 camadas intermediárias, sendo que uma possui 3 neurônios e a outra possui 4 neurônios e por fim 1 camada de saída. No caso de (b), a estrutura é formada por 1 camada de entrada com 2 sinais de entrada, 1 camada intermediária com 5 neurônios e 1 camada de saída com 2 sinais.

1.2 Neurônio Artificial

O conceito de rede neural se baseia no processamento de dados emulando uma rede de neurônios biológicos que seja capaz de memorizar e recuperar uma grande quantidade de dados, que são reconhecidos de acordo com padrões baseados na experiência pré adquirida ou fornecida à rede. Devido a essa característica, as redes neurais possuem em sua composição neurônios artificiais que são excitados para fornecerem de forma integrada os resultados da rede (CORRAR; THEÓPHILO, 2004).

As primeiras ideias de neurônio artificial foram formuladas por McCulloch e Pitts (MCCULLOCH; PITTS, 2010). Os neurônios artificiais representam os elementos processadores de uma rede neural e são interligados entre si. Sua comparação com o cérebro humano se dá pelos seguintes motivos:

- a rede consegue aprender a partir de informações captadas em seu ambiente;
- a rede consegue memorizar o conhecimento adquirido através da conexão entre os neurônios.

Haykin (2001) apresenta o neurônio artificial (Figura 1) proposto por McCulloch e Pitts (2010) em 1943. Trata-se de um neurônio composto por três elementos estruturais, da seguinte forma:

1. um conjunto de n conexões de entrada $x_1, x_2, \dots, x_n$, caracterizadas por pesos $w_1, w_2, \dots, w_n$;
2. um somador Σ para acumular os sinais de entrada e seus respectivos erros b ;
3. uma função de ativação f que limita o intervalo permissível de amplitude do sinal de saída (y) a um valor fixo.

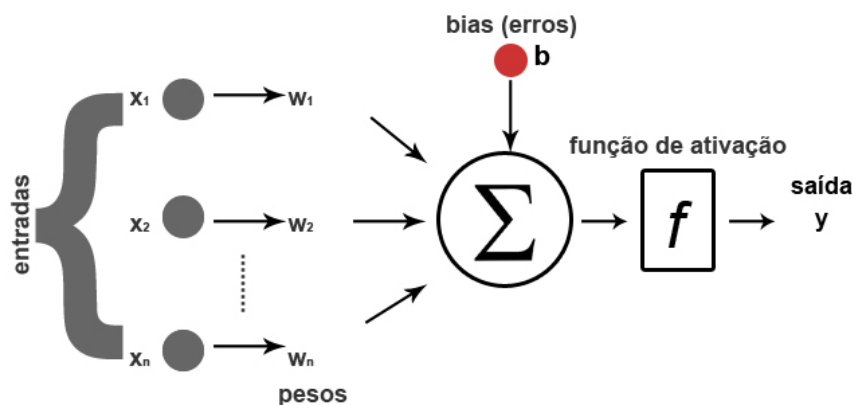


Figura 1: Exemplo de Neurônio Artificial (HAYKIN, 2001)

O comportamento das conexões entre os neurônios é simulado por meio de seus pesos w . Os valores de tais pesos podem ser negativos ou positivos, dependendo de as conexões serem inibitórias ou excitatórias. O efeito de um sinal proveniente de um outro neurônio é determinado pela multiplicação do valor do sinal recebido pelo peso da conexão correspondente (x, w). Então é efetuada a soma dos valores (x, w) de todas as conexões da rede neural, e o valor resultante é enviado para a função de ativação f , que define a saída y do neurônio.

1.3 Aprendizado Supervisionado e Não Supervisionado

Para que uma rede neural aprenda e tenha condições de fornecer resultados após uma estimulação dos neurônios, é necessário que a rede aprenda por meio de exemplos. O aprendizado de uma rede neural consiste em ajustar de forma iterativa os parâmetros e os pesos de cada conexão existente na rede. Esse aprendizado pode ser realizado de duas formas: a) supervisionado e b) não supervisionado (PURWANTO; ESWARAN; LOGESWARAN, 2010) (BRAGA; CARVALHO; LUDERMIR, 2007).

O aprendizado supervisionado consiste na comparação entre os resultados da rede após uma iteração com os padrões pré definidos inicialmente. Este tipo de aprendizado é aplicável em situações em que se deseja mapear os valores de entrada e saída da rede e um exemplo comum é o algoritmo de retro-propagação (back-propagation) em redes perceptron de múltiplas camadas. Perceba que os parâmetros da rede sempre são ajustados com base nos valores de entrada e saída (BRAGA; CARVALHO; LUDERMIR, 2007). No aprendizado não supervisionado, a comparação entre os pares de valores (entrada - saída) não existe, ou seja, é apresentado para a rede apenas o valor de entrada e o ajuste dos parâmetros é realizado por qualquer regra que não considere os valores de saída. Este tipo de aprendizado se aplica em conjuntos de dados em que possuem características muito relevantes nos valores de entrada, como, por exemplo, a descoberta de agrupamentos ou classes. Um exemplo comum que adota esse tipo de aprendizado são os mapas auto organizáveis de Kohonen (BRAGA; CARVALHO; LUDERMIR, 2007).

1.4 Rede Perceptron de Múltiplas Camadas (MLP)

Diante da dificuldade da rede perceptron simples em resolver problemas reais, que geralmente são não-lineares e complexos, foi desenvolvida a rede perceptron de múltiplas camadas (MLP) (HAYKIN, 2001).

1.4.1 Estrutura da Rede MLP

No geral, esse tipo de rede é composta por uma camada de entrada, uma ou mais camadas intermediárias (ocultas) e uma camada de saída. Na Figura 2 pode-se observar uma rede MLP com 1 camada de entrada, 1 camada intermediária contendo 4 neurônios e 1 camada de saída. Segundo Hyndman e Khandakar

(2009), no contexto das séries temporais esta rede pode ser escrita como $NNA(p, P, k)_m$, o que é similar ao modelo $ARIMA(p, 0, 0)(P, 0, 0)_m$.

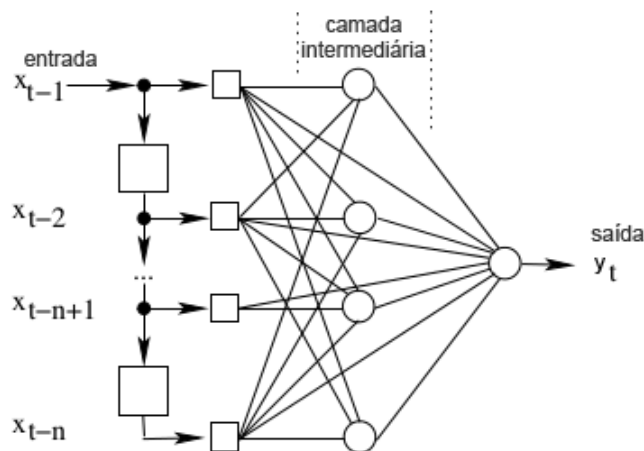


Figura 2: Exemplo de Rede MLP (CORTEZ et al., 2002)

1.4.2 Função de Ativação

De acordo com Haykin (2001), as funções de ativação ϕ mais utilizadas em redes neurais são:

- Função Degrau: Também conhecida como função discreta, seus valores podem assumir somente valores 0 ou 1;
- Função Sigmoid: Funções desse tipo podem assumir valores entre 0 e 1 e são do tipo contínuas;
- Função Linear: Uma função linear é caracterizada quando a resposta de um conjunto de dados comporta-se linearmente quando alterados os valores de entrada.

De forma geral, as redes perceptron de múltiplas camadas utilizam funções do tipo sigmóide para ativação das camadas intermediárias e funções lineares nas camadas de saída (CORRAR; THEÓPHILO, 2004).

1.4.3 Back-propagation

O treinamento de redes utilizando o back-propagation ocorre em duas fases, em que cada fase percorre a rede em um sentido. Essas duas fases são chamadas de fase de avanço (forward) e de retardo (backward) (HAYKIN, 2001). A fase forward é utilizada para definir a saída da rede para um dado padrão de entrada e a fase backward utiliza a saída desejada e a saída fornecida pela rede para atualizar os pesos de suas conexões. A Figura 3 ilustra de forma simplificada o funcionamento deste algoritmo.



Figura 3: Funcionamento do algoritmo back-propagation (OKASHA; YASEEN, 2013)

Este é o algoritmo mais utilizado em aplicações das redes neurais. O desenvolvimento deste algoritmo de aprendizagem foi um dos marcos mais importantes das pesquisas em redes neurais artificiais (LAWRENCE, 1997).

2 Aplicação

2.1 Apresentação dos Dados

Os dados utilizados nesta aplicação referem-se uma série temporal com dados mensais da produção de leite entre janeiro de 1962 e dezembro de 1975, disponíveis em (QLIKTEK, 2016). Portanto, o conjunto de dados é composto por 168 observações e cada observação representa a produção de leite (libras por vaca) referente a 1 (um) mês dentro do período da coleta dos dados. Os dados são apresentados na Figura 4.

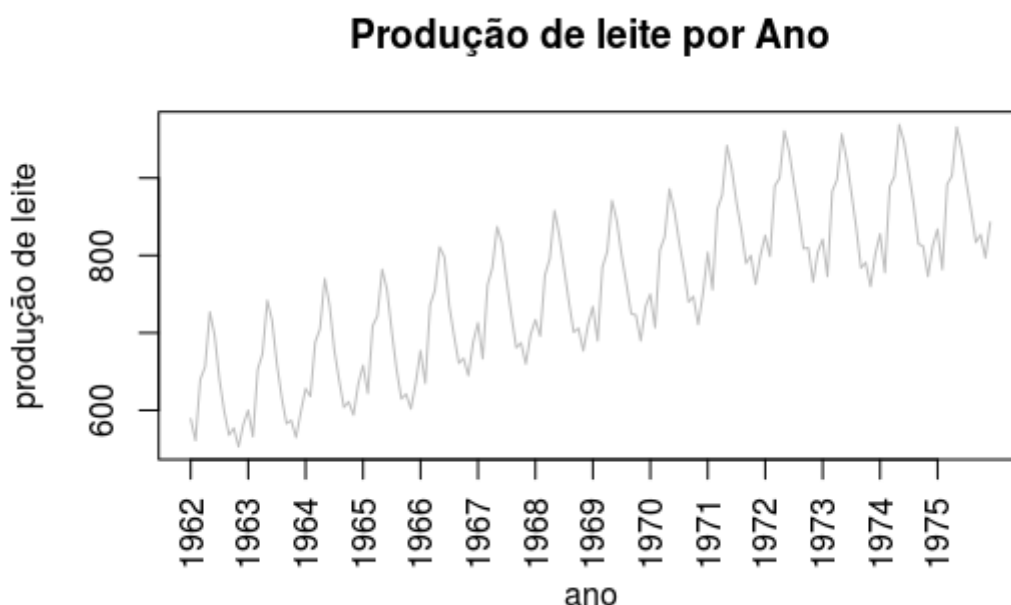


Figura 4: Série temporal da produção mensal de leite entre 1962 e 1975

2.2 Simulação Computacional

Para simulação e modelagem da rede neural aos dados foi utilizado o software R Core Team (2016) com a extensão do pacote forecast (HYNDMAN, 2016). O script desenvolvido realizou o trabalho de importação e transformação dos dados para o formato *time series*, geração e ordenação dos modelos segundo o R^2 e plotagem dos gráficos. O script pode ser consultado no apêndice deste trabalho.

Considerando o modelo genérico $NNA(p, P, k)_m$, a simulação foi realizada mediante considerando as seguintes limitações:

$p = 1, 2, 3, 4, 5, 6$; (p = parâmetro de defasagem da parte sistemática)

$P = 1, 2, 3, 4, 5, 6$; (P = parâmetro de defasagem da parte sazonal)

$k = 1, 2, \dots, 60$; (k = quantidade de neurônios na camada intermediária)

$m = 12$. (m = parâmetro de sazonalidade)

Ao todo foram gerados 2160 modelos e o tempo de execução do script foi de 72 minutos.

2.3 Resultados

O melhor modelo de rede obtido com após a simulação foi $NNA(6, 6, 57)_{12}$. Este modelo apresentou um $R^2 = 0.0521$, ou seja, o ajuste da rede aos dados é quase perfeita. Na Figura 9 é possível observar, na cor azul, o resultado para uma previsão da produção de leite mensal para os anos de 1976 e 1977.

Em contrapartida o pior modelo obtido foi o $NNA(1, 3, 1)_{12}$. Este modelo apresentou um $R^2 = 245.8$. A Figura 5 mostra o comparativo entre os valores preditos entre o pior e melhor modelo *versus* os valores da série original. Perceba que o melhor modelo (cor verde) utilizou um conjunto de dados maior na fase de aprendizagem quando comparado ao pior modelo. Essa maior quantidade de informações melhorou bastante o grau de predição do modelo $NNA(6, 6, 57)_{12}$.

Comparação entre modelos - Valores Preditos

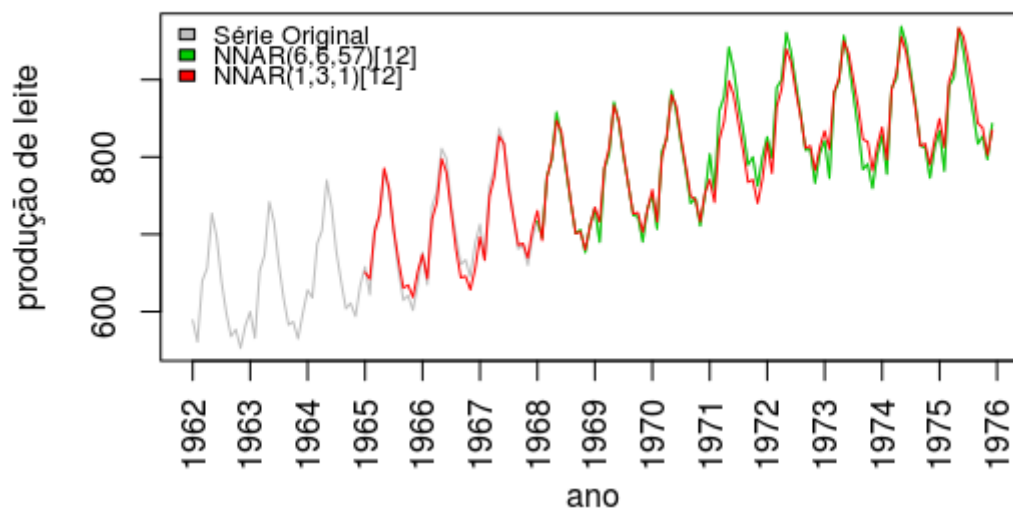


Figura 5: Comparação de valores preditos

A Figura 6 mostra um comparativo dos valores residuais entre o pior e o melhor modelo. Perceba que os resíduos do melhor modelo, $NNA(6, 6, 57)_{12}$, permanecem de forma contínua próximos ao valor 0. Enquanto isso, os valores residuais do pior modelo, $NNA(1, 3, 1)_{12}$, apresentam grande variação no tempo.

Comparação entre modelos - Resíduos

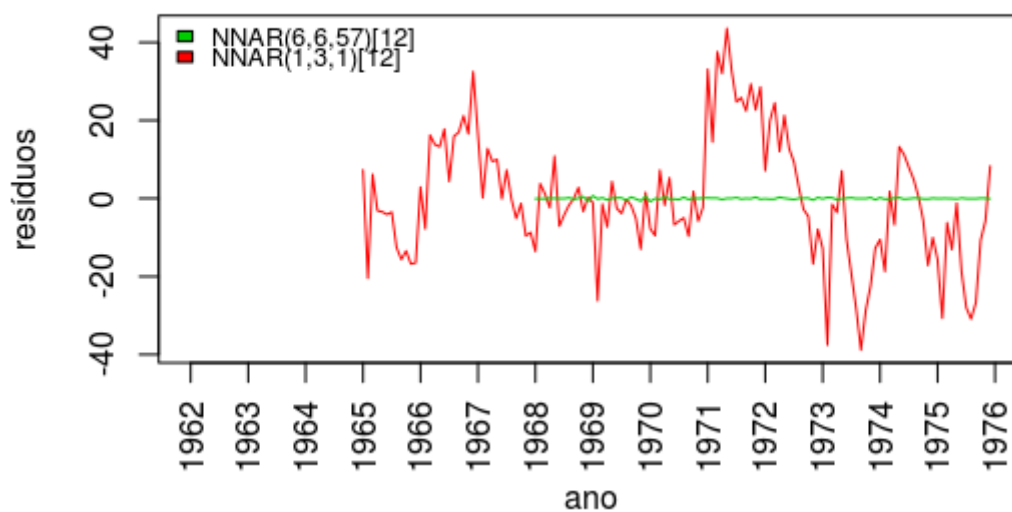
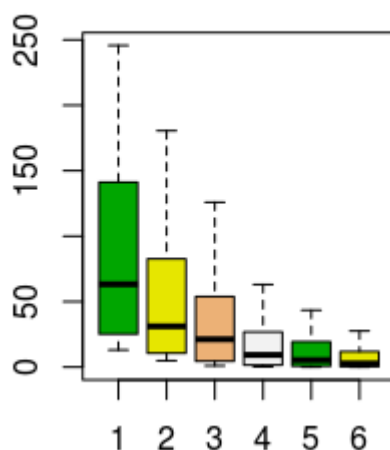


Figura 6: Comparação de resíduos

Em relação a evolução dos parâmetros p e P é notório que quanto maior seus respectivos valores, melhor é a qualidade do modelo. Perceba na Figura 7 que conforme aumenta os valores dos parâmetros p e P , o valor de R^2 diminui, ou seja, a qualidade de ajuste do modelo aos dados é melhorada.

R2 para o parâmetro p



R2 para o parâmetro P

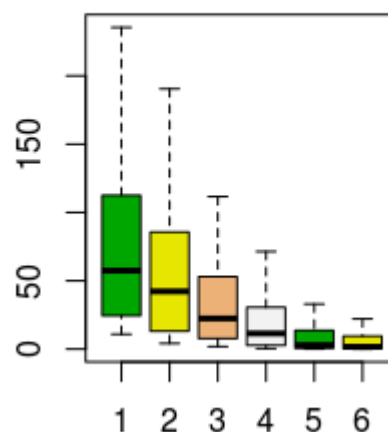


Figura 7: Evolução dos parâmetros p e P

Para o parâmetro k , que representa o número de neurônios da camada intermediária, nota-se que quanto maior o valor de k , menor é o valor de R^2 . Em resumo isso mostra que o aumento do número de neurônios melhora o ajuste do modelo aos dados, porém é perceptível que esta melhoria tende a se tornar estacionária para $k > 20$. Essa característica é importante, uma vez que o aumento do número de neurônios na camada intermediária influencia negativamente no desempenho computacional da rede neural.

R2 para o parâmetro k

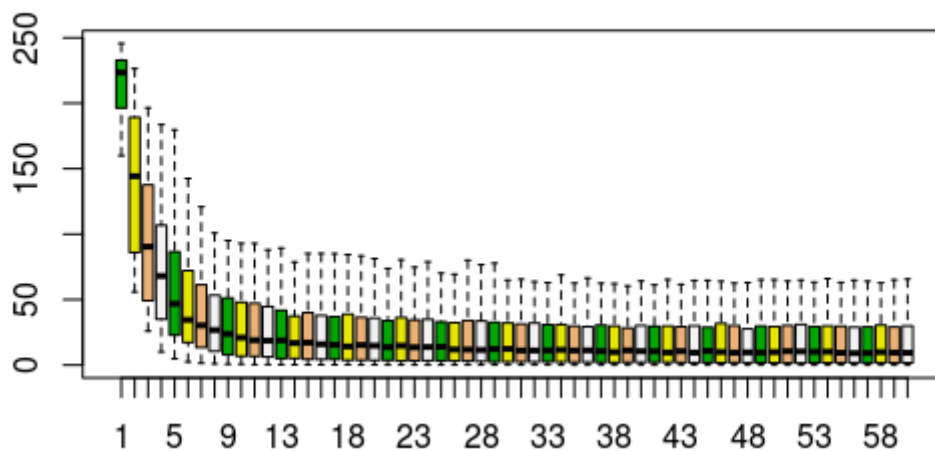


Figura 8: Evolução dos parâmetros k

Por fim, a Figura 9 mostra os valores previstos para os anos de 1976 e 1977, com a rede $NNA(6,6,57)_{12}$.

Melhor modelo - NNAR(6,6,57)[12]

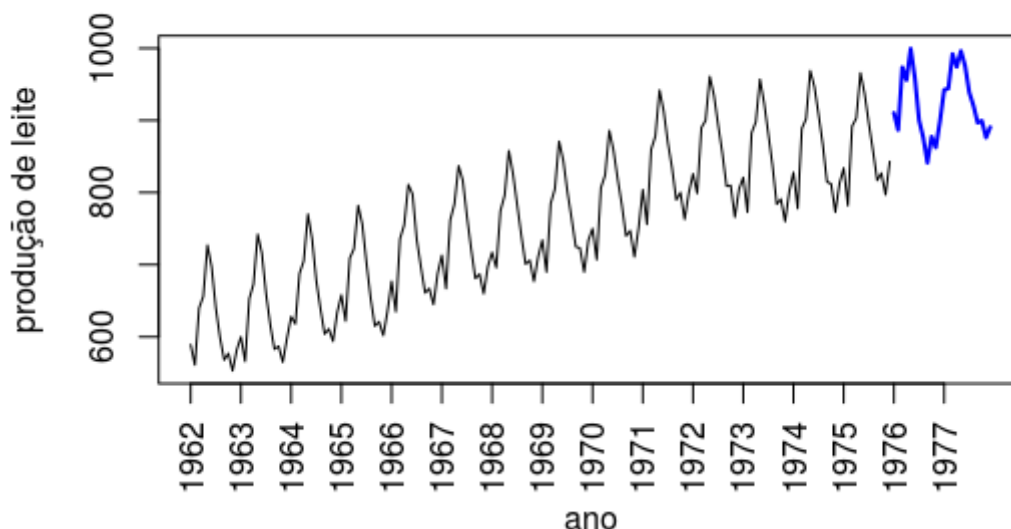


Figura 9: Produção de Leite: Previsão com o melhor modelo

3 Considerações Finais

O objetivo deste trabalho foi apresentar brevemente os conceitos sobre redes neurais, mais especificamente o tipo MLP e realizar a modelagem de uma série temporal para prever valores futuros.

Na simulação realizada foi utilizado o pacote *forecast* no software R. Este pacote permite ao analista realizar modelagens de séries temporais adotando modelos ARIMA com ou sem sazonalidade. Nos dados utilizados neste trabalho, (QLIKTEK, 2016), foi considerado que a série pode ser prevista por um modelo $ARIMA(1,0,1) \times (0,1,1)_{12}$, conforme apresentado em (ALMEIDA, 2016).

Com os resultados apresentados confirma-se que as redes neurais artificiais, mais especificamente as rede MLP com 1 camada intermediária se mostrou uma excelente alternativa para realizar previsões de

séries temporais. Além disso, foi detectado que o ajustamento da rede neural aos dados é notadamente melhorado quando se aumenta a quantidade de neurônios na camada intermediária.

Referências

- ALMEIDA, C. A. D. *Series Temporais no Domínio do Tempo: Aplicação da Metodologia de Box-Jenkins*. Lisboa, Portugal: Trabalho de Modelação Estatística II, Atividade 2, Universidade Aberta, 2016. page.88
- BRAGA, A. D. P.; CARVALHO, A. P. D. L. F. D.; LUDERMIR, T. B. *Redes Neurais Artificiais: Teoria e Aplicações*. 1. ed. Rio de Janeiro, Brasil: LTC, 2007. page.11, page.22, page.33
- CORRAR, L. J.; THEÓPHILO, C. R. *Pesquisa Operacional para Decisão*. 1. ed. São Paulo, Brasil: Atlas, 2004. page.22, page.44
- CORTEZ, P. et al. Real-time forecasting by bio-inspired models. *Artificial Intelligence and Applications : proceedings of the IASTED International Conference*, IASTED ACTA Press, Málaga, Spain, v. 2, p. 52–57, 2002. page.44
- HAYKIN, S. *Redes Neurais: Princípios e Prática*. 1. ed. Porto Alegre, Brasil: Bookman, 2001. page.22, page.33, page.44
- HYNDMAN, R. J. *forecast: Forecasting functions for time series and linear models*. [S.l.], 2016. R package version 7.1. Disponível em: <http://github.com/robjhyndman/forecast>. page.55
- HYNDMAN, R. J.; KHANDAKAR, Y. Automatic time series forecasting: the forecast package for r. *Journal of Statistical Software*, v. 27, n. 3, p. 1–22, 2009. Disponível em: <http://www.jstatsoft.org/article/view/v027i03>. page.44
- KHASHEI, M.; BIJARI, M. An artificial neural network (p, d, q) model for timeseries forecasting. *Bulletin of Mathematical Biophysics*, v. 5, p. 115–133, 1943. page.11
- LAWRENCE, R. Using neural networks to forecast stock market prices. In: . Manitoba, USA: Department of Computer Science, University of Manitoba, 1997. p. 1–21. page.55
- MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *Expert Systems with Applications*, v. 37, p. 479–489, 2010. page.22
- OKASHA, M. K.; YASEEN, A. A. Comparison between arima models and artificial neural networks in forecasting al-quds indices of palestine stock exchange market. In: . [S.l.: s.n.], 2013. page.44
- PURWANTO, D.; ESWARAN, C.; LOGESWARAN, R. A comparison of arima, neural network and linear regression models for the prediction of infant mortality rate. In: . Washington, USA: IEEE Computer Society, 2010. p. 34–39. page.33
- QLIKTEK. *DataMarket - Monthly milk production: pounds per cow. Jan 62 - Dec 75*. 2016. Disponível em: <https://datamarket.com/data/set/22ox/monthly-milk-production-pounds-per-cow-jan-62-dec-75\#!ds=22ox&display=line>. page.55, page.88
- R Core Team. *R: A Language and Environment for Statistical Computing*. Vienna, Austria, 2016. Disponível em: <https://www.R-project.org/>. page.55
- ZHAO, L. Neural networks in business time series forecasting: Benefits and problems. *Review of Business Information Systems*, v. 13, n. 3, p. 57–62, 2009. page.22

Apêndice

Script do R

```
#Load database
milk <- read.csv("/media/cleibson/Backup/HD_bruelas/doutorado/
UAB/7_modelacao_estatistica_2/atividade4/recursos/milk.csv")

#Create dataset in the type Time Series
milk_data = ts(milk$Production, start = c(1962,1), frequency = 12)

#Load Package Modeling Time Series
library(forecast)

begin_time = proc.time()
#BEGIN Generate simulation table of results
#ARIMA(p,0,0)(P,0,0)m = NNAR(p,P,k)m
p=6
P=6
k=60 #number neurons as MLP neural network model
results = as.data.frame(matrix(nrow = p*P*k, ncol = 6))
names(results) = c('Model', 'R2', 'p', 'P', 'k', 'm')
i = 1
p_little=1
while (p_little <= p){
  p_big=1
  while (p_big <= P) {
    neurons=1
    while (neurons <= k) {
      fit <- nnetar(milk_data, p=p_little, P=p_big, size=neurons)
      print(fit)
      results[i,1]=fit$method
      results[i,2]=mean(fit$residuals^2,na.rm = TRUE)
      results[i,3]=fit$p
      results[i,4]=fit$P
      results[i,5]=fit$size
      results[i,6]=fit$m
      neurons = neurons + 1
      i = i + 1
    }
    p_big = p_big + 1
  }
  p_little = p_little + 1
}
#END
end_time = proc.time() - begin_time
script_time = paste('Tempo de Execução=',round(end_time[3]/60,digits = 2),
'minutos', sep=' ')
script_time
model_total = i - 1
```

```
#Ordering Results
results = results[order(results$R2),]

#Choose best result
best_model = results[1,]
#Choose worse result
worse_model = results[p*P*k,]

#Create best and worse model
best_fit <- nnetar(milk_data, p=best_model$p, P=best_model$P,
size=best_model$k)
worse_fit <- nnetar(milk_data, p=worse_model$p, P=worse_model$P,
size=worse_model$k)

#plot forecast best model
plot(forecast(best_fit), xlab = 'ano', xaxt = 'n', ylab = 'produção de leite',
main=paste('Melhor modelo - ', best_fit$method, sep=''))
axis(1, at = seq(1962, 1977, by = 1), las=2)

#comparing parameters
par(mfrow = c(1,2))
boxplot(results$R2 ~ results$p, data=results, col=terrain.colors(4),
outline = F, main = 'R2 para o parâmetro p')
boxplot(results$R2 ~ results$P, data=results, col=terrain.colors(4),
outline = F, main = 'R2 para o parâmetro P')
par(mfrow = c(1,1))
boxplot(results$R2 ~ results$k, data=results, col=terrain.colors(4),
outline = F, main = 'R2 para o parâmetro k')

#generate graphics compare fitted values
best_values=best_fit$fitted
worse_values=worse_fit$fitted
plot(milk_data, xlab = 'ano', xaxt = 'n', ylab = 'produção de leite',
      main='Comparação entre modelos - Valores Preditos', col='gray', pch=10)
lines(best_values, col=3, pch=20)
lines(worse_values, col=2, pch=20)
axis(1, at = seq(1962, 1977, by = 1), las=2)
legend("topleft", c("Série Original",best_fit$method,worse_fit$method), cex=0.8,
fill=c('gray',3,2), bty="n")

#generate graphics compare error values
best_residuals=best_fit$residuals
worse_residuals=worse_fit$residuals
plot(worse_residuals, xlab = 'ano', xaxt = 'n', ylab = 'resíduos',
      main='Comparação entre modelos - Resíduos', col=2, pch=20)
lines(best_residuals, col=3, pch=20)
axis(1, at = seq(1962, 1977, by = 1), las=2)
legend("topleft", c(best_fit$method,worse_fit$method), cex=0.8, fill=c(3,2), bty="n")
```