UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE GEOCIÊNCIAS
PROGRAMA DE PÓS-GRADUAÇÃO EM GESTÃO DE RISCOS E DESASTRES NATURAIS NA AMAZÔNIA

DISSERTAÇÃO DE MESTRADO

VICTOR DA CRUZ PERES

SISTEMA HIDROLOGICO PARA PREVISÃO DE RISCO NA AMAZÔNIA UTILIZANDO REDES NEURAIS ARTIFICIAIS

Belém – PA
2019

VICTOR DA CRUZ PERES

SISTEMA HIDROLOGICO PARA PREVISÃO DE RISCO NA AMAZÔNIA UTILIZANDO REDES NEURAIS ARTIFICIAIS

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Gestão de Risco e Desastres Naturais na Amazônia, do Instituto de Geociências da Universidade Federal do Pará, como requisito parcial à obtenção do título de Mestre em Gestão de Riscos e Desastres Naturais na Amazônia.

Área de Concentração: Minimização de Riscos e Mitigação de Desastres Naturais na Amazônia.

Linha de Pesquisa: Ameaças Naturais no Ambiente Amazônico.

Orientador: Dr. Edson José Paulino da Rocha.

Belém – PA
2019

Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD
Sistema de Bibliotecas da Universidade Federal do Pará
Gerada automaticamente pelo módulo Ficat, mediante os dados fornecidos pelo(a) autor(a)

P434s Peres, Victor da Cruz.
Sistema hidrológico para previsão de risco na Amazônia utilizando redes neurais artificiais / Victor da Cruz Peres. — 2019.
126 f. : il. color.

Orientador(a): Prof. Dr. Edson José Paulino da Rocha
Dissertação (Mestrado) - Programa de Pós-Graduação em Gestão de Risco e Desastre na Amazônia, Instituto de Geociências, Universidade Federal do Pará, Belém, 2019.

1. Previsão Hidrológica. 2. Modelagem Hidrológica. 3. Redes Neurais Artificiais. 4. Sistemas de alerta. 5. Amazônia. I. Título.

CDD 551.480285

VICTOR DA CRUZ PERES

SISTEMA HIDROLOGICO PARA PREVISÃO DE RISCO NA AMAZÔNIA UTILIZANDO REDES NEURAIS ARTIFICIAIS

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Gestão de Risco e Desastres Naturais na Amazônia, do Instituto de Geociências da Universidade Federal do Pará, como requisito parcial à obtenção do título de Mestre em Gestão de Riscos e Desastres Naturais na Amazônia.

Data de aprovação: _____/_____/_____
Banca Examinadora:

_____
Prof. Edson José Paulino da Rocha - Orientador
Doutorado em Meteorologia
Universidade Federal do Pará

_____
Prof. João Batista Miranda Ribeiro – Membro Interno
Doutorado em Ciências da Engenharia Ambiental
Universidade Federal do Pará

_____
Prof. Francisco de Souza Oliveira – Membro Interno
Doutorado em Geofísica
Universidade Federal do Pará

_____
Profª. Eliane de Castro Coutinho – Membro Externo
Doutorado em Ciências Ambientais
Universidade do Estado do Pará

Ao meu pai João e ao meu tio Augusto que sempre acreditaram em mim.
Espero nunca decepcioná-los.

AGRADECIMENTOS

A **Deus**, que iluminou meu longo caminho, mostrando as melhores soluções para tantos problemas difíceis que se apresentaram durante a realização deste trabalho.

Ao meu orientador Professor **Edson José Paulino da Rocha**, pela oportunidade de elaboração e acompanhamento deste trabalho, me proporcionando melhores oportunidades na vida acadêmica e profissional.

Aos membros da banca que se prontificaram a participar, opinar e melhorar o trabalho.

A minha namorada **Shaolyne Costa**, por todo seu amor, carinho, apoio e principalmente compreensão em tantos momentos difíceis nesta caminhada tão longa e difícil. Obrigado por tudo!

Aos meus pais, **João Bosco Rodrigues Peres (*in memorian*) e Maria do Socorro da Cruz Peres**, por toda a educação que me proporcionaram durante minha vida. A conclusão deste trabalho é mais uma vitória que eu conquistei com o apoio deles.

Aos meus familiares, especialmente, meu irmão **João Bosco Rodrigues Peres Junior**, pelo suporte a mim oferecidos. Sempre fui muito feliz por vos ter ao meu lado, sei que tudo o que fazem é porque querem sempre o melhor para mim.

A todos os funcionários do **CENSIPAM**, especialmente ao **Altieri** com a ajuda no fornecimento dos dados para treinamento da RNA.

Ao programa de Mestrado em Gestão de Rico e Desastre na Amazônia do Instituto de Geociências na Universidade Federal do Pará.

A todos os funcionários do **Instituto Federal do Pará – Campus Óbidos**, especialmente os professores **Fabrício, Natanael e Enéias**, pelo companheirismo, lealdade e amizade.

E a todos que contribuíram direta ou indiretamente para a conclusão deste trabalho.

“Os dias prósperos não vêm por acaso; nascem de muita fadiga e persistência.”
Henry Ford

RESUMO

A estimativa do comportamento futuro dos níveis de uma bacia hidrográfica é fundamental para a elaboração do plano de gerenciamento dos seus recursos hídricos. O objetivo desta pesquisa foi modelar a relação entre chuva e nível através de uma técnica conhecida por redes neurais artificiais (RNA). As RNA são modelos empíricos com funcionamento semelhante ao funcionamento do cérebro humano. Nesta pesquisa foi avaliada a capacidade das RNA modelarem o processo chuva-nível em base diário. Foi considerado durante o treinamento das RNA as influências da arquitetura da rede, da inicialização dos pesos e da extensão das séries de dados. As cinco RNA que produziram os melhores resultados foram confrontados com os resultados observados. Os resultados foram muito satisfatórios. Findando em um sistema de alerta de seca e cheia em Itaituba/Pa.

Palavras-chave: Previsão Hidrológica. Modelagem Hidrológica. Redes Neurais Artificiais. Sistema de Alerta. Amazônia.

ABSTRACT

The estimation of the future behavior of the levels of a river basin is fundamental for the elaboration of the plan of management of its water resources. The objective of this research was to model the relationship between rainfall and level through a technique known as artificial neural networks (RNA). RNAs are empirical models with functions similar to the functioning of the human brain. In this research, the ability of RNA to model the rain-level process on a daily basis was evaluated. Influences of network architecture, initialization of weights, and extension of data series were considered during RNA training. The five RNAs that produced the best results were confronted with the observed results. The results were very satisfactory. Finding in a dry and full alert system in Itaituba-Pa.

Key words: Hydrological Forecasts. Hydrologic Modelling. Artificial Neural Networks. Alert System. Amazon.

LISTA DE ILUSTRAÇÕES

Figura 1- Neurônio de McCulloch e Pitts. 19
Figura 2- Modelo Geral de Neurônio Artificial. 19
Figura 3- Função de Ativação Linear. 22
Figura 4- Função de Ativação Sigmoide. 23
Figura 5- Função de Ativação Tangente Hiperbólica. 24
Figura 6- Aprendizado Supervisionado 25
Figura 7- Aprendizado Não-Supervisionado. 26
Figura 8- Arquitetura de Rede Direta de Múltiplas Camadas. 28
Figura 9- Fases de uma RNA Multicamadas com Algoritmo *Backpropagation* 29
Figura 10- Mapa de Localização da Área de Estudo da Bacia Hidrográfica do Rio Tapajós. 41
Figura 11- Região da área da pesquisa com a divisão das áreas de influência dos pluviômetros segundo o método de Thiessen elaborado com o auxílio do software ArcGis 9.1. 43
Figura 12- Casos de USO do sistema. 48
Figura 13- Modelo de Classes do sistema. 49
Figura 14- Classes do sistema de alerta. 50
Figura 15- Trecho do código do cálculo da área em JAVA do sistema. 50
Figura 16- Utilização do MVC no sistema. 51
Figura 17- Hipsometria da sub-região TAP-01. 54
Figura 18- Hipsometria da sub-região TAP-02. 54
Figura 19- Hipsometria da sub-região TAP-03. 55
Figura 20- Hipsometria da sub-região TAP-04. 56
Figura 21- Hipsometria da sub-região TAP-05. 56
Figura 22- Hipsometria da sub-região TAP-06. 57
Figura 23- Hipsometria da sub-região TAP-07. 58
Figura 24- Hipsometria da sub-região TAP-08. 59
Figura 25- Estrutura do cálculo do Tempo de Concentração Real. 60
Figura 26- Gráfico dos níveis observado e calculado utilizando arquiteturas de RNA 9-5-1, 9-7-1, 9-4-3-1 e 9-4-2-1. 62

Figura 27- Taxa de erro por iterações nas camadas 9-5-1, 9-7-1, 9-4-3-1 e 9-4-2-1. 62
Figura 28- Taxa de erro por iterações nas camadas 9-4-1. 63
Figura 29- Gráfico dos níveis observado e calculado utilizando arquiteturas de RNA 9-4-1. 64
Figura 30- Página do sistema de alerta.....65
Figura 31- Tela inicial ao selecionar a opção Diário. 66
Figura 32- Tela ao selecionar uma Data. 67
Figura 33- Tela ao clicar no botão Carregar. 67
Figura 34- Tela ao clicar no botão Carregar. 68
Figura 35- Tela ao clicar no botão Calcular. 69
Figura 36- Tela ao clicar no botão Calcular. 70
Figura 37- Tela da opção Mensal. 70
Figura 38- Tela da opção Mensal. 71
Figura 39- Tela da opção Anual. 71
Figura 40- Tela da opção Anual. 72
Figura 41- Tela da opção Anual. 72
Figura 42- Tela da opção Anual. 73
Figura 43- Tela da opção Anual. 73

LISTA DE TABELAS

Tabela 1- Tempo de concentração nas sub-bacias.....59
Tabela 2- Tempo de concentração nas sub-bacias até Itaituba/PA. 60

LISTA DE ABREVIATURAS E ACRÔNIMOS

PCD	Plataforma de Coleta de Dados
RNA	Redes Neurais Artificiais
MLP	Redes Perceptron Multicamadas
Backpropagation	Retropropagação
tcr	Tempo de Concentração Real
CPC/NCEP	Centro de Previsão do Clima
tc	Tempo de Concentração
UML	Linguagem de Modelagem Unificada
MVC	*Model-View-Controller*

SUMÁRIO

1 INTRODUÇÃO 15

2. OBJETIVOS 17

2.1 Objetivo geral **17**

2.2 Objetivos específicos **17**

3 REFERÊNCIAL TEÓRICO 18

3.1 Redes neurais artificiais **18**

3.1.1 Função de ativação 21

3.1.2 Função linear 21

3.1.3. Função sigmoide 22

3.1.4 Função tangente hiperbólica 23

3.2 Algoritmo de aprendizado **24**

3.2.1 Aprendizado supervisionado 25

3.2.2 Aprendizado não-supervisionado 26

3.3 Redes perceptron multicamadas **27**

3.4 Algoritmo de retropropagação do erro **29**

3.5 Aplicação de rna em recursos hídricos **32**

4 MATERIAIS E MÉTODOS 40

4.1 Área da pesquisa **40**

4.2 Dados hidrometeorologicos **41**

4.2.1 Divisão da região em sub-bacias 42

4.2.2 Cálculo do tempo de concentração 43

4.2.3 Função do cálculo de concentração 44

4.2.4 Retirada dos dados das precipitações nas sub-bacias 45

4.3 Treinamento e teste **46**

4.4 Sistema de alerta **47**

4.4.1 Planejamento e projeto do sistema 47

4.4.2 Classificação do nível da bacia 49

4.4.3 Implementação do sistema 50

5 RESULTADOS E DISCUSSÃO 53

5.1 Caracterização da bacia 53
5.1.1 Tempo de concentração 59
5.2 Resultados obtidos com a rede MLP 61
5.3 O sistema 65
6 CONCLUSÃO 74
REFERÊNCIAS 76
APÊNDICE A - ALGORITMO REDES NEURAIS ARTIFICIAIS 81
APÊNDICE B - ALGORITMO DE TREINAMENTO BACKPROPAGATION 87
APÊNDICE C - CÓDIGO FONTE: INDEX.XHTML 95
APÊNDICE D - CÓDIGO FONTE: CONTROLADOR.JAVA 122

1 INTRODUÇÃO

A previsão de nível de um sistema hídrico é de grande importância em todo mundo, principalmente na região Amazônica que possui a mais extensa rede hidrográfica do planeta, com 25.000 quilômetros de rios (ANA, 2011), que são usados para transporte de pessoas e mercadorias, irrigação, fornecimento de água, turismo, geração de energia elétrica, além de servir de subsistência para as comunidades ribeirinhas.

Sendo assim, para reduzir os impactos provenientes da incerteza da previsão hidrológica e que na Amazônia muita das vezes podem chegar a níveis catastróficos, diversos métodos de mensuração de vazão vêm sendo utilizados, como por exemplo: por meio de régua, plataforma de coleta de dados (PCD). No entanto, estes métodos podem conter erros de leitura ou de processamento, devido a isso muitos trabalhos como o de (OLIVEIRA; MONTINI; BERGMANN, 2007), entre outros citados neste trabalho, estão utilizando a informação da precipitação que ocorreu em uma determinada bacia ou a concatenação de diversas bacias. Dessa forma, as evoluções no conhecimento da previsão de vazões vêm contribuindo para que o governo tenha mais controle sobre o ciclo hidrológico para tomada de decisão (FERNÁNDEZ BOU; SÁ; CATALDI, 2015).

Na previsão usa-se modelos hidrológicos que podem ser definidos por meio da Inteligência computacional que consistem em um aglomerado de metodologias computacionais inspirada na observação da natureza, como Algoritmos Genéticos, Colônia de Formiga, Lógica Fuzzy, Redes Neurais Artificiais (RNA). Essas técnicas computacionais resolvem problemas de modelagem de sistemas não-lineares. A RNA é uma estrutura matemática não-linear que é capaz de representar processos não-lineares que relacionam entradas e saídas de um sistema (HAYKIN, 2001). Elas permitem modelar processos que envolvem séries temporais de sinais de entrada e saída que apresentem algum teor de complexidade em muitos campos da pesquisa e do desenvolvimento científico (VEMURI, 1994). Desta forma, o sucesso das RNA em muitos campos da ciência e da engenharia surgiu a aplicabilidade na hidrologia.

Assim, o principal atrativo de utilizar a RNA em qualquer área da ciência é a capacidade que ela possui de aprender com exemplos e de generalizar informações.

A generalização, por sua vez, está relacionada com a capacidade da rede aprender por meio de um conjunto de dados a partir de experiências e ulteriormente dar resposta correta para dados não conhecidos, é uma demonstração de que a capacidade da RNA vai muito além de mapear relação de entradas e saídas. Elas são capazes de extrair informações não relacionadas de forma explícita no sistema.

Com isto, o problema desta pesquisa consiste em definir quais são os parâmetros de uma RNA que possa predizer o nível da bacia do Rio Tapajós utilizando como entrada dados de precipitação da região.

2 OBJETIVOS

2.1 Objetivo geral

Criar um sistema de alerta hidrológico a partir da análise do processamento de dados de precipitação utilizando Redes Neurais Artificiais no município de Itaituba/Pa.

2.2 Objetivos específicos

Os objetivos específicos do trabalho são:

a) Caracterizar a bacia hidrográfica;
b) Desenvolver um modelo Chuva-Nível baseado em RNA.

3 REFERÊNCIAL TEÓRICO

A proposta abordada no presente trabalho surge no sentido também de atender a linha de pesquisa do curso de mestrado em questão e as necessidades ligadas a Minimização de Riscos e Mitigação de Desastres Naturais na Amazônia, com sua aplicação na prática, assim se faz importante uma abordagem mesmo que sucinta de termos condizentes com os desastres naturais, como gestão, planejamento, gerenciamento e principalmente monitoramento.

3.1 Redes neurais artificiais

O cérebro humano funciona de forma complexa, não linear e paralela, na qual aproximadamente 100 bilhões de neurônios estão sempre recebendo e processando informações do meio externo, tornando-se aptos a tomar decisões (LENT, 2010).

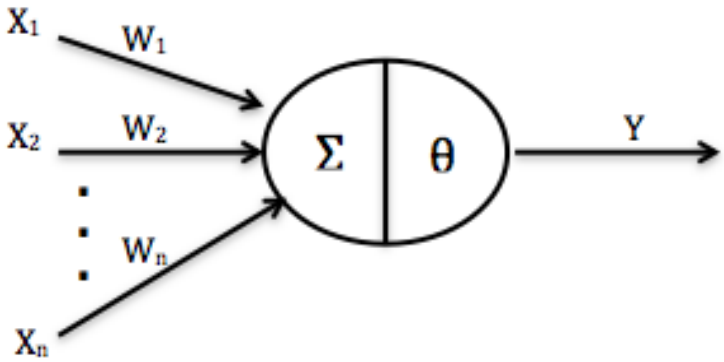
A partir do ponto de vista conceitual, o cérebro humano, como um modelo de dispositivo de computação paralela, surge a base para o estudo de RNA, formada por neurônios e um conjunto de conexões entre eles, em que o funcionamento se baseia na aquisição de conhecimento, aprendizado e tomada de decisão baseada nessa aprendizagem (ZILOUCHIAN; JAMSHIDI, 2001).

A abordagem das RNA consiste em apreender os princípios básicos de manipulação de informação do cérebro humano e pôr esse conhecimento na resolução de problemas que exigem aprendizado a partir de experiência (BEALE; JACKSON, 1990).

O primeiro modelo de neurônio artificial foi apresentado por McCulloch e Pitts (MCCULLOCH; PITTS, 1943), onde consiste em um modelo matemática com **n** terminais de entradas (que representam os dendritos) e apenas um terminal de saída **y** (representado pelo axônio).

Este modelo, apresentado na Figura 1, elucida o funcionamento do neurônio com um circuito binário. Para imitar o comportamento das sinapses, os neurônios possuem pesos acoplados os quais podem ser positivos ou negativos, de formar similar ao complexo modelo neural, dependendo das sinapses correspondentes serem inibitórias ou excitatórias.

Figura 1- Neurônio de McCulloch e Pitts.



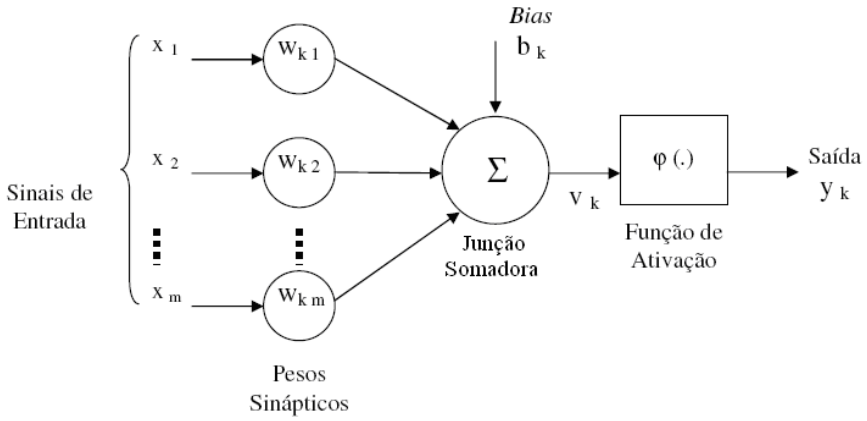
Fonte: Adaptado de (MCCULLOCH; PITTS, 1943).

Em 1958, Frank Rosenblatt apresentou em seu trabalho o conceito de aprendizado em Redes Neurais (ROSENBLATT, 1958). O modelo proposto por ele ficou conhecido como Perceptron, era composto de rede tendo como unidades básicas nodos MCP e de uma regra de aprendizado.

Alguns anos mais tarde, Rosenblatt demonstrou o teorema de convergência do perceptron, o qual mostra que um nodo MCP treinado com o algoritmo de aprendizado do perceptron sempre converge, caso o problema em questão seja linearmente separável (ROSENBLATT, 1962).

A topologia original descrita por Rosenblatt (1962) era composta por unidades de entrada e por um nível de saída formado pelas unidades de resposta conforme a Figura 2.

Figura 2- Modelo Geral de Neurônio Artificial.



Fonte: Adaptado de Haykin (2001).

Os principais componentes identificados neste modelo são:

- Um conjunto de conexões sinápticas, cada uma delas associada a um peso sináptico (w_{k1}, w_{k2}, ..., w_{km});
- A junção somadora (Σ), que soma todos os sinais de entrada multiplicados pelos respectivos pesos;
- A função de ativação (φ(.)), geralmente não-linear, tem como argumento a soma ponderada dos sinais de entrada. Esta função limita a amplitude de saída do neurônio no intervalo normalizado de [0,1] ou [-1,1];
- As bias (b_k), tem o papel de aumentar ou diminuir a influência do valor das entradas na rede.

Matematicamente, considerando o bias (b_k) com entrada de valor fixo x_0= +1 e peso w_{k0}=b_k a saída para o k-ésimo neurônio pode ser expressa pela Equação 3.1.

$$y_k = f\left(v_k\right) = f\left(\sum_{j=1}^{m} w_{kj} x_j\right) \tag{3.1}$$

Em que:

m → número de entradas do neurônio k;
j = 1, 2, ..., m;
x_j → *j-ésima* entrada do neurônio k;
w_{kj} → peso atribuído à *j-ésima* entrada do neurônio k;
v_k → variável de entrada (ou potencial de ativação) da função de ativação do neurônio k;
y_k → saída do neurônio.

O somatório de todas as entradas ponderadas por seus respectivos pesos sinápticos é denominado *net*, como pode ser observado na Equação 3.2.

$$y_k = \varphi\left(v_k\right) = \varphi\left(\sum_{j=1}^{m} w_{kj} x_j\right) = \varphi\left(net_k\right) \tag{3.2}$$

3.1.1 Função de ativação

O modelo de cada unidade da rede pode incluir uma não-linearidade na sua saída. É importante enfatizar que a não-linearidade deve ser suave, diferentemente da função sinal utilizada pelo perceptron original.

A função de ativação representa o efeito que a entrada interna e o estado atual de ativação exercem na definição do próximo estado de ativação da unidade. Quando propriedades dinâmicas estão envolvidas na definição do estado de ativação, equações diferenciais ou a diferenças são empregadas. Tendo em vista a simplicidade desejada para as unidades processadoras, geralmente define-se seu estado de ativação como uma função algébrica da entrada interna atual, independente de valores passados do estado de ativação ou mesmo da entrada interna. Geralmente, esta função é monotonamente não-decrescente e apresenta um tipo de não-linearidade associada ao efeito da saturação. A seguir são descritos alguns tipos de função de ativação empregados na literatura para as arquiteturas do tipo MLP (HAYKIN, 2001).

3.1.2 Função linear

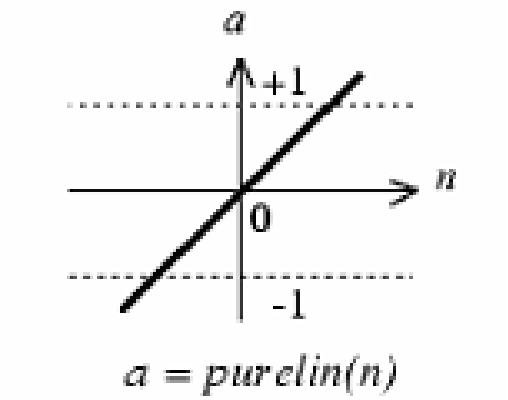
A função de ativação Linear mostrada graficamente na figura 3 é representada matematicamente pela Equação 3.3.

$$\varphi(v) = av \tag{3.3}$$

Em que:

$a \rightarrow$ é um número real que define a saída linear para os valores de entrada;
$v \rightarrow$ é o valor de ativação do neurônio.

Figura 3- Função de Ativação Linear.



Fonte: Adaptado de (DEMUTH; BEALE; HAGAN, 2000).

3.1.3 Função sigmoide

A função Sigmoide (Equação 3.4) trata-se de uma função monotônica crescente que apresenta propriedades assintóticas e de suavidade que as fazem bastante utilizadas em RNA (HAYKIN, 2001). A Figura 4 apresenta uma função Sigmoide com as saídas no intervalo [0,1].

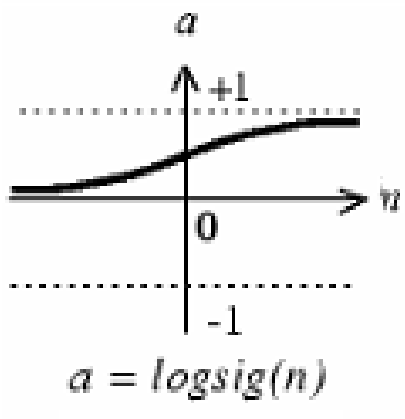
$$\varphi(v) = \frac{1}{1 + e^{(-av)}} \tag{3.4}$$

Em que,

$a \rightarrow$ é o parâmetro de inclinação da função;

$v \rightarrow$ é o valor de ativação do neurônio.

Figura 4- Função de Ativação Sigmoide.



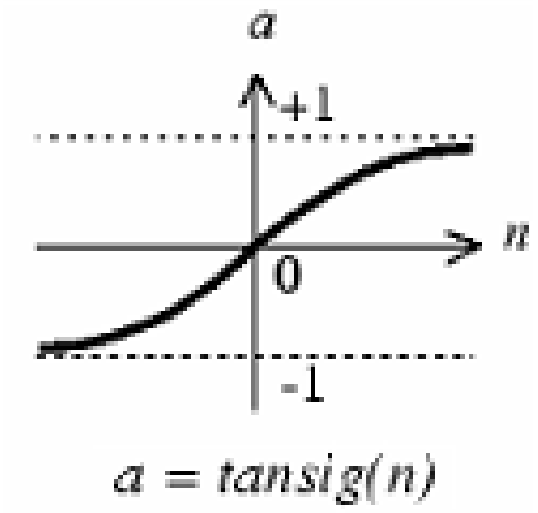
Fonte: Adaptado de (DEMUTH; BEALE; HAGAN, 2000).

3.1.4 Função tangente hiperbólica

Em muitas aplicações é interessante que a saída da função sigmoidal esteja entre [-1, 1]. Nestes casos utiliza-se a função tangente hiperbólica, Figura 5, representada matematicamente pela Equação 3.5.

$$\varphi(v) = \frac{1 - e^{-av}}{1 + e^{(-av)}} \tag{3.5}$$

Figura 5- Função de Ativação Tangente Hiperbólica.



Fonte: Adaptado de Demuth, Beagle e Hagan (2000).

3.2 Algoritmo de aprendizado

Um dos principais atrativos que as RNA possuem é a capacidade de aprender por exemplos e fazer interpolações e extrapolações do que aprendem.

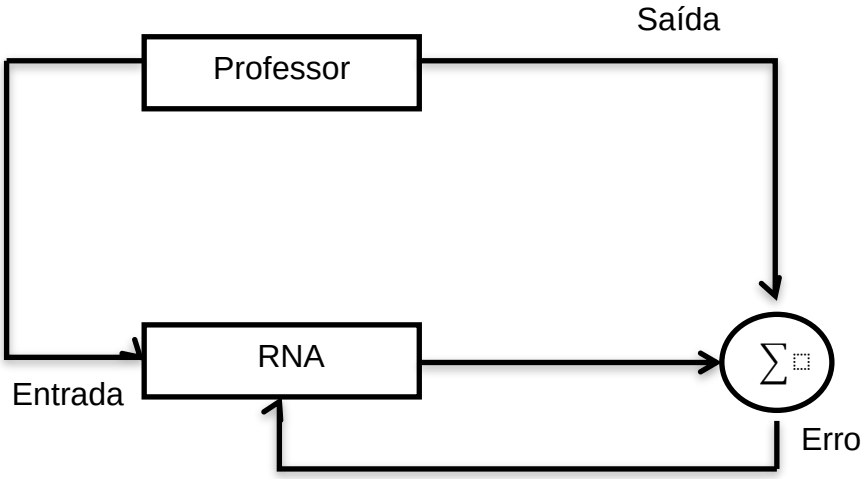
Algoritmo de aprendizado se dá a um conjunto de procedimentos bem definidos para adaptar os parâmetros para que a mesma possa aprender uma determinada função. Não há um único algoritmo de aprendizado, o que há é um conjunto de ferramentas representadas por diversos algoritmos, no qual diferem basicamente pela maneira como os pesos sinápticos da rede são ajustados, cada qual com suas vantagens e desvantagens.

Diversos métodos para treinamentos foram criados, dentre eles podemos destacar em dois paradigmas principais: aprendizado supervisionado e aprendizado não supervisionado (BRAGA; PONCE; TERESA, 2007).

3.2.1 Aprendizado supervisionado

Este método é o mais comum no treinamento das RNA, sendo chamado aprendizado supervisionado pelo fato de a entrada e saída desejadas para a rede serem fornecidas por um supervisor (professor) externo. A Figura 6 ilustra o mecanismo desse método.

Figura 6- Aprendizado Supervisionado



Fonte: Adaptado de Braga (2007).

O processo de aprendizado supervisionado consiste em o professor indicar a rede neural uma resposta desejada a uma determinada entrada. A rede tem sua saída corrente (calculada) então é feita uma comparação com a saída desejada, sendo feito ajuste nos pesos minimizando os erros a cada iteração.

Os exemplos mais conhecidos de algoritmos de aprendizado supervisionado são a regra delta e a sua generalização para redes de múltiplas camadas, o algoritmo *backpropagation* (WINDROW, B.; HOFF, 1960) e (RUMELHART; HINTON; WILLIAMS, 1986) que serão abordados posteriormente neste trabalho.

O cálculo do erro **e**(t) é a diferença entre a saída desejada **d**(t) e a resposta calculada **y**(t) no instante t, conforme a Equação 3.6.

$$e(t) = d(t) - y(t) \tag{3.6}$$

Depois de calculado o erro, as correções dos pesos sinápticos são realizadas pela equação genérica a seguir (Equação 3.7).

$$w_i(t+1) = w_i(t) + \eta e(t) x_i(t) \tag{3.7}$$

Em que, η é a taxa de aprendizado e $x_i(t)$ é a entrada do neurônio *i* para o tempo *t.* Ou seja, o ajuste dos pesos deve ser a proporcional ao produto do erro pelo valor de entrada da sinapse naquele instante de tempo.

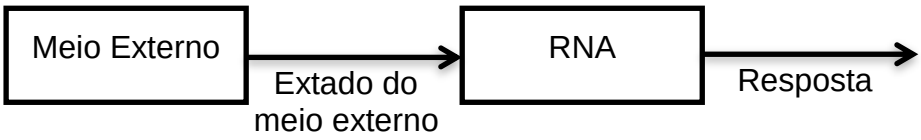
A desvantagem do aprendizado supervisionado é que na ausência do professor, para informar a saída desejada, a rede não conseguirá aprender novas estratégias.

3.2.2 Aprendizado não-supervisionado

No Aprendizado Não-Supervisionado, como o próprio nome sugere, não possui um professor para fazer o processo de aprendizagem. Este método está ilustrado na Figura 7.

Nestes tipos de algoritmos, são repassados à rede neural apenas as entradas, devido não conter os valores das saídas. Esse processo só é possível quando ocorre a redundância dos dados de entradas, com a redundância é possível fazer o reconhecimento de padrões nos dados de entradas.

Figura 7- Aprendizado Não-Supervisionado.



Fonte: adaptado de Braga (2007).

Existem vários métodos para a implementação do aprendizado não-supervisionado são: Modelo de Linsker, Regra de Oja, Regra de Yuille e Aprendizado por Competição. Por motivo deste trabalho fazer uso de aprendizado supervisionado, estes métodos não serão detalhados.

3.3 Redes perceptron multicamadas

Conforme foi abordado na seção anterior, as redes de uma só camada, *Perceptron*, resolvem apenas problemas linearmente separáveis. A solução para problemas não linearmente separáveis é resolvida com a utilização de redes com uma ou mais camadas intermediárias denominadas Redes Perceptron Multicamadas (MLP do inglês *Multi Layer Perceptron*).

Segundo (CYBENKO, 1989), uma rede MLP com apenas uma camada intermediária pode implementar qualquer função contínua. Se forem utilizadas duas camadas intermediárias permitirá a aproximação de qualquer função.

Na rede Perceptron multicamadas a permissão da implementação de uma função não implica na garantia da implementação da mesma. Dependendo da distribuição do conjunto de dados e das disposições dos seus pesos, a rede pode convergir para um mínimo ou pode demorar para encontrar a solução desejada.

Então o problema passa a ser como convergir estas redes, uma possibilidade seria dividir a rede em um conjunto de sub-redes. Uma sub-rede para cada camada seria uma espécie de várias redes perceptron interligadas. Porém, esta solução iria apenas criar vários problemas menores, geralmente esta divisão não é possível ou é muito complicada.

Outra alternativa seria treinar a rede completa de uma vez só, no entanto, isso causaria outra dificuldade de como treinar as camadas intermediárias. Assim, essa solução implicaria em várias perguntas: Qual seria a resposta desejada para os nodos intermediários? Como seriam definidos os erros? O problema passa agora a ser a definição do erro dos nodos da camada intermediaria, portanto, essa outra solução fica inviável.

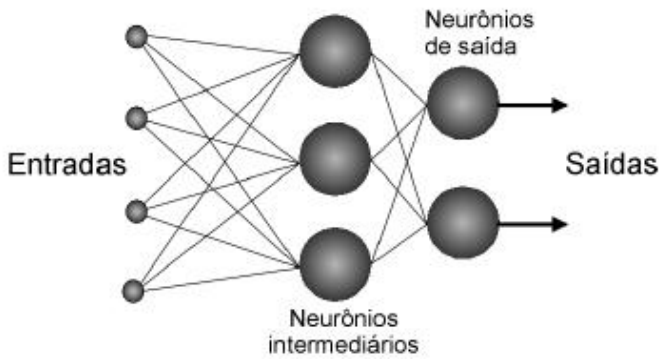
Para treinar as redes com mais de uma camada intermediária utiliza-se o método baseado em gradiente descendente de (RUMELHART; HINTON; WILLIAMS, 1986). A fim de que este método possa ser empregado, a função de ativação necessita ser contínua, diferenciável e preferencialmente não decrescente, onde a função de ativação deve informar os erros cometidos pela rede para as camadas anteriores com maior precisão.

A inexistência ou desconhecimento de algoritmos de treinamento para rede com uma ou mais camada intermediária relatada no capítulo anterior, foi uma das causas da redução das pesquisas em Redes Neurais Artificiais na década de 70.

A organização dos neurônios dentro das camadas e os modelos de conexão entre as camadas são chamados de arquitetura de rede (FAUSSET, 1994), sendo essa arquitetura de fundamental importância, uma vez que restringe o tipo de problema que pode ser tratado pela rede (BRAGA; CARVALHO; LUDEMIR, 2000).

Conforme visto anteriormente, a rede com uma ou mais camadas intermediarias contém neurônios intermediários ou escondidos (neurônios entre os nós fontes e os neurônios de saída) como pode ser observado na Figura 8.

Figura 8- Arquitetura de Rede Direta de Múltiplas Camadas.



Fonte: Adaptado de Braga, Carvalho e Ludemir (2000).

A propagação do sinal é unidirecional (*feedforward*). Porém, com adição de camadas intermediárias, no qual aumenta o poder computacional permitindo a classificação de padrões não linearmente separáveis.

3.4 Algoritmo de retropropagação do erro

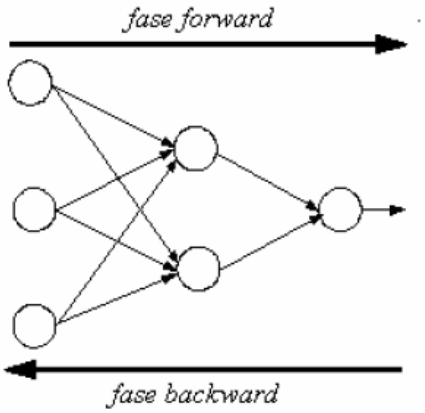
Existe uma grande variedade de algoritmos para treinamentos de rede MLP (RUMELHART; HINTON; WILLIAMS, 1986), (RIEDMILLER, 1994) (PEARLMUTTER, 1992). Cada algoritmo tem sua particularidade em que, geralmente, são do tipo supervisionado.

Os algoritmos de rede MLP, podem ser classificados em: estático e dinâmico. Os estáticos variam apenas os seus pesos não alterando na estrutura da rede. Os dinâmicos, por sua vez, podem tanto reduzir quanto aumentar o tamanho da rede (número de camadas, número de nodos nas camadas intermediárias e número de conexões).

O Algoritmo de retropropagação do erro (*Backpropagation*) é o mais conhecido para treinamento (RUMELHART; HINTON; WILLIAMS, 1986). Este algoritmo foi um dos principais responsáveis pelo ressurgimento do interesse em Redes Neurais Artificiais.

O algoritmo *backpropagation* é um algoritmo supervisionado, o qual faz uso de pares (entrada, saída desejada) para, por meio de um mecanismo de correção de erros, ajustar os pesos da rede. Ele consiste em realizar a retropropagação do erro calculando entre a saída da rede e a saída alvo com o objetivo de minimizar o erro total da saída gerada pela rede. Através dele, o treinamento é realizado em duas fases: uma para frente, ou *forward* e outro para trás, ou *backward* (Figura 9).

Figura 9- Fases de uma RNA Multicamadas com Algoritmo *Backpropagation*



Fonte: Adaptado de (BRAGA; CARVALHO; LUDEMIR, 2000).

A primeira fase, um passo à frente ou *forward*, os dados são iniciados na camada c^0, onde para cada camada c^i (i>0) são calculadas seus sinais obtendo sinais de saídas, estes servem como entrada para a definição das saídas produzidas pelos nodos da camada c^{i+1}, por fim os dados da saídas produzidas pelos nodos da última camada são comparadas às saídas desejadas.

A segunda e última fase, um passo para trás ou *backward*, consiste em percorrer o caminho inverso ao anterior, os erros dos neurônios de saída são calculados e propagados, camada a camada, na direção contraria ao fluxo de entrada, promovendo o ajuste dos pesos. Somente após a alteração de todos os pesos da rede, o processo recomeça.

O sinal do erro recebido por cada neurônio na(s) camada(s) intermediaria(s) corresponde apenas a uma porção do sinal do erro total, esse valor de erro é proporcional à contribuição relativa de cada neurônio para a formação da saída original. Esta é a razão pela qual a retropropagação do erro por meio da rede permite o correto ajuste dos pesos sinápticos entre todas as camadas do modelo conexionista, pois existem pesos que não exigem grandes ajustes.

Os pesos sinápticos (Equação 3.8) têm seus valores atualizados da seguinte forma (HAYKIN, 2001).

$$w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji}(n) \tag{3.8}$$

Em que,

w_{ji} $(n+1)$ → novo peso da j-ésima sinapse do neurônio i, em seu passo de iteração $(n+1)$;

w_{ji} (n) → antigo peso da j-ésima sinapse do neurônio i, em seu passo de iteração n;

Δw_{ji} (n) → correção aplicada a cada peso w_{ji} (n).

A correção Δw_{ji} (*n*) aplicada ao peso w_{ji} (*n*) é definida pela Regra Delta (Equação 3.9) como:

$$\Delta w_{ji}(n) = h d_j(n) y_i(n) \tag{3.9}$$

Em que,

δ_i (n) → gradiente local do erro para o neurônio j;

η → parâmetro da taxa de aprendizagem do algoritmo;

Δw_{ji} (n) → correção aplicada ao peso w_{ji} (n);

y_i (n) → saída do neurônio i precedente ao neurônio j.

O gradiente local do erro δi (n) é determinado pela execução do método Gradiente Decrescente que é dado por (Equação 3.10):

$$\delta_j(n) = -\frac{\partial \xi(\mathrm{n})}{\partial \upsilon_{\mathrm{j}}(n)} = e_j(n)\varphi'_j\left(\upsilon_j(n)\right) \tag{3.10}$$

Em que,

$\delta_i(n)$ → gradiente local do erro para o neurônio *j*, cujo sinal negativo indica a descida do gradiente no espaço de pesos, isto é, busca uma direção para a mudança de peso que reduza o valor de ξ (*n*);

ξ (*n*) → soma dos erros quadráticos produzidos pelos neurônios de saída na iteração *n*;

υ_j (*n*) → sinal de ativação para o neurônio *j* na iteração *n*;

e_j (*n*) → erro do neurônio *j* na iteração *n*;

φ'_j → derivada da função de ativação.

Por fim, o gradiente local para a saída do neurônio *j* é igual ao produto do sinal de erro e_j (*n*) correspondente para esse neurônio pela derivada φ'_j da função de ativação associada.

3.5 Aplicação de rna em recursos hídricos

As RNA surgem como uma metodologia útil na regionalização do nível, uma vez que, considera aspectos diversos das sub-bacias contidas numa região de interesse, como: área de drenagem; comprimento do rio principal; declividade média do rio, pluviometria, etc. Por outro lado, também, leva-se em conta a não linearidade do processo de transformação da chuva em nível, as redes neurais, composta por funções não lineares, revelam-se numa alternativa eficaz, sendo altamente recomendadas em estudos que envolvam relações complexas de entrada e saída, tendo, por isso, alcançado bons resultados na modelagem dos processos hidrológicos.

A principal vantagem do uso da técnica de RNA é o fato dela não requerer conhecimentos explícitos da bacia, dispensando-se uma análise teórica dos processos implícitos do ciclo hidrológico. No entanto, deve-se ressaltar que tal vantagem pode, eventualmente, se reverter numa desvantagem, uma vez que, trata-se de um modelo do tipo caixa-preta, ou seja, não se pode conhecer os motivos pelos quais levaram tal modelo a um certo resultado, podendo-se chegar a resultados inconsistentes. Apenas o bom senso do hidrólogo poderá reparar as falhas de um modelo como o de RNA.

As pesquisas com redes neurais artificiais tentam simular o cérebro humano, principalmente a sua capacidade de aprender e se adaptar a eventuais mudanças por meio de exemplos. Assim, as RNAs podem executar tarefas que os programas convencionais não conseguem realizar, pois não tem essa característica de aprendizagem e adaptabilidade. Os dados de níveis de um rio são registrados num linígrafo. Analisar e prever eventos futuros, fundamentados nesses registros é uma tarefa difícil, porque, várias variáveis como chuva, infiltração e características do solo influenciam na altura do rio de uma maneira não linear (SANTOS, 2001).

A utilização do método empírico de Redes Neurais Artificiais tem obtido resultados satisfatórios em diversos campos do conhecimento, inclusive como método eficaz no tratamento de variáveis temporais complexas, tais como das variáveis hidrológicas chuva e nível. Porém, seu emprego ainda é parco, pois se trata de uma ferramenta relativamente recente, e vem ganhando credibilidade a ponto de concorrer ou ser compatível com métodos mais tradicionais, como no caso da regressão múltipla, que segundo (GARDNER; DORLING, 1998), a RNA têm uma grande vantagem sobre as demais técnicas estatísticas pelo fato de não precisar de suposições iniciais sobre a distribuição estatística dos dados. Suas recentes aplicações na hidrologia têm-se mostrado particularmente efetiva na transformação chuva-vazão e, conforme Favoreto et al. (2001), uma das áreas promissoras para grande aplicabilidade das RNA é na previsão de dados hidrológicos, vazão, nível e precipitação, nos seus mais diferentes usos. Baseado nisso, julgou-se o método de redes neurais artificiais como apropriado para o desenvolvimento desta dissertação.

Muller e Fill (2003) estudaram o comportamento de um modelo de RNA na propagação de vazões em rios. O local de estudo foi o rio Iguaçu, entre Fluviópolis e União da Vitoria. Neste caso, foi criada uma rede perceptron multicamadas, onde, na primeira camada tinham-se dois neurônios, um responsável pela informação da vazão em Fluviópolis e o outro pelo incremento de vazão; depois, mais três camadas escondidas, cada uma com oito neurônios com funções de ativação do tipo sigmoide logística e, finalmente, na última camada, o neurônio de saída, responsável pela resposta da propagação da vazão para União da Vitoria. Os autores concluíram que o método era interessante, pois, apesar do mesmo tratar-se de uma técnica puramente matemática, onde não se levara em conta a teoria intrínseca do problema, tal método obteve resultados animadores.

Nayak *et al*. (2003) realizaram um estudo intitulado "*A neuro-fuzzy computing technique for hydrological time series*", que teve como objetivo a concatenação das técnicas de RNA às de logica difusa para a modelagem do escoamento no rio Baitarani no Estado de Orissa na Índia. O estudo obteve bons resultados; segundo os autores, já que a vazão simulada com o modelo desenvolvido acompanhará o comportamento histórico da série e suas características estatísticas e ressaltaram o aumento da

facilidade no processo de modelar e concluíram afirmando haver viabilidade de tal técnica na modelagem de vazões em rios.

Olívio et al. (2002) aplicaram as técnicas de redes neurais artificiais ao problema de previsão de cheias fluviais, o tipo de rede neural escolhido foi a perceptron de múltiplas camadas e o algoritmo de treinamento usado no estudo foi o da retropropagação do erro. Foi apresentado à rede, como padrão de entrada, o nível do rio a montante de uma seção de interesse e esperou-se uma resposta adequada, da rede neural, com relação ao nível do rio nessa mesma seção de interesse. O interesse inicial, desse estudo, foi a previsão de níveis fluviométricos para 4, 6, 8 e 10 horas de antecedências, contudo, apenas a previsão para 4 horas de antecedência obteve resultados satisfatórios. Os autores concluíram que, redes neurais se prestam para a previsão de cheias e que os resultados poderiam melhorar caso fossem utilizados dados de entrada mais apropriados.

O setor elétrico é o maior responsável pelo gerenciamento dos grandes reservatórios brasileiros, devido a energia elétrica em nosso país é oriunda, em sua grande parte, do potencial hidroelétrico. Portanto, faz-se imprescindível a boa previsão de vazão para garantir a devida operacionalidade destes reservatórios, com vista à diminuição de prejuízos sociais, políticos e econômicos. Até recentemente, para a previsão de vazões, o Setor Elétrico Brasileiro fazia uso dos modelos estocásticos para análise de séries temporais, com ênfase na metodologia de Box e Jenkins (1976).

Ballini *et al*. (1997) trabalharam com o modelo de RNA com o objetivo de prever vazão média mensal e compararam os resultados com os do modelo de Box & Jenkins. As séries históricas temporais foram retiradas das usinas de Furnas, Itumbiara e Sobradinho. A rede neural usada foi do tipo múltiplas camadas. Ballini et al. formularam a pesquisa em duas abordagens, na primeira, a série temporal original foi convertida em uma série padronizada, removendo-se a média e o desvio padrão sazonal. Com isso, foi possível explorar a estrutura das funções de autocorrelação e autocorrelação parcial da série padronizada para determinação das entradas da rede. Na segunda, exploraram a estrutura das funções de autocorrelação e autocorrelação parcial da série original para posterior definição das entradas. Assim, pode-se treinar a rede para que essa também aprendesse a sazonalidade supostamente estocástica. A pesquisa vem destacando a importância da padronização da série e mostraram as

vantagens ao se utilizar redes neurais quando essa padronização não é realizada. Com tudo, a ressalva que se faz é que o desempenho das redes neurais é sempre afetado por fatores como: topologia das redes; parâmetros de treinamento e natureza das séries temporais.

Almeida e Barbosa (2004) associaram um modelo de previsão baseado nas técnicas de redes neurais à teoria dos Runs, com o objetivo de conhecer a previsão da vazão para um horizonte de cinco dias. A rede neural possuía na camada de entrada onze neurônios, que representavam as condições antecedentes da bacia hidrográfica dos últimos cinco dias referentes à vazão e à precipitação e mais uma variável de entrada referente ao total previsto de precipitação para os próximos cinco dias. Na camada de saída, estava o neurônio responsável pela correspondente previsão da vazão média para os próximos cinco dias. O algoritmo de treinamento, usado pelos autores, nesse trabalho, foi o do gradiente conjugado escalonado que utilizam as informações de segunda ordem (matriz Hessiana). Os autores concluíram o trabalho afirmando que os resultados foram satisfatórios na previsão de vazão média diária para um e evento de seca hidrológica e de que, com um acompanhamento diário é possível se detectar a evolução dos déficits e a indicação do possível termino do evento dentro de intervalos de tempo de cinco dias de duração.

Santos (2001) utilizou o conceito de redes neurais na modelagem hidrológica de bacias urbanas com dados de telemetria na previsão de vazão. Seus resultados se mostraram satisfatórios na previsão com RNAs, através dos dados de radar, com até 90 min. de antecedência. A autora comenta, em seu trabalho de dissertação, que os modelos envolvendo RNAs oferecem a vantagem de não requererem um conhecimento explícito da bacia estudada e tem apresentado bons resultados na modelagem de processos hidrológicos de transformação de chuva em vazão, e que devido a isso, essa técnica deva ser utilizada nesse contexto, especialmente nos casos onde os conhecimentos dos processos hidrológicos sejam muito limitados.

Barp e Barbosa (1999) compararam o modelo hidrológico conceitual chuva-vazão SMAP, em sua versão mensal, com modelos desenvolvidos através das técnicas de redes neurais artificiais. Foram trabalhados três casos diferentes. No primeiro caso, investigou-se o potencial individual de cada modelo na geração da vazão, com o SMAP sendo calibrado de forma manual e automática, através de um

método de otimização de primeira ordem; e o modelo de RNA, do tipo perceptron de múltiplas camadas, com um algoritmo de treinamento de segunda ordem utilizado por Von Zuben (1996) e desenvolvido por Moller (1993) e denominado de método do gradiente conjugado escalonado. Segundo os autores, este método mostrou-se superior ao retropropagação do erro, que possui um algoritmo de otimização de primeira ordem. Foram escolhidas duas estruturas de rede neural, uma com um neurônio na camada de entrada, onde se apresentava a variável precipitação como dado de entrada, sete neurônios na camada escondida e um neurônio na camada de saída, responsável pela resposta do modelo para definição da vazão mensal, sendo portando, uma rede do tipo 1-7-1; a segunda estrutura de rede possuía dez neurônios na camada escondida, tendo a estrutura 1-10-1. No segundo e terceiro caso, trabalhou-se com o modelo de redes neurais acoplado ao modelo SMAP para a geração da vazão, com o objetivo de substituição dos parâmetros do modelo conceitual através da implementação de RNAs. Assim, tinham-se, para uma segunda situação, as vazões superficiais e a de base, obtidos pelo modelo SMAP, como padrões de entrada da rede, tendo, portanto, a estrutura do tipo 2-7-1 e 2-10-1. Na terceira versão, a rede neural recebia, como padrão de entrada, os volumes armazenados nos reservatórios solo e subterrâneos, também obtidos através do modelo conceitual SMAP. Os autores fizeram uso de métodos estatísticos dos resíduos para comparação entre os três casos citados, como o erro padrão de estimativa, erro percentual de volume, erro percentual de vazão máxima e correlação. Segundo os autores, o modelo de redes neurais que tem como entrada a precipitação e saída a vazão mensal (caso 1), apresentou desempenho próximo ao do SMAP calibrado manualmente. Porém, quando as entradas do modelo de redes neurais foram as variáveis que já incorporavam processos hidrológicos, advindas do modelo conceitual SMAP (casos 2 e 3), observou-se consideráveis melhorias nos resultados, mostrando-se sempre superior ao modelo conceitual quando esse trabalha de forma isolada. Os autores concluíram, afirmando, que a técnica de redes neurais artificiais é um campo promissor para a modelagem da transformação chuva-vazão em bacias hidrográficas.

Valença (2009) realizou um estudo avaliando a aplicação de redes neurais perceptron multicamadas em recursos hídricos, com ênfase ao processo de transformação chuva em vazão, comparando os resultados, neste caso, com os dos

modelos conceituais MOHTSA e SMAP e na previsão de vazão média mensal com os dos modelos de regressão e Box-Jenkins. Segundo o autor, a comparação dos resultados obtidos entre os modelos tradicionais e os da rede neural para a modelagem chuva-vazão foram de ótima qualidade, destacando a grande vantagem da versatilidade desse último método, devido à sua não-linearidade e de permitir, no processo chuva versus vazão, incorporar a representatividade de cada posto pluviométrico, ao invés de se trabalhar apenas com a chuva média. O autor destaca, também, a vantagem do uso de RNA em permitir a regionalização quando se levam em consideração aspectos diversos das bacias, como: área de drenagem, declividade, pluviometria média, coeficientes de forma da bacia, tipo de solo, etc. No que diz respeito à previsão de vazão média mensal, a técnica de redes neurais obteve, mais uma vez, excelentes resultados e o autor, ainda, completa afirmando que, já era de se esperar pelo fato de redes neurais terem em sua essência potentes modelos não lineares que incorporam os de regressão e Box-Jenkins.

Cannon e Whitfield (2002) utilizaram uma rede neural do tipo Perceptron Multi Layer na modelagem da vazão diária numa bacia canadense. Para isso, eles relacionaram as vazões de 21 estacoes fluviométricas com os dados de temperatura e precipitação correspondente ao período estudado.

Khalil et al. (2001) desenvolveram um modelo baseado nas técnicas de redes neurais capaz de preencher as lacunas de dados hidrológicos existentes nas séries históricas, para isso, foram usadas duas abordagens distintas: (a) o preenchimento das falhas foi obtido a partir da própria série estudada, e nesse caso a vazão atual era função da vazão anterior; ou (b) preenchimento das falhas baseado nas séries dos postos vizinhos, ou seja, os neurônios eram compostos pelas séries dos postos vizinhos. Em ambos os casos a função de ativação utilizada foi a sigmoide.

Diniz e Clarke (2001) apresentaram um estudo, para quatorze bacias da região do semiárido nordestino, que objetivou a utilização das técnicas de redes neurais artificiais na regionalização dos parâmetros do modelo conceitual chuva-vazão SMAP mensal, com base nas características físicas e climáticas mais relevantes das bacias hidrográficas. Num primeiro momento, os autores obtiveram os parâmetros, calibrando o modelo SMAP para cada bacia. Posteriormente, foi realizada a regionalização desses parâmetros por meio da rede neural. A RNA foi do tipo

multicamadas, sendo formada por quatro camadas, tendo uma camada de entrada, duas intermediárias e uma de saída. Na primeira camada, tinham-se seis neurônios, cada um responsável pela recepção de uma informação característica das bacias (área, altitude média, desnível especifico, precipitação média anual, evaporação média anual e permeabilidade média do solo). Cada camada intermediária foi composta por doze neurônios com funções sigmoides. A camada de saída era composta por seis neurônios que representavam os parâmetros do modelo SMAP (capacidade de saturação, coeficiente de infiltração, coeficiente de descarga, constante de selecionamento do reservatório subterrâneo e os parâmetros de escoamento). Foi adotada a validação cruzada na avaliação do desempenho da rede neural, onde se faziam uso de treze sub-bacias para o treinamento da rede e foi obtido o conjunto de parâmetros, o qual foi introduzido no modelo conceitual para simulação da vazão na bacia omitida durante a fase de treinamento, comparando-se o desempenho do modelo SMAP agindo individualmente na simulação da vazão e depois integrado ao modelo de RNA, este procedimento foi repetido quatorze vezes. Segundo os autores, esta técnica obteve resultados bons para algumas bacias; porém, ruim para outras, provavelmente devido às bacias não estarem dentro de uma mesma zona hidro logicamente homogênea.

Freitas (2002), publicou seu trabalho e um dos objetivos de sua pesquisa foi o de analisar a aplicação de Redes Neurais Artificiais como uma ferramenta de regionalização e comparar está com métodos estatísticos usuais. Foram estudadas as sub-bacias 46, 47,48 e 49 da Bacia Hidrográfica do São Francisco. Para isso, foram utilizados dois modelos de RNA para a regionalização da vazão média. No primeiro, os parâmetros utilizados foram a área e os percentuais de solos classificados quanto às características hidrológicas; no segundo, os padrões de entrada foram a área de drenagem, comprimento do rio principal, declividade do rio principal, densidade de drenagem, desnível especifico, precipitação total anual e precipitação máxima diária. Em ambos os casos, a autora dividiu os dados em dois grupos, um de vazões de menores magnitudes e outro de vazões de grandes magnitudes. Os modelos eram calibrados através dos dados de dez estacoes fluviométricas, e validados, normalmente, em cinco estacoes para o grupo (I) e em quatro para o grupo (II) de cada caso. A arquitetura da rede variou de acordo com cada caso e grupo analisado, para o primeiro caso, as redes tiveram as arquiteturas 5-2-1 e 5-3-2-1 para os grupos

(I) e (II), respectivamente, e, para o segundo caso, 7-2-1 e 7-3-1 também para os grupos (I) e (II), respectivamente. Para todas as estruturas foram empregadas funções de ativação gaussiana na camada intermediária e na camada de saída a sigmoide. Segundo a autora, de forma geral, na comparação entre as técnicas estatísticas tradicionais e a de RNA, está última conduziu a melhorias dos índices estatísticos, resultando num maior grau de liberdade dos modelos, o que, os torna mais consistentes. A autora ainda conclui que, tais modelos apresentam bom potencial para a regionalização de vazões em áreas com escassez de dados.

4 MATERIAIS E MÉTODOS

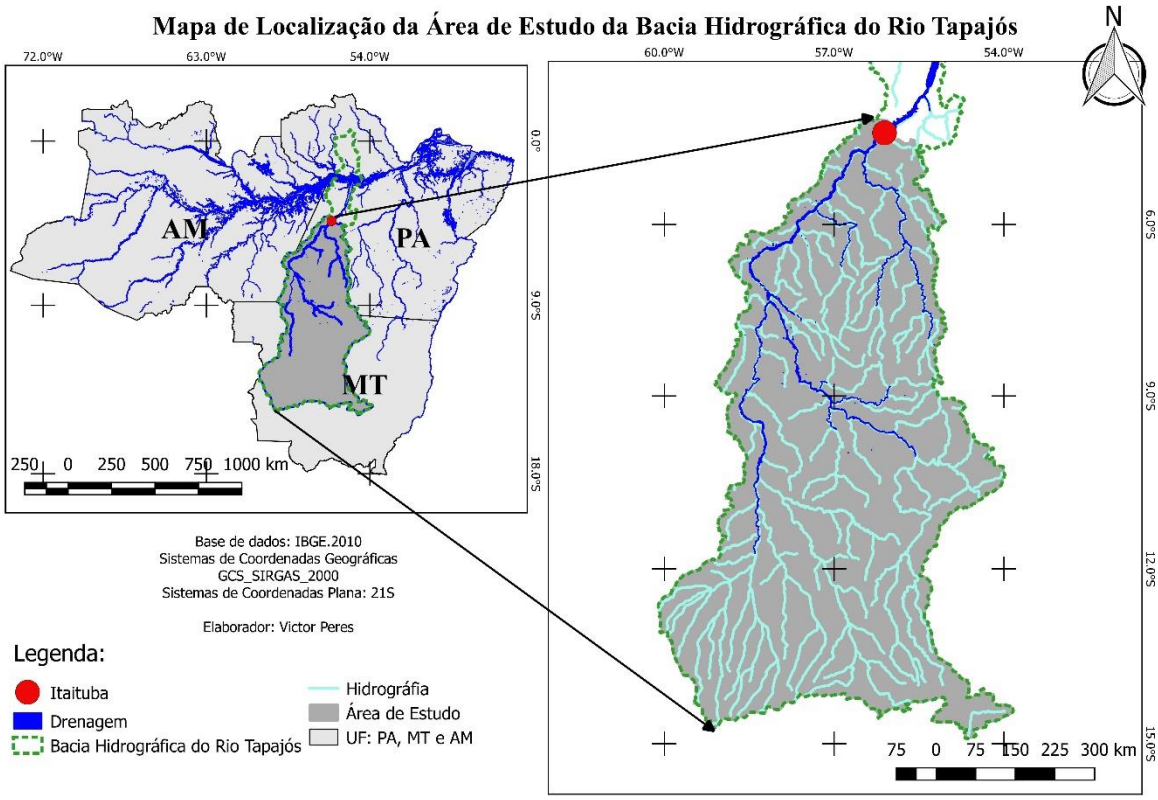
Para o desenvolvimento deste projeto e elaboração de um modelo hidrológico a partir da análise e processamento de dados de precipitação, as seguintes etapas foram seguidas:

- **Área da pesquisa:** escolha do local onde foi aplicado a metodologia.
- **Coleta de dados de treinamento:** coleta dos dados pertinentes e potencialmente úteis a tarefa;
- **Projeto da estrutura e configuração da rede:** escolha da configuração e adequação da RNA;
- **Treinamento e teste:** treinamento onde foram realizadas diferentes arquiteturas de rede. As arquiteturas foram validadas para avaliar o melhor desempenho;
- **Desenvolvimento do sistema:** desenvolver um sistema de alerta.

4.1 Área da pesquisa

A área utilizada está inserida na Bacia do Rio Tapajos, sendo esta formada pela confluência do rio Teles Pires com o rio Juruena, em Barra de São Manuel na fronteira entre Pará e Mato Grosso, e percorre uma extensão de aproximadamente 800 km até desaguar no Amazonas. A área de estudo esta distribuída pelos estados do Mato Grosso, Pará, Amazonas, ocupando uma área total de 492.263 km2, que apresenta largura da ordem de 555 km e comprimento de 1.457 km, com uma direção geral SSE-NNW. Sendo o modelo hidrologico realizado em Itaituba/Pa, logo os dados analisados foram até está cidade, como podemos verificar no Figura 10.

Figura 10- Mapa de Localização da Área de Estudo da Bacia Hidrográfica do Rio Tapajós.



Fonte: Do autor.

4.2 Dados hidrometeorologicos

Para o desenvolvimento da RNA, normalmente foram utilizados dois passos: a coleta de dados relativos ao problema e a sua separação em conjunto de treinamento e testes.

Esta tarefa requer uma análise cuidadosa sobre o problema para minimizar erros e ambiguidades dos dados, além disso, os dados coletados devem cobrir amplamente o domínio do problema.

Nesta pesquisa os dados foram utilizados e consistidos pela Agência Nacional de Água e CPC/NCEP de 1968 à 2017, tanto na fase de treinamento, quanto na fase de teste. Leva-se em consideração que os cuidados necessários em relação à escolha dos dados, como descrito acima, foram tomados. Isto porque a contribuição do trabalho não está na escolha da base e em criação de uma rede neural para que possa ser classificado e validado os dados experimentais obtidos.

Para a coleta dos dados foram necessárias fazer a classificação ao nível da bacia conforme as seguintes etapas:

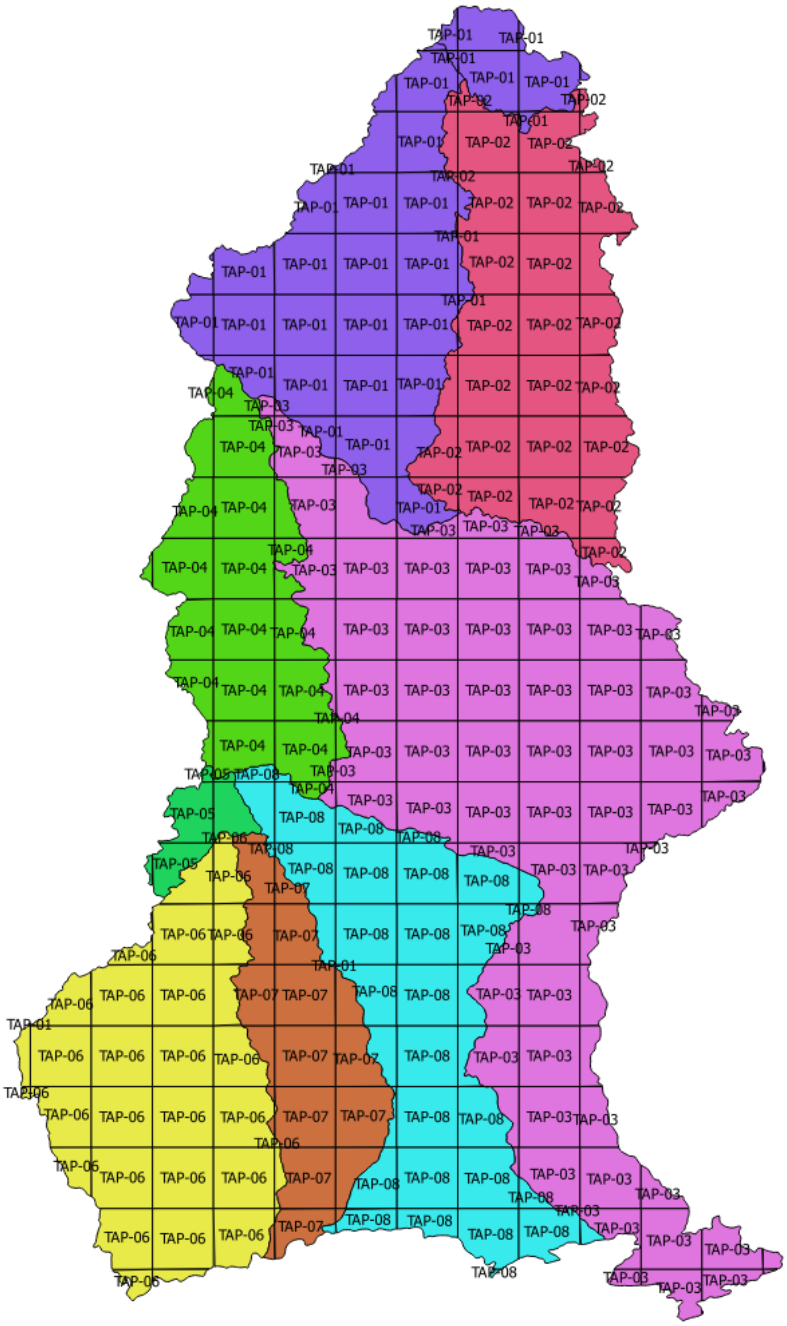
- Divisão da Região em Sub-Bacias;
- Cálculo do Tempo de Concentração;
- Sombreamento para retirada dos dados das precipitações nas sub-bacias;

Essas etapas serão descritas abaixo.

4.2.1 Divisão da região em sub-bacias

A área de influência foi calculada de acordo com a metodologia descrita por (BERTONI; TUCCI, 2001) sobre o polígono de Thiessen: onde os pluviômetros foram ligados por trechos retilíneos; em seguida foram traçadas linhas perpendiculares aos trechos retilíneos passando pelo meio da linha que liga os dois pluviômetros; as linhas perpendiculares foram prolongadas até encontrar outra linha. O polígono foi formado pela intersecção das linhas, correspondendo a área de influência de cada pluviômetro denominadas de (TAP-01, TAP-02, TAP-03, TAP-04, TAP-05, TAP-06, TAP-07, TAP-08). A criação das entradas de dados das redes neurais, tendo em vista que cada sub-região possui um tempo de concentração, ou seja, tempo que a água percorre sobre a sub-Bacia. Como podemos observar na Figura 11.

Figura 11- Região da área da pesquisa com a divisão das áreas de influência dos pluviômetros segundo o método de Thiessen elaborado com o auxílio do software ArcGis 9.1(SANTOS, 2007).



Fonte: Do autor.

4.2.2 Cálculo do tempo de concentração

A determinação dos valores da precipitação total diária ocorrida na bacia hidrográfica foi calculada através do método de Thiessen, tendo como referência os limites de cada sub-bacia. O método de Thiessen se caracteriza pela atribuição de

pesos relativos aos postos considerados mais representativos para uma área específica. O método assume que em qualquer ponto da bacia o valor da precipitação é igual à medida no posto mais próximo (Villela, 1975). Caso a grade extrapolar áreas à bacia, essa porção será eliminada no cálculo. Desta forma, a determinação da precipitação da bacia será obtida através do produto entre a área parcial de cada grade $\boldsymbol{A_n}$ e a sua respectiva precipitação $\boldsymbol{P_n}$. Desta forma, a precipitação total diária da bacia foi calculada através da Equação 4.11.

$$\bar{P} = \frac{\sum_{i=1}^{n} A_i \bar{P}_i}{A} \tag{4.11}$$

Em que:

$\bar{h}$ → precipitação média na bacia (mm);

h_i → precipitação no posto *i* (mm);

A_i → área do respectivo polígono, dentro da bacia (km^2);

A → área total da bacia.

4.2.3 Função do cálculo de concentração

Tempo de concentração, tc, de uma bacia hidrográfica, numa dada secção de um curso de água, é o tempo para que a totalidade da bacia contribua para o escoamento superficial na secção considerada. Pode também ser definido como o tempo necessário para que uma gota de água caída no ponto hidraulicamente mais afastado da bacia atinja a secção considerada.

Segundo Lencastre e Franco (1984), o tempo de concentração é considerado uma característica constante da bacia, sendo independente das características das chuvadas. O tempo de concentração é aplicado na determinação do caudal de ponta de cheia quando se utilizam expressões cinemáticas, que entram em linha de conta com as características do movimento da água na bacia hidrográfica.

Para calcular os tempos de concentração das sub-bacias hidrográficas, foi utilizada as propostas de Ventura, como podemos observar na Equação 4.12 e Tabela 2.

$$t_c = 240 \left(\frac{A_b \, L_b}{\Delta h} \right)^{\frac{1}{2}} \tag{4.12}$$

Em que:

t_c → tempo de concentração (min);

A_b → área da sub-bacia hidrográfica (km^2);

Δh → diferença de cotas entre as extremidades da linha de água principal (m);

L_b → comprimento do curso de água principal da bacia (km).

4.2.4 Retirada dos dados das precipitações nas sub-bacias

Após os procedimentos para coleta dos dados, serão selecionados dados diários a partir de 1968 até 01/01/2017, portanto no banco de dados contém 8 entradas de dados que diz respeito as sub-bacias, de acordo com seu tempo de concentração e nível em Itaituba como dado de saída da rede neural para facilitar a convergência dos dados na rede.

Para o desenvolvimento do modelo utilizando rede neural foi implementada, treinada e testa utilizando a Linguagem de Programação JAVA, por meio do software Netbeans 8.2, em ambiente Macintosh. Para melhor compreensão dos resultados será utilizado o software Grapher para plotar os resultados.

As RNA implementadas foram do tipo MLP (*MultiLayer Perceptron*) com o treinamento realizado pelo algoritmo backpropagation. Na elaboração de um projeto de RNA do tipo MLP a tarefa consiste em determinar o número de camadas intermediárias, bem como a número de neurônios de processamento de cada camada intermediária, embora não existam metodologias para determinar o número de camadas e neurônios. Assim sendo, foram projetadas várias arquiteturas para alcançar o resultado desejado.

4.3 Treinamento e teste

Como analisado anteriormente, os dados foram treinados em uma ferramenta de Inteligência Computacional. Dessa forma, o problema de determinar o nível torna-se um problema de classificação de padrões tratável por algoritmos de reconhecimentos de padrões, tais como redes neurais MLP.

A camada de entrada e saída foram selecionadas, como mencionado no capítulo anterior, onde constam 9 (nove) neurônios de entrada referente a quantidade de precipitação diárias nas sub-bacias (Nível, TAP-01, TAP-02, TAP-03, TAP-04, TAP-05, TAP-06, TAP-07, TAP-08) e um neurônio de saída onde é será determinado o resultado.

Foram projetadas e implementadas várias arquiteturas diferentes: com diferentes camadas intermediárias e números de neurônios, objetivando encontrar a melhor arquitetura a ser utilizada no trabalho.

A definição da estrutura e configuração da rede pode ser dividida em três etapas: seleção da arquitetura da rede apropriada à aplicação; determinar a topologia a ser utilizada, isto é, o número de camadas e quantidades de neurônios em cada camada; e determinação de parâmetros do algoritmo de treinamento e funções de ativação.

A rede neural apresentada neste trabalho foi implementada, treinada e testada utilizando a Linguagem de Programação JAVA, por meio do software Netbeans 8.2, em ambiente Macintosh. Para melhor compreensão dos resultados será utilizado o software Grapher para plotar os resultados.

As RNAs implementadas foram do tipo MLP com o treinamento realizado pelo algoritmo backpropagation. Na elaboração de um projeto de RNAs do tipo MLP a tarefa consiste em determinar o número de camadas intermediárias, bem como a número de neurônios de processamento de cada camada intermediária, embora não existam metodologias para determinar o número de camadas e neurônios. Assim sendo, foram projetadas várias arquiteturas para alcançar o resultado desejado.

Outras configurações estão listadas abaixo:

a) Função de ativação utiliza foi a sigmoide;
b) Taxa de aprendizagem (η) é igual a 0,2;
c) Os valores dos pesos w são inicializados aleatoriamente com valores diferentes de *thresholds* internos e com conexões de pequenos pesos, variando de -1 à 1 e a soma dos pesos nos intervalos de cada camada é igual a 0;
d) A taxa de parada de treinamento é quando o Erro médio quadrático for menor que 0,01 (1%).

4.4 Sistema de alerta

Para o desenvolvimento do Sistema de alerta foram necessárias três etapas:

a) **Planejamento e projeto do sistema:** foram coletados todos os requisitos pertinentes e potencialmente úteis à tarefa para que pudesse definir o escopo do projeto;
b) **Classificação do nível da bacia:** foram criadas classes para facilitar uma identificação mais significativa das áreas alagadas;
c) **Implementação do sistema:** codificação do sistema SIGMA-Óbidos, utilizando linguagem de programação Java;

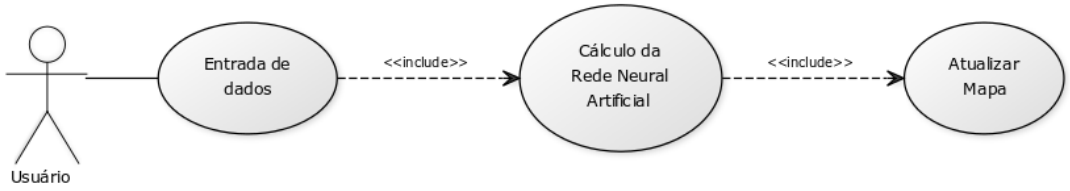
4.4.1 Planejamento e projeto do sistema

Foram feitos estudos com profissionais da área, onde constatou a necessidade do desenvolvimento da ferramenta computacional para auxiliar os governantes em uma possibilidade de visualização mais transparente do problema da cidade. Os requisitos foram então modelados na forma de Diagramas de Casos de Uso UML (Linguagem de Modelagem Unificada), permitindo ao usuário um melhor esclarecimento com relação ao sistema, além de permitir um refinamento dos requisitos levantados.

A Figura 12 ilustra o cenário do sistema. O sistema possui apenas um ator que é o usuário e não necessita de segurança, haja vista não trabalhar com informações confidenciais. O caso de uso inicia no cenário – Entrada de dados, onde

o usuário insere os dados manuais dos neurônios inicias da redes neurais (Nível do dia, Tap1, Tap2, Tap3, Tap4, Tap5, Tap6, Tap7, Tap8), logo depois irá iniciar o cálculo da RNA, tendo como resultado o nível do Rio em Itaituba e atualizando o mapa com as ruas alagadas.

Figura 12- Casos de USO do sistema.

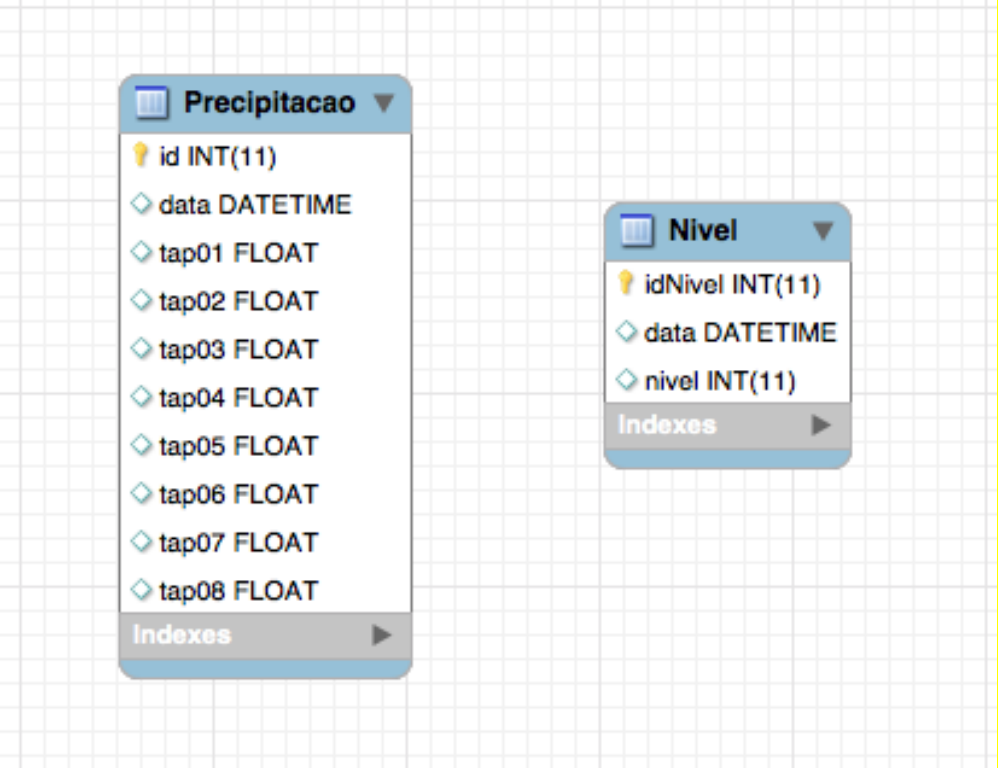


Fonte: Do autor.

Para criação do diagrama entidade-relacionamento, foi utilizado como ferramenta o software DBDesigner. O sistema possui um Banco de Dados que poderá ser alterado apenas pelos desenvolvedores, portanto o sistema não possui cadastros.

Após a modelagem do banco de dados, foi feito o Diagrama de Classes (Figura 13), modelagem muito útil que contribui para o início da fase de implementação do sistema (SOMMERVILLE, 2003). Para a criação do Diagrama de Classes, foi utilizada a ferramenta ArgoUML, aplicação *open source* que usa UML para modelar o desenho de software de computador, além de providenciar suporte para quase todos os tipos de diagrama UML padrão, incluindo suporte cognitivo.

Figura 13- Modelo de Classes do sistema.



Fonte: Do autor.

4.4.2 Classificação do nível da bacia

Foram criadas classes para facilitar uma identificação mais significativa na identificação das áreas alagadas, portanto o sistema possui 4 classes (leve, moderado, intenso e muito intenso), onde podemos observar na Figura 14:

- **Leve:** apresentará a cor amarela se o nível do rio estiver entre 1cm á 10cm acima da cota da rua;
- **Moderado:** apresentará a cor laranja se o nível estiver entre 11cm e 50cm acima da cota da rua;
- **Intenso:** apresentará a cor marrom se o nível estiver entre 51cm e 100cm acima da cota da rua;
- **Muito Intenso:** apresentará a cor vermelha se o nível estiver acima de 100cm da cota da rua;

Figura 14- Classes do sistema de alerta.

Fonte: Do autor.

4.4.3 Implementação do sistema

A etapa de implementação teve como suporte ferramental a plataforma de desenvolvimento Netbeans 8.2, em ambiente Macintosh, utilizando o padrão MVC, para facilitar em uma possível fatoração da Visão *View* para um dispositivo móvel. A linguagem de programação utilizada foi a JAVA por ser uma linguagem gratuita. Com esses recursos foi desenvolvida toda lógica de programação e a interface com o usuário. A Figura 15 demonstra o cálculo da área em JAVA.

Figura 15- Trecho do código do cálculo da área em JAVA do sistema.

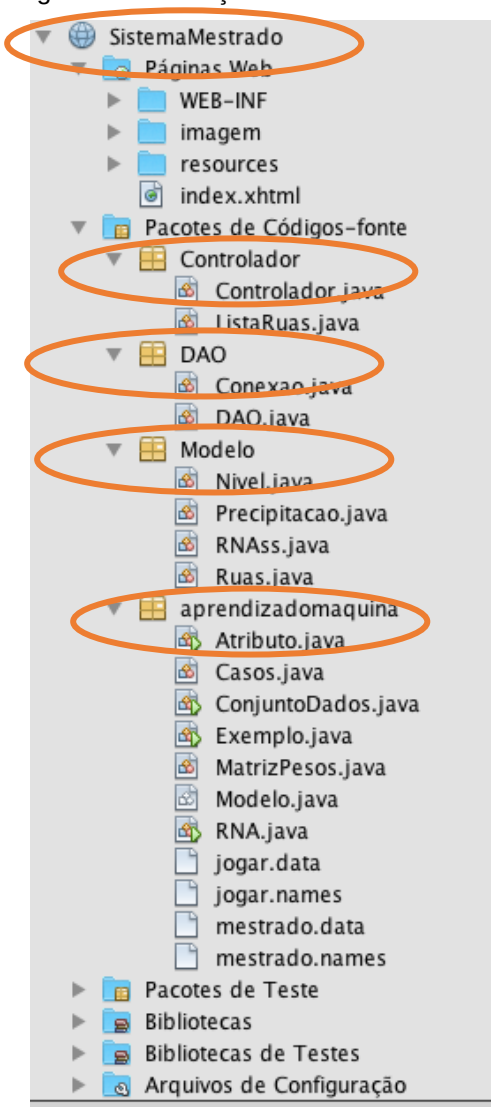
```
public void verificar(SlideEndEvent event) {
    nivelRio = event.getValue();
    modeloMapa = new DefaultMapModel();
    for (Ruas ruas : temp) {
        int temp = nivelRio - ruas.getNivel();
                    if (temp > 0) {
            if (temp <= valor1) {
                Polyline polyline = new Polyline();
                polyline.getPaths().add(ruas.getLocalizacao());
                polyline.getPaths().add(ruas.getLocalizacao2());
                polyline.setStrokeWeight(10);
                polyline.setStrokeColor("#FFFF00");
                polyline.setStrokeOpacity(0.7);
                modeloMapa.addOverlay(polyline);
            } else if (temp <= valor2) {
                Polyline polyline = new Polyline();
                polyline.getPaths().add(ruas.getLocalizacao());
                polyline.getPaths().add(ruas.getLocalizacao2());
                polyline.setStrokeWeight(10);
                polyline.setStrokeColor("#FFA500");
                polyline.setStrokeOpacity(0.7);
                modeloMapa.addOverlay(polyline);
            } else if (temp <= valor3) {
                Polyline polyline = new Polyline();
                polyline.getPaths().add(ruas.getLocalizacao());
                polyline.getPaths().add(ruas.getLocalizacao2());
                polyline.setStrokeWeight(10);
                polyline.setStrokeColor("#A0522D");
                polyline.setStrokeOpacity(0.7);
```

Fonte: Do autor.

O padrão de arquitetura MVC *(Model-View-Controller)* (Krasner and Pope 1998) é bastante utilizado no desenvolvimento de aplicações para dispositivos móveis, pois determina a separação de uma aplicação em três elementos.

O *Model* é formado por entidades que representam os dados da aplicação. A *View* tem por objetivo realizar a apresentação destes dados e capturar os eventos do usuário; sendo representada pelas telas. O *Controller* faz a ligação entre o *Model* e a *View*, realizando o tratamento dos eventos, atuando sobre o *Model* e alterando os elementos da *View* para representar a nova forma dos dados. A Figura 16 ilustra a utilização do padrão MVC no sistema de alerta.

Figura 16- Utilização do MVC no sistema.



Fonte: Do autor.

Para o desenvolvimento foi necessário a utilização de APIs e serviços de manipulação de mapas. A Google Maps Javascript API v3 foi escolhida por ser gratuita e dispor de ferramentas que atendem as necessidades do sistema. Foram utilizados os serviços de elevação e geocodificação que operam por meio de solicitações assíncronas a um servidor externo e respondem na forma de JSON ou XML. Além disso, sobre seu componente de mapa pode-se desenhar e pintar as áreas alagadas.

5 RESULTADOS E DISCUSSÃO

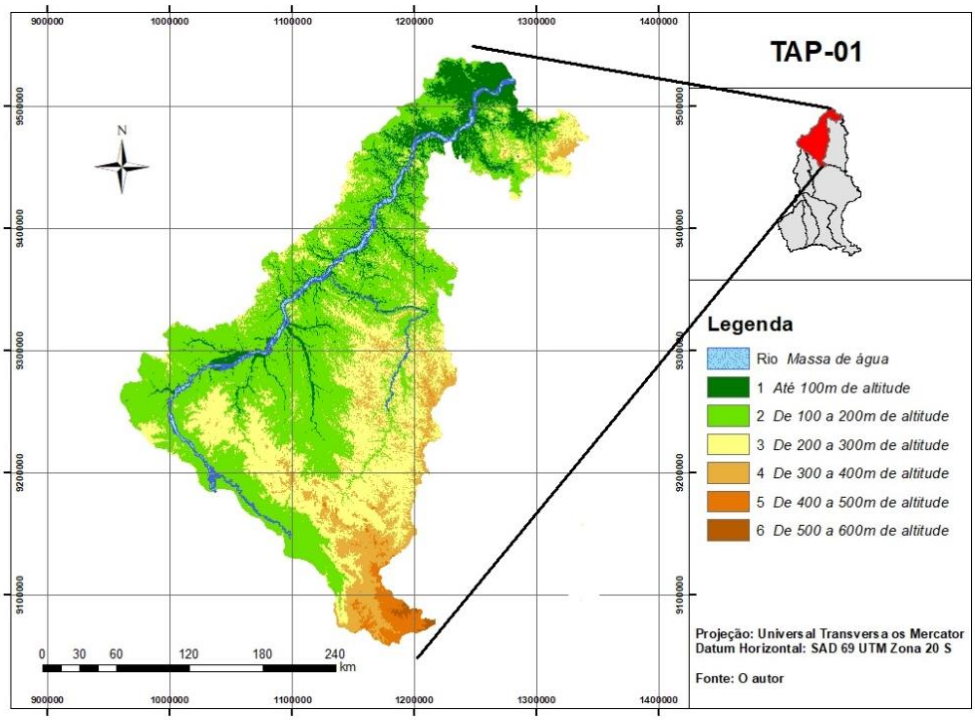
5.1 Caracterização da bacia

No experimento desenvolvido, a compreensão do problema é necessária para um melhor entendimento sobre o comportamento de certas variáveis. Diante disso, foi feito a hipsometria das sub-Bacias para desenvolver o cálculo do tempo de concentração.

Na sub-região (TAP-01), pode ser observado na Figura 17 a hipsometria e massa de água. A área dessa sub-região é de 67.898,39 km^2, a maior incidência de altitudes está entre 100 a 200 m, que correspondem a 48,74% do total, e menor incidência de altitudes entre 500 a 600, corresponde a 0,27% do total. A altitude mais alta é de 590m, a massa de água (Rio Tapajós) percorre toda a sub-bacia com distância de 575km com o desnível de 153m.

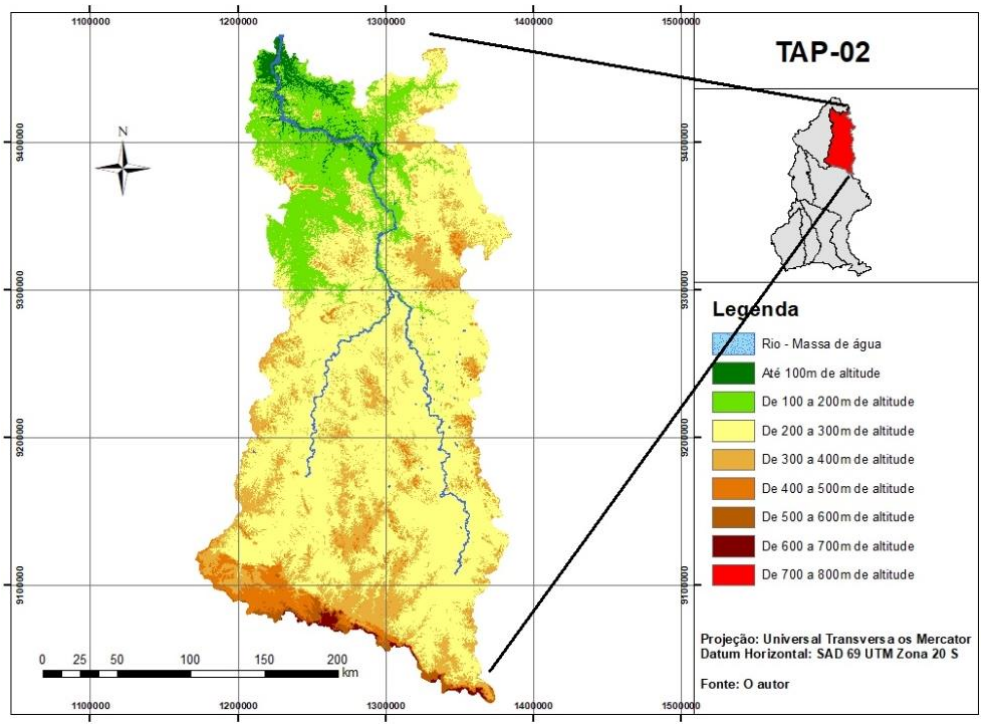
Na sub-região (TAP-02), pode ser observado na Figura 18 a hipsometria e massa de água. A área dessa sub-região é de 58.848,09 km^2, a maior incidência de altitudes está entre 300 a 400 m, que correspondem a 17,12% do total, e menor incidência de altitude entre 700 a 800, corresponde a 0,07% do total. A altitude mais alta é de 758m, a massa de água (Rio Tapajos) percorre por toda a sub-bacia com um comprimento de 447km com o desnível de 343m.

Figura 17- Hipsometria da sub-região TAP-01.



Fonte: Do autor.

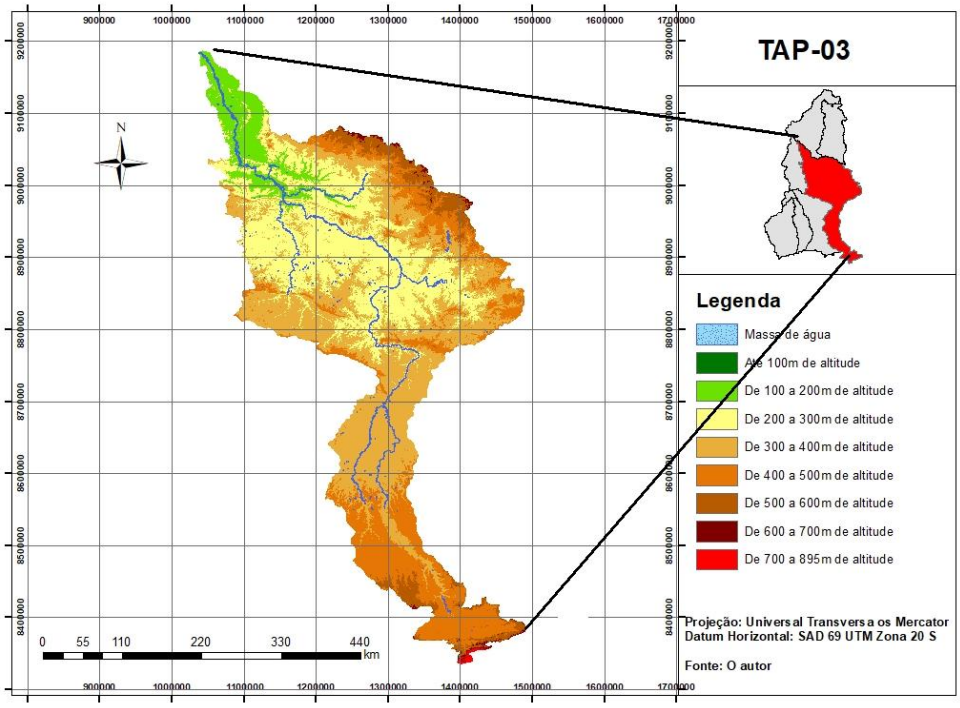
Figura 18- Hipsometria da sub-região TAP-02.



Fonte: Do autor.

Na sub-região (TAP-03), pode ser observado na Figura 19 a hipsometria e massa de água. A área dessa sub-região é de 143.907,48 km^2, a maior incidência de altitudes está entre 300 a 400 m, cujas áreas correspondem a 35,97% do total, e menor incidência de altitude entre 0 a 100, corresponde 0,01% do total. A altitude mais alta é de 895m e a massa de água (Rio Tapajós) percorre toda a sub-bacia com um comprimento de 876km com o desnível de 481m.

Figura 19- Hipsometria da sub-região TAP-03.



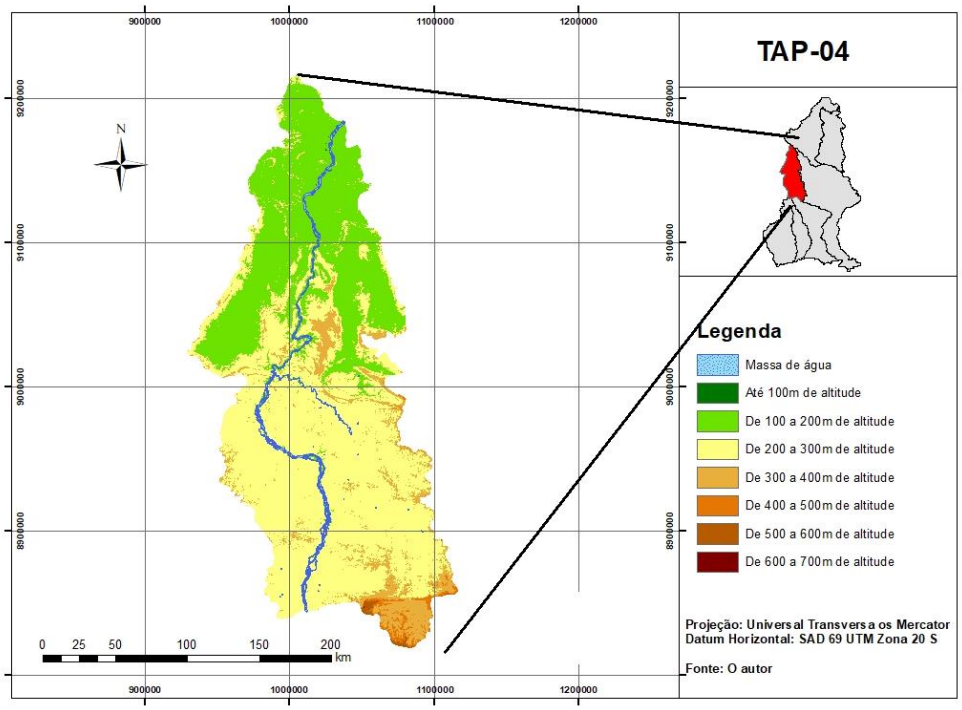
Fonte: Do autor.

Na sub-região (TAP-04), pode ser observado na Figura 20 a hipsometria e massa de água. A área dessa sub-região é de 38.570 km^2, a maior incidência de altitudes está entre 200 a 300 m, cujas áreas correspondem a 56,62% do total, e menor incidência de altitude entre 600 a 700, corresponde 0,001% do total. A altitude mais alta é de 662m e massa de água (Rio Tapajos) percorre por toda a sub-bacia com um comprimento de 404km com o desnível de 590m.

Na sub-região (TAP-05), pode ser observado na Figura 21 a hipsometria e massa de água. A área dessa sub-região é de 38.570 km^2, a maior incidência de altitudes está entre 200 a 300 m, cujas áreas correspondem a 58,04% do total, e menor incidência de altitude entre 100 a 200, corresponde 0,01% do total. A altitude

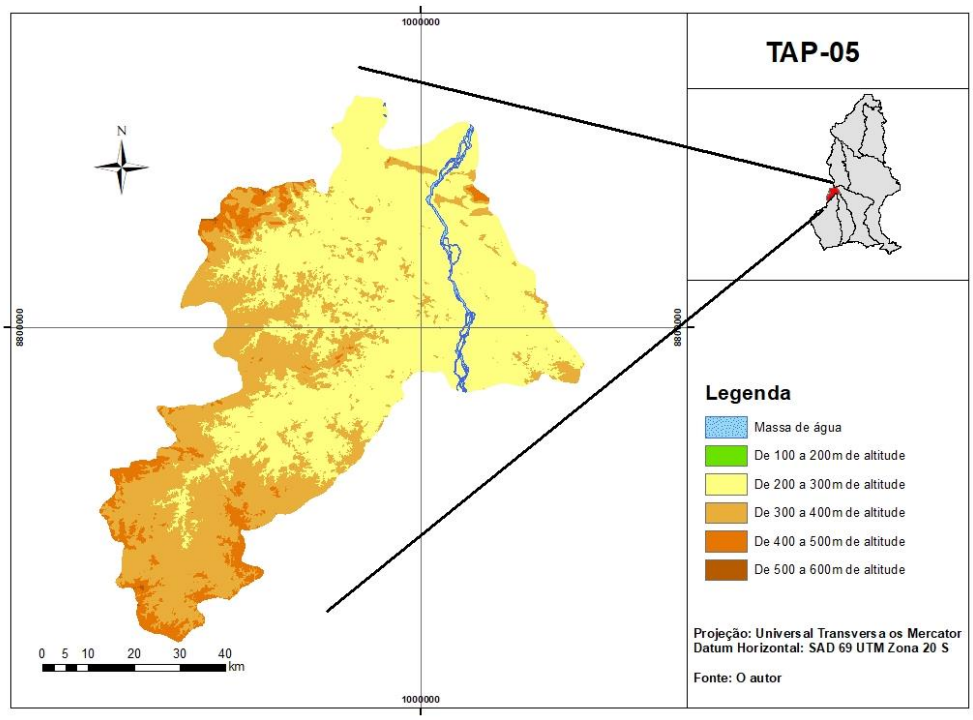
mais alta é de 538m e massa de água (Rio Tapajos) percorre por toda a sub-bacia com um comprimento de 65km com o desnível de 315m.

Figura 20- Hipsometria da sub-região TAP-04.



Fonte: Do autor.

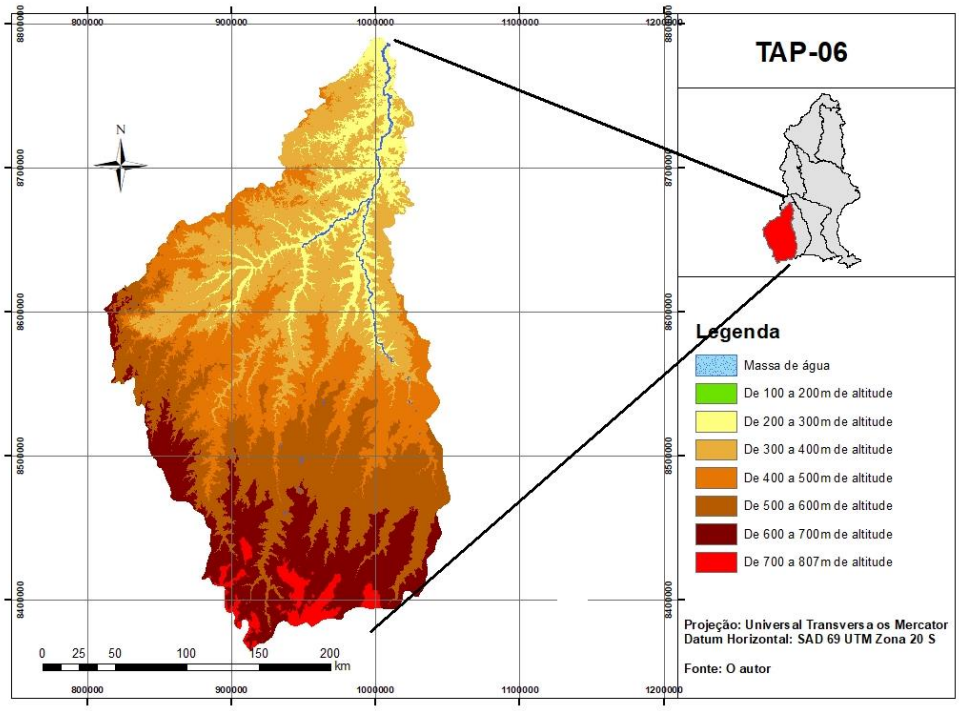
Figura 21- Hipsometria da sub-região TAP-05.



Fonte: Do autor.

Na sub-região (TAP-06), pode ser observado na Figura 22 a hipsometria e massa de água. A área dessa sub-região é de 59.496 km^2, a maior incidência de altitudes está entre 500 a 600 m, cujas áreas correspondem a 26,46% do total, e menor incidência de altitude entre 700 a 807, corresponde 2,37% do total. A altitude mais alta é de 807m e massa de água (Rio Tapajos) percorre por toda a sub-bacia com um comprimento de 246km com o desnível de 250m.

Figura 22- Hipsometria da sub-região TAP-06.

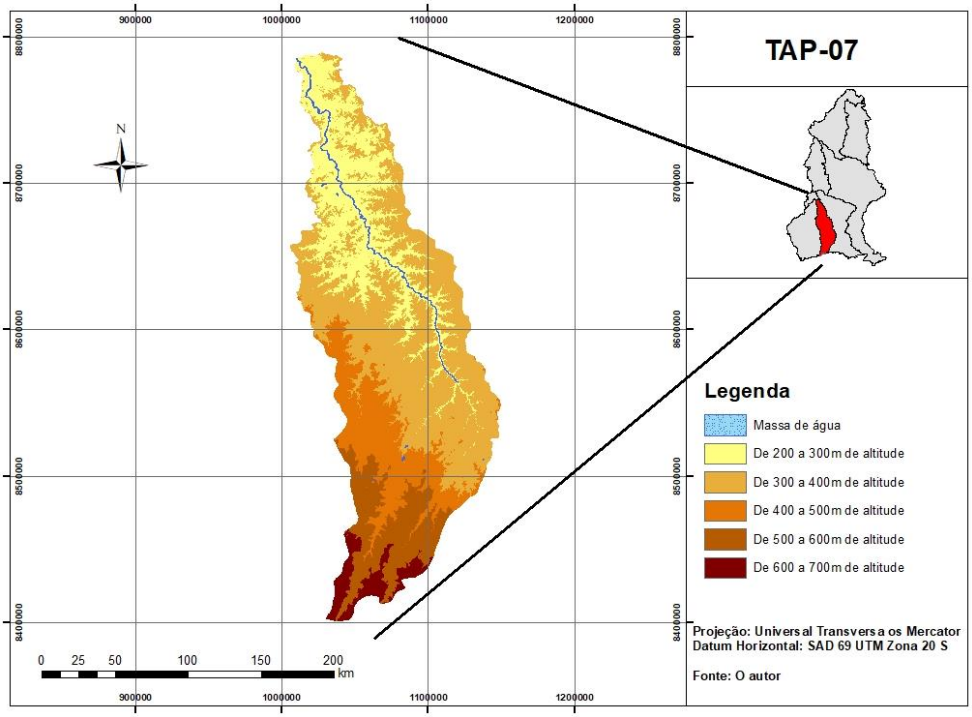


Fonte: Do autor.

Na sub-região (TAP-07), pode ser observado na Figura 23 a hipsometria e massa de água. A área dessa sub-região é de 29.081 km^2, a maior incidência de altitudes está entre 500 a 600 m, cujas áreas correspondem a 44,53% do total, e menor incidência de altitude entre 600 a 694, corresponde 3,99% do total. A altitude mais alta é de 694m e massa de água (Rio Tapajos) percorre por toda a sub-bacia com um comprimento de 271km com o desnível de 358m.

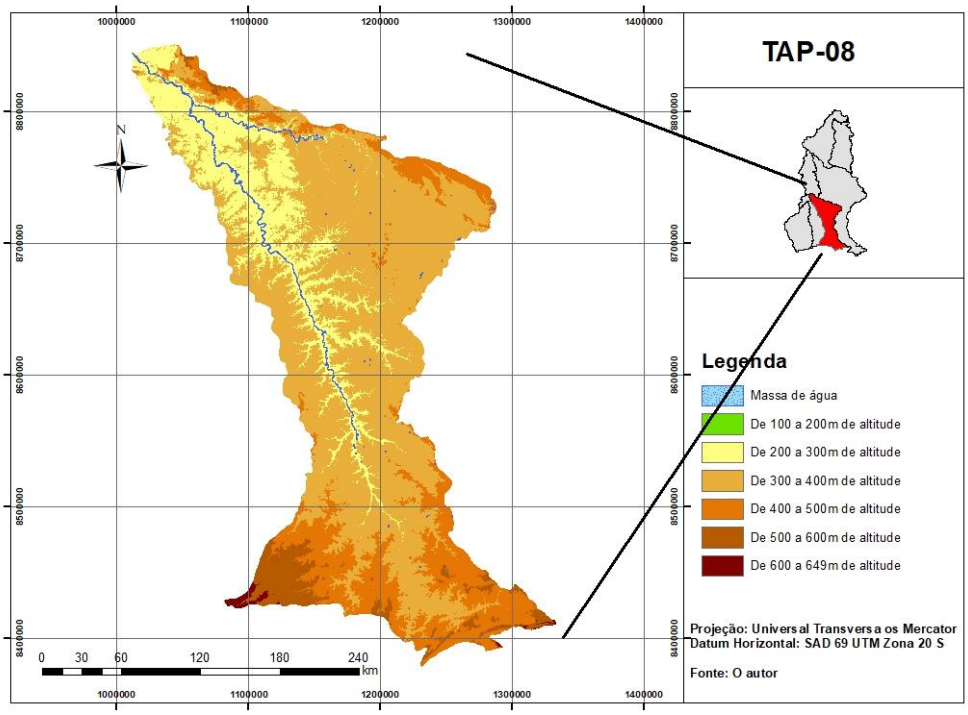
Na sub-região (TAP-08), pode ser observado na Figura 24 a hipsometria e massa de água. A área dessa sub-região é de 59.003 km^2, a maior incidência de altitudes está entre 300 a 400 m, cujas áreas correspondem a 57,73% do total, e menor incidência de altitude entre 100 a 200, corresponde 0,01% do total. A altitude mais alta é de 649m e massa de água (Rio Tapajos) percorre por toda a sub-bacia com um comprimento de 374km com o desnível de 365m.

Figura 23- Hipsometria da sub-região TAP-07.



Fonte: Do autor.

Figura 24- Hipsometria da sub-região TAP-08.



Fonte: Do autor.

5.1.1 Tempo de concentração

A Tabela 1 mostra os resultados dos tempos de concentração estimados pelas equações de Ventura utilizando as analises morfométrica das sub-Bacias demonstrada no resultado anterior, para isso é necessário conhecer a área total (Ab) drenada pela bacia, o comprimento do talvegue principal (Lb) e a diferença de cotas (Δh). O resultado do método de ventura é fornecido em minutos, diante disso foi realizado os arredondamentos sempre para cima.

Tabela 1- Tempo de concentração nas sub-bacias.

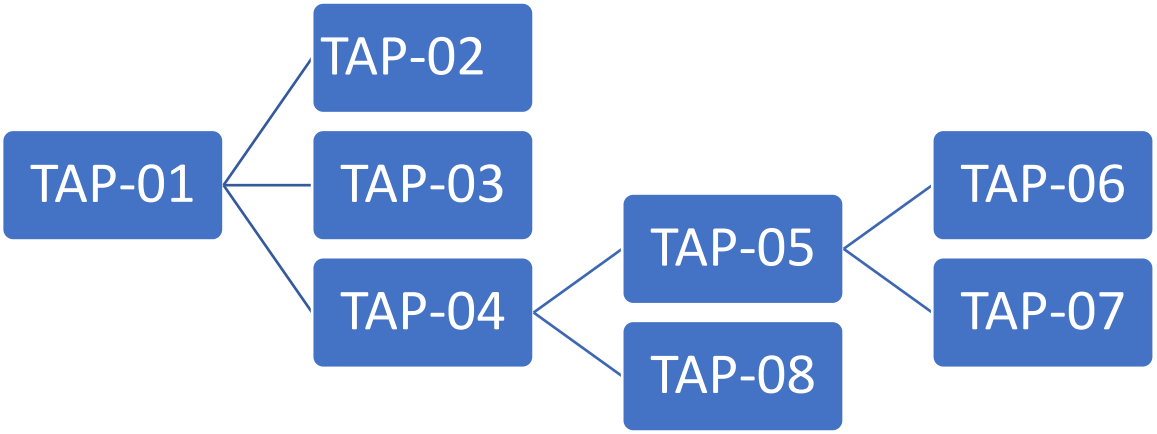
Nome	A_b	L_b	Δh	t_c(Minutos)	t_c(Dias)
TAP-01	67898	575	153	7825	6
TAP-02	58848	447	343	4290	3
TAP-03	143907	876	481	7930	6
TAP-04	38570	404	590	2517	2
TAP-05	5354	65	315	514	1
TAP-06	59496	246	250	3748	3
TAP-07	29081	271	358	2298	2
TAP-08	59003	374	365	3809	3

Na tabela 2, podemos observar o tempo de concentração nas sub-bacias até o município de Itaituba/PA. O cálculo do tempo de concentração real de uma sub-bacia (tcr) foi realizado calculando a somatória dos tempos de contrações das sub-bacias que a antecedem (Figura 25), por exemplo, o tempo de concentração do TAP-02 é de 9 dias, ou seja, 3 dias do próprio TAP-02 mais 6 dias do TAP-01. Devido a precipitação que escorre nessa sub-Bacia tem que passar pela sub-bacia TAP-01.

Tabela 2- Tempo de concentração nas sub-bacias até Itaituba/PA.

Nome	t_{cr}(Dias)
TAP-01	6
TAP-02	9
TAP-03	12
TAP-04	8
TAP-05	9
TAP-06	12
TAP-07	11
TAP-08	11

Figura 25- Estrutura do cálculo do Tempo de Concentração Real.



Fonte: Do autor.

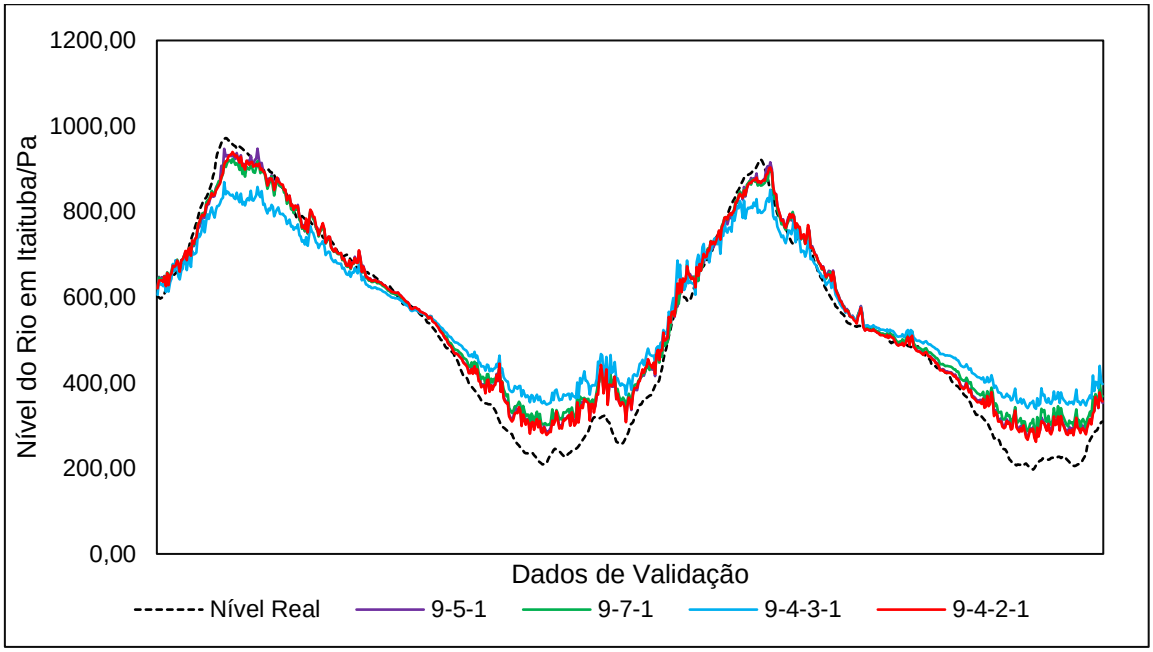
5.2 Resultados obtidos com a rede MLP

Durante todo o processo de criação de redes neurais, houve a necessidade de encontrar um modelo que devolvesse os resultados com 100% de acertos. Para tanto, no desenvolvimento do trabalho criou-se algumas arquiteturas, até encontrar uma que se adequasse ao que era previsto, utilizando valores dos pesos de modo aleatório, para assim, realizar todo o treinamento utilizando o Algoritmo de *Backpropagation* com a finalidade de encontrar o menor erro possível.

Ressalta-se ainda, que em todas as arquiteturas testadas foram utilizadas aproximadamente 94.000 iterações, porém para melhor visualizar graficamente foi plotada no gráfico somente até a iteração 10.000, uma vez que, a partir desse momento de iteração o valor do erro alterava-se pouco.

A primeira arquitetura a ser testada foi o modelo 9-5-1, rede com 3 camadas, ou seja, 9 neurônios de entrada, 5 neurônios intermediários e 1 de saída. Com a utilização deste modelo, foram feitas mais de 94.000 iterações, obtendo desta forma mais de 40% de taxa de erro, pois foi observado que a partir de 2000 iterações (Figura 26), os resultados obtidos não convergiam tanto, com isso foi necessário obter outro modelo para testar a sua convergência com o problema em questão. Sendo assim, foram testados outros quatro modelos, 9-7-1, 9-4-2-1, 9-4-3-1 e 9-4-1, como pode ser observado na Figura 26.

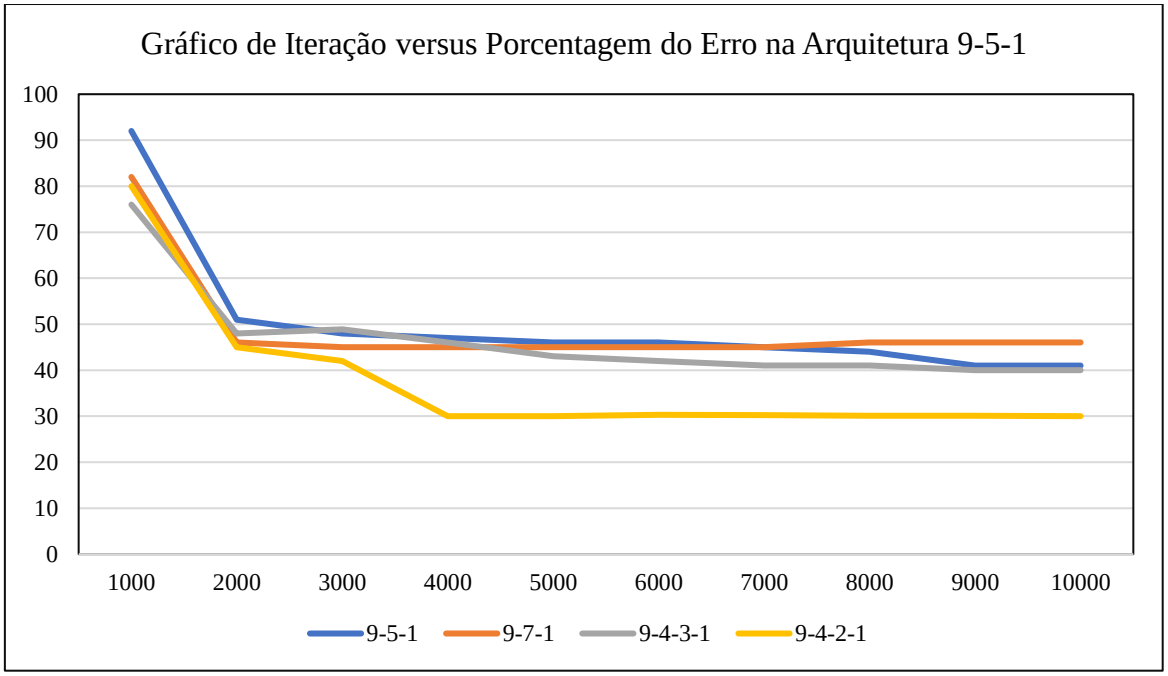
Figura 26- Gráfico dos níveis observado e calculado utilizando arquiteturas de RNA 9-5-1, 9-7-1, 9-4-3-1 e 9-4-2-1.



Fonte: Do autor.

Os modelos 9-5-1, 9-7-1, 9-4-2-1 e o 9-4-3-1 apresentaram 45%, 40% 30% de taxa de erro respectivamente, os resultados desses s modelos, assim como o primeiro testado não convergiam de maneira satisfatória, como pode ser observado na Figura 27.

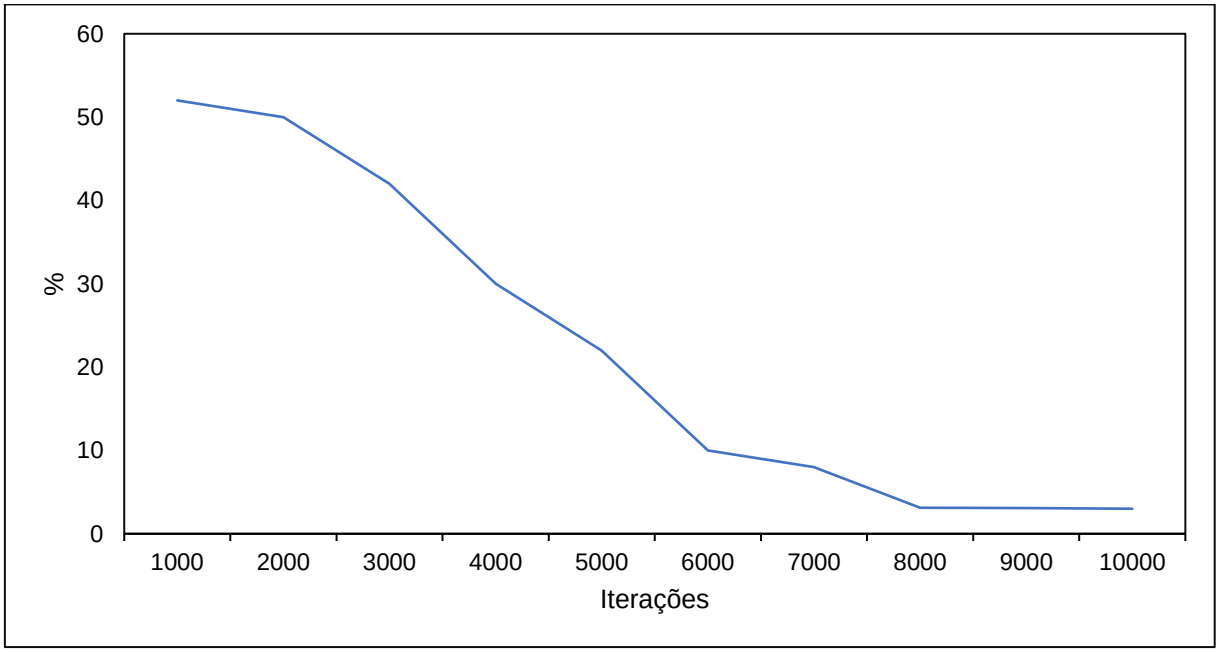
Figura 27- Taxa de erro por iterações nas camadas 9-5-1, 9-7-1, 9-4-3-1 e 9-4-2-1.



Fonte: Do autor.

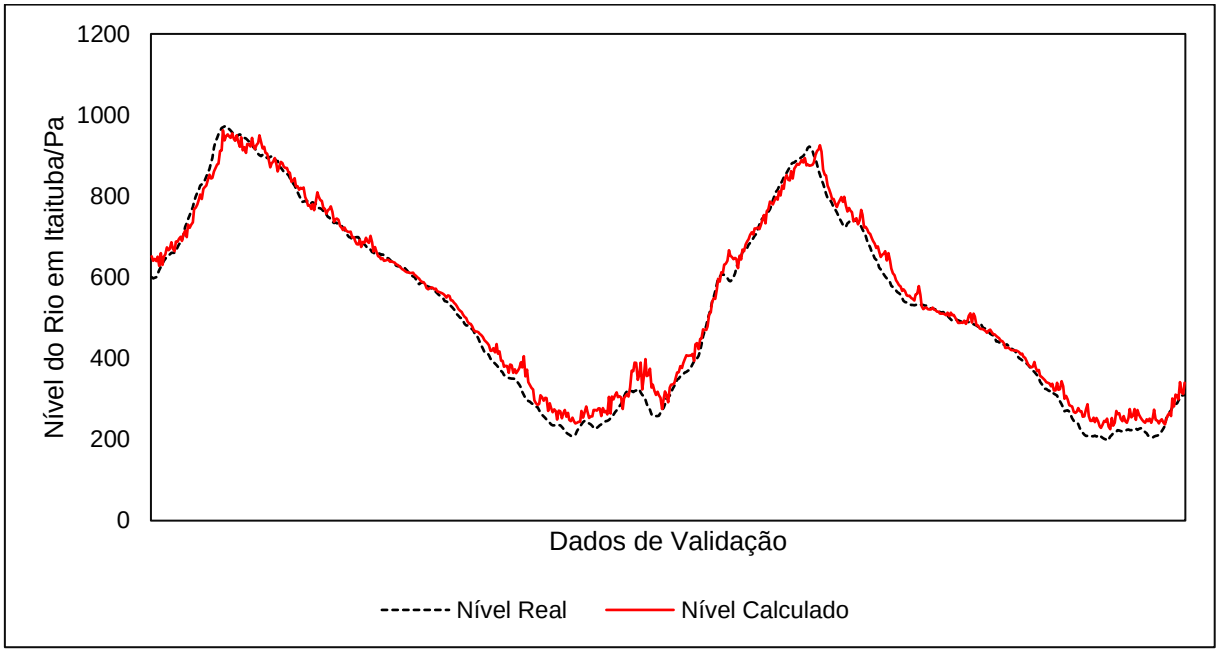
O modelo 9-4-1 foi o último modelo a ser testado e mostrou-se ser a arquitetura perfeita possui três camadas: com uma camada de entrada com nove neurônios; com uma camadas intermediária, com quatro neurônios; e uma camada saída com um neurônio, com a utilização deste modelo, foram feitas mais de 94.000 iterações, obtendo desta forma 3% de taxa de erro (Figura 28), parecido com alguns modelos anteriores este tem uma camada intermediária atingindo assim a convergência com o problema em questão. Na Figura 29, apresenta os resultados obtidos com o cálculo da rede neural.

Figura 28- Taxa de erro por iterações nas camadas 9-4-1.



Fonte: Do autor.

Figura 29- Gráfico dos níveis observado e calculado utilizando arquiteturas de RNA 9-4-1.



Fonte: Do autor.

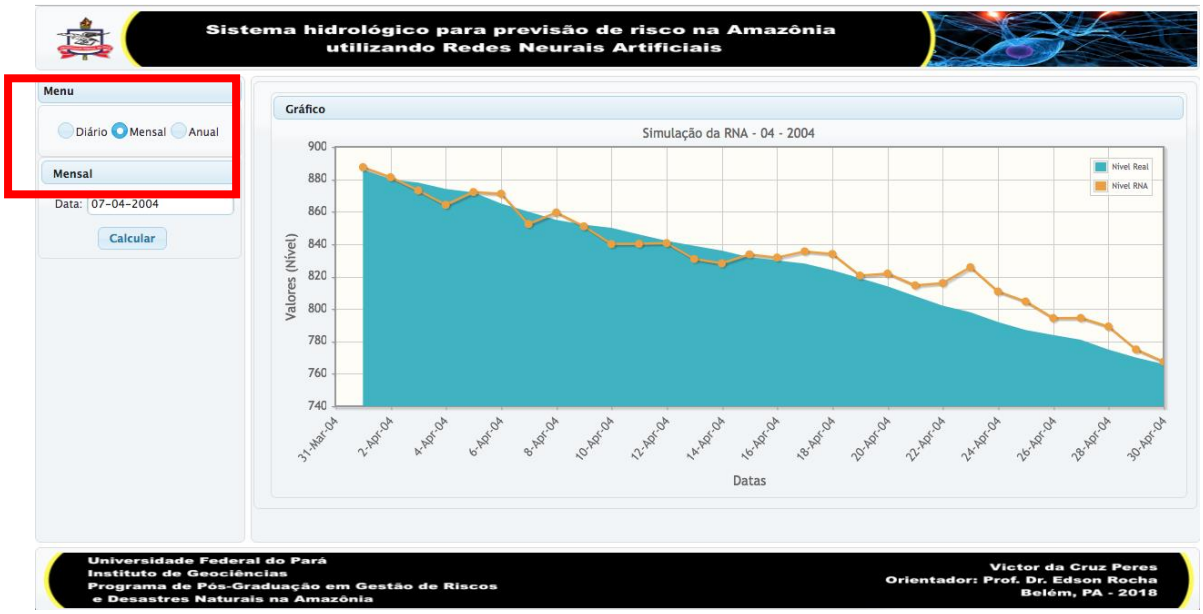
5.3 O sistema

O objetivo principal inicial deste trabalho é criar um sistema de alerta hidrológico a partir da análise do processamento de dados de precipitação utilizando Redes Neurais Artificiais no município de Itaituba/Pa.

Para isso, é necessária principalmente a implementação em uma linguagem de programação do modelo de Rede Neural escolhido entre diversas arquiteturas já comparadas em tópicos anteriores. Para demonstração da Rede Neural, foi implementado um sistema WEB com (Diário, Mensal e Anual) onde é calculado o nível do Rio 6 dias após a data da análise.

O sistema tem como função o cálculo da Rede Neural a partir de um conjunto de dados, já detalhado na metodologia do trabalho. A página WEB apresentara ao usuário uma coluna a esquerda com o título (Menu), onde o usuário irá selecionar a opção desejada, como podemos observar na Figura 30.

Figura 30- Página do sistema de alerta



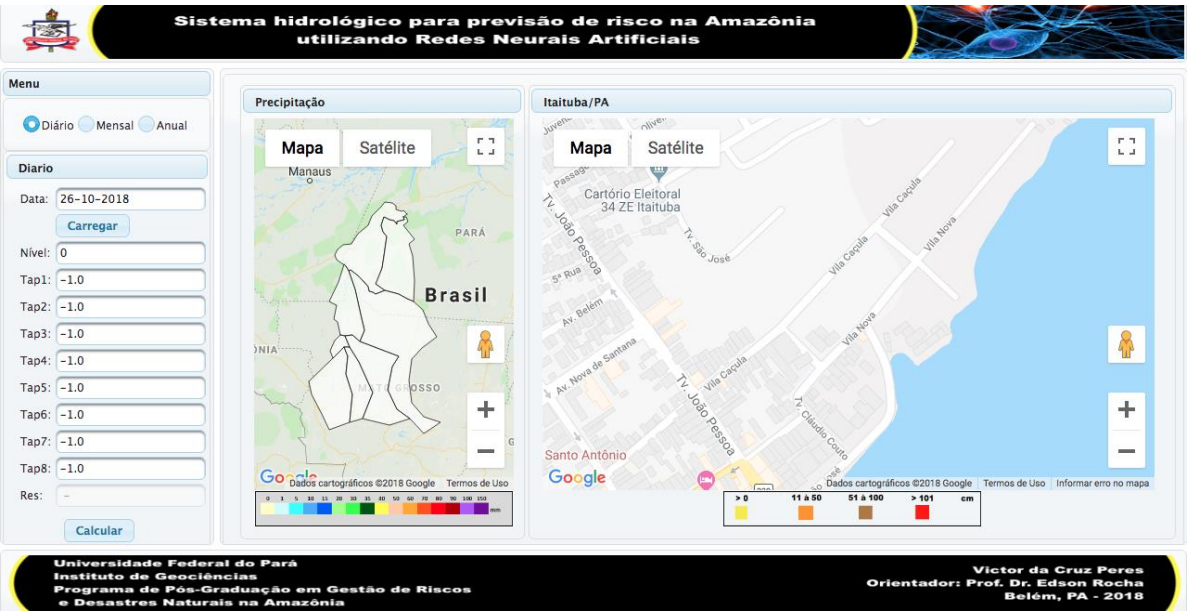
Fonte: Do autor.

Portanto o usuário poderá selecionar 3 opções: Diário, Mensal e Anual:

a) Diário

Ao selecionar a opção (Diário), aparecerá no menu à esquerda as opções (Figura 31).

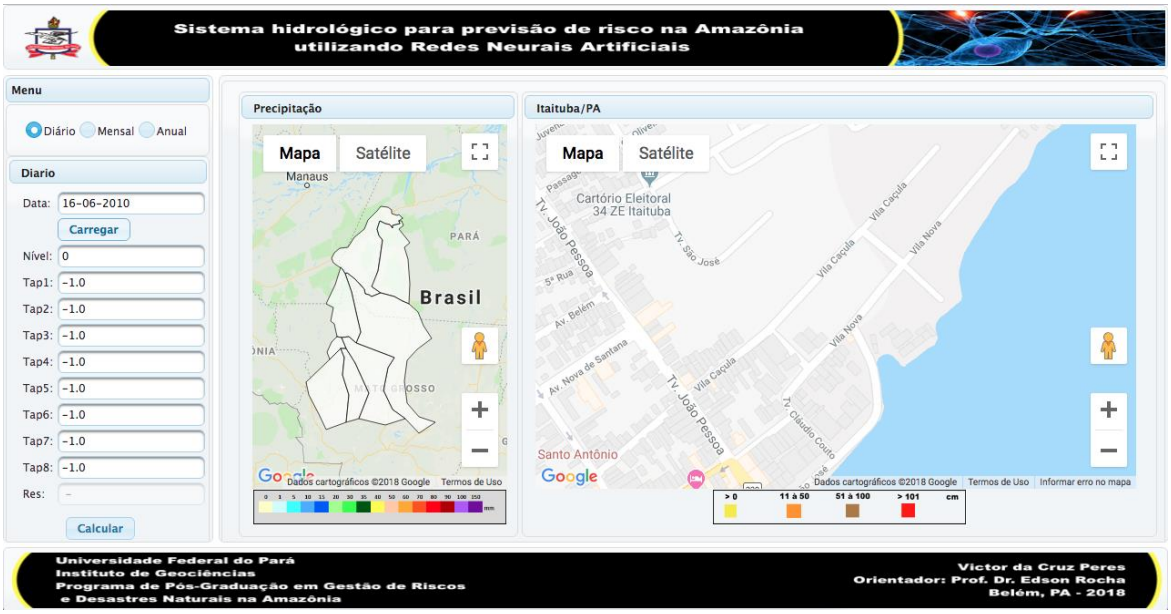
Figura 31- Tela inicial ao selecionar a opção Diário.



Fonte: Do autor.

- **Data:** O usuário poderá selecionar uma data desde 1979 até 2016, onde poderá buscar a quantidade de precipitação nas sub-Bacias e o nível do dia, como podemos observar na Figura 32.

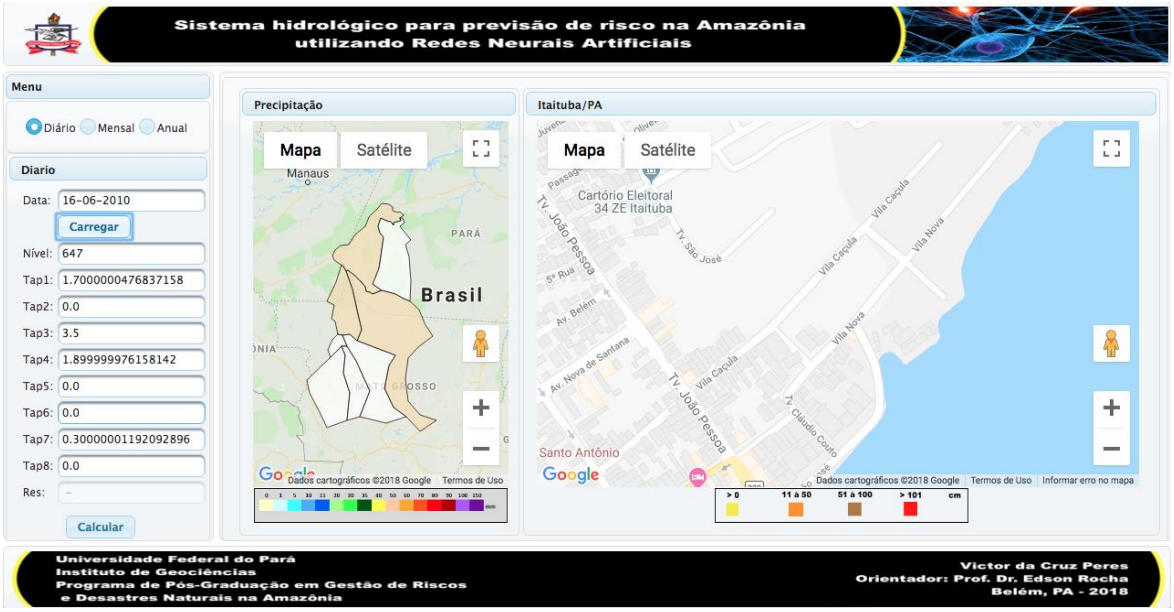
Figura 32- Tela ao selecionar uma Data.



Fonte: Do autor.

- **Botão Carregar:** Botão onde fará a ação de buscar no banco os dados de precipitação e nível, também fará a atualização do mapa de precipitação, de acordo com a legenda de precipitação (Figura 33);

Figura 33- Tela ao clicar no botão Carregar.

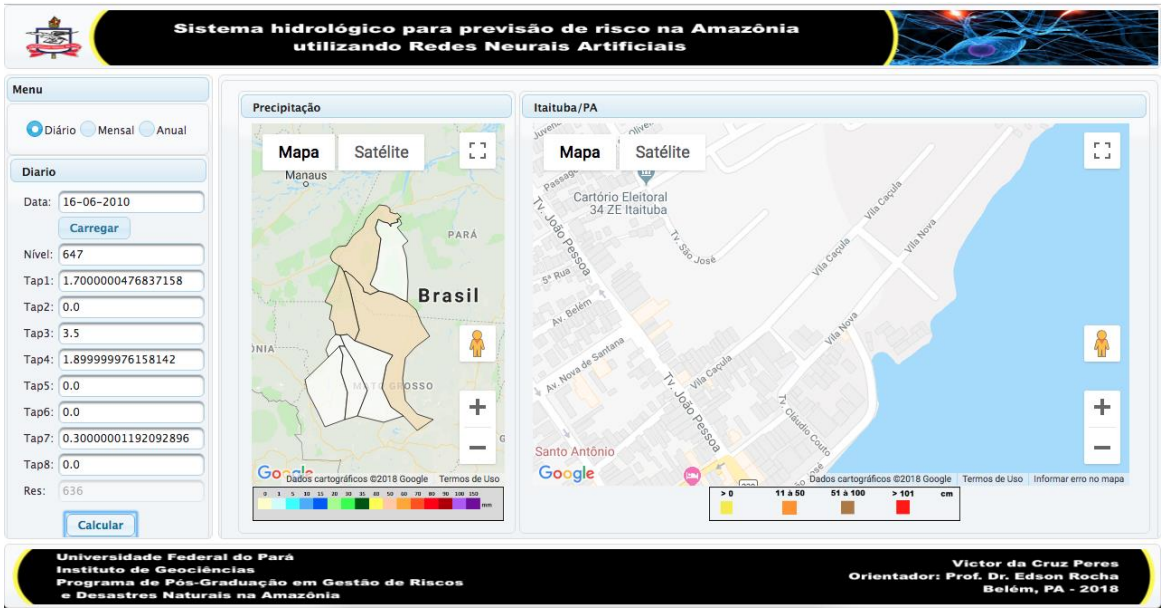


Fonte: Do autor.

- **Nível:** Valor do nível na data selecionada (Figura 34);

- **Tap1:** Quantidade de precipitação na área da sub-Bacia, de acordo com o tempo de concentração real, já verificado neste trabalho (Figura 34);
- **Tap2:** Quantidade de precipitação na área da sub-Bacia, de acordo com o tempo de concentração real, já verificado neste trabalho (Figura 34);
- **Tap3:** Quantidade de precipitação na área da sub-Bacia, de acordo com o tempo de concentração real, já verificado neste trabalho (Figura 34);
- **Tap4:** Quantidade de precipitação na área da sub-Bacia, de acordo com o tempo de concentração real, já verificado neste trabalho (Figura 34);
- **Tap5:** Quantidade de precipitação na área da sub-Bacia, de acordo com o tempo de concentração real, já verificado neste trabalho (Figura 34);
- **Tap6:** Quantidade de precipitação na área da sub-Bacia, de acordo com o tempo de concentração real, já verificado neste trabalho (Figura 34);
- **Tap7:** Quantidade de precipitação na área da sub-Bacia, de acordo com o tempo de concentração real, já verificado neste trabalho (Figura 34);
- **Tap8:** Quantidade de precipitação na área da sub-Bacia, de acordo com o tempo de concentração real, já verificado neste trabalho (Figura 34);

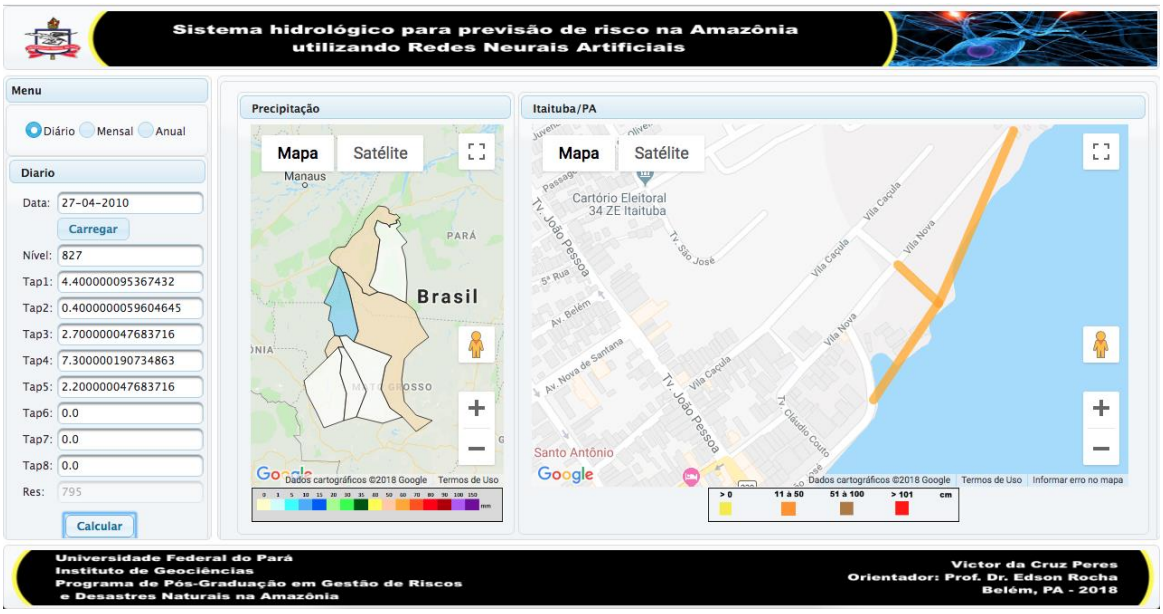
Figura 34- Tela ao clicar no botão Carregar.



Fonte: Do autor.

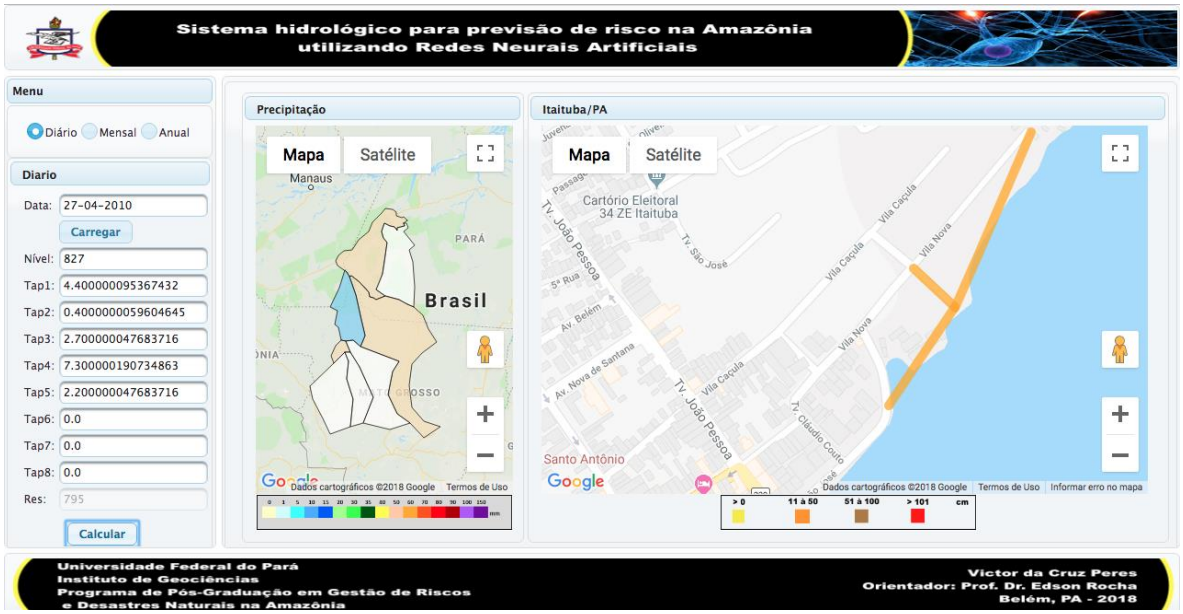
- **Res:** Resultado do nível calculado utilizando a Rede Neural Artificial treinada, esse resultado é para o valor do nível acrescido 6 dias da data selecionada, como apresentada no cálculo do tempo de concentração real (Figura 35);
- **Botão calcular:** O sistema tem como função o cálculo da Rede Neural a partir de um conjunto de dados, já detalhado na metodologia do trabalho. Após o cálculo da RNA, é atualizado o campo Res, como valor de nível calculado também é atualizado o mapa de Itaituba com as ruas que irão sofrer inundação (Figura 36).

Figura 35- Tela ao clicar no botão Calcular.



Fonte: Do autor.

Figura 36- Tela ao clicar no botão Calcular.

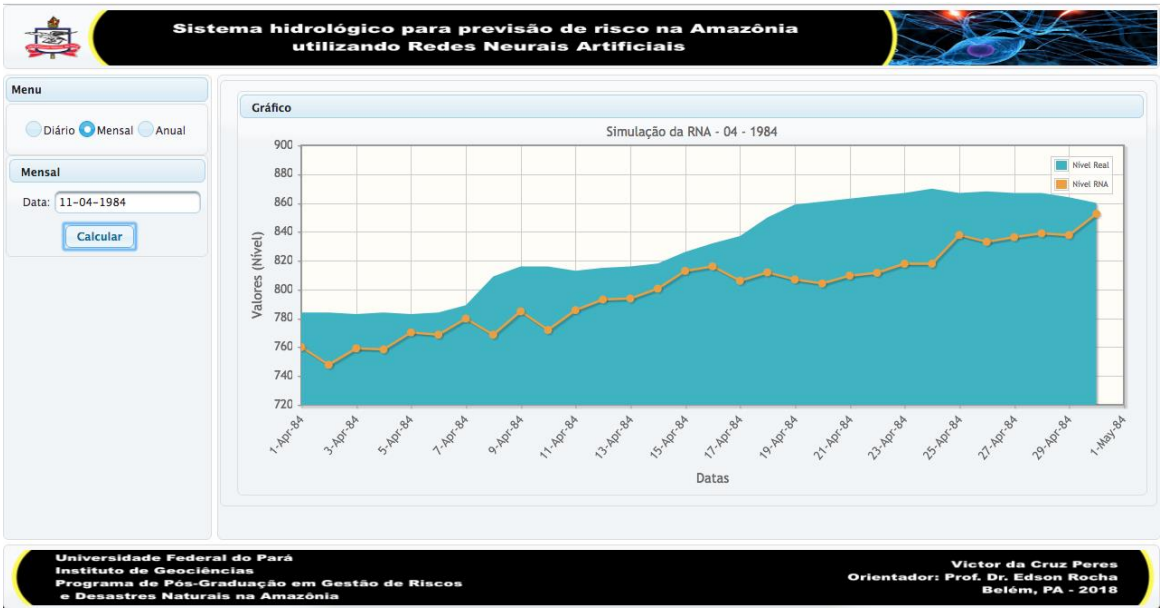


Fonte: Do autor.

b) Mensal

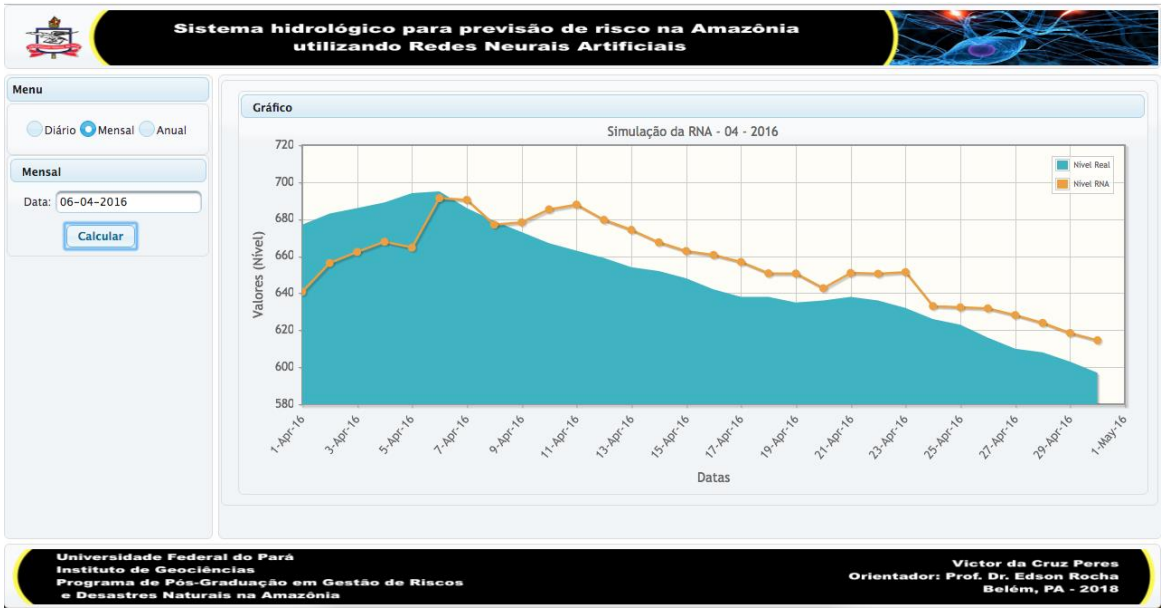
Ao selecionar a opção (Mensal), aparecerá no menu a esquerda a opção para selecionar a Data e o botão Calcular. Ao selecionar a data (11/04/1984), foi feito a seleção no banco de dados todas as datas do mês 04 de 1984 e plotado o gráfico (Figura 37). Na Figura 38 foi selecionado o mês 04 de 2016:

Figura 37- Tela da opção Mensal.



Fonte: Do autor.

Figura 38- Tela da opção Mensal.

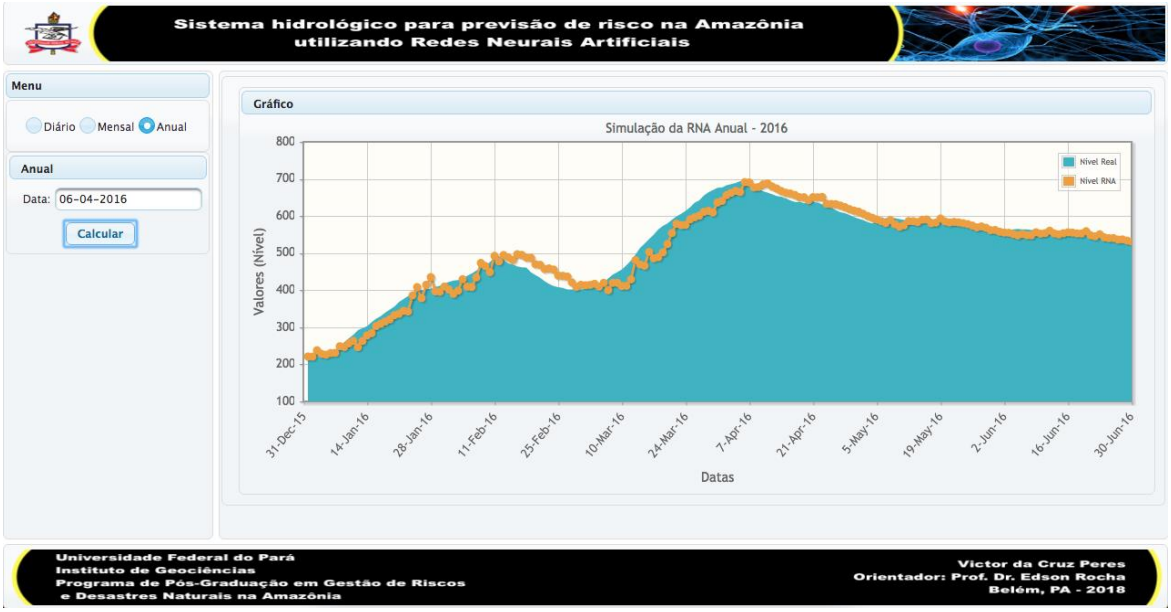


Fonte: Do autor.

c) Anual

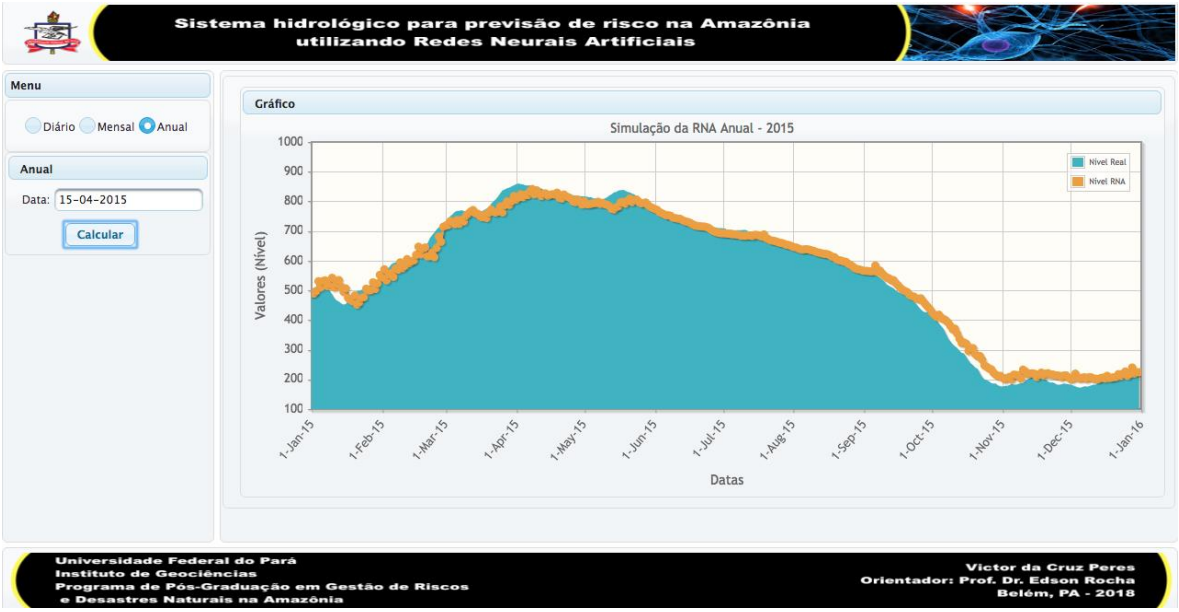
Ao selecionar a opção (Anual), aparecerá no menu a esquerda a opção para selecionar a Data e o botão Calcular. Ao selecionar a data (06/04/2016), foi feito a seleção no banco de dados todas as datas do ano de 2016 e plotado o gráfico (Figura 39). Na Figura 40 foi selecionado o ano de 2015 e na Figura 41 o ano de 1985:

Figura 39- Tela da opção Anual.



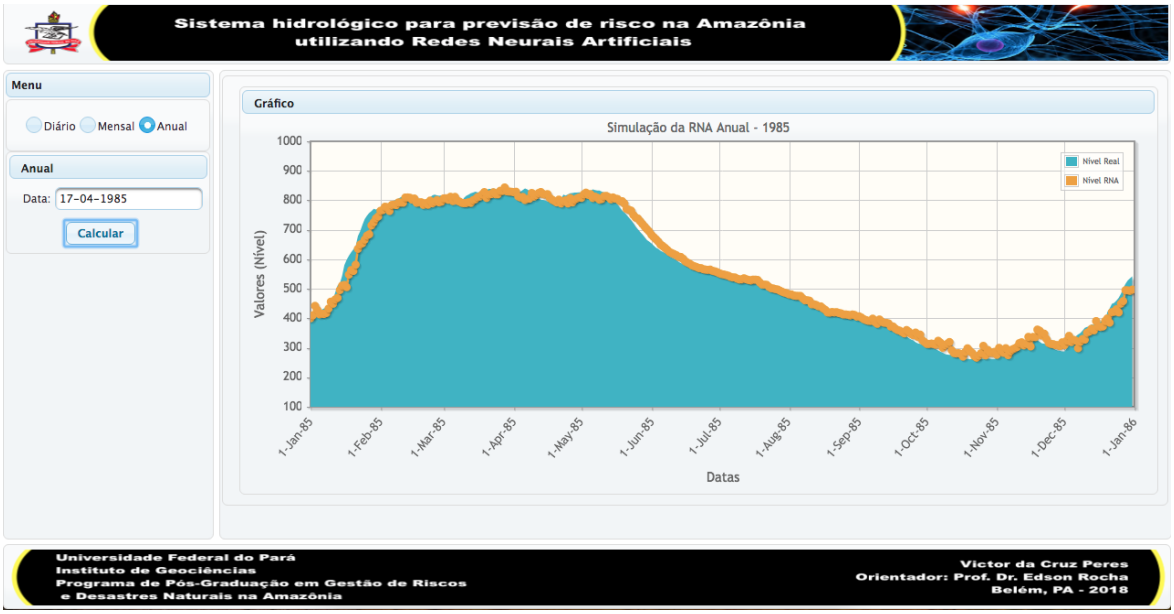
Fonte: Do autor.

Figura 40- Tela da opção Anual.



Fonte: Do autor.

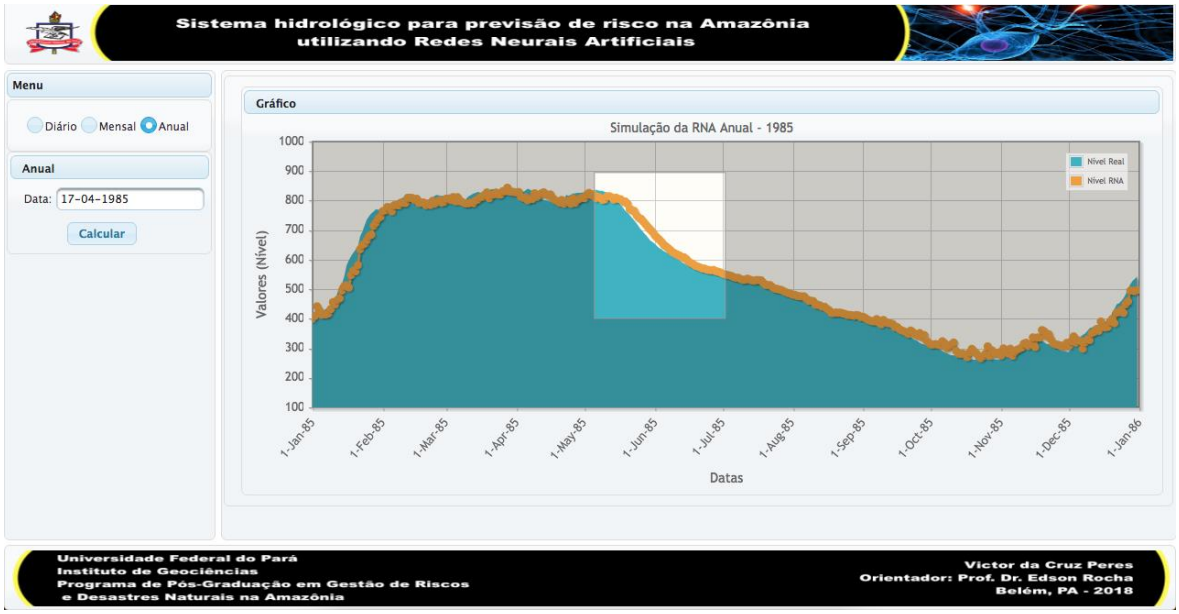
Figura 41- Tela da opção Anual.



Fonte: Do autor.

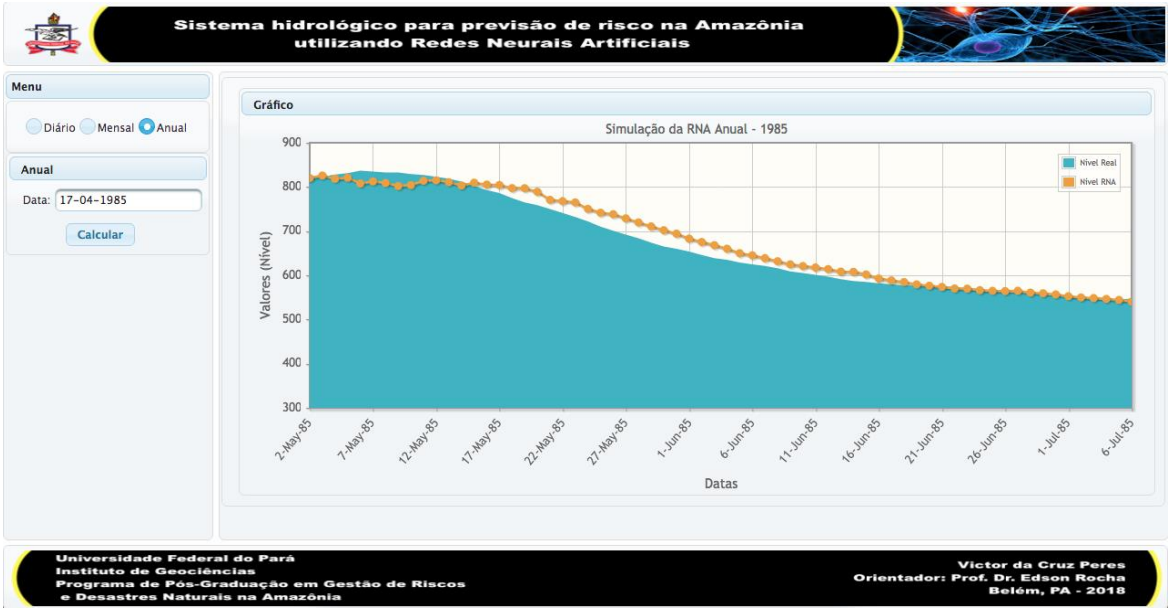
Na Figura 42, mais uma funcionalidade do sistema onde o usuário pode selecionar uma área do gráfico para ampliar como resultado a Figura 43.

Figura 42- Tela da opção Anual.



Fonte: Do autor.

Figura 43- Tela da opção Anual.



Fonte: Do autor.

6 CONCLUSÃO

O objetivo principal deste trabalho foi modelar e apresentar uma rede MLP (*Multilayer Perceptron*) para definir o nível do município de Itaituba/PA com dados de precipitação.

Inicialmente, este trabalho foi proposto sob a ótica de duas problemáticas: a inundação no município de Itaituba/Pa e a modelagem hidrologia utilizando Redes Neurais, descrevendo o processo de seleção dos dados das precipitações nas sub-Bacias, o tempo de concentração do escoamento da água e desenvolvimento de um sistema de alerta.

Em seguida, foram mostradas as principais características das redes biológicas, sendo este o principal ponto de motivação do estudo das RNA. Foram apresentadas as características das RNA e mostrado um histórico da evolução dos estudos relacionados a neuro-computação. A partir de todos estes conceitos, foi introduzida a ideia de aprendizado das RNA, mostrando a forma como as redes aprendem e tentam, num momento seguinte, deduzir respostas a partir do conhecimento adquirido em seu treinamento. A seguir, foram mostrados um modelo de RNA e suas características, o perceptron de múltiplas camadas, que foi utilizado neste trabalho para realizar do resultado de dados em padrões pré-definidos. Um algoritmo de aprendizado muito comum para este modelo de rede é apresentado logo em seguida, o algoritmo Backpropagation. Tendo em vista estes conceitos, o trabalho apresenta, por fim, uma implementação do modelo de rede perceptron de múltiplas camadas treinada com o algoritmo Backpropagation, mostrando várias análises feitas com uma base de dados real.

A partir de todos estes conceitos apresentados, é possível observar o grande poder computacional que as RNA possuem e, desta forma, entender o motivo do grande crescimento dos estudos e aplicações nesta área.

Apesar deste trabalho mostrar as possibilidades de treinamento de uma rede com o algoritmo Backpropagation, este método apresenta algumas desvantagens, como por exemplo ausência de garantia de convergência para uma solução ótima no treinamento e possibilidade de treinamento somente com bases de dados que tenham

uma resposta desejada para seus padrões de entrada. Desta forma, é deixada como sugestão para trabalhos futuros, a implementação de um modelo de RNA com outros algoritmos de treinamento, como por exemplo, o QuickProp e o Rprop (variações do Backpropagation) e o algoritmo c-means (para um aprendizado não-supervisionado).

REFERÊNCIAS

AGÊNCIA NACIONAL DE ÁGUA (ANA). **Região hidrográfica Amazônica**. 2011 Disponível em: http://www2.ana.gov.br/Paginas/portais/bacias/amazonica.aspx. Acesso em: 20 dez. 2017.

ALMEIDA, R.; BARBOSA, P. S. F. Previsão de secas hidrológicas com base em um modelo de redes neurais artificiais. *In*: SIMPÓSIO DE RECURSOS HÍDRICOS DO NORDESTE, 7, 2004, São Luís. ABRH. **Anais**... 1 CD ROM.

BALLINI, R.; FRANÇA, E.; KADOWAKI, M.; SOARES, S.; ANDRADE, M. Modelos de redes neurais e box e jenkins para previsão de vazões médias mensais. *In*: SIMPOSIO BRASILEIRO DE RECURSOS HÍDRICOS, 12., 1997, VItória -ES. [**Anais…**] VItória – ES: ABRH, 1997.

BEALE, R.; JACKSON, T. **Neural computing**: an introduction. Bristol: Adam Hilger, 1990.

BOX, G. E. P.; JENKINS, G. M. **Time series analysis forecasting and control**. Edição revisada. San Francisco: Holden Day, 1976.

BARP, A.R.B.; BARBOSA, P.S.F. Comparação entre modelo hidrológico conceitual chuva-vazão (SMAP) e modelo de redes neurais artificiais (RNA). *In*: SIMPOSIO DE HIDRÁULICA E RECURSOS HÍDRICOS DOS PAÍSES DE LÍNGUA PORTUGUESa, 4., 1999, Coimbra, Portugal. [**Anais..**]. Coimbra, Portugal, 1999.

TUCCI, Carlos E. M. (Org.). **Hidrologia**: ciência e aplicação. [2. ed.]. Porto Alegre: Editora da Universidade Federal do Rio Grande do Sul, [2001]. 943 p. (Coleção ABRH de recursos hídricos, v.4).

BRAGA, A. P.; CARVALHO, A. C. P. L. F.; LUDEMIR, T. B. **Redes neurais artificiais**: teoria e aplicações. LTC, 2000.

BRAGA, A. de P.; PONCE, A. L. de C.; TERESA, L. B. **Redes neurais artificiais**: teorias e aplicações. LTC, 2007.

CANNON, A.J.; WHITFIELD, P.H. Downscaling recent streamflow conditions in British Columbia, Canada using ensemble neural network models. **Journal of Hydrology**, v. 259, p. 136-151, 2002.

CYBENKO, G. Approximation by superpositions of a sigmoid function. **Mathematics of Control, Signals and Systems**, v. 2, p. 303–314, 1989.

DEMUTH, H.; BEALE, M.; HAGAN, M. **Neural network toolbox user's guide. The mathworks**. Sixth printing Version 4, [S.l.: s.n], 2000.

DINIZ, L.S.; CLARKE, R.T. Regionalização de parâmetros de modelo chuva-vazão usando redes neurais. *In*: SIMPÓSIO BRASILEIRO DE RECURSOS HÍDRICOS, 14., 2001, Aracaju / SE, 2001

FAUSSET, L.V; **Fundamentals of neural network**: architecture, algorithm, and application. [S.l.]: Prentice Hall, 1994.

FAVORETO, R. S.; ROHN, M. C.; MINE, M. R. M. A técnica de rede neural artificial aplicada na previsão de vazão. *In*: SIMPÓSIO BRASILEIRO DE RECURSOS HÍDRICOS E SIMPÓSIO DE HIDRÁULICA E RECURSOS HÍDRICOS DOS PAÍSES DE LÍNGUA OFICIAL PORTUGUESA: GESTÃO DE RECURSOS HÍDRICOS, 5., 2001, Aracaju. **O desafio da prática**: anais. Aracaju: ABRH. 1 CD ROM.

FERNÁNDEZ BOU, A. S.; SÁ, R. V. de; CATALDI, M. Flood forecasting in the upper Uruguay River basin. **Natural Hazards**, v. 79, n. 2, p. 1239–1256, 2015.

FIORIN, D. V.; MARTINS, F. R.; SCHUCH, N. J.; PEREIRA, E. B. Aplicações de redes neurais e previsões de disponibilidade de recursos energéticos solares. **Revista Brasileira de Ensino de Física**, v. 33, n. 1, p. 01–20, 2011. Disponível em: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1806-11172011000100009&lng=pt&tlng=pt. Acesso em: 20 dez. 2017.

FREITAS, V.A.F.; RAMEH, I.M.B.; VIEIRA, H.B.G.; ASFORA, M.C.; CIRILO, J.A. Regionalização hidrológica nas bacias 46, 47, 48, 49 do rio São Francisco. *In*: SIMPÓSIO DE RECURSOS HÍDRICOS DO NORDESTE, 6., 2002, Maceió – AL. [**Anais...]** : Maceió – AL : ABRH, 2002. 1 CD-ROM.

GARDNER, M. W.; DORLING, S. R. Artificial neural networks (the multilayer perceptron) – A review of applications in the Atmospheric Sciences. **Atmospheric Environment**, v. 32, p. 2627–2636, 1998.

HAYKIN, S. **Redes neurais**: princípios e práticas. Tradução de Paulo Martins Engel. 2.ed. Porto Alegre: Bookman, 2001.

HOPFIELD, J. J. Neural networks and physical systems with emergent collective properties. **Proc. Nat. Acad. Sci.**, v. 79, p. 2554–2558, 1982.

KHALIL, M.; PANU, U.S.; LENNOX, W.C. Groups and neural networks based streamflow data infilling procedures. **Journal of Hydrology**. v. 241, p. 153-176, 2001.

LEMOS, E. P.; STEINER, M. T. A.; NIEVOLA, J. C. Análise de crédito bancário por meio de redes neurais e árvores de decisão : uma aplicação simples de data mining. **Revista de Administração**, v. 40, n. 3, p. 225–234, 2005.

LENT, R. **Cem bilhões de neurônios**: conceitos fundamentais da neurociência. São Paulo. Atheneu. 2010.

LENCASTRE, A.; FRANCO, F. M. **Lições de hidrologia**. Lisboa: Universidade Nova de Lisboa, Faculdade de ciencias e Tecnologia, 1984.

MCCULLOCH, W. S.; PITTS, W. A. A logical calulus of the ideas in imanente in revous activity. **Bulletin of Mathematical Biophysics**, v. 5, p. 115–133, 1943.

MOLLER, M.F. A Scaled conjugate gradient algorithm for fast supervised learning.

Neural Networks, v. 6, no. 4, p. 525-533, 1993.

MISMKYM, M.; PAPERT, S. **Perceptrons**: na introduction to computational geometry. MIT Press, 1996.

MULLER, A. **Uma previsão de redes neurais artificiais na previsão do mercado acionário**. Florianópolis, 1996.

MULLER, M.; FILL, H. D. Redes neurais aplicadas na propagação de vazões. *In*: SIMPÓSIO BRASILEIRO DE RECURSOS HÍDRICOS, 15, 2003. Curitiba. **Anais…** Curitiba: ABRH. 1CD ROM.

NAYAK, P C et al. A neuro-fuzzy computing technique for modeling hydrological time series. **Journal of Hydrology**, v. 291, n. 1-2, p. 52–66, 2004.

OLIVEIRA, M. A. de; MONTINI, A. de A.; BERGMANN, D. R. Previsão de retornos de ações de empresas dos setores financeiro, alimentos, industrial e de serviços por meio de redes neurais e modelos ARIMA-GARCH. *In*: EnANPAD 2007, [S.l]. **Anais**... [S.l.]: 2007.

OLIVO, A. A.; SILVA, J. D. S.; VIJAYKUMAR, N. L. Previsão de cheias fluviais usando redes neurais artificiais. *In*: SIMPÓSIO DE RECURSOS HÍDRICOS DO NORDESTE, 6, 2002. Maceió. Anais... Maceió: ABRH. CD Rom.

PEARLMUTTER, B. A. Gradient descent: second-order momentum and saturating error. Disponível em: https://papers.nips.cc/paper/454-gradient-descent-second-order-momentum-and-saturating-error.pdf. Acesso em: 20 dez. 2017.

RIEDMILLER, M. **Rprop** - description and implementation details. Tech. Rep. - University of Karlsruhe, 1994.

ROSENBLATT, F. **The perceptron**: a probabilistic model for information storage and organization in the brain. [S.l.]: Spartan Books, 1958.

ROSENBLATT, F. **Principles of neurodynamics**: perceptron and theory of brain mechanisms. [S.l.]: Spartan Books, 1962.

ROSSI, D. J. Previsão da Velocidade dos Ventos por Redes Neurais Artificiais e ARIMA de Box & Jenkins. *In*: CONGRESSO DE MATEMÁTICA APLICADA E COMPUTACIONAl - CMAC, 2013. **Anais**...2013. Disponível em: http://repositorio.unesp.br/handle/11449/111121. Acesso em: 20 dez. 2017.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by backpropagation errors. **Nature**, v. 323, p. 533–536, 1986a.

SANTOS, I.; FILL, H. D.; SUGAI, M. R. V. B.; BUBA, H.; KISHI, R. T.; MARONE, E., LAUTERT, L. F. **Hidrometria aplicada**. Curitiba: Instituto de Tecnologia para o Desenvolvimento, 2001.

SOMMERVILLE, I. **Engenharia de software**. 6. ed., São Paulo: Addison Wesley, 2003, 592p.

SPÖRL, C.; CASTRO, E.; LUCHIARI, A. Aplicação de redes neurais artificiais na construção de modelos de fragilidade ambiental. **RDG Revista do Departamento de Geografia**, USP, v. 21, p. 113–135, 2011. Disponível em: http://citrus.uspnet.usp.br/rdg/ojs/index.php/rdg/article/view/114%5Cnhttp://citrus.uspnet.usp.br/rdg/ojs/index.php/rdg/article/view/114/113. Acesso em: 20 dez. 2017.

VALENÇA, Mêuser Jorge Silva. **Fundamento das redes neurais**. Olinda: Livro Rápido, 2009.

VEMURI, V. R. **Artificial neural networks** – forecsating times series. [S.l]: IEEE Computer Society Press, 1994.

VON ZUBEN, F.J. **Modelos paramétricos e não-paramétricos de redes neurais artificiais e aplicações**. 1996. 244f. Tese (Doutorado) - Faculdade de Engenharia Elétrica e de Computação, Unicamp, 1996.

WIDROW, B.; HOFF, M. E. **Adaptive switching circuits**. Institute of Radio Engineers, Western Electronic Show and Convention, 1960.

ZILOUCHIAN, A.; JAMSHIDI, M. **Intelligent control systems using soft computing methodologies**. Boca Raton, London, New York, Washington, D.C.: CRC Press, 2001.

APÊNDICES

APÊNDICE A - ALGORITMO REDES NEURAIS ARTIFICIAIS

```
package aprendizadomaquina;

import Modelo.RNAss;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;

public class RNA extends Modelo {

    final int IT_MAX = 10000; // N?mero m►ximo de itera►?es
    final int MOSTRAR = 100; // Mostrar erro de treino a cada MOSTRAR itera►?es

    protected MatrizPesos matriz; // Matriz de pesos

    protected int[] nodos; // N?mero de nodos em cada camada

    public RNA() {
    } // Construtor vazio

    public RNA(int entradas, int intermedios, int saidas) // Construtor simples
    // entradas - n?mero de nodos de entrada
    // intermedios - n?mero de nodos da camada interm►dia
    // saidas - n?mero de nodos de sa►da
    {
        nodos = new int[4];
        nodos[0] = entradas;
        nodos[1] = intermedios;
        nodos[2] = intermedios;
        nodos[3] = saidas;
        matriz = new MatrizPesos(nodos);
        matriz.inicializar(-1.0, 1.0);
    }

    public void construirModelo(ConjuntoDados cd) // Cria e treina a RNA
    {
        // criar a rede
```

```
        nodos = new int[3];
        nodos[0] = Casos.numEntradas(cd);
        //nodos[1]=Math.round(nodos[0]/2);
        nodos[1] = 9;
        //nodos[2]=2;
        nodos[2] = Casos.numSaidas(cd);
        // System.out.println("Rede: "+Casos.numEntradas(cd)+" - "+Math.round(nodos[0]/2)+" -
"+Casos.numSaidas(cd));
        matriz = new MatrizPesos(nodos);

        matriz.inicializar(-1.0, 1.0);
        matriz.imprimir();

        Casos c = new Casos(cd, Casos.ZERO_UM);
        treinar(IT_MAX, 0.2, c); // treinar a RNA
        matriz.imprimir();
    }

    public void construirModelo2(ConjuntoDados cd) // Cria e treina a RNA
    {
        // criar a rede
        nodos = new int[3];
        nodos[0] = Casos.numEntradas(cd);
        nodos[1] = 4;
        nodos[2] = 1;
        matriz = new MatrizPesos(nodos);

        matriz.inicializar2();

        //Casos c=new Casos(cd,Casos.ZERO_UM);
        // treinar(IT_MAX,0.2,c); // treinar a RNA
        //matriz.imprimir();
    }

    public void imprimir() // Imprime a RNA no ecr?
    {
        System.out.println("Nodos :" + nodos[0]);
        System.out.println("-" + nodos[1] + "-" + nodos[2]);
        matriz.imprimir();
    }
```

```
public void activar(double[] entrada, double[] saida) // Computa a sa►da de uma RNA
// entrada - Vector com as entradas da rede
// saida - Vector com as sa►das da rede
{
    matriz.activar(nodos, entrada, saida);
}

public void activar(ConjuntoDados cd) // Imprime no ecr? todas as sa►das para o Conjunto de Dados
cd
{
    System.out.println("\nSaidas da RNA\nExemplo");
    System.out.println("\t: \tSaida\t: \tRNA");
    Atributo a = cd.saida();
    for (int i = 0; i < cd.numExemplos(); i++) {
        System.out.print(i + "\t: \t");

        if (a.tipo == Atributo.NUMERICO) {
            System.out.print(cd.exemplo(i).valorSaida() + " \t");
            System.out.print(prever(cd, cd.exemplo(i)) + "\n");
        } else {
            System.out.print(a.valorNominal(
                    (int) cd.exemplo(i).valorSaida()) + "\t: \t");
            System.out.print(classificar(cd, cd.exemplo(i)) + "\n");
        }
    }
}

public double prever(ConjuntoDados cd, Exemplo ex) // Prev► a sa►da de uma RNA
{
    double[] entrada = new double[nodos[0]];
    double[] saida = new double[nodos[2]];
    Casos.codificar(cd, ex, entrada);
    activar(entrada, saida);
    return Casos.descodificar(cd, Casos.ZERO_UM, saida);
}

public void retroPropagacao(double taxa, Casos c) // Treina a rede com uma itera►?o do algoritmo
de retropropaga►?o
// taxa - a taxa de aprendizagem (valor entre 0,0 e 1,0)
```

```
// c - o conjunto de casos de treino
{
    MatrizPesos E = new MatrizPesos(nodos);

    for (int i = 0; i < c.casos(); i++) {
        matriz.gradiente(nodos, c.entrada(i), c.saida(i), E);
        matriz.retroPropagacao(taxa, E);
    }
}

static public double quadradoErros(double[] a, double[] b) // Calcula o quadrado dos erros
{
    double res = 0.0;
    for (int i = 0; i < a.length; i++) {
        res += (a[i] - b[i]) * (a[i] - b[i]);
    }
    return res;
}

public double erro(Casos c) // Calcula o erro de treino (RMQE)
{
    double res = 0.0;
    double[] saida = new double[nodos[2]];
    for (int i = 0; i < c.casos(); i++) {
        activar(c.entrada(i), saida);
        res += quadradoErros(saida, c.saida(i));
    }
    return Math.sqrt(res / (c.casos() * nodos[2]));
}

public void treinar(int max, double taxa, Casos c) // Treina a rede durante o m►ximo de max
itera►?es
// max - n?mero m►ximo de itera►?es
// taxa - taxa de aprendizagem (valor entre 0,0 e 1,0)
// c - conjunto de casos de treino
{
    System.out.println("\nTreino da RNA:\nIteracao\t: \tRMQE");
    int i;
    for (i = 0; i < max; i++) {
        retroPropagacao(taxa, c);
```

```
        if (i % MOSTRAR == 0) {
            System.out.println(i + "\t: \t" + erro(c));
        }
    }
    System.out.println("Erro final:\n" + i + "\t: \t" + erro(c));
}

public static void main(String[] args) // Exemplo de utiliza►?o
{
    try {

        ConjuntoDados d = new ConjuntoDados("mestrado");
        Casos c = new Casos(d, Casos.ZERO_UM);
        //c.imprimir();
        RNA rna = new RNA();
        rna.construirModelo(d);
        rna.activar(d);

    } catch (Exception e) {
        System.out.println(e);
    }
}

public ArrayList<RNAss> calcularDiario(ArrayList<RNAss> rnas) throws Exception {
    ConjuntoDados d = null;
    d = new ConjuntoDados("mestrado", rnas);
    Casos c = new Casos(d, Casos.ZERO_UM);



    construirModelo2(d);

    ArrayList<RNAss> rnas2 = activar2(d, rnas);

    return rnas2;
}

private ArrayList<RNAss> activar2(ConjuntoDados d, ArrayList<RNAss> rnas) {

    Atributo a = d.saida();
```

```
        for (int i = 0; i < d.numExemplos(); i++) {

            if (a.tipo == Atributo.NUMERICO) {

                double res = prever(d, d.exemplo(i));

                rnas.get(i).setResposta(res);
            }
        }
        return rnas;
    }

}
```

APÊNDICE B - ALGORITMO DE TREINAMENTO BACKPROPAGATION

```
package aprendizadomaquina;

public class MatrizPesos {

    protected double[][] w; // Matriz de pesos

    public MatrizPesos(MatrizPesos A) // Construtor a partir de outra matriz de pesos
    {
        int Y = A.w.length;
        int X = A.w[0].length;
        w = new double[Y][X];
        for (int i = 0; i < Y; i++) {
            for (int j = 0; j < X; j++) {
                w[i][j] = A.w[i][j];
            }
        }
    }

    public MatrizPesos(int[] n) // Construtor a partir dos nodos
    // n - Vector de nodos, com o numero de entradas, interm►dios e sa►das
    {
        int Y = n[1] + n[2];
        int X = 1 + n[0] + n[1] + n[2];
        w = new double[Y][X];
        //   System.out.println("Y: "+Y+" - "+X);
        for (int i = 0; i < Y; i++) {
            for (int j = 0; j < X; j++) {
                if (i < n[1] && j <= n[0]) {
                    //     System.out.println("w1 "+i+"-"+j);
                    w[i][j] = 0.0;
                } else if (j == 0) {
                    //     System.out.println("w2 "+i+"-"+j);
                    w[i][j] = 0.0;
                } else if (j > n[0] && j <= (n[0] + n[1]) && i >= n[1]) {
                    //     System.out.println("w3 "+i+"-"+j);
                    w[i][j] = 0.0;
                } else {
                    //System.out.println("w"+i+"-"+j);
```

```
                w[i][j] = Double.NaN;
            }
        }
    }
}

static public double sigmoid(double x) // Calcula a fun►?o sigmoid
{
    return 1.0 / (1.0 + Math.exp(-x));
}

static public double derivar(double x) // Calcula a derivada da fun►?o sigmoid
{
    double aux = sigmoid(x);
    return aux * (1.0 - aux);
}

public void inicializar(double a, double b) // Inicia de modo aleat►rio todos os pesos, usa a gama
[a,b[
// a - limite inferior, b - limite superior
{
    int Y = w.length;
    int X = w[0].length;
    //  System.out.println("teste: "+Y+" - "+X);
    for (int i = 0; i < Y; i++) {
        for (int j = 0; j < X; j++) {
            if (Double.isNaN(w[i][j]) == false) {
                w[i][j] = (b - a) * Math.random() + a;
            }
        }
    }
}

public void imprimir() // Imprime no ecr? uma matriz de pesos
{
    int Y = w.length;
    int X = w[0].length;
    for (int i = 0; i < Y; i++) {
        for (int j = 0; j < X; j++) {
            if (Double.isNaN(w[i][j])) {
```

```
                System.out.print("- \t");
            } else {
                System.out.print(Casos.valor(w[i][j], 3) + " \t");
            }
        }
        System.out.println("");
    }
}

public double produto(int linha, double[] v) // Calcula o produto interno entre uma linha da matriz e
um vector
{
    double res = 0.0;
    int X = w[0].length;
    for (int i = 0; i < X; i++) {
        if (Double.isNaN(w[linha][i]) == false) {
            res += w[linha][i] * v[i];
        }
    }
    return res;
}

public void activar(int[] n, double[] entrada, double[] saida) // Activa a sa►da da MLP
// n - Vector com o n?mero de nodos por camada
// entrada - Vector de entrada, saida - Vector com as sa►das (valor de retorno)
{
    double[] u = new double[n[0] + n[1] + n[2]];
    activarUi(n, entrada, u); // activa►?es sem sigmoid
    for (int i = 0; i < n[2]; i++) {
        saida[i] = sigmoid(u[n[0] + n[1] + i]);
    }
}

public void activarUi(int n[], double[] entrada, double[] u) // Calcula os ui da MLP (activa►?es
pr►vias ? fun►?o sigmoid)
// n - vector com o n?mero de nodos por camada
// entrada - Vector de entrada, u - vector com as saidas (valor de retorno)
{
    int i, j;
    int Y = w.length;
```

```
    int X = w[0].length;
    double[] aux = new double[X]; // todas activa►?es

    for (i = 0; i < X; i++) // preencher vector auxiliar
    {
        if (i == 0) {
            aux[0] = 1.0; // bias
        } else if (i <= n[0]) {
            aux[i] = entrada[i - 1];
            u[i - 1] = entrada[i - 1];
        } else {
            aux[i] = 0.0;
            u[i - 1] = 0.0;
        }
    }

    for (i = 0, j = n[0] + 1; i < Y; i++, j++) // propagar pelas camadas
    {
        u[j - 1] = produto(i, aux);
        aux[j] = sigmoid(u[j - 1]);
    }
}

public void gradiente(int n[], double[] entrada, double[] saida, MatrizPesos E) // Calcula o gradiente
// n - Vector com o n?mero de nodos por camada
// entrada - Vector de entrada, saida - Vector com as sa►das
// E - matriz com o gradiente calculado
{
    int i, j;
    int Y = w.length;
    double[] u = new double[n[0] + n[1] + n[2]];
    double[] g = new double[Y];

    activarUi(n, entrada, u); // propaga►?o em frente
    for (i = Y - 1; i >= 0; i--) // retropropaga►?o
    {
        if (i >= n[1]) // i - camada de sa►da
        {
            g[i] = sigmoid(u[i + n[0]]) - saida[i - n[1]];
        } else // outras camadas
```

```
            {
                for (j = 0, g[i] = 0.0; j < Y; j++) {
                    if (Double.isNaN(w[j][i + n[0] + 1]) == false) {
                        g[i] += g[j] * w[j][i + n[0] + 1];
                    }
                }
            }
            g[i] = g[i] * derivar(u[i + n[0]]);
            E.w[i][0] = g[i] + E.w[i][0]; // conex?o de bias
            for (j = 1; j < (i + n[0] + 1); j++) // outras conex?es
            {
                if (Double.isNaN(w[i][j]) == false) {
                    if (j <= n[0]) // n?o tem sigmoid
                    {
                        E.w[i][j] = g[i] * u[j - 1] + E.w[i][j];
                    } else // j ⬦ nodo interm⬦dio, tem sigmoid
                    {
                        E.w[i][j] = g[i] * sigmoid(u[j - 1]) + E.w[i][j];
                    }
                }
            }
        }
    }

    public void retroPropagacao(double taxa, MatrizPesos E) // Efectua a actualiza⬦?o do algoritmo de
retropropaga⬦?o
    // taxa - taxa de aprendizagem (valor entre 0,0 e 1,0)
    // E - matriz com o gradiente calculado
    {
        int Y = w.length;
        int X = w[0].length;
        for (int i = 0; i < Y; i++) {
            for (int j = 0; j < X; j++) {
                if (Double.isNaN(w[i][j]) == false) {
                    w[i][j] = w[i][j] - taxa * E.w[i][j];
                    E.w[i][j] = 0.0;
                }
            }
        }
    }
```

```
void inicializar2() {
    //carregarMatriz

    w[0][0] = 3.056;
    w[0][1] = -1.205;
    w[0][2] = 0.203;
    w[0][3] = 0.458;
    w[0][4] = 1.129;
    w[0][5] = 0.341;
    w[0][6] = 1.516;
    w[0][7] = 0.574;
    w[0][8] = 0.442;
    w[0][9] = 1.062;
    w[0][10] = Double.NaN;
    w[0][11] = Double.NaN;
    w[0][12] = Double.NaN;
    w[0][13] = Double.NaN;
    w[0][14] = Double.NaN;

    w[1][0] = 3.528;
    w[1][1] = 2.215;
    w[1][2] = 1;
    w[1][3] = 0.617;
    w[1][4] = 0.149;
    w[1][5] = -0.428;
    w[1][6] = 0.2;
    w[1][7] = -0.076;
    w[1][8] = -0.089;
    w[1][9] = 1.391;
    w[1][10] = Double.NaN;
    w[1][11] = Double.NaN;
    w[1][12] = Double.NaN;
    w[1][13] = Double.NaN;
    w[1][14] = Double.NaN;

    w[2][0] = 3.934;
    w[2][1] = 2.284;
    w[2][2] = -0.455;
    w[2][3] = -0.109;
```

```
w[2][4] = 0.892;
w[2][5] = 0.923;
w[2][6] = 1.14;
w[2][7] = 0.656;
w[2][8] = 0.391;
w[2][9] = -0.434;
w[2][10] = Double.NaN;
w[2][11] = Double.NaN;
w[2][12] = Double.NaN;
w[2][13] = Double.NaN;
w[2][14] = Double.NaN;

w[3][0] = -2.105;
w[3][1] = 3.943;
w[3][2] = 0.233;
w[3][3] = 0.177;
w[3][4] = 0.558;
w[3][5] = 0.271;
w[3][6] = 0.304;
w[3][7] = 0.168;
w[3][8] = 0.172;
w[3][9] = 0.316;
w[3][10] = Double.NaN;
w[3][11] = Double.NaN;
w[3][12] = Double.NaN;
w[3][13] = Double.NaN;
w[3][14] = Double.NaN;

w[4][0] = -2.653;
w[4][1] = Double.NaN;
w[4][2] = Double.NaN;
w[4][3] = Double.NaN;
w[4][4] = Double.NaN;
w[4][5] = Double.NaN;
w[4][6] = Double.NaN;
w[4][7] = Double.NaN;
w[4][8] = Double.NaN;
w[4][9] = Double.NaN;
w[4][10] = -1.24;
w[4][11] = 1.85;
```

```
        w[4][12] = 1.922;
        w[4][13] = 3.481;
        w[4][14] = Double.NaN;

    }

}
```

APÊNDICE C - CÓDIGO FONTE: INDEX.XHTML

```
package Controlador;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import DAO.DAO;
import Modelo.Nivel;
import Modelo.Precipitacao;
import Modelo.RNAss;
import Modelo.Ruas;
import aprendizadomaquina.RNA;
import java.sql.SQLException;
import java.text.DecimalFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.primefaces.event.SlideEndEvent;
import org.primefaces.model.chart.AxisType;
import org.primefaces.model.chart.DateAxis;
import org.primefaces.model.chart.LineChartModel;
import org.primefaces.model.chart.LineChartSeries;
import org.primefaces.model.map.DefaultMapModel;
import org.primefaces.model.map.LatLng;
import org.primefaces.model.map.MapModel;
import org.primefaces.model.map.Polygon;
import org.primefaces.model.map.Polyline;

/**
 *
 * @author victorperes
 */
@ManagedBean
@SessionScoped
public class Controlador {

    private String tipo = "Diario";
```

```
private boolean temSelecao = false;
private boolean temDiario = true;
private boolean temMensal = false;
private boolean temAnual = false;
private Date dataSelecao = new Date();
private int nivel;
private double tap1 = -1;
private double tap2 = -1;
private double tap3 = -1;
private double tap4 = -1;
private double tap5 = -1;
private double tap6 = -1;
private double tap7 = -1;
private double tap8 = -1;
private int saidaNivel;

public double getTap1() {
    return tap1;
}

public void setTap1(double tap1) {
    this.tap1 = tap1;
}

public double getTap2() {
    return tap2;
}

public void setTap2(double tap2) {
    this.tap2 = tap2;
}

public double getTap3() {
    return tap3;
}

public void setTap3(double tap3) {
    this.tap3 = tap3;
}
```

```
public double getTap4() {
    return tap4;
}

public String duasCasas(double numero1) {
    DecimalFormat formato = new DecimalFormat("#.##");
    return Double.valueOf(formato.format(numero1)) + "";
}

public String imprimir() {
    if (temDiario) {
        if (rnas.isEmpty()) {
            return " - ";
        } else {
            return (int) rnas.get(0).getResposta() + "";
        }
    } else {
        return " - ";
    }
}

public void setTap4(double tap4) {
    this.tap4 = tap4;
}

public double getTap5() {
    return tap5;
}

public void setTap5(double tap5) {
    this.tap5 = tap5;
}

public double getTap6() {
    return tap6;
}

public void setTap6(double tap6) {
    this.tap6 = tap6;
}
```

```
public double getTap7() {
    return tap7;
}

public void setTap7(double tap7) {
    this.tap7 = tap7;
}

public double getTap8() {
    return tap8;
}

public void setTap8(double tap8) {
    this.tap8 = tap8;
}

public String getConsole() {
    return tipo;
}

public int getNivel() {
    return nivel;
}

public void setNivel(int nivel) {
    this.nivel = nivel;
}

public void setConsole(String tipo) {

    this.tipo = tipo;
}

public void selecionarTipo() {
    if (tipo.equalsIgnoreCase("Diario")) {
        temSelecao = true;
        temDiario = true;
        temMensal = false;
        temAnual = false;
```

```
    } else if (tipo.equalsIgnoreCase("Mensal")) {
        temSelecao = true;
        temDiario = false;
        temMensal = true;
        temAnual = false;
    } else if (tipo.equalsIgnoreCase("Anual")) {
        temSelecao = true;
        temDiario = false;
        temMensal = false;
        temAnual = true;
    } else {
        temSelecao = false;
    }

}

public Date getDataSelecao() {
    return dataSelecao;
}

public void setDataSelecao(Date dataSelecao) {
    this.dataSelecao = dataSelecao;
}

public String getTipo() {
    return tipo;
}

public void setTipo(String tipo) {
    this.tipo = tipo;
}

public boolean isTemSelecao() {
    return temSelecao;
}

public void setTemSelecao(boolean temSelecao) {
    this.temSelecao = temSelecao;
}
```

```
public boolean isTemDiario() {

    return temDiario;
}

public void setTemDiario(boolean temDiario) {
    this.temDiario = temDiario;
}

public boolean isTemMensal() {
    return temMensal;
}

public void setTemMensal(boolean temMensal) {
    this.temMensal = temMensal;
}

public boolean isTemAnual() {
    return temAnual;
}

public void setTemAnual(boolean temAnual) {
    this.temAnual = temAnual;
}

//carregar dados
public void carregarDadosDiario() throws ClassNotFoundException, SQLException {

    int saidaNivel = 0;//0
    dao = new DAO();

    boolean ok = true;

    Calendar c = Calendar.getInstance();
    Calendar ca = Calendar.getInstance();

    c.setTime(dataSelecao);
    ca = c;
    Date d = new Date();
    d.setTime(c.getTimeInMillis());
```

```
Date DataTemp = new Date();

ok = true;
ca.setTime(d);
//  ca.add(Calendar.DATE, +1);
d.setTime(ca.getTimeInMillis());

Nivel nivel;
Precipitacao precipitacao;
/*if (ok) {
        //nivel de resposta; dia 13
        nivel = obterNivel(d, dao);
        if (nivel == null) {
            ok = false;
        } else {
            if (nivel.getNivel() == 0) {
                ok = false;
            } else {
                saidaNivel = nivel.getNivel() + "";
            }
        }
    }*/

if (ok) {
    //niv e tap1
    c.add(Calendar.DATE, -6);
    DataTemp.setTime(c.getTimeInMillis());
    nivel = obterNivel(DataTemp, dao);
    if (nivel == null) {
        ok = false;
    } else {
        if (nivel.getNivel() == 0) {
            ok = false;
        } else {
            this.nivel = nivel.getNivel();
        }
    }
    precipitacao = obterPrecipitacao(DataTemp, dao);

    if (precipitacao == null) {
```

```
        ok = false;
    } else {
        tap1 = precipitacao.getTap01();
    }
}

if (ok) {
    //tap4
    c.add(Calendar.DATE, -2);
    DataTemp.setTime(c.getTimeInMillis());

    precipitacao = obterPrecipitacao(DataTemp, dao);
    if (precipitacao == null) {
        ok = false;
    } else {
        tap4 = precipitacao.getTap04();
    }
}

if (ok) {
    //tap5 e tap2
    c.add(Calendar.DATE, -1);
    DataTemp.setTime(c.getTimeInMillis());

    precipitacao = obterPrecipitacao(DataTemp, dao);
    if (precipitacao == null) {
        ok = false;
    } else {
        tap5 = precipitacao.getTap05();
        tap2 = precipitacao.getTap02();
    }
}

if (ok) {
    //tap7 e tap8
    c.add(Calendar.DATE, -2);
    DataTemp.setTime(c.getTimeInMillis());

    precipitacao = obterPrecipitacao(DataTemp, dao);
    if (precipitacao == null) {
```

```
            ok = false;
        } else {
            tap7 = precipitacao.getTap07();
            tap8 = precipitacao.getTap08();
        }
    }

    if (ok) {
        //tap3 e tap6
        c.add(Calendar.DATE, -1);
        DataTemp.setTime(c.getTimeInMillis());

        precipitacao = obterPrecipitacao(DataTemp, dao);
        if (precipitacao == null) {
            ok = false;
        } else {
            tap3 = precipitacao.getTap03();
            tap6 = precipitacao.getTap06();
        }
    }

    if (ok) {
        System.out.println("," + tap1 + "," + tap2 + "," + tap3 + "," + tap4 + "," + tap5 + "," + tap6 + "," +
tap7 + "," + tap8 + "," + saidaNivel);
    }
    construtorMapa1();

  }
  SimpleDateFormat formato = new SimpleDateFormat("yyyy-MM-dd");
  SimpleDateFormat formato2 = new SimpleDateFormat("yyyy");
  SimpleDateFormat formato3 = new SimpleDateFormat("MM - yyyy");

  public Nivel obterNivel(Date d, DAO dao) throws SQLException, ClassNotFoundException {

    String data1 = formato.format(d);

    Calendar c = Calendar.getInstance();
    c.setTime(d);
    c.add(Calendar.DATE, +1);
    Date da = new Date();
```

```
        da.setTime(c.getTimeInMillis());
        String data2 = formato.format(da);

        return dao.buscarNivel(data1, data2);

    }

    public Precipitacao obterPrecipitacao(Date d, DAO dao) throws SQLException,
ClassNotFoundException {

        String data1 = formato.format(d);

        Calendar c = Calendar.getInstance();
        c.setTime(d);
        c.add(Calendar.DATE, +1);
        Date da = new Date();
        da.setTime(c.getTimeInMillis());
        String data2 = formato.format(da);
        return dao.buscarPreciptacao(data1, data2);

    }
    ArrayList<RNAss> rnas = new ArrayList<>();

    public void calcularRNA() {
        if (tipo.compareToIgnoreCase("Diario") == 0) {
            try {
                //diario
                calcularRNADiario();

            } catch (Exception ex) {
                Logger.getLogger(Controlador.class.getName()).log(Level.SEVERE, null, ex);
            }
        } else if (tipo.compareToIgnoreCase("Mensal") == 0) {
            try {
                //diario
                calcularRNAMensal();
            } catch (Exception ex) {
                Logger.getLogger(Controlador.class.getName()).log(Level.SEVERE, null, ex);
            }
        } else if (tipo.compareToIgnoreCase("Anual") == 0) {
```

```
        try {
            calcularRNAAnual();
        } catch (Exception ex) {
            Logger.getLogger(Controlador.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}

public void calcularRNADiario() throws Exception {
    RNA rna = new RNA();
    double res = 1;
    rnas = new ArrayList<>();
    rnas.add(new RNAss(nivel, tap1, tap2, tap3, tap4, tap5, tap6, tap7, tap8, res, dataSelecao));
    rnas = rna.calcularDiario(rnas);
    verificar();
    System.out.println("Resposta: " + rnas.get(0).getResposta());

}

//graficoDiario
private LineChartModel dateModelGraficoDiario;

public LineChartModel getDateModelGraficoDiario() {
    createDateModelGraficoDiario();
    return dateModelGraficoDiario;
}

private void createDateModelGraficoDiario() {

    if (rnas.isEmpty()) {
        dateModelGraficoDiario = new LineChartModel();
        LineChartSeries series1 = new LineChartSeries();
        series1.setLabel("Series 1");

        series1.set("2014-01-01", 51);
        series1.set("2014-01-06", 22);

        dateModelGraficoDiario.addSeries(series1);

        dateModelGraficoDiario.setTitle("Titulo");
```

```
        dateModelGraficoDiario.setZoom(true);
        dateModelGraficoDiario.getAxis(AxisType.Y).setLabel("Valores (Nível)");
        DateAxis axis = new DateAxis("Datas");
        axis.setTickAngle(-50);
        //  axis.setMax("2014-01-07");
        axis.setTickFormat("%#d-%b-%y");

        dateModelGraficoDiario.getAxes().put(AxisType.X, axis);
    } else {

        dateModelGraficoDiario = new LineChartModel();
        LineChartSeries series1 = new LineChartSeries();
        series1.setLabel("Series 1");
        Date data = rnas.get(0).getData();
        Date data2 = dataSomar(data, +6);
        series1.set(formato.format(data), rnas.get(0).getNivel());
        series1.set(formato.format(data2), rnas.get(0).getResposta());

        dateModelGraficoDiario.addSeries(series1);

        dateModelGraficoDiario.setTitle("Titulo");
        dateModelGraficoDiario.setZoom(true);
        dateModelGraficoDiario.getAxis(AxisType.Y).setLabel("Valores (Nível)");
        DateAxis axis = new DateAxis("Datas");
        axis.setTickAngle(-50);
        //  axis.setMax("2014-01-07");
        axis.setTickFormat("%#d-%b-%y");

        dateModelGraficoDiario.getAxes().put(AxisType.X, axis);
    }
}

public Date dataSomar(Date data, int valor) {
    Date dat = new Date();
    Calendar c = Calendar.getInstance();
    c.setTime(data);
    c.add(Calendar.DATE, valor);
    dat.setTime(c.getTimeInMillis());
    return dat;
}
```

```
private void calcularRNAMensal() throws Exception {
    RNA rna = new RNA();
    double res = 1;
    rnas = new ArrayList<>();
    rnas = carregarMes();
    rnas = rna.calcularDiario(rnas);

}

public int getSaidaNivel() {
    return saidaNivel;
}

public void setSaidaNivel(int saidaNivel) {
    this.saidaNivel = saidaNivel;
}

private ArrayList<RNAss> carregarMes() throws ClassNotFoundException, SQLException {
    rnas = new ArrayList<>();
    dao = new DAO();
    int mes = dataSelecao.getMonth();
    Date dataControle = new Date();
    dataControle.setDate(01);
    dataControle.setMonth(mes);
    dataControle.setYear(dataSelecao.getYear());

    for (int i = 0; i < 32; i++) {
        if (dataControle.getMonth() == mes) {

            boolean ok = true;

            Calendar c = Calendar.getInstance();
            Calendar ca = Calendar.getInstance();

            c.setTime(dataControle);
            ca = c;
            Date d = new Date();
            d.setTime(c.getTimeInMillis());
            Date DataTemp = new Date();
```

```
ok = true;
ca.setTime(d);
//  ca.add(Calendar.DATE, +1);
d.setTime(ca.getTimeInMillis());

Nivel nivel;
Precipitacao precipitacao;
if (ok) {
    //nivel de resposta; dia 13
    nivel = obterNivel(d, dao);
    if (nivel == null) {
        ok = false;
    } else {
        if (nivel.getNivel() == 0) {
            ok = false;
        } else {
            this.saidaNivel = nivel.getNivel();
        }
    }
}

if (ok) {
    //niv e tap1
    c.add(Calendar.DATE, -6);
    DataTemp.setTime(c.getTimeInMillis());
    nivel = obterNivel(DataTemp, dao);
    if (nivel == null) {
        ok = false;
    } else {
        if (nivel.getNivel() == 0) {
            ok = false;
        } else {
            this.nivel = nivel.getNivel();
        }
    }
    precipitacao = obterPrecipitacao(DataTemp, dao);

    if (precipitacao == null) {
        ok = false;
```

```
    } else {
        tap1 = precipitacao.getTap01();
    }
}

if (ok) {
    //tap4
    c.add(Calendar.DATE, -2);
    DataTemp.setTime(c.getTimeInMillis());

    precipitacao = obterPrecipitacao(DataTemp, dao);
    if (precipitacao == null) {
        ok = false;
    } else {
        tap4 = precipitacao.getTap04();
    }
}

if (ok) {
    //tap5 e tap2
    c.add(Calendar.DATE, -1);
    DataTemp.setTime(c.getTimeInMillis());

    precipitacao = obterPrecipitacao(DataTemp, dao);
    if (precipitacao == null) {
        ok = false;
    } else {
        tap5 = precipitacao.getTap05();
        tap2 = precipitacao.getTap02();
    }
}

if (ok) {
    //tap7 e tap8
    c.add(Calendar.DATE, -2);
    DataTemp.setTime(c.getTimeInMillis());

    precipitacao = obterPrecipitacao(DataTemp, dao);
    if (precipitacao == null) {
        ok = false;
```

```
                } else {
                    tap7 = precipitacao.getTap07();
                    tap8 = precipitacao.getTap08();
                }
            }

            if (ok) {
                //tap3 e tap6
                c.add(Calendar.DATE, -1);
                DataTemp.setTime(c.getTimeInMillis());

                precipitacao = obterPrecipitacao(DataTemp, dao);
                if (precipitacao == null) {
                    ok = false;
                } else {
                    tap3 = precipitacao.getTap03();
                    tap6 = precipitacao.getTap06();
                }
            }

            if (ok) {
                //add aray aqui
                rnas.add(new RNAss(this.nivel, tap1, tap2, tap3, tap4, tap5, tap6, tap7, tap8, 0.0,
dataControle, saidaNivel));
                //   System.out.println("," + tap1 + "," + tap2 + "," + tap3 + "," + tap4 + "," + tap5 + "," +
tap6 + "," + tap7 + "," + tap8 + "," + saidaNivel);
            }

        }

        dataControle = dataSomar(dataControle, +1);
    }

    //  System.out.println("data: "+formato.format(dataControle));
    return rnas;
}

private LineChartModel areaModel;

public LineChartModel getAreaModel() {
```

```
    criarGraficoMes();
    return areaModel;
}

private void criarGraficoMes() {

    if (rnas.isEmpty()) {
        areaModel = new LineChartModel();
        LineChartSeries series1 = new LineChartSeries();
        series1.setLabel("Series 1");

        series1.set("2014-01-01", 51);
        series1.set("2014-01-06", 22);

        areaModel.addSeries(series1);

        areaModel.setTitle("Titulo");
        areaModel.setZoom(true);
        areaModel.getAxis(AxisType.Y).setLabel("Valores (Nível)");
        DateAxis axis = new DateAxis("Datas");
        axis.setTickAngle(-50);
        //  axis.setMax("2014-01-07");
        axis.setTickFormat("%#d-%b-%y");

        areaModel.getAxes().put(AxisType.X, axis);
    } else {

        areaModel = new LineChartModel();
        LineChartSeries series1 = new LineChartSeries();
        LineChartSeries series2 = new LineChartSeries();
        series1.setLabel("Nível Real");
        series2.setLabel("Nível RNA");

        for (RNAss rna : rnas) {
            series1.set(formato.format(rna.getData()), rna.getNivelDoDia());
            series2.set(formato.format(rna.getData()), rna.getResposta());
        }

        series1.setFill(true);
```

```
        areaModel.addSeries(series1);
        areaModel.addSeries(series2);
        areaModel.setLegendPosition("ne");
        areaModel.setTitle("Simulação da RNA - " + formato3.format(dataSelecao));
        areaModel.setZoom(true);
        areaModel.getAxis(AxisType.Y).setLabel("Valores (Nível)");
        DateAxis axis = new DateAxis("Datas");
        axis.setTickAngle(-50);
        //  axis.setMax("2014-01-07");
        axis.setTickFormat("%#d-%b-%y");

        areaModel.getAxes().put(AxisType.X, axis);
    }
}

private void calcularRNAAnual() throws Exception {

    RNA rna = new RNA();
    double res = 1;
    rnas = new ArrayList<>();
    rnas = carregarAno();
    //  rnas.add(new RNAss(nivel, tap1, tap2, tap3, tap4, tap5, tap6, tap7, tap8, res, dataSelecao));
    rnas = rna.calcularDiario(rnas);

}
DAO dao;

private ArrayList<RNAss> carregarAno() throws ClassNotFoundException, SQLException {
    dao = new DAO();
    rnas = new ArrayList<>();

    String ano = formato2.format(dataSelecao);
    Date dataControle = new Date();
    dataControle.setDate(01);
    dataControle.setMonth(00);
    dataControle.setYear(dataSelecao.getYear());

    for (int i = 0; i < 370; i++) {

        if (formato2.format(dataControle).compareToIgnoreCase(ano) == 0) {
```

```
boolean ok = true;

Calendar c = Calendar.getInstance();
Calendar ca = Calendar.getInstance();

c.setTime(dataControle);
ca = c;
Date d = new Date();
d.setTime(c.getTimeInMillis());
Date DataTemp = new Date();

ok = true;
ca.setTime(d);
//  ca.add(Calendar.DATE, +1);
d.setTime(ca.getTimeInMillis());

Nivel nivel;
Precipitacao precipitacao;
if (ok) {
    //nivel de resposta; dia 13
    nivel = obterNivel(d, dao);
    if (nivel == null) {
        ok = false;
    } else {
        if (nivel.getNivel() == 0) {
            ok = false;
        } else {
            this.saidaNivel = nivel.getNivel();
        }
    }
}

if (ok) {
    //niv e tap1
    c.add(Calendar.DATE, -6);
    DataTemp.setTime(c.getTimeInMillis());
    nivel = obterNivel(DataTemp, dao);
    if (nivel == null) {
        ok = false;
```

```
    } else {
        if (nivel.getNivel() == 0) {
            ok = false;
        } else {
            this.nivel = nivel.getNivel();
        }
    }
    precipitacao = obterPrecipitacao(DataTemp, dao);

    if (precipitacao == null) {
        ok = false;
    } else {
        tap1 = precipitacao.getTap01();
    }
}

if (ok) {
    //tap4
    c.add(Calendar.DATE, -2);
    DataTemp.setTime(c.getTimeInMillis());

    precipitacao = obterPrecipitacao(DataTemp, dao);
    if (precipitacao == null) {
        ok = false;
    } else {
        tap4 = precipitacao.getTap04();
    }
}

if (ok) {
    //tap5 e tap2
    c.add(Calendar.DATE, -1);
    DataTemp.setTime(c.getTimeInMillis());

    precipitacao = obterPrecipitacao(DataTemp, dao);
    if (precipitacao == null) {
        ok = false;
    } else {
        tap5 = precipitacao.getTap05();
        tap2 = precipitacao.getTap02();
```

```
            }
        }

        if (ok) {
            //tap7 e tap8
            c.add(Calendar.DATE, -2);
            DataTemp.setTime(c.getTimeInMillis());

            precipitacao = obterPrecipitacao(DataTemp, dao);
            if (precipitacao == null) {
                ok = false;
            } else {
                tap7 = precipitacao.getTap07();
                tap8 = precipitacao.getTap08();
            }
        }

        if (ok) {
            //tap3 e tap6
            c.add(Calendar.DATE, -1);
            DataTemp.setTime(c.getTimeInMillis());

            precipitacao = obterPrecipitacao(DataTemp, dao);
            if (precipitacao == null) {
                ok = false;
            } else {
                tap3 = precipitacao.getTap03();
                tap6 = precipitacao.getTap06();
            }
        }

        if (ok) {
            //add aray aqui
            rnas.add(new RNAss(this.nivel, tap1, tap2, tap3, tap4, tap5, tap6, tap7, tap8, 0.0,
dataControle, saidaNivel));
            //   System.out.println("," + tap1 + "," + tap2 + "," + tap3 + "," + tap4 + "," + tap5 + "," +
tap6 + "," + tap7 + "," + tap8 + "," + saidaNivel);
        }

    }
```

```
        dataControle = dataSomar(dataControle, +1);
    }

    //  System.out.println("data: "+formato.format(dataControle));
    return rnas;
}

private LineChartModel areaModelAnual;

public LineChartModel getAreaModelAnual() {
    criarGraficoAnual();
    return areaModel;
}

private void criarGraficoAnual() {

    if (rnas.isEmpty()) {
        areaModel = new LineChartModel();
        LineChartSeries series1 = new LineChartSeries();
        series1.setLabel("Series 1");

        series1.set("2014-01-01", 51);
        series1.set("2014-01-06", 22);

        areaModel.addSeries(series1);

        areaModel.setTitle("Titulo");
        areaModel.setZoom(true);
        areaModel.getAxis(AxisType.Y).setLabel("Valores (Nível)");
        DateAxis axis = new DateAxis("Datas");
        axis.setTickAngle(-50);
        //  axis.setMax("2014-01-07");
        axis.setTickFormat("%#d-%b-%y");

        areaModel.getAxes().put(AxisType.X, axis);
    } else {

        areaModel = new LineChartModel();
        LineChartSeries series1 = new LineChartSeries();
```

```
        LineChartSeries series2 = new LineChartSeries();
        series2.setLabel("Nível RNA");
        series1.setLabel("Nível Real");

        for (RNAss rna : rnas) {
            series1.set(formato.format(rna.getData()), rna.getNivelDoDia());
            series2.set(formato.format(rna.getData()), rna.getResposta());
        }

        series1.setFill(true);

        areaModel.addSeries(series1);
        areaModel.addSeries(series2);
        areaModel.setLegendPosition("ne");
        areaModel.setTitle("Simulação da RNA Anual - " + formato2.format(dataSelecao));
        areaModel.setZoom(true);
        areaModel.getAxis(AxisType.Y).setLabel("Valores (Nível)");
        DateAxis axis = new DateAxis("Datas");
        axis.setTickAngle(-50);
        //  axis.setMax("2014-01-07");
        axis.setTickFormat("%#d-%b-%y");

        areaModel.getAxes().put(AxisType.X, axis);
    }
}

//mapa
private MapModel modeloMapa;
Ruas rua = new Ruas();
ArrayList<Ruas> temp = new ListaRuas().getRuas();
private int valor1 = 10;
private int valor2 = 50;
private int valor3 = 100;
private int nivelRio = nivel;

public void verificar() {
    nivelRio = (int) rnas.get(0).getResposta();
    modeloMapa = new DefaultMapModel();
    for (Ruas ruas : temp) {
        int temp = nivelRio - ruas.getNivel();
```

```
        if (temp > 0) {
            if (temp <= valor1) {
                Polyline polyline = new Polyline();
                polyline.getPaths().add(ruas.getLocalizacao());
                polyline.getPaths().add(ruas.getLocalizacao2());
                polyline.setStrokeWeight(10);
                polyline.setStrokeColor("#FFFF00");
                polyline.setStrokeOpacity(0.7);
                modeloMapa.addOverlay(polyline);
            } else if (temp <= valor2) {
                Polyline polyline = new Polyline();
                polyline.getPaths().add(ruas.getLocalizacao());
                polyline.getPaths().add(ruas.getLocalizacao2());
                polyline.setStrokeWeight(10);
                polyline.setStrokeColor("#FFA500");
                polyline.setStrokeOpacity(0.7);
                modeloMapa.addOverlay(polyline);
            } else if (temp <= valor3) {
                Polyline polyline = new Polyline();
                polyline.getPaths().add(ruas.getLocalizacao());
                polyline.getPaths().add(ruas.getLocalizacao2());
                polyline.setStrokeWeight(10);
                polyline.setStrokeColor("#A0522D");
                polyline.setStrokeOpacity(0.7);
                modeloMapa.addOverlay(polyline);
            } else {
                Polyline polyline = new Polyline();
                polyline.getPaths().add(ruas.getLocalizacao());
                polyline.getPaths().add(ruas.getLocalizacao2());
                polyline.setStrokeWeight(10);
                polyline.setStrokeColor("#FF0000");
                polyline.setStrokeOpacity(0.7);
                modeloMapa.addOverlay(polyline);
            }
        }

    }
}

public Controlador() {
```

```
}

public MapModel getModeloMapa() {
    return modeloMapa;
}

//
private MapModel polygonModel;
Pontos pontos = new Pontos();

public void construtorMapa1() {
    polygonModel = new DefaultMapModel();

    //Shared coordinates
    Polygon tap01 = pontos.tap01;
    Polygon tap02 = pontos.tap02;
    Polygon tap03 = pontos.tap03;
    Polygon tap04 = pontos.tap04;
    Polygon tap05 = pontos.tap05;
    Polygon tap06 = pontos.tap06;
    Polygon tap07 = pontos.tap07;
    Polygon tap08 = pontos.tap08;

    tap01.setStrokeColor("#000000");
    tap01.setFillColor(verificarCor((int) tap1));
    tap01.setStrokeOpacity(0.7);
    tap01.setFillOpacity(0.7);

    tap02.setStrokeColor("000000");
    tap02.setFillColor(verificarCor((int) tap2));
    tap02.setStrokeOpacity(0.7);
    tap02.setFillOpacity(0.7);

    tap03.setStrokeColor("#000000");
    tap03.setFillColor(verificarCor((int) tap3));
    tap03.setStrokeOpacity(0.7);
    tap03.setFillOpacity(0.7);

    tap04.setStrokeColor("#000000");
```

```
    tap04.setFillColor(verificarCor((int) tap4));
    tap04.setStrokeOpacity(0.7);
    tap04.setFillOpacity(0.7);

    tap05.setStrokeColor("#000000");
    tap05.setFillColor(verificarCor((int) tap5));
    tap05.setStrokeOpacity(0.7);
    tap05.setFillOpacity(0.7);

    tap06.setStrokeColor("#000000");
    tap06.setFillColor(verificarCor((int) tap6));
    tap06.setStrokeOpacity(0.7);
    tap06.setFillOpacity(0.7);

    tap07.setStrokeColor("#000000");
    tap07.setFillColor(verificarCor((int) tap7));
    tap07.setStrokeOpacity(0.7);
    tap07.setFillOpacity(0.7);

    tap08.setStrokeColor("#000000");
    tap08.setFillColor(verificarCor((int) tap8));
    tap08.setStrokeOpacity(0.7);
    tap08.setFillOpacity(0.7);

    polygonModel.addOverlay(tap01);
    polygonModel.addOverlay(tap02);
    polygonModel.addOverlay(tap03);
    polygonModel.addOverlay(tap04);
    polygonModel.addOverlay(tap05);
    polygonModel.addOverlay(tap06);
    polygonModel.addOverlay(tap07);
    polygonModel.addOverlay(tap08);
}

public MapModel getPolygonModel() {
    construtorMapa1();
    return polygonModel;
}

public String verificarCor(int valor) {
```

```
    String cor = "#FFFFFF";
    if (valor <= 0) {
        cor = "#FFFFFF";
    } else if (valor >= 1 & valor < 5) {
        cor = "#F5DEB3";
    } else if (valor >= 5 & valor < 10) {
        cor = "#87CEFA";
    } else if (valor >= 10 & valor < 15) {
        cor = "#00BFFF";
    } else if (valor >= 15 & valor < 20) {
        cor = "#1E90FF";
    } else if (valor >= 20 & valor < 30) {
        cor = "#98FB98";
    } else if (valor >= 30 & valor < 35) {
        cor = "#00FF7F";
    } else if (valor >= 35 & valor < 40) {
        cor = "#006400";
    } else if (valor >= 40 & valor < 50) {
        cor = "#FFFF00";
    } else if (valor >= 50 & valor < 60) {
        cor = "#F0E68C";
    } else if (valor >= 60 & valor < 70) {
        cor = "#FFA500";
    } else if (valor >= 70 & valor < 80) {
        cor = "#FF4500";
    } else if (valor >= 80 & valor < 90) {
        cor = "#FF0000";
    } else if (valor >= 90 & valor < 100) {
        cor = "#8B0000";
    } else if (valor >= 100 & valor < 150) {
        cor = "#EE82EE";
    } else if (valor >= 150) {
        cor = "#800080";
    } else {
        cor = "#FFFFFF";
    }
    return cor;
}}
```

APÊNDICE D - CÓDIGO FONTE: CONTROLADOR.JAVA

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:p="http://primefaces.org/ui">

  <f:view contentType="text/html">
    <h:head>
      <script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyAOEKeOGEegZorw-
oJzr_dhl12-Sa7y0Xw"/>
      <f:facet name="first">
        <meta content='text/html; charset=UTF-8' http-equiv="Content-Type"/>
        <title>Sistema Mestrado</title>
      </f:facet>
      <style type="text/css">
        .ui-widget {
          font-size: 75%;
        }

        .themeMenu {
          overflow: auto;
          height:300px;
          width:200px;
        }

      </style>

    </h:head>

    <h:form >

      <p:layout id="lay" fullPage="true">

        <p:layoutUnit position="north" size="70" >
          <h:graphicImage url="imagem/Norte2.png" width="100%" height="100%"/>
        </p:layoutUnit>
```

```
<p:layoutUnit position="south" size="70" >
   <h:graphicImage url="imagem/Sul2.png" width="100%" height="100%"/>
</p:layoutUnit>

<p:layoutUnit position="west" size="230" header="Menu" >
   <p:panel>
      <h:panelGrid columns="2" style="margin-bottom:2px" cellpadding="1">
         <p:selectOneRadio id="console"  value="#{controlador.console}">
            <p:ajax update="config, saidaDiario" listener="#{controlador.selecionarTipo}" />
            <f:selectItem itemLabel="Diário" itemValue="Diario" />
            <f:selectItem itemLabel="Mensal" itemValue="Mensal" />
            <f:selectItem itemLabel="Anual" itemValue="Anual" />
         </p:selectOneRadio>
      </h:panelGrid>
   </p:panel>
   <p:panel id="config" header="#{controlador.tipo}">


      <h:panelGrid id="dados" rendered="#{controlador.temDiario}"  columns="2"
style="margin-bottom: 10px">
         <p:outputLabel for="german" value="Data:" />
         <p:calendar id="german" value="#{controlador.dataSelecao}" locale="pt_BR"
navigator="true" pattern="dd-MM-yyyy" >

         </p:calendar>
         <p:outputLabel />
         <p:commandButton action="#{controlador.carregarDadosDiario()}" update="dados,
map" value="Carregar" />

         <p:outputLabel  value="Nível:" />
         <p:inputText id="txt1" value="#{controlador.nivel}"  />

         <p:outputLabel  value="Tap1:" />
         <p:inputText id="txt2" value="#{controlador.tap1}" />

         <p:outputLabel  value="Tap2:" />
         <p:inputText id="txt3" value="#{controlador.tap2}" />

         <p:outputLabel  value="Tap3:" />
```

```
                <p:inputText id="txt44" value="#{controlador.tap3}" />

                <p:outputLabel  value="Tap4:" />
                <p:inputText id="txt4" value="#{controlador.tap4}" />

                <p:outputLabel  value="Tap5:" />
                <p:inputText id="txt5" value="#{controlador.tap5}" />

                <p:outputLabel  value="Tap6:" />
                <p:inputText id="txt6" value="#{controlador.tap6}" />

                <p:outputLabel  value="Tap7:" />
                <p:inputText id="txt7" value="#{controlador.tap7}" />

                <p:outputLabel  value="Tap8:" />
                <p:inputText id="txt8" value="#{controlador.tap8}" />

                  <p:outputLabel  value="Res:" />
                  <p:inputText id="txt9" disabled="true" value="#{controlador.imprimir()}" />


            </h:panelGrid>

            <h:panelGrid rendered="#{controlador.temMensal}" columns="2" style="margin-bottom:
10px" >
                <p:outputLabel for="german3" value="Data:" />
                <p:calendar id="german3" value="#{controlador.dataSelecao}" locale="pt_BR"
navigator="true" pattern="dd-MM-yyyy" >

                </p:calendar>

            </h:panelGrid>

            <h:panelGrid rendered="#{controlador.temAnual}" columns="2" style="margin-bottom:
10px" >
                <p:outputLabel for="german4" value="Data:" />
                <p:calendar id="german4" value="#{controlador.dataSelecao}" locale="pt_BR"
navigator="true" pattern="dd-MM-yyyy" >

                </p:calendar>
```

```
                </h:panelGrid>

                <p:commandButton update="saida1, saida2, saida3, config" style="margin-left: 50px"
action="#{controlador.calcularRNA()}" value="Calcular" />

            </p:panel>




        </p:layoutUnit>

        <p:layoutUnit  position="center">
            <p:panel id="saidaDiario" >
                <h:panelGrid id="saida1" rendered="#{controlador.temDiario}" columns="2"
width="100%" >

                    <p:panel header="Precipitação" style="width: 300px;" >
                        <p:gmap id="map" center="-8.915983, -56.516050"
model="#{controlador.polygonModel}" zoom="5" type="MAP" style="width:280px;height:400px" />
                        <h:graphicImage url="imagem/legenda_prp_ncep.png" width="280px"
height="40px"/>
                    </p:panel>




                    <p:panel header="Itaituba/PA" style="width: 100%" >

                        <p:gmap id="map2" center="-4.2704009, -55.978338"
model="#{controlador.modeloMapa}" zoom="17" type="MAP" style="width: 660px;height:400px" />
                        <center>
                            <h:graphicImage url="imagem/legenda.png" width="280px" height="40px"/>
                        </center>
                    </p:panel>
```

```
                    </h:panelGrid>

                    <h:panelGrid id="saida2" rendered="#{controlador.temMensal}" columns="1"
width="100%"  >

                        <p:panel header="Gráfico" style="width: 100%; " >
                            <p:chart type="line" model="#{controlador.areaModel}" style="height:400px"/>

                        </p:panel>


                    </h:panelGrid>

                    <h:panelGrid id="saida3" rendered="#{controlador.temAnual}" columns="1"
width="100%"  >

                        <p:panel header="Gráfico" style="width: 100%; " >
                            <p:chart type="line" model="#{controlador.areaModelAnual}"
style="height:400px"/>
                        </p:panel>


                    </h:panelGrid>
                </p:panel>
            </p:layoutUnit>

        </p:layout>

    </h:form>

  </f:view>
</html>
```