UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

Marinara Rübenich Fumagalli

APLICAÇÃO DE REDES NEURAIS ARTIFICIAIS PARA PREDIÇÃO DE PALEOTEMPERATURAS COM BASE EM AMOSTRAS DE FORAMINÍFEROS

Santa Maria, RS
2019

Marinara Rübenich Fumagalli

APLICAÇÃO DE REDES NEURAIS ARTIFICIAIS PARA PREDIÇÃO DE PALEOTEMPERATURAS COM BASE EM AMOSTRAS DE FORAMINÍFEROS

Monografia apresentada ao Curso de Graduação em Sistemas de Informação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Bacharela em Sistemas de Informação** .

ORIENTADOR: Prof. Joaquim Vinicius Carvallho Assunção

Santa Maria, RS
2019

Marinara Rübenich Fumagalli

APLICAÇÃO DE REDES NEURAIS ARTIFICIAIS PARA PREDIÇÃO DE PALEOTEMPERATURAS COM BASE EM AMOSTRAS DE FORAMINÍFEROS

Monografia apresentada ao Curso de Graduação em Sistemas de Informação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Bacharela em Sistemas de Informação** .

Aprovado em 5 de julho de 2019:

Joaquim Vinicius Carvallho Assunção, Dr. (UFSM)
(Presidente/Orientador)

Giliane Bernardi, Dra. (UFSM)

Luís Álvaro de Lima Silva, Dr. (UFSM)

Santa Maria, RS
2019

DEDICATÓRIA

Dedico este trabalho ao meu pai Carlos, ao meu marido Diego e aos meus cachorrinhos, que considero como filhos, com quem sempre pude contar. Mas, dedico especialmente à minha mãe, Maria Izabel, a pessoa que mais me incentivou a estudar e que sempre me deu forças quando pensei em desistir. Ela é a minha inspiração. ♥

AGRADECIMENTOS

Agradeço primeiramente aos meus pais, que sempre me deram todo o apoio e a estrutura necessária para eu seguir firme nesta caminhada, que também me deram a melhor educação possível, me ensinaram a ser uma mulher forte, com respeito qualquer ser e que sei que sou capaz de ir atrás dos meus sonhos e, além disso, me deram meus irmãos cachorrinhos. Amo muito, muito vocês!

Agradeço também ao meu marido, que me proporcionou muitas coisas boas até hoje, me mostrou lugares incríveis e que me deu os melhores presentes: meus cachorrinhos que considero como filhos. Amo muito, muito vocês!

Agradeço também ao meu orientador, o professor Joaquim, que me sugeriu este tema maravilhoso e inspirador, que foi sempre muito solícito, empenhado e compreensivo, mostrou estar realmente interessado em orientar e sempre esteve muito presente. Muitíssimo obrigada!

Agradeço a todos professores e servidores que ajudaram na minha formação, cada um tem um papel muito importante na minha vida.

Agradeço aos meus amigos - inclusive os que já não estão mais lá - da empresa Viação União Santa Cruz que trabalham na agência da rodoviária de Santa Maria, que me viciaram em café (haha), sempre me tiraram sorrisos (mesmo nos piores dias), me fizeram sentir bem e me deixavam mais leve até o ônibus chegar no box. Agradeço também, aos motoristas que sempre me levaram em segurança neste trajeto e a todos os outros colaboradores que sempre me auxiliaram e trataram muito bem.

Agradeço aos amigos que fiz na universidade e a todos os colegas que me ajudaram de alguma forma ou de outra nesta jornada.

Agradeço a Prefeitura Municipal de Júlio de Castilhos pelo auxílio aos estudantes, graças a este auxílio eu tive a oportunidade de viajar todos os dias até Santa Maria para poder estudar.

Agradeço a todos os responsáveis pela existência da UFSM e por ter tido a honra e a oportunidade de estudar nesta instituição maravilhosa.

Por fim um conselho: amem e respeitem todos os seres, sem distinção. #GOVEGAN

“Quando o homem aprender a respeitar até o menor ser da Criação, seja animal ou vegetal, ninguém precisará ensiná-lo a amar seu semelhante”

(Albert Schweitzer)

RESUMO

APLICAÇÃO DE REDES NEURAIS ARTIFICIAIS PARA PREDIÇÃO DE PALEOTEMPERATURAS COM BASE EM AMOSTRAS DE FORAMINÍFEROS

AUTORA: Marinara Rübenich Fumagalli
ORIENTADOR: Joaquim Vinicius Carvallho Assunção

Foraminíferos são protistas unicelulares marinhos que existem desde o período Cambriano (entre 542 milhões e 488 milhões de anos atrás). Possuem duas formas: bentônicas e planctônicas, as bentônicas podem ter hábito infaunal: que vivem abaixo ou dentro dos sedimentos; ou epifaunal: vivem na superfície dos sedimentos e as planctônicas vivem dispersas no oceano e podem movimentar-se verticalmente. São conhecidas milhares de espécies que se apresentam como microfósseis conservados nos estratos ou vivos até os dias atuais, são altamente evolutivos e sensíveis às mudanças ambientais. Sua carapaça, composta de carbonato de cálcio, é bem resistente e capaz de armazenar, dentre outras informações, variações paleoclimáticas. Através da análise de sua carapaça conseguimos analisar a proporção de isótopos de oxigênio acumulados nelas e assim predizer quais eram as temperaturas em determinados períodos geológicos. Para este trabalho, as predições serão dadas por meio da aplicação de redes neurais artificiais com base em arquivos que contam com amostras da presença de espécies de foraminíferos. Redes Neurais Artificiais (RNAs) são baseadas nas comunicações e conexões entre os neurônios do cérebro humano, que nos possibilitam armazenar informações e aprendizagens. RNAs são capazes de aprender qual a melhor saída através de configurações referentes as entradas. São aplicados diferentes pesos entre as conexões, estes são calculados e podem ser ativados (1) ou não (0), portanto, a rede deve se adaptar a quaisquer entradas. Os treinamentos e testes da rede foram realizados através da IDE RStudio, com códigos escritos na linguagem R e utilizando os pacotes neuralnet: que permite a criação da rede neural e Shiny: que possibilita a criação de aplicações web. Na aplicação, o usuário seleciona o arquivo e tem como resultado a estimativa da temperatura da época em que as amostras viveram e, também, gráficos e cálculos estatísticos que informam a acurácia do valor estimado.

Palavras-chave: Aprendizagem de Máquina. Foraminíferos. Inteligência Artificial. Paleoclima. Redes Neurais.

ABSTRACT

APPLICATION OF ARTIFICIAL NEURAL NETWORKS FOR PALEOTEMPERATURES PREDICTION BASED ON FORAMINIFERA SAMPLES

AUTHOR: Marinara Rübenich Fumagalli
ADVISOR: Joaquim Vinicius Carvallho Assunção

Foraminifera are unicellular marine protists that have existed since the Cambrian period (between 542 million and 488 million years ago). They have two forms: benthic and planktonic, benthic may have an infaunal habit: living below or inside sediments; or epifaunal: they live on the surface of the sediments and the plankton live scattered in the ocean and can move vertically. Thousands of species are known that present themselves as microfossils preserved in the strata or living to the present day, are highly evolutionary and sensitive to environmental changes. Its carapace, composed of calcium carbonate, is very resistant and capable of storing, among other information, paleoclimatic variations. Through the analysis of its carapace we were able to analyze the proportion of oxygen isotopes accumulated in them and thus to predict which were the temperatures in certain geologic periods. For this work, the predictions will be given through the application of artificial neural networks based on archives that have samples of the presence of foraminifera species. Artificial Neural Networks (ANNs) are based on the communications and connections between the neurons of the human brain, which enable us to store information and learning. RNAs are able to learn the best output through configurations pertaining to inputs. Different weights are applied between the connections, these are calculated and can be activated (1) or not (0), so the network must adapt to any inputs. The training and testing of the network were carried out through the IDE RStudio, with codes written in the R language and using the packages neuralnet: which allows the creation of the neural network and Shiny: application creation web. In the application, the user selects the file and results in the estimation of the temperature at the time the samples were taken, as well as graphs and statistical calculations that inform the accuracy of the estimated value.

Keywords: Machine Learning. Foraminifera. Arificial Intelligence. Paleoclimate. Neural Networks.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BP	Backpropagation (retropropagação)
IA	Inteligência Artificial
MSE	Mean Squared Error (Erro Médio Quadrático)
RNA	Rede Neural Artificial
RPROP	Retropropagação Resiliente
SSE	Sum of Squared Errors (Soma dos Quadrados do Erro Residual)
UFSM	Universidade Federal de Santa Maria

LISTA DE SÍMBOLOS

$CaCO_3$	Carbonato de Cálcio
O^{16}	Isótopo de oxigênio com número de massa 16
O^{18}	Isótopo de oxigênio com número de massa 18

SUMÁRIO

1 INTRODUÇÃO

Foraminíferos são excelentes indicadores paleoambientais (temperatura, salinidade, profundidade e turbidez, etc.), bioestratigráficos (idade das rochas sedimentares) e isotópicos (átomos de um mesmo elemento químico que diferem no número de nêutrons). Encontrados desde o período Cambriano, possuem alta taxa de evolução, são altamente sensíveis às mudanças que ocorrem no ambiente e se mantém preservados nos sedimentos por milhões de anos. São predominantemente abundantes nas costas marítimas, encontrados principalmente em água salgada, mas também, raras vezes, em águas doces ou em águas mixohalinas (HAYWARD, B.W.; LE COZE, F.; GROSS, O., 2019).

Zucon et al. (2011) diz que, cerca de 4000 espécies são bentônicas e 40 espécies são planctônicas. Registros destes microfósseis nos estratos são datados desde 300 milhões de anos atrás. São amplamente estudados, pois, a carapaça traz consigo valiosas informações. É possível reconstruir ambientes do passado, reconhecer rochas petrolíferas, entender as mudanças nos ambientes (naturais ou causadas pela ação do homem), entre outras.

Uma informação muito útil que pode ser analisada através das carapaças coletadas é o paleoclima. Segundo resultados obtidos em (BJÖRN; NORDLUND, 1997), as RNAs são excelentes ferramentas para predição de paleotemperaturas com base em arquivos de amostras de foraminíferos. É uma técnica que vem crescendo e tornando-se muito confiável nestes cenários.

Trabalhar com Redes Neurais Artificiais parte da necessidade de fazer os computadores serem autônomos e, portanto, serem capazes de aprender – sozinhos – a partir de dados de entrada (treinamento), quais são as saídas mais próximas do que é real (aprendizagem). Conforme as afirmações de (HAYKIN, 2001) e (SILVA; SPATTI; FLAUZINO, 2010), é interessante salientar que o uso de Redes Neurais Artificiais oferece algumas propriedades úteis e capacidades:
Adaptabilidade: a RNA é capaz de adaptar os pesos das entradas a diferentes exemplos, através da experiência, adequando-se aos padrões de arquivos de entrada distintos;
Analogia Neurobiológica: trata-se da inspiração que esta abordagem tem com a capacidade rápida, paralela e eficaz que o cérebro humano tem de aprender;
Capacidade de Aprendizado: a Rede pode ser treinada e a partir dessa experiência aprender o comportamento das variáveis de entrada e qual o resultado (saída) esperado;
Facilidade de Prototipagem: RNAs podem ser implementadas facilmente em *hardware* ou *software*, pois, seu treinamento baseia-se em ajustes de pesos e seus resultados são obtidos através de fórmulas matemáticas;
Generalização e Uniformidade: na aprendizagem não-supervisionada, a variável de resultado é desconhecida. Mesmo assim a Rede Neural pode ser treinada e encontrar resultados satisfatórios;
Não-Linearidade: uma RNA é capaz de fazer cálculos complexos até encontrar uma boa saída. São inclusas camadas ocultas (*hidden layers:*) que possibilitam ajustar as curvas para dados que

não são possíveis separar através de uma reta (não linearmente separáveis);
Organização dos Dados: a rede se organiza de modo que os agrupamentos são feitos através de padrões e particularidades encontrados nos dados;
Tolerância a Falhas: mesmo que algum neurônio ou alguma conexão entre eles falhe, a saída não será seriamente afetada. Isto ocorre porque existem inúmeras interconexões em uma rede.

1.1 OBJETIVOS

Esta monografia tem como objetivo evidenciar a eficiência do uso de redes neurais artificiais para estimar paleotemperaturas por meio de uma aplicação web desenvolvida pela autora.

1.1.1 Objetivos Específicos

a Pesquisar e implementar uma rede neural artificial utilizando o algoritmo de retropropagação resiliente (*Resilient Backpropagation*);

b Criar uma aplicação web útil, de fácil utilização e entendimento para o usuário;

c Analisar e aplicar o algoritmo de retropropagação resiliente no arquivo de amostras de foraminíferos;

d Ajustar os pesos dos atributos de entrada (para obter maior acurácia);

e Treinar e testar a RNA, esta deve aprender e retornar com boa exatidão as paleotemperaturas;

f Exibir gráficos e estatísticas referentes aos resultados.

2 REVISÃO BIBLIOGRÁFICA

Este capítulo conta com duas seções, ele traz revisões bibliográficas que são primordiais para entendimento dos temas desta monografia. Primeiramente, são descritos os foraminíferos, bem como sua classificação biológica e como eles são capazes de armazenar, dentre outras, informações paleoclimáticas. Na segunda seção são trazidas definições importantes acerca de Redes Neurais e como funciona o seu ciclo, desde a coleta dos dados de entrada até a saída com as predições. Por fim, é explicado o funcionamento do algoritmo de Retropropagação Resiliente que foi o algoritmo escolhido para a implementação da rede neural.

2.1 FORAMINÍFEROS

Foraminíferos são protistas unicelulares marinhos existentes desde o período Cambriano (entre 542 milhões e 488 milhões de anos atrás). São muito utilizados em paleoceanografia devido sua alta capacidade evolutiva e por se manterem preservados nos sedimentos durante milhões de anos. A coleta e análise destes protistas possibilita a predição de paleotemperaturas, a reconstrução de ambientes paleoecológicos e a descoberta de poços de petróleo, dentre outras valiosas informações (PETRÓ, 2018). Foraminíferos são envolvidos por uma testa (carapaça), esta pode conter uma ou várias câmaras que são interligadas por uma ou várias aberturas (ver figuras 2.1 e 2.2). Conforme diz (LOEBLICH et al., 1992 apud PETRÓ, 2018) estas conexões entre as câmaras por aberturas formam o nome do grupo, pois, em latim *foramen* (orifício) e *ferre* (possuir).

O modo de vida destes protistas os classifica como bentônicos ou planctônicos. As formas bentônicas foram as primeiras a surgir na terra e habitam o fundo dos oceanos. Elas possuem hábito infaunal ou epifaunal, quer dizer que vivem dentro ou abaixo dos substratos, ou vivem presas sobre o substrato respectivamente. E as planctônicas vivem à deriva, em águas mais altas, movimentando-se verticalmente. (UNIVERSITY COLLEGE LONDON, 2002).

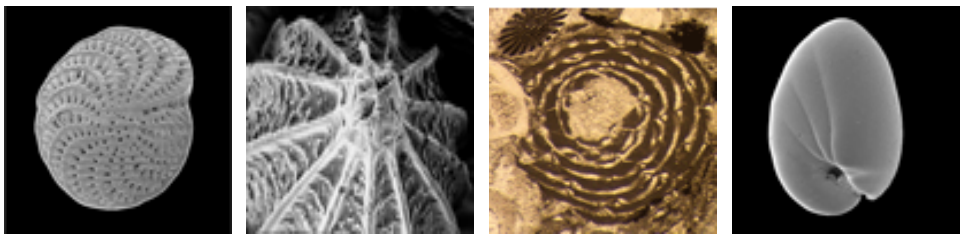
Eles habitam, em sua maioria, as águas salgadas, porém, são encontrados também nas águas doces ou então nos estuários. Podem viver tanto na superfície quanto nas profundezas dos mares, rios e oceanos. São capazes de se adaptar desde as águas mais frias até as mais quentes. Sua abundância e sua variabilidade os tornam os microfósseis mais estudados do mundo. (HAYWARD, B.W.; LE COZE, F.; GROSS, O., 2019).

Figura 2.1 – Exemplos de foraminíferos planctônicos

(a) *Globigerinoides sacculifer* (b) *Globotruncana linneiana* (c) *Pseudoguembelina excolata* (d) *Heterohelix globulosa*

Fonte: (UNIVERSITY COLLEGE LONDON, 2002).

Figura 2.2 – Exemplos de foraminíferos bentônicos



(a) *Elphidium macellum* (b) *Nodosaria sp.* (c) *Alveolina sp.* (d) *Nonionella opima*

Fonte: (UNIVERSITY COLLEGE LONDON, 2002).

2.1.1 Classificação

Em 1753, Nilsson Linnæus, foi o precursor no que conhecemos como taxonomia moderna e publicou o livro Sistema Naturæ, que traz a classificação hierárquica das espécies (MANKTELOW, 2010), um exemplo é ilustrado na figura 2.3.

Desde então, muitos autores tem divergências quanto a classificação exata dos foraminíferos, isso ocorre porque tais classificações sofrem bastantes mudanças devido ao descobrimento de novos seres e, portanto, novas especificações. Por isso, (RUGGIERO et al., 2015) propôs em 2015 a classificação mais atual que temos hoje, ele os define como organismos pertencentes ao reino *Chromista*, infrarreino *Rizharia*, filo *Retaria* e subfilo *Foraminifera*. Este subfilo conta com 3 classes: *Monothalamea* (com 4 ordens), *Globothalamea* (com 9 ordens) e *Tubothalamea* (com 2 ordens), totalizando 15 ordens.

Figura 2.3 – Categorias taxonômicas básicas de classificação

Fonte: Blog do Desconversa.[1]

2.1.2 Coleta de Amostras

A coleta das amostras se dá pela extração de sedimentos no fundo do oceano. Um determinado ponto de extração é definido pelos pesquisadores (geralmente as costas oceânicas), no local definido, uma broca perfura e retira um grande pedaço de sedimento. Estes pontos onde eles são encontrados é chamado de testemunho. Ocorrem diversas etapas após a extração: o sedimento é lavado, peneirado, pesado, padronizado, enxugado, então é feita a triagem através de uma lupa estereoscópica e os microfósseis são colocados em lâminas. Por fim, os foraminíferos são identificados (FARIA, 2011; ARARIPE et al., 2016). [1]

Tabela 2.1 – Exemplo de uma tabela de amostras de foraminíferos.

Amostras	Pontos	Profundidade (m)	Local	Point_X	Point_Y
PL15	WPT 454	18,00	ITAMARACA	303546	9143163
PL16	WPT 458	25,00	ITAMARACA	308085	9141819
PL17	WPT 462	31,00	ITAMARACA	314280	9141660
PL18	WPT469	19,80	ITAMARACA	304856	9140444

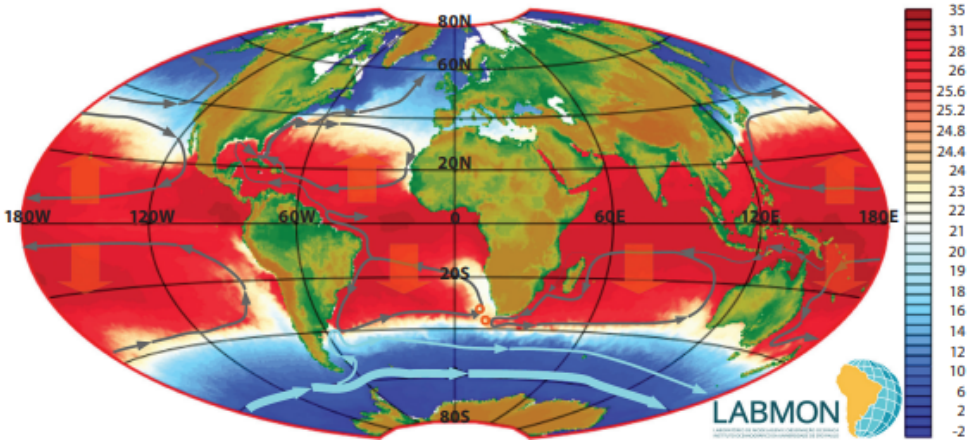
Fonte: Adaptado de (ARARIPE et al., 2016).

2.1.3 Predição de Paleotemperaturas

Aproximadamente 70% do planeta é coberto por águas que ficam em constante movimento. Este movimento (que pode ser causado por ventos, gravidade, variação da superfície) sempre varia e as águas se deslocam, isto explica as variações climáticas. O mar possui diferentes temperaturas, são mais altas nos trópicos e mais frias conforme chegam perto dos polos. Nas baixas latitudes as águas são mais leves e nas altas são mais pesadas (CAMPOS, 2014).

[1] Disponível em: <https://descomplica.com.br/blog/biologia/resumo-taxonomia-saude/> Acesso em abr. 2019.

Figura 2.4 – Representação esquemática da circulação forçada pelo vento na camada superior do oceano, sobreposta à distribuição média de temperatura da superfície do mar



Fonte: (CAMPOS, 2014).

A testa dos foraminíferos utilizados na predição de paleotemperaturas é envolvida por um biomineral: a calcita (carbonato de cálcio [$CaCO_3$]). Ela é capaz de armazenar informações sobre a proporção dos isótopos de oxigênio O^{16} (isótopo leve) e O^{18} (isótopo pesado). Quando ocorre o período glacial a geleira armazena o O^{16} e o oceano se enche de O^{18}. Em períodos interglaciais, ocorre o degelo e então a devolução destes isótopos ao mar, trazendo o equilíbrio novamente. (PETRÓ, 2013). É justamente essa proporção que define a estimativa da temperatura da época em que elas viveram.

2.2 REDES NEURAIS ARTIFICIAIS (RNAS)

Nos últimos anos as máquinas tem tido cada vez mais poder computacional, além disso, cada vez mais dados são produzidos e também recuperados. A Inteligência Artificial busca permitir que as máquinas pensem e ajam como humanos. A necessidade cada vez maior de armazenamento, comunicação e a complexidade dos problemas computacionais faz com que seja fundamental automatizar e dar autonomia às ferramentas computacionais (FACELI et al., 2011). Através de técnicas de *Machine Learning* (Aprendizagem de Máquina), computadores podem ser treinados e a partir da experiência definir qual é a melhor saída para quaisquer problemas apontados. A RNA é uma abordagem que está inclusa em Machine Learning e busca reproduzir as conexões e aprendizagens cerebrais.

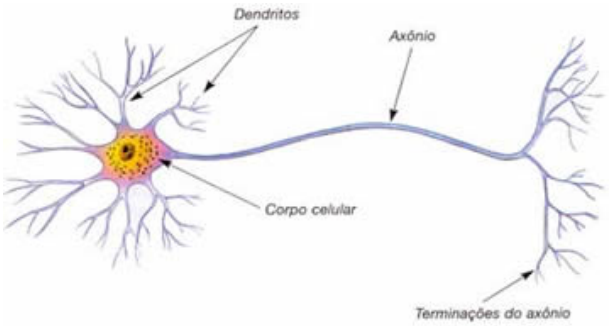
O cérebro humano é uma estrutura complexa e com alta taxa de aprendizagem, através de estímulos entre neurônios ele é capaz de retornar rapidamente informações. Diversos neurônios são conectados entre si, essa combinação de neurônios é chamada de rede neural (TAFNER; XEREZ; FILHO, 1995). Redes Neurais Artificiais são baseadas nestas mesmas conexões, onde cada entrada (nó) simula um neurônio e as combinações entre as entradas revelam

a melhor saída. Parte de uma amostra de dados é selecionada para treino, então estes nós, que possuem diferentes pesos, se interligam com outros nós. Estas informações são calculadas e seus resultados entram em uma função de ativação. A função de ativação é responsável por ativar (1) ou não ativar (0) uma ligação. A outra parte é definida para teste, neste momento a RNA já assimilou o padrão dos dados de treino e é capaz de predizer quais são as saídas para os dados de teste. (HAYKIN, 2001). As RNAs podem possuir uma ou mais camadas, as redes multicamadas possuem outras camadas além da inicial, conhecidas como camadas ocultas que propiciam a resolução de problemas mais complexos. As próximas subseções descrevem mais claramente estas afirmações.

2.2.1 Neurônio Biológico

O ser humano é capaz de realizar inúmeras ações, como: andar, pensar, falar, enxergar, ouvir, etc. Isso ocorre pela grande capacidade de aprendizagem que o cérebro apresenta. O cérebro possui cerca de 100 bilhões de células nervosas, chamadas de neurônios, que são capazes de processar e transmitir informações com sincronia (KOVÁCS, 1997). Eles operam paralelamente, conduzindo impulsos nervosos, também chamados de estímulos elétricos ou potencial de ação. O neurônio é basicamente composto por três partes: dendritos, axônios e corpo celular, também conhecido como soma (KOVÁCS, 2006). Estas partes podem ser observadas na Figura 2.5. [2]

Figura 2.5 – Partes de um neurônio biológico



Fonte: Página do site Só Biologia.[2]

Segundo (TAFNER; XEREZ; FILHO, 1995), a transmissão de um impulso nervoso ocorre quando o neurônio capta, através de reações físico-químicas, o estímulo dos dendritos e o transmite pelo axônio. Este estímulo é transformado em um impulso elétrico, percorre o axônio e atinge um outro neurônio, esta conexão se chama sinapse. Um neurônio biológico é disparado quando ele atinge seu limiar de excitação ou potencial excitatório. Quando este li-

[2]Disponível em: <https://www.sobiologia.com.br/conteudos/FisiologiaAnimal/nervoso2.php> Acesso em abr. 2019.

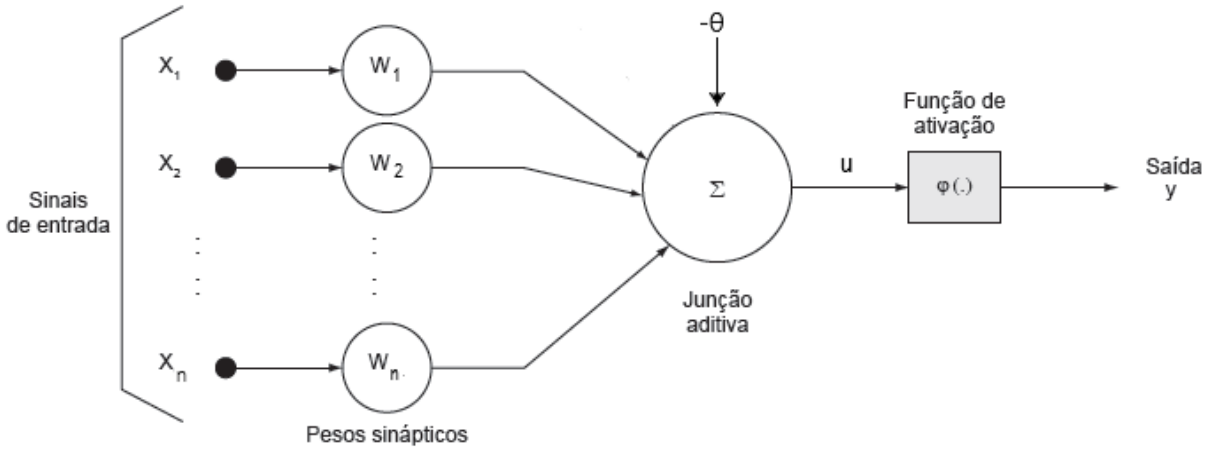
mite não é atingido, o neurônio é descartado, ou seja, seu potencial é inibitório. (CARVALHO; BRAGA; LUDERMIR, 2012).

2.2.2 Neurônio Artificial

A base de funcionamento de um neurônio artificial são cálculos matemáticos. Conforme explicações de (CARVALHO; BRAGA; LUDERMIR, 2012; DATA SCIENCE ACADEMY, 2019; HAYKIN, 2001; KOVÁCS, 2006) e observação da figura 2.6, as n variáveis de entrada (dendritos) recebem valores x_1, x_2, x_3, x_n, a variável de saída (axônio) recebe um valor y e são aplicados pesos em cada interconexão w_1, w_2, w_3, w_n. O neurônio artificial irá passar por cálculos (corpo celular ou soma), as entradas são multiplicadas pelos pesos e seus resultados $x_i w_i$ são somados (Σ). O neurônio artificial será disparado (saída igual a 1) toda vez que esta soma ultrapassar o seu limite ou *threshold* (limiar de ativação). O limite possui uma polarização, também chamada de *bias* que é representado por θ, seu valor pode ser positivo ou negativo. Dependendo do peso aplicado ao limiar, ele é capaz de inverter o resultado de uma soma e trazer novas possibilidades de aprendizagem. Então, chagamos ao valor u que é resultado da junção aditiva, se o valor for ≥ 0 ele tem um potencial excitatório, do contrário tem potencial inibitório.

Posteriormente, temos a saída, que pode ser tanto o resultado final, como também pode ser a entrada de um outro neurônio e nesse caso ocorre a conexão entre eles (sinapse).

Figura 2.6 – Modelo de um neurônio artificial



Fonte: Adaptado de (HAYKIN, 2001)

Segundo (DATA SCIENCE ACADEMY, 2019), a fórmula utilizada no processo de aprendizagem é:

$$\boldsymbol{y = \text{Ativação}(\Sigma(x * w) + \theta)} \tag{2.1}$$

2.2.2.1 Função de Ativação

Nesta subseção serão apresentadas as funções de ativação mais comumente utilizadas, baseadas nas variáveis da figura 2.6 e nas descrições de (SILVA; SPATTI; FLAUZINO, 2010) e explicações de (DATA SCIENCE ACADEMY, 2019). O domínio u é aplicado a função g que limita a saída y em intervalos aceitáveis para as imagens. A função de ativação φ(u) define qual será a saída y da rede neural. Portanto, a saída y, é a relação da combinação de todas as entradas a rede, ela pode ser o valor final ou o valor que outros neurônios a serem interligados podem utilizar.

Abaixo temos alguns exemplos de funções de ativação:

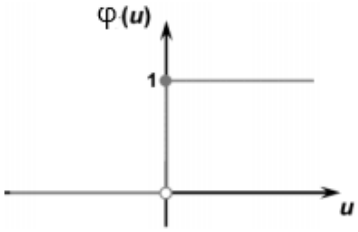
a) **Funções de Ativação parcialmente diferenciáveis:** ocorrem quando a primeira derivada da função é inexistente, ou seja, todos ou alguns pontos do domínio não são deriváveis fazendo a reta tangente ser paralela ao eixo dos x.

   Funções:

   – **Função degrau:** quando o potencial de ativação u for $\geq$ a 0 o resultado da função será 1, ou seja, o neurônio será ativado. Do contrário, quando u for < que 0, o neurônio não será ativado.

$$\varphi(u) = \begin{cases} 1 & \text{se} \quad u \geq 0 \\ 0 & \text{se} \quad u < 0 \end{cases} \tag{2.2}$$

Gráfico 2.1 – Gráfico da função degrau

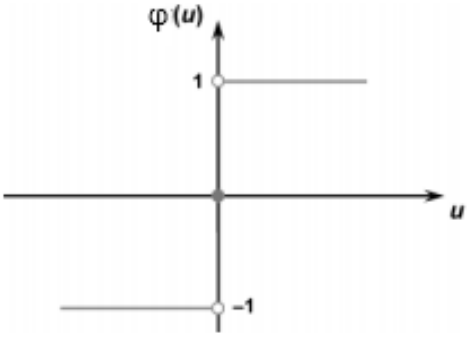


Fonte: Adaptado de (SILVA; SPATTI; FLAUZINO, 2010)

   – **Função degrau bipolar ou função sinal:** quando o potencial de ativação u for positivo, o valor do resultado será 1, se for 0 ele será nulo e quando u for < que 0, assumirá valores negativos. Esta função também traz mais duas possibilidades: se u for = 0 o resultado assume valor 1 ou também pode permanecer inalterado, assumindo o valor da saída anterior.

$$\varphi(u) = \begin{cases} 1 & \text{se} \quad u > 0 \\ 0 & \text{se} \quad u = 0 \\ -1 & \text{se} \quad u < 0 \end{cases} \tag{2.3}$$

Gráfico 2.2 – Gráfico da função degrau bipolar



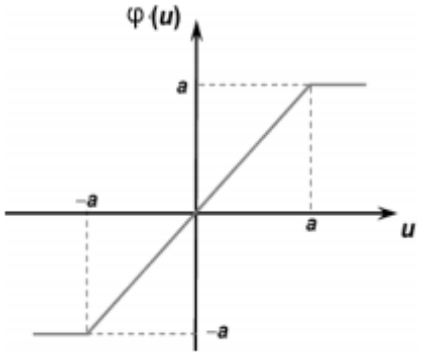
Fonte: Adaptado de (SILVA; SPATTI; FLAUZINO, 2010)

– **Função rampa simétrica:** nesta função, todos os valores assumidos como resultado são os mesmos valores do potencial de ativação *u* quando eles estiverem dentro do limite de valores definido, caso contrário eles assumirão os valores dos limites.

$$\varphi(u) = \begin{cases} a & \text{se} \quad u > a \\ u & \text{se} \quad -a \leq u \leq a \\ -a & \text{se} \quad u < a \end{cases} \tag{2.4}$$

onde -*a* e *a* são o intervalo de valores que *u* pode assumir.

Gráfico 2.3 – Gráfico da função rampa simétrica



Fonte: Adaptado de (SILVA; SPATTI; FLAUZINO, 2010)

b) **Funções de Ativação totalmente diferenciáveis:** todos os pontos do domínio possuem derivadas de primeira ordem. A reta tangente se inclina em pontos conforme a direção da curva do gráfico, estes pontos são as derivadas da função. Em geral, proporcionam resolver problemas mais complexos e que não são linearmente separáveis. Funções:

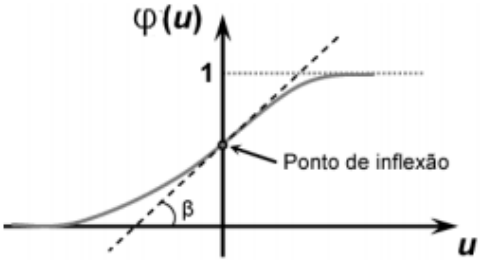
– **Função sigmoidal (logística):** é a função mais comumente utilizada entre as aplicações. Os resultados apresentados por esta função sempre serão valores reais entre 0 e 1, ou seja, a saída *y* sempre terá valor positivo. Possui um formato similar

ao da função degrau, porém, utilizando a função sigmóide, conseguimos resolver problemas de uma forma não linear. O único problema desta função de ativação é que os valores de *y* são sempre positivos, mesmo assim ela é ótima para ser usada em classificadores.

$$\varphi(u) = 1 + \frac{1}{1 + e^{-\beta . u}} \tag{2.5}$$

onde β é uma constante real associada a inclinação no ponto de inflexão (ponto onde a curvatura da função troca o sinal).

Gráfico 2.4 – Gráfico da função sigmoidal



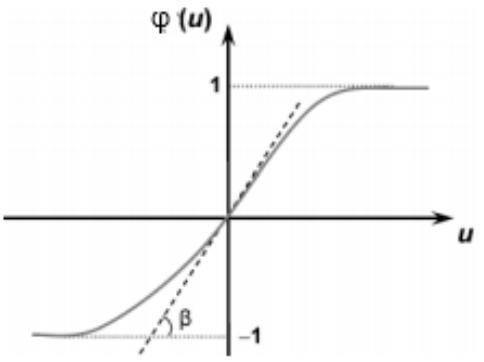
Fonte: Adaptado de (SILVA; SPATTI; FLAUZINO, 2010)

– **Função tanh (tangente hiperbólica):** também é uma função sigmoidal. Semelhante a função degrau bipolar, porém, como é totalmente diferenciável, permite resolver problemas não lineares. Resolve o problema da função sigmoidal, pois, o resultado se encontra entre o intervalo de valores [-1, 1].

$$\varphi(u) = \frac{1 - e^{-\beta . u}}{1 + e^{-\beta . u}} \tag{2.6}$$

onde β é uma constante real associada a inclinação no ponto de inflexão (ponto onde a curvatura da função troca o sinal).

Gráfico 2.5 – Gráfico da função tangente hiperbólica



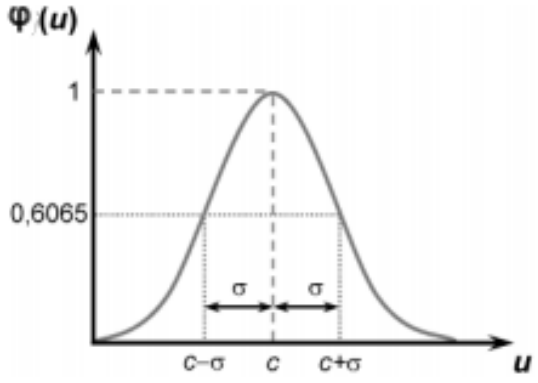
Fonte: Adaptado de (SILVA; SPATTI; FLAUZINO, 2010)

– **Função gaussiana:** os resultados gerados serão iguais para aqueles valores de u que estejam posicionados em uma mesma distância de c (média). A curva da função é simétrica em relação a esta média e σ está diretamente associado ao ponto de inflexão da função.

$$\varphi(u) = e^{\frac{-u-c^2}{2\sigma^2}} \tag{2.7}$$

onde c é o centro da função e σ é o desvio padrão da função.

Gráfico 2.6 – Gráfico da função gaussiana

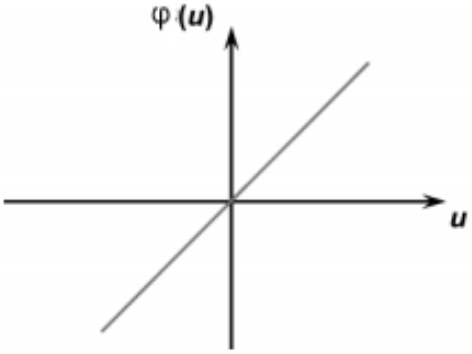


Fonte: Adaptado de (SILVA; SPATTI; FLAUZINO, 2010)

– **Função linear:** os valores resultantes desta função serão sempre os mesmos valores do potencial de ativação u. Ela é ideal para problemas simples e permite alta interpretabilidade do resultado.

$$\varphi(u) = u \tag{2.8}$$

Gráfico 2.7 – Gráfico da função linear



Fonte: Adaptado de (SILVA; SPATTI; FLAUZINO, 2010)

2.2.3 Comparação de Desempenhos

Como as redes neurais artificiais foram criadas para reproduzir a aprendizagem de redes neurais biológicas, é interessante fazer uma equiparação entre ambos. A tabela 2.2 traz a comparação de desempenho entre eles.

Tabela 2.2 – Aspectos comparativos entre o neurônio artificial e o biológico.

Parâmetro	Neurônio Artificial	Neurônio Biológico
Eficiência energética (operação\segundo)	10^{-6} joules (J)	10^{-16} joules (J)
Tempo de processamento (operação\neurônio)	10^{-9}s (clock em GHz)	10^{-3}s
Mecanismo de processamento	Tipicamente sequencial	Tipicamente paralelo

Fonte: Adaptado de (SILVA; SPATTI; FLAUZINO, 2010).

O cérebro tem em torno de 10 bilhões de neurônios que se interligam por aproximadamente 60 trilhões de sinapses e trabalham totalmente em paralelo. O número de operações por segundo que o cérebro é capaz de fazer é bem superior ao número de operações feitas por um computador. Porém, utilizando redes neurais artificiais conseguimos processamentos de 5 a 6 vezes mais rápidos do que os do cérebro humano, o neurônio artificial trabalha na ordem dos nanossegundos, já o neurônio biológico trabalha com milissegundos (FAGGIN, 1991 apud HAYKIN, 2001).

2.2.4 Arquiteturas

A arquitetura de uma rede neural define a estrutura que ela irá apresentar, ou seja, mostra a forma como os neurônios estão interligados e para onde estão direcionadas as suas saídas (HAYKIN, 2001).

O processo de aprendizagem da rede está totalmente ligado a arquitetura escolhida. O algoritmo de aprendizagem (processo de ajuste) é um processo importante na criação de uma RNA, pois é nesta etapa que ocorre o ajuste dos pesos e limiares dos neurônios e isto está diretamente ligado com a acurácia da predição da rede (SILVA; SPATTI; FLAUZINO, 2010). O ajuste do limiar de ativação vai permitir que apenas uma ou algumas saídas sejam modificadas conforme a necessidade da rede, pois, geralmente quando se modificam apenas os pesos das entradas, todas as saídas acabam sendo modificadas e isto pode tornar a rede menos precisa.

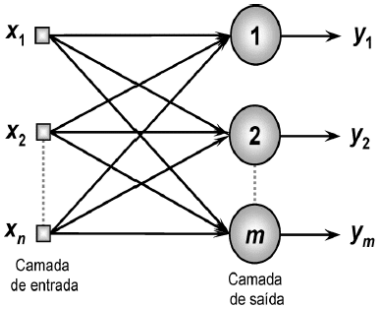
É bem fácil de entender como funcionam as camadas de entrada e saída, pois conhecemos as entradas e também qual a provável saída que vamos obter. Porém o funcionamento das camadas ocultas é bem mais complexo (DATA SCIENCE ACADEMY, 2019), nesse caso temos muitas possibilidades de cálculos para encontrar as melhores configurações de combinações de

entradas e como as saídas devem se comportar. Por isso, se torna bem complexo trabalhar com problemas não linearmente separáveis, porém os resultados são bem satisfatórios.

As arquiteturas de RNAs possuem 4 categorias: redes *feedforward* (alimentação para a frente) camada única, redes *feedforward* multicamadas, redes recorrentes e redes reticuladas que serão expostas neste capítulo com base nos livros escritos por (DATA SCIENCE ACADEMY, 2019), (HAYKIN, 2001) e (SILVA; SPATTI; FLAUZINO, 2010).

a) **Camada única:** nesta arquitetura temos a camada de entrada e apenas uma camada de neurônios que já é a própria camada de saída. As informações são sempre levadas à frente e existem tantas saídas quanto quantidades de neurônios na rede.

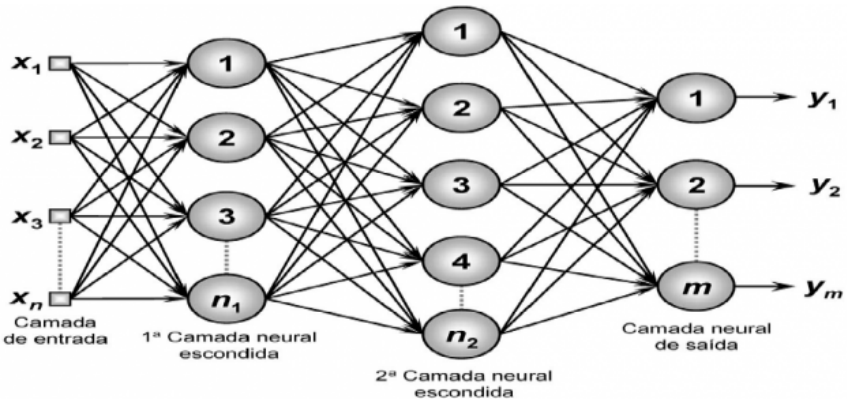
Figura 2.7 – Exemplo de uma rede *feedforward* de camada única



Fonte: (SILVA; SPATTI; FLAUZINO, 2010)

b) **Multicamadas:** ela possui este nome porque possui uma nova camada de neurônios além da camada de saída: a camada oculta. É possível observar na figura 2.2 que foram inclusas duas camadas ocultas e, como na camada única, o número de saídas depende da quantidade de neurônios na camada de saída. O treinamento envolve ajustes dos pesos e limiares para minimizar o erro da saída, se este erro for muito grande, os pesos das camadas anteriores são recalculados, trazendo resultados cada vez mais próximos da realidade, por isso ela se mostra muito eficiente na classificação de padrões.

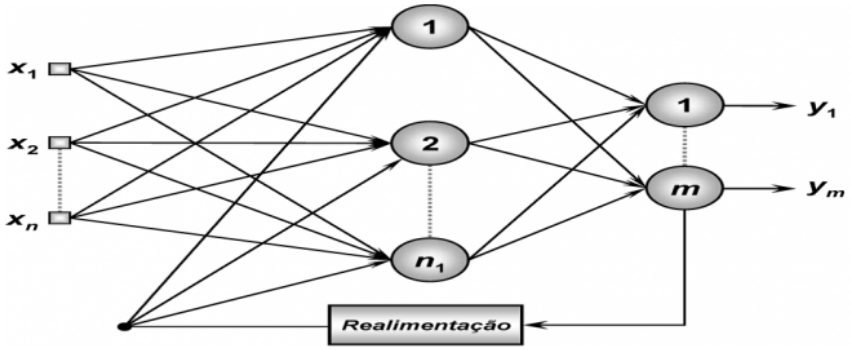
Figura 2.8 – Exemplo de uma rede *feedforward* multicamadas



Fonte: (SILVA; SPATTI; FLAUZINO, 2010)

c) **Recorrente:** também chamada de realimentada, produz novas saídas considerando o valor das saídas antecedentes. As saídas dos neurônios servem como entrada para outros neurônios e assim ocorre a recorrência, esta rede pode ter uma ou mais camadas. Esta arquitetura geralmente é utilizada quando as informações a serem analisadas variam conforme o tempo passa, como, por exemplo, na previsão de séries temporais.

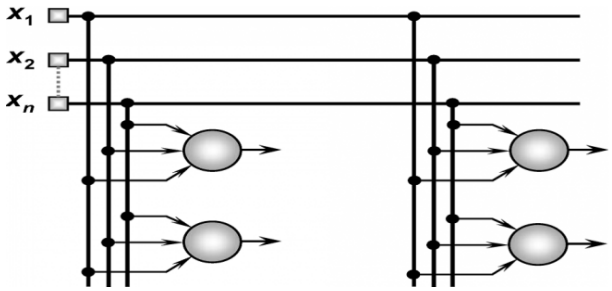
Figura 2.9 – Exemplo de uma rede recorrente



Fonte: (SILVA; SPATTI; FLAUZINO, 2010)

d) **Reticulada:** nesta arquitetura todos os nós recebem os valores de todas as entradas. Os pesos são atualizados repetidamente de acordo com regras pré-estabelecidas até que se encontre uma configuração final satisfatória. É muito utilizada na extração de características de dados e na divisão de grupos que possuem semelhança entre si, como, por exemplo, em *clustering*.

Figura 2.10 – Exemplo de uma rede reticulada



Fonte: (SILVA; SPATTI; FLAUZINO, 2010)

2.2.5 Aprendizagem

As redes neurais artificiais possuem muita notoriedade no que diz respeito a aprendizagem por meio de padrões de amostras. A partir de diferentes dados de entrada a rede é capaz de encontrar as melhores combinações entre eles e encontrar uma saída bastante fiel à realidade e que pode ser generalizada a quaisquer problemas análogos.

Silva, Spatti e Flauzino (2010) enfatiza que "O conjunto destes passos ordenados visando o treinamento da rede é chamado de algoritmo de aprendizagem". Para termos um bom algoritmo, é importante termos amostras consistentes e ter atenção no ajuste dos pesos e divisão das amostras.

Geralmente existem dois subconjuntos: o subconjunto das amostras para treinamento e o subconjunto das amostras para teste. A porcentagem de amostras que vai para treinamento tem a função de elucidar para a rede o seu funcionamento e é nesse ponto em que ocorre o aprendizado, enquanto as amostras de teste tem a função de garantir que a rede aprendeu de forma correta e de apresentar resultados aceitáveis e que possibilitem soluções genéricas.

Baseado em (CARVALHO; BRAGA; LUDERMIR, 2012) temos dois tipos de aprendizagem: a supervisionada e a não-supervisionada. As descrições desta subseção serão baseadas na citação acima, em (HAYKIN, 2001) e em (SILVA; SPATTI; FLAUZINO, 2010).

a) **Supervisionada:** nesse caso, cada entrada já traz consigo a saída esperada e a partir disso as estruturas da rede vão formular as estimativas de aprendizagem. É como se existisse um "supervisor" da rede que tem a função a ensinar e mostrar a saída mais correta através de comparações com a saída desejada. Diante disso, os pesos devem ser ajustados tantas vezes quantas forem necessárias até que o erro seja o menor possível fazendo com que o resultado seja suficientemente próximo do verdadeiro.

   Dois tipos de implementação são possíveis para este tipo de aprendizagem: o *online* e o *offline*. No treinamento *online* os dados mudam a todo o momento e a adaptação de rede deve ser contínua, já no *offline*, os dados não se modificam e por consequência a solução obtida pela rede se mantém; caso novos dados entrem, um novo treinamento deve ser feito, considerando-se o treinamento anterior, até o processo de aprendizagem for completado.

   – **Aprendizagem por correção de erros:** é a aprendizagem mais comumente utilizada, onde se procura minimizar o erro da saída da rede em relação a saída esperada. Segundo (CARVALHO; BRAGA; LUDERMIR, 2012), a expressão genérica para se encontrar este erro é dada pela equação 2.9.

$$e(t) = y_d(t) - y(t) \tag{2.9}$$

   onde e é o erro da rede e y_d é a saída desejada.
   A fórmula genérica de atualização dos pesos da rede é dada pela equação 2.10.

$$w_i(t+1) = w_i(t) + \eta e(t) x_i(t) \tag{2.10}$$

   onde $w_i(t)$ é o peso da entrada i, η é a taxa de aprendizado, $e(t)$ é o erro e $x_i(t)$ é a entrada i.
   A taxa de aprendizado define o quão rápida a rede é capaz de aprender, ela as-

sume, geralmente, valores entre o intervalo [0, 1]. Uma alta taxa (perto de 1) garante um salto maior na mudança dos pesos, pois, aumenta mais significativamente o valor dos pesos sinápticos e os corrige de forma mais rápida; porém, o ajuste feito na rede não é tão fino e consequentemente a acurácia também não é. O ideal é haver um equilíbrio entre ambos.

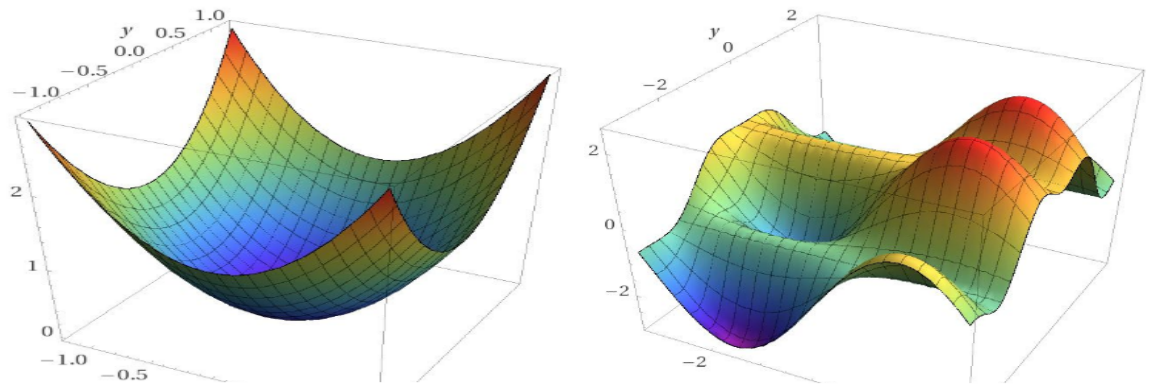
A equação correta de ajuste de pesos para cada modelo envolve minimizar os erros quadráticos das saídas, para isso tem-se a equação 2.11.

$$\varepsilon^2 = \frac{1}{2}\sum_{i=1}^{p}(y_d^i - y)^2 \tag{2.11}$$

onde ε^2 é a superfície de erro do conjunto, p é o número de exemplos de treinamento, y_i^d é a saída desejada para o vetor de entrada x_i e y é a saída atual da rede para o vetor x_i.

Desta forma, o aprendizado por correção de erros tem como objetivo mover-se, a partir de um ponto qualquer, até o menor erro encontrado entre a saída atual e a saída desejada (erro mínimo global). A figura 2.11 mostra as superfícies de erro para neurônios que utilizam funções de ativação parcialmente diferenciáveis (sem camadas ocultas) (*a*) e para os que utilizam funções de ativação totalmente diferenciáveis (com camadas ocultas) (*b*), estas possuem mínimos globais (menor erro no geral) e mínimos locais (menor erro em uma determinada região).

Figura 2.11 – Superfícies de erro quadrático (gradientes)



(a) *Neurônios com funções de ativação lineares* (b) *Neurônios com funções de ativação não-lineares*

Fonte: (KHATURIA, AYOOSH, 2018).

– **Aprendizagem por reforço:** nessa aprendizagem a rede visa, através de tentativas e erros, maximizar seu desempenho reforçando seus resultados satisfatórios baseando-se em informações (qualitativas ou quantitativas) referentes a contínuas interações com o sistema. Ou seja, o supervisor busca manter as execuções corretas e assim a rede passa a criar seu aprendizado. Se a rede acerta, então esta ação é continuada (reforçada) até chegar uma saída aceitável; caso contrário, esta ação

deve ser descontinuada (enfraquecida).

b) **Não-supervisionada:** neste tipo de aprendizagem, o treinamento ocorre sem um supervisor, ou seja, não se tem conhecimento sobre os dados de saída, apenas os dados de entrada são recebidos pela aplicação. Então, a rede deve se auto-organizar conforme identifica semelhança entre os grupos de dados e dessa forma o algoritmo de aprendizagem deve ajustar os pesos e limiares a fim de ecoar cada subconjunto descoberto. O algoritmo pode ser livre para designar quantos *clusters* pode encontrar, como também, o desenvolvedor pode definir esse valor.

   – **Aprendizagem Hebbiana:** se um neurônio recebe uma entrada proveniente de outro neurônio e ambos são ativados, então esta conexão deve ser mais forte; caso contrário, se a entrada recebida for ativa mas o que vier a receber for inativo, esta conexão deve ser enfraquecida ou então eliminada. Nesta aprendizagem é levada em conta a influência do neurônio pré-sináptico no neurônio pós-sináptico.

   – **Aprendizagem por competição:** determinado conjunto de entradas é recebido, então as saídas passam a competir a fim de, a que tiver maior ativação, consiga o privilégio de ter o ajuste dos seus pesos atualizados e a capacidade de inibir as saídas dos seus concorrentes e excitar a dos seus vizinhos. Quando uma área de vizinhança aceitável é formada e conforme ocorre o treinamento, ocorre um estreitamento deste resultado.

2.2.6 Algoritmo de Retropropagação Resiliente (RPROP)

O algoritmo mais utilizado para aprendizagem supervisionada em redes *feedforward* multicamadas é o algoritmo de retropropagação (*backpropagation*). A derivada parcial de cada peso é calculada e, tendo conhecimento de seus resultados, é possível reduzir o erro através da execução da descida do gradiente, conforme figura 2.11 (RIEDMILLER; BRAUN, 1993).

Segundo (RIEDMILLER, 1994), o algoritmo de retropropagação resiliente permite uma aprendizagem mais rápida do que o algoritmo de retropropagação, pois, apenas o sinal é levado em conta e não toda a magnitude da derivação e age livremente sobre cada peso, com isso, os pesos são ajustados mais rapidamente. Assim, a cada iteração o erro é calculado e multiplicado por um valor de atualização η que é semelhante a taxa de aprendizagem. Se houver qualquer mudança no sinal da derivada parcial da função erro em comparação ao da última iteração, o valor de atualização é multiplicado por um fator η^-; se o sinal for o mesmo, o valor de atualização é multiplicado por um valor η^+ e se o valor da derivação for 0, o valor de atualização não é modificado. Resumindo: $0 < \eta^- < 1 < \eta^+$. (RIEDMILLER, 1994), por experiências mostra que o melhor valor definido para η^- é 0.5 e para η^+ é 1.2.

(PRASAD; SINGH; LAL, 2013) e (SOUZA et al., 2004) fazem comparações entre os algoritmos de retropropagação e retropropagação resiliente e (RIEDMILLER, 1994) faz esta comparação entre quatro algoritmos: os dois mencionados e os a algoritmos SuperSAB (SSAB) e quickProp (QP). Sinteticamente falando, o algoritmo SSAB todas as taxas são inicializadas com mesmo valor e as multiplicações de η^+ só são feitas quando as derivadas parciais do erro mantiverem o mesmo sinal por várias iterações e η^- quando ela muda de sinal por muitas vezes, já o QP leva em conta e faz a atualização de pesos através da magnitude da superfície de erro, o cálculo de atualização de pesos se dá através da segunda derivada (ROISENBERG, MAURO, 2004). Em suas conclusões, todos estes autores mostram que a RPROP é superior aos outros algoritmos em questão de rapidez, iterações e acurácia da rede.

3 METODOLOGIA

Este capítulo tem a função de elucidar a motivação para utilizar este tema, as ferramentas utilizadas e os códigos realizados durante o desenvolvimento. As próximas seções explicam todos os passos (seleção dos atributos de entrada, ferramentas utilizadas e partes principais do código desenvolvido em R) utilizados para a criação do Preditor de Paleotemperaturas, além de uma descrição sobre a motivação e considerações sobre este trabalho.

3.1 MOTIVAÇÃO

De acordo com (EEROLA, 2003) nosso planeta tem 4,5 bilhões de anos e durante todo esse tempo ocorreram diversas mudanças climáticas drásticas. De certa forma todas as variações que ocorreram possuem ligações. O clima congelante fica estável, depois desértico, após volta à estabilidade e tudo isso se repete.

Como diz o Painel Brasileiro de Mudanças Climáticas (2014):

> Reconstituições paleoclimáticas assumem marcante relevância atualmente, em face à necessidade de se atribuir causas às alterações ocorridas no clima da Terra durante as últimas décadas e, também, a fim de auxiliar o estabelecimento de cenários climáticos futuros.

Como já mencionado na seção 2, foraminíferos são excelentes indicadores paleoclimáticos, outrossim, Redes Neurais Artificiais são excelentes ferramentas de predição. Consequentemente, aplicar RNAs em amostras de foraminíferos, traz resultados bem precisos, satisfatórios e importantes para a história do planeta.

3.2 SELEÇÃO DOS ATRIBUTOS DE ENTRADA

Os atributos de entrada estão presentes em um arquivo *.csv* que possui dados fictícios sobre amostras de foraminíferos. No *dataset* encontramos a densidade da população de cada espécie de foraminíferos encontradas em determinado testemunho e a última coluna conta com a temperatura estimada após a análise dos isótopos de carbono. O arquivo conta com 250 linhas que representam quantas espécies foram encontradas em determinadas profundidades do sedimento e 45 colunas, onde, 44 delas representam as espécies encontradas e a última representa a temperatura média da superfície que vai de 2,5ºC a 24,4ºC.

Na página inicial da aplicação *web* criada, o usuário tem a opção de selecionar este arquivo que contém as amostras e após o *upload* ele define quais serão os atributos (variáveis)

de entrada e qual será a classe utilizada, que no caso da aplicação mencionada, trata-se da temperatura média encontrada. Como tratamos de uma aprendizagem supervisionada, esta classe, segundo Fritsch, Guenther e Guenther (2019, p. 7, tradução nossa), "é uma descrição simbólica sobre o modelo a ser predito"[3].

Uma visão mais detalhada desta página e de outras se encontra na seção 3.4 e as capturas de tela referentes a aplicação se encontram no capítulo 4.

3.3 FERRAMENTAS UTILIZADAS

A aplicação para predição de paleotemperaturas foi totalmente desenvolvida na linguagem R, utilizando principalmente os pacotes *shiny* e o *neuralnet*. O pacote *shiny* permite a criação de aplicações *web*, onde temos um script para interface do usuário (*user side*) que cria o *layout* da aplicação: **ui.R** e um script para o lado do servidor (*server side*) que é responsável pela lógica da programação **server.R**.

É no *shiny server* que são feitos os códigos referentes a leitura do arquivo, seleção de atributos, criação da rede neural, computação dos dados e estatísticas. Primeiramente são separados 90% dos dados do *dataset* para treinamento da rede, os outros 10% são dados de teste. O pacote *neuralnet* recebe como parâmetro os atributos de entrada selecionados pelo usuário, os dados de treino, o número de camadas ocultas, o limite e o algoritmo (RPROP). A partir de agora a rede já aprendeu as configurações de entrada, é capaz computar os dados e predizer as temperaturas com base nos dados de teste.

Todos os dados do *dataset* e as predições são visíveis ao usuário através aplicação. Bem como dados estatísticos sobre a acurácia, iterações e o erro final de treino da rede. A subseção 3.3.1 visa demonstrar o porquê da escolha destas ferramentas e a seção 3.4 vai apresentar os códigos criados.

a) **Linguagem R:** a linguagem R é uma ferramenta bem completa no que diz respeito a estatística, análise e manipulação de dados, possibilitando fazer cálculos e apresentar gráficos de uma forma bem abrangente.

   Esta linguagem foi escolhida principalmente porque conta com o pacote *neuralnet* e também com o pacote *Shiny*. O pacote *neuralnet* é um pacote bem completo que permite criar redes neurais artificiais, estatísticas e gráficos de forma descomplicada, já que, a tarefa de criar redes neurais a partir do zero, pode ser uma tarefa bem complexa e demorada visto que tantos cálculos diferentes e complexos precisam ser elaborados. E o pacote *Shiny* permite criar aplicações *web* completas, onde o usuário não precisa ter nenhum conhecimento em programação e nem experiência profunda em informática, pois, o uso da aplicação é bastante intuitivo.

[3] "a symbolic description of the model to be fitted"

Os ítens seguintes trazem mais detalhes sobre estes dois pacotes.

– **Pacote neuralnet:** baseado em informações de (GÜNTHER; FRITSCH, 2010), ele é construído para treinar redes *feedforward* multicamadas através de aprendizagem supervisionada utilizando os algoritmos BP ou RPROP. A RNA vai aprendendo através de iterações e assim se adaptando as configurações dos atributos. Este pacote também traz liberdade ao desenvolvedor, permitindo o mesmo escolher o algoritmo (BP ou RPROP e variações), o número de camadas ocultas, o limite, a condição de parada, a função de ativação diferenciável (sigmoidal ou tanh), a função de erro diferenciável, entre outras possibilidades. Também permite visualizar a rede criada e estatísticas prontas sobre a mesma.
Por fim, como consta em (GÜNTHER; FRITSCH, 2010), é interessante saber como funciona o código de atualização de pesos dentro do pacote *neuralnet*.
Para RPROP:

```
for all weights{
    if (grad.old*grad>0){
        delta := min(delta*eta.plus, delta.max)
        weights := weights - sign(grad)*delta
        grad.old := grad
    }
    else if (grad.old*grad<0){
        weights := weights + sign(grad.old)*delta
        delta := max(delta*eta.minus, delta.min)
        grad.old := 0
    }
    else if (grad.old*grad=0){
        weights := weights - sign(grad)*delta
        grad.old := grad
    }
}
```

Para BP:

```
for all weights{
weights := weights - grad*delta
}
```

– **Pacote Shiny:** como já mencionado, ele permite que sejam criadas aplicações *web* com bastantes funcionalidades. Baseado na documentação oficial (R STUDIO, 2017), podemos criar aplicações com dois scripts: *ui.R* e *server.R*, ou, nas versões mais atuais é possível ter o script em um único arquivo: *app.R* apenas chamando *ui.R* e *server.R* como funções. Para um melhor entendimento, um exemplo

de código exatamente como está presente no tutorial da documentação oficial será dado a seguir.

UI:

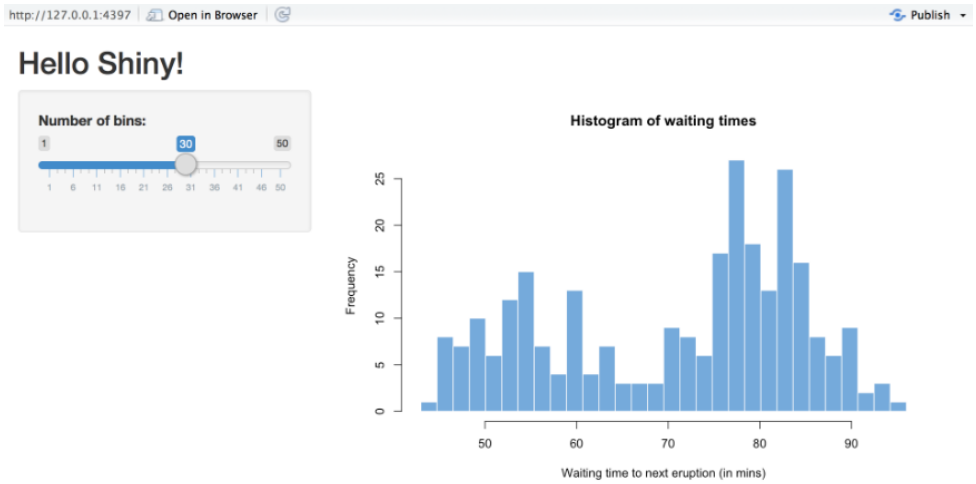
```
library(shiny)
# Define UI for app that draws a histogram -----
ui <- fluidPage(
  # App title -----
  titlePanel("Hello Shiny!"),
  # Sidebar layout with input and output definitions -----
  sidebarLayout(
    # Sidebar panel for inputs -----
    sidebarPanel(
      # Input: Slider for the number of bins -----
      sliderInput(inputId = "bins",
                  label = "Number of bins:",
                  min = 1,
                  max = 50,
                  value = 30)
    ),
    # Main panel for displaying outputs -----
    mainPanel(
      # Output: Histogram -----
      plotOutput(outputId = "distPlot")
    )
  )
)
```

SERVER:

```
# Define server logic required to draw a histogram -----
server <- function(input, output) {
  # Histogram of the Old Faithful Geyser Data -----
  output$distPlot <- renderPlot({
    x    <- faithful$waiting
    bins <- seq(min(x), max(x), length.out = input$bins + 1)
    hist(x, breaks = bins, col = "#75AADB", border = "orange",
         xlab = "Waiting time to next eruption (in mins)",
         main = "Histogram of waiting times")
    })
}
```

Que resulta na seguinte estrutura:

Figura 3.1 – Exemplo de uma aplicação web utilizando o pacote *Shiny*



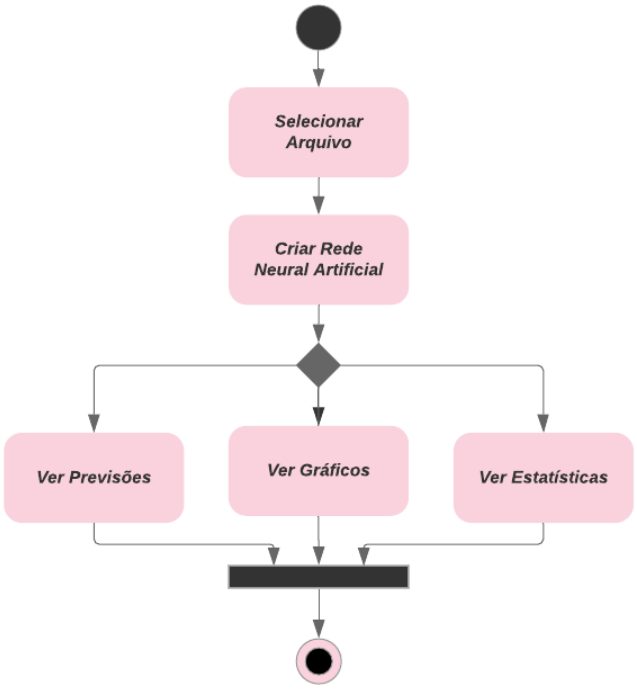
Fonte: (R STUDIO, 2017)

3.4 DESENVOLVIMENTO E VISÃO GERAL

Nesta seção, são apresentadas as funcionalidades e algumas partes do código *server.R* referentes a aplicação web As capturas de tela da mesma se encontram no capítulo 4.

Em suma, a forma como o usuário irá interagir com a aplicação, pode ser descrita conforme o diagrama de atividades (figura 3.1), que demonstra como ela é facilmente entendível.

Figura 3.2 – Diagrama de atividades



Fonte: Captura de tela realizada pela autora no dia 27 de jun de 2019.

Iniciando as explicações sobre os códigos, podemos ver o código para a leitura dos dados:

```
1 #Lendo os dados
2 dS <- import(input$file1$datapath)
```

Aqui, o usuário deve ter a possibilidade de selecionar os atributos e a classe, isso acontece através do *splitLayout* (linha 1) que divide o canto inferior esquerdo da tela em 2 partes (figura 4.1). A primeira parte conta com *checkboxes* (linha 2) para a seleção dos atributos (colunas gerais) e a segunda parte conta com *radioButtons* (linha 6) que permitem selecionar apenas uma classe (coluna resultado real). Para os atributos, exclui-se a última coluna que é a classe:

```
1  splitLayout(
2      (checkboxGroupInput(
3          "atributos",
4          label = h5(strong("Atributos")), choices = colnames(dS), selected =
               colnames(dS[1:(length(dS)-1)])
5      )),
6      (radioButtons(
7          "classe",
8          label = h5(strong("Classe")), choices = colnames(dS), selected =
              colnames(dS[length(dS)])
9       ))
10 )
```

Antes de codificar a execução da rede, um passo muito importante e que garante maior acurácia da RNA é padronizar os valores que vem do *dataset*, para isto, se faz necessário fazer uma normalização destes dados. Assim, todos os valores recebidos ficam dentro do intervalo [0, 1], onde o maior valor = 1 e o menor valor = 0, utilizando o método **min-max**. A linha 5 tem o papel de garantir a correta distribuição dos valores, x é o valor de temperatura recebido pela função, min(x) é o menor valor de temperatura encontrado dentro de todo o *dataset* e max(x) é o maior valor:

```
1 #Funcao para normalizar o dataset
2 normalize <- function(x) {
3         z = x
4   if(min(x) < max(x)){
5     z = (x - min(x)) / (max(x) - min(x))
6   }
7   return(z)
8 }
```

Para aplicar a normalização aos valores do *dataset*, deve-se aplicar a normalização min-max a todos os objetos da lista de valores. Então é feito o seguinte:

```
1 #Normalizacao do dataset. Valores ficam no intervalo [0, 1]
2 dataset <- as.data.frame(lapply(dS, normalize)))
```

A parte mais importante é aplicar a biblioteca *neuralnet* a estes dados já normalizados, e então obter a previsão. O padrão para criar uma rede neural com a biblioteca *neuralnet* é que tenhamos a classe, separada pelo símbolo " ~ " seguida de todos os atributos separados pelo símbolo " + " ($classe \sim atributo_1 + atributo_2 + ... + atributo_n$), a linha 2 se encarrega de criar esta formula. A linha 4 se encarrega de dividir as linhas do *dataset* em 90% e 10%, as linhas 5 e 6 mostram que 90% dos dados vão para treinamento e 10% vão para teste (validação), respectivamente. Da linha 9 a linha 14 é onde são definidas informações importantes de como a RNA vai lidar com estes dados. Então, é escolhido o algoritmo RPROP e os pesos iniciais recebem 'NULL', ou seja, serão aleatórios (linha 11), sequencialmente, 10 + 6 camadas ocultas e o limite máximo de iterações que a rede pode fazer em treinamento (linha 12), depois, definir não imprimir os erros a cada iteração e os limtes máximos para os cálculos (linha 13). A linha 17 traz as previsões e a última linha se encarrega de fazer o cálculo da acurácia da RNA, fazendo uma média de valores das comparações dos dados de teste com os valores previstos:

```
1 #Criando a formula que pega os nomes das colunas e concatena cada um com o
      simbolo '+'
2 formula = str_c(input$atributos[1:length(input$atributos)], collapse="+");
3 #Separacao de dados do dataset, parte para treino e parte teste
4 index = sample(seq_len(nrow(dataset)), size = 0.90 * nrow(dataset))
5 treino <<- dataset[ index, ];
6 teste <<- dataset[ -index, ];
7
8 #Aplicando a Rede Neural
9 NN = neuralnet(
10      str_c(input$classe, " ~ ", formula), treino,
11      algorithm = "rprop+", startweights = NULL,
12      hidden = c(10, 6), stepmax = 1e+06,
13      lifesign = "none", threshold = 0.03,
14   );
15
16 #Resultados
17 previsao = compute(NN, teste);
18 #Precisao
19 precisao <- (((abs(mean((previsao$net.result - teste[2, input$classe]) /
      previsao$net.result)))) * 100)
```

Para desnormalizar os dados após a previsão, basta fazer a operação contrária à normalização:

```
#Desnormalizando os valores obtidos no resultado
resultadoDesnorm <- data.frame(min(dS[, input$classe]) + previsao$net.
    result * (max(dS[, input$classe]) - min(dS[, input$classe])))
```

Os códigos para os gráficos da comparação (figura 4.5) entre o valor desejado e o valor previsto são criados a partir da combinação dos resultados real e predito (linha 3), são criados pontos vermelhos que representam os dados preditos (linha 4), são definidos os limites do gráfico com base nas temperaturas (linha 5) e uma linha preta é traçada para identificar onde estão os valores reais (linha 6), quanto mais próximos os pontos ficarem desta linha, mais precisa é a rede neural. A última linha imprime graficamente a rede neural, sendo possível ver sua estrutura (figura 4.4):

```
#Grafico de comparacao
output$plotLinha <- renderPlot({
  plot(cbind(testeCDesnorm, resultadoDesnorm),
    col = 'red', cex = 2, pch = 18,
    xlab = 'Valor Real', ylab = 'Valor Predito', xlim = c(0, 30), ylim = c
        (0, 30))
    abline(0, 1, lwd = 2)
})

#Grafico da Rede Neural
output$plot <- renderPlot({
    plot.nnet(NN)
})
```

Finalizando, são apresentados os códigos referentes as estatísticas, como erro de treino, número de iterações ocorridas na aprendizagem e a acurácia. Quanto menores forem os erros obtidos, mais precisa é a rede, consequentemente podemos afirmar que a rede produz resultados satisfatórios:

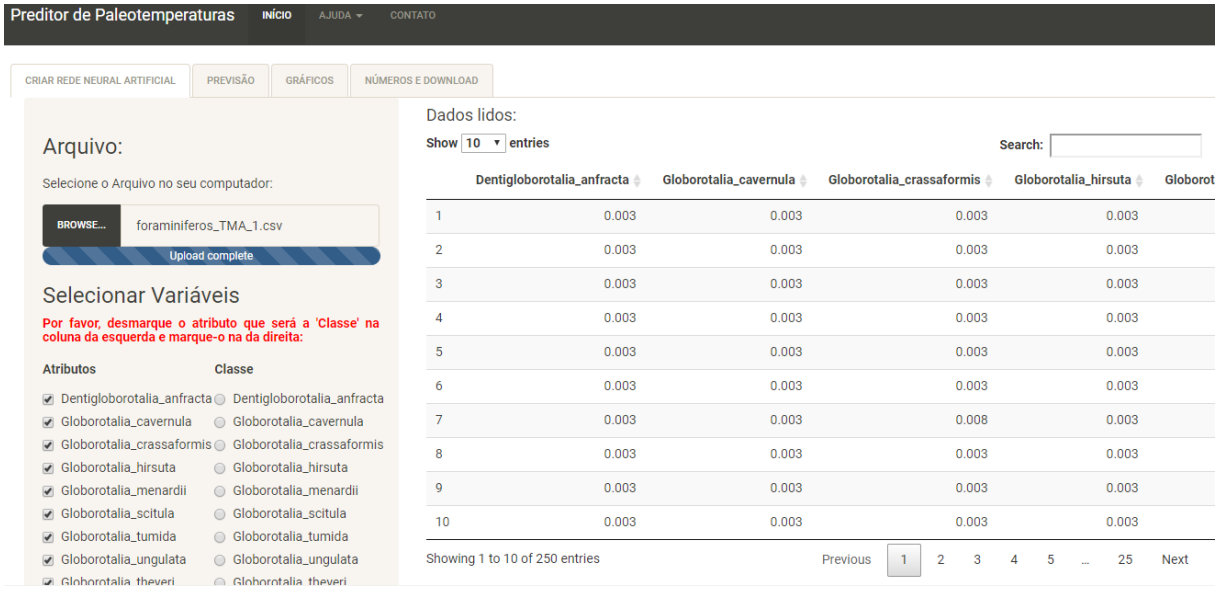
```
h4(strong("Erro de Treino (SSE): "), NN$result.matrix[1,], align = "center"
    ),
h4(strong("Erro de Teste (MSE): "), sum((teste[2, input$classe] - previsao$
    net.result)^2)/nrow(teste), align = "center"),
h4(strong("Numero de Iteracoes: "), NN$result.matrix[3], align = "center"),
h4(strong("Precisao: "), str_c(precisao,"%"), align = "center"),
```

4 RESULTADOS

Este capítulo apresenta o passo a passo do ponto de vista do usuário e as capturas de tela da aplicação web, com a finalidade de apresentar uma melhor visualização do que foi desenvolvido. No início serão mostradas as capturas de tela e definições da estrutura da aplicação e por último os resultados dos experimentos feitos na própria aplicação.

Primeiramente, conforme pode ser observado na figura 4.1, o usuário se depara com a página inicial que começa com a aba 'CRIAR REDE NEURAL ARTIFICIAL'. Ele seleciona o arquivo .csv contendo as informações sobre a densidade de amostras de foraminíferos, após feito o *upload*, a tabela a direita é carregada e o painel de seleção a esquerda aparece.

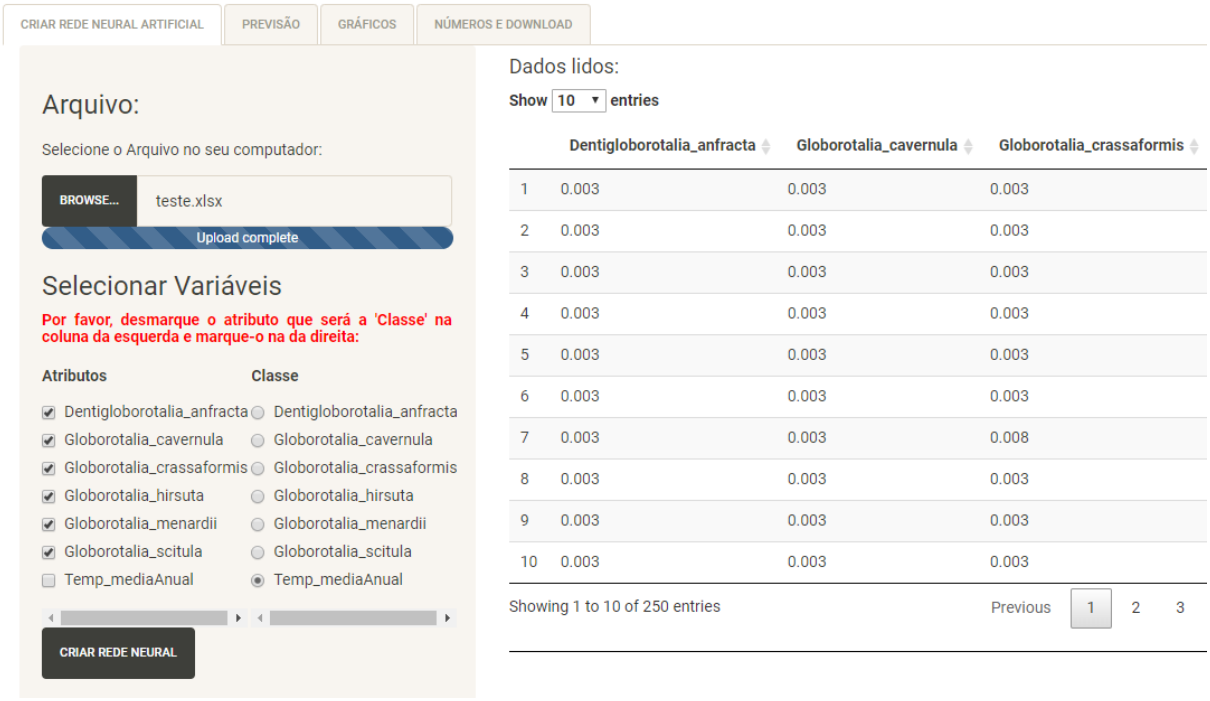
Figura 4.1 – Leitura e apresentação dos dados na tela e seleção dos atributos de entrada



Fonte: Captura de tela realizada pela autora no dia 18 de julho de 2019.

O usuário seleciona quais serão as variáveis de entrada (atributos) e a classe necessárias para criar a rede, geralmente os atributos são todas as colunas do dataset exceto a coluna que contém os dados das temperaturas reais. Feito isso, ele clica em 'CRIAR REDE NEURAL' que fica logo abaixo da seleção das variáveis, vide figura 4.2. A partir deste ponto a aplicação se encarrega de realizar todos os cálculos necessários para o correto funcionamento da rede.

Figura 4.2 – Leitura e apresentação dos dados na tela e seleção dos atributos de entrada



Fonte: Captura de tela realizada pela autora no dia 18 de julho de 2019.

O segundo passo é clicar na aba ‘PREVISÃO’ pois é nela que se encontra a comparação entre o valor desejado e o valor previsto, conforme ilustra a figura 4.3.

Esta aba vai apresentar a tabela com todos os atributos normalizados, pois é interessante podermos observar como ficaram estes dados após aplicarmos o método min-max, já a previsão será apresentada nas duas formas: normalizada e desnormalizada, pois, nosso objetivo é saber se a aplicação foi capaz de se aproximar dos dados reais, bem como a percebemos no nosso cotidiano.

Neste exemplo a rede neural teve acurácia de 92,71% (a mais alta dos testes), erro de treino de 0,098 e erro de teste de 0,490 (lembrando que os valores de erro ideais são os que estiverem mais próximos de 0).

Figura 4.3 – Visualização da tabela de comparação dos valores real x predito

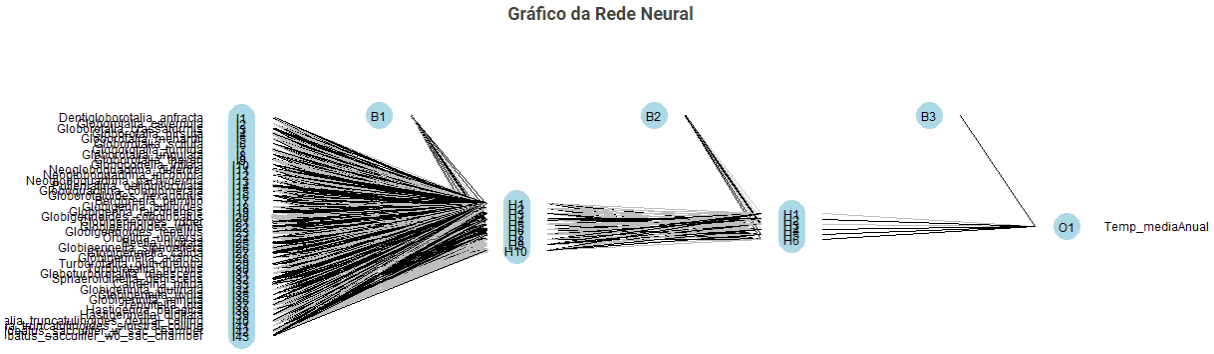
Temp_mediaAnual	Previsão	Previsão Desnormalizada
0.778132666463293	0.775565721859251	19.483690792583
0.698332946680233	0.675024254338548	17.2821021050126
0.806967245306732	0.80394971964346	20.1052242749431
0.89563563029631	0.920013344683622	22.6467066128076
0.848652711838709	0.856796713590623	21.2624318124422
0.860617646360377	0.850456763875428	21.1236039047447
0.765459882761909	0.715875228372485	18.1766289557454
0.776342493053944	0.722455311892115	18.3207151502014
0.752755131678215	0.707858227707241	18.0010781267383
0.777374582825661	0.812962214364596	20.3025737558502

Fonte: Captura de tela realizada pela autora no dia 28 de junho de 2019.

Em seguida, na aba ‘GRÁFICOS’, é possível visualizar graficamente a rede criada: a camada de entrada, as camadas ocultas, os limiares e a saída, respectivamente. Nesta mesma aba podemos ver um gráfico que é capaz de comparar o quão próximos os pontos previstos ficaram da linha que representa os valores reais. Ambos podem ser observados nas figuras 4.4 e 4.5, respectivamente.

Neste exemplo também é mostrado o exemplo de teste que teve acurácia mais alta; vale lembrar que os os pontos se encontram bem próximos da linha, demonstrando resultados bem satisfatórios.

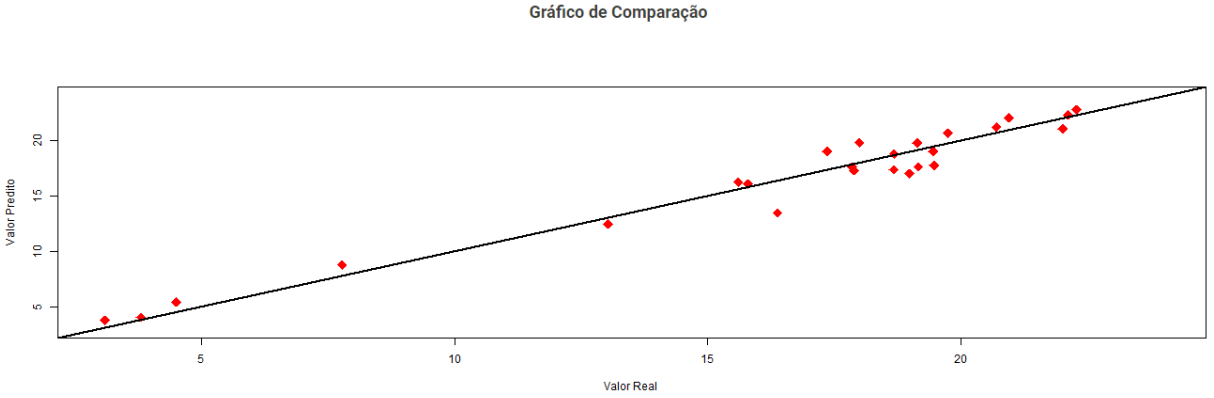
Figura 4.4 – Visualização da rede neural criada pela aplicação web



Fonte: Captura de tela realizada pela autora no dia 18 de julho de 2019.

Vale ressaltar que a sobra na escala da figura 4.5 se dá porque os 10% de valores que sobraram para teste possuiam, em geral, temperaturas mais altas.

Figura 4.5 – Visualização do gráfico de comparação dos valores real x predito



Fonte: Captura de tela realizada pela autora no dia 18 de julho de 2019.

Finalmente, na aba ‘NÚMEROS E DONWLOAD’, são apresentadas as estatísticas. Podemos ver o erro de treinamento (SSE), o erro de teste (MSE), ambos erros relacionados a saída y da rede e a saída desejada, quantas passos foram necessários para chegar a estas saídas, bem como a acurácia da rede neural artificial. Também é possível fazer o download de um arquivo *.csv* contendo a tabela da aba ‘PREVISÃO’ com todos os resultados produzidos pela rede neural artificial.

Também existem abas na parte superior, que tem a finalidade de auxiliar o usuário, trazendo uma espécie de manual de utilização e um arquivo de amostras fictício para download e testes.

Figura 4.6 – Estatísticas da rede neural artificial criada

Fonte: Captura de tela realizada pela autora no dia 18 de julho de 2019.

Foram feitas algumas experiências e, após uma bateria de 15 testes, a maior acurácia foi de 92,71% (figura 4.6) e a menor foi de 67,94%. Esta quantidade de testes é o suficiente para encontrarmos a média e desvio padrão confiáveis referentes a acurácia da RNA criada pela aplicação. A seguir, a tabela 4.1 apresenta a tabela representando os experimentos, onde SSE é o erro de treino da rede:

Tabela 4.1 – Tabela de experimentos.

-	SSE de Treino	Iterações	Acurácia
1	0,121	122	81,48%
2	0,066	100	80,06%
3	0,079	77	80,09%
4	0,136	67	82,59%
5	0,143	65	82,46%
6	0,094	140	92,08%
7	0,073	83	82,38%
8	0,089	136	83,13%
9	0,112	71	81,69%
10	0,098	57	92,71%
11	0,153	75	78,42%
12	0,098	78	81,85%
13	0,082	63	75,57%
14	0,077	66	67,94%
15	0,090	130	69,62%
Média Aritmética			**80,80%**
Desvio Padrão			**6,60**

Fonte: Autora.

Apesar da média aritmética ser boa, existe um desvio padrão um pouco alto a ser considerado. Após inúmeros testes e experimentos, pode-se constatar que o volume de dados utilizados nos testes é pequeno, por isso a acurácia mantém essa média de 80,80% e as variações chegam a ser altas. As RNAs costumam ganhar acurácia conforme aumenta a quantidade de dados para treinamento.

5 CONCLUSÃO

Esta monografia mostrou que a aplicação de RNAs se revelou eficiente no cenário de predição de paleotemperaturas, resultando em uma acurácia, apesar de, um tanto longe dos 100%, satisfatória. Este fator de acurácia, está intimamente ligado aos dados que são selecionados, se o volume de dados for pequeno ou inconsistente, a acurácia tende a baixar.

Justificando esta afirmação, foi criada uma aplicação web baseada na linguagem R e, conforme consta no capítulo 4 (resultados), os experimentos com esta aplicação se mostraram promissores. Esta aplicação web pode facilitar o trabalho de usuários interessados nestes dados, mas que ainda não dispõe de tecnologias como esta e precisam anotar as previsões e fazer os cálculos manualmente. Além disso ela é entendível e são poucos os passos necessários para se criar a rede neural artificial e ver os gráficos e estatísticas.

Para finalizar, as maiores contribuições deste trabalho foram as pesquisas que se encontram na revisão bibliográfica, a aplicação web criada e os códigos em R que podem ser modificados e utilizados para outras predições em diferentes cenários.

5.1 TRABALHOS FUTUROS

Como trabalhos futuros, pode-se sugerir que se faça um cálculo do quão robusta é a rede neural, de como ela se comporta com novos dados, como ela se comporta com dados imprecisos ou que contenham erros. Também é interessante existirem mais gráficos de comparação dos resultados obtidos pela RNA com os resultados desejados, como também, gráficos sobre sua acurácia e médias. Outra análise interessante seria comparar a eficiência da RNA em comparação com outras técnicas de predição e também comparar o algoritmo RPROP com outros algoritmos de aprendizagem.

REFERÊNCIAS BIBLIOGRÁFICAS

ARARIPE, R. V. C. d. et al. Caracterização da fauna de foraminíferos bentônicos da plataforma continental de Itamaracá, PE - Brasil. **Estudos Geológicos**, Laboratório de Paleontologia do Departamento de Geologia, Centro de Tecnologia e Geociências, Universidade Federal de Pernambuco, v. 26, n. 2, p. 91–107, 2016.

BJÖRN, A. M.; NORDLUND, U. Application of artificial neurla networks to paleoceanographic data. **Palaeogeography, palaeoclimatology, papaleoecology**, Elsevier Science B. V., v. 136, n. 1-4, p. 359–373, 1997.

CAMPOS, E. J. D. O papel do oceano nas mudanças climáticas globais. **Revista USP**, n. 103, p. 55–66, 2014.

CARVALHO, A. C. P. de L. F.; BRAGA, A. de P.; LUDERMIR, T. B. **Redes neurais artificiais: teorias e aplicações**. 2. ed., 1ª reimpressão. ed. Rio de Janeiro: Livros Técnicos e Científicos Editora, 2012. 226 p.

DATA SCIENCE ACADEMY. **Deep Learning Book, 2019**. 2019. Acesso em 17 jun. 2019. Disponível em: <http://www.deeplearningbook.com.br>.

EEROLA, T. T. Mudanças climáticas globais: passado, presente e futuro. **Fórum de ecologia. Instituto de Ecologia Política, Universidade do Estado de Santa Catarina.**, 2003.

FACELI, K. et al. **Inteligência Artificial: uma abordagem de aprendizado de máquina**. Rio de Janeiro: Livros Técnicos e Científicos Editora, 2011. 378 p.

FAGGIN, F. Vlsi implementation of neural network', tutorial notes. In: **International Joint Conference on Neural Network, Seattle, WA**. [S.l.: s.n.], 1991.

FARIA, G. R. de. **Biodiversidade de foraminíferos bentônicos e planctônicos da plataforma continental da região de Cabo Frio - RJ, Brasil**. 2011. 74 f. Monografia (Monografia) — Curso de Bacharelado em Ciências Biológicas, Universidade Federal Fluminense, Rio de Janeiro, 2011.

GÜNTHER, F.; FRITSCH, S. neuralnet: Training of neural networks. **The R journal**, v. 2, n. 1, p. 30–38, 2010.

HAYKIN, S. S. **Redes Neurais: princípios e práticas**. 2. ed., 1ª reimpressão. ed. Porto Alegre: Artmed Editora, 2001. 900 p. Tradução de Paulo M. Engel.

HAYWARD, B.W.; LE COZE, F.; GROSS, O. **World Foraminifera Database**. World Register of Marine Species (WoRMS), 2019. Acesso em 30 abr. 2019. Disponível em: <http://www.marinespecies.org/foraminifera/>.

KHATURIA, AYOOSH. **Intro to optimization in deep learning: Gradient Descent**. Paperspace Blog, 2018. Acesso em 26 jun. 2019. Disponível em: <https://blog.paperspace.com/intro-to-optimization-in-deep-learning-gradient-descent/>.

KOVÁCS, Z. L. **O cérebro e sua mente: uma introdução à Neurociência Computacional**. São Paulo: Livraria Triângulo Editora, 1997. 214 p.

____. **Redes Neurais Artificiais fundamentos e aplicações: um texto básico**. 4. ed. São Paulo: Editora Livraria da Física, 2006. 174 p.

MANKTELOW, M. History of taxonomy. **Lecture from Dept. of Systematic Biology, Uppsala University**, v. 29, 2010.

Painel Brasileiro de Mudanças Climáticas. Informações paleoclimáticas brasileiras. In: ____. **Base Científica das Mudanças Climáticas**. Rio de Janeiro: PMBC, 2014. cap. 4, p. 130.

PETRÓ, S. M. **Evolução paleoceanográfica e estratigrafia isotópica com foraminíferos planctônicos no quaternário tardio da Bacia de Campos**. 2013. 60 f. Dissertação (Mestrado em Geociências) — Universidade Federal do Rio Grande do Sul, Porto Alegre, 2013.

PETRÓ, S. M. **Introdução ao estudo dos foraminíferos**. Porto Alegre: IGEO/UFRGS, 2018. 218 p.

PRASAD, N.; SINGH, R.; LAL, S. P. Comparison of back propagation and resilient propagation algorithm for spam classification. In: IEEE. **2013 Fifth International Conference on Computational Intelligence, Modelling and Simulation**. [S.l.], 2013. p. 29–34.

R STUDIO. **Welcome to Shiny**. Shiny from R Studio, 2017. Acesso em 27 jun. 2019. Disponível em: <https://shiny.rstudio.com/tutorial/written-tutorial/lesson1/>.

RIEDMILLER, M. Rprop-description and implementation details. Citeseer, 1994.

RIEDMILLER, M.; BRAUN, H. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In: SAN FRANCISCO. **Proceedings of the IEEE international conference on neural networks**. [S.l.], 1993. v. 1993, p. 586–591.

ROISENBERG, MAURO. **Variações do BP**. Florianópolis: Universidade Federal de Santa Catarina – Laboratório de Inteligência Computacional Aplicada, 2004. Acesso em 27 jun. 2019. Disponível em: <https://www.inf.ufsc.br/~mauro.roisenberg/ine5377/Cursos-ICA/RN-6\%20-\%20backpropagation_varia\%E7\%F5es.pdf>.

RUGGIERO, M. A. et al. Correction: A higher level classification of all living organisms. **Plos one**, Public Library of Science, v. 10, n. 6, p. e0130114, 2015.

SILVA, I. N. da; SPATTI, D. H.; FLAUZINO, R. A. **Redes Neurais Artificiais: para engenharia e ciências aplicadas**. São Paulo: Artliber Editora, 2010. 399 p.

SOUZA, B. A. et al. Comparison between backpropagation and rprop algorithms applied to fault classification in transmission lines. In: IEEE. **2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541)**. [S.l.], 2004. v. 4, p. 2913–2918.

TAFNER, M. A.; XEREZ, M. de; FILHO, I. W. R. **Redes Neurais Artificiais: introdução e princípios de neurocomputação**. Blumenau: Editora FURB e Editora Eko, 1995. 199 p.

UNIVERSITY COLLEGE LONDON. **Foraminifera**. Gower Street: Postgraduate Unit of Micropalaeontology, Department of Earth Sciences, 2002. Acesso em 25 abr. 2019. Disponível em: <https://www.ucl.ac.uk/GeolSci/micropal/foram.html>.

ZUCON, M. H. et al. Microfósseis. In: ____. **Paleontologia Geral**. São Cristovão: Universidade Federal de Sergipe / CESAD, 2011. p. 55–67.