Universidade Federal de Pernambuco
Centro de Informática

Graduação em Ciencia da Computação

Previsão de séries temporais utilizando uma combinação não-linear entre ARIMA e Redes Neurais Artificiais

Pedro Henriques de Faria Neves

Trabalho de Graduação

Recife
Julho de 2017

Universidade Federal de Pernambuco
Centro de Informática

Pedro Henriques de Faria Neves

Previsão de séries temporais utilizando uma combinação não-linear entre ARIMA e Redes Neurais Artificiais

Trabalho apresentado ao Programa de Graduação em Ciencia da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: *Paulo Salgado Gomes de Mattos Neto*

Recife
Julho de 2017

Dedico este trabalho à Deus, pela força, perseverança e por ter me permitido esta oportunidade.
Dedico à minha família, em especial minha mãe, Sílvia, que não mede esforços para me apoiar nos momentos difíceis e está sempre presente.
Dedico também à minha sempre companheira Myra, pelo seu carinho, compreensão e paciência, e aos amigos Décio, Eduardo, Gabriel e Lucas, que com humor ácido, sempre me apoiaram.
Sou eternamente grato à todos.

Agradecimentos

Gostaria de agradecer à minha mãe, Silvia, por por todo o suporte e por nunca desistir de seus filhos, não importem as adversidades.

Aos meu colegas de trabalho Arthur Padilha, David Benko, Telmo Filho, Andréa de Castro e Érika Andrade e toda equipe da Artics pelo suporte inestimável que recebi.

Ao meu orientador, Professor Dr. Paulo Salgado Gomes de Mattos Neto, pela orientação e auxilio durante a elaboração deste trabalho.

À Universidade Federal de Pernambuco e ao Centro de Informática, por oferecer o suporte necessário para minha formação.

Who dares wins
—DAVID STIRLING

Resumo

Análise de séries temporais é uma atividade fundamental para muitos processos reais. Esse trabalho de graduação tem como objetivo analisar uma abordagem híbrida, com uso de modelos ARIMA e Redes Neurais Artificiais, que visa tornar o processo de predição de séries temporais mais preciso e eficaz.

Fenômenos reais, os quais são medidos e acompanhando por de tais séries, em geral possuem a característica de serem difíceis de se modelar. Isto se dá em parte, por não conhecer totalmente o processo que regem estes fenômenos (se são de natureza linear ou não linear), em parte pela complexidade das variáveis envolvidas no processo.

Métodos tradicionais, de natureza estocástica, foram propostos com considerável sucesso, como por exemplo, *Autoregressive Integrated Moving Average* (ARIMA) e a metodologia Box-Jenkins. Estes modelos conseguem modelar e predizer valores futuros baseados em um conjunto de medições anteriores com certa precisão.

Com o novo ganho de força das Redes Neurais Artificiais (RNA), as atenções novamente se voltaram para elas e estudos vêm mostrando a sua eficácia no reconhecimento e classificação de padrões, especialmente para processos não lineares. Este trabalho visa analisar um modelo híbrido de previsão de séries temporais através da combinação não-linear de modelos ARIMA e de RNA.

Palavras-chave: Redes Neurais Artificiais, ARIMA, Análise de Séries Temporais, Modelo Híbrido, RBF, Redes de Função de Base Radial, Multilayer Perceptron, MLP.

Abstract

Time series analysis is a major activity in many real-world processes. This graduation thesis has the objective to analyse a proposed hybrid approach, which will make use ARIMA models together with Artificial Neural Networks, and combine them in a nonlinear manner, in order to get more precise forecasts.

Real world phenomena, which are measured through such time series, has the feature of being hard to model. This is caused in part by the fact of not knowing the underlying process, which is responsable to produce the observations of the time series, and part by the complexity of the process' variables.

Traditional methods, of stochastic nature, were proposed with considerable success, like for instance, the Autoregressive Integrated Moving Average together with the Box-Jenkings methodology for model specification. These models are able to model and forecast future values based in a set of previous ovservations.

With the advances in Artificial Neural Networks field, the attention has turned back to them and studies shows it's efficiency in pattern recognition, as well in classification - very nonlinear activities. This work has the objective to analyse the use of nonlinear combination for linear models (ARIMA) and nonlinear models (ANN) in order to achieve better predictions.

Keywords: Artificial Neural Networks, ARIMA, Time Series Analysis, Hybrid, Hybrd Approach, Hybrd Model, RBF, Radial Basis Function, Multilayer Perceptron, MLP.

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

1.1 Séries temporais

Séries temporais são um conjunto de dados obtidos a partir da análise de um fenômeno, ou propriedades de um fenômeno no decorrer do tempo, com amostragens realizadas em intervalos equidistantes. Dessa forma, uma das características das séries temporais é que observações adjacentes são dependentes [1].

A importância desse tipo de série é evidente em diversos campos da ciência. Por exemplo, economistas podem estar interessados nas flutuações do câmbio de conversão diário entre duas moedas. Para epidemiologistas, a verificação do número de ocorrências de uma determinada doença, ano a ano, pode ser fundamental. As áreas que se valem de um intenso uso de séries temporais são a física e as ciências ambientais. A série de *Sunspot* é uma das séries mais antigas já registradas, datada do ano de 1906 registrada por Sir Franz Arthur Schuster, e representa a quantidade de manchas solares apresentadas pela atmosfera solar. Tais fenômenos interferem no clima espacial e estes por sua vez, atrapalham a comunicação com satélites [2].

Dos exemplos dados, podemos perceber que ao estudarmos séries temporais estamos interessados em descrever a série, e portanto, o fenômeno, e em prever valores futuros. Em muitos desses fenômenos, os mecanismos que os provocam são complexos demais, o que torna sua modelagem bastante difícil. Em vários casos, prever eventos futuros é uma característica crítica [1].

Tradicionalmente, estatísticos utilizam uma gama de modelos estocásticos que têm como objetivo modelar tais séries. O mais famoso destes, é o método *Autoregressive Integrated Moving Average* (ARIMA) proposto por Box e Jenkins na década de 1970. Usualmente, assume-se que o processo que gera a série em estudo é produzida por um processo linear pois, como dito anteriormente, não se conhece *a priori* tal mecanismo, e para estes, a classe de modelos ARIMA apresenta desempenho satisfatório [1].

Entretanto, fazer tal suposição - de que o processo inerente à série é linear - pode incorrer em um uso totalmente inapropriado dos modelos. Em geral, em casos reais as séries temporais são geradas por sistemas não-lineares. Mais recentemente, pesquisas realizadas com Redes Neurais Artificiais (RNAs) vêm tendo bons resultados com previsão de séries temporais. Isso é uma consequência, basicamente, da característica de serem métodos auto adaptativos *data-driven*, ou seja, RNAs não requerem um conhecimento do processo que gera a série *a priori*. As RNAs aprendem a partir de exemplos e são capazes de capturar relacionamentos sutis. Além disso, RNAs são capazes de generalizar, ou seja, após aprenderem, conseguem extrapolar o conjunto de dados fornecido para treino e, a partir deles, inferir resultados futuros. Finalmente, Redes Neurais são excelentes para estimar funções. Qualquer modelo estatístico assume que

exista uma relação entre os valores futuros e os passados, e RNAs são bons modelos para encontrar esses relacionamentos [3] [4].

1.2 Motivação

Os modelos tradicionais, como os do método ARIMA, juntamente com as RNAs possuem bons resultados, se aplicados isoladamente. Porém, estudos mostram que utilizar mais de um modelo, formando sistemas híbridos, em geral provêm melhores resultados [4]. De fato, em casos reais, dificilmente um problema é puramente linear ou não linear, o que torna o uso dos modelos clássicos inapropriados pois são incapazes de modelar os relacionamentos não lineares. Embora novos modelos tenham sido propostos para lidar com esse problema, tais como o modelo bilinear, o modelo *Threshold Autoregressive* (TAR) e o modelo *Autoregressive Conditional Heteroscedastic* (ARCH), o ganho não é significativo [3]. Além disso, esses modelos foram desenvolvidos para tratar padrões específicos de não linearidade, perdendo efetividade em outros tipos de séries. Por outro lado, Redes Neurais são ineficazes se a natureza do problema for puramente linear [3].

O modelo proposto por [5] consiste em duas etapas básicas. Na primeira, parte modela-se a série usando ARIMA. Nesse ponto, qualquer estrutura não-linear que a série contenha, permanecerá nos resíduos. A partir daí, na segunda etapa, modela-se esses resíduos usando RNAs. Os resultados da Rede Neural associados à previsão do ARIMA são então usados para prever a série. Dessa forma, o modelo híbrido explora as características únicas tanto do modelo ARIMA, quanto da RNA em diferenciar padrões, e combinando os resultados para ter um resultado geral melhor.

1.3 Objetivos

No modelo proposto por [5], a combinação dos resultados é feita de maneira simples. Este trabalho de graduação têm como objetivo explorar como a combinação desses resultados, utilizando métodos não-lineares, podem influenciar na qualidade geral da previsão do modelo.

Para isto, serão consideradas três séries temporais: *Sunspot*, que representa a quantidade de manchas solares, ano a ano; *Canadian Lynx*, que registra a quantidade de linces capturados no distrito do Rio Mackenzie, Canadá; e por fim, a *BPUSD*, que representa a taxa de conversão da libra britânica (*British Pounds*) para dólar americano (*United States Dollars*). Para esse conjunto, serão apresentados os resultados obtidos pela combinação simples (soma), como descrito em [5], e também os resultados obtidos pela combinação usando-se Redes Neurais MLP e RBFNN.

CAPÍTULO 2

Embasamento Teórico

2.1 Análise de séries temporais

Uma série temporal é o conjunto de observações ou medições realizadas ao longo do tempo, de alguma variável de interesse, para um determinado fenômeno. Índices diários da bolsa de valores, precipitação mensal em uma determinada área do país, total mensal de automóveis vendidos são apenas alguns exemplos, de um vasto conjunto de séries temporais. É importante notar que essas observações se dão em intervalos igualmente espaçados: anualmente, mensalmente, a cada hora, e assim sucessivamente. Dessa forma, séries temporais são intrinsecamente discretas. Para o caso em que a propriedade em análise seja contínua, a discretização dessa propriedade pode ser feita através de amostragem, em intervalos equidistantes, como é o caso por exemplo da análise de viscosidade de fluídos no decorrer do tempo. Outra característica importante é ordem dos dados. Em muitos casos de regressão, a análise pode ser realizada com os dados em uma ordem qualquer, porém não é o caso das séries temporais, uma vez que uma determinada observação y_t depende de seus resultados anteriores [1], [2], [6–8]. Desse modo, podemos interpretar uma série temporal como descrito em (2.1), em que um determinado valor x_t é definido a partir de seus k antecessores, tal que:

$$x_t = f(x_{t-1}, x_{t-2}, \ldots, x_{t-k}). \tag{2.1}$$

Idealmente, gostaríamos de descobrir a função f em (2.1) de modo que estaríamos assim definindo o processo que forma a série temporal em análise. Evidentemente, para processos reais, são inúmeras as varíaveis que afetam o fenômeno estudado e tudo que podemos fazer é gerar uma aproximação. Portanto para a mesma série, procuramos uma função h tal que:

$$x_t = h(x_{t-1}, x_{t-2}, \ldots, x_{t-k}) + e \tag{2.2}$$

em que e representa o resíduo (diferença entre o x_t e $h(\cdot)$, ou seja, o *erro*). Uma das atividades quando se estuda séries temporais é a de encontrar a função h de modo a minimizar o erro e.

É evidente a importância das séries temporais para os diversas áreas de estudos. Em muito desses campos, é importante não somente compreender os dados, mas também prever resultados futuros. Em alguns casos, essa pode ser uma característica crítica. Para esse fim, recorremos aos modelos. Através deles somos capazes de analisar características do fenômeno em questão, além de nos prover ferramentas para extrapolarmos os dados atuais, e prevermos eventos futuros.

Previsões podem ser classificadas três categorias, com base no seu alcance temporal: curto prazo, médio prazo e longo prazo [6]. Nas últimas décadas, diversos modelos foram desen-

volvidos com o propósito de nos prover de ferramentas capazes de realizarem previsões mais precisas. Dentre os modelos clássicos, o que mais se destaca é o ARIMA. Sua popularidade se deve às suas propriedades estatísticas, bem como ao uso da metodologia Box-Jenkins, usada em sua construção [5]. Não obstante, com o novo ganho de fôlego introduzido pelo algoritmo de *backpropagation*, Redes Neurais Artificiais voltaram a ser amplamente estudadas, e área de análise de séries temporais não foi exceção.

Nesta seção serão explorados conceitos e fundamentos que serão utilizados na seção seguinte para realização dos experimentos. Primeiramente será apresentado o modelo ARIMA e seus conceitos. Em seguida uma visão geral sobre Redes Neurais Artificiais bem como o modelo específico de RBFNN. Por fim, será apresentado o modelo híbrido.

2.2 Modelo ARIMA

A princípio, acreditava-se que determinada séries temporais eram compostas por uma componente determinística, que governa o comportamento geral da série, e por uma componente aleatória, que provê seu comportamento estocástico. Assumia-se que essa componente aleatória era formada independentemente do processo. Porém, observou-se que isso não é verdade, e essa correlação precisa ser modelada [6]. A constatação de que algum relacionamento ainda permanece nos resíduos (a diferença entre o valor previsto e o esperado) levou ao desenvolvimento do modelo *Autoregressive Integrated Moving Average* (ARIMA) por Box e Jenkins, juntamente com toda uma metodologia para a correta especificação do modelo [2].

Segundo [2], o modelo ARIMA é uma generalização composta por três partes. A primeira, os modelos Autoregressivos (*Autoregressive* - AR) se baseiam na idéia de que um determinado valor em uma dada série temporal, x_t pode ser definido em função de seus p valores passados, de modo que, o modelo de autorregressivo de ordem p, AR(p) é definido por:

$$x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + \cdots + \phi_p x_{t-p} + w_t \tag{2.3}$$

em que x_t é estacionário, $\phi_1, \phi_2, \ldots, \phi_p$ são constantes tais que, $\phi_p \neq 0$ é w_t é ruído branco [2]. Por ruído branco, entende-se um sinal com comportamento imprevisível, de modo que duas amostras não possuem correlação entre si e são uniformemente distribuídas com média zero [8], [9]. De (2.3), podemos observar que espera-se que x_t seja obtido linearmente a partir de seus valores passados.

A segunda parte do modelo ARIMA, vem do modelo de médias móveis (*Moving Average* - MA) que é definio por:

$$x_t = \mu + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \cdots - \theta_q \varepsilon_{t-q} \tag{2.4}$$

em que $\{\varepsilon_t\}$ é uma série de ruído branco e $\theta_1, \theta_2, \cdots, \theta_q$ são os parâmetros do modelo tais que, $\theta_q \neq 0$. Em (2.4), a série $\{x_t\}$ é considerada estacionária e é chamado um processo de médias móveis de ordem q (MA(q)) [2]. Os modelos AR e MA, se usados em conjunto, formam um modelo mais geral, *Autoregressive Moving Average*, abreviado por ARMA(p, q) e é definido por:

$$x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + \cdots + \phi_p x_{t-p} + w_t + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \cdots - \theta_q \varepsilon_{t-q} \tag{2.5}$$

em que p e q são ditas as ordens autorrgressiva e de médias móveis, respectivamente, $\phi_p \neq 0$, $\theta_q \neq 0$ e $\{\varepsilon_t\}$ é uma série de ruído branco [2]. Em (2.5), assume-se que o processo x_t é estacionario [2].

A suposição de que os processos inerentes às séries temporais são estacionários, em geral, é grosseira. Séries temporais reais, em via de regra possuem um caráter não estacionário, portanto, nesses casos, o modelo ARMA se torna pouco eficiente. Em geral, é necessário transformar a série primeiro, tornando-a estacionária. Para esse propósito, tem-se a terceira e última parte parte do modelo ARIMA, que propõe sucessivas diferenciações com o objetivo de torná-la estacionária [6]. A diferenciação é uma operação tal que:

$$\nabla x_t = x_t - x_{t-1}. \tag{2.6}$$

Com isso, define-se que uma série $\{x_t\}$ é dita ARIMA(p, d, q) se, após d diferenciações, a série resultante for um processo ARMA(p, q), em que p é a ordem *autoregressive* e q é a ordem de *moving average* [6]. Dessa forma pode-se lidar tanto com processos estacionários, quanto não estacionários.

2.3 Redes Neurais Artificiais

O cérebro humano possui uma imensa capacidade de reconhecer e processar padrões. Vivenciamos isso a cada instante quando nos comunicamos, nos orientamos, escutamos música, percebemos as estações do ano, lemos este texto, e assim diariamente. Estas tarefas, de caráter altamente paralelo, são executadas por nós de maneira muito fácil e trivial, de modo que nos questionamos por que os computadores modernos ainda não são capazes de realizar tais operações. Após muito estudar, verificamos que programar computadores para realizar tais tarefas não é um processo tão simples. Muito se deve à natureza sequencial do design dos computadores. Apesar de serem excelentes em processar operações aritméticas, são ineficientes para compreender imagens, por exemplo. Com isto em mente, estudiosos do campo de inteligência artificial buscam mimetizar os princípios fundamentais do modo como cérebro humano soluciona o problema de reconhecimento [10], [11].

Pouco se sabe sobre como o cérebro representa a informação, mas sabemos que funciona com um conglomerado de unidades menores, os neurônios. A Figura 2.1 exibe um esquema do neurônio biológico. Nela, podemos visualizar os principais componentes do neurônio, que são: Axônio (*axon*) responsável pela condução do sinal elétrico; Os dendritos (*dendrite*), nas quais os axônios se conectam e recebem o sinal transmitido por eles; A sinapse (*synapse*), que é a região entre o axônio e o dendrito por ondem transitam os neurotransmissores; e por fim, o corpo celular (*soma*), que contém o núcleo e o citoplasma da célula.

Estas unidades são extremamente interconectadas, e são responsáveis por realizar atividades mais simples. Apesar dos componentes físicos dos computadores possuírem um arranjo que favorece o processamento sequencial, as Redes Neurais Artificiais foram desenvolvidas

para modelar a estrutura do cérebro biológico, de modo que permita a máquina realizar tarefas e solucionar problemas que atualmente só humanos são capazes de atacar [10]. Entretanto, algumas características do sistema neural biológico foram deliberadamente ignoradas de modo a tornar a implementação compreensível, bem como viável nos computadores digitais atuais. Capturar as características essenciais em detrimento das demais é um princípio básico de modelagem [10], [12].

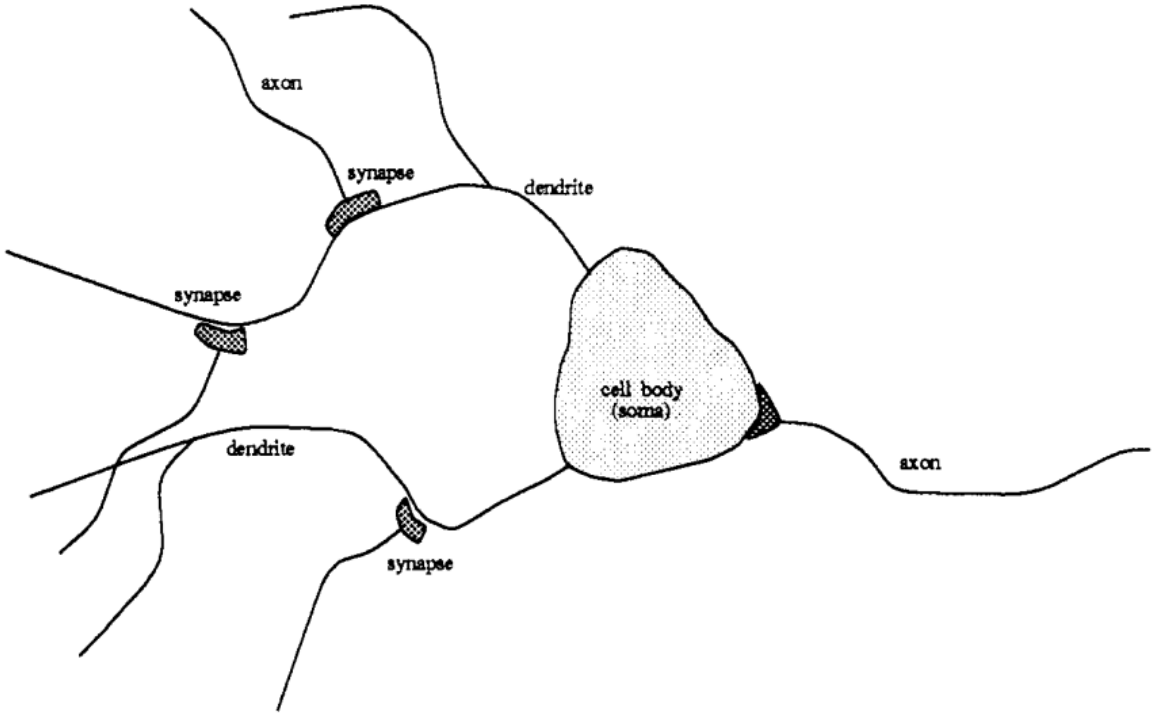

Figura 2.1 Características básicas do neurônio biológico. Fonte: [10].

2.3.1 Aprendizagem

O aprendizado é uma das características mais importantes do cérebro. Ele é capaz de ensinar a si próprio baseando-se, na maior parte das vezes, em exemplos [10]. Biologicamente, acredita-se que o aprendizado ocorre quando há alterações nas conexões sinápticas entre os neurônios [10], [12]. Essas alterações aumentam o acoplamento entre os dois neurônios, reforçando a conexão entre eles. Essa característica do sistema biológico é extremamente importante, e é trazida para o modelo computacional sob a interpretação de pesos das conexões neurais. Portanto, alterações nestes pesos representam, no modelo de Rede Neural, o processo de aprendizagem [10], [12].

Um dos problemas mais comuns em muitas áreas é o de aproximar uma função a partir de um conjunto de entrada e saída. Em geral, é mais fácil obter dados de entrada e de saída, do que a função propriamente dita. No estudo de redes neurais, o paradigma de aprendizado

supervisionado refere-se ao método em que a função é aproximada (ou aprendida) através dos exemplos (ou seja, entrada e saída esperada para tal entrada) fornecidos previamente. Nesse caso, os exemplos formam o conjunto de treinamento [12].

A Figura 2.2 apresenta o esquema de aprendizagem supervisionada. No processo de aprendizagem supervisionada, os parâmetros são ajustados iterativamente até que a RNA consiga emular o professor. Neste contexto, o professor é representado pelo conjunto de treinamento, com dados de entrada e de saída. Uma vez que o conhecimento foi transferido do professor para a RNA, a Rede Neural pode lidar com o ambiente por si só [12].

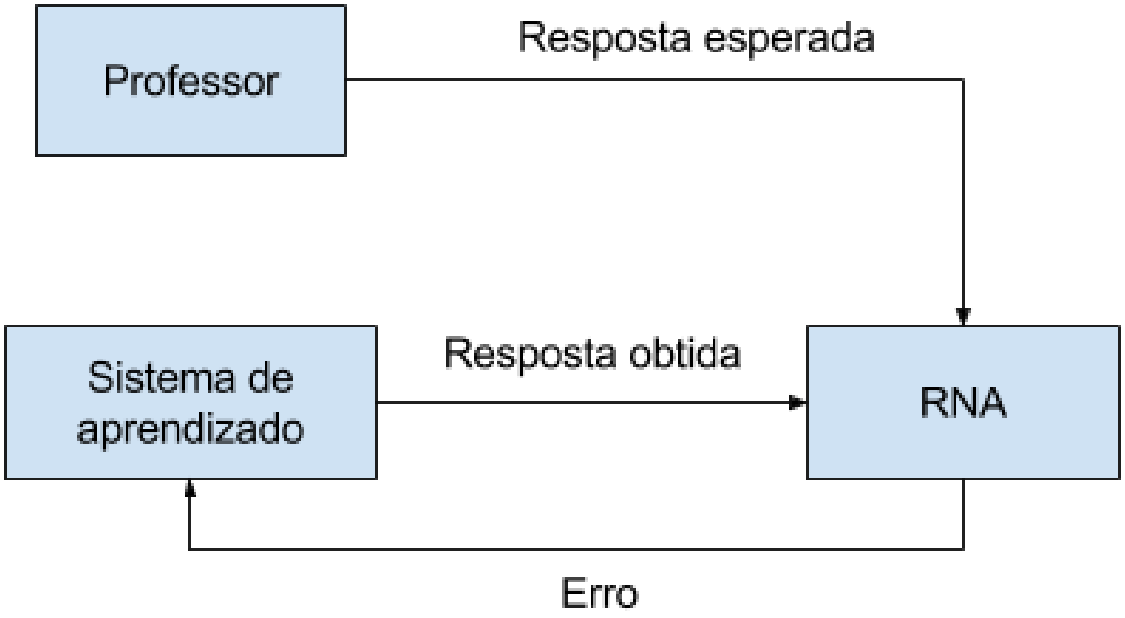

Figura 2.2 Diagrama representativo do paradigma de aprendizagem supervisionada.

Em contrapartida, o processo de aprendizagem não supervisionada ocorre sem a presença de um professor. Ou seja, a Rede Neural deve ser capaz de identificar as características da entrada e classificá-las com pouco ou nenhum auxílio externo. No primeiro caso, pode-se usar um crítico, ou seja, um mecanismo que indica, sem a necessidade de exatidão, se a rede está evoluindo na direção correta ou não. No segundo, nenhuma informação extra é fornecida [12].

No contexto deste trabalho, as RNA's consideradas utilizam o processo de aprendizado supervisionado, uma vez que uma determinada observação depende das observações passadas. Dessa forma, dada uma série temporal $\{x_t\}$, sempre podemos construir um conjunto de treinamento tal que, é composto por vetores de dimensão K em que os valores $x_{t-1}, x_{t-2}, \ldots, x_{t-K}$ são usados para definir o valor x_t, para algum valor de t.

2.3.2 Perceptrons

A partir da Figura 2.1, identifica-se que o modelo de neurônio artificial é composto por um conjunto de *sinapses* caracterizadas como pesos, um *somador* para somar os sinais de entrada multiplicados pelos pesos das sinapses, uma *função de ativação* que determinará se o neurônio irá emitir o sinal de saída. A Figura 2.3 mostra o esquema da modelagem de um neurônio artificial.

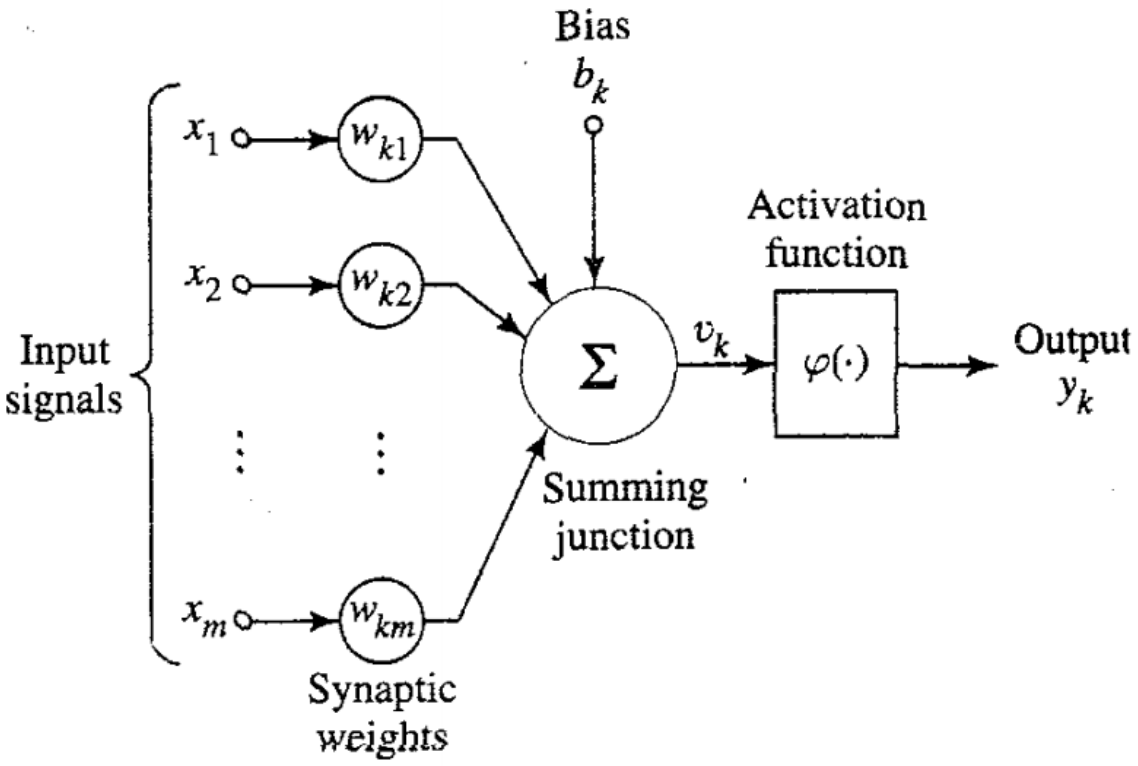

Figura 2.3 Modelagem básica do neurônio. Fonte: [12].

Podemos observar que contém o conjunto de receptores de sinais (*Input signals*) $x_1, x_2, \ldots, x_m$ que serão responsáveis por sensoriar o ambiente (domínio do problema) ou receber os sinais de saída de outros neurônios, para o caso de uma Rede Neural Artificial. Estes receptores de sinais são ampliados (ou reduzidos) através dos pesos (*Synaptic weights*) $w_{k1}, w_{k2}, \ldots, w_{km}$, que indicam o quão forte é a conexão sináptica x_i. Estes valores de entrada, multiplicados pelos seus pesos são então somados. Em seguida, a função de ativação (*Activation function*) $\varphi(\cdot)$ exibe o sinal de saída. No somatório pode-se notar a presença da componente b_k, que representa o *bias*. Sua função é realizar uma tranformação afim no somatório de modo que, quando quando a função de ativação for maior que zero, o neurônio produz uma saída. Caso contrário permanece desativado [10], [12].

A idéia para este tipo de modelo de neurônio foi introduzido por McCulloch e Pitts, em 1943 [10], [12]. Ele captura as características princiais de um neurônio biológico em detrimento

de outros elementos que tornam complexo o sistema nervoso real [10]. Ainda assim, representa apenas um modelo base. Sabe-se, por exemplo, que *feedback* é uma característica comum nos sistemas neurais biológicos dos animais, mas o modelo da Figura 2.3 não as apresenta. Por *feedback* entende-se que um dado neurônio utiliza sua própira saída como sua entrada [12].

2.3.3 Redes Multilayer Perceptron

Existem diversos tipos de arquiteturas de redes neurais, porém dentre elas, uma que se destaca é a *Multilayer (feedforward) Perceptron* (MLPs) [12]. Este tipo de rede comumente é composta por uma camada de entrada (sensoriamento), uma ou mais camadas escondidas (*hidden layers*) responsáveis pela computação, e uma camada de saída. A Figura 2.4 demonstra uma arquitetura básica de uma MLP. Nela, observa-se a camada de entrada (*Input signal* e *Input layer*), duas camadas escondidas (*First* e *Second input layer*) e a camada de saída, com três nós. Observa-se também a característica *feedforward* da rede, onde os dados são processados camada-à-camada, unidirecionalmente, da camada de entrada para a camada de saída [12].

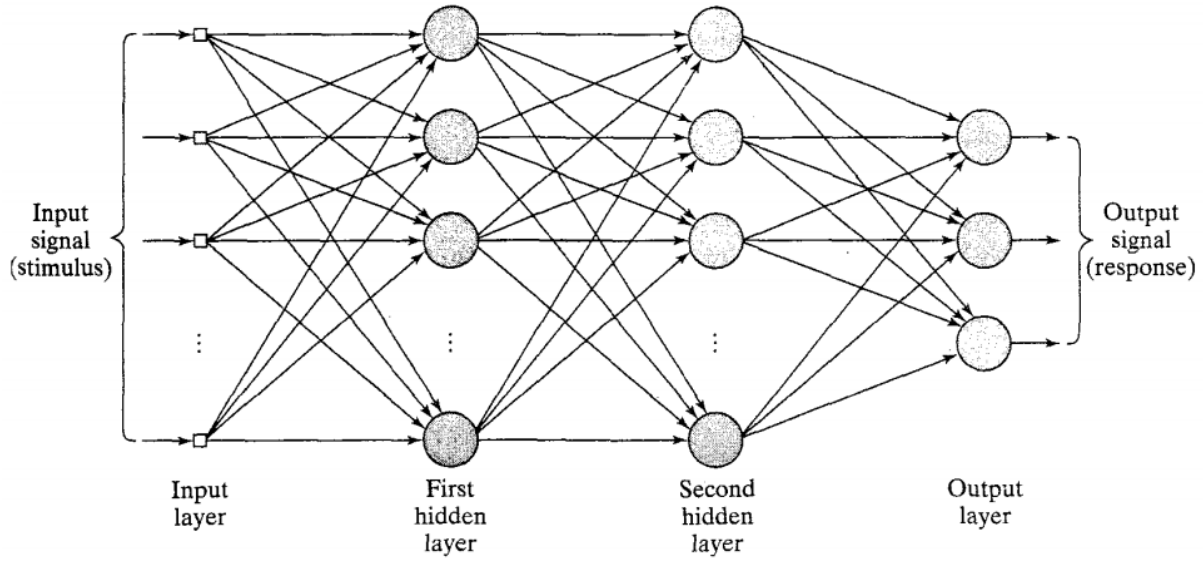

Figura 2.4 Arquitetura básica de uma rede MLP. Fonte: [12].

Os fatores mais importantes das redes MLPs, que permitem o seu poder computacional, são eles: O algoritmo *back-propagation*, a presença funções não lineares como funções de ativações dos neurônios da camada interna, o conjunto de camadas internas e seu auto grau de conectividade [12].

O *back-propagation* é um algoritmo desenvolvido no trabalho de Rumelhart, Hinton e Williams em 1986 [10] e foi de grande influência para o aprendizado de redes MLP, pois a correção de erros permite que as redes MLP possuam diversas camadas, e que se aproxime dos resultados esperandos durante a etapada de aprendizagem supervisionada (treino) [10]. De modo superficial, o algoritmo possui duas etapas: A etapa *forward* em que um vetor de entrada é apli-

cado à camada de sensoriamento e seus resultados propagados até a camada de saída com os pesos fixos durante o processo, e a etapa *backwards*, em que são feitos os ajustes dos pesos [12].

Como demonstrado por [12], redes MLP treinadas com através do algoritmo *back-propagation* podem ser tratadas como aproximadores universais de funções, segundo o teorma da aproximação universal para um dado mapeamento não linear entre a entrada e saída. Maiores informações sobre redes *multilayer perceptrons*, podem ser encontradas em [12]

2.3.4 Redes de Função de Base Radial

Funções de base radial (*radial basis function*, ou RBF) são um conjunto de funções, tipicamente não-lineares, que se baseiam em alguma forma de distância para um dado centro, $h(||x-y||)$, em que $h : \Re^n \rightarrow \Re$, y é denominado centro e $||...||$ representa algum tipo de distância, normalmente a Euclidiana. Uma característica fundamental das RBF é que seu valor é proporcional à distância do vetor x de entrada para o centro y. A Tabela 2.1 exibe as RBF que mais se destacam. Em todos os casos, $r = ||x-y||$ e ε é chamada de *largura* da RBF, e representa o quão rápido varia o valor de h a medida que x se afasta do centro y.

Tabela 2.1 Lista das principais funções de base radial.

Função	$h(r)$
Gaussiana	$e^{-(\varepsilon r)^2}$
Multiquadrática	$\sqrt{1+(\varepsilon r)^2}$
Quadrática inversa	$\frac{1}{1+(\varepsilon r)^2}$
Multiquadrática inversa	$\frac{1}{\sqrt{1+(\varepsilon r)^2}}$

Funções de base radial podem ser usadas como funções de ativações em Redes Neurais Artificiais. Estas redes, chamadas de *Radial Basis Function Neural Network* (RBFNN) compõem um grupo redes tipo *feedforward* que usam RBFs como função de ativação de seus neurônios da camada escondida. Em sua forma mais básica, possui três camadas: camada de entrada, uma camada escondida e uma de saída. Segundo [12], as RBFNN atacam o problema de reconhecimento de padrões segundo um problema de ajuste de curvas no sentido de que o objetivo está em encontrar uma superfície multidimensional que melhor se adequa ao conjunto de dados de treinamento. Por sua vez, a generalização vem da interpolação dos dados de teste, nestes superelipsóides [12].

A Figura 2.5 exemplifica uma Rede Neural de função de base radial clássica. A camada de entrada, contendo m nós (dimensão m) é responsável pelo sensoriamento do ambiente. Os valores dessa camada são então mapeados para um espaço de dimensão n, através dos n nós da

camada escondida, onde estão as funções de base radial. Cada um destes nós possui um centro y, a ser determinado durante a fase de treinamento da rede. Em seguida, os dados são somados nos k nós de saída. No caso deste trabalho, trata-se de uma regressão e portanto, apenas um nó na camada de saída. Dessa forma, a rede da Figura 2.5 aproxima uma função $f : \Re^m \rightarrow \Re$ [12].

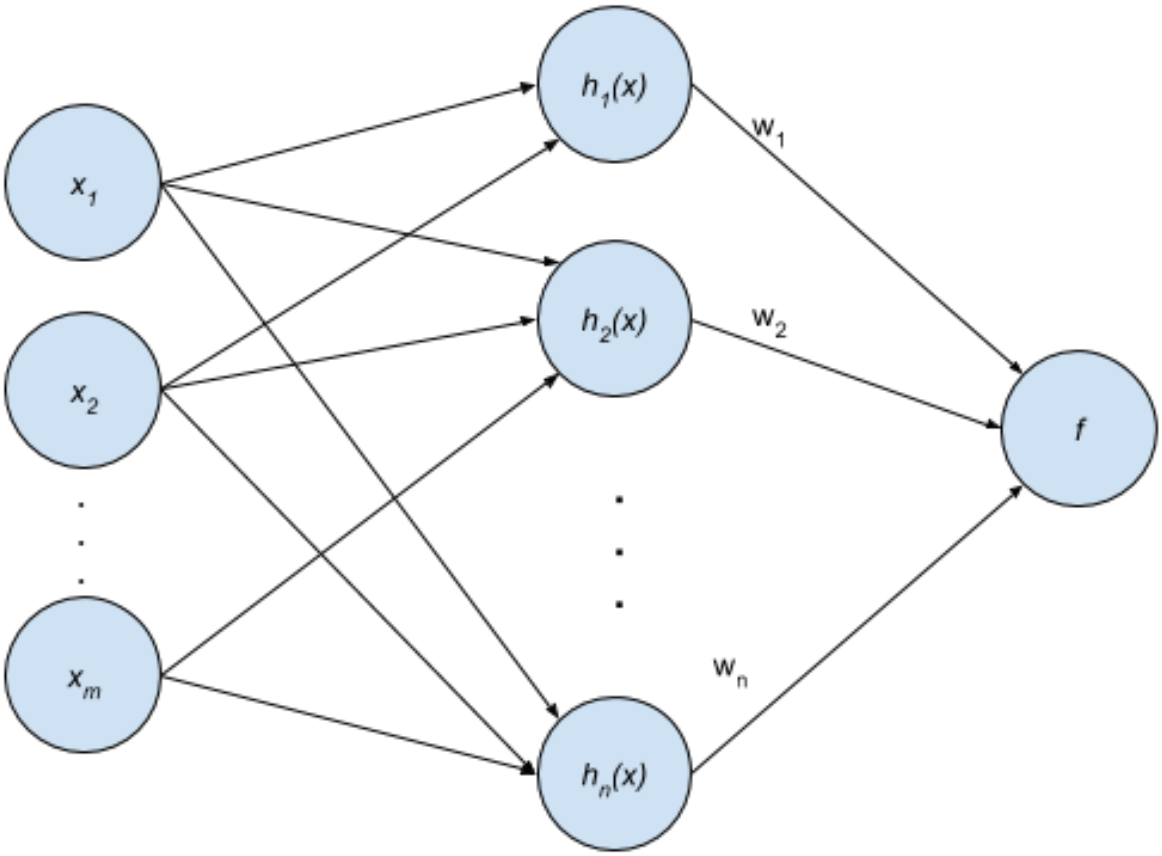

Figura 2.5 Representação esquemática de uma Rede Neural RBF com três camadas.

Para o caso de análise de séries temporais, estamos interessados em prever um único valor a partir de seus m antecessores, procuramos uma função $f : \Re^m \rightarrow \Re$ tal que:

$$f(\mathbf{x}) = \sum_{i=1}^{n} w_i h_i(\mathbf{x}) \tag{2.7}$$

em que $\mathbf{x} = \left[x_1, x_2, ..., x_m\right]$ é o vetor de entrada, de dimensão m, $h_i(\mathbf{x})$ é o valor função de base radial, do i-ésimo neurônio da camada escondida, em relação ao vetor de entrada $\mathbf{x}$ e w_i é o peso associado entre o i-ésimo neurônio da camada escondida e o neurônio de saída

Como apresentado por [12], um dos principais fundamentos das RBFNN está no teorema de Cover sobre a diferenciação de padrões que, como descrito pela referência, atesta:

Teorema 2.1 (Teorema de Cover)**.** *Um determinado problema complexo de separação de padrões tratado em um domínio de maior dimensão, de maneira não linear, é mais fácil de ser linearmente separável do que em um domínio de baixa dimensão.*

Ou seja, dado um problema de classificação de padrões, em uma dada dimensão, mapearmos para uma dimensão maior (de $\Re^2$ para $\Re^3$, por exemplo), de maneira não linear, é mais provável que possamos separar esse problema linearmente. Ao observamos a descrição da rede RBF, verificamos que essa é precisamente a função da camada escondida. Para mais detalhes sobre a prova do teorema em [12].

Existem diversos meios para treinar uma RNA RBF dispostos na literatura [12]. Da estrutura da rede, podemos inferir que os parâmetros que precisam ser definidos para uma completa descrição de uma rede RBF são:

1. Tipo de função de base radial;
2. A quantidade e posição dos centros usados pelas RBF nos nós da camada escondida;
3. A largura das RBF (parâmetro ε das funções apresentadas na Tabela 2.1);
4. E por fim, os pesos entre a camada escondida e a camada de saída.

Vale notar que a dimensão da camada de entrada e da camada de saída são específicos do problema e cabe ao projetista interpretar o ajuste dessas camadas. Neste estudo, o tamanho da camada de entrada representa o quantos valores predecessores serão necessários para especificar um determinado valor. Por sua vez, a camada de saída sempre terá apenas um nó, pois especificará o valor que desejamos prever, dados seus predecessores. Outro ponto importante vêm da seleção da RBF usada em cada nó da camada escondida. A princípio não foi identificada nenhuma restrição para ela, mas a mais comum é a Gaussiana. Esta função possui propriedades interessantes para a corretude e precisão do modelo e, como era de se esperar, é a mais amplamente utilizada [12], [13].

O método mais simples de treinamento, e por sua vez mais comum, seleciona-se a partir do conjunto de treinamento, alguns elementos para agirem como centros da camada escondida. Segundo [10], é garantido que tenhamos uma superfície que se adeque à todos os pontos do conjunto de dados de treino, contanto que existam RBFs o suficiente. Para tal, poderíamos usar cada um dos pontos de entrada como centro, porém isso significaria que o ruído e dados anormais seriam também modelados. Essa atitude causaria distorções, que refletirão na generalização, incorrendo em uma baixa qualidade de previsões. Dessa forma, precisamos encontrar um número reduzido de nós da camada escondida, de modo que seja aceitável para compreender todos os dados [10]. Em geral, esse número é encontrado a partir de experimentação. Uma vez fixada a quantidade de centros, seleciona-se aleatoriamente, porém de maneira uniforme, pontos do conjunto de treino que servirão como centros dos nós da camada escondida. Para a largura, utiliza-se um único valor para cada um dos nós da camada interna. A largura, nesse caso, é definida por [12]:

$$\sigma = \frac{d_{max}}{\sqrt{2n}} \tag{2.8}$$

em que d_{max} é maior distância entre os centros selecionados e n a quantidade de centros. Não obstante, todos os centros usam a mesma RBF, a qual mais comum é a Gaussiana, como definida na Tabela 2.1. O valor da largura definido em (2.8) para Gaussiana garante que nenhuma delas será nem muito alta nem muito baixa, extremos que devem ser evitados. Por fim, com todos estes elementos definidos falta definir os pesos entre a camada escondida e de saída. Estes, de (2.7), são obtidos a partir da solução do sistema [12]:

$$\begin{aligned} h_1(\mathbf{x}_1)w_1 + h_2(\mathbf{x}_1)w_2 + \ldots + h_n(\mathbf{x}_1)w_n &= y_1 \\ h_1(\mathbf{x}_2)w_1 + h_2(\mathbf{x}_2)w_2 + \ldots + h_n(\mathbf{x}_2)w_n &= y_2 \\ &\vdots \\ h_1(\mathbf{x}_k)w_1 + h_2(\mathbf{x}_k)w_2 + \ldots + h_n(\mathbf{x}_k)w_n &= y_k \end{aligned} \tag{2.9}$$

tal que:

$$\begin{bmatrix} h_1(\mathbf{x}_1) & h_2(\mathbf{x}_1) & \cdots & h_n(\mathbf{x}_1) \\ h_1(\mathbf{x}_2) & h_2(\mathbf{x}_2) & \cdots & h_n(\mathbf{x}_2) \\ \vdots & & \ddots & \vdots \\ h_1(\mathbf{x}_k) & h_2(\mathbf{x}_k) & \cdots & h_n(\mathbf{x}_k) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{2.10}$$

em que:

$$\mathbf{Hw} = \mathbf{y} \tag{2.11}$$

então:

$$\mathbf{w} = \mathbf{H}^{+}\mathbf{y} \tag{2.12}$$

em que $\mathbf{H}^{+}$ é a matriz pseudo-inversa de $\mathbf{H}$, tal que $\mathbf{H}^{+} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T$.

O segundo método, faz uso de algoritmos de clusterização e de classificação para inferir os parâmetros necessários da RBFNN. Assim como o método anterior, a determinação de quantos nós da camada escondida existirão é através de experimentação. Uma vez fixado o valor para n centros, utiliza-se um algoritmo de *kmeans* para se determinar onde serão posicionados os centros. Essa abordagem permite uma melhor aproximação dos dados de treino uma vez que sabe-se onde uma maior quantidade de dados está localizada. Após determinar-se a quantidade e a posição de cada um dos centros, com o auxílio do algoritmo *kmeans*, utiliza-se o algoritmo *K Nearest Neighbors* (KNN) sobre o conjunto de centros encontrado nos passo anterior, para se determinar a largura de cada uma das Gaussianas. Como apresentado em [13], um valor de K=2 para o algoritmo de KNN é usualmente satisfatório. Por fim, assim como no método anterior, os pesos entre a camada escondida e a camada de saída são definidos pelo meio de operação com pseudo-inversa.

2.4 Modelo híbrido

Modelos baseados na combinação de modelos estocásticos tradicionais com as mais novos avanços no campo de Redes Neurais Artificiais têm apresentado bons resultados [3], [5], [14]. Em geral, é difícil detectar na prática, se uma determinada série representa um processo linear ou não-linear. Esse fato complica a seleção do modelo apropriado bem como se um determinado modelo é mais eficiente que outro. Normalmente, diversos modelos são testados, e aquele com melhor desempenho, é selecionado. Além disso, em casos reais, raramente uma série temporal é puramente linear ou não-linear. Por fim, está bem claro na literatura a respeito de previsões, que não existe um modelo capaz de resolver todos os problemas. Em vista destes fatos descritos, diversos modelos híbridos estão disponíveis. Vários desses exemplos mostram que a combinação de diversos modelos trazem, em geral, melhores resultados. Além disso, modelos híbridos conseguem tratar melhor diversas características distintas do problema, como por exemplo, a linearidade ou não-linearidade [5].

No trabalho desenvolvido por [5], ficou evidente como modelos de ARIMA e RNA conseguiram grande sucesso no domínio linear e não-linear, respectivamente. Em casos reais, entretanto, onde é difícil ter conhecimento sobre todas as características do sistema, o modelo híbrido é mais adequado. Em seu trabalho, Zhang assume que uma determinada série temporal é da seguinte formato:

$$y_t = L_t + N_t \tag{2.13}$$

em que L_t representa a componente linear e N_t a componente não-linear. Dessa forma, o ARIMA fica responsável modelar a componente linear, de modo que, os resíduos dessa previsão conterão apenas o padrões não-lineares. Nesse ponto, a RNA recebe como entrada a série temporal original e também os resíduos da previsão do ARIMA. A Rede Neural possui então, a função de modelar a componente não linear, que encontra-se no resíduo do ARIMA. Por fim, ambas as previsões, da componente linear (ARIMA) e da componente não-linear (RNA), somam-se, fornecendo a previsão do modelo. A Figura 2.6 representa uma visão geral do modelo proposto por [5].

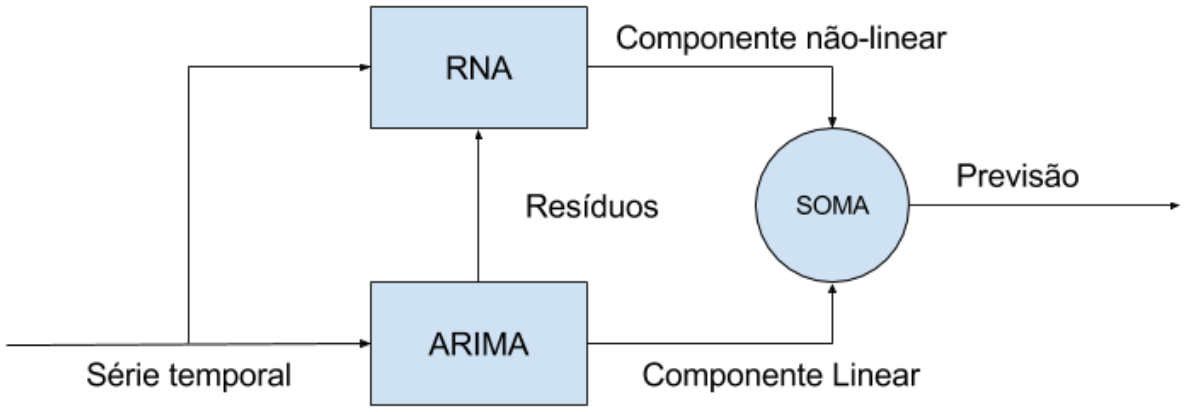

Figura 2.6 Esquema do modelo híbrido proposto por [5].

Neste trabalho, o modelo proposto tratará a série temporal primeiramente pelo ARIMA, que modela a componente linear, em seguida, os resíduos da previsão, juntamente com a própria série são usados para treinar a rede RBF, como descrito no tópico sobre RBFNN. Com a componente não-linear modelada, ambos os resultados (ARIMA e RBFNN) são usados em um combinador, que pode ser uma soma simples, como apresentado no modelo proposto por [5], uma rede MLP ou uma outra RBFNN. A Figura 2.7 apresenta um esquema do modelo proposto.

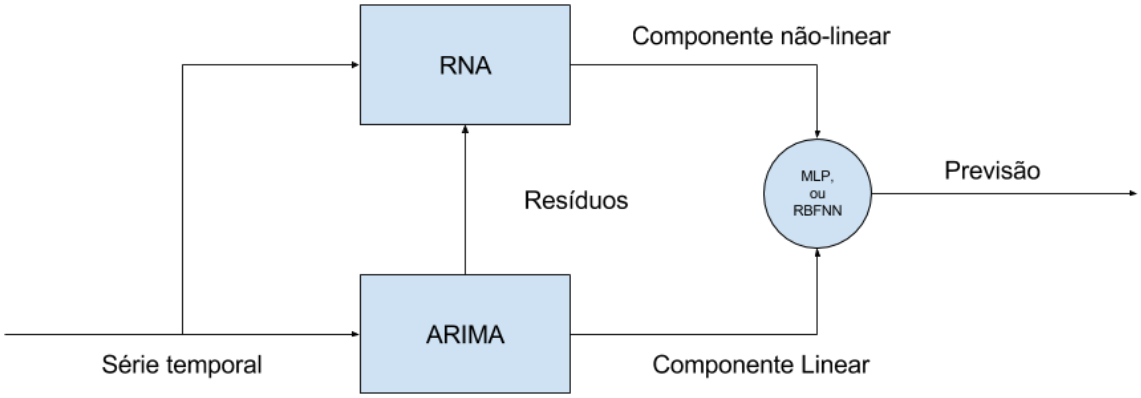

Figura 2.7 Esquema do modelo híbrido proposto com uso de combinadores não lineares.

CAPÍTULO 3

Metodologia de Experimentos

Nesta seção, será apresentada a metodologia dos experimentos realizados.

3.1 Conjunto de dados

As séries utilizadas foram a *Canadian Lynx*, que representa o total de linces capturados por ano no distrito canadense do Rio Mackenzie no período de 1821 a 1934. A segunda série é a *Sunspot*, que representa a quantidade anual de manchas solares. Essa foi uma das primeiras séries temporais já registradas e possui 269 registros desde 1700. Manchas solares (*sunspots*) são importantes indicadores de variações eletromagnéticas provocadas pelo Sol. Como consequência, sentimos interferência nas comunicações de nossos satélites artificiais além de variações climáticas. A terceira e última série é taxa de conversão semanal da Libra Esterlina para Dólar Americano, *BPUSD*. Todas as séries são comumente utilizadas para testes de modelos para séries temporais e possuem características distintas que as tornam úteis para testes apresentados [5]. Por conveniência, as séries *Lynx*, *Sunspot* e *BPUSD* serão abreviadas como *LY*, *SU* e *BU*, respectivamente. Todas as séries foram normalizadas entre os valores segundo (3.1) e estão exibidas nas Figuras 3.1 a 3.3.

$$x_t = \frac{x_t - x_{min}}{x_{max} - x_{min}} \tag{3.1}$$

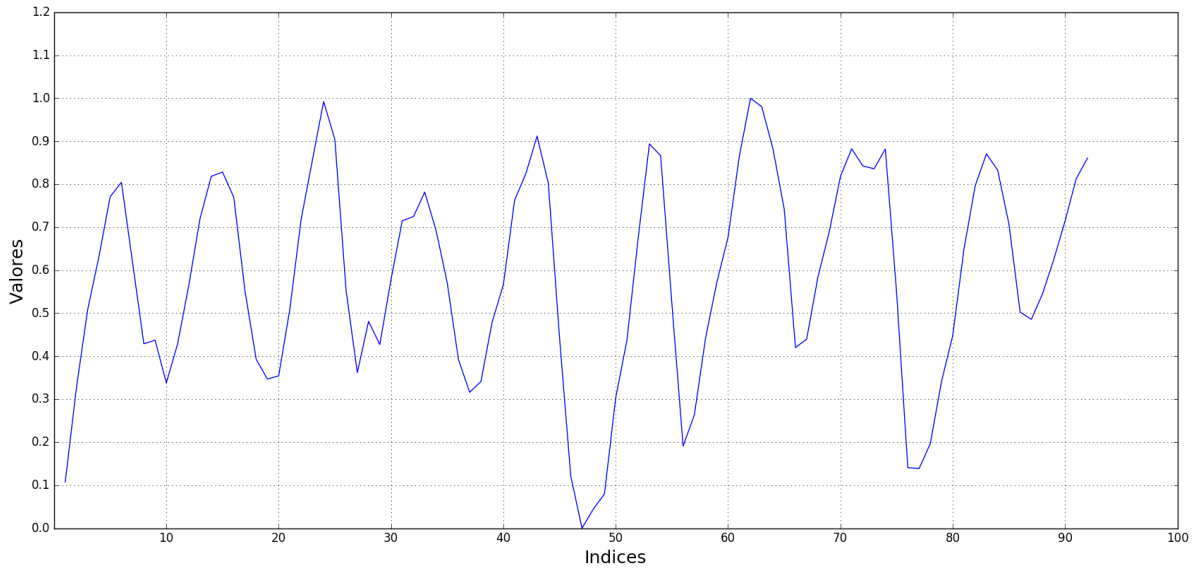

Figura 3.1 Série temporal *Lynx* normalizada.

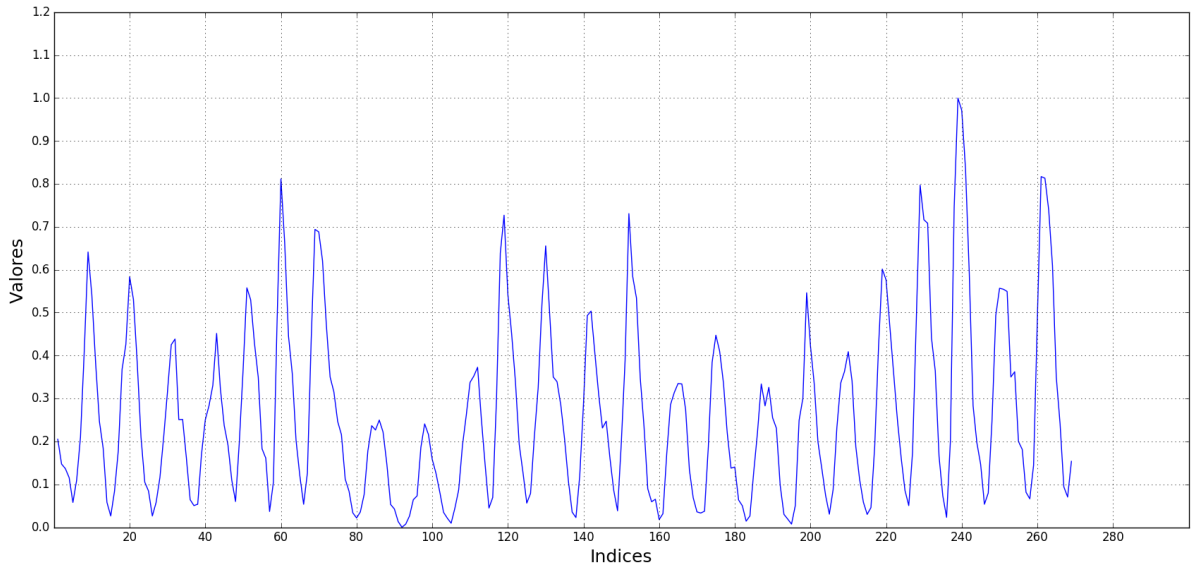

Figura 3.2 Série temporal *Sunspot* normalizada.

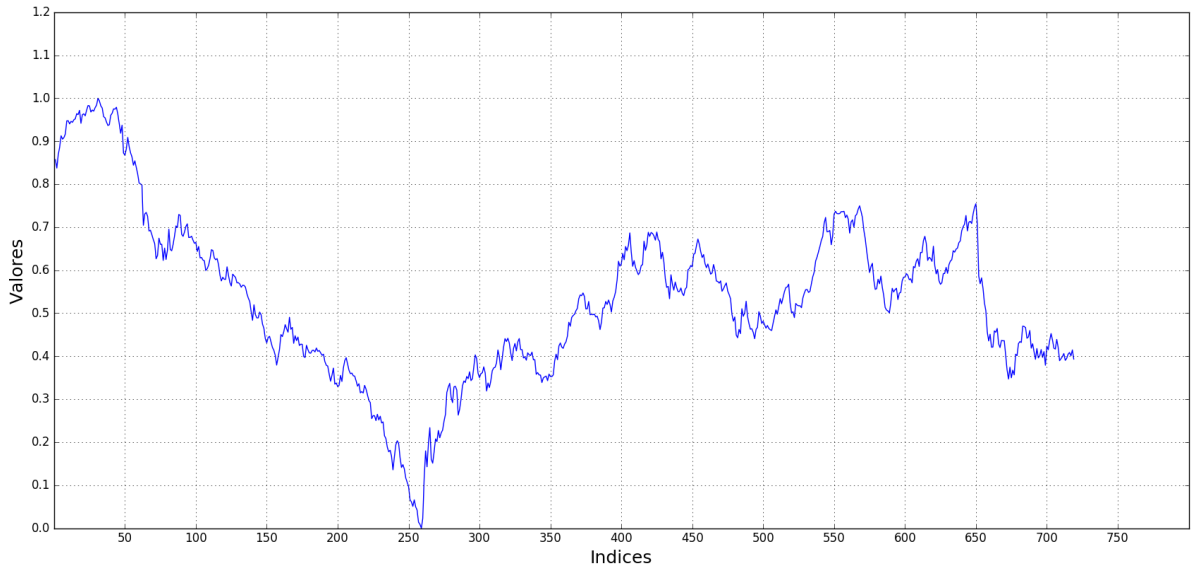

Figura 3.3 Série temporal *BPUSD* normalizada.

O modelo ARIMA, ao processar os dados das séries, gera uma saída que representa sua previsão. Essa saída é também uma série e, nesse caso, usamos as abreviações LY_{ARIMA}, SU_{ARIMA} e BU_{ARIMA} para as previsões para as séries de *Lynx*, *Sunspot* e *BPUSD*, respectivamente. As Figuras 3.4 a 3.6 mostram as séries geradas pelo ARIMA em relação à série original.

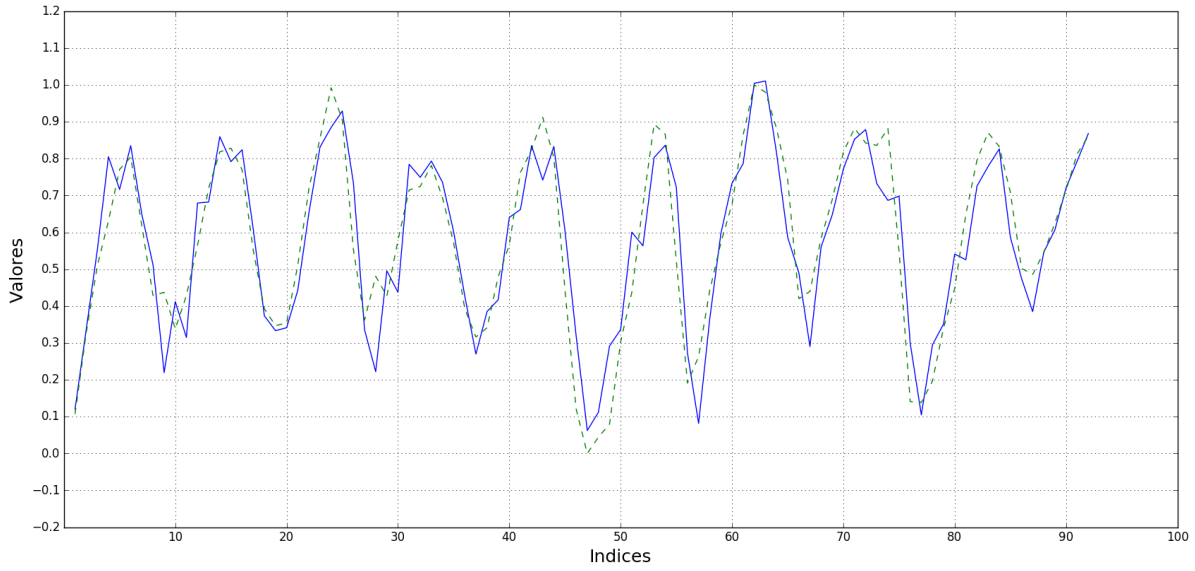

Figura 3.4 Série temporal da previsão do ARIMA (contínua azul) com a real (tracejada verde) para *Lynx*.

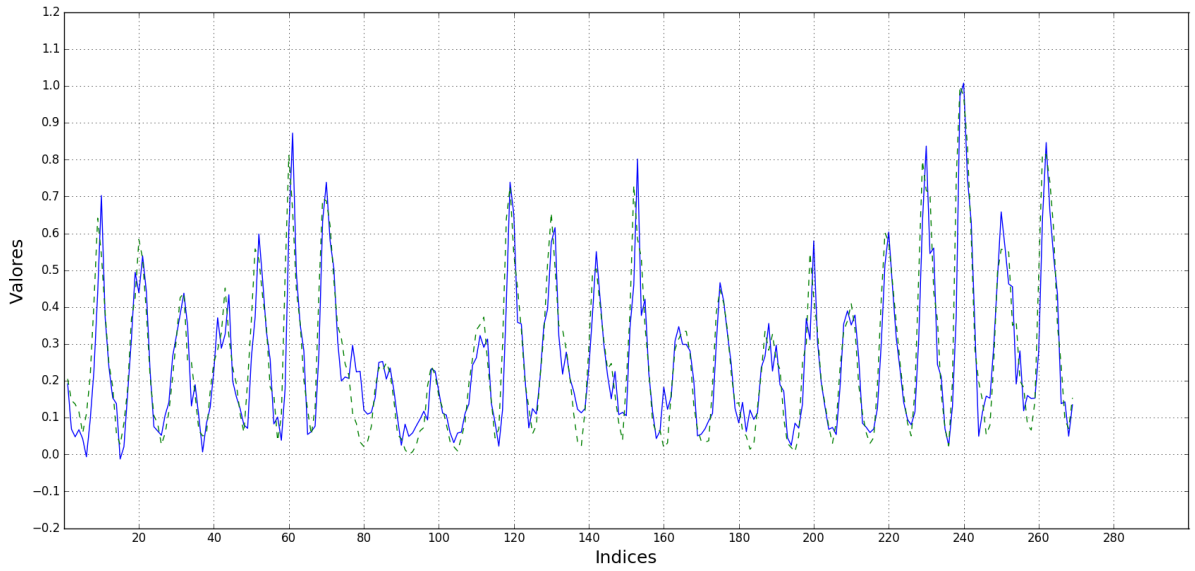

Figura 3.5 Série temporal da previsão do ARIMA (contínua azul) com a real (tracejada verde) para *Sunspot*.

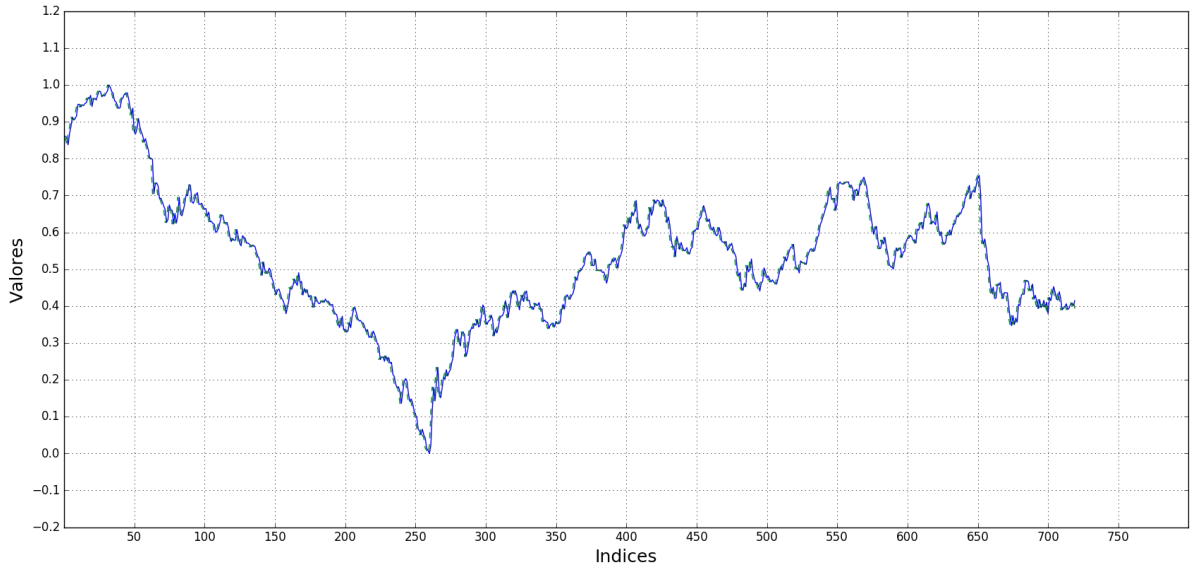

Figura 3.6 Série temporal da previsão do ARIMA (contínua azul) com a real (tracejada verde) para *BPUSD*.

De modo similar, a série de resíduos geradas pela previsão do ARIMA, definimos como $LY_{ARIMA-ERR}$, $SU_{ARIMA-ERR}$ e $BU_{ARIMA-ERR}$, estão apresentadas nas Figuras 3.7 a 3.9.

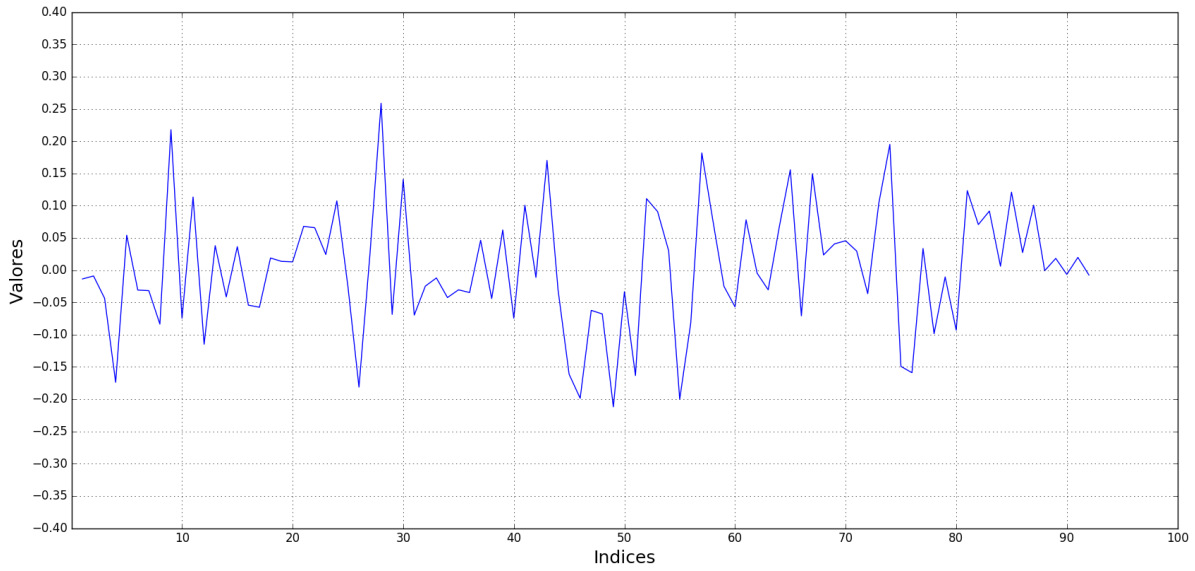

Figura 3.7 Série temporal do erro da previsão ARIMA para *Lynx*.

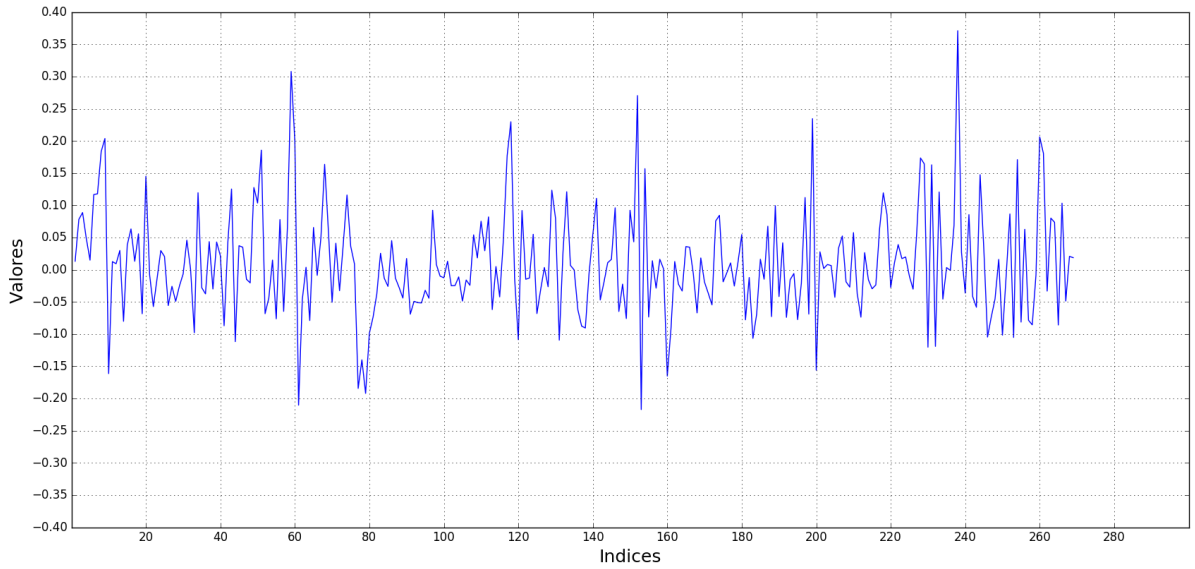

Figura 3.8 Série temporal do erro da previsão ARIMA para *Sunspot*.

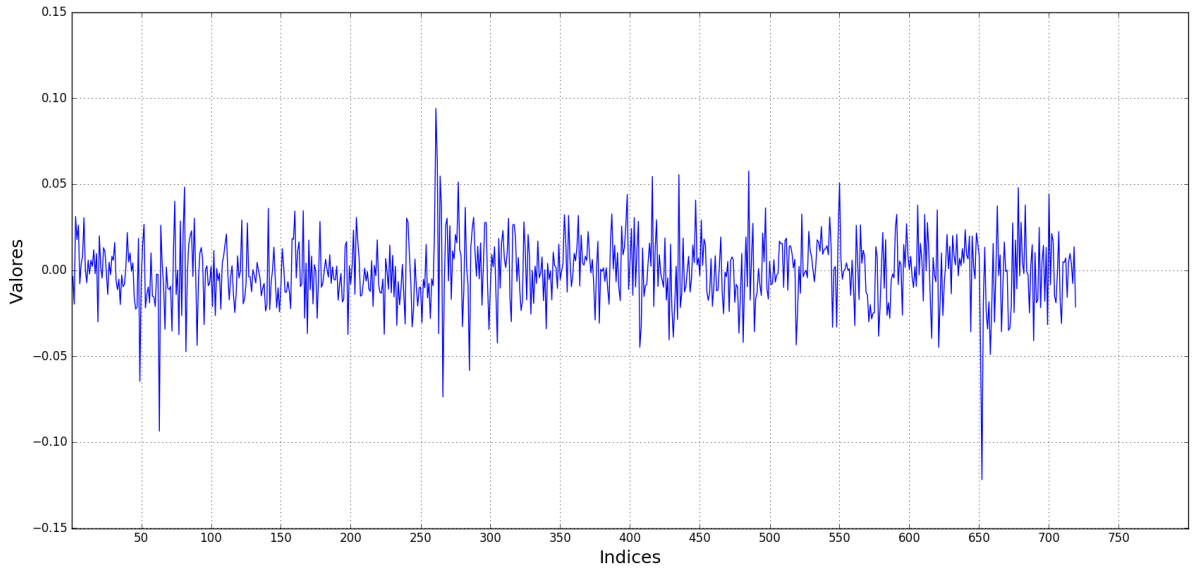

Figura 3.9 Série temporal do erro da previsão ARIMA para *BPUSD*.

3.2 Métricas utilizadas

As métricas usadas neste trabalho são apenas algumas de um vasto grupo de métricas possíveis. Para cada uma das métricas descritas a seguir, considere N o tamanho da série, u_t o valor da série no índice t, v_t o valor da previsão para o mesmo índice t e finalmente, $\bar{u}$ a média da série. *Mean Squared Error* (MSE) é uma das métricas mais comuns no processo de predição de séries temporais. Representa uma medida de qualidade geral do preditor, avaliando sua variabilidade [6]. A métrica *Mean Absolute Percentage Error* (MAPE) assim como o MSE avalia o erro absoluto da previsão, porém possui uma perspectiva independente da escala (absoluto) do erro da previsão. As Equações 3.4 e 3.5 definem a métrica *Prediction Of Change In Direction* (POCID) e refere-se à capacidade de prever mudanças na taxa de crescimento da série temporal. Em outra palavras, indica se a série irá crescer ou decrescer. A métrica Theil refere-se à *U of Theil Statistics*, e nos informa a precisão do modelo em relação ao modelo de *Random Walk* de modo que, se o valor for superior à 1, então o modelo possui um desempenho pior que o modelo *Random Walk*. Em contrapartida, se for inferior à 1 e mais próximo de 0, o modelo possui desempenho melhor. Se for igual à 1, o modelo é tão bom quanto o modelo *Random Walk*. Por fim, *Average Relative Variance* (ARV) nos diz quão melhor é a previsão baseando-se na média da série. De modo semelhante à métrica Theil, se o ARV=1 então as previsões são tão boas quanto se usássemos apenas a média simples da série. Se for ARV < 1 as predições são melhores que a média, caso contrário, se ARV > 1 as predições são piores que a média [14]. Cada uma das métricas apresentadas estão definidas a seguir:

$$MSE = \frac{1}{N}\sum_{t=1}^{N}(u_t - v_t)^2 \tag{3.2}$$

$$MAPE = \frac{100}{N}\sum_{t=1}^{N}\left|\frac{u_t - v_t}{u_t}\right| \tag{3.3}$$

$$POCID = 100 \cdot \frac{\sum_{t=1}^{N} D_t}{N} \tag{3.4}$$

$$D_t = \begin{cases} 1, & \text{se } (u_t - u_{t-1})(v_t - v_{t-1}) > 0. \\ 0, & \text{caso contrário.} \end{cases} \tag{3.5}$$

$$Theil = \frac{\sum_{t=1}^{N}(u_t - v_t)^2}{\sum_{t=1}^{N}(v_t - v_{t+1})^2} \tag{3.6}$$

$$ARV = \frac{\sum_{t=1}^{N}(u_t - v_t)^2}{\sum_{t=1}^{N}(v_t - \bar{u})^2} \tag{3.7}$$

3.3 Experimentos

Como visto no capítulo anterior, uma das características principais das séries temporais é que dado um resultado x_t, em um determinado instante t, este valor é dependente de seus valores passados da série como exposto em (2.1). Foi visto também em (2.2) que buscamos aproximar uma função $h(\cdot)$ de $f(\cdot)$ de modo à minimizar seu resíduo. De (2.2), para uma determinada série temporal com M observações, temos que em que $y_t = h(x_{t-1}, x_{t-2}, \ldots, x_{t-k})$ é a previsão da série no instante t e $k < M$, onde k é chamado de *lag*, e representa quantos valores passados são necessários para realizarmos uma determinada previsão. A partir de (2.2), verificamos que precisamos transformar o conjunto de entradas da série temporal em um conjunto de vetores tais que, sejam $x_1, x_2, \ldots, x_M$ o conjunto de valores da série temporal para os períodos $t = 1, 2, ..., M$, queremos

$$\mathbf{X} = \begin{bmatrix} x_1 & x_2 & \cdots & x_k \\ x_2 & x_3 & \cdots & x_{k+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{M-k} & x_{M-k-1} & \cdots & x_M \end{bmatrix} \tag{3.8}$$

em que $\mathbf{X}$ representa a matriz com os vetores de entrada. Assim, podemos usar estes vetores como entrada para o modelo. Portanto, a transformação de (3.8) é realizada em todas as séries.

A seguir, serão apresentados os cinco experimentos, realizados para cada uma das séries.

3.3.1 Experimento 1: Previsão da série através da RBFNN

Inicialmente, para critério de comparação, foi realizado um experimento para se verificar como o modelo de RBFNN se comporta ao tentar prever as séries isoladamente. Nesse experimento, não foi utilizado o modelo ARIMA. Para tal, precisamos treinar o modelo sobre o conjunto de dados, que ficou dividido da forma como exposto na Tabela 3.1:

Tabela 3.1 Divisão dos dados realizada para treino da RBFNN.

Série	Treino	Validação	Teste	Total
LY	64 (70%)	14 (15%)	14 (15%)	92
SU	189 (70%)	40 (15%)	40 (15%)	269
BU	503 (70%)	108 (15%)	108 (15%)	719

O ciclo de treino consiste em treinar o conjunto de pesos da rede sobre o conjunto de treino, para cada uma das combinações possíveis da quantidade de nós de entrada e quantidade de nós da camada escondida. Então, o melhor modelo é escolhido a partir de sua métrica MSE (3.2).

3.3.2 Experimento 2: Modelagem do Erro com uso de RBFNN

O próximo experimento tem como objetivo treinar a Rede Neural RBF para modelar o erro gerado pelo ARIMA. Esse experimento é importante, pois nele definiremos os parâmetros do modelo de RBFNN que será usada nos experimentos seguintes de teste do modelo híbrido proposto por [5], o modelo híbrido com combinador usando uma rede MLP e o modelo híbrido com combinador RBFNN. Aqui, repete-se a mesma abordagem usada no experimento 1 para o treino dos parâmetros da rede, ou seja, cada ciclo de treino consiste em treinar o conjunto de pesos da rede para cada uma das combinações possíveis da quantidade de nós de entrada e quantidade de nós da camada escondida. Então, o melhor modelo é escolhido a partir de sua métrica MSE (3.2). Uma vez que as quantidades de entradas para a série de resíduos do ARIMA, a divisão dos dados é a mesma exposta na Tabela 3.1.

3.3.3 Experimento 3: Combinação das previsões do modelo ARIMA e RBFNN através de soma

Com o resultado do segundo experimento, temos as RNAs RBF que melhor se adaptam ao erro do ARIMA para cada uma das séries. Agora podemos realizar o teste do modelo híbrido, com combinador com soma aritmética. Nenhum treino é necessário nesse caso, uma vez que ambos os modelos já foram previamente treinados. Dessa forma, usa-se a rede RBFNN para prever os últimos 15% das séries de erro do ARIMA, juntamente com os últimos 15% das séries ARIMA, e soma-se os valores de índices equivalentes. O resultado é a série de previsão do modelo híbrido.

3.3.4 Experimento 4: Combinação das previsões do modelo ARIMA e RBFNN através de rede MLP

Para o quarto experimento, continuamos usando a rede RBFNN obtida no segundo experimento para a previsão do erro do ARIMA, porém usaremos uma RNA MLP como combinador

da componente linear e não-linear. Redes MLP foram e continuam sendo extensamente pesquisadas e atualmente, existem inúmeras implementações desse modelo. Isso se deve ao fato da grande diversidades de funções que redes desse tipo conseguem aproximar.

No caso deste experimento, a quantidade de camadas é fixa e definida como: uma camada de entrada, uma camada escondida, e uma camada de saída. A camada de entrada possui apenas dois neurônios, um para os valores vindos da previsão do ARIMA e outro para os valores vindo da previsão da rede RBF. Na camada escondida, foi feita uma busca incremental, com o objetivo de identificar o número ideal de neurônios nessa camada. A camada de saída só possui um único neurônio, já que se trata de uma regressão.

Uma vez que a rede RBF, adquirida do segundo experimento, necessita de uma certa quantidade de dados passados para que se possa prever o imediatamente adjacente, nesse experimento alguns pontos (em particular, os iniciais) foram deixados de fora devido à esse fato. Primeiro, não foi feito o seccionamento dos conjuntos pois, ambos os modelos (ARIMA e RBFNN) já haviam sido treinados. Após o ARIMA e a RBFNN fornecerem suas previsões, elas serão usadas então para treinar a rede MLP que atuará como combinadora dos dados. A segmentação dos dados para os conjuntos de treino, validação e teste da MLP estão expostos na Tabela 3.2, bem como a quantidade de valores prévios usados necessários.

Tabela 3.2 Divisão dos dados realizada para treino da MLP.

Série	Pontos descartados	Treino	Validação	Teste	Total
LY	6	60 (70%)	13 (15%)	13 (15%)	86
SU	34	164 (70%)	36 (15%)	35 (15%)	269
BU	110	426 (70%)	92 (15%)	91 (15%)	719

3.3.5 Experimento 5: Combinação das previsões do modelo ARIMA e RBFNN através de rede RBFNN

Assim como no experimento anterior, o quinto e último experimento investiga o uso de uma outra Rede Neural como combinador das componentes linear (ARIMA) e não linear (RBFNN). Neste caso, além da rede RBF adquirida no experimento 2, para parevisão do erro do ARIMA, uma outra rede RBF será usada como combinadora. Os conjuntos de treino, validação e testes são os mesmo apresentados no experimento anterior e estão dispostos na Tabela 3.2

CAPÍTULO 4

Resultados

Nesta seção serão apresentados os resultados dos experimentos apresentados co capítulo anterior. A Tabela 4.1 apresenta resumo dos resultados obtidos.

Tabela 4.1 Resumo dos resultados dos experimentos. Para esta tabela, por uma questão de formatação, considere: H_{SUM} = SUM(ARIMA, RBFNN), H_{MLP} = MLP(ARIMA, RBFNN) e H_{RBFNN} = RBFNN(ARIMA, RBFNN)

Série	Métricas	ARIMA	RBFNN	H_{SUM}	H_{MLP}	H_{RFFNN}
LY	MSE	0.0046	**0.0031**	0.0062	0.0079	0.0054
	MAPE	**8.11**	8.53	11.07	11.77	8.98
	POCID	78.57	**92.86**	**92.86**	84.62	91.67
	Theil	0.2997	**0.1802**	0.4082	0.7430	0.4775
	ARV	0.1743	**0.1095**	0.1663	0.3245	0.2336
SU	MSE	**0.0119**	0.0511	0.0148	0.0155	0.1059
	MAPE	**33.27**	44.01	40.56	43.46	62.51
	POCID	70.73	**87.80**	73.17	72.50	57.50
	Theil	**0.3258**	2.6531	0.3264	0.6103	1.3194
	ARV	**0.1630**	0.7988	0.1947	0.3172	1.6713
BU	MSE	**0.0006**	**0.0006**	**0.0006**	**0.0006**	0.0018
	MAPE	3.64	3.70	**3.62**	4.15	5.21
	POCID	40.67	41.67	**42.59**	41.38	36.78
	Theil	1.0000	1.0172	**0.9876**	1.1520	4.7705
	ARV	**0.0403**	0.0409	0.0420	0.0488	0.1924

Ao analisarmos os dados da Tabela 4.1, notamos um comportamento inesperado. Apesar do modelo proposto por [5] demonstrarem resultados satisfatórios, para as mesmas séries estudadas, este trabalho de graduação não foi capaz de confirmar os resultados obtidos por Zhang, exceto para o caso da série *BPUSD*, nem o modelo aqui proposto apresentou resultados satisfatórios, comparativamente.

Observamos que para a série *Lynx* os melhores resultados foram apresentados pelo modelo de Rede Neural RBF (RBFNN), com ressalva apenas para a deficiência mínima na métrica MAPE. Ainda assim dentre todos os modelos, apresentou o segundo melhor resultado para essa métrica. Portanto, dadas as características da série *Lynx*, o modelo indicado para sua previsão, dentre os estudados, seria o de RBFNN. Para essa série, verifica-se também que dentre os modelos híbridos, o de RBFNN(ARIMA, RBFNN) teve performance superior. Um fator agravante para a série *Lynx* é a sua quantidade escassa quantidade observações, o que dificulta o treinamento das redes RBF e em particular, MLP.

Para a série de *Sunspot*, observa-se que o modelo ARIMA isoladamente apresentou métricas superiores aos demais modelos. Assim como as métricas da série *Lynx*, ficou atrás apenas com o POCID. Este nesse caso, obteve o pior resultado. Dados os resultados, conclui-se que para as características apresentadas pela série *Sunspot*, o melhor modelo que a caracteriza, dentre os estudados, é o ARIMA.

Para o caso da série *BPUSD*, verificamos um comportamento diferente do padrão das demais séries. Esta foi a única série em que algum dos modelos híbridos obteve sucesso. Vale ressaltar também, que a quantidade de entradas que a série possui é muito superior às demais. Na Tabela 4.4 pode-se observar que a quantidade de nós da camada de entrada para *BPUSD* é superior em uma escala de no mínimo vinte vezes.

O detalhamento de cada experimento está detalhado nas seções seguintes.

4.1 Resultados do experimento 1

Para seleção de centros, foi usado o método *K-Means* no conjunto de entrada, onde o número de centróides foi determinado através de uma busca incremental. Seja N_c o número de centróides, onde $N_c = 3,4,...,30$. Da mesma forma, o tamanho do *lag* foi selecionado a partir de uma busca incremental, em que $k = 1,2,...,0.25 \cdot N$ e N é a quantidade de entradas da série. O parâmetro da largura, que será usado pelos centros das RBFs, por sua vez, foi determinado utilizando o algoritmo de *KNN*, como especificado no capítulo anterior. Portanto, cada nó da camada escondida possui seu próprio valor de largura. Para o treinamento dos pesos foi utilizado procedimento de pseudo-inversa. A Tabela 4.2 exibe os parâmetros das redes RBF usadas para prever cada uma das séries. As Figuras 4.1 à 4.3 exibem a previsão do modelo assim como a série real.

Tabela 4.2 Parâmetros da rede RBFNN para predição das séries.

Série	Nós de entrada	Nós da camada escondida	Nós de saída
LY	2	12	1
SU	7	18	1
BU	1	9	1

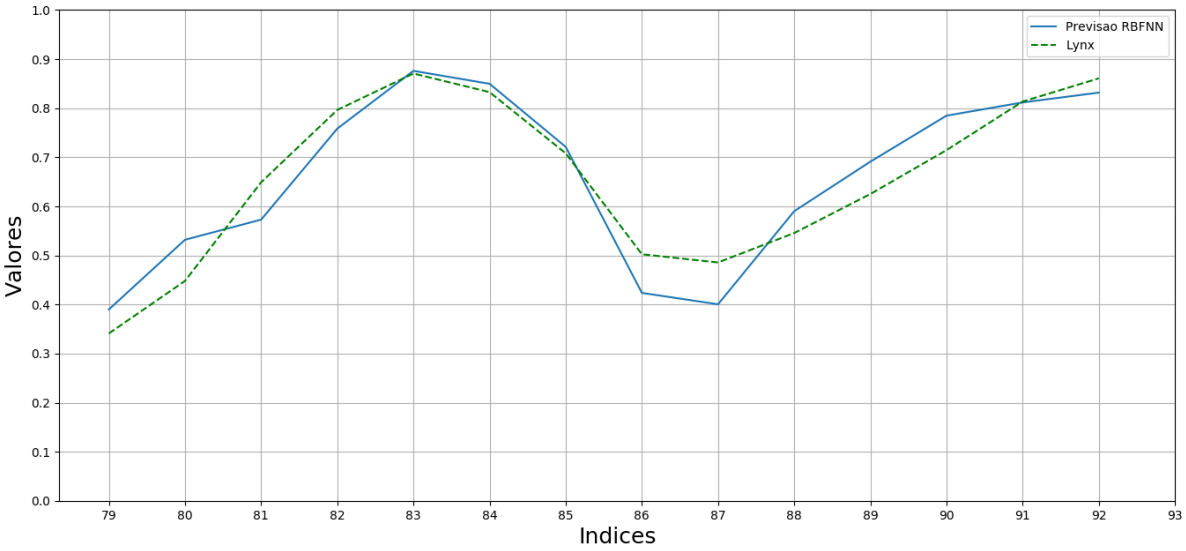

Figura 4.1 Previsão da série *Lynx* através da RBFNN com relação à série original.

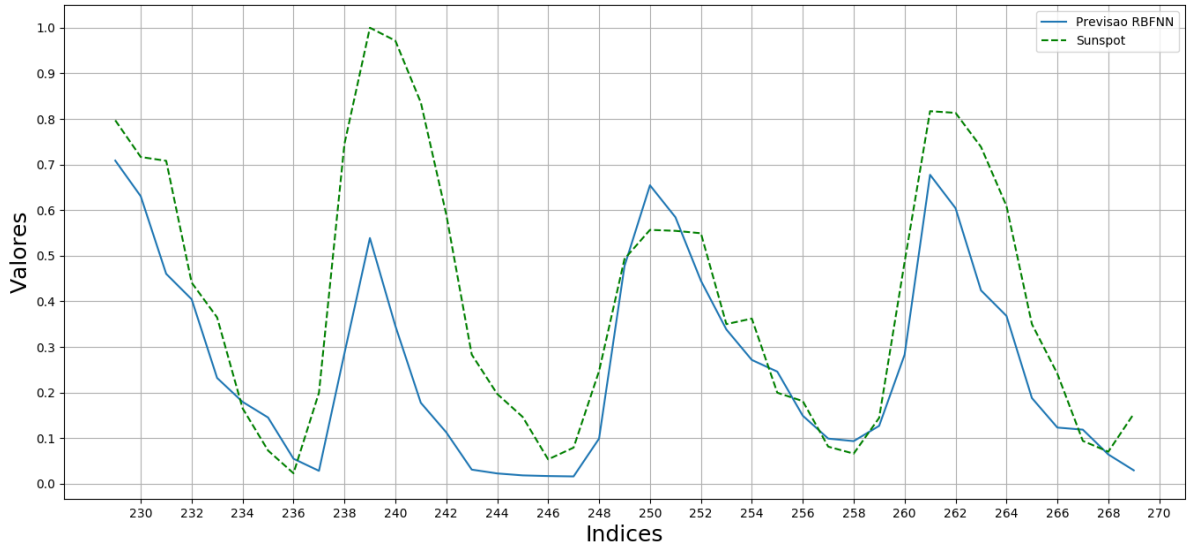

Figura 4.2 Previsão da série *Sunspot* através da RBFNN com relação à série original.

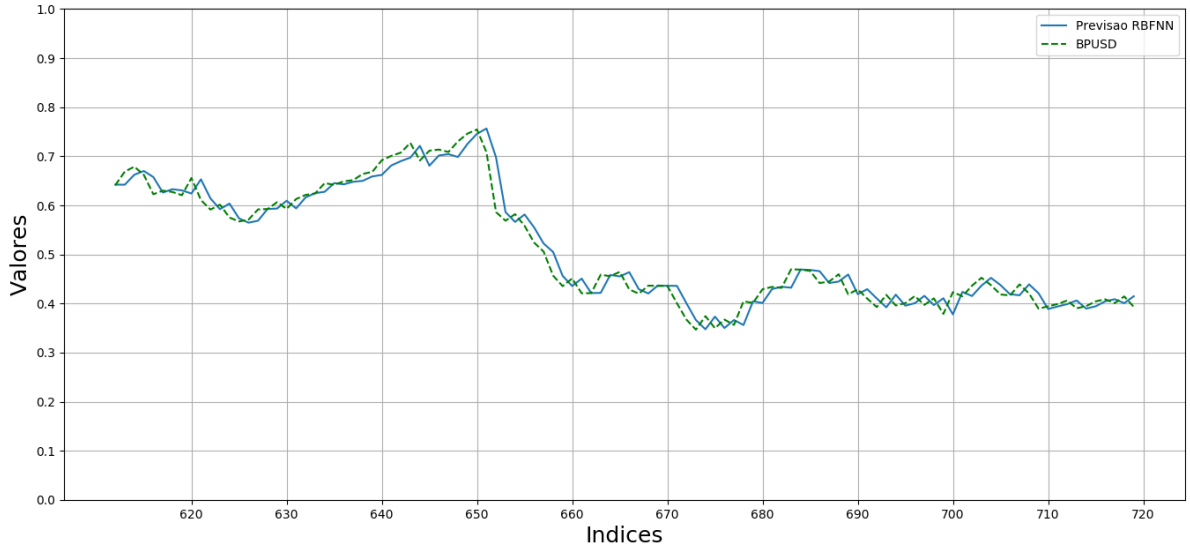

Figura 4.3 Previsão da série *BPUSD* através da RBFNN com relação à série original.

4.2 Resultados do experimento 2

Na Tabela 4.3 estão dispostas as métricas obtidas para a previsão do erro do ARIMA com uso das RBFNN com estruturas descritas na Tabela 4.4. As Figuras 4.4 a 4.6 exibem a série prevista pelo modelo bem como série real.

Tabela 4.3 Métricas para previsão das séries de erro do ARIMA com RBFNN.

Série	MSE	MAPE	POCID	Theil	ARV
$LY_{ARIMA-ERR}$	0.0068	592.84	85.71	0.5710	0.8850
$SU_{ARIMA-ERR}$	0.0151	152.78	34.15	6.1640	11.3069
$BU_{ARIMA-ERR}$	0.0006	1.66	51.85	87.7358	37.6179

Tabela 4.4 Parâmetros das redes RBFNN para predição do erro do ARIMA.

Série	Nós de entrada	Nós da camada escondida	Nós de saída
$LY_{ARIMA-ERR}$	6	28	1
$SU_{ARIMA-ERR}$	1	12	1
$BU_{ARIMA-ERR}$	135	12	1

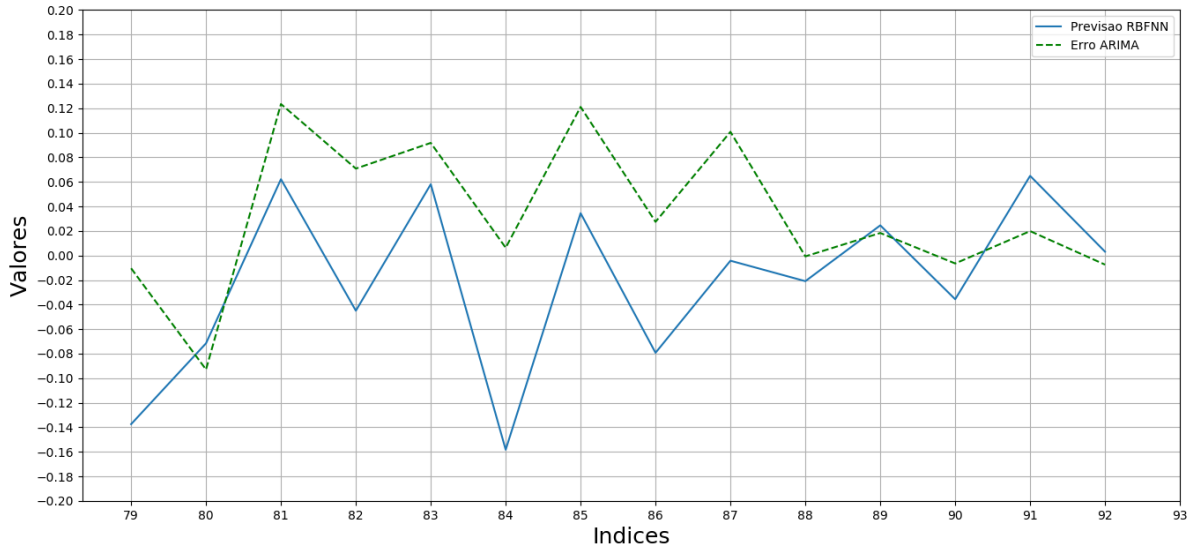

Figura 4.4 Previsão do erro do ARIMA para a série *Lynx* através da RBFNN com relação à série original.

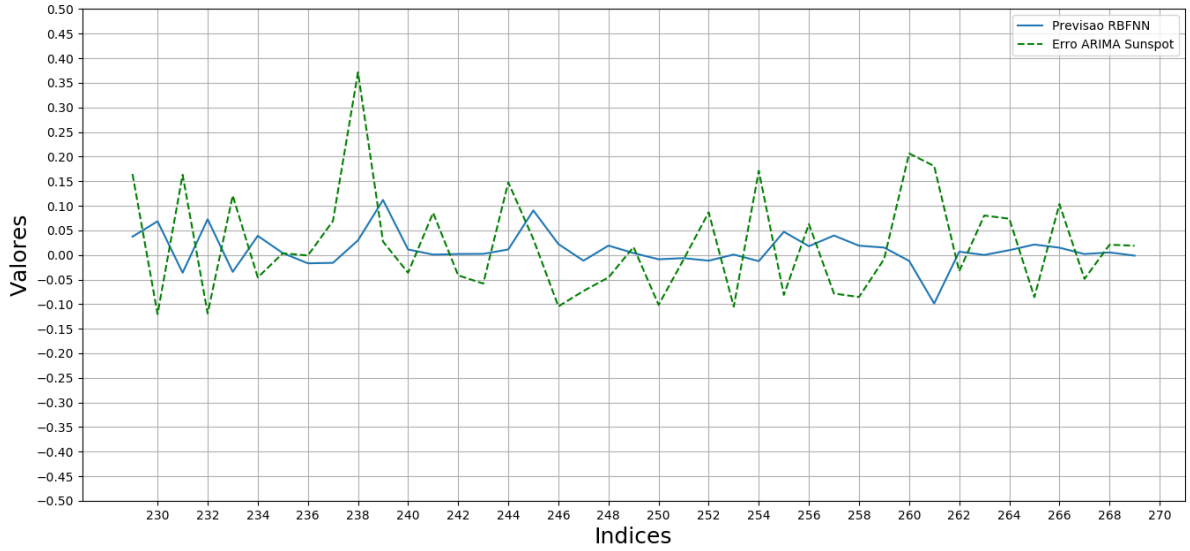

Figura 4.5 Previsão do erro do ARIMA para a série *Sunspot* através da RBFNN com relação à série original.

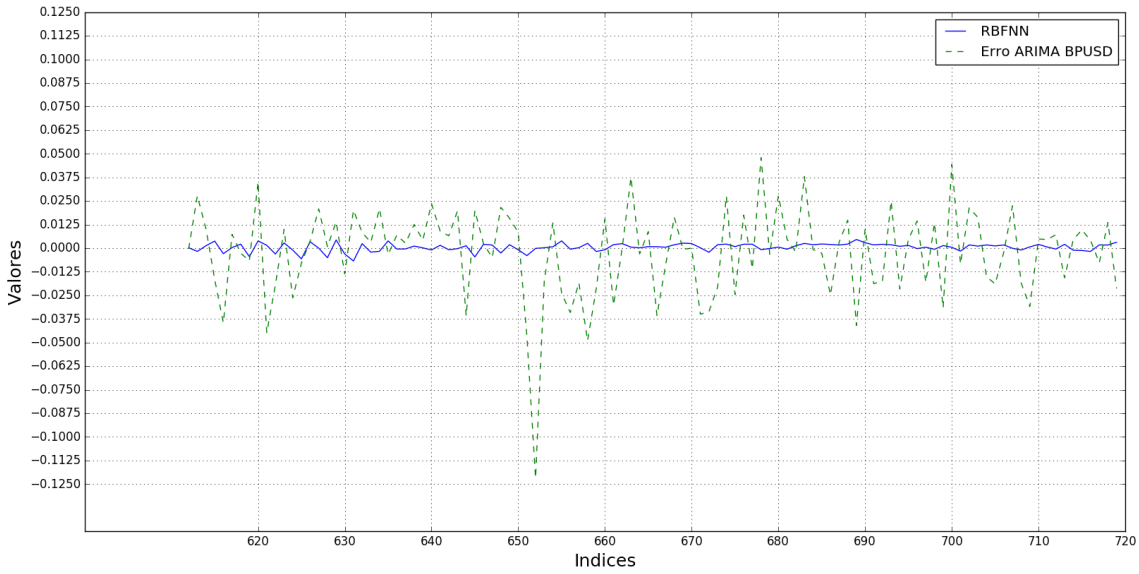

Figura 4.6 Previsão do erro do ARIMA para a série *BPUSD* através da RBFNN com relação à série original.

4.3 Resultados do experimento 3

Para este experimento, a RBFNN que irá tratar o erro do ARIMA com relação às séries é o mesma que foi obtida no experimento 2, e sua configuração está exposta na Tabela 4.4. As Figuras 4.7 a 4.9 exibem a série real bem como a série prevista, para cada uma das séries estudadas.

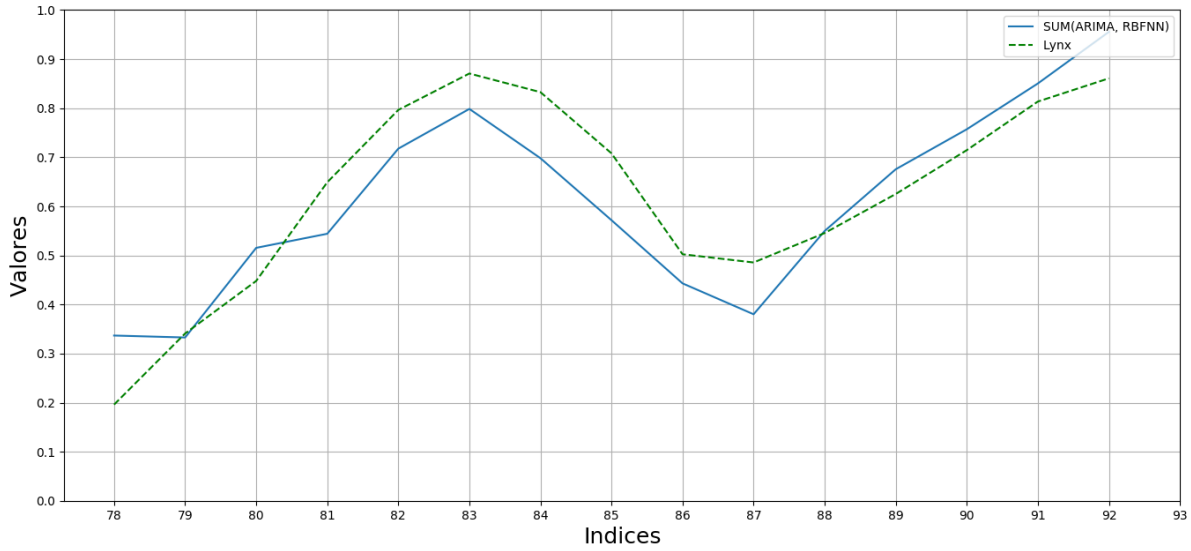

Figura 4.7 Previsão da série *Lynx* através do modelo híbrido SUM(ARIMA, RBFNN) relação à série original.

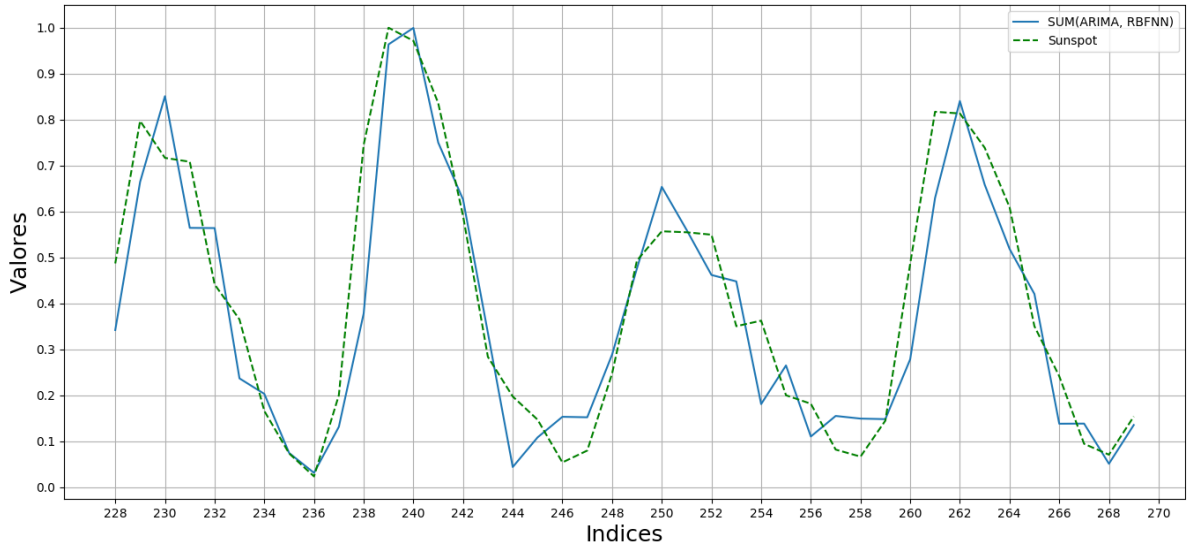

Figura 4.8 Previsão da série *Sunspot* através do modelo híbrido SUM(ARIMA, RBFNN) relação à série original.

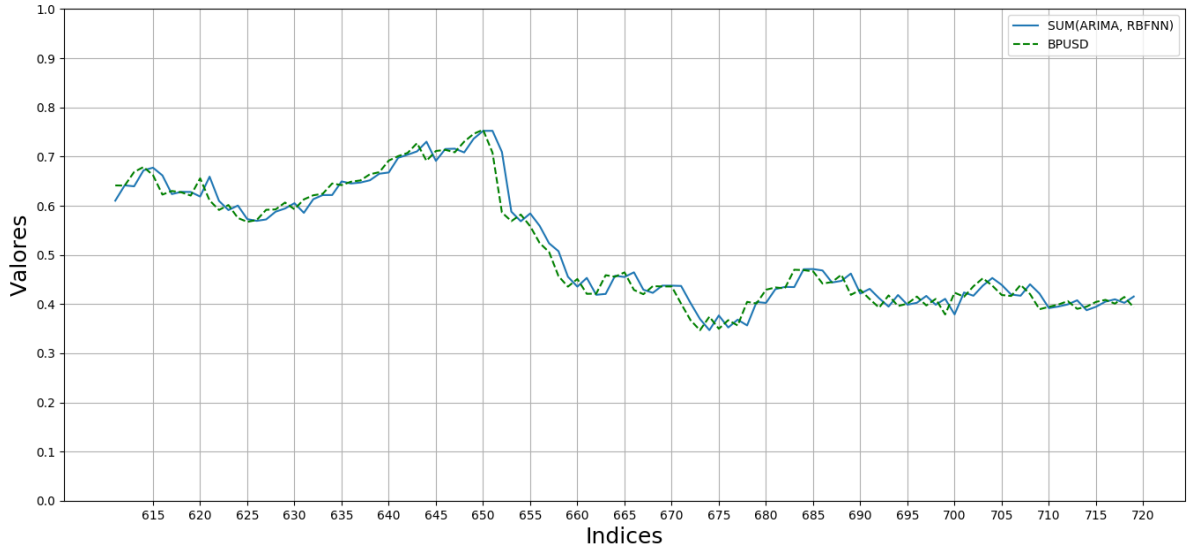

Figura 4.9 Previsão da série *BPUSD* através do modelo híbrido SUM(ARIMA, RBFNN) relação à série original.

4.4 Resultados do experimento 4

Para o quarto experimento, de previsão com uso do modelo híbrido com uma rede MLP como combinadora. As Figuras 4.10 a 4.12 exibem a previsão em relação à série real.

Para um funcionamento ideal, diversos parâmetros de configurações precisam ser trabalhados. Por questões de simplicidade e limitação de tempo, este trabalho de graduação se restringiu ao modelo padrão implementado na biblioteca *SciKit Learn*, para um *MLPRegressor*, em linguagem de programação *Python*.

A rede MLP neste experimento tem somente uma camada escondida, e para determinar a quantidade de nós necessários, foi realizada uma busca incremental. Portanto seja N_h a quantidade de neurônios na camada escondida tal que $N_h = 3, 4, \ldots, 200$. A Tabela 4.5 exibe a estrutura das redes MLPs para cada uma das séries

Tabela 4.5 Parâmetros da rede combinadora MLP no modelo MLP(ARIMA, RBFNN).

Série	Nós de entrada	Nós da camada escondida	Nós de saída
LY	2	178	1
SU	2	196	1
BU	2	64	1

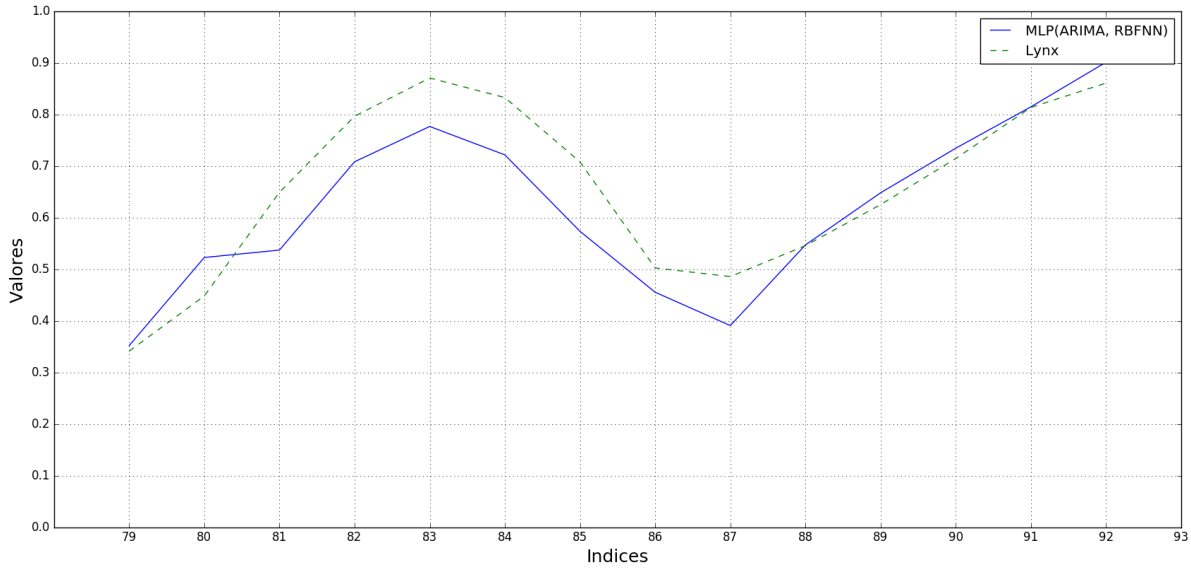

Figura 4.10 Previsão da série *Lynx* através do modelo híbrido MLP(ARIMA, RBFNN) relação à série original.

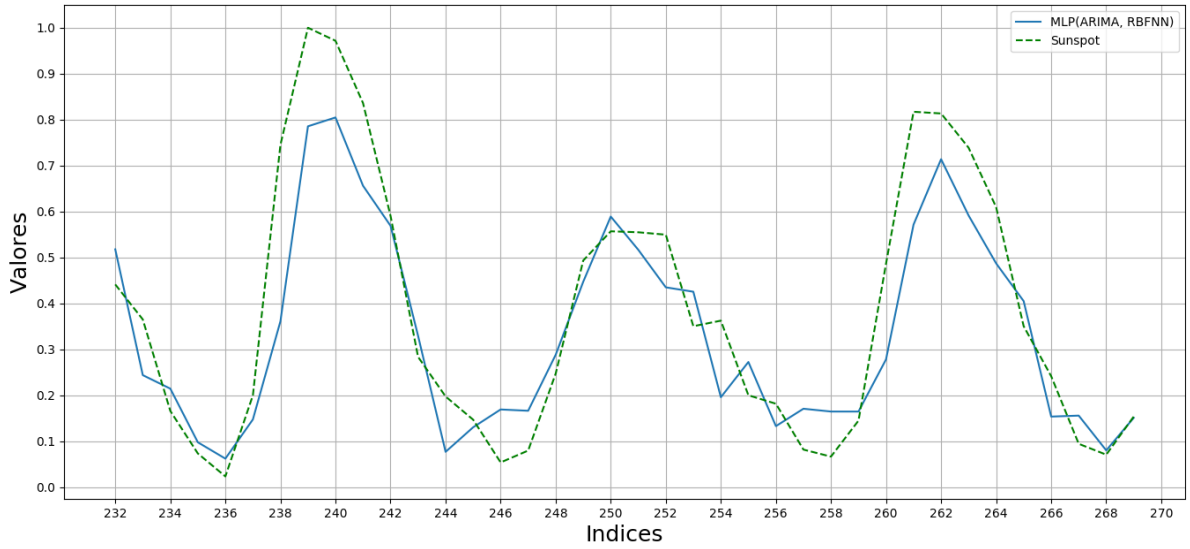

Figura 4.11 Previsão da série *Sunspot* através do modelo híbrido MLP(ARIMA, RBFNN) relação à série original.

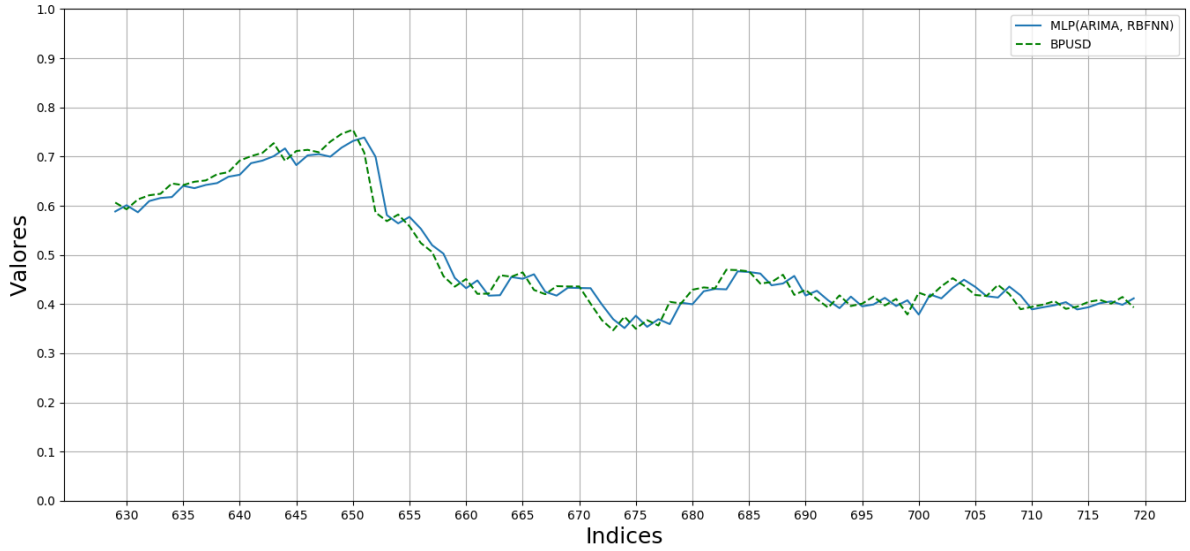

Figura 4.12 Previsão da série *BPUSD* através do modelo híbrido MLP(ARIMA, RBFNN) relação à série original.

4.5 Resultados do experimento 5

Para cada uma das séries analisadas neste experimento, a RBFNN combinadora terá um única camada de entrada, com dois neurônios de entrada (um para o resultado vindo do modelo

ARIMA, outro para o resultado fornecido pela previsão do erro do ARIMA com a primeira RBFNN), uma camada escondida, e uma camada de saída, esta última com apenas nó. No caso da camada escondida, a quantidade de nós (centros) precisa ser identificada. Para isso, foi realizada uma busca incremental variando-se a quantidade de centros de N_c até uma quantidade máxima N_{c-max}, para cada uma das séries.

As Figuras 4.13 a 4.15 exibem a previsão em relação à série real.

A estrutura da rede usada para prever o erro do ARIMA está especificada na Tabela 4.4, para cada uma das séries. Para a RBFNN combinadora, a especificação dos parâmentros das redes estão descritas na Tabela 4.6.

Tabela 4.6 Parâmetros da rede RBFNN para combinação das previões do ARIMA e da RBFNN para o erro do ARIMA.

Série	**Nós de entrada**	**Nós da camada escondida**	**Nós de saída**	N_{c-max}
LY	2	30	1	40
SU	2	11	1	125
BU	2	25	1	400

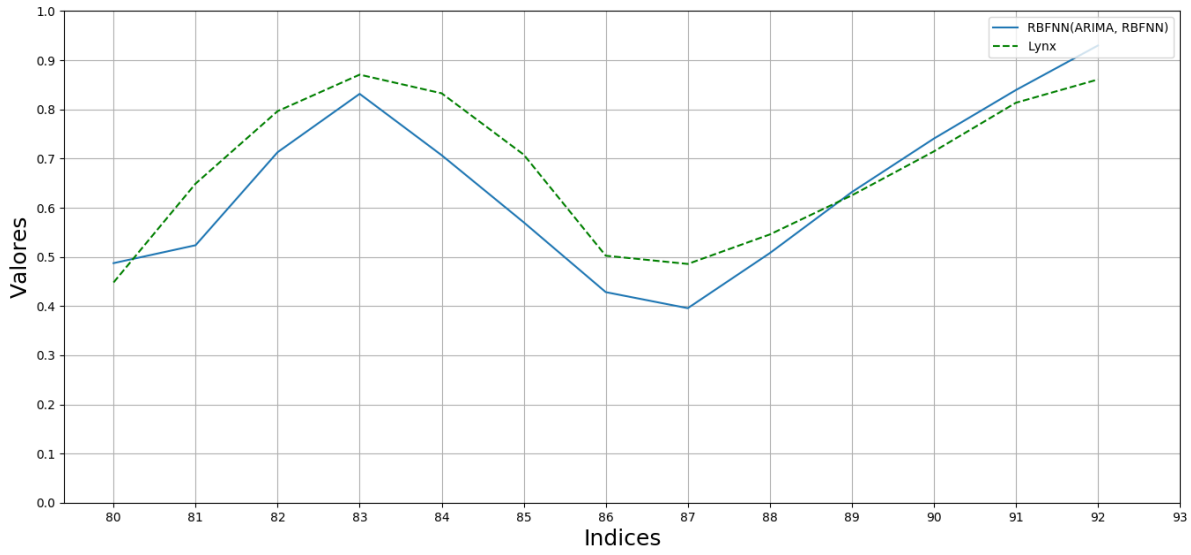

Figura 4.13 Previsão da série *Lynx* através do modelo híbrido RBFNN(ARIMA, RBFNN) relação à série original.

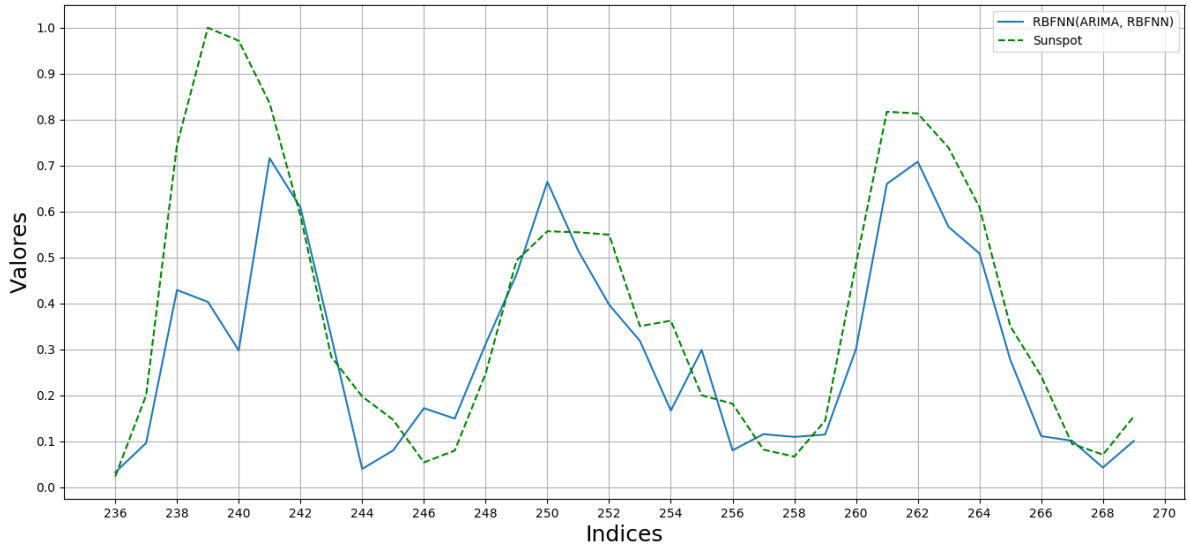

Figura 4.14 Previsão da série *Sunspot* através do modelo híbrido RBFNN(ARIMA, RBFNN) relação à série original.

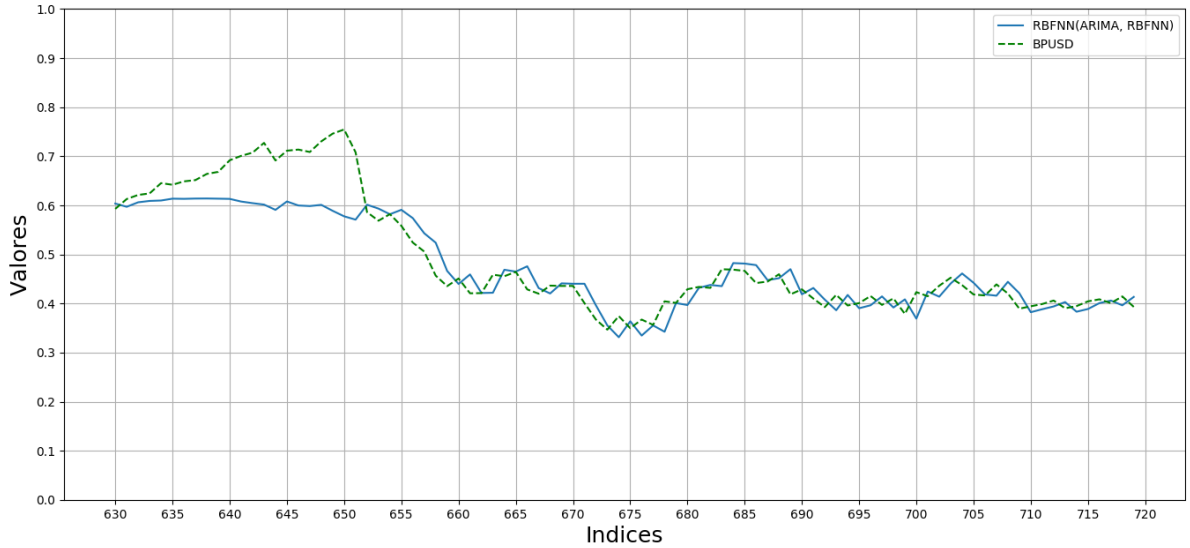

Figura 4.15 Previsão da série *BPUSD* através do modelo híbrido RBFNN(ARIMA, RBFNN) relação à série original.

CAPÍTULO 5

Conclusões e trabalhos futuros

Com base nos resultados obtidos e nos trabalhos de [5], [9], pode-se concluir que redes de função de base radial, embora sejam capazes de aproximar uma função não-linear, não foram capazes de caracterizar com sucesso a correlação não-linear presente nos resíduos do modelo ARIMA. Além disso, não se pode concluir sobre seu papel como combinadora uma vez que, nem mesmo o modelo MLP(ARIMA, RBFNN) obteve êxito. Os resultados indicam porém, que a quantidade de entradas da série influencia na precisão do modelo híbrido.

Os modelos utilizados neste trabalho são de extenso conhecimento da comunidade científica já a algumas décadas, de modo que, cada um deles já foi amplamente estudado. Embora as Redes Neurais de função de base radial e o ARIMA necessitem de poucos parâmetros para sua completa especificação, o modelo de rede MLP possui uma considerável quantidade de parâmetros de configuração. Isoladamente, cada um desses modelos podem ter suas configurações selecionadas manualmente de maneira satisfatória. Porém, quando combinados, constituem um conjunto extenso de parâmetros para a correta especificação do modelo híbrido. A modificação de apenas um deles pode ter grande impacto no desempenho final do modelo. Para tal, técnicas de otimização podem ser aplicadas. Algoritmos bioinspirados estabelecem, si próprios, um extenso campo de estudo e já há, na literatura, trabalhos com a proposta de uso desses algoritmos para especificação de parâmetros para modelos [9]. Tais algoritmos oferecem uma alternativa mais eficiente para selecionar a melhor configuração do modelo híbrido aqui idealizado. Por outro lado, eles também adicionam complexidade à solução. De fato, o autor investigou previamente a implementação desses algoritmos, mais especificamente Algoritmos Genéticos, na otimização dos parâmetros das redes RBF, motivado pela experiência prévia com uso de Estratégias Evolutivas para treinamento de redes MLP, que estas por sua vez, atuariam como decisor sobre o jogo de Flappy Birds. Entretanto, devido à complexidade e ao tempo disponível para a finalização deste trabalho, o autor achou prudente não seguir por esse caminho. Entretanto ainda é uma modificação plausível.

Outro ponto a ser considerado como melhoria em trabalhos futuros, seria a utilização de redes MLP para predição do erro do ARIMA, bem como o uso de séries temporais com um maior conjunto de dados.

Referências Bibliográficas

[1] P. A. Morettin and C. M. de Castro Toloi, *Modelos para Previsão de Séries Temporais*. Instituto de matemática pura e aplicada, 1981.

[2] R. H. Shumway and D. S. Stoffer, *Time Series Analysis and Its Applications With R Examples*. Springer, 2006.

[3] G. Zhang, B. E. Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks: The state of the art," *International Journal of Forecasting 14*, pp. 35–62, 1998.

[4] F. Zheng and S. Zhong, "Time series forecasting using a hybrid rbf neural network and ar model based on binomial smoothing," *International Journal of Mathematical and Computational Sciences*, vol. 5, no. 3, pp. 419–423, 2011.

[5] P. Zhang, "Time series forecasting using a hybrid arima and neural network model," *Neurocomputing*, no. 50, pp. 159–175, 2003.

[6] D. C. Montgomery, C. L. Jennings, and M. Kulahci, *Introduction to Time Series Analysis and Forecasting*. Wiley-Interscience, 2008.

[7] J. D. Cryer and K. Chan, *Time Series Analysis with Applications in R*. Spring, 2 ed., 2008.

[8] P. S. P. Cowpertwait and A. V. Metcalfe, *Introductory Time Series with R*. Spring, 2009.

[9] P. S. de Mattos Neto, G. D. Cavalcanti, and F. Madeiro, "Nonlinear combination method of forecasters applied to pm time series," *Pattern Recognition Letters*, Nov. 2016.

[10] R. Beale and T. Jackson, *Neural Computing: An Introduction*. Adam Hilger, 1991. Department of Computer Science, New York University.

[11] D. S. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Systems*, vol. 2, no. 3, pp. 321–355, 1988.

[12] S. Haykin, *Neural Networks: A comprehensive foundation*. Pearson Education, 2 ed., 1999.

[13] N. Benoudjit and M. Verleysen, "On the kernel widths in radial-basis function networks," *Neural Processing Letters*, no. 18, pp. 139–154, 2003.

[14] P. R. A. Firmino, P. S. de Mattos Neto, and T. A. Ferreira, "Error modeling approach to improve time series forecasters," *Neurocomputing*, no. 153, pp. 242–254, 2015.

Este volume foi tipografado em LaTeX na classe UFPEThesis (www.cin.ufpe.br/~paguso/ufpethesis).