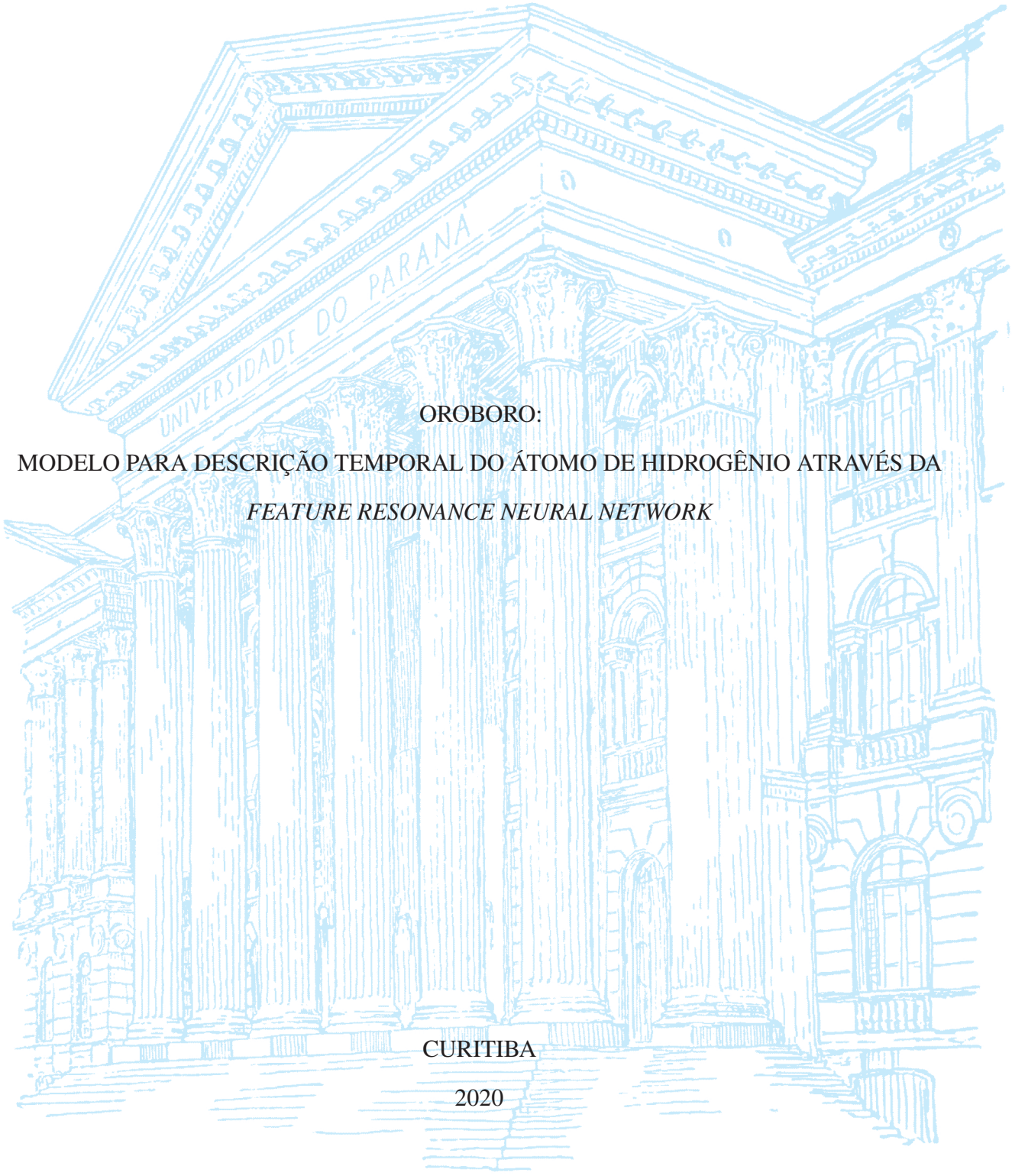
UNIVERSIDADE FEDERAL DO PARANÁ

CAMILA PEREIRA PERICO



OROBORO:
MODELO PARA DESCRIÇÃO TEMPORAL DO ÁTOMO DE HIDROGÊNIO ATRAVÉS DA *FEATURE RESONANCE NEURAL NETWORK*

CURITIBA

2020

CAMILA PEREIRA PERICO

OROBORO:
MODELO PARA DESCRIÇÃO TEMPORAL DO ÁTOMO DE HIDROGÊNIO ATRAVÉS DA *FEATURE RESONANCE NEURAL NETWORK*

Dissertação apresentada ao Programa de Pós-Graduação em Bioinformática, Setor de Educação Profissional e Tecnológica, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Mestre em Bioinformática.

Área de concentração: *Inteligência Artificial*.

Orientador: Prof. Dr. Roberto Tadeu Raittz.

Coorientador: Prof. Dr. Dietmar William Foryta.

CURITIBA

2020

Catalogação na publicação
Sistema de Bibliotecas UFPR
Biblioteca de Educação Profissional e Tecnológica

Perico, Camila Pereira
OROBORO: modelo para descrição temporal do átomo de Hidrogênio através da *Feature Resonance Neural Network* – Curitiba, 2020.
177 p.: il.

Dissertação (Mestrado) – Universidade Federal do Paraná, Setor Educação Profissional e Tecnológica, Programa de Pós-Graduação em Bioinformática, 2020.
Orientador: Roberto Tadeu Raittz
Coorientador: Dietmar William Foryta

1. Física Computacional. 2. Redes Neurais. 3. Aprendizado por Otimização. 4. Bioinformática. I. Raittz, Roberto Tadeu. II. Foryta, Dietmar William III. Título. IV. Universidade Federal do Paraná.

Elaboração: Rafaela Paula Schmitz - CRB 9/1882

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ
SETOR DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA

Pós-Graduação em Bioinformática WWW.BIOINFO.UFPR.BR
E-mail: bioinfo@ufpr.br Tel: 41 33614906

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em BIOINFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da dissertação de Mestrado de **CAMILA PEREIRA PERICO** intitulada: "**OROBORO: Modelo para descrição temporal do átomo de Hidrogênio através da Feature Resonance Neural Network**", sob orientação do Prof. Dr. ROBERTO TADEU RAITTZ e coorientação do Prof. Dr. DIETMAR WILLIAM FORYTA, que após terem inquirido a aluna e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa. A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

Curitiba, 15 de junho de 2020.

Dr. Roberto Tadeu Raittz
Presidente/Universidade Federal do Paraná - UFPR

Documento assinado digitalmente
Fernando Alvaro Ostuni Gauthier
Data: 30/06/2020 13:41:56-0300
CPF: 395.442.220-49

Dr. Fernando Alvaro Ostuni Gauthier
Avaliador Externo/Universidade Federal de Santa Catarina - UFSC

Dr. Sergio Luiz Meister Berleze
Avaliador Externo/Universidade Federal do Paraná – UFPR

Dr. Dieval Guizelini
Avaliador Interno/Universidade Federal do Paraná – UFPR

Aos meus pais.

AGRADECIMENTOS

Gostaria de agradecer primeiramente a meus pais, Marisa e Dilson, não apenas pelo apoio e paciência que dispuseram durante essa etapa importante, mas por todos os ensinamentos e o grande incentivo que me deram durante toda a vida. E principalmente, por me fornecerem aquela pequena fagulha que fez em mim nascer o interesse pela busca pelo conhecimento e paixão pela ciência. Não teria alcançado o que alcancei sem seu auxílio, apoio e amor.

Agradeço imensamente aos professores Roberto Raittz e Dietmar Foryta não apenas pela sua orientação, mas também pela sua amizade e pelos diálogos instigantes e bem humorados que tivemos, e que pretendo continuar a ter. Elogio a ambos pela sua capacidade de pensarem além dos paradigmas, cada um a seu modo, o que me inspirou a persistir e enfim realizar o presente trabalho.

Sou grata à Mariane Kulik pela sua disposição a ajudar sempre que precisei e pelas sempre tão interessantes conversas que tivemos nesses anos.

Também sou grata a Ricardo C. Volert que prontamente me auxiliou na análise e visualização dos dados.

Agradeço aos professores do programa de Bioinformática, que permitiram meu crescimento como profissional e pessoa.

À Monique pela organização dos eventos fornecendo oportunidades a todos do programa, e pela nossa representação.

À Suzana, Gracieli, Charlie, Monique, Letícia, Camilla, Mariane, Bruno e a todos os citados, e a muitos não mencionados, pela amizade, companhia e apoio.

Sou muito grata à banca avaliadora deste trabalho pela sua disposição e auxílio na correção deste trabalho.

E ao fomento pela CAPES, pois o desenvolvimento à ciência e pesquisa exige muito mais que tempo e disposição.

“All models are wrong but some are useful.”
— George E. P. Box

“Sempre foi mais fácil destruir do que criar.”
— Spock

. . . São os curiosos e insatisfeitos que movem o mundo . . .

RESUMO

Com importância ampla e inquestionável, o estudo da modelagem e dinâmica moleculares necessita constantemente de avanços e o desenvolvimento de novas técnicas é bem-vindo. O aprendizado de máquina vem sendo reconhecido na literatura como um grande aliado na simulação de sistemas químico-quânticos. Sem solução analítica para sistemas de N-corpos, a equação de Schrödinger exige solução por aproximações, as quais possuem falhas. Dentre os trabalhos revisados da literatura, não encontramos nenhum trabalho que busque obter trajetórias ou uma representação pontual do elétron no tempo utilizando recursos do ML, nem foram encontradas aplicações de redes neurais para a previsão de energias do elétron ligado a um núcleo. Propomos, portanto, a elaboração de um modelo baseado em inteligência artificial, com o uso de redes neurais, para representar o comportamento do elétron utilizando a interpretação de Copenhagen. Por não haver na literatura nenhum método de aprendizado que satisfaça as exigências para o modelo de comportamento do elétron, foi desenvolvida a rede *Feature Resonance Neural Network* (FRes). A FRes é uma rede generalista – não especializada – com função de custo customizável, de otimização evolutiva por algoritmo genético, arquitetura baseada em ressonância, e aprendizado local (supervisionada) e/ou global (reforço esparso). Validamos a FRes com testes do 'ou exclusivo' (XOR), de classificação e *augmentation*. Tais testes demonstraram que a ressonância é essencial para a solução de problemas não lineares, e que a rede é capaz de solucionar problemas simples e complexos. Obtivemos acurácia média de (93.99 ± 2.96)% e (96.69 ± 1.12)% na classificação dos dados da *Iris* e câncer de mama respectivamente, e acurácia de (96.32 ± 0.84)% para a classificação dos dados gerados pelas redes FRes geradoras. Após a validação da FRes, apresentamos um modelo alternativo para representação elétron no tempo e pontualmente, mas não por trajetórias literais. Desenvolvemos um modelo sem supervisão capaz de avaliar apenas globalmente o comportamento da partícula, respeitando a densidade probabilidade de Schrödinger. Após testes iniciais para verificação dos parâmetros adequados, obtivemos 12 redes com erro médio inferior a 10% nos parâmetros avaliados. A rede de melhor desempenho foi a código FRes14B#34, que obteve erros percentuais de 3.51 ± 0.44 %, 6.91 ± 0.39 %, 5.36 ± 0.43 % e 0.72 ± 0.42 % na comparação dos histogramas radial, angulares (ϕ e θ) e no raio médio, respectivamente. Também obteve desvio do centro de massa de 0.0597 ± 0.0054 Å. A análise dos resultados revelaram uma propriedade atratora das redes – atratores caóticos. Conseguimos idealizar e desenvolver um modelo de representação do hidrogênio preliminar bem sucedido, que ainda necessita de mais estudos, mas que abre portas para a elaboração de um novo modelo alternativo para interações de partículas. Além do modelo para o átomo de um elétron, a rede FRes possui potencial para outras aplicações.

Palavras-chave: Física Computacional.Ressonância.Redes Neurais.Aprendizado por Otimização.

ABSTRACT

With broad and unquestionable importance, the study of molecular modeling and dynamics constantly needs advances and the development of new techniques is welcome. The machine learning has been recognized in the literature as a great ally in the simulation of chemical-quantum systems. Without an analytical solution for N-body systems, the Schrödinger equation requires solution by approximations, which have flaws. Among the reviewed papers in the literature, we didn't find any work that seeks to obtain trajectories or a point representation of the electron over time using ML resources. Nor were found any applications of neural networks for predicting the energies of the binded electron to a nucleus. Therefore, we propose the development of a model based on artificial intelligence, with neural networks, to represent the behavior of the electron using the Copenhagen interpretation. As there is no learning method in the literature that satisfies the requirements for the electron behavior model, the *Feature Resonance Neural Network* (FRes) network was developed. FRes is a generalist network - not specialized - with a customizable cost function, evolutionary optimization by genetic algorithm, architecture based on resonance, and local (supervised) and/or global (sparse reinforcement) learning. We validate FRes with 'exclusive or '(XOR), classification and *augmentation* tests. Such tests demonstrated that resonance is essential for solving nonlinear problems, and that the network is capable of solving simple and complex problems. We obtained average accuracy of (93.99 ± 2.96) % and (96.69 ± 1.12) % in the classification from *Iris* and breast cancer data respectively, and accuracy of (96.32 ± 0.84) % for the classification of generated data by FRes generating networks. After validation of FRes, we present an alternative model for electron representation in time and punctualy, but not by literal trajectories. We developed an unsupervised model capable of evaluating the particle behavior only globally, respecting Schrödinger's probability density. After initial tests to check the appropriate parameters, we obtained 12 networks with an average error of less than 10% in the evaluated parameters. The network with the best performance was code FRes14B #34, which got percentage errors of 3.51 ± 0.44 %, 6.91 ± 0.39 %, 5.36 ± 0.43 % and 0.72 ± 0.42 % in the comparison of the radial and angular (ϕ and θ) histograms, and in the mean radius, respectively. It also obtained a 0.0597 ± 0.0054 Åcenter of mass deviation. The analysis of the results revealed an attractive property of the networks - chaotic attractors. We were able to devise and develop a successful preliminary hydrogen representation model, which still needs further studies, but which opens doors to the development of a new alternative model for particle interactions. In addition the model for an electron atom, the FRes network has potential for other applications.

Keywords: Computational Physics. Resonance. Neural Networks. Learning by Optmization.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE SIGLAS

ABO	Aproximação de Born-Oppenheimer
alg.	Algoritmo
BP	Algoritmo de Retropropagação de Erros (do inglês, *Backpropagation Algorithm*)
cap.	Capítulo
CM	Centro de Massa
DFT	*Density Functional Theory*
ECMC	*Event-Chain Monte Carlo*
eq.	Equação
fig.	Figura
Freq.Rel.	Frequência Relativa
FRes	*Feature Resonance Neural Network*
GA	Algoritmo Genético (do inglês *Genetic Algorithm*)
GAN	Redes Generativas Adversárias (do inglês *Generative adversarial networks*)
IA	Inteligência Artificial
MCMC	*Markov Chain Monte Carlo Method*
MD	Dinâmica Molecular (do inglês *Molecular Dynamics*)
MDP	Processo de Decisão de Markov (do inglês *Markov Decision Process*)
ML	Aprendizado de Máquina (do inglês *Machine Learning*)
MLP	Rede neural *Multilayer Perceptron*
NN	Redes Neurais (do inglês *Neural Networks*)
ReLU	*Rectified Linear Unit*
RL	Aprendizado por Reforço (do inglês *Reinforcement Learning*)
sec.	Seção
SA	Anelamento Simulado (do inglês *Simulated Annealing*)
SOM	Mapas Auto-organizáveis (do inglês *Self-organizing map*)
tab.	Tabela
tansig	Função de Ativação Tangente Hiperbólica Sigmóide (do inglês *Hyperbolic tangent sigmoid transfer function*)
XOR	ou exclusivo

C.S	comprimento da sépala(dados da *Iris*)
C.P	comprimento da pétala (dados da *Iris*)
L.S	largura da sépala (dados da *Iris*)
L.P	largura da pétala (dados da *Iris*)

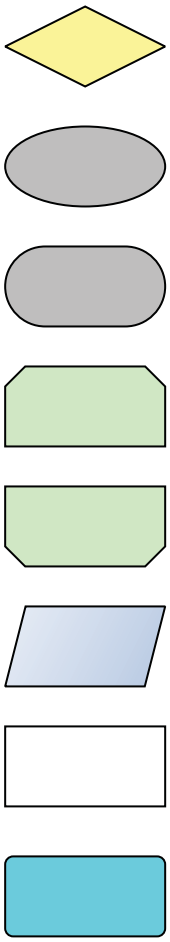
LISTA DE SÍMBOLOS E CONSTANTES FÍSICAS

ρ	Densidade de Probabilidade Radial
Ψ	Função de Onda
r	Raio (coordenadas esféricas)
θ	Azimute – variação $[-\pi, \pi]$ (coordenadas esféricas)
φ	Elevação – variação $[-\pi/2, \pi/2]$ (coordenadas esféricas)
μ	Massa reduzida
∇	Operador Laplaciano
i	Unidade imaginária
Z	Número atômico
Å	Ângstron ($1 = 10^{-10}$ m)
a_0	Raio de Bohr ($a_0 = 5.291772109 \cdot 10^{-11}$ m)*
c	Velocidade da luz no vácuo ($v = 299792458$ m/s)*
$\hbar$	Constante de Planck reduzida ($\hbar = 1.054571817 \cdot 10^{-34}$ J Hz^{-1})*
m_e	Massa do elétron ($m_e = 9.1093837 \cdot 10^{-31}$ kg)*
m_p	Massa do próton ($m_p = 1.672621923 \cdot 10^{-27}$ kg)*
m_n	Massa do nêutron ($m_n = 1.674927498 \cdot 10^{-27}$ kg)*
e	Carga elementar ($e = 1.602176634 \cdot 10^{-19}$ C)*
ϵ_0	Permissividade elétrica no vácuo ($\epsilon_0 = 8.85418781 \cdot 10^{-12}$ C^{2}N^{-1}m^{-2})*
$\varphi(\cdot)$	Função de ativação
σ	Desvio padrão populacional
ρ_P	Correlação de Pearson
r_{eson}	parâmetro de ressonância da rede
n_{iter}	parâmetro número de iterações da rede
n_{layers}	parâmetro de configuração das camadas da rede
b_{ias}	parâmetro de uso de bias
n_{ger}	parâmetro número de gerações da GA
n_{pop}	parâmetro tamanho da população da GA
m_{rate}	parâmetro taxa de mutação da GA
n_{elite}	parâmetro tamanho da elite da GA
n_{iterSA}	parâmetro de número de gerações da SA
t_{emp}	parâmetro de escolha do algoritmo de temperatura da SA
Hr	erro entre histogramas de distribuição radial
$H\phi$	erro entre histogramas de distribuição angular sobre ϕ
$H\theta$	erro entre histogramas de distribuição angular sobre θ
dCM	desvio do Centro de Massa

$\bar{r}$	erro entre os raios médios
RD	radialmente distribuído – homogeneidade da distribuição em todos os ângulos (alg. B.4)

*Os valores das constantes foram obtidas em Mohr, Newell e Taylor (2016) e Linstrom e Mallard (2020).

LISTA DE SÍMBOLOS DO FLUXOGRAMA



SUMÁRIO

1 INTRODUÇÃO

A necessidade de compreender o comportamento molecular da matéria se torna cada vez mais presente em nossas vidas. Contudo, compreender o comportamento da natureza em níveis microscópicos ainda é um desafio a ser superado. Já faz quase 100 anos de história desde o surgimento da interpretação quântica dada por Schrödinger e ainda não conseguimos resolver analiticamente problemas mais complexos que o átomo de hélio sem recorrer a aproximações (EISBERG; RESNICK, 1979).

As aproximações às densidades de probabilidade da posição dos elétrons são de grande utilidade e se mostraram eficientes para representar uma grande variedade sistemas (CAPELLE, 2006), contudo possuem falhas (ZIMMERLI; PARRINELLO; KOUMOUTSAKOS, 2004; VONDRÁŠEK *et al.*, 2005; COHEN; MORI-SÁNCHEZ; YANG, 2008; MCMAHON *et al.*, 2012). Não há na literatura referência do sucesso da observação ou simulação da trajetória de um elétron ligado.

Trabalhos recentes tiveram sucesso em solucionar problemas clássicos da física com o uso de inteligência artificial (IA), tai como o problema de 3 corpos (BREEN *et al.*, 2019), que é analiticamente insolúvel. Trabalhos como este demonstram a utilidade da IA como ferramenta em diversas aplicações analiticamente complexas. Não apenas isso, o aprendizado de máquina (ML, do inglês *machine learning*) vem sendo reconhecido na literatura como um meio alternativo de se simular sistemas químico-quânticos (SELLIER *et al.*, 2019).

As aplicações do ML são amplas, abrangendo vários níveis de complexidade, indo desde a participação na solução de aproximações da mecânica quântica em seu cerne (SNYDER *et al.*, 2012; BROCKHERDE *et al.*, 2017) até o uso em dinâmica molecular para sistemas de muitos corpos tipo *coarsed grained* (ZHANG *et al.*, 2018a). Dentre os vários métodos de aprendizado de máquina, o uso de redes neurais se torna cada vez mais comum para a modelagem molecular pela sua demonstrada versatilidade de aprendizado permitindo solucionar variados problemas (BEHLER; PARRINELLO, 2007; RUPP *et al.*, 2012; HANSEN *et al.*, 2013; LI *et al.*, 2013; DOLGIREV; KRUGLOV; OGANOV, 2016; BEHLER, 2017; CHMIELA *et al.*, 2017; CHMIELA *et al.*, 2018; ZHANG *et al.*, 2018a; BUTLER *et al.*, 2018). Apesar de promissora, o uso de redes exige amostras de dados suficientemente diversas de moléculas e em grande quantidade, além de ser um constante desafio a representação dos dados de forma compreensível para a máquina (MATER; COOTE, 2019). A falta de dados é uma barreira, especialmente, para o aprendizado supervisionado.

A solução da estrutura eletrônica das moléculas, materiais e interfaces é de importância em diversos campos como: física, química, ciência de materiais (MILLS; SPANNER; TAMBLYN, 2017; SELLIER *et al.*, 2019), biologia, farmácia, medicina (CAPELLE, 2006), etc. A reprodução

computacional das propriedades físicas e bioquímicas da matéria por meio de simulações numéricas pode ser realizada por modelagem molecular.

Entende-se por Modelagem Molecular a construção de modelos que "*mimetizam o comportamento de moléculas e sistemas moleculares*", associados a modelos matemático-computacionais. Modelagem aparece aqui como sinônimo de simplificação, considerando que os modelos não reproduzem com perfeição os sistemas reais, mas sim são aproximações destes (LEACH, 2001). A Dinâmica Molecular (MD, do inglês *Molecular Dynamics*) é um conjunto de técnicas computacionais que aplicam estes modelos no tempo, permitindo o estudo do movimento e interação das moléculas. Como consequência, consegue-se reproduzir e até predizer o que é observado experimentalmente (RAPAPORT, 1995a).

A dinâmica molecular é uma área de estudo com diversas aplicações. Dentre elas, destacam-se o desenho de fármacos (DOERR *et al.*, 2016), o estudo de materiais diversos, como por exemplo o grafeno (ZHANG *et al.*, 2015), estudos em toxicologia (SELVARAJ *et al.*, 2018), de materiais condutores (ZHANG *et al.*, 2018b), a formação de moléculas em ambiente interestelar (PEZZELLA; UNKE; MEUWLY, 2018), inclusive construções completas de capsídeos virais (PERILLA *et al.*, 2016), etc. A MD contempla a configuração no espaço (conformação das moléculas), a obtenção de propriedades como as termodinâmicas, e a dinâmica em si (movimento e reações químicas) (KARPLUS; MCCAMMON, 2002; PERILLA *et al.*, 2015).

Quando falamos em modelagem química a nível de elétrons e núcleos, a equação de Schrödinger é consenso (SELLIER *et al.*, 2019), e esta, por sua vez, é de difícil solução. Como alternativa, as aproximações permitem resolver problemas com múltiplos elétrons e átomos, mas quanto mais partículas envolvidas maior é a dificuldade em resolvê-lo. Temos em mãos a dificuldade em se representar espacial e temporalmente o elétron e, consequentemente, moléculas e suas reações, e aqui encontra-se o principal desafio a ser solucionado.

Quando se busca qualidade nas simulações, os métodos quânticos são a melhor opção. Contudo, possuem como desvantagem o seu grande custo computacional, exigindo grande processamento e, principalmente, tempo, além de possuírem falhas em diversas aplicações (ZIMMERLI; PARRINELLO; KOUMOUTSAKOS, 2004; VONDRÁŠEK *et al.*, 2005; COHEN; MORI-SÁNCHEZ; YANG, 2008; MCMAHON *et al.*, 2012).

A constante busca por melhores modelos é uma realidade. A dificuldade na representação do elétron espacial e temporalmente está relacionado à incapacidade na solução analítica de elementos mais pesados que o hidrogênio. Esses são os problemas essenciais ainda existentes na literatura, e este será o foco de nossos esforços.

Não encontramos na literatura nenhum trabalho que busque obter trajetórias ou uma representação pontual do elétron no tempo utilizando recursos do ML. Assim como não foram encontradas aplicações das redes para a previsão de energias do elétron ligado a um núcleo. Também não foram encontrados na literatura trabalhos que utilizem aprendizado por reforço para o estudo de sistemas atômico-moleculares, exceto alguns trabalhos recentes voltados para o *design* molecular (POPOVA; ISAYEV; TROPSHA, 2018). Os achados referentes às atualidades

e conhecimentos no campo da física quântica, referentes à essas observações, são discutidos no Capítulo 2 de revisão.

Aqui propomos a utilização de inteligência artificial na busca por um melhor modelos de descrição atômico-molecular. Dentre os vários métodos de IA existentes, são investigados os com potencial para implementação em MD.

1.1 PROPOSTA

Propomos o desenvolvimento de um modelo para a descrição temporal do elétron utilizando-se apenas de informações estatísticas de sua posição, provenientes da teoria de Schrödinger. O método para o desenvolvimento de tal modelo será baseado em inteligência artificial com ênfase no uso de redes neurais artificiais pois, como será melhor detalhado no Capítulo 4 da proposta, se mostrou a melhor método a ser empregado para o presente propósito.

Como primeira aplicação, escolhemos o átomo de hidrogênio em seu nível fundamental e isolado de forças externas (não perturbado). Essa escolha se deve ao fato do hidrogênio ser o átomo mais simples, e por ser este o único elemento com solução analítica conhecida, e portanto com densidade probabilidade do elétron conhecida e exata (EISBERG; RESNICK, 1979).

O movimento do elétron não ocorre a uma distância fixa do núcleo, como descrito classicamente, mas numa coleção de raios variados conhecidos apenas do ponto de vista estatístico. Além disso, não possuímos a coleção completa, uma vez que ocorrem 10^{15} órbitas do elétron em um único segundo. O que conhecemos é a existência de um raio mais provável, o raio de Bohr, e a existência de certa distribuição, que obtemos pelas equações de Schrödinger. Não conseguimos saber onde o elétron está a cada instante, portanto o único modo de se realizar uma predição é utilizando a estatística.

Os métodos de inteligência artificial podem ser supervisionados e não supervisionados. Pelo fato de não se possuir trajetórias conhecidas para o elétron, apenas estatísticas, não podemos utilizar métodos de treinamento supervisionado, onde a posição do elétron é fornecida sequencialmente ponto a ponto, o que vamos denominar de avaliação local. Como apenas conhecemos o panorama, o comportamento do átomo após várias órbitas realizadas pelo elétron, a avaliação deve ser sobre este panorama – o que denominamos de avaliação global. Esta é uma das principais metas de avaliação.

O comportamento do elétron possui características bem definidas que precisam ser avaliadas sobre o resultado global (uma sequência temporal) fornecido pelo modelo, que será baseado em uma rede neural como mencionado anteriormente. Tal resultado irá definir a qualidade do resultado dessa rede.

Esse tipo de avaliação exige uma rede com função de custo maleável, ou seja, que seja capaz de avaliar características complexas do sistema e não apenas distâncias como nas funções tradicionais de classificação e regressão. Com o objetivo de incorporar o modelo de

representação do elétron ligado à MD, consideramos que o modelo deve ser veloz em fornecer uma previsão.

Nossa proposta não pode ser considerada um modelo para a trajetória do elétron, mas sim um modelo do comportamento do elétron. Nesta proposta, objetivamos que a rede aprenda a representar o elétron tanto como partícula quanto como onda, i.e., que aprenda a representar ambos os aspectos do comportamento do elétron. O conceito da dualidade partícula-onda é desafiador, pois não conseguimos pensar de forma dual, uma vez que são familiares para nós apenas os conceitos de partícula ou de onda. Essa nossa incapacidade abre espaço para que uma IA realize este trabalho por nós, e aprenda como representar esse comportamento.

Tendo em vista a limitação dos modelos de aproximação quântica para N-corpos (MCMAHON *et al.*, 2012; COHEN; MORI-SÁNCHEZ; YANG, 2008; VONDRÁŠEK *et al.*, 2005), e a inabilidade dos métodos de aprendizado de máquina existentes satisfazerem os requisitos para a construção do modelo (detalhado no Capítulo 4), propomos a construção de um modelo preliminar de aprendizado baseado em rede neural artificial para a representação do sistema atômico. Tal aprendizado será baseado em informações de densidade de probabilidade do elétron. Não foi encontrado nenhum modelo que utilize as curvas de densidade de probabilidade para prever as posições do elétron, o que torna a proposta inovadora.

1.2 OBJETIVOS

Possímos dois objetivos gerais: (1) desenvolver um modelo para representação do comportamento elétron ligado, por meio de posições pontuais, alternativo aos métodos de aproximação;(2) desenvolver uma rede neural artificial com aprendizado não supervisionado por meio de reforço global para propósitos de generalização.

Como objetivos específicos, visamos:

a) realizar um levantamento de características que descrevem o comportamento do átomo de um elétron;

b) definir os requisitos do método para construir um modelo de representação do sistema;

c) com base nos requisitos, desenvolver uma nova rede neural artificial capaz de satisfazê-los;

d) validar a nova rede neural;

e) desenvolver um modelo capaz de representar o comportamento do elétron utilizando a nova rede desenvolvida.

1.3 ORGANIZAÇÃO DA DISSERTAÇÃO

O presente trabalho divide-se em 8 Capítulos, e seu conteúdo é disposto como descrito abaixo. O referencial teórico abrangerá os Capítulos 2 e 3: no cap. 2 são levantados os fundamentos básicos da mecânica quântica e métodos computacionais para modelagem quântica relevantes; e no cap. 3 são apresentados os conceitos básicos das técnicas de IA utilizadas, assim como um levantamento de modelos de ML relevantes, não se limitando a redes neurais. No Capítulo 4 detalha-se a proposta do trabalho, os requisitos para o método, assim como a estrutura da rede desenvolvida em detalhes. A metodologia utilizada encontra-se no Capítulo 5. O Capítulo 6 traz os resultados, que incluem a validação da rede proposta, e os resultados do modelo treinado para o comportamento do elétron. Segue-se a discussão no Capítulo 7, e a conclusão e perspectivas para próximos trabalhos no Capítulo 8. Por fim, são fornecidos três anexos, o primeiro com a descrição dos conjuntos de dados (*Benchmarks*) utilizados para validação da rede juntamente a gráficos e tabelas adicionais, que complementam os resultados. O segundo anexo com códigos utilizados e considerados relevantes. E o terceiro com os gráficos resultantes das melhores redes treinadas.

2 FUNDAMENTAÇÃO NA FÍSICA COMPUTACIONAL E MECÂNICA QUÂNTICA

Inicialmente apresentamos uma breve fundamentação que traz as informações mais relevantes com relação à mecânica quântica que serão consideradas no desenvolvimento de um modelo para a descrição do comportamento do átomo de um elétron. Buscamos responder 'o que é', 'onde está', e 'como se comporta' o elétron no átomo de hidrogênio em seu estado fundamental, informações utilizadas para obter o modelo desenvolvido neste trabalho. Em seguida abordamos alguns tópicos relacionados à modelagem quântica computacional e levantamos suas vantagens e desafios.

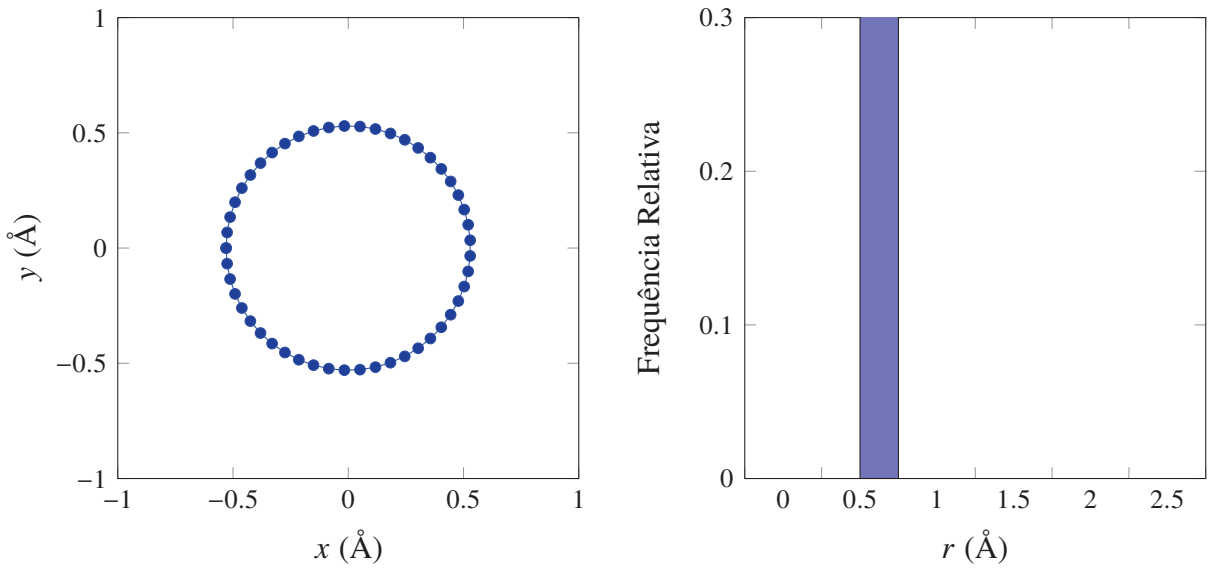
2.1 FUNDAMENTOS DA MECÂNICA QUÂNTICA

A física clássica é capaz de explicar os fenômenos macroscópicos do nosso dia-a-dia. Entretanto, quando adentramos no universo microscópico, isso deixa de ser verdade. Tomemos um exemplo simples: quando descrevemos classicamente o comportamento do elétron ao redor de um núcleo. Se realizarmos uma integração inocente, sem considerar uma série de condições e ação de diversas forças, utilizando apenas a força eletrostática de Coulomb como fonte de aceleração entre as partículas, obteremos uma órbita plana. A trajetória resultante executada pelo elétron será circular se o raio inicial da órbita do elétron tiver um raio de Bohr de distância (ou 0.529Å), como apresentado na Figura 2.1, ou será elíptica se o raio inicial for diferente. Estamos desconsiderando o movimento do núcleo, já que é mínimo em comparação com o do elétron. Na mesma figura podemos observar um histograma que representa sua distribuição no tempo. Como a trajetória possui raio fixo, o histograma de sua distribuição será pouco distribuído, limitando-se a uma única barra com 100% de probabilidade de encontrar o elétron no raio de Bohr.

Quando se observa, entretanto, o comportamento real das partículas, descobrimos que existem órbitas proibidas e permitidas, e que a energia é quantizada em pacotes chamados atualmente de fótons (EISBERG; RESNICK, 1979, p.151-161), fenômenos que não podem ser explicados pela física clássica (JENNINGS; LEIFER, 2016; HOFMANN, 2016). Como afirmam Jennings e Leifer (2016) dramaticamente, a física clássica morreu há um século com o advento da mecânica quântica. Diversos fenômenos quânticos não podem ser explicados classicamente, dentre os quais: a aleatoriedade, discretização, indistinguibilidade dos estados, incerteza da medida, perturbação da medida, complementariedade, interferência, colapso dos pacotes de onda, etc.

Pela Interpretação de Copenhagen da mecânica quântica – que é a mais aceita pela comunidade – o comportamento das partículas passa a ser descrito pela equação de onda de Schrödinger. Nesse modelo, o elétron passa a ter descrição ondulatória e, como consequência, uma densidade de probabilidade específica, como apresentado no gráfico da Figura 2.2.

FIGURA 2.1: INTEGRAÇÃO CLÁSSICA DA INTERAÇÃO NÚCLEO-ELÉTRON.



FONTE: O autor (2020).
LEGENDA: Foi considerada apenas a força Coulombiana como meio de interação entre as partículas. Se o ponto inicial for justamente sobre o raio de Bohr (0.529 Å), a trajetória do elétron será circular.

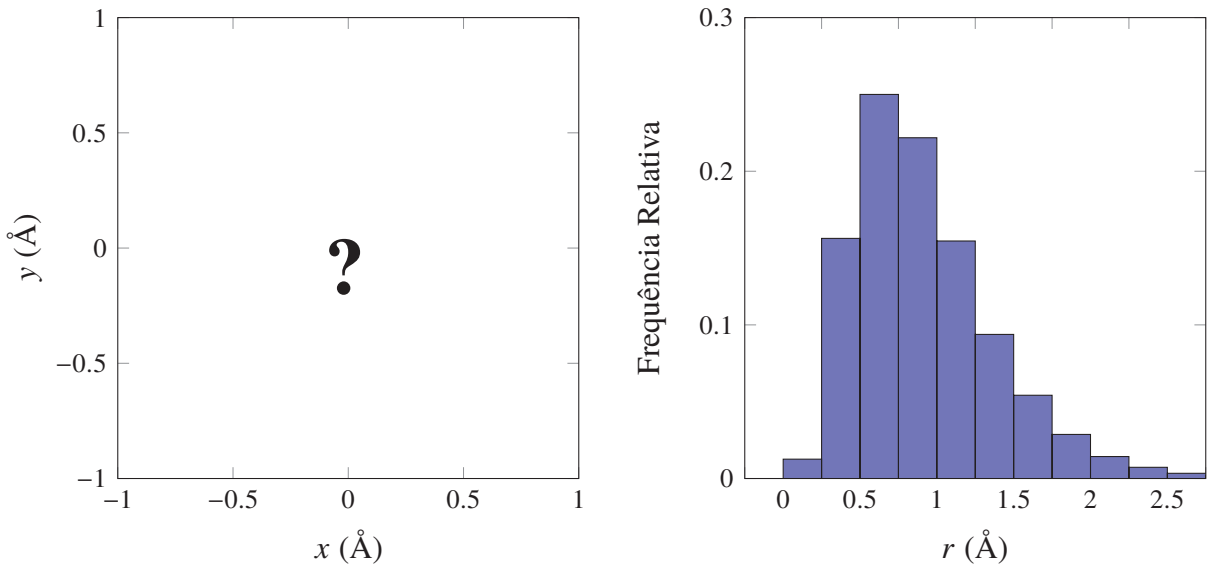
Apesar de se conhecer esta densidade de probabilidade, a trajetória das partículas não pode ser observada por experimentação. A observação sequencial das partículas não é possível, como será explicado com mais detalhes na Subseção 2.1.2. Também não existe teoria ou modelo que consiga fornecer uma trajetória temporal do elétron. A Interpretação de Copenhagen é puramente estatística e independente do tempo, como será melhor detalhado na Subseção 2.1.3 (EISBERG; RESNICK, 1979, p.171-189). Existe uma teoria alternativa menos conhecida e menos aceita que é a teoria Bohmiana. Ela possui uma representação determinística, permitindo obter a trajetória das partículas (BÖHM, 1952). Contudo, apesar de ser bastante antiga, poucos são os estudos utilizando esta perspectiva, e além de ser um modelo de difícil implementação, seus resultados ainda estão "engatinhando" e exigem muito mais estudos (SCHLOSSHAUER, 2005).

Buscamos agora definir o que podemos afirmar sobre o comportamento do elétron. Para isso, faremos uma breve explanação sobre o que há de mais concreto na literatura.

2.1.1 Experimento da Dupla Fenda de Young

Este experimento ajuda-nos a compreender o que realmente é o elétron. Classicamente, o elétron era descrito como uma partícula. O experimento da dupla-fenda de Young em 1802 revelou que o comportamento de fótons, ao passar pelas duas fendas, não era como partículas, mas sim dual partícula-onda (YOUNG, 1802; JIN *et al.*, 2010). Posteriormente Broglie (1925) experimentou e demonstrou esse comportamento para os elétrons. O comportamento dual já foi demonstrado para a luz, elétrons, nêutrons, átomos e inclusive moléculas (JIN *et al.*, 2010).

FIGURA 2.2: REPRESENTAÇÃO QUÂNTICA DA DISTRIBUIÇÃO DO ELÉTRON.



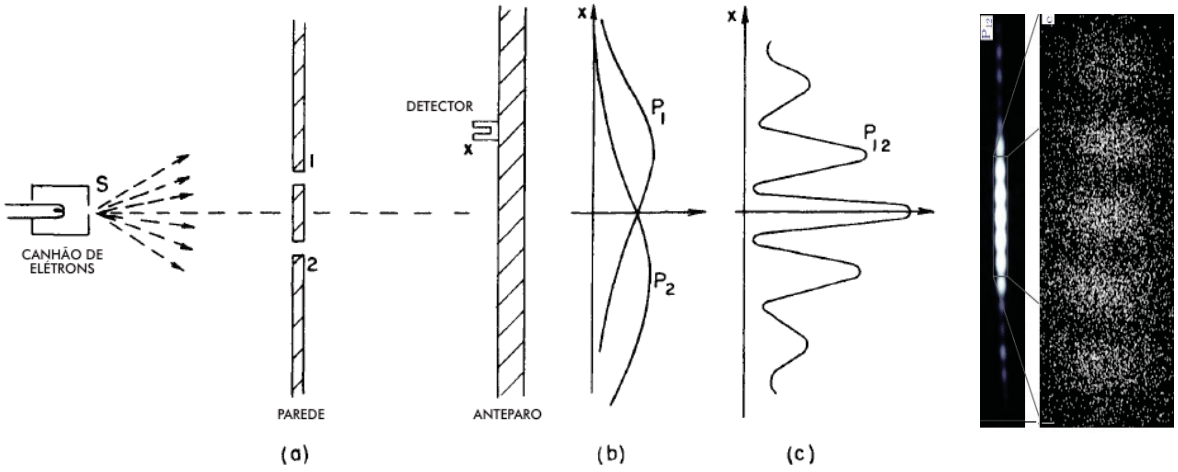
FONTE: O autor (2020).
LEGENDA: Representação quântica da distribuição do elétron com base na densidade de probabilidade do elétron de hidrogênio em nível n=1. Não há trajetórias conhecidas para o elétron em torno do núcleo.

O experimento consiste em um emissor de elétrons que emite elétrons um a um. Esses elétrons emitidos passam por uma parede com 2 fendas e colidem então em um anteparo que registra a colisão (Figura 2.3). Por intuição imaginamos que, independentemente se elétrons, fótons, ou qualquer outra partícula, irá formar um padrão de duas colunas no anteparo, i.e, a partícula passará por uma ou por outra fenda. Contudo, o observado é um padrão de interferência como apresentado na Figura 2.3(d). A analogia comumente usada é o de uma pedra que cai na água e forma uma onda na superfície. Ao atravessar as duas fendas, a onda se dividiria em duas e do outro lado da parede formaria um padrão de interferência (FEYNMAN; LEIGHTON; SANDS, 2008a; BACH *et al.*, 2013; EISBERG; RESNICK, 1979).

A grande conclusão que podemos obter desse experimento é que o elétron é corpuscular, mas possui propriedades ondulatórias, consistindo a dualidade partícula-onda. Corpuscular pois atinge um ponto único no anteparo, mas como propriedades ondulatórias pois gera interferência sobre si mesmo.

O Princípio da Complementaridade de Bohr nos fornece uma luz na compreensão da natureza dual. Quando tentamos determinar a natureza da radiação ou matéria como ondulatória ou corpuscular, podemos ter dois possíveis resultados: ou o experimento irá realçar suas características ondulatórias fazendo desaparecer o caráter corpuscular, ou justamento o oposto, evidenciar o caráter corpuscular em detrimento do ondulatório. Nunca se consegue observar ambas faces do comportamento em detalhes (EISBERG; RESNICK, 1979, p.87-114).

FIGURA 2.3: EXPERIMENTO DE INTERFERÊNCIA COM ELÉTRONS.



FONTE: (a-c) Feynman, Leighton e Sands (2008a, p.3-1 a 3-7);
(d) Modificado de: Bach *et al.* (2013)

LEGENDA: No esquema apresenta-se uma fonte de elétrons lançados contra uma parede com duas fendas em (a), que colidem com um anteparo. O padrão resultante apresentado em (c) é decorrente da sobreposição de duas frentes de onda (b), ocasionando o padrão de interferência. Em (d) apresenta-se o resultado do experimento. Cada ponto representa a colisão de um elétron contra o anteparo, resultando no padrão apresentado.

2.1.2 Princípio da Incerteza de Heisenberg

Tendo conhecimento do que se trata um elétron, a próxima pergunta está em saber onde este elétron se localiza no tempo. O Princípio de Incerteza de Heisenberg nos ajuda a compreender porque é tão difícil localizá-lo. Este afirma que não podemos medir, simultaneamente e com precisão, posição e momento linear (ou velocidade) de uma partícula (LANDAU; LIFSHITZ, 1977; HEISENBERG, 1985).

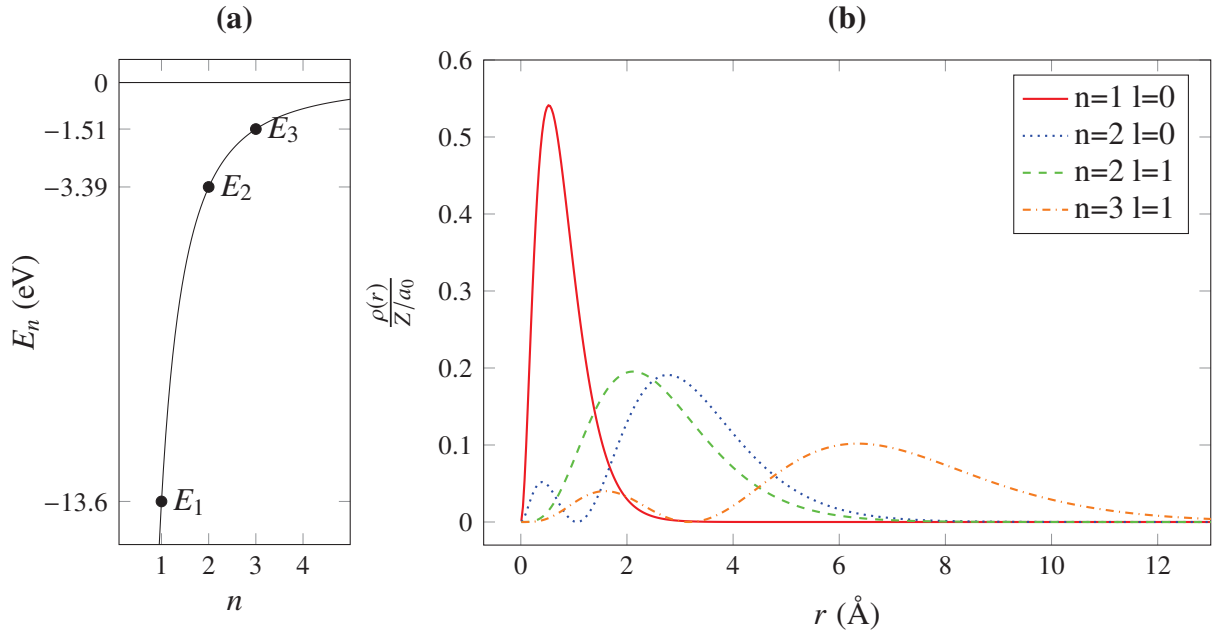
O princípio da incerteza aponta que o caminho para a compreensão dos sistemas quânticos reside na estatística de probabilidades. Classicamente podemos prever o comportamento de um sistema como um todo tomando conhecimento da posição e momento das partículas em dado instante do tempo. Infelizmente isso não é viável para um sistema microscópico, já que não se pode conhecer posição e momento instantâneo de uma partícula nessa escala. O que podemos ter são apenas probabilidades sobre seu comportamento (EISBERG; RESNICK, 1979, p.112-114).

A incerteza de Heisenberg pode ser expressa em uma equação:

$$\Delta x_i \Delta p_i \geq \frac{\hbar}{2} \tag{2.1}$$

onde Δx_i é a incerteza da medida da posição e Δp_i é a incerteza da medida do momento linear. O produto dessas incertezas não pode ser inferior à constante de Planck sobre 4π, o que delimita a precisão da medida (EISBERG; RESNICK, 1979, p.112-114).

FIGURA 2.4: DENSIDADE DE PROBABILIDADE RADIAL PARA O ÁTOMO DE UM ELÉTRON E SEUS NÍVEIS DE ENERGIA.



FONTE: O autor (2020).
LEGENDA: n e l são os números quânticos principal e azimutal, respectivamente. $n = 1, l = 0$ corresponde ao estado fundamental, e os demais aos estados excitados. O esquema **(a)** apresenta os níveis de energia E_n do elétron para cada número quântico principal n. **(b)** apresenta a densidade de probabilidade $\frac{\rho(r)}{Z/a_0}$ de se encontrar o elétron no raio r para cada combinação de números quânticos, onde Z é o número atômico ($Z = 1$), a_0 o raio de Bohr e $\phi(r)$ é a probabilidade de se encontrar o elétron entre r e $r + dr$.

A constante de Planck é uma constante relacionada com a quantidade de energia contida em um fóton $E = h\nu$, onde ν é a frequência da radiação, e E a sua energia, correspondente a um quantum de energia. A energia é quantizada, portanto, há níveis energéticos permitidos e proibidos, e sua unidade é dada por essa relação.

2.1.3 Equação de Schrödinger

Como não podemos saber passo a passo por onde anda o elétron, podemos tentar compreender como seu comportamento ondulatório afeta a probabilidade de encontrá-lo em certo lugar em torno do núcleo. A teoria de Schrödinger é uma das bases da mecânica quântica, e uma generalização das leis do movimento de Newton que funciona bem para corpos microscópicos.

A teoria de Schrödinger envolve a previsão de autovalores e autofunções, conceitos provenientes da álgebra que permitem descrever os estados dos sistemas quânticos. Com as autofunções em mãos, se torna possível determinar as densidades de probabilidades, isto é, a probabilidade de se encontrar o elétron em dado ponto no espaço, como apresentado na Figura 2.4 (EISBERG; RESNICK, 1979, p.301-327). Na verdade, com a equação de Schrödinger pode-se explicar quase todos os fenômenos atômicos, com exceção do magnetismo e relatividade.

Apesar de possuir a capacidade de explicar os fenômenos de interação entre partículas, a mecânica quântica é capaz de obter solução *exata* apenas para o átomo de um elétron – o hidrogênio – o sistema quântico mais simples existente. O átomo de Hélio, que possui apenas dois elétrons, não possui solução analítica. Isto é, apesar de quase um século[1] após a elaboração da equação de Schrödinger, ainda não se conseguiu obter uma solução exata para sistemas mais complexos que um hidrogênio. Deve-se observar aqui que estamos analisando apenas a eletrosfera, pois a estrutura nuclear, inclusive do hidrogênio, é ainda mais complexa (MCMAHON *et al.*, 2012).

Para o átomo de hélio ainda existem aproximações de grande precisão, entretanto, para sistemas mais complexos a solução das equações se torna tão complicada que passam a ser utilizadas aproximações da realidade, muitas das quais bastante rudimentares (FEYNMAN; LEIGHTON; SANDS, 2008b, p.16-14 a 16-15). Neste ponto entra o uso de modelos de aproximação como método de Hartree-Fock e *Density Functional Theory* (DFT) para a solução de sistemas envolvendo múltiplos corpos (problemas de 3 ou mais corpos).

O hidrogênio é o átomo mais simples e de solução precisa e confiável. Mas, como descrever um sistema quântico? A autofunção Ψ, ou função de onda dependente do tempo, descreve o estado do sistema e é dada pela relação de igualdade:

$$\left[\frac{-\hbar^2}{2\mu}\nabla^2 + V(x,y,z)\right]\Psi(x,y,z,t) = i\hbar\frac{\partial}{\partial t}\Psi(x,y,z,t) \tag{2.2}$$

onde $\hbar$ é a constante de Planck reduzida e Ψ é a função de onda (EISBERG; RESNICK, 1979, p.301-327). μ representa a massa reduzida, dada por

$$\mu = \frac{M}{M+m_e}m_e,$$

onde M a massa nuclear e m_e a massa do elétron. V é o potencial Coulombiano dado por

$$V(x,y,z) = \frac{-Ze^2}{4\pi\epsilon_0\sqrt{x^2+y^2+z^2}}, \tag{2.3}$$

onde Z é o número atômico do núcleo. ∇^2 é o operador Laplaciano que em coordenadas cartesianas é dado por

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}, \tag{2.4}$$

e em coordenadas esféricas, como será utilizado a seguir, é dado por

$$\nabla^2 = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial}{\partial r}\right) + \frac{1}{r^2\,\mathrm{sen}^2\varphi}\frac{\partial^2}{\partial\theta^2} + \frac{1}{r^2\,\mathrm{sen}\varphi}\frac{\partial}{\partial\varphi}\left(\mathrm{sen}\varphi\frac{\partial}{\partial\varphi}\right), \tag{2.5}$$

[1] A equação de Erwin Schrödinger foi publicada em 1926 como uma solução válida para a mecânica quântica (SCHRÖDINGER, 1926), e é amplamente aceita pela comunidade na chamada Interpretação de Copenhagen.

onde r é o raio, θ é o azimute, variando de $[-\pi, \pi]$, e φ é a elevação, variando de $[-\pi/2, \pi/2]$.

Podemos enxergar a equação de Schrödinger (eq. 2.2) como um problema de dois corpos transformado em um problema de apenas um corpo (o elétron). Isso se deve ao fato de que o núcleo é centralizado na origem e apenas o movimento do elétron é considerado. A equação 2.2 de Schrödinger dependente do tempo não é solucionável neste formato, e exige simplificação. Pode-se, por meio da técnica de separação de variáveis, obter a equação de onda independente do tempo, através de relações como

$$\Psi(x, y, z, t) = \psi(x, y, z)\ \exp -iEt/\hbar, \tag{2.6}$$

que mantém a relação na equação independente do tempo

$$\left[\frac{-\hbar^2}{2\mu}\nabla^2 + V(x, y, z)\right]\psi(x, y, z) = i\hbar\frac{\partial}{\partial t}\psi(x, y, z). \tag{2.7}$$

A função de onda independente do tempo, $\psi(x, y, z)$, pode ser resolvida com uma nova separação de variáveis, decompondo-a em três funções, cada qual dependente de apenas uma das coordenadas. A solução pode ainda ser obtida mais facilmente utilizando-se coordenadas esféricas: porções radial $R(r)$ e angulares $\Theta(\theta)$ e $\varphi(\varphi)$

$$\psi(x, y, z) = R(r)\Theta(\theta)\varphi(\varphi). \tag{2.8}$$

A solução da equação de Schrödinger independe do tempo (eq. 2.8) origina os chamados três números quânticos, cada qual proveniente de uma das três funções de coordenadas espaciais esféricas. Os números quânticos são: n - principal, l azimutal, e m_l - magnético.

Cada combinação de números quânticos corresponde a um nível energético. O estado fundamental do hidrogênio, ou seu estado de menor energia, é o estado mais estável correspondente aos números quânticos principal $n = 0$, azimutal $l = 0$ e magnético $m_l = 0$, com valor de -13.6 eV. Os demais níveis energéticos são denominados estados excitados (para o átomo de hidrogênio, que possui apenas um elétron), e possuem energia maior que o fundamental, sendo mais instáveis e capazes de liberar fótons de energia até alcançar o estado fundamental. A Figura 2.4 (a) apresenta os níveis energéticos dos primeiros números quânticos, onde é possível observar que a energia é definida pelo número quântico principal[2] (EISBERG; RESNICK, 1979, p.301-327).

[2]Não estaremos aqui levando em conta a estrutura fina e hiperfina, que demonstra haver nuances nos níveis energéticos com relação aos números quânticos azimutal, magnético, momento angular, spin e efeito relativístico (EISBERG; RESNICK, 1979, p.301-327).

As soluções particulares para cada combinação de números quânticos é obtida resolvendo o Laplaciano da função de onda (eq. 2.5). Tomando a solução para o estado fundamental[3], obtemos

$$\psi_{100} = \frac{1}{\sqrt{\pi}} \left(\frac{Z}{a_0} \right)^{3/2} e^{-Zr/a_0}, \tag{2.9}$$

onde '100' corresponde aos três números quânticos $n = 1, l = 0$ e $m_l = 0$, respectivamente, e a_0 é o raio de Bohr ou o raio da órbita do elétron do átomo de hidrogênio. É possível observar que a função é independente das coordenadas angulares, variando apenas com relação ao raio r. Isso caracteriza a distribuição de posições do elétron no tempo como homogênea para todos os ângulos θ e φ, apresentando simetria esférica. Todas as direções são equivalentes e igualmente prováveis, ou, de outra maneira, a distribuição radial é igual para todos os ângulos observados.

Multiplicando a função de onda por seu conjugado, $\psi^*\psi$, obtemos a densidade de probabilidade, isto é, a probabilidade de se encontrar o elétron em dado ponto, neste caso, com relação a distância do núcleo. A Figura 2.4 (b) apresenta a distribuição radial para o nível fundamental e os primeiros níveis excitados que também são independentes das coordenadas angulares. Os níveis excitados são apresentados apenas para fins comparativos, pois o enfoque será dado sobre o estado fundamental.

Para o estado fundamental, a maior densidade de probabilidades se dá em torno do raio de Bohr ($a_0 = 0.529$ Å), que também é denominado de raio mais provável. O raio médio $\bar{r}$ (ou raio esperado) possui valor de $\bar{r} = 0.7935$ Å. Esse valor é obtido da integração da probabilidade vezes a posição (EISBERG; RESNICK, 1979, p.189-199)

$$\begin{aligned} \bar{r} &= \int r P(r,t) dr, \\ \bar{r} &= \frac{3}{2} a_0 \;=\; 0.7935\,, \end{aligned} \tag{2.10}$$

onde P é a probabilidade de se encontrar o elétron em dada posição r em dado instante t.

2.2 MODELAGEM QUÂNTICA

O que realmente conhecemos sobre o comportamento das partículas? O apresentado na Seção 2.1 fornece um bom panorama do que há de mais concreto na teoria quântica. Entretanto, existem várias vertentes teóricas sobre a mecânica quântica, que incluem a teoria de vários mundos e a teoria bohmiana, além da vertente mais aceita que é a Interpretação de Copenhagen, que foi apresentada na seção anterior. Mas qual delas é a que mais se aproxima da realidade? Esta é uma área de estudo que enfrenta desafios constantemente. Como levantado na "Conferência em Oxford de Física Quântica e a Realidade da Natureza" de 2010 – *The Oxford Conference on Quantum Physics and the Nature of Reality*, publicado por Briggs, Butterfield e Zeilinger (2013)

[3]Não serão apresentadas as soluções das funções de onda neste trabalho, apenas seus resultados mais relevantes. A resolução exige vários passos e não acrescenta nenhuma informação relevante para o propósito deste trabalho, no que diz respeito ao desenvolvimento de um modelo para o comportamento do elétron.

– existem inúmeras questões a serem respondidas, e não apenas isso, há uma série de dúvidas sobre o que realmente conhecemos.

Como conclusão do artigo, apresentam-se as duas principais "nuvens" que pairam sobre o horizonte: a primeira delas remete ao velho problema da medição quântica (incerteza de Heisenberg) e a dificuldade em explicar o mundo clássico a partir da mecânica quântica (generalizar a mecânica quântica para o mundo macroscópico). A segunda nuvem referida está com relação à teoria quântica da gravidade, já que as demais forças fundamentais – eletromagnética, fraca e forte – já possuem explicação quântica bem-sucedida Briggs, Butterfield e Zeilinger (2013); contudo esse segundo aspecto não vem ao caso para o presente trabalho.

Notamos que mais uma importante questão foi deixada de fora da lista: a função de onda é mesmo uma entidade física real, ou é apenas um instrumento matemático que facilita a interpretação da realidade? Esta discussão foi lançada por Merali (2015) com a pergunta "*O que é realmente real?*". Esta é uma questão recorrente que divide a comunidade em 2 vertentes: os chamados *psi-epistemic* afirmam que a função de onda não existe, que se trata de uma interpretação matemática da realidade; e os *psi-ontic* apoiam a teoria de que a função de onda existe, é uma entidade real. Tomando como exemplo o experimento mental do gato de Schrödinger, pela visão *psi-epistemic*, na caixa haverá apenas um dos possíveis estados para o gato, apenas não sabemos qual é; enquanto que pela *psi-ontic* os dois estados existem simultaneamente até o colapso dos estados (MERALI, 2015).

Vários trabalhos recentes apoiam a vertente *psi-ontic*, de que a função de onda é uma entidade física e real, mas não descartam totalmente sua vertente alternativa Ringbauer *et al.* (2015), Zhou *et al.* (2017). Isso indica que utilizar as funções de onda, da teoria de Schrödinger, é um bom caminho para se chegar ao comportamento real da natureza.

Zhou *et al.* (2017) realiza um experimento de dupla-fenda "trançada" (*twisted double-slit experiment*) com fótons e conseguem demonstrar a causalidade dos eventos temporalmente. Queira dizer, os autores conseguem observar a existência de fótons individuais após a passagem pela fenda, o que apresenta o aspecto determinístico da evolução do sistema no tempo, permitindo a previsão dos eventos. Suas descobertas enfatizam a causalidade dos eventos como no colapso das funções de onda, descartando a aleatoriedade do sistema.

Trabalhos como o de Ringbauer *et al.* (2015) e Zhou *et al.* (2017) justificam o uso da equação de onda de Schrödinger como melhor maneira de descrever os sistemas microscópicos. Contudo, não são descartadas ainda outras vertentes como a Bohmiana, sobre as quais não adentraremos.

A maioria dos trabalhos também utilizam a Interpretação de Copenhagen como base para a modelagem de sistemas quânticos. Contudo, o descrito na Seção 2.1 é aplicado unicamente para o átomo de hidrogênio. Quando se deseja modelar sistemas mais complexos que isso, a solução da equação 2.7 se torna inviável. Os cálculos se tornam tão complexos que seria impossível encontrar uma solução exata. Por este motivo, para sistemas de N-corpos (como

átomos multieletrônicos) é necessário utilizar métodos de aproximação (EISBERG; RESNICK, 1979).

2.2.1 Métodos de Aproximação da Equação de Schrödinger para sistemas de muitos corpos

Há principalmente duas abordagens da mecânica quântica que são utilizadas: as *ab initio* e as semi-empíricas. As *ab initio* são mais precisas e também demandam maior poder computacional. Elas obtêm soluções aproximadas, não empíricas, da equação de Schrödinger independente do tempo. Os métodos mais utilizados são o Hartree-Fock, mais antigo, e o *Density Functional Theory* (DFT), que é computacionalmente mais eficiente que o primeiro (SCHLICK, 2010).

As *semi-empíricas* utilizam um atalho empírico que reduz o número de cálculos. Ao invés de realizar todas as aproximações computacionalmente, elas utilizam valores tabelados, parâmetros obtidos experimentalmente. Estes parâmetros são geralmente energias conhecidas de orbitais atômicos e/ou geometrias dos orbitais. Isso torna o sistema mais parecido com os dados experimentais (SCHLICK, 2010), contudo engessado no que diz respeito a possibilidade de se explorar novas possibilidades de interações e geometrias.

Hartree-Fock, ou Método de Campo Autoconsistente, é uma solução aproximada da equação de Schrödinger realizada iterativamente. Inicialmente, são consideradas apenas as forças mais relevantes atuantes sobre o elétron, e o movimento dos elétrons é tratado como independente da presença dos demais elétrons. Essa aproximação é necessária para evitar a solução (ou a sua incapacidade) da equação de Schrödinger dependente do tempo para múltiplas partículas, e preferindo-se resolver a equação independente do tempo para uma única partícula (EISBERG; RESNICK, 1979, p.406-411).

DFT, um dos sucessores do método de Hartree-Fock, foi um método pioneiro no cálculo da estrutura eletrônica para a matéria condensada (MARTIN, 2004). Seu princípio baseia-se na possibilidade de descrever sistemas de múltiplos corpos em interação como um funcional[4] (HOHENBERG; KOHN, 1964). No entanto, não se conhecem funcionais exatos para descrever qualquer sistema de múltiplos elétrons. O uso dos métodos DFT se baseiam em funcionais aproximados aos estados fundamentais de muitos elétrons, funcionais derivados das equações de Kohn-Sham (KOHN; SHAM, 1965).

O DFT já demonstrou ser extremamente útil e versátil, permitindo sua aplicação em diversas áreas como no estudo da supercondutividade, efeitos relativísticos em elementos pesados, núcleo atômico, líquidos clássicos, propriedades magnéticas, entre outros. A sua vantagem também está no custo computacional menor que outros métodos como Hartree-Fock e suas vertentes (CAPELLE, 2006).

Apesar de muito vantajosa, há ainda uma série de dificuldades a serem superadas. Como citado por McMahon *et al.* (2012), as técnicas computacionais modernas como o DFT

[4]Funcional é uma classe de funções algébricas.

possuem dificuldade em calcular com precisão os átomos de hidrogênio e hélio, se tornando piores conforme os elementos se tornam mais pesados. O modelo DFT também apresenta erros no cálculo de energia na delocalização – que ocorre em moléculas como H_2^+, ozônio e benzeno (COHEN; MORI-SÁNCHEZ; YANG, 2008) – além de dificuldades na modelagem da força de Van der Waals (VONDRÁŠEK *et al.*, 2005), entre outros (BUTLER *et al.*, 2018). Para cada um desses erros, exigem-se correções específicas na modelagem, como correções para a força de Van der Waals (LILIENFELD *et al.*, 2004) e para interações aromáticas com a água, inclusive podendo haver a necessidade de correções a cada nova classe de sistemas(ZIMMERLI; PARRINELLO; KOUMOUTSAKOS, 2004).

Como mencionado, a constante necessidade de melhorias nos modelos é uma realidade. O uso de métodos alternativos pode ser uma solução para este tipo de problema. O aprendizado de máquina vem se mostrando efetivo para a solução de problemas físicos (BREEN *et al.*, 2019; ITEN *et al.*, 2020) O próximo capítulo trará os conceitos utilizados de aprendizado de máquina e inteligência artificial, além de uma revisão acerca deste tópico.

2.2.2 Aproximação de Born-Oppenheimer

Uma interessante aproximação utilizada pela dinâmica molecular é a chamada Aproximação de Born-Oppenheimer (ABO) (BORN; OPPENHEIMER, 1927). Seu conceito baseia-se no fato de que o núcleo possui grande massa e baixa velocidade, sofrendo pouca ação do elétron, i.e., o elétron, da perspectiva do núcleo, é uma nuvem carregada. Por outro lado, o elétron, que tem pequena massa e grande velocidade, percebe o núcleo como se fosse um objeto parado, e qualquer movimento do núcleo é rapidamente acompanhado da adaptação da nuvem eletrônica (SCHLICK, 2010).

A aplicação da ABO na modelagem molecular realiza-se ao tomar o núcleo como fixo na escala de tempo de vibração do elétron. Assim, para uma dada molécula, calcula-se a energia eletrônica com base nas posições dos núcleos, criando as superfícies de energia de Born-Oppenheimer. Tendo a energia da nuvem eletrônica, pode-se então definir o movimento para os núcleos, por meio de aproximações quânticas (SCHLICK, 2010).

Essa aproximação é amplamente utilizada nos métodos de aproximação a sistemas de N-corpos, especialmente em modelos semiclássicos de dinâmica molecular, no qual núcleos têm integração clássica, e o estado dos elétrons provém de cálculos quânticos (SCHLICK, 2010; GRIEBEL; KNAPEK; ZUMBUSCH, 2007). Este princípio será considerado na modelagem do comportamento do elétron neste trabalho.

3 INTELIGÊNCIA ARTIFICIAL

A inteligência artificial (IA) é uma área do conhecimento que busca obter um equivalente computacional à inteligência humana. Um de seus subcampos é o aprendizado de máquina (ML, do inglês *Machine Learning*), uma área do conhecimento voltada ao desenvolvimento de tecnologia capaz de aprender por si própria. Este capítulo será voltado a apresentar os principais conceitos de IA e ML abordados neste trabalho, assim como um levantamento da literatura relevante.

3.1 REDES NEURAIS ARTIFICIAIS

Redes Neurais Artificiais (ANN, do inglês *Artificial Neural Network*) podem ser brevemente definidas como estruturas de processamento paralelo com capacidade de extração e armazenamento de informação, viável para seu uso posterior. Essas estruturas são inspiradas no sistema nervoso e em seus neurônios, absorvendo parte de sua nomenclatura. Sua estrutura é composta por neurônios (cada qual correspondendo a um núcleo de processamento) interligados por conexões com pesos sinápticos que fornecem a capacidade de aprendizado. O aprendizado da rede se dá por meio de algoritmos de aprendizado que permitem a otimização dos pesos sinápticos para a obtenção do melhor aprendizado durante o treinamento (HAYKIN, 2008a).

A estrutura mais básica de uma rede neural é o neurônio. O modelo de neurônio mais simples é apresentado na Figura 3.1, denominado de *perceptron* de Rosenblatt (ROSENBLATT, 1957). Ele é composto por m entradas (*inputs*), cada qual possuindo um peso sináptico w_k em cada neurônio k de dada camada. Para cada neurônio k é realizada uma combinação linear dos inputs x no formato

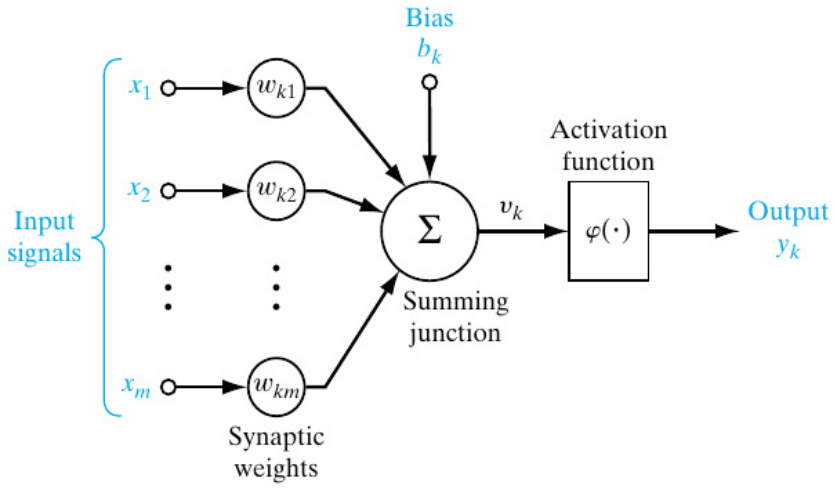
$$v_k = b_k + \sum_{i=1}^{m} x_i w_{ki}, \tag{3.1}$$

onde b_k é denominado de *bias*. Após realizar a combinação linear dos dados de entrada, o resultado v_k passa pela função de ativação $\varphi_k(v_k)$, que transforma o resultado por meio de uma função, fornecendo então a saída, que corresponde ao valor do neurônio (HAYKIN, 2008a).

Os pesos sinápticos e bias são otimizados durante o processo de treinamento da rede. A função de ativação é uma função customizável que possui o papel de limitar a amplitude da saída do neurônio. Existem várias funções de ativação, lineares e não lineares. Cada tipo de função de ativação é útil para um tipo diferente de problema. Essas são geralmente funções simples e monotônicas[1].

[1]Funções monotônicas ou monótonas $\varphi(x)$ são aquelas que, conforme cresce o valor de x, podem ser crescentes ($\varphi(x_1) \leq \varphi(x_2)$) ou decrescentes ($\varphi(x_1) \geq \varphi(x_2)$) em um dado intervalo ($x_1 < x_2$), mas nunca ambos. Ou seja, $\varphi(x)$ não oscila nesse intervalo, como ocorre com uma função periódica (CLAPHAM, 2009).

FIGURA 3.1: ESTRUTURA DE UM NEURÔNIO PERCEPTRON.



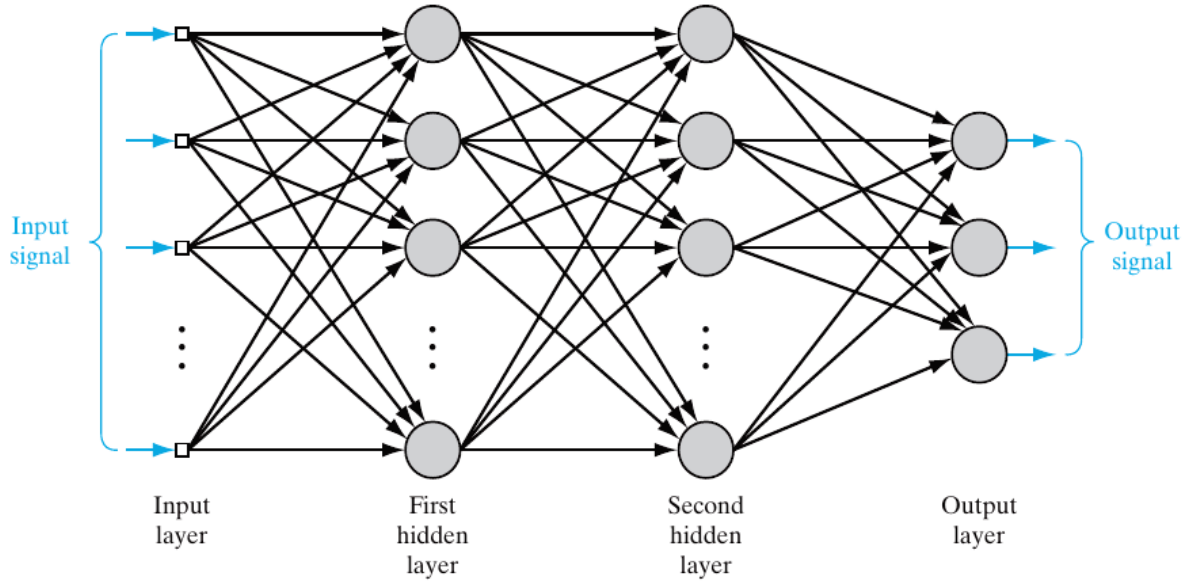
FONTE: Haykin (2008a, p.11)
LEGENDA: Trata-se de um modelo linear de um neurônio k. O exemplo possui m *inputs* x, cada qual com um peso sináptico w e mais um bias b que participa da combinação linear. O resultado da combinação linear, dado pela equação 3.1, passa por uma função de ativação $\varphi(\cdot)$ que resulta no valor final do neurônio

Entenda que cada um dos m inputs x é um atributo (informação) distinto. O mesmo processo de transformação – os mesmos pesos, função de ativação, bias – são aplicáveis para todas as instâncias (amostras) de um conjunto. O mesmo princípio é válido para as estruturas de rede.

Existem inúmeras arquiteturas de redes neurais, cada qual possui variações na estrutura de ligações sinápticas, nas funções internas, nas funções de treinamento, etc. A arquitetura mais simples é a *Multilayer Perceptron* (MLP), que consiste na união de vários *perceptrons* em camadas, como na Figura 3.2 (HAYKIN, 2008a). A conexão dos neurônios de uma MLP é limitada aos neurônios da camada anterior (que são combinados linearmente e transformados pela função de ativação) e com os neurônios da camada seguinte (cujo valor será combinado com o valor dos neurônios da mesma camada no mesmo processo de transformação). Neurônios da mesma camada não se ligam entre si na MLP. O sentido das setas na Figura 3.2 indicam o sentido de fluxo da informação utilizado, denominado *feedfoward*. Alguns tipos de redes e algoritmos de treinamento realizam o caminho reverso, denominado *backward*.

A MLP possui uma camada de *input* (entrada), uma de *output* (saída) e ao menos uma camada oculta (*hidden*). Cada conexão entre dois neurônios possui um peso e um valor de bias únicos (HAYKIN, 2008a). Tomando a estrutura do *perceptron*, as entradas deste neurônio são as saídas dos neurônios da camada anterior, exceto a primeira camada que tem como entrada os dados fornecidos pelo usuário.

FIGURA 3.2: ESTRUTURA DA REDE *MULTILAYER PERCEPTRON* (MLP).



FONTE: Haykin (2008a, p.124).
LEGENDA: Cada círculo em cinza corresponde a um perceptron, como o apresentado na Figura 3.1. A MLP consiste na combinação de uma série de transformações das entradas (camada de *input*) por neurônios nas camadas intermediárias (camadas ocultas ou *hidden*) acarretando em um certo número de saídas (camada de *output*).

Pelo fato de cada neurônio possuir uma função de ativação, esta pode ser customizável, isto é, cada neurônio da rede pode possuir uma função diferente da outra (HAYKIN, 2008a). É comum definir diferentes funções para cada camada, no qual todos os neurônios de uma camada possuem a mesma função de ativação.

3.1.1 Funções de Custo

A função de custo ou função de perda consiste no cálculo de uma medida relativa de erro entre a predição fornecida pela rede e o valor correto esperado, ou o custo dado por um modelo matemático referente à predição realizada pela rede. Há várias maneiras de se construir a função de custo, e esta depende do objetivo da rede que está sendo treinada.

Para a classificação a função mais simples é a "*zero-one loss*", que consiste em atribuir 0 a uma classificação correta e 1 para uma incorreta (SAMMUT; WEBB, 2017). A função de custo possui o mesmo princípio da função *fitness* dos algoritmos genéticos, como será melhor explicado na Subseção 3.2.2.

3.2 ALGORITMOS DE TREINAMENTO DE REDES NEURAIS

Os algoritmos de treinamento têm como papel ajustar os pesos de uma rede de forma a reduzir o erro entre a saída da rede e a saída esperada, isto é, garantir o aprendizado. O

algoritmo mais utilizado para redes tipo *feedfoward* é o *backpropagation* (BP) ou algoritmo de retropropagação de erros (MUNRO, 2017). Também possuem bons resultados os algoritmos evolutivos, como os algoritmos genéticos (GA) e *simulated annealing*. Via de regra, os algoritmos de treinamento são iterativos.

3.2.1 *Backpropagation*

O algoritmo de *backpropagation* (BP) é um processo iterativo de ajuste dos parâmetros de peso da rede que visa reduzir o erro de cada uma das saídas da rede utilizando um gradiente do erro baseado em derivadas parciais (MUNRO, 2017; HAYKIN, 2008a). Isto significa que a otimização é realizada por meio de gradiente descendente, como esquematizado na Figura 3.3, de maneira a otimizar os pesos com dada taxa de aprendizado a cada iteração. Este formato se enquadra a um tipo específico de treinamento denominado supervisionado, que será melhor abordado na Subseção 3.3, o qual depende de conhecimento prévio das saídas esperadas. Portanto o BP não é aplicável aos aprendizados não supervisionado e por reforço. O BP é vinculado ao mecanismo de combinação linear *feedfoward*, sendo específico para esse tipo de arquitetura, ganhando seu nome devido ao processo de ajuste de pesos ocorrer em sentido inverso, *backward*.

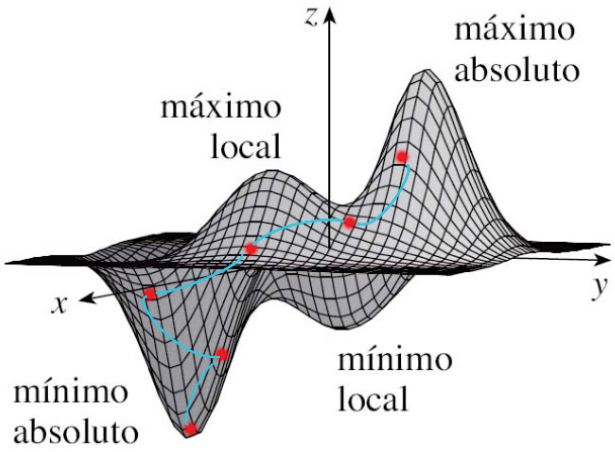
O algoritmo de Levenberg-Marquardt (LM) foi desenvolvido para resolver problemas de mínimos quadrados não-lineares por meio da minimização da soma dos quadrados por gradiente descendente (MARQUARDT, 1963). Ele foi posteriormente incorporado ao BP (HAGAN; MENHAJ, 1994), o tornando mais rápido para otimizar redes pequenas e medianas (BEALE; HAGAN; DEMUTH, 2010).

3.2.2 Algoritmos Genéticos

Os algoritmos genéticos (GA, do inglês *genetic algorithm*) é um algoritmo de busca inspirado na seleção natural Darwiniana (GOLDBERG, 1989), pertencendo a uma classe de algoritmos denominados evolutivos. O GA, sob um olhar biológico, é baseado na sobrevivência de indivíduos ao longo de gerações sob a ação de seleção e randomicidade. E sob um olhar computacional, realiza otimização por busca[2], visando obter o mínimo ou máximo global de uma função. Este algoritmo de otimização pode ser utilizado como função de treinamento para redes neurais. Apesar do BP ser o algoritmo mais utilizado, não existe uma padronização ou algoritmo mais eficiente para todos os problemas – algoritmo generalista. Ainda, como afirma Ding *et al.* (2013), a combinação ANN (rede neural artificial) e GA não apenas torna as redes neurais mais capazes de aprender e evoluir – uma inteligência mais completa – mas também resolve problemas de design (construção) da rede, melhorando sua performance. A combinação GA+ANN se mostrou especialmente eficiente para redes complexas e de larga escala.

[2]Observe que o conceito de busca da inteligência artificial difere do conceito de busca usado pelo GA. Para o GA busca envolve o uso de uma população enquanto que no conceito mais geral a busca é realizada por apenas um "ponto".

FIGURA 3.3: DESCIDA NO GRADIENTE.



FONTE: Modificado de Stewart (2013).
LEGENDA: Cada ponto representa um passo do algoritmo, e a sua convergência para o mínimo. Para uma função, existem uma série de combinações de parâmetros que fornecem pontos de máximos e mínimos. O objetivo de uma otimização é buscar o ponto mínimo ou máximo global. Geralmente os algoritmos são capazes de encontrar mínimos/máximos locais, e não os globais. As soluções locais podem ser consideradas satisfatórias para diversos problemas.

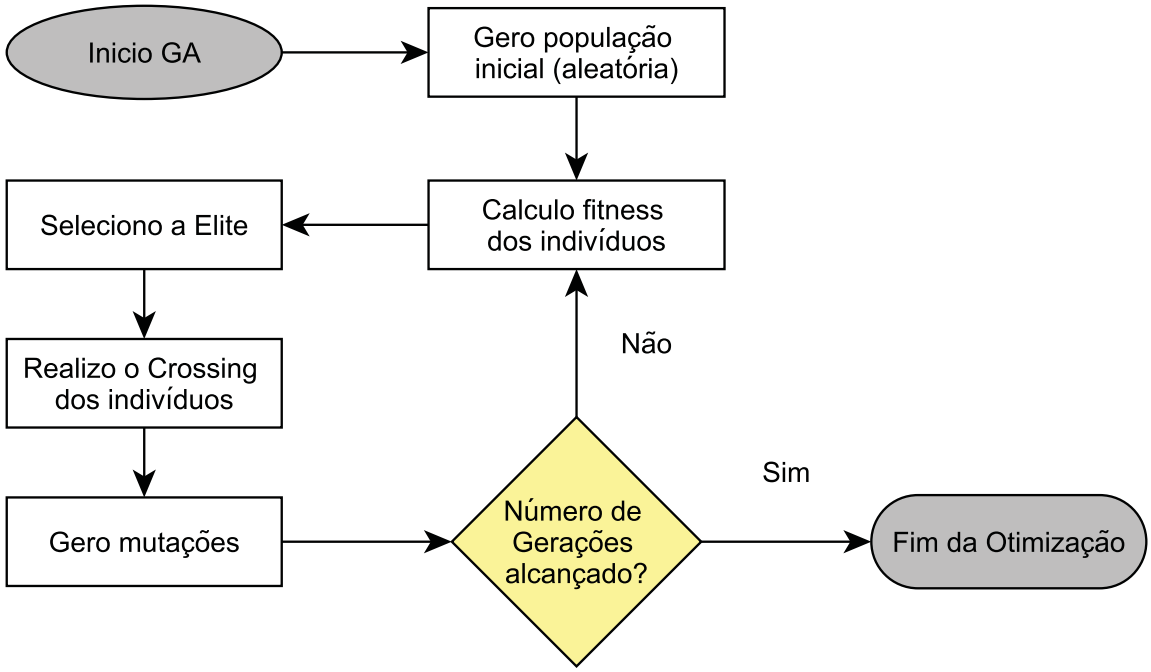
Os GA são modelos bastante flexíveis, de fácil compreensão e implementação. Um algoritmo genético simples é composto por uma população binária, que mantém seu tamanho constante ao longo das gerações, e que sofre a ação de 3 operações de reprodução bioinspiradas: *crossing*, mutação e elitismo (GOLDBERG, 1989).

Como apresentado no fluxograma da Figura 3.4, o primeiro passo em um algoritmo genético é a geração da *população inicial* aleatória. Cada *indivíduo* da população corresponde a uma solução para um dado problema. Para cada indivíduo calcula-se o *fitness* de cada solução dada, aplicando-o ao problema e verificando o resultado para cada solução proposta. Tal *fitness* pode ser uma distância, um erro ou um custo dependendo do tipo do problema. Então encontra-se os indivíduos com melhor *fitness*, os quais irão constituir a *elite* e serão parte da geração seguinte sem sofrer modificações. Todos os indivíduos sofrem cruzamento denominado *crossing over*, processo no qual os indivíduos são sorteados aos pares para se combinar e constituir a nova geração de *filhos*. Um percentual desses filhos irão sofrer *mutação* aleatória. Por fim inicia-se um novo ciclo onde os filhos passam a ser os parentais e sofrem as mesmas operações até completar o *número de gerações* especificado (GOLDBERG, 1989).

O tamanho da população, número de gerações, número de indivíduos da elite e taxa de mutação são parâmetros ajustáveis. E a função de *fitness* (ou função objetivo) é a função que corresponde ao problema que se busca solucionar.

O elitismo permite que a próxima geração tenha os melhores indivíduos, mantendo o melhor *fitness* caso a geração seguinte seja pior que a anterior. O *crossing over* e a mutação

FIGURA 3.4: FLUXOGRAMA DE UM ALGORITMO GENÉTICO SIMPLES.



FONTE: O autor (2020).
LEGENDA: As principais operações de um GA – mutação, *crossing* e elitismo – apresentadas ocorrem iterativamente sobre a população até que o número de gerações seja alcançado.

permitem explorar o espaço da função, fugindo dos mínimos/máximos locais, e permitindo encontrar o mínimo/máximo global.

O modelo original de algoritmo genético utiliza binários. A mutação consiste em alterar um bit do byte em dado percentual da população. O *crossing over* consiste em sortear (para cada par de parentais) um ponto de cruzamento, como no esquema abaixo:

	bits	byte
Parental1:	1 0 0 0 1 1 | 0 1	= 141
Parental2:	0 1 1 0 1 0 | 0 0	= 104
Filho1 (1|2):	1 0 0 0 1 1 | 0 0	= 140
Filho2 (2|1):	0 1 1 0 1 0 | 0 1	= 105

Em suma, a formação da próxima geração é realizada através de três operações: (1) o *crossing over* é a combinação dos parentais para a formação dos filhos, permitindo explorar o espaço da função; (2) a *mutação* realiza mudanças pequenas e aleatórias num percentual dos indivíduos, aumentando a diversidade; e (3) o *elitismo* seleciona os melhores indivíduos parentais para fazerem parte da geração filha, garantindo o mantenimento do *fitness* ao longo das gerações.

O processo de otimização dos algoritmos evolutivos, como o GA, tem papel similar ao BP aos realizar uma busca pelo mínimo global. Da mesma forma que o BP e outros algoritmos

de otimização, o GA tende a encontrar mínimos locais, mas diferentemente da BP, a GA não realiza uma descida no gradiente. A vantagem dos algoritmos evolutivos sobre outros tipos de algoritmos está no fato de seu processo de busca envolver uma população de soluções e possuir certo nível de aleatoriedade na busca. Essas características permitem explorar a superfície da função, permitindo a fuga de mínimos locais e aumentando a chance de encontrar o mínimo global.

3.3 APRENDIZADO DE MÁQUINA

Há basicamente três tipos de aprendizado de máquina: o supervisionado, o não supervisionado e o por reforço. O *aprendizado supervisionado* consiste num processo de aprendizado de máquina no qual tanto *input* quanto *output* são conhecidos e fornecidos para o aprendizado. Dois exemplos desse tipo de aprendizado são a classificação e a regressão, onde em ambos casos a supervisão é feita sobre a diferença entre o dado real (rótulo) e a previsão do modelo. O *aprendizado não supervisionado* se dá quando não é fornecido um referencial ou não há nenhum *feedback*, fazendo com que o aprendizado ocorra apenas possuindo em mãos os dados de *input*, o que é um desafio (SAMMUT; WEBB, 2017). Um exemplo deste é a clusterização, que busca separar os dados em grupos por similaridade. Há ainda o aprendizado *semi-supervisionado*, realizado quando há dados disponíveis, mas a disponibilidade de *outputs* correspondentes é limitada (BUTLER *et al.*, 2018).

O *aprendizado por reforço* (RL, do inglês *reinforcement learning*), por sua vez, não possui valor de saída esperado para cada input como supervisão, contudo também não ocorre às cegas como o aprendizado não supervisionado. O reforço se refere a um direcionador, uma recompensa para uma previsão boa (KAELBLING; LITTMAN; MOORE, 1996; STONE, 2017). Como menciona Kaelbling, Littman e Moore (1996), no aprendizado por reforço não é necessário fornecer dados rotulados, e as "ações subótimas" não precisam ser corrigidas, pois isso permite explorar o que se sabe (conhecimento atual) e o que não se sabe (território desconhecido). Esse modelo se adequa ao problema proposto neste trabalho, ao passo que se conhece muito bem certos aspectos que regem a interação das partículas, contudo ainda há muito "território desconhecido" acerca da sequência temporal do elétron.

3.4 REDES NEURAIS PARA OTIMIZAÇÃO

Os tipos de redes neurais aplicadas ao aprendizado supervisionado são várias e obtêm resultados satisfatórios nos mais diversos problemas e aplicações. Contudo, como será melhor detalhado na Seção 4.1, não conhecemos a trajetória do elétron ponto a ponto, portanto, não podemos usar modelos supervisionados para tal fim.

A implementação de redes para o aprendizado não supervisionado é um desafio, havendo uma lista limitada de modelos que obtiveram sucesso, na qual se destacam: Hopfield, Mapas

Auto-organizáveis (SOM, do inglês *Self-organizing map*), Redes Generativas Adversárias (GAN, do inglês *Generative adversarial networks*), *Autoencoders*, *Deep Belief Nets*.

Na literatura, o uso de redes para resolver problemas de otimização é escasso (BELLO *et al.*, 2016), sendo a maioria dos trabalhos voltados à chamada otimização combinatória [3]

As redes de otimização realizam um processo de "otimização externa", além da otimização interna de seus pesos. A otimização interna dos pesos é característica de todas as redes, uma vez que precisam ajustar seus pesos ao problema fornecido. A otimização externa é aquela que a rede busca fornecer uma saída que corresponda a solução do problema. Esta nomenclatura será utilizada neste trabalho. Os principais tipos de redes dessa classe são a Hopfield, Mapas Auto-organizáveis (SOM) e Redes Generativas Adversárias (GAN).

Hopfield

Foi a primeira rede a ser utilizada para resolver problemas de otimização, já nascendo com este intuito (HOPFIELD; TANK, 1985; TANK; HOPFIELD, 1986). É uma rede binária totalmente recorrente, isto é, todos os neurônios da rede são interligados entre si com fluxo da informação possível em ambos sentidos. Entretanto, pouco tempo depois de sua publicação, Wilson e Pawley (1988) publicam uma crítica ao modelo de Hopfield, apontando falhas em seu uso para otimização, afirmando que o modelo era muito susceptível ao uso de hiperparâmetros[4].

Mapas Auto-organizáveis

O modelo surgiu logo após o Hopfield. Atualmente, a maioria dos problemas de otimização envolvendo rede são resolvidos por SOM, dentre as quais a rede Kohonen (KOHONEN, 1982). Os mapas possuem aprendizagem competitiva e não supervisionada, sendo também utilizadas como método de redução de dimensionalidade e de clusterização (HAYKIN, 2008b; KASKI, 2017). As SOM ainda não possuem resultados muito satisfatórios para otimização (HUANG *et al.*, 2019), mas também são apontadas como promissoras, especialmente em combinação com outras técnicas (FAIGL, 2018).

As Redes Generativas Adversárias

As Redes Generativas Adversárias ou GAN vem sendo cada vez mais exploradas. A GAN visa minimizar a divergência entre a distribuição dos dados reais e a distribuição de dados gerados por uma rede, o que é feito otimizando duas redes simultaneamente.

[3] Um problema de otimização combinatória consiste basicamente na otimização de uma função de restrição (limitação dos estados permitidos) aplicada sobre um sistema de elementos finitos. Como resultado, se busca uma solução de menor custo dentre um conjunto também finito, mas muito grande, de soluções possíveis (DORIGO; BIRATTARI, 2017). Como exemplificação de otimização combinatória está o problema do caixeiro viajante que consiste num conjunto de cidades a serem visitadas, e o objetivo está em encontrar a "menor rota para percorrer uma série de cidades (visitando uma única vez cada), e retornando à cidade de origem" (FAIGL, 2018).

[4] Hiperparâmetro: diferente dos parâmetros internos que são otimizados durante o aprendizado, os hiperparâmetros são valores definidos antes do início do processo de aprendizado/treinamento pelo usuário (CLAESEN; MOOR, 2015). São considerados hiperparâmetros o número de iterações, tamanho inicial da população, taxa de mutação, tamanho da elite, número de nós por camada da rede, função de ativação, etc.

A ideia central das redes GAN está num sistema de duas redes que competem entre si: a *generativa* tem a função de "imitar" certo conjunto de dados, replicando-o; e a *discriminativa* avalia a saída da primeira rede, tentando diferenciar se os dados são originais ou falsos (GOODFELLOW *et al.*, 2014). Uma analogia que pode-se fazer é o de que as duas redes são otimizadas conjuntamente como numa espécie de evolução convergente, onde parasita e hospedeiro competem, o parasita tentando enganar o hospedeiro, e o hospedeiro tentando identificar o parasita. A GAN baseia-se na geração sintética de dados, utilizando a distribuição conjunta de probabilidade[5] (SAMMUT; WEBB, 2017). A maioria das GAN se enquadram no denominado aprendizado semi-supervisionado, já que a rede discriminativa exige um parâmetro para comparação (GOODFELLOW *et al.*, 2014), contudo novos trabalhos como o de Springenberg (2016) e Yoon, Jarrett e Schaar (2019) vêm criando modelos GAN para aprendizado não supervisionado e inclusive voltado a séries temporais, respectivamente.

Existem trabalhos, como o de Sanchez-Lengeling *et al.* (2017) que utilizam a combinação de métodos como GAN e RL. Neste artigo o sistema foi construído para a modelagem de estruturas de moléculas com certas propriedades desejadas. Para tal fim, como se trata de uma modelagem tridimensional com infinitas possibilidades, os autores utilizaram a codificação SMILES[6] para transformar tal problema em um problema de otimização combinatória.

Outras combinações que vêm se tornando mais populares é a associação dos métodos GAN, RL, etc com a *deep learning*. Podemos conceituar a *deep learning*, ou redes profundas, simplesmente como "redes neurais grandes", capazes de aprender padrões complexos devido à sua grande estrutura e ao seu grande número de pesos otimizáveis. Mas ainda assim observa-se uma grande dificuldade nos métodos em escapar da otimização combinatória.

3.4.1 Aprendizado por Reforço

O Aprendizado por Reforço (RL, do inglês *Reinforcement Learning*) não é apenas uma classe de aprendizado (como apresentado na Seção 3.3), mas também um conjunto de métodos que possuem 4 elementos em comum (HAYKIN, 2008a; SUTTON; BARTO, 2018):

recompensa define as ações como boas ou ruins, e assim fornece um recompensa ou punição pela ação.

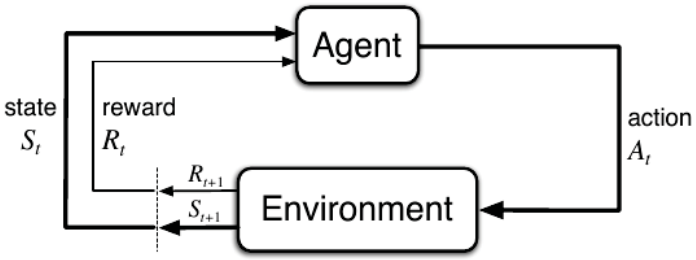
função valor determina o que é bom ou ruim ao longo do tempo, ou seja, a recompensa total acumulada – equivalente a uma função de custo.

política trata-se do conjunto de regras ou ainda as associações estímulo-resposta, definindo as ações que o agente de aprendizagem pode tomar. Como podem haver vários estados

[5] A distribuição conjunta, um sinônimo para a distribuição multivariada, é a distribuição de probabilidades simultânea de um conjunto de variáveis aleatórias (EVERITT; SKRONDAL, 2002)

[6] O método *simplified molecular-input line-entry system* (SMILES) utiliza uma notação linear para representar estruturas químicas variadas.

FIGURA 3.5: PROCESSO ITERATIVO NO APRENDIZADO POR REFORÇO.



FONTE: Sutton e Barto (2018).
LEGENDA: O esquema apesenta a interação entre o agente e o ambiente num processo de decisão de Markov. O agente que aprende realiza uma ação sobre um ambiente. O ambiente muda com esta ação, alterando o estado do sistema, e retorna uma recompensa/punição pela ação, modulando a ação do agente.

possíveis, a política trata das ações possíveis em determinado estado. A política ótima é aquela com maior recompensa, ou menor punição.

modelo do ambiente é opcional. Refere-se a como o ambiente responde a uma ação do agente.

O esquema da Figura 3.5 torna mais claro o mecanismo de aprendizado. O *input* é o estado atual do sistema. O agente (quem aprende) toma uma ação sobre o ambiente. Esta ação é avaliada, fornecendo uma recompensa ou penalidade. O ambiente é alterado, e fornece o novo estado do sistema. O objetivo do agente é obter o máximo de recompensa (SUTTON; BARTO, 2018). Podemos interpretar a *recompensa* como uma "avaliação local" ou ponto a ponto, e a *função valor* como uma "avaliação global" ou de contexto[7].

Os métodos de aprendizagem por reforço são guiadas por tentativa e erro, não exigindo a presença de um "professor" para ensinar quais ações tomar (sem supervisão). De outra maneira, o RL pode ser explicado como o aprendizado pela experiência no qual o agente adquire uma *política de ações* (política aprendida) que maximiza a sua recompensa (otimização), ou seja, por RL se aprende *comportamento* (HAYKIN, 2008a).

A ação se dá sobre um sistema dinâmico, levando a mudanças no ambiente conforme a ação do agente, isto é, uma ação pode alterar as condições para a próxima ação. Para ficar mais claro, tomemos o exemplo do jogo da velha. A ação se encontra entre as jogadas possíveis – conforme vai avançando o jogo, o número de possíveis jogadas vai diminuindo pois vão diminuindo o número de casas vazias – e a política é a sequência de ações tomadas. Uma ação/jogada altera o ambiente e a possibilidade de ações para a próxima jogada: se faço uma boa jogada, isso evita que meu adversário vença e assim eu tenho a oportunidade de uma nova jogada, enquanto que se faço uma jogada ruim, posso dar a vitória para meu oponente e não ter a chance

[7] Apesar de estar utilizando os termos local e global para descrever os custos dos métodos de RL, essa terminologia é nossa, e provavelmente não será encontrada na literatura.

de uma nova jogada, ou ainda ter opções mais limitadas para as próximas jogadas (SUTTON; BARTO, 2018).

Formalmente, a definição de estado é dada por uma cadeia de decisão de Markov. De maneira geral, o aprendizado por reforço se assemelha muito ao Processo de Decisão de Markov (MDP, do inglês *Markov Decision Process*). Uma MDP consiste num processo de controle estocástico de tempo discreto, isto é, uma modelagem para tomada de decisões baseada num conjunto de estados e ações discretas associadas a probabilidades. A transição entre os estados é delimitada por ações discretas possíveis cada qual com uma probabilidade associada, e a tarefa do agente é escolher a ação sobre seu ambiente (HOWARD, 1960; LITTMAN, 1994). O conceito de MDP existe desde a década de 50 (BELLMAN, 1957), consistindo de uma extensão das Cadeias de Markov melhor explicadas na Subseção 3.5.3.

Há ainda dois tipos de RL: os *model-based* e os *model-free*. Os denominados *model-free* não usam a probabilidade de transição do MDP. Esse modelo não é baseado em Markov, e por isso também não utiliza a avaliação local (recompensa), apenas a global (função valor). Este tipo de algoritmo é explicitamente conduzido por tentativa e erro já que não precisa de um modelo de ambiente (SUTTON; BARTO, 2018). Uma vantagem desse tipo de RL está no fato de não precisar conhecer as probabilidades de transição entre os estados.Um exemplo de *model-free* é o chamado *Q-learning*.

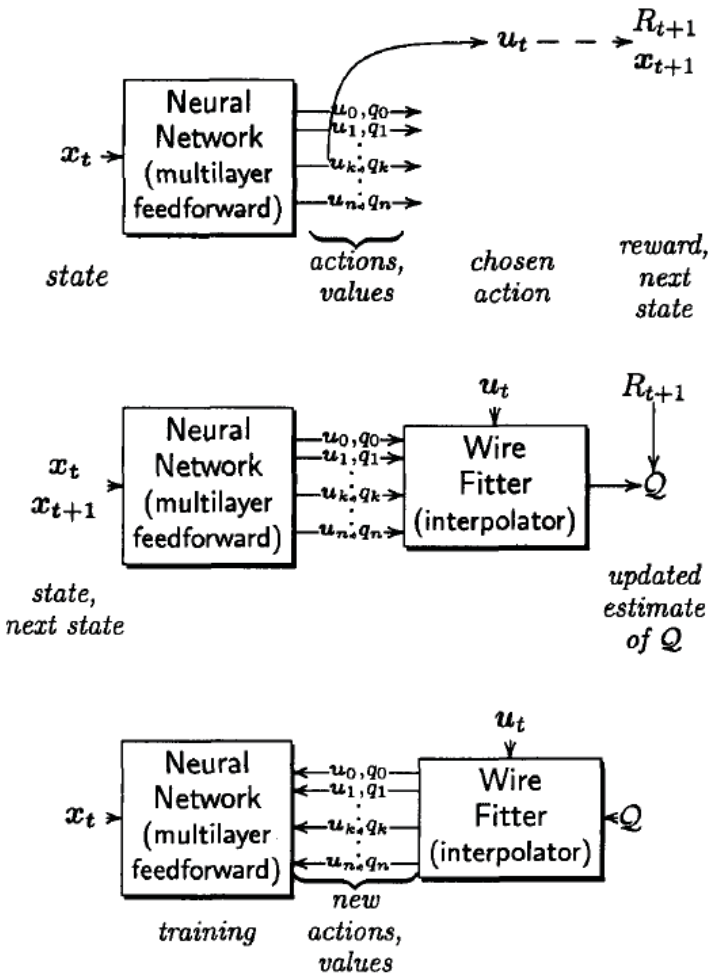
Algoritmo Q-learning

O algoritmo Q-learning é um dos métodos de RL mais populares. Ele é voltado para resolver problemas onde a probabilidade de transição nas cadeias de Markov não é conhecida. O objetivo se mantém, encontrar uma política ótima, de custo mínimo, utilizando-se de tentativa e erro. O custo mínimo é encontrado após a observação de várias ações, determinando o custo para cada uma (HAYKIN, 2008a).

O modelo é usado basicamente para sistemas de estados e ações discretos. Possui aplicações principalmente em sistemas discretos finitos, contudo há muitos estudos acerca da aplicação do Q-learning em estados e ações contínuos. Para tornar mais claro, tomemos um exemplo referente à automação do movimento robótico. Geralmente há apenas quatro possibilidades de movimento discretos – frente, trás, direita, esquerda – enquanto que uma saída contínua permite um movimento mais fluido (CHEN *et al.*, 2019). Isso é essencial para se determinar as posições do elétron, por exemplo.

Para permitir estados e ações contínuas (não discretas) em aplicações com RL, é necessário o uso de técnicas acessórias como redes neurais (DOYA, 2000; HEINEN; ENGEL, 2010). O trabalho de Gaskett, Wettergreen e Zelinsky (1999) é um dos pioneiros no uso do RL em aprendizado de estados e ações contínuos (não discretos). Os autores elencam as capacidades essenciais para que um sistema *Q-learning* consiga lidar com estados e ações contínuos: (1) Seleção de ação – maior valor esperado, (2) Avaliação de estado – encontrar valor para atualização

FIGURA 3.6: ALGORITMO DE TREINAMENTO DE REDE NEURAL *WIRE-FITTED*.



FONTE: Gaskett, Wettergreen e Zelinsky (1999).
LEGENDA: *Quadro superior* – para uma entrada x_t são fornecidas em tempo real uma população de ações, correspondentes a uma população de redes iniciais, cada qual com um custo diferente. É escolhida a ação de melhor custo e se obtém um x_{t+1} como saída, e obtém-se a recompensa. *Quadro central* – forneço x_{t+1} como entrada e obtenho o valor de Q como a saída esperada, considerando a entrada fornecida, a ação tomada, a recompensa por esta ação e uma taxa de aprendizado, ou seja, é o valor que permitirá a correção dos pesos. *Quadro inferior* – a partir do valor de Q, calcula-se os ajustes de pesos em sentido backward utilizando interpolação. Repetem-se os passos.

usando o estado de maior valor, (3) *model-free* – requer ausência de modelo do sistema dinâmico para ser aprendido, (4) política flexível – permite a representação de um amplo range de políticas, (5) continuidade – variação suave entre os estados , (6) – generalização do estado – reduz a necessidade de exploração requerida no espaço de estados , (7) generalização de ação – reduz a exploração no espaço de ação.

Após esse levantamento inicial, Gaskett, Wettergreen e Zelinsky (1999) propõem alguns modelos com *Q-learning* dentre os quais um modelo de rede neural treinada com RL. O esquema

da Figura 3.6 apresenta o funcionamento do treinamento da rede que envolve o uso de interpolação para o ajuste dos pesos, e uma função de custo que permite delimitar a recompensa pelas ações tomadas.

O uso de redes RL tem grande aplicação para jogos (LITTMAN, 1994; SILVER *et al.*, 2016; ELFWING; UCHIBE; DOYA, 2018) e na robótica (DEISENROTH *et al.*, 2013; LI *et al.*, 2018), mas também em problemas de otimização (BELLO *et al.*, 2016), na computação quântica (POROTTI *et al.*, 2019), e em várias outras áreas. Os jogos são uma boa maneira de se realizar testes de algoritmos novos e comparações de eficiência (ELFWING; UCHIBE; DOYA, 2018). A maioria desses trabalhos com RL *model-based* utiliza ações discretas.

No recente trabalho de Sehgal *et al.* (2019), foi demonstrado que algoritmos genéticos são eficientes na otimização do ajuste de parâmetros no RL – aplicação na robótica. O algoritmo utilizado foi uma combinação *Deep Deterministic Policy Gradient* (DDPG) e *Hindsight Experience Replay* (HER). O método DDPG é do tipo "*actor–critic*", que consiste em duas redes neurais iteragentes no sistema similar ao GAN anteriormente apresentado. Enquanto que HER é um modelo de aprendizado que tenta imitar o comportamento humano aprendendo com as falhas cometidas obtendo recompensas esparsas (recompensa global).

3.5 SISTEMAS DINÂMICOS

Sistemas dinâmicos envolvem problemas que surgem nas mais diversas áreas, como na física, química e biologia. Um sistema dinâmico é composto por duas partes: o espaço de fase (as possíveis configurações do sistema físico) e a dinâmica (descreve como o estado do sistema muda no tempo). A evolução temporal de variados sistemas das mais diversas áreas de estudo – física, química, biologia – pode ter um comportamento caótico. Isto significa que do ponto de vista determinístico, a evolução temporal do sistema pode ser considerado imprevisível. Certos problemas, que parecem muito complicados, podem ser simplificados ao se concentrar nos comportamentos a longo prazo, observando a repetição de um padrão (RUELLE, 1989; ROBINSON, 2006).

3.5.1 Atratores

O conceito de atratores participa da compreensão dos fenômenos dinâmicos em diferentes matérias, adentrando no domínio de estudo da teoria do caos. A sua definição é bastante variável, e Milnor (1985) discute as várias definições de atratores. Contudo, pode-se afirmar que há o consenso de que um atrator é (1) o sistema que atrai um grande conjunto de condições iniciais para uma região aberta do conjunto denominada "bacia de atração", e (2) que atende a propriedades de minimalidade (MILNOR, 1985; ROBINSON, 2006).

Definição: Dado um conjunto B de pontos é denominado de *atrator* de um *sistema dinâmico* $A(t+1) = f(A(t))$, se existir um número $\delta > 0$ tal que se $|A(0) - b| < \delta$ para algum ponto $b \in B$, então

$$\lim_{t \to \infty} |A(t) - B| = 0 \tag{3.2}$$

onde t é o passo da iteração de uma função f que age sobre o sistema dinâmico A (Sandefur, 1991 *apud* Neto, 2004). De outra maneira, podemos afirmar que

$$\lim_{t \to \infty} A(t) = B. \tag{3.3}$$

Ao longo do tempo, após várias iterações, os pontos são atraídos para a bacia de atração. Um sistema dinâmico pode evoluir para três possíveis níveis de complexidade da bacia de atração (RUELLE, 1989; ROBINSON, 2006):

1. atrator fixo ou de ponto único: tendência a um ponto único no espaço. Exemplo: pêndulo amortecido por arrasto.

2. atrator periódico ou cíclico: realiza uma curva fechada no espaço com certa periodicidade. Exemplo: órbita planetária.

3. atrator estranho ou quasiperiódico: oscila entre diferentes estados de forma caótica, mas não aleatória. Exemplo: Atrator de Lorenz.

O termo "atrator estranho" cunhada por Ruelle e Takens (1971) foi utilizada para designar conjuntos de atração bastante complicados. Não há uma definição fixa para o chamado "atrator estranho", mas está fortemente relacionado com os conceitos de dinâmica caótica, turbulência e fractais. Um sistema dinâmico com atrator caótico possui instabilidade local mas é globalmente estável, obedecendo a um padrão que aparece após várias iterações. Ou seja, os pontos, a cada iteração, divergem mas depois de várias iterações voltam a se aproximar (GREBOGI; OTT; YORKE, 1987), formando por fim os padrões que conceituam o atrator.

3.5.2 Rede neurais atratoras

As funções que transformam sistemas dinâmicos em atratores citadas na Subseção 3.5.1 podem ser redes neurais. Um exemplo disso é o tratamento realizado pelas já mencionadas redes tipo Hopfield (HOPFIELD; TANK, 1985; HERTZ, 2006). Estas redes aprendem os padrões a ela apresentados estabelecendo a memória associativa, isto é, a partir de certa entrada de padrões, a rede aprende a restaurar padrões corrompidos a ela apresentados (após o treinamento), de forma a devolver o padrão mais próximo (HOPFIELD; TANK, 1985; HERTZ, 2006). Tomando o conceito de atração, uma outra maneira de se explicar o funcionamento da rede Hopfield é a sua busca por um mínimo de energia, que corresponde à bacia de atração.

Redes treinadas por aprendizado por reforço também podem originar atratores. No artigo de Hamid e Braun (2017), modelos de redes atratoras de aprendizado associativo são

apresentados como capazes de memorizar sequências de eventos durante seu treinamento obtendo a capacidade de gerar "estados estáveis auto-sustentáveis", ou seja, se estabilizar numa bacia de atração. Para tal processo é preciso um "atraso" temporal entre o ponto de partida e a obtenção do estado estável. Este atraso (*delayed reward*) também denominado de reforço esparso corresponde à nossa avaliação global. O reforço esparso é frequentemente utilizado em jogos, no qual após uma sequência de jogadas é realizada a avaliação e fornecido o custo (HAMID; BRAUN, 2017; STANLEY; MIIKKULAINEN, 2002).

Hamid e Braun (2017) afirmam que o ponto central da teoria do atrator neural está em duas características presentes em redes neurais: (1) a rede é plástica, desenvolve suas conexões seguindo a Regra de Hebb[8], (2) as associações que levam à atração são padrões persistentes auto-sustentados pela dinâmica neural. À semelhança da busca do mínimo global realizada por algoritmos evolutivos na descida do gradiente, o estado estável da bacia de atração funciona como um potencial energético, onde a região mais estável para o sistema é o de menor energia.

3.5.3 Markov Chain Monte Carlo

Markov Chain Monte Carlo, ou apenas MCMC, possui a capacidade de lidar com problemas bastante variados e complexos, com amplo uso no aprendizado de máquina e aplicações na física, estatística, economia, etc. Alguns exemplos de aplicações são o aprendizado por inferência Bayesiana, a mecânica estatística, otimização (ANDRIEU *et al.*, 2003), tópicos de interesse para o presente trabalho.

Podemos definir MCMC, de maneira simplificada, como sendo a integração de *Monte Carlo* por meio de *cadeias de Markov* (GILKS; RICHARDSON; SPIEGELHALTER, 1995). Para compreender o funcionamento do MCMC, primeiramente vamos entender o que são esses dois métodos que o compõem.

Cadeias de Markov

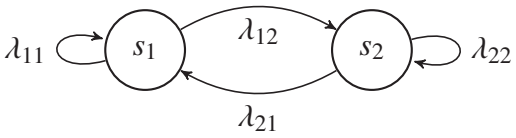
São a base para o Processo de Decisão de Markov (MDP) apresentado na Subseção 3.4.1. Podem ser descritas como um tipo especial de processo estocástico[9], onde ocorre a análise e caracterização de sequências de variáveis aleatórias com probabilidades associadas para a transição entre estados discretos. A probabilidade de transição para o estado futuro independe do estado passado, desde que o estado atual pertença ao conjunto de estados possíveis. O processo sequencial entre os estados pode levar ao equilíbrio obtendo uma distribuição invariante (um atrator) (GAMERMAN; LOPES, 2006).

Formalmente, uma cadeia de Markov de tempo discreto, dado um conjunto finito de estados S, denominado espaço-estado, há N estados $s \in S$, onde s_i o i-ésimo estado.

[8] A Regra Hebbiana afirma algo como: "Neurônios que disparam juntos permanecem conectados, e os que não disparam juntos falham em conectar-se" (GERSTNER, 2017).

[9] Processo Estocástico - referente a estatística sobre a evolução sequencial de variáveis randômicas, podendo ser uma sequência temporal (EVERITT; SKRONDAL, 2002).

FIGURA 3.7: CADEIA DE MARKOV.



FONTE: O autor (2020).
LEGENDA: s_i representa o estado i e λ_{ji} a probabilidade de transição do estado j para o i.

Associado ao conjunto, há uma medida $\lambda = (\lambda_i : s_i \in S)$ que representa uma distribuição se $\sum_{s_i \in S} \lambda_i = 1$. Dada uma variável randômica X com valor pertencente ao conjunto S, então X é função $X : \Omega \rightarrow S$. Suponha que

$$\lambda_i = \Pr(X = s_i) = \Pr(w : X(w) = s_i), \tag{3.4}$$

λ define a distribuição de X, e X assume o valor s_i com probabilidade λ_i. Ainda, a probabilidade de transição do estado s_j para s_i é dada por λ_{ji} (NORRIS, 1998), como na Figura 3.7.

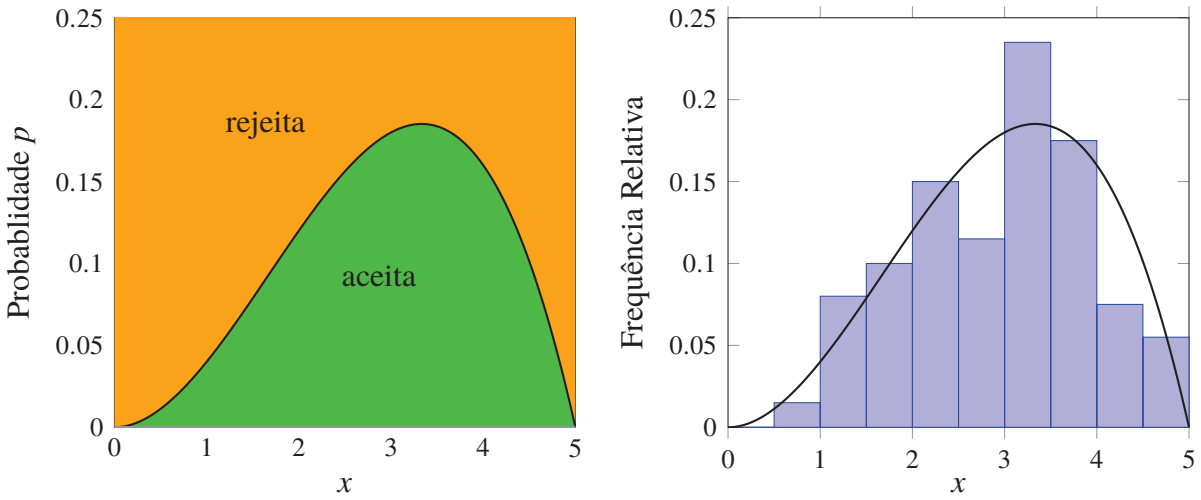
Usando a saída da cadeia de Markov é possível obter uma estimava de uma distribuição estacionária $\phi(\cdot)$. O grande objetivo é descobrir como obter esta distribuição $\phi(\cdot)$, que seja uma solução para um dado problema (GILKS; RICHARDSON; SPIEGELHALTER, 1995).

Monte Carlo

São métodos computacionais que encontram soluções para problemas matemáticos e estatísticos. São utilizados quando métodos analíticos não são capazes de solucionar o problema, ou quando estes exigem grande tempo de solução (EVERITT; SKRONDAL, 2002). Digamos que se deseja estimar um valor θ esperado a partir de uma variável aleatória x, $\theta = E(x)$. Suponha que se possa gerar variáveis aleatórias independentes (o valor de uma não interfere na geração da próxima) com a mesma distribuição de probabilidades pertencente a x. Serão geradas N variáveis aleatórias $x_1, x_2, \ldots x_N$ de tal modo que o valor médio destes será um bom estimador de θ. Pelo teorema central do limite, quanto maior o amostral, melhor será a estimativa, já que o centro da distribuição normal dos pontos x gerados se aproximarão de θ (RUBINSTEIN; KROESE, 2017).

Um dos algoritmos mais famosos e simples de Monte Carlo é o algoritmo de Metropolis–Hastings, apresentado no Algoritmo 3.1. Neste são gerados pontos $x_1, x_2, \ldots, x_n$ com distribuição equivalente à distribuição buscada; a Figura 3.8 torna este conceito mais claro. Simplificadamente, o algoritmo utiliza a probabilidade associada a cada valor de x como decisor para o aceite ou a recusa, a partir de um segundo valor randômico entre [0,1] gerado. Este algoritmo é utilizado para simular uma Cadeia de Markov com

FIGURA 3.8: ALGORITMO DE MONTE CARLO METROPOLIS-HASTINGS.



FONTE: O autor (2020).
LEGENDA: Os valores de x possuem uma probabilidade p associada. O algoritmo de Monte Carlo utiliza esta probabilidade como limite para aceitar ou rejeitar um valor de x randômico sorteado e assim compor a distribuição. Se um segundo valor randômico γ entre 0 e 1 gerado for menor que o valor p correspondente para este x, este será aceito, senão será rejeitado. Se rejeitado, o valor atual será uma cópia do anterior.

distribuição estacionária de maneira que esta distribuição coincida com uma distribuição alvo (RUBINSTEIN; KROESE, 2017).

O MCMC é um modelo bayesiano, e pode ser utilizado para gerar uma distribuição que se aproxima de outra distribuição alvo que busca-se determinar (GILKS; RICHARDSON; SPIEGELHALTER, 1995). MCMC permite a geração de um amostral ϕ enquanto explora o espaço Φ utilizando cadeia de Markov. A ideia centra-se no fato de se gastar mais tempo em regiões mais importantes e menos tempo em regiões menos prováveis. Queria dizer, há regiões de maior probabilidade em uma distribuição alvo $L(\phi)$ normalizada, onde a amostragem ocorre com maior frequência – como realizam os algoritmos de Monte Carlo. De forma iterada realiza-se a amostragem onde cada novo valor amostrado obedece a uma distribuição de probabilidades. Ao realizar novas amostras, caminha-se com passos pequenos, indo de um lado para explorar o espaço. O tamanho do passo dado depende da probabilidade do ponto atual (novo) ϕ_t e do ponto anterior ϕ_{t-1}. A Figura 3.9 esquematiza a amostragem. Cada ponto é uma amostragem realizada, as verdes são os pontos aceitos e os vermelhos os rejeitados. O objetivo é obter uma distribuição equivalente à $L(\phi)$. Como mencionado, nas cadeias de Markov a probabilidade do próximo valor na cadeia de Markov é dependente unicamente do valor atual (ANDRIEU *et al.*, 2003). Este exemplo (Fig. 3.9) é utilizado para problemas onde é necessário realizar amostragem.

Os métodos de MCMC não possuem amplo uso no estudo da dinâmica molecular, nem em áreas correlatas, como no estudo de estruturas proteicas tridimensionais, química, etc. Contudo, alguns trabalhos envolvendo dinâmica molecular clássica, como os de Faulkner *et al.* (2018) e Lei, Krauth e Maggs (2019), tiveram bons resultados.

ALGORITMO 3.1 ALGORITMO METROPOLIS-HASTINGS.

procedimento MONTE CARLO(N,P) Para dado conjunto Ω alvo, obter conjunto $Y = y_1, y_2, \ldots y_N$
    para $i = 1$ até N **faça**
        Gero x randômico entre o mínimo e o máximo da distribuição alvo
        $x = \text{rand}(min(\Omega), max(\Omega))$
        Calculo a probabilidade deste x na distribuição alvo P
        $p = P(x)$
        Gero γ randômico no intervalo $[0, 1]$
        $\gamma = \text{rand}(0, 1)$
        se $\gamma <= p$ **então** // Aceito x
            $y_i = x$
        senão se $\gamma > p$ **então** // Rejeito x, repito o passo
        fim se
    fim para
fim procedimento

Tanto Faulkner *et al.* (2018) quanto Lei, Krauth e Maggs (2019) utilizam uma variante da MCMC, a *event-chain Monte Carlo* (ECMC) para simular interações interatômicas clássicas. Faulkner *et al.* (2018) simula a ação de forças Coulombianas sobre sistemas de N-partículas interagentes. Como resultado, conseguiram obter uma aproximação bem sucedida, e termodinamicamente satisfatória, de um campo Coulombiano sem o uso do tradicional P^3M[10] e não necessitando de cálculos partícula-partícula da força Coulombiana. Enquanto isso, Lei, Krauth e Maggs (2019) compara os métodos tradicionais de MD, MCMC e ECMC a um problema de interação unidimensional de partículas com aplicação sobre o potencial de Lennard-Jones[11] por meio de um modelo harmônico. Como conclusão, obtiveram que os métodos de MCMC e ECMC são melhores que a MD em precisão, pois não sofrem com os efeitos da integração, e ainda que o ECMC supera o MCMC em velocidade para este problema.

Contudo, o método ECMC apresentado é altamente supervisionado. Ambos métodos, MCMC e ECMC, não envolvem um aprendizado propriamente dito; toda vez que se deseja um novo amostral, novas tentativas e erros são realizados (à semelhança do Monte Carlo). Esse processo leva algum tempo até atingir o amostral ótimo global exigido, dependendo da complexidade da função envolvida, já que a maioria dos métodos MCMC envolvem apenas uma movimentação local (por pequenos passos) (KIM; PARK; LEE, 2009).

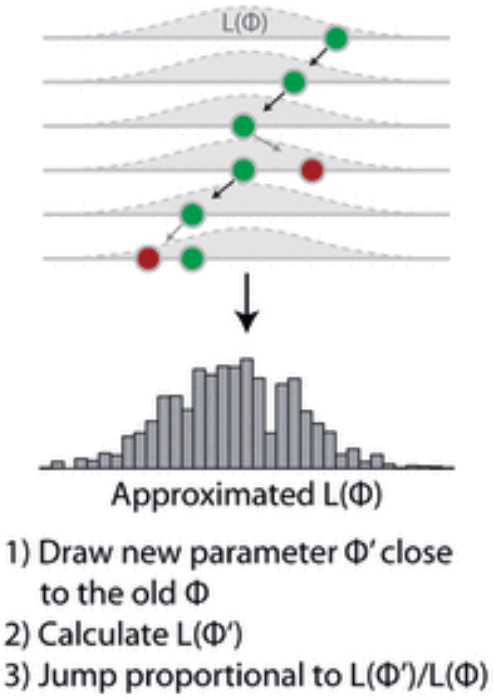
3.5.4 Adaptive Resonance Theory (ART)

Adaptive Resonance Theory (ART) é uma arquitetura de aprendizado inspirada em redes neurais e em modelos neurobiológicos. Ela, como sugere o nome, possui um processo de

[10] Particle-Particle Particle-Mesh (P^3M) é um método similar ao SPH (Smoothed-Particle Hydrodynamics), onde ocorre uma interpolação do potencial sobre um sistema de N-partículas em uma grade, sobre a qual o potencial é resolvido.

[11] O Potencial de Lennard-Jones é utilizado para representar a interação interatômica clássica de átomos de gases nobres(LENNARD-JONES, 1931).

FIGURA 3.9: ESQUEMA DO MÉTODO DA CADEIA DE MARKOV MONTE CARLO.



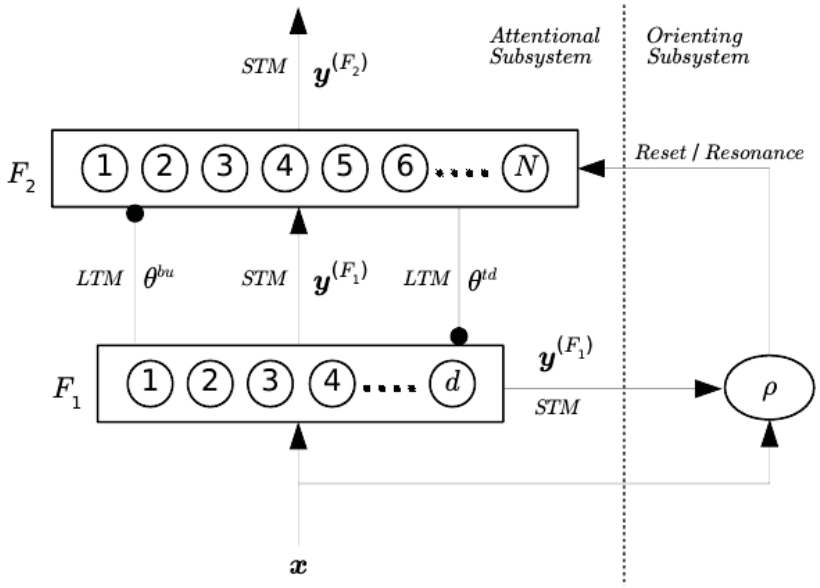
FONTE: Hartig *et al.* (2011)
LEGENDA: Visando aproximar uma distribuição $L(\phi)$, são realizadas amostragens sequenciais, indicadas pelos pontos coloridos. Os pontos em verdes são os valores aceitos e os vermelhos os rejeitados. A rejeição é definida por Monte Carlo. Caso um ponto seja rejeitado, retorna-se ao ponto anterior e realiza-se nova amostragem. A amostra seguinte (ϕ') é proporcional a um passo de tamanho $L(\phi')/L(\phi)$, na qual ϕ' é o novo passo e ϕ é o passo anterior.

ressonância em seu treinamento (CARPENTER; GROSSBERG, 2017; SILVA; ELNABARAWY; II, 2019). Existem vários tipos de rede ART, algumas das quais não iremos adentrar, como a ART1 que possui entradas exclusivamente binárias. Em termos de aprendizado, diferentes arquiteturas ART possuem diferentes tipos de aprendizado: supervisionado, não supervisionado e aprendizado por reforço; mas sua base é não supervisionada. Além disso, a arquitetura ART pode ser configurada para diferentes aplicações, cada ART passa a ter uma aplicação específica para classificação, ou regressão, etc (SILVA; ELNABARAWY; II, 2019).

Sua estrutura básica é composta por duas camadas totalmente conectadas: F1 e F2 (SILVA; ELNABARAWY; II, 2019). As fases de treinamento são descritas abaixo, com o auxílio da Figura 3.10:

1. F1, denominada camada de representação de características, é a camada de *input*. Ela funciona em dois sentidos: *feedfoward* e *feedback*. Primeiro, em sentido *feedfoward*, F1 propaga o *input* x para F2, também denominado de memória de longo prazo (LTM, θ^{bu}, *bottom-up*).

FIGURA 3.10: ARQUITETURA ELEMENTAR DE REDES ART.



FONTE:Silva, Elnabarawy e II (2019).
LEGENDA: x – input, F1 – camada de representação de características e de comparação, F2 – camada de representação de categorias, STM – memória de curto prazo, LTM – memória de longo prazo, ρ – parâmetro de vigilância, $y^{(F_1)}$ – STM saída de F1, $y^{(F_2)}$ – STM saída de F2, θ^{td} – LTM *top-down* é a expectativa, θ^{bu} – LTM *bottom-up*.

2. em F2, os nós competem pela ativação, onde o nó que tiver máximo de uma medida de similaridade vence – regra do "vencedor leva tudo".

3. em sentido *feedback*, a camada F1 recebe o valor processado pela camada F2, denominado de expectativa (θ^{td}, *top-down*) e compara com o *input* x – por isso F1 também é conhecido como camada de comparação.

4. após comparar, F1 fornece $y^{(F_1)}$, ou memória de curto prazo (STM), para F2, que serve como orientação para as mudanças.

5. em recursão, entra-se no ciclo de testes de vigilância para avaliar se os nós ativados em F2 são capazes de aprender o padrão fornecido. O valor de confiança, entre $y^{(F_1)}$ e o *input* x, é comparado com o parâmetro de vigilância ρ. Se a confiança for maior que ρ, a rede entra em estado de ressonância, senão a ressonância é inibida e os nós ativos em F2 são atualizados, retomando a fase de busca. Isto é, o mecanismo de busca e aprendizado são modulados pela ativação ou inibição da ressonância.

Há dois tipos de aprendizado: o rápido é usado na ART1, onde a atualização de pesos é feita rapidamente durante a ressonância. O lento, usado na ART2, tem mudanças mais lentas dos pesos, que ocorrem durante a fase de testes. ART tem sistema de aprendizado rápido associado

a memórias de longo prazo, e aprendizado lento com mudança gradual das memórias a cada tentativa.

As fases e processos de treinamento, dependendo da arquitetura ART, sofrem variações. As classes de ART possuem variações nos padrões de ativação de F2, que é equivalente a uma camada oculta de redes (redes onde o aprendizado é lento). Também sofre variações com relação ao número de nós vencedores em F2, como na *distributed ART* (dART), o que confere maior maleabilidade no aprendizado da rede (CARPENTER, 1997; CARPENTER; GROSSBERG, 2017).

Um dos principais problemas resolvidos pela ART é o dilema estabilidade *versus* plasticidade. Isto é, a capacidade de ter resposta adequada para entradas significantes, sem ser afetado por entradas não significativas (ruído), e ainda assim manter a capacidade de aprender e se adaptar a novos padrões (CARPENTER; GROSSBERG, 2017; SILVA; ELNABARAWY; II, 2019).

Apesar de possuir várias vantagens como grande velocidade no treinamento, possibilidade de paralelização, "configurabilidade" (capacidade de adaptar a arquitetura a diversos tipos de aprendizado e aplicação, especialmente ao não supervisionado, destacando-se por não necessitar definir o número de *cluster* previamente), pesos internos interpretáveis (não é uma caixa-preta); as ART têm consideráveis desvantagens (SILVA; ELNABARAWY; II, 2019).

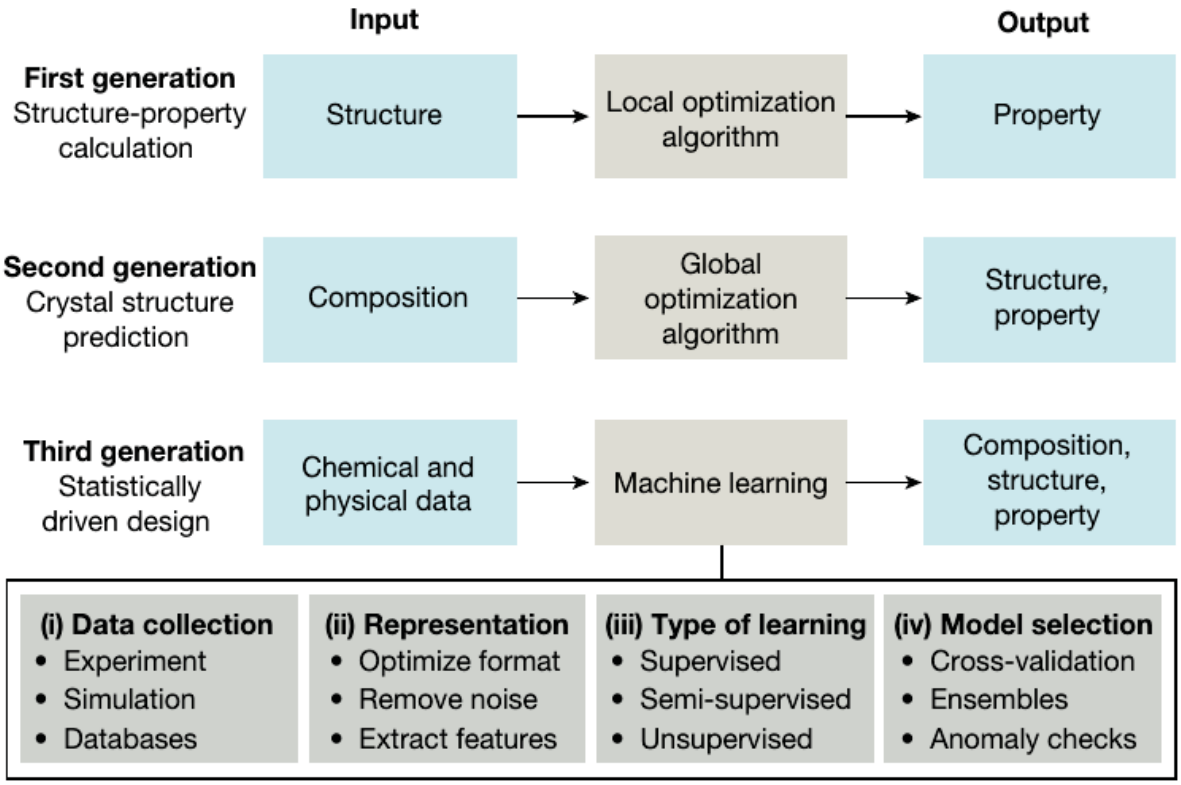
Uma das grandes desvantagens da ART é o aprendizado *online*, isto é, o aprendizado obedece a ordem de apresentação dos dados. O treinamento ocorre usando um dado por vez, aprimorando a rede passo a passo, ao invés de utilizar o conjunto completo de entrada, como nas demais redes. Várias técnicas são utilizadas para mitigar o problema, incluindo diversos tipos de pré-processamento e até o uso de GA (SILVA; ELNABARAWY; II, 2019).

O treinamento de redes ART ainda é desafiador quando se considera a definição de parâmetros, como o de vigilância. O uso de nó (ou nós) vencedor pode ser considerado um limitador do modelo, uma vez que impede o aprendizado de padrões mais complexos. Entretanto, é essa característica que torna o aprendizado mais rápido, e substituí-la pode tornar o aprendizado mais lento (SILVA; ELNABARAWY; II, 2019).

3.6 MACHINE LEARNING APLICADO A MECÂNICA QUÂNTICA E DINÂMICA MOLECULAR

Quando pensamos em simulações e representações atomísticas, supõe-se que se conseguimos simular de maneira apropriada as propriedades atômicas, deveríamos conseguir bons resultados na simulação de qualquer molécula. Entretanto, quanto maior o número de átomos, maior a complexidade e maior o esforço computacional. Nesse contexto, a busca por simplificações e atalhos é constante (BUTLER *et al.*, 2018). Um exemplo de atalho é a aproximação de Bohr-Oppenheimer para o caso elétron-núcleo. Não bastasse isso, nem sempre conseguimos simular as propriedades atômicas corretamente. O DFT é um método bem-sucedido

FIGURA 3.11: EVOLUÇÃO DA PESQUISA EM QUÍMICA COMPUTACIONAL.



FONTE:Butler *et al.* (2018).
LEGENDA: Estamos atualmente na terceira geração do aprendizado de máquina.

em várias classes de compostos, com grande precisão e custo computacional médio, o que lhe deu popularidade e grande número de estudos; mas seu método é limitado e apresenta problemas, como já apresentado no capítulo anterior.

O aprendizado de máquina já é aplicado com frequência em problemas de dinâmica molecular assim como em mecânica quântica. O uso de ML vem sendo cada vez mais reconhecido na literatura como um meio alternativo de se simular sistemas químico-quânticos (SELLIER *et al.*, 2019). Com afirma Butler *et al.* (2018), estamos na geração do aprendizado de máquina, no que se refere à pesquisa na química computacional (Fig. 3.11). Mencionamos aqui a química computacional, sem esquecer da física computacional, áreas que sofrem sobreposição especialmente em níveis mais baixos de complexidade, como o atômico e de moléculas simples. As técnicas de ML envolvem processos não lineares, isso confere vantagem na resolução de problemas em comparação a métodos computacionais convencionais (BUTLER *et al.*, 2018).

Cálculos *ab initio* de diferentes níveis – atômicos e moleculares – são usados para treinar redes neurais e criar modelos de predição de grandes estruturas (BEHLER; PARRINELLO, 2007; LI *et al.*, 2013; DOLGIREV; KRUGLOV; OGANOV, 2016). Tais predições auxiliam muitas vezes na descoberta de propriedades em materiais (ARTRITH; URBAN, 2016; RUPP *et al.*, 2012).

Em trabalhos iniciais, utilizando dados de aproximações quânticas (frequentemente DFT) para o treinamento, permitiram que rede neurais com entradas baseadas apenas em informações de carga nuclear e coordenada dos átomos aprendessem a predizer a energia de atomização de moléculas (RUPP *et al.*, 2012; HANSEN *et al.*, 2013; LI *et al.*, 2013). Esses mesmos dados de aproximações, mais recentemente, são usados para treino de redes para fornecer campos de força (CHMIELA *et al.*, 2017; CHMIELA *et al.*, 2018), a energia individual de átomos em um sistema em interação (BEHLER; PARRINELLO, 2007; BEHLER, 2017), modelos tipo *coarsed-grained* baseados em redes profundas (ZHANG *et al.*, 2018a), entre outros (DOLGIREV; KRUGLOV; OGANOV, 2016; BUTLER *et al.*, 2018). A partir desses treinamentos, dados que antes eram obtidos de forma lenta e custosa passam a ser, com certo erro associado, obtidos de forma muito mais rápida.

O ML também vem sendo aplicado como substituto de certas partes dos cálculos de aproximações, como na solução de funcionais presentes no DFT (treinados inclusive com dados experimentais em alguns casos (SNYDER *et al.*, 2012)), e nas equações de Kohn-Sham, que pode ser substituída pelo aprendizado de mapas de densidade de energia e de potencial (BROCKHERDE *et al.*, 2017). Isto é, a ML não mais usa apenas os resultados das aproximações quânticas em seu treinamento, mas também se demonstrou capaz de participar da construção desses resultados, melhorando sua performance.

Na dinâmica molecular, quanto maior a estrutura que se deseja simular, menor é a precisão da simulação, e vice-versa. Pesquisadores são forçados a escolher entre acurácia ou grandes sistemas, pois é difícil obter precisão e um sistema suficientemente grande, com uma simulação que ocorra em tempo viável. Podemos elencar quatro pontos principais que a dinâmica molecular busca constante melhoria: tamanho do sistema, escala de tempo (maior tempo simulado), duração da simulação (menor tempo de processamento computacional) e acurácia (HANSSON; OOSTENBRINK; GUNSTEREN, 2002; BRUNK; ROTHLISBERGER, 2015). A ML acelera a predição por meio de seu aprendizado, permitindo criar sistemas moleculares maiores e simulá-los por mais tempo, pois exige menor esforço computacional. Isso tanto sob a perspectiva atômica e de moléculas pequenas (BROCKHERDE *et al.*, 2017) como para resolver interações de grande número de molecular interagentes (BEHLER, 2017).

Na literatura, quando se busca acerca da implementação de redes neurais ou outras metodologias de aprendizado à predição química, encontra-se uma literatura muito vasta, com aplicações das mais variadas, generalistas e específicas. Mas podemos elencar 3 níveis de aprofundamento (ou fases como sugere Mater e Coote (2019)): (1) compreender moléculas: acelerar aproximações da mecânica quântica, e exploração conformacional; (2) desenho de moléculas: inclui o *design* de materiais, drogas e outras moléculas; (3) sintetizar moléculas: predição e otimização de reações, retrosíntese.

Nosso foco está na primeira fase. Estudos dessa fase incluem aqueles que visam acelerar as aproximações da mecânica quântica (SNYDER *et al.*, 2012; BROCKHERDE *et al.*, 2017), exploração conformacional e predição de propriedades e atividades de moléculas e sistemas

de moléculas[12] (MONTAVON *et al.*, 2013), e outras propriedades físico-químicas de átomos e materiais (SIGMAN; RIVES, 1994; ARTRITH; URBAN, 2016). E inclusive trabalhos mais basais como elétrons presos a campos potenciais com predições de energia (MILLS; SPANNER; TAMBLYN, 2017).

Dentre os exemplos de trabalhos como esse estão as redes que fornecem a energia dos átomos com base em treinamento utilizando dados de DFT (BEHLER; PARRINELLO, 2007; BEHLER, 2017). Não foram encontrados trabalhos que busquem estudar o átomo ou moléculas utilizando posições como saída da rede.

Há uma tendência recente no uso de redes neurais profundas (*deep learning*) para tratar de problemas envolvendo química computacional, apresentando em muitos casos resultados que superam as expectativas (GOH; HODAS; VISHNU, 2017; MATER; COOTE, 2019). Goh, Hodas e Vishnu (2017) fizeram uma meta-análise comparando métodos usando redes profundas contra métodos de ponta não baseados em redes na literatura, e em quase todos os testes o resultado utilizando redes foi melhor. A grande eficiência das redes profundas é decorrente de seu grande tamanho, que permite aprender grande número de padrões presentes nos dados. As aplicações da *deep learning* são diversas, e mais informações podem ser encontradas nos trabalhos de Goh, Hodas e Vishnu (2017) e Mater e Coote (2019).

Mas há algo em comum em quase todos os métodos apresentados até o momento: são supervisionados, e geralmente associados a grandes bancos de dados confiáveis e curados. Poucos trabalhos envolvem aprendizado não supervisionado, como as novas maneiras de se visualizar dados por mapas topográficos generativos de Kireeva *et al.* (2012). E também são poucos são os trabalhos que relacionam diretamente a teoria física ao treinamento de seus modelos, como os mencionados Faulkner *et al.* (2018), Lei, Krauth e Maggs (2019) (trabalhos que não utilizam redes neurais, mas ECMC).

Não foram encontradas aplicações similares para a previsão de energias do elétron ligado a um núcleo assim como não foi encontrado na literatura trabalhos que busquem prever a posição do elétron.

[12] Os campos da relação quantitativa da propriedade/atividade da estrutura (QSPR e QSAR, *Quantitative Structure Property/Activity Relationships.*) buscam prever as propriedades de sistemas moleculares e a atividade de moléculas em sistemas biológicos, respectivamente (MATER; COOTE, 2019).

4 PROPOSTA

O hidrogênio é o átomo mais simples e de solução precisa e confiável, se tornando um candidato ideal para estudo. Por este motivo nosso foco será dado no átomo hidrogênio em um ambiente isolado (não perturbado), servindo de ponto de partida para o desenvolvimento do modelo computacional. O objetivo futuro para o modelo aqui construído está na sua incorporação a simulação de átomos e moléculas em interação.

Para se desenvolver um modelo que represente o átomo de hidrogênio é preciso ter um método capaz de representar o elétron. Com base no levantado na revisão da literatura (Capítulos 2 e 3), pudemos organizar uma série de requisitos para o método e para o modelo a ser construído são levantados na Seção 4.1 a seguir. Em seguida, na Seção 4.2, são discutidos os métodos levantados na revisão em contraposição aos requisitos estabelecidos. Por fim, na Seção 4.3, é apresentada a estrutura da rede neural desenvolvida. A rede denominada *Feature Resonance Neural Network* é parte dos resultados desse trabalho, contudo, para tornar clara a nossa proposta tal como a necessidade da metodologia empregada, sua descrição foi inserida neste capítulo.

4.1 REQUISITOS DO MÉTODO

Considerando o levantamento das características do elétron ligado apresentado no Capítulo 2, realizamos abaixo uma breve discussão sobre os requisitos acerca do método a ser utilizado para a construção do modelo de comportamento do elétron.

O primeiro ponto levantado foi o de que não possuímos dados conhecidos de trajetória do elétron para treinar uma rede, portanto redes capazes de realizar apenas treinamento supervisionado foram descartadas. A Interpretação de Copenhagen é puramente estatística, logo a informação disponível é apenas um *resultado global*, um resultado que se enquadra melhor no aprendizado por reforço como forma de avaliação, ou em redes que realizam otimização.

O termo “global” será utilizado neste trabalho para se referir ao resultado estatístico da posição do elétron e à avaliação realizada sobre esta estatísitica. A avaliação global trata de uma avaliação estatística realizada sobre o resultado de várias iterações, obtendo-se diversas posições do elétron no tempo – como a distribuição final de todas as posições. Por outro lado, a avaliação local é referente à uma avaliação ponto a ponto ou a avaliação sobre o resultado imediato da transformação das entradas – como quando sabemos qual é exatamente o próximo passo da série temporal. A avaliação local assemelha-se a avaliação supervisionada, onde dada uma posição a posição seguinte seria conhecida.

Para se obter um resultado global, é necessário uma série de pontos (posições) em sequência temporal – pensando no problema da “trajetória” do elétron. Esse tipo de sequência é obtida por processo iterativo, onde o formato de *output* e *input* devem ser compatíveis. Isto é,

conhecendo-se a posição presente, podemos extrapolar a posição seguinte ou o próximo estado. Por fim, o resultado da iteração é um conjunto de pontos em sequência que devem ter uma distribuição similar à densidade de probabilidade do elétron ligado.

O processo iterativo, ou recursivo, no tempo é resolvido por modelos denominados *time series forecasting*, um conjunto de modelos aplicados à previsão de valores futuros baseados nos valores passados conhecidos. Contudo, esse tipo de modelagem é geralmente supervisionada, com uma sequência temporal conhecida, o que não condiz com a nossa disponibilidade.

O modelo deve fornecer o resultado com rapidez. Com o objetivo maior de realizar sua integração à dinâmica molecular, modelos como redes neurais se tornam atraentes pois, mesmo que seu treinamento seja lento, a rede treinada possui uma transformação das entradas muito rápida.

Outro pré-requisito é a necessidade de se ter uma função de custo maleável. Função de custo é um termo emprestado da nomenclatura do aprendizado por reforço. Tal função pode ser considerada tanto uma função avaliativa quanto uma função de otimização externa à otimização dos pesos internos da rede. Uma função customizável, no presente contexto, significa que deve permitir uma avaliação em nível global e local simultaneamente. Isto é, a função de custo permita incluir diversos formatos de avaliação do resultado da rede como a possibilidade de recursão interna, avaliação sobre múltiplos resultados globais como distribuições das coordenadas radial e angulares da posição do elétron, etc.

Também consideramos como requisito a não necessidade de se realizar integração das posições. Objetivamos que o modelo de aprendizado forneça a próxima posição do elétron diretamente, sem precisar de nenhuma operação matemática adicional.

Em suma, podemos listar nossos pré-requisitos para o método como: (1) possibilidade de avaliação local e global, (2) rapidez na previsão, (3) função de custo maleável, (4) possibilidade de iteração – sequência temporal, (5) ausência de supervisão, e (6) ausência de integração de posições.

4.2 DETALHAMENTO

Após haver definido os pré-requisitos acima, foram examinados os métodos de IA existentes na literatura (cap. 3).

Redes de Otimização

Pudemos observar que as redes de otimização apresentadas não satisfazem as exigências para o presente trabalho, já que a maioria oferece solução para problemas de otimização combinatória, ou exigem algum grau de supervisão que não dispomos. E os demais modelos não fornecem meios de utilizar uma função de custo maleável.

Precisamos levar em consideração que o problema da modelagem do comportamento do elétron é um problema de otimização, pois visa otimizar as saídas da rede para corresponder a

uma distribuição conhecida. E não apenas isso, para uma solução mais precisa é necessário a incorporação de mais informações como valores médios e centro de massa. Apesar do problema proposto ser um problema de otimização, as redes com aprendizado não supervisionado com esse fim não se apresentaram compatíveis com a modelagem do comportamento do elétron.

Apesar do comportamento do elétron poder ser definido por estados permitidos e com um conjunto de soluções ótimas (e quasi-ótimas), ele não tem um número de elementos finitos nem mesmo um número finito de estados. Portanto, o problema deste trabalho não pode ser considerado do tipo combinatório.

Redes ART

Para o propósito deste trabalho, os modelos ART existentes não apresentam a inclusão de uma estrutura para função de custo variável. Existe na verdade um grande limitador com relação ao tipo de uso, uma vez que existem arquiteturas específicas para a solução de cada tipo de problema – regressão, classificação, etc. A menção da ART se viu necessária, uma vez que a rede aqui desenvolvida apresenta características de ressonância, contudo, como será detalhado, a rede proposta não se assemelha às arquiteturas ART.

Aprendizado por Reforço

Os métodos de aprendizado por reforço foram o que mais se aproximaram da solução. Os métodos *model-free* apresentam vantagem por não precisar conhecer as probabilidades de transição entre os estados, o que é necessário para satisfazer os requisitos estipulados .

Os métodos não contínuos apresentados (estados e ações discretas) não permitem a descrição da posição do elétron, contudo os contínuos sim. O modelo que mais se aproxima em satisfazer as exigências para a solução do problema é o apresentado pelo trabalho de Gaskett, Wettergreen e Zelinsky (1999). Este trabalho abre a possibilidade da integração do apredizado por reforço às redes neurais com o uso de estados e ações contínuos.

Markov Chain Monte Carlo

Como já mencionado, o método ECMC é altamente supervisionado e os métodos MCMC e ECMC não envolvem um aprendizado propriamente dito. As aplicações dos métodos MCMC, também os ECMC, são centradas em amostragem (ANDRIEU *et al.*, 2003; FAULKNER *et al.*, 2018; VATS *et al.*, 2019; LEI; KRAUTH; MAGGS, 2019), ou seja, não é como uma rede neural que armazena os pesos e permite posterior uso da rede já treinada, uma vez que não há um processo de treinamento. O MCMC irá gerar um amostral novo a cada uso a partir da geração de números aleatórios. Esse processo pode ser lento e é incompatível com a proposta, que exige velocidade na previsão as próximas posições. Não foi encontrado nenhum trabalho que utilize MCMC, sem associação a outros métodos, para aprendizado.

Levando em consideração o que conhecemos acerca do comportamento do átomo de um elétron, percebemos que nenhuma das redes e outros métodos presentes na literatura satisfaziam os requisitos para a construção de um modelo para a descrição do comportamento do elétron, ou apresentaram alguma desvantagem em seu uso. Foi a inabilidade de outros métodos em satisfazer estes pré-requisitos que motivou a construção de uma nova rede. A rede denominada *Feature Resonance Neural Network* (FRes) é descrita na Seção 4.3 a seguir.

4.3 *FEATURE RESONANCE NEURAL NETWORK*

Como características principais da rede podemos destacar que se trata de uma rede independente de supervisão, pois realiza otimização com base em uma função de custo externa e customizável a cada problema investigado. Para melhor compreensão de seu funcionamento poderíamos realizar a analogia com um jogo, no qual as entradas da rede são como informações sobre o estado atual do jogo, as saídas como o estado seguinte do jogo após a jogada, e a função de custo se comportaria como um avaliador da qualidade da jogada incluindo as regras do jogo (jogadas válidas). Neste trabalho, as regras do jogo serão as leis físicas relacionadas ao comportamento do elétron em torno de um núcleo de hidrogênio, as entradas serão sua posição atual, as saídas sua posição seguinte, e o avaliador determina o quão próximo do esperado foram as saídas de maneira global.

4.3.1 Neurônios da FRes

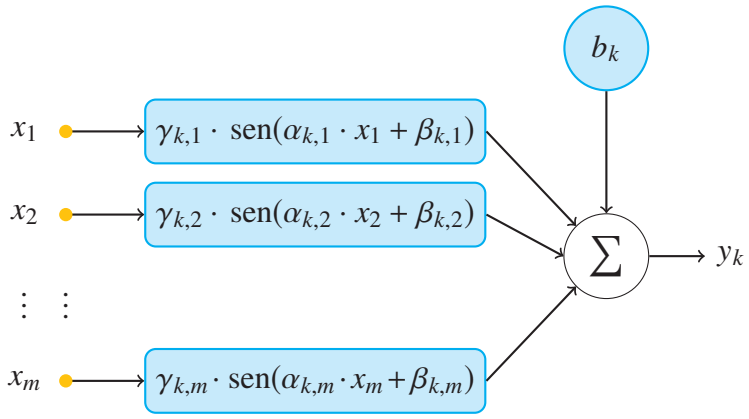
A FRes fornece duas opções de neurônios, um com e um sem ressonância. Todos os neurônios da rede serão ou de um tipo ou do outro, conforme a escolha. O neurônio *sem ressonância* se assemelha ao perceptron (Fig. 3.1), mas sem o uso de uma função de ativação e com um bias opcional.

O neurônio *com ressonância*, por sua vez, é dado pela Figura 4.1, onde x_i corresponde ao valor do i-ésimo neurônio da camada anterior de um total de m nós, α, β, γ são os pesos da rede, e b_k é o *bias*, opcional para a constituição do neurônio k, e o valor atual deste neurônio é y_k. Ou seja, existem três pesos por neurônio, que são otimizados durante o treinamento, e mais o *bias* que também é otimizado. Pode-se observar que somente após a transformação senoidal dos valores da camada anterior, mediada pelos pesos, é realizado o somatório. Essa estrutura senoidal foi inspirada na série de Fourier, contudo diferente desta não consiste de um somatório, mas sim de uma estrutura em recursão, onde a saída de uma camada é a entrada da camada seguinte. Essa estrutura permite a solução de problemas não lineares, como será demonstrado na Seção 6.1.1.

A série de Fourier pode ser descrita como uma série trigonométrica baseada em senos e cossenos, descrita como

$$F(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left[a_n \cdot \cos\left(n\omega t + \varphi\right) + b_n \cdot \operatorname{sen}\left(n\omega t + \varphi\right) \right].$$

FIGURA 4.1: ESTRUTURA DO NEURÔNIO COM RESSONÂNCIA DA REDE FRES.



FONTE: O autor (2020).
LEGENDA: x_i corresponde ao valor do i-ésimo neurônio de m nós totais da camada anterior, α, β, γ são os pesos da rede otimizados durante o treinamento. b_k é o *bias*, também otimizado no treinamento, opcional para a constituição do neurônio. y_k é a saída do neurônio k.

Ela é utilizada para representar funções contínuas periódicas complexas, especialmente aplicado a problemas físicos (BUTKOV, 1973). Fourier é uma série com variadas aplicações e amplo estudo. De forma similar à série de Fourier, com esta rede objetivamos que ela tenha a capacidade de representar padrões variados para os quais ela for apresentada. A ressonância aqui descrita não é um fenômeno observável, mas sim uma inspiração para o desenvolvimento da função.

Apesar de fornecermos duas opções de neurônio, é no neurônio com ressonância que será dada a ênfase deste trabalho, e onde estão localizados os resultados mais interessantes.

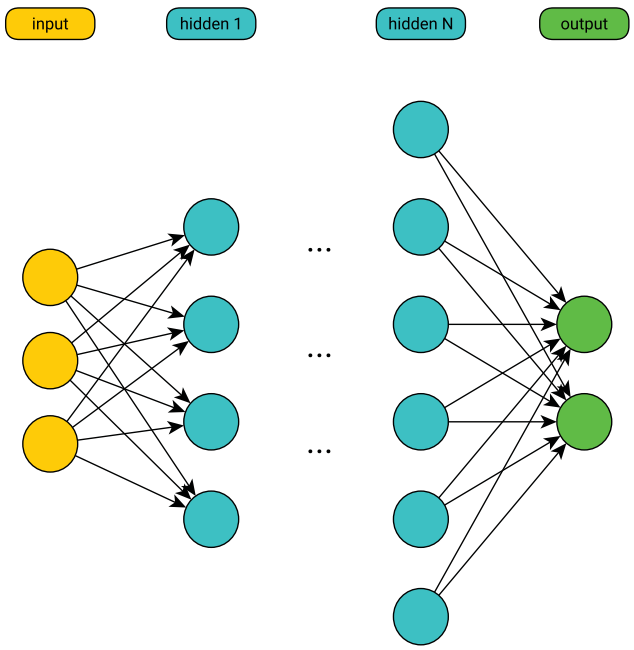
4.3.2 Estrutura da FRes

Sua estrutura é do tipo *feedfoward* não recorrente, todos os neurônios de uma camada se conectam com todos os neurônios da próxima camada em sentido único (estrutura similar a *Multilayer Perceptron* (MLP), apresentado na Figura 3.2). A FRes possui uma camada de entrada, uma de saída e N camadas ocultas, cada camada com um número variável de neurônios. A Figura 4.2 apresenta uma visão geral da estrutura da rede proposta.

Com a integração dos neurônios descritos na Subseção 4.3.1 anterior, a rede pode ser descrita como:

$$
\begin{aligned}
x_{j,k} &= \begin{cases} \sum_{i=1}^{n_{j-1}} \gamma_{j,k} \, \text{sen}(\alpha_{j,k} x_{j-1,i} + \beta_{j,k}) & , r_{eson} = 1 \\ \sum_{i=1}^{n_{j-1}} \alpha_{j,k} x_{j-1,i} & , r_{eson} = 0 \end{cases} \\
x_{j,k} &= \begin{cases} x_{j,k} + b_{j,k} & , b_{ias} = 1 \\ x_{j,k} & , b_{ias} = 0 \end{cases}
\end{aligned}
\tag{4.1}
$$

FIGURA 4.2: ESTRUTURA GERAL DA REDE PROPOSTA.



FONTE: O autor (2020).
LEGENDA: Em amarelo a camada de entrada (*input*) com 3 neurônios, em azul as camadas intermediárias (*hidden*), que vai de 1 a N, com variado número de neurônios por camada, e em verde a camada de saída (*output*) com 2 neurônios. As setas indicam o sentido do fluxo de informação (*foward*), conectando todos os neurônios entre as camadas subjacentes, mas sem conexões entre os neurônios da mesma camada.

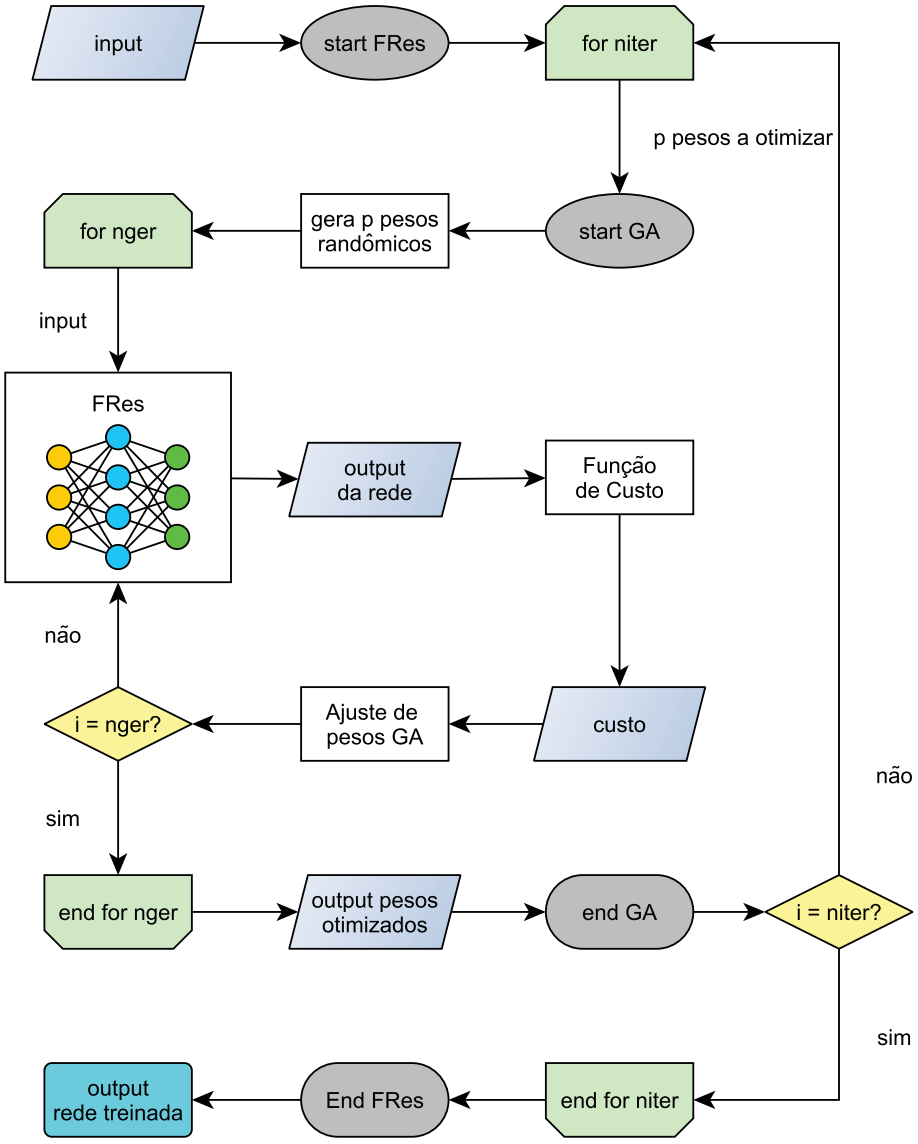
onde r_{eson} indica a presença de ressonância (definindo o tipo de neurônio a ser usado), e b_{ias} a presença de *bias*. Cada camada j possui n_j nós. $x_{j,k}$ corresponde ao valor do k-ésimo nó da camada j atual que está sendo calculada, e $x_{j-1,i}$ o i-ésimo nó da camada anterior. $b_{j,k}$ é o *bias* do k-ésimo nó da camada j. $\alpha_{j,k}, \beta_{j,k}, \gamma_{j,k}$ são os pesos da camada j e nó k atuais, pesos que são otimizados durando o processo de treinamento. O *input* corresponde a todos os n_j nós da camada $j = 1$, e o *output* aos nós da última camada $j = n_{layers}$.

4.3.3 Processo de Treinamento (Otimização de pesos)

Desenvolvemos duas funções principais, uma de treinamento e uma de uso da rede. Cada uma é subdividida nas seguintes funções e subrotinas

1. `FResTR`: função de treinamento;
   (a) FRes: a rede em si: distribuição de pesos nas camadas e transformação das entradas;
   (b) Função de otimização: algoritmo genético;
   (c) Função de custo: uma função específica para cada problema, podendo ser fornecida pelo usuário.

FIGURA 4.3: FLUXOGRAMA DA FUNÇÃO DE TREINAMENTO `FRESTR`.



FONTE: O autor (2020).

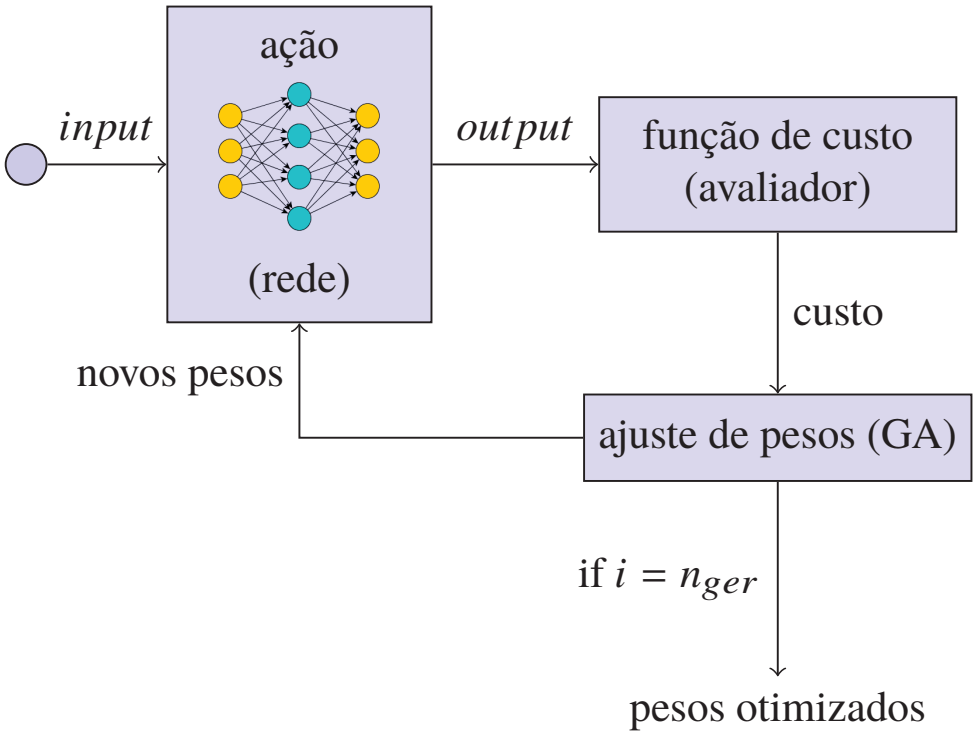
LEGENDA: Seguindo o fluxo de informação, a função de treinamento `FResTR` chama a GA, a qual otimiza os pesos verificando o *fitness* de cada conjunto de pesos chamando a função de custo. A GA realiza n_{ger} *loops* de otimização, e a `FResTR` realiza n_{iter} *loops* de otimização. A saída do processo é a rede com os pesos otimizados. Este fluxograma está descrito no Algoritmo B.6 de forma mais detalhada.

2. `FResTS`: função de uso da rede (após o treinamento);

   (a) FRes: a rede em si.

O fluxograma de treinamento, apresentado na Figura 4.3, e o Algoritmo B.6 em anexo apresentam as funções utilizadas passo a passo. Seguindo o fluxo de informação, inicialmente se fornecem os parâmetros a serem utilizados durante o processo de otimização, juntamente a função de custo a ser utilizada e o *input*. Inicia-se um *loop* de n_{iter} iterações onde a GA é chamada. A GA recebe informação de quantos pesos otimizar, inicializando com uma população de n_{pop} conjuntos de p pesos aleatórios (p é o tamanho do indivíduo da população da GA). Inicia-se um *loop* de n_{ger} gerações onde, primeiramente, o *input* fornecido é transformado em um conjunto de n_{pop} *outputs*, um para cada conjunto de p pesos fornecido. Em seguida é chamada a função de custo para avaliar cada *output* e fornecer o valor do custo, ou *fitness*. Por fim é realizado o processo evolutivo da GA, otimizando os pesos, iniciando novo *loop* de gerações. Ao fim das gerações, repete-se o processo para n_{iter} iterações. Concluindo todos os *loops*, a função de treinamento devolve a rede de melhor *fitness* com seus pesos otimizados. A Figura 4.4 traz o mesmo esquema de maneira simplificada e didática.

FIGURA 4.4: FLUXOGRAMA DA FUNÇÃO DE TREINAMENTO `FRESTR` SIMPLIFICADO.



FONTE: O autor (2020).

LEGENDA: De maneira simplificada, o treinamento `FResTR` inicia com uma entrada fornecida. Pesos aleatórios processam tal entrada e devolvem uma saída correspondente. Essa saída é avaliada pela função de custo que informa o custo à GA. Com base nos custos, a GA define novos pesos à rede. Esse ciclo ocorre n_{ger} até se obter pesos otimizados.

A FRes realiza otimização com base em uma função de custo externa e customizável a cada problema investigado. É importante ressaltar que existem duas otimizações ocorrendo no treinamento: (1) a otimização dos pesos da rede, como ocorre no treinamento de qualquer rede, e (2) a otimização da saída da rede por meio da função de custo, que avalia as saídas da rede a cada passo de ajuste de seus pesos. A otimização dos pesos é diretamente dependente da otimização das saídas.

A GA utilizada é melhor descrita na Seção 5.1. Optou-se em utilizar otimização evolutiva na FRes por vários motivos. Primeiramente, a GA possui uma estrutura própria que permite o uso de funções de custo customizáveis, o que permitiu integrá-la de forma natural a estrutura da FRes. Segundo, os algoritmos *backpropagation*, ou Levenberg-Marquardt BP, mais comumente utilizado, são voltados para problemas supervisionados, dependendo de se conhecer a saída para entrada fornecida. Além disso, BP depende inerentemente da estrutura da rede a qual otimiza, ou seja, seria necessário desenvolver um algoritmo de otimização específico para a arquitetura da FRes, o que não foi realizado, por ora. Por fim, apesar de a GA possuir como desvantagem o grande tempo dispendido na busca, o fator busca é também uma vantagem, uma vez que permite explorar o espaço da função, e assim ter maior chance de alcançar o mínimo global, ou um bom mínimo local.

Listam-se abaixo os parâmetros ajustáveis da FRes:

r_{eson}	ressonância da rede	n_{pop}	tamanho da população da GA
n_{iter}	número de iterações da rede	m_{rate}	taxa de mutação da GA
n_{layers}	configuração das camadas da rede	n_{elite}	tamanho da elite da GA
b_{ias}	uso de bias	n_{iterSA}	número de gerações da SA
n_{ger}	número de gerações da GA	t_{emp}	algoritmo de temperatura da SA

4.3.4 Função de Custo

A função de custo é fornecida pelo usuário ou pode-se utilizar as funções nativas da rede: classificação, regressão e matriz de distâncias. A função de *classificação* (supervisionada), como já apresentada na equação 5.1 da metodologia, calcula a diferença entre o previsto e alvo fornecido. A *regressão* possui igual equação, apenas não realiza arredondamento da saída, pois sua saída não se trata de classes:

$$\begin{aligned} c_{class} &= \sum_{i=1}^{N} |\text{round}(y'_i) - y_i| \\ c_{reg} &= \sum_{i=1}^{N} |y'_i - y_i| \end{aligned} \tag{4.2}$$

onde c_{class} é o custo da classificação, c_{reg} da regressão. `round` remete ao inteiro mais próximo, i é a i-ésima instância de N totais, y' é a previsão da rede e y é a classe correta. O absoluto garante que o erro seja sempre positivo, evitando que os custos se cancelem.

Há ainda a opção do uso de *matriz de distâncias* que realiza uma redução de dimensão das entradas.Esta função não foi utilizada e nem serão apresentados testes sobre ela, pois foge ao escopo do trabalho, contudo é preciso descrevê-la pois é uma opção existente da FRes. A opção de custo c_{pdist} de matriz de distâncias pode ser descrita como

$$\begin{aligned} x &= \text{norm}_{min-max}[\text{pdist}(x_{input})] \\ y &= \text{norm}_{min-max}[\text{pdist}(y_{output})] \\ c_{pdist} &= -\rho_P(x, y) \end{aligned} \tag{4.3}$$

onde ρ_P é a correlação de Pearson (eq. 5.3), $\text{norm}_{min-max}(\cdot)$ é a função de normalização min-max dada por

$$\text{norm}_{min-max}(y) = \frac{y - min(y)}{max(y) - min(y)}, \quad \forall\ y_i \in y,$$

e pdist$(\cdot)$ é a matriz de distâncias euclidiana superior linearizada – como uma matriz de distâncias euclidiana possui diagonal principal nula e simétrica, tomamos apenas os valores da matriz superior. Depois linearizamos a matriz para realizar a correlação. Com c_{pdist} obtém-se a correlação entre a matriz de distâncias normalizada do *input*, com todos os seus atributos, e a matriz de distâncias normalizada do *output*. Quanto maior a correlação entre o *input* e *output* melhor, logo menor o custo (negativo). Podendo ser utilizada para redução de dimensionalidade onde a entrada e saída mantém as distâncias originais.

Apresentadas as funções nativas, há ainda a possibilidade de utilizar funções de custo fornecidas pelo usuário. O padrão de entrada se dá por

```
1  function custo = FCusto(win,wout,redeTS,opts)
2  ...
3  end
```

As entradas correspondem a: `win` é o *input* da rede fornecida pelo usuário, `wout` é o *output* da rede após a transformação da entrada correspondente, `redeTS` a rede com os pesos atuais, pronta para uso, e `opts` uma entrada opcional fornecida pelo usuário, que pode corresponder a qualquer entrada simples ou a um `struct` com múltiplas constantes.

Ao fornecer a rede atual `redeTS` – com pesos em fase de otimização – para a função de custo, permite-se gerar dados por iteração (Alg. 5.2) e assim construir uma avaliação global. Os valores da entrada opcional podem incluir parâmetros e dados adicionais a serem utilizados durante a avaliação.

A *Feature Resonance Neural Network* (FRes) está depositada no Github[1] com informações adicionais e instruções de uso.

[1]Código e informações adicionais depositados no endereço: <https://github.com/CamilaPPerico/FeatureResonanceNetwork>

5 METODOLOGIA

Como o levantamento na literatura não obteve êxito em encontrar um modelo de rede apropriado para satisfazer os requisitos levantados na Seção 4.1, é então proposto um novo modelo de rede neural. A rede desenvolvida, *Feature Resonance Neural Network* (FRes), passou por um processo de validação e por fim foi implementada ao problema do átomo de hidrogênio.

De maneira geral, a metodologia empregada pode ser resumida nos seguintes tópicos:

a) realizar um levantamento de características que descrevem o comportamento do átomo de um elétron (já apresentado na proposta na Seção 4.1);

b) desenvolver um modelo de rede neural capaz de representar este comportamento (apresentado na proposta na Seção 4.3);

c) validar o novo modelo de rede;

d) definir o melhor descritor do sistema;

e) comparar as várias funções avaliativas possíveis e identificar a melhor maneira de construir uma função avaliativa global para representar o comportamento do elétron em torno do núcleo;

f) analisar os resultados obtidos pelas melhores redes treinadas.

São apresentados a seguir o algoritmo genético utilizado para otimização dos pesos internos da rede na Seção 5.1, os testes para a validação da rede na Seção 5.2, consistindo do problema do "ou exclusivo", problemas de classificação e um gerador de dados baseado em distribuição. A estrutura das funções de custo testadas para treinamento da rede para a representação do átomo de hidrogênio na Seção 6.3, e por fim a análise realizada sobre as redes treinadas para sua validação do modelo do átomo de hidrogênio na Seção 5.4.

Os códigos utilizados foram implementados em MATLAB versão R2018a[1]. Os gráficos apresentados foram produzidos utilizando MATLAB, gnuplot e o própio LaTeX, e os fluxogramas com o yEd[2]. Os valores das constantes utilizadas foram obtidos no NIST (LINSTROM; MALLARD, 2020). O treinamento das redes foi realizado em um servidor Linux CentOS release 7.6.1810, com 252 Gb de memória, CPU Intel© Xeon© 2.60 GHz.

Como *default*, a rede utiliza algoritmo genético como seu algoritmo de treinamento, função detalhada na Seção 5.1. A rede também oferece a opção de treinamento com *simulated annealing* (SA), um algoritmo evolutivo alternativo ao GA, como demonstração de que outros algoritmos evolutivos podem ser utilizados no lugar da GA. Sua função está presente na Toolbox "*Global Optimization 8.1*" do MATLAB.

[1] MATLAB versão 9.4.0.813654 (R2018a) The MathWorks, Inc., Natick, Massachusetts, United States.
[2] `gnuplot` versão 5.2, `yEd` versão 3.19.1.1, softwares gratuitos.

ALGORITMO 5.1 ALGORITMO GENÉTICO.

```
1: função AlgoritmoGenético(p,optionsGA,NomeFunçãoCusto)
2:     Gerar população inicial aleatória de tamanho n_pop, com indivíduos de comprimento p
3:     para i = 1 até n_ger faça
4:         custo = FunçãoCusto(inputdata,pesos)    // cálculo do fitness de cada indivíduo
5:         seleciono a elite com n_elite indivíduos
6:         realizo crossing over (cruzamento entre indivíduos)
7:         gero mutações em m_rate percentual da população
8:         gero nova população para a próxima geração
9:     fim para
10:    output: melhor indivíduo de população final
11: fim função
```

5.1 ALGORITMO GENÉTICO

O algoritmo genético utilizado como algoritmo de treinamento da rede é similar ao modelo convencional de algoritmo genético descrito na Seção 3.2.2. O GA foi o algoritmo principal escolhido para este propósito pois, além da arquitetura da rede não permitir utilizar o algoritmo de *backpropagation* como apresentado na revisão da literatura, o GA possui características interessantes de exploração da superfície de otimização.

O Algoritmo 5.1 em pseudocódigo apresenta a sequência lógica do funcionamento do GA[3]. Os parâmetros ajustáveis da GA, utilizados no algoritmo apresentado, são:

n_{ger} parâmetro número de gerações da GA (*default* = 100 gerações)

n_{pop} parâmetro tamanho da população da GA (*default* = 100 indivíduos)

m_{rate} parâmetro taxa de mutação da GA (*default* = 15%)

n_{elite} parâmetro tamanho da elite da GA (*default* = 5 indivíduos)

A estrutura deste GA, em detalhes, é composta pelos seguintes passos:

Geração da população

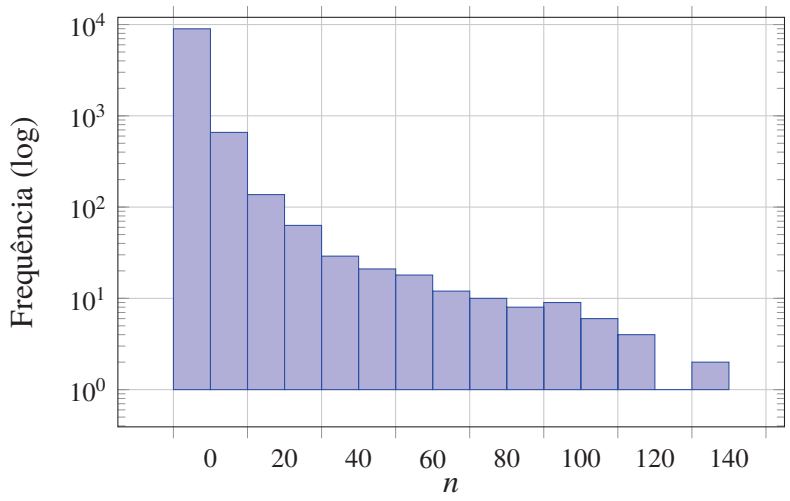
uma população de números tipo `float` (ponto flutuante) é gerada. Cada número é obtido pela divisão de dois números gerados no intervalo [0, 1] pelo método pseudorandômico *Mersenne twister* (MT19937) multiplicado por outro randômico que fornece o sinal (positivo ou negativo). Os números gerados por esse método povoam abundantemente a região entre $|0 - 1|$ e tem queda exponencial na frequência para maiores que 1, como no histograma de frequência da Figura 5.1. Esse valores permitem explorar os pesos para a rede, mas com certo comedimento, mantendo a maioria dos valores pequenos no intervalo [-1,1].

Cálculo do fit

uso a função de custo da FRes como função de *fitness* para os indivíduos da população do GA – a função é customizável para cada problema.

[3]Código depositado no link, e função acessória já inclusa na função de treinamento FResTR.

FIGURA 5.1: GERAÇÃO DE POPULAÇÃO RANDÔMICA PARA O GA.



FONTE: O autor (2020).
LEGENDA: Há maior frequência para valores próximos de zero, e menores, em queda exponencial para valores maiores. Observe que o eixo y do gráfico está em logaritmo, onde foram gerados 10 000 pontos.

Crossover

inserção default uma tendência no cruzamento dos indivíduos com melhor *fitness*. A tendência é gerada ao elevar um conjunto de números randômicos no intervalo de [0, 1] por um fator de 1.3, gerando uma tendência de 30% para os números menores, como abaixo.

$$f(N) = \text{fix}(N \cdot \text{rand}(N)^{1.3}) + 1$$

onde N é o número de números aleatórios a serem gerados, `fix` é a função de arredondamento para o inteiro mais próximo, e a elevação a 1.3 corresponde à tendência gerada.

Por utilizarmos valores de ponto flutuante, e não binários, utilizamos percentuais para o cruzamento. É sorteado um percentual, por exemplo 24%, então um filho será a soma de 24% do parental 1 e 76% do parental 2, e o outro filho 76% do parental 1 e 24% do parental 2.

Mutação

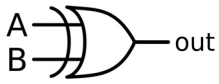
a taxa de mutação *default* utilizada foi de 15% da população. Como os indivíduos tipo *float* não permite alteração de um bit, como no modelo binário tradicional, optou-se pela simples troca do indivíduo sorteado por outro gerado aleatoriamente – da mesma forma como é gerada a população inicial.

Elitismo

o tamanho *default* da elite foi de cinco indivíduos.

TABELA 5.1: OPERAÇÃO LÓGICA “OU EXCLUSIVO”.

A	B	out
1	1	0
1	0	1
0	1	1
0	0	0



FONTE: O autor (2020).
LEGENDA: A tabela apresenta as 4 possíveis entradas (A e B) e suas correspondentes saídas (out). A Figura ao lado mostra a representação simbólica do mesmo.

5.2 VALIDAÇÃO DA REDE

A maioria dos testes realizados sobre a rede para sua validação foram classificações. Para tal, a função de custo consistiu em uma função de similaridade (medida de distâncias), utilizada consistiu da simples diferença entre a saída prevista e a saída esperada, também conhecida como distância de Manhattan, como na equação:

$$custo = \sum_{i=1}^{N} |\text{round}(y'_i) - y_i| \tag{5.1}$$

onde `round` remete ao inteiro mais próximo, i é a i-ésima instância de N, y' é a previsão da rede e y é a classe correta. É necessário o arredondamento pois a saída da rede é ponto flutuante enquanto que as classes são inteiros. O absoluto garante que o erro ou distância seja sempre positivo, evitando que os erros(custos) se cancelem. Esta função foi utilizada para todos os problemas de classificação apresentados neste trabalho.

Esta função permite fornecer um direcionamento para o GA, indicando qual o valor correto para a saída, pois quanto mais distante do alvo, maior é o erro. Diferentemente da função *zero-one* que simplesmente devolve 0 quando correta a classificação, e 1 quando incorreto (SAMMUT; WEBB, 2017), fornecendo nenhum direcionamento para onde estaria a melhor saída.

5.2.1 O ‘ou exclusivo’ (XOR)

O problema do “ou exclusivo”, ou simplesmente XOR, trata-se de uma operação lógica no qual há duas entradas de valor binário (1 e 0, verdadeiro e falso), e uma saída também binária. Se as entradas forem diferentes, a saída é 1 (verdadeiro), mas se as entradas forem iguais, a saída é 0 (falso), como na Tabela 5.1:

Treinamos a rede para reconhecer os padrões apresentados, fornecendo todas as 4 combinações possíveis de entrada. O treino foi uma classificação supervisionada, utilizando já mencionada função de custo da equação 5.1. Avaliamos a diferença entre o treino de rede *com* e *sem* ressonância. Foram realizados 1000 treinamentos para cada um dos dois casos. Utilizamos como entrada as 4 instâncias (4 padrões distintos) de 2 atributos e uma saída, usando as camadas

[2 2 1], i.e., 2 *inputs*, uma camada oculta com 2 neurônios e um *output*. Como parâmetros comuns:

$r_{eson} = 1$ (com ressonância);

$n_{iter} = 2$ iterações da rede;

$n_{pop} = 50$ indivíduos na população do GA;

$n_{ger} = 50$ gerações da GA;

A entrada da rede seguiu o formato abaixo (exemplo com ressonância):

```
[netout] = FResTR(TrainDataset, 'Layers', [2 1], 'Resonance',1,
                  'OptFunc', 'classification','nger',2,
                  'PopulationSize',50,'Generations',50);
```

Foi realizado um treinamento adicional, com mais 1000 réplicas, utilizando os mesmos parâmetros, exceto que aumentamos para 100 o tamanho da população.

Maiores detalhes das opções da rede e de seu uso podem ser encontradas no arquivo `README` no link depositado no Github.

5.2.2 Classificação da *Iris*

Uma das primeiras tarefas com a estrutura nova da rede FRes foi determinar se esta funciona realmente como uma rede. Para isso realizamos alguns testes que permitiram verificar sua funcionalidade. O primeiro e mais básico foi verificar a capacidade de classificar as instâncias da *Iris*. Utilizando a estrutura da rede que será detalhada na Seção 4.3, utilizamos a função de custo supervisionada para classificação da equação 5.1.

Os conjuntos de dados para treinamento foram os dados da *Iris* (FISHER, 1936; DUA; GRAFF, 2017), com 150 instâncias, 4 atributos e 3 classes, e o de Câncer de Mama (WOLBERG; MANGASARIAN, 1990; DUA; GRAFF, 2017), com 683 instâncias, 9 atributos e 2 classes. Os conjuntos são descritos com maiores detalhes na Seção A.1 em Anexo.

Foram realizados 4 treinamentos distintos utilizando uma proporção de 70:30[4], com 1000 réplicas cada, para cada conjunto de dados, como descrito na Tabela 5.2. O primeiro conjunto de treinos foi realizado com a rede MLP do MATLAB para comparação com os resultados de nossa rede. A MLP foi treinada com configurações *default*: função de ativação *tansig*, algoritmo de otimização Levenberg-Marquardt *backpropagation*. Os demais 3 testes performados utilizando a FRes envolveram a variação dos parâmetros de ressonância r_{eson}, número de gerações n_{ger} e tamanho da população n_{pop} da GA, além do número de iterações n_{iter}.

Para o dados da *Iris* foram utilizados 3 neurônios na camada oculta e para os dados de Câncer 5 neurônios. Essa escolha se deve ao fato do problema do Câncer possuir maior número de atributos que a *Iris*.

[4]Proporção 70:30 – 70% da amostra utilizada para treino e 30% para teste

TABELA 5.2: PARÂMETROS PARA TREINO DAS REDES DE CLASSIFICAÇÃO.

Rede	r_{eson}	n_{iter}	n_{pop}	n_{ger}
MLP	–	–	–	–
FResA	1	10	100	25
FResB	1	5	500	10
FResC	0	10	100	25

FONTE: O autor (2020).
LEGENDA: Foram treinadas 4 redes (com 1000 réplicas cada) com os parâmetros especificados na tabela. Cada uma das 4 configurações foram usadas para a classificação da *Iris Dataset*, com camadas ocultas [3 2], e outras 4 com igual configuração para o *Breast Cancer Dataset* com camadas ocultas [5 2]. A primeira rede foi uma MLP e as demais com a rede em teste.

5.2.3 Geração de Dados (*Augmentation*)

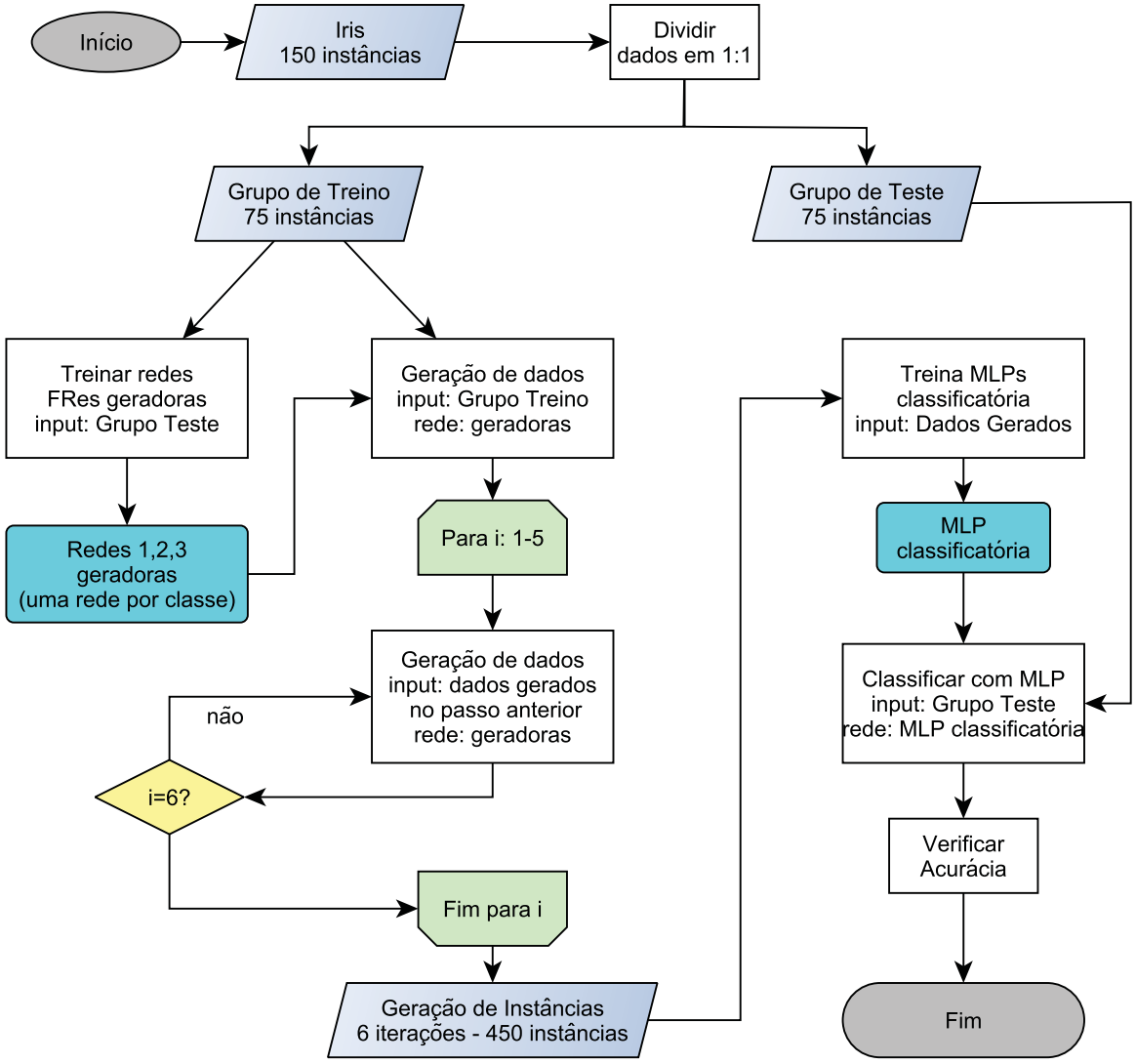
Utilizando os dados da *Iris* (Seção A.1.1 em anexo), treinamos redes de forma a aumentar o *pool* de dados, isto é, gerar dados falsos com as mesmas características dos originais. Essa técnica, também denominada de *augmentation*, já foi utilizada em outros trabalhos (WANG; PEREZ, 2017; GUPTA, 2019) permitindo às instâncias adicionais auxiliar no treinamento de rede classificatórias, por exemplo. Este não é o objetivo deste trabalho, contudo nos permite avaliar se a rede é capaz de reproduzir distribuições apenas por avaliação global, sem avaliação local (sem supervisão).

Fornecendo a uma rede treinada os atributos de uma classe, objetivamos obter um dado similar capaz de representar essa mesma classe. Como a *Iris* possui 3 classes, foram treinadas 3 redes independentes, uma para reproduzir cada classe. Consideramos que cada espécie (classe) possui suas características próprias, e treinar apenas uma rede para reproduzir dados das 3 classes (com 4 atributos de saída cada) deixaria o processo muito mais complexo desnecessariamente.

A Figura 5.2 traz o *pipeline* utilizado. Partindo de um conjunto de dados com 150 instâncias (*benchmark Iris*), dividimos este nos conjuntos de treino e teste numa proporção 1:1. O grupo de treino foi utilizado integralmente para treinar as redes geradoras – três redes, uma para cada classe da *Iris* com cerca de 25 instâncias cada[5]. Após o treino, o mesmo conjunto treino foi submetido à rede (cada classe submetida à rede correspondente) obtendo a primeira leva de instâncias geradas. O conjunto gerado é novamente submetido às redes (6 iterações totais), levando a formação de um conjunto gerado de aproximadamente 450 instâncias. O conjunto gerado é então utilizado integralmente para treinar uma rede MLP classificatória. Após o treino, a MLP é utilizada para classificar o conjunto de teste de dados originais reservado, e a sua acurácia é verificada. Foram realizadas 100 réplicas de treinos da MLP classificatória para obter a acurácia média a partir dos dados gerados por cada uma das 5 réplicas de redes geradoras.

[5]Os valores são aproximadamente 25 instâncias pois antes da divisão dos dados em conjunto de treino e teste eles são embaralhados, e não fornecendo a mesma proporção para cada classe, o que torna a simulação mais verossímil.

FIGURA 5.2: *PIPELINE* DO GERADOR DE INSTÂNCIAS.



FONTE: O autor (2020).

LEGENDA: Primeiramente, os dados são divididos nos conjuntos de treino e teste. Então redes geradoras são treinadas com o grupo de treino, uma rede por classe de dados, 5 réplicas cada. É gerado um novo conjunto de dados a partir dos dados de treino por meio de iterações para cada réplica. Os dados gerados são então utilizados para treinar MLPs para classificação, por supervisão. É verificada a acurácia das 100 réplicas das MLPs com os dados de teste. Obtém-se a média de acurácia do modelo gerador.

Treinamento das redes FRes geradoras

Foram treinadas 5 réplicas de cada rede com proporção 1:1 entre treino e teste (50% dos dados para treino e 50% para teste). Foram usadas 4 entradas, 4 saídas, e uma camada oculta com 8 neurônios. Os parâmetros utilizados foram número de iterações da rede $n_{iter} = 10$, número de indivíduos na população $n_{pop} = 250$, e número de gerações da GA $n_{ger} = 50$.

```
netout = FResTR(data,'Layers',[8 4],'niter',10, 'bias',0,
          'Resonance',1, 'Generations',50, 'PopulationSize',250,
          'Function',@(win,wout,x,layers) Fcusto(win,wout,redeTS,opts));
```

Função de Custo

Utilizamos 3 parâmetros avaliativos: distribuições, correlação e médias. A *diferença entre a distribuição* de cada atributo do conjunto de treinamento contra a distribuição dos dados gerados a partir deste mesmo conjunto foi a primeira avaliação da função de custo. Esse valor foi obtido por meio da diferença entre histogramas, da seguinte maneira: dados dois histogramas normalizados, o dos dados originais H_O e o dos dados gerados H_G, ambos com mesmos intervalos (*bins*),

$$e_{hist} = \frac{1}{2}\frac{1}{4}\sum_{i}^{4}\sum_{j}^{n}|H_{O,i,j} - H_{G,i,j}|, \tag{5.2}$$

onde i é o i-ésimo atributo de quatro, e j é a j-ésima barra do histograma, de um total de $n = 10$ subdivisões do intervalo. A soma da diferença entre os histogramas normalizados, dividido por 4, permite obter a diferença percentual. Como a diferença entre 2 histogramas, fornece na verdade o dobro do erro[6], dividimos o resultado final ainda por 2.

Foram utilizados os intervalos específicos para os histogramas de cada atributo, de forma a abranger todos os pontos dos dados originais. Os intervalos, com 10 subdivisões, foram: C.S.: [4.0,8.0], L.S.: [1.8,5.0], C.P.: [0.8,7.0], L.P.: [0.0,2.7], onde C.S - comprimento da sépala, C.P - comprimento da pétala, L.S - largura da sépala, L.P - largura da pétala.

Como os 4 atributos possuem certa *correlação* entre si, utilizou-se também a diferença entre o valor das correlações esperadas e as obtidas dos dados gerados. A função de correlação escolhida foi a de Pearson (ρ_P):

$$\begin{aligned} \delta_{xi} &= x_i - \bar{x}, \\ \delta_{yi} &= y_i - \bar{y}, \\ \rho_P(\delta_{xi}, \delta_{yi}) &= \frac{\sum_{i=1}^{n}\delta_{xi}\delta_{yi}}{\sqrt{\sum_{i=1}^{n}\delta_{xi}^2}\cdot\sqrt{\sum_{i=1}^{n}\delta_{yi}^2}} \end{aligned} \tag{5.3}$$

onde x e y correspondem a dois conjuntos de amostras distintas (neste caso previsto e esperado), no qual x_i é a i-ésima amostra de x e $\bar{x}$ é a média das suas n amostras. A correlação de Pearson varia no intervalo [-1,1], i.e., entre 100% de correlação negativa a 100% de correlação positiva.

[6]Se a diferença entre dois histogramas for de 10%, a subtração dará +10% para um histograma e -10% para outro.

Cada conjunto de dados gera uma matriz $\rho_P = [\rho]_{4\times4}$, e o erro calculado entre as matrizes pode ser dada como

$$\begin{aligned} S_O &= \sum_i^4 \sum_j^4 |\rho_{O\ i,j}|, \\ e_{corr} &= \frac{1}{S_O} \sum_i^4 \sum_j^4 |\rho_{O\ i,j} - \rho_{G\ i,j}|, \end{aligned} \tag{5.4}$$

onde $\rho_{O\ i,j}$ é a correlação entre os atributos i e j dos dados originais, e $\rho_{G\ i,j}$ dos dados gerados. A divisão por S_O do somatório das correlações fornece o erro percentual da correlação.

Adicionalmente, utilizamos a *diferença das médias* de cada atributo como parâmetro de avaliação. O erro das médias pode ser dado por

$$e_{mean} = \sum_i^4 |\bar{x}_{O,i} - \bar{x}_{G,i}|, \tag{5.5}$$

onde $x_{O,i}$ é o i-ésimo atributo dos dados originais e $x_{G,i}$ dos dados gerados.

Todos os erros calculados são normalizados, e o custo final é composto pela sua soma

$$\text{custo} = e_{corr} + e_{mean} + e_{hist}.$$

Como objetiva-se utilizar a rede iterativamente para obter mais dados, a função de custo também incluiu 6 passos de iteração em seu treinamento, onde a saída da rede no primeiro passo, se tornava a entrada no segundo passo. E sobre cada passo, a mesma verificação de custo era realizada.

Treinamento das MLPs classificatórias

Com as 3 redes treinadas em mãos, geramos instâncias a partir do conjunto teste de dados, em 6 iterações. Utilizamos estes dados para treinar uma rede MLP com configuração *default* já mencionada. Foram usadas duas camadas ocultas [2 1] e uma camada de saída com a classe. Observe que treinamos apenas uma rede de classificação para a saída das 3 redes geradoras, já que o objetivo é conseguir a classe de cada instância gerada. Foram realizadas 100 treinamentos para cada uma das 5 réplicas de treinamento da rede geradora.

Teste das MLPs, Cálculo da Acurácia e Avaliação dos Dados Gerados

Tomamos o conjunto de dados de teste reservado e submetemos a cada uma das 100 réplicas MLP e calculamos a acurácia de cada uma. Obtivemos a acurácia média para cada uma das 5 réplicas de rede geradoras.

Adicionalmente, utilizando como *input* os dados originais da *Iris* completos, foram gerados atributos em apenas uma iteração e então avaliados os erros percentuais das distribuições,

correlações, médias e desvios padrões. O desvio padrão foi considerado como um bom parâmetro de avaliação adicional.

5.3 APLICAÇÃO À MODELAGEM DO ÁTOMO DE HIDROGÊNIO

A primeira pergunta a ser realizada é "o que queremos que a rede nos forneça?". Desejamos obter uma espécie de "trajetória" com um passo de tempo arbitrário, portanto a saída da rede deve ser uma posição. Como posição é o que temos de informação a fornecer para a rede, a sua entrada também dever uma posição. Portanto, entrada e saída da rede possuem mesmo formato, o que permite um processo iterativo de retroalimentação.

Fornecendo um ponto inicial com as coordenadas do elétron (`PontoInicial`) relativas a um núcleo localizado na origem, e com a rede previamente treinada (`RedeTreinada`), podemos utilizar a rede pela função `FResTS`. O exemplo abaixo, em liguagem MATLAB, demonstra a obtenção da trajetória em 100 passos de iteração:

ALGORITMO 5.2 USO DA REDE (MATLAB).

```
1   trajetoria(t,:) = zeros(100,3); % crio vetor nulo com 100 linhas e 3
        colunas
2   for t = 1:100               % itero em 100 passos
3      trajetoria(t,:) = FResTS(PontoInicial,RedeTreinada);
4      PontoInicial = trajetoria(t,:); % atualizo o Ponto inicial
5   end
```

Havendo compreendido a maneira como desejamos fazer uso da rede, precisamos definir o mecanismo pela qual ela será treinada. O primeiro ponto a ser considerado é qual descritor[7] utilizar. Em seguida, precisamos definir como gerar dados para treinar a rede, isto é, quais serão os pontos de entrada para a rede durante seu treinamento. Por fim, e como parte mais importante, definir quais serão as avaliações a serem realizadas sobre o sistema para compor a função de custo. Seguindo este roteiro, dividimos a investigação em 3 fases:

1. Testes Preliminares: usando apenas uma dimensão (raio);
   (a) Tipo de função: escolha de qual é a melhor função, A ou B;
   (b) Parâmetros: testes sobre os hiperparâmetros utilizados;
   (c) Número de Camadas: verificação da acurácia com relação ao número e disposição das camadas;
2. Testes Finais: usando as três dimensões;
   (a) Descritor: coordenadas cartesianas ou esféricas;

[7] Descritor é a maneira pela qual iremos descrever o sistema para a rede, em outras palavras, é o formato de entrada da rede.

(b) Função de Custo: escolha das avaliações na função de custo;

3. Avaliação das redes: análise dos resultados gerados pelas melhores redes.

5.3.1 Geração de Dados

Para se poder treinar uma rede é preciso fornecer dados iniciais (*input*). Como estado inicial para a rede, era necessário que os pontos estivessem em uma distribuição correspondente a de Schrödinger. Para simplificação foi utilizado o Ângstron como unidade de distância, tendo como limites [-3:3] para o estado fundamental. A geração de pontos aleatórios foi feita sorteando 3 pontos em coordenadas cartesianas, formando um cubo com limites citados [-3,3].

São descartados todos os pontos com módulo (raio) superior aos mesmos 3 Ângstrons, resultando numa esfera com pontos aleatórios uniformemente distribuídos em coordenadas cartesianas. Esse pontos são convertidos em coordenadas esféricas, seguindo a Tabela 5.3 de conversão.

Com os valores de ângulos θ e φ em mãos, a coordenada r era descartada e substituída por um valor de raio correspondente à distribuição de Schrödinger, obtido por meio do algoritmo de Monte Carlo Metropolis-Hastings, como apresentado no Algoritmo 3.1. Havendo conseguido as 3 coordenadas esféricas, os mesmos pontos, conforme a necessidade, foram convertidos para coordenadas cartesianas. Esses passos de geração foram agrupados no Algoritmo B.1.

5.3.2 Testes Preliminares

Como teste inicial, consideramos *apenas a distribuição radial*, isto é, ignoramos as distribuições angulares θ e φ – isso torna mais clara a visualização da acurácia final, pois avaliamos apenas uma variável.

Parâmetros

Foram realizados 5 testes no total variando parâmetros, como detalhado na Tabela 5.4. Todos os testes foram replicados 50 vezes cada (exceto FRes05 e FRes06, com 20 réplicas cada, que serão

TABELA 5.3: CONVERSÃO DE COORDENADAS CARTESIANA-ESFÉRICA.

esférica para cartesiana	**cartesiana para esférica**
$x = r\,\text{sen}\theta\cos\varphi$	$r = \sqrt{x^2 + y^2 + z^2}$
$y = r\,\text{sen}\theta\,\text{sen}\varphi$	$\varphi = \arctan\left(\frac{\sqrt{x^2+y^2}}{z}\right)$
$z = r\cos\theta$	$\theta = \arctan\left(\frac{y}{x}\right)$

FONTE: O autor (2020).
LEGENDA: θ corresponde ao azimute $[-\pi, \pi]$, φ à elevação $[-\pi/2, \pi/2]$, e r ao raio $[0, \infty[$.

descritos a seguir). Os parâmetros, e um custo, variados foram: presença de bias (b_{ias}), presença de ressonância (r_{eson}), presença do custo raio médio ($\bar{r}$), e o número de entradas da rede (n_{input}). O parâmetro n_{input} remete ao uso de apenas o ponto presente, ou o uso do ponto presente e de um passado.

Adicionalmente, foram testadas duas formas de avaliação, com as chamadas funções de custo A e B. As duas variantes de função de custo utilizadas são apresentadas no Algoritmo B.3 em anexo. Em testes futuros, elas corresponderão às FRes#A e FRes#B (Tab. 5.4). `FCustoA` corresponde ao uso de poucos pontos iniciais (15) com um grande número de iterações (100), sendo avaliados individualmente, i.e., cada trajetória originária de um único ponto é avaliada individualmente e cada custo é somado aos demais. Enquanto que a `FCustoB` usa vários pontos inciais (200) com poucas iterações internas (5) e uma avaliação comum – trajetórias de diferentes pontos iniciais são unidas em um conjunto só que é analisado.

A avaliação, para todos os testes listados, foi referente a distribuição radial (densidade de probabilidade de Schrödinger) e de raio médio (as funções serão melhor detalhadas na Subseção 3.1.1). Demais parâmetros não foram alterados entre os testes: $n_{layers} = [661]$, $n_{pop} = 100$, $n_{ger} = 25$, $m_{rate} = 0.15$, $n_{elite} = 5$. Os pontos de *input* foram gerados por Monte Carlo Metropolis-Hastings (pelo Algoritmo B.4, RadialmenteDistrib, em coordenadas esféricas, das quais foi utilizada apenas a coordenada radial).

Camadas

Após definir as melhores configurações de parâmetros, realizamos um teste preliminar para avaliar quantas camadas seriam necessárias para obter um resultado minimamente satifatório – configuração descrita na Tabela 5.4 como FRes05 e FRes06. Foram variados o número de nós e camadas ocultas, com 20 réplicas de cada combinação. As camadas de entrada e saída tinham apenas um nó cada para todos os casos, variando as ocultas no seguinte padrão: FRes05 – [1 n 1], e FRes06 – [1 n n 1], onde $n = 1, 2, \ldots, 10$. Note que na chamada da função de treinamento da rede o número de nós da camada de entrada é omitido, ficando [n 1] e [n n 1], respectivamente. Tanto para FRes05 quanto 06, foram utilizadas as funções A e B.

De maneira similar, foram testadas camadas para 3 dimensões (*input* e *output* de 3 nós), e a variação das ocultas foi idêntica: FRes25 – [3 n 3], e FRes26 – [3 n n 3], onde $n = 1, 2, \ldots, 10$. Sendo também utilizadas as funções A e B. Neste contexto, enquanto para 1D foi avaliado apenas o histograma radial e o raio médio, para 3D foram avaliados os histogramas radial e angulares, e o raio médio.

5.3.3 Testes Finais

Inicialmente desconsideramos as distribuições angulares θ e φ. Agora iremos otimizar as saídas das redes utilizando as três coordenadas do elétron em relação ao núcleo.

TABELA 5.4: PARÂMETROS VARIADOS NOS TESTES FRES# DE 1D.

Teste	camadas	r_{eson}	$\bar{r}$	b_{ias}	n_{input}
FRes01A	[6 6 1]	0	1	1	1
FRes02A	[6 6 1]	1	1	1	1
FRes02B	[6 6 1]	1	1	1	1
FRes03A	[6 6 1]	1	0	1	1
FRes03B	[6 6 1]	1	0	1	1
FRes04A	[6 6 1]	1	1	0	1
FRes04B	[6 6 1]	1	1	0	1
FRes08A	[6 6 1]	1	1	1	2
FRes08B	[6 6 1]	1	1	1	2
FRes05A	[n 1]	1	1	1	1
FRes05B	[n 1]	1	1	1	1
FRes06A	[n n 1]	1	1	1	1
FRes06B	[n n 1]	1	1	1	1

Abreviações: $\bar{r}$ – raio médio (3/2 do raio de Bohr), n_{input} – número de entradas.

FONTE: O autor (2020).

LEGENDA: Cada teste foi replicado 50 vezes. Os testes 'A' utilizam poucos pontos (15) com muitas iterações (100) e avaliação individualizada, enquanto que os 'B' utilizam muitos pontos (200), poucas iterações (5) e avaliação comum (Alg. B.3). Os valores de n refere-se ao número de neurônios nas camadas ocultas de FRes05 e FRes06, com camada oculta única e dupla, respectivamente, onde $n = 1, 2, \ldots, 10$.

Formato da Entrada

O formato da entrada, ou o descritor do sistema, tem sua importância pois é como vamos apresentar as informações à rede. Optamos pelos descritores mais simples, baseados apenas nas *coordenadas cartesianas* (x, y, z) ou *esféricas* (r, θ, φ) de posição da partícula. Portanto, a entrada e saída da rede são somente as coordenadas de posição relativa do elétron com relação ao núcleo em um espaço tridimensional. A geração dos pontos foi previamente descrita, com o uso do Algoritmo B.1.

Utilizamos 4 variantes dos Algoritmos B.3 descritos, que passam a ter 3 dimensões e a receber funções avaliativas adicionais para compôr o custo (que será descrito a seguir). Manteremos a nomenclatura, onde `FCustoA` e `FCustoB` utilizam coordenadas esféricas e `FCustoC` e `FCustoD` cartesianas. A função 'C' corresponde a 'A', e 'D' a 'B' em termos de número de pontos e iteração. A Tabela 5.5 resume os testes realizados nessa fase final.

TABELA 5.5: PARÂMETROS VARIADOS NOS TESTES FRES# 3D.

	coord.	n_{layers}	n_{pop}	Hr	CM	$\bar{r}$	H$\varphi\theta$	RD	σ
FRes07A	esf.	[6 6]	100	1	0	0	1	1	0
FRes07B	esf.	[6 6]	100	1	0	0	1	1	0
FRes07C	cart.	[6 6]	100	1	0	0	1	1	0
FRes07D	cart.	[6 6]	100	1	0	0	1	1	0
FRes09A	esf.	[6 6]	250	1	0	0	1	1	0
FRes09B	esf.	[6 6]	500	1	0	0	1	1	0
FRes10A	esf.	[6 6]	100	1	1	1	1	1	0
FRes10B	esf.	[6 6]	100	1	1	1	1	1	0
FRes11A	esf.	[8 8]	100	1	0	1	1	1	0
FRes11B	esf.	[8 8]	100	1	0	1	1	1	0
FRes12A	esf.	[8 8]	250	1	0	1	1	1	1
FRes12B	esf.	[8 8]	500	1	0	1	1	1	1
FRes13A	esf.	[8 6]	100	1	1	1	0	1	0
FRes13B	esf.	[8 6]	100	1	1	1	0	1	0
FRes14A	esf.	[8 8]	250	1	1	1	1	1	1
FRes14B	esf.	[8 8]	500	1	1	1	1	1	1

Abreviações: coord. - coordenadas, cart. - cartesianas, esf. - esféricas, Hr - histograma radial, CM - centro de massa, $\bar{r}$ - raio médio (3/2 de a_0), H$\varphi\theta$ - histogramas angulares, RD - radialmente distribuído, σ - desvio padrão.

FONTE: O autor (2020).
LEGENDA: Apresentam-se as funções de avaliação e outros parâmetros utilizados em cada teste com as 3 dimensões. Os 1 e 0 na tabela indicam a presença ou ausência da avaliação na função de avaliação. FRes07 C e D utilizam coordenadas cartesianas, no qual a função 'C' é equivalente a 'A', e a 'D' é equivalente a 'B'.

Realizamos o comparativo para 3 dimensões das 4 variantes das funções de custo mencionadas, denominados de FRes07A, B, C e D. Neste teste foram considerados como parte do custo as distribuições radial e angulares e a avaliação "RadialmenteDistrib" (descrita a seguir).

Função de Custo

Esta é uma das partes mais importantes para o treinamento. A função de custo é o que determina o problema a ser otimizado pela rede. Para o aprendizado do comportamento do elétron, como no problema da geração de dados, a função de custo também deve ter um custo global não supervisionado, uma vez que não há uma saída esperada para cada entrada. Em linhas gerais, foi realizada uma série de treinamentos com diversas combinações de funções avaliativas para construir a função de custo. O objetivo era prever a trajetória do elétron (entrada e saída da rede das coordenadas do elétron) de forma a obedecer os pré-requisitos estabelecidos com relação a distribuição global destes pontos em iteração.

A função de custo teve diversas combinações de custos, como condensado na Tabela 5.5. As funções de custo utilizadas nessa etapa têm o mesmo formato apresentado pela `FCustoA` e

`FCustoB` (Alg. B.3). Os parâmetros comuns para todos os testes foram um amostral de 50 redes cada, $n_{iter} = 7$, $r_{eson} = 1$, $b_{ias} = 0$, $n_{ger} = 25$, $m_{rate} = 0.15$, $n_{elite} = 5$. As funções de avaliação utilizadas são detalhadas abaixo:

diferenças entre distribuições (histograma da coordenada radial - Hr)

Foram calculadas as diferenças entre os histogramas de distribuição das coordenadas esféricas (caso a rede tenha sido treinada para utilizar coordenadas cartesianas, a saída era convertida para esféricas). O histograma radial obedece a densidade radial de Schrödinger, que foi convertida em histogramas com o padrão de limites $[0 : 0.1 : 3]$ (de 0 até 3Å com passo de 0.1 Å), para padronizar o histograma gerado com os dados previstos. A Figura 5.3(a) traz a distribuição de referência.

diferenças entre distribuições (histogramas das coordenadas angulares - $H\varphi\theta$)

Em coordenadas esféricas, os ângulos de azimute (θ) variam de $-\pi$ a π, e de elevação (φ) variam de $-\pi/2$ a $\pi/2$ (ou de 0 a π). Valores além desses limites podem ser considerados múltiplos (ângulos congruentes), e assim podem ser considerados ângulos válidos. Como a rede não tem saída limitada (tem possibilidade de valores $]-\infty, \infty[$), o uso de ângulos congruentes parece válido e útil.

A conversão de ângulos múltiplos para os valores nos limites acima mencionados é trabalhosa, pois envolve a divisão por 2π radianos, e o uso do resto. Como saída, optamos pelo uso dos cossenos dos ângulos. A ideia de utilizar este método provém do fato de que o cosseno (ou seno) dos ângulos oscilam apenas no intervalo $[-1, 1]$, enquanto que ângulos côngruos têm valores distribuídos de $]-\infty, \infty[$. Assim, o padrão de limites utilizado foi de $[-1 : 1/15 : 1]$ (de -1 até 1 com passo de $\frac{1}{15}$).

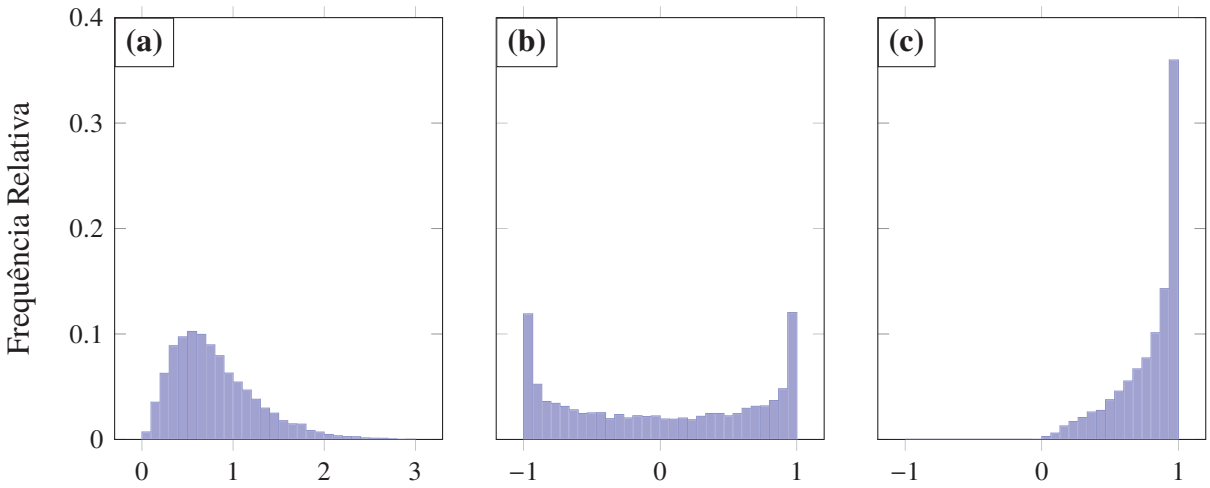
A questão então seria "*qual a distribuição do cosseno dos ângulos θ e φ esperados?*" Pelo fato da densidade eletrônica ser independente do ângulo, os ângulos devem ser homogeneamente distribuídos. Uma amostragem de uma esfera homogênea corresponde a distribuição de ângulos aleatórios. Para verificar se a distribuição estava de acordo com uma distribuição randômica de ângulos, geramos um conjunto de pontos randômicos (como explicado acima na Subseção 5.3.1 de "Geração de Dados"), e gerada a distribuição de seus cossenos como distribuição referencial (fig. 5.3(b) e (c)).

Adicionalmente, o ângulo θ precisa de distribuição na totalidade dos 360º (de $-\pi$ a π), enquanto que φ existe em apenas 180º (de $-\pi/2$ a $\pi/2$). Por este motivo, foi calculado o absoluto do cosseno de φ da saída. O custo foi calculado a partir da diferença entre os histogramas normalizados.

número de iterações (n_{iter})

Observamos a necessidade de se realizar iteração dentro da função custo. Isso permite obter a trajetória de posições do elétron e assim a distribuição. A questão se voltou para o que é mais importante: obter uma trajetória longa, utilizando poucos pontos

FIGURA 5.3: DISTRIBUIÇÕES DE REFERÊNCIA.



FONTE: O auto (2020).
LEGENDA: **(a)** distribuição baseada na densidade de probabilidade de Schrödinger para o estado fundamental, **(b)** distribuição do cosseno do ângulo θ azimutal, **(c)** distribuição do absoluto do cosseno do ângulo φ de elevação.

(`FCustoA`) ou obter em poucos passos a bacia de atração com um número maior de pontos inciais (`FCustoB`) (alg. B.3).

Como conhecemos os pesos da rede de cada indivíduo gerado pela GA, durante o treinamento esses pesos são fornecidos para a função de custo que utiliza a função de teste `FResTS` para realizar as iterações (alg. 5.2). A recursão não está necessariamente associada ao tempo, mas sim ao atrator e a um amostral. A recursão visa representar uma evolução temporal, contudo, o passo de tempo do treinamento é arbitrário, uma vez que o treinamento não exige respeitar o passo a passo, e sim a uma estatística.

raio médio (valor esperado - $\bar{r}$)

O raio médio, também conhecido como valor esperado, é o raio mais provável de ser encontrado e corresponde à média da distribuição de pontos. Para o átomo de um elétron, o valor esperado da coordenada r é de $\bar{r} = \frac{3}{2}a_0$, portanto $\bar{r} \approx 0.7935$ Å, como apresentado na equação 2.10. O cálculo do custo é feito pela determinação do percentual da diferença entre os raios previsto (r_{prev}) e real (0.7935Å)

$$\text{custo} = \left| \frac{r_{prev} - 0.7935}{0.7935} \right|$$

distribuição uniforme dos ângulos (radialmente distribuído - RD)

Uma das características da densidade eletrônica (distribuição radial) é ser independente do ângulo, portanto igualmente distribuído para todos os ângulos, como apontado na Seção 2.1. Para poder verificar se a distribuição radial é igual para todos os ângulos

seria necessária realizar uma integração para todos os ângulos, o que seria complicado e custoso.

Pensando de forma inversa, podemos considerar que para cada raio a distribuição angular deve ser homogênea. Considerando esferas ocas concêntricas de dada espessura, podemos inferir a distribuição angular em cada uma delas e calcular o erro na sua distribuição. Cada nuvem de pontos foi dividida em 20 esferas ocas. Para calcular o erro foi construída a função apresentada no Algoritmo B.4 em anexo. A verificação das distribuições angulares pode ser vista como redundante a essa avaliação, mas não o inverso.

desvio padrão entre distribuições (σ)

Foi calculado o desvio padrão (eq. 5.6) entre os histogramas previsto e de referência, para as distribuições radial e angulares. A minimização do desvio minimiza a diferença entre as distribuições ao fornecer à rede um melhor caminho para aproximar as curvas.

$$\begin{aligned} \delta_i &= x_i - \bar{x}, \\ \sigma &= \sqrt{\frac{\sum_{i=1}^{n} \delta_i^2}{n}}, \end{aligned} \tag{5.6}$$

onde σ é o desvio padrão, x_i é o valor da amostra e $\bar{x}$ é a média das n amostras.

centro de massa (CM)

O centro de massa foi considerado, pois na ausência da ação de forças externas, o centro de massa deve permanecer constante. Sua avaliação F_{CM} foi simplesmente o somatório das posições de cada eixo cartesiano (convertido das coordenadas esférias se necessário). Como a quantidade negativa deve se equilibrar com a positiva em todos os eixos para um centro de massa na origem, então

$$\text{custo} = \sum_{i=1}^{N} x_i + \sum_{i=1}^{N} y_i + \sum_{i=1}^{N} z_i$$

Este "centro de massa" é realizado apenas sobre a posição relativa do elétron, isto é, não é o CM real. Contudo, como a distribuição deve estar centrada na origem, este cálculo possui equivalente resultado.

Foram construídas várias funções de custo visando verificar qual possuía melhor eficiência em apresentar o problema para a rede. Considerando uma função de custo que tenha incorporado todas as funções avaliativas listadas, apresentamos um modelo no Algoritmo B.5 em anexo – A função corresponde com a utilizada para a FRes14.

A função de custo aqui utilizada é a mesma utilizada pelo GA, e o custo é o valor otimizado pelo algoritmo. Como o GA objetiva alcançar o menor valor, o custo deve ser sempre positivo, permitindo ao treinamento minimizar o erro.

5.4 ANÁLISE

Tomando as redes treinadas (`FRes#`) na fase de testes, tomamos apenas as melhores redes para uma análise mais aprofundada. Como critério para definir quais eram as melhores redes geradas na fase de testes finais, consideramos como ideais as redes com erro inferior a 10% em todos os 4 parâmetros de avaliação: erro nos histogramas radial, angulares (φ e θ) e no raio médio.

A análise da rede treinada consistiu primeiramente da verificação se os requisitos estabelecidos foram satisfeitas. Então, avaliou-se a presença de outras características que pudessem ser exploradas, como a capacidade atratora. O *script* montado para a análise das redes treinadas consistiu dos seguintes passos:

1. *Trajetória iterada:* a partir de um ponto inicial aleatório (alg. B.4), foram realizadas 10^5 iterações das posições (alg. 5.2), 50 réplicas cada rede, e então verificamos:

   (a) *Desvio do Centro de Massa (d_{CM})* módulo da posição média dos pontos, medida de 100 em 100 pontos

   (b) *Distribuição angular uniforme (RD)* (agl.B.4)

   (c) *Histograma em r e raio médio (Hr e $\bar{r}$)* erro dos histogramas normalizados

   (d) *Histograma em θ e φ ($H\theta\varphi$)* erro dos histogramas normalizados

   (e) *Iterações a partir de pontos iniciais distintos* comparação das distribuições de pontos com diferentes pontos de partida.

2. *Atrator* gero um conjunto de pontos ordenados ou aleatórios e passo pela rede em um passo único (não iterado). Para todos os casos avalio a diferença dos histogramas radiais e angulares, e sua esfericidade.

   (a) *Dados ordenados* Um cubo e uma esfera. Entre os limites de $[-k, k]$, com $k = 5$ e um intervalo de 0.25 Å entre cada ponto, gerando um cubo com pontos igualmente espaçados. A esfera é obtida a partir do cubo, com raio igual a k.

   (b) *Dados não ordenados* gero pontos randômicos em forma cúbica e esférica. O cubo randômico

   $$cubo = (\text{rand}(N, 3) * 2 * k) - k,$$

   onde $N = 10^5$ é o tamanho amostral, e $k = 5$ é o limite superior da distribuição. Desejo, aqui, obter uma série de pontos de $[-k, k]$. A função `rand`(a, b) gera uma matriz de $a \times b$ com números randômicos entre [0,1]. Obtenho a esfera de modo

similar, limitando o raio a partir da origem. Também gero dados com tendência para o centro e para as margens, para verificar se há algum efeito na tendência.

$$c = k\text{rand}(N, 3)^t \cdot \text{sign}[\text{rand}(N, 1)2 - 1]$$

onde t corresponde ao fator de tendência. A função `sign()` fornece o sinal da matriz recebida:

$$\text{sign}(x) = \begin{cases} -1, & x < 0 \\ 0, & x = 0 \\ 1, & x > 0 \end{cases}$$

Foram utilizados os fatores de tendência $t = 0.8$ e $t = 1.2$, sabendo-se que $t < 1$ geram tendências para as margens e $t > 1$ tendências para o centro da esfera.

(c) *Fração* verifico se os dados ganham tendência para fora ou para dentro dependendo do conjunto inicial – comparação das distribuições. Gero 6 conjuntos de pontos iniciais randômicos, esferas ocas concêntricas: $F_1 = [0, 0.5]$ Å, $F_2 =]0.5, 1.0]$ Å, $F_3 =]1.0, 1.5]$ Å, $F_4 =]1.5, 2.0]$ Å, $F_5 =]2.0, 2.5]$ Åe $F_6 =]2.5, 3.0]$ Å.

5.4.1 Integração a um sistema dinâmico

Foi realizada uma simulação semiclássica para testar as redes treinadas no que diz respeito a estabilidade do sistema e para verificar a possibilidade de integração a um modelo de dinâmica molecular.

Os algoritmos para Dinâmica Molecular possuem estrutura bastante simples. Partindo do mais básico, o algoritmo é constituído por um conjunto de partículas numéricas que possuem basicamente posição, velocidade e massa. Estas partículas irão interagir por meio de potenciais (forças atuantes), como representado na Figura 5.4 (GRIEBEL; KNAPEK; ZUMBUSCH, 2007; RAPAPORT, 1995b; HOSPITAL *et al.*, 2015). Após calcular as forças atuantes sobre o sistema, é preciso definir a próxima posição das partículas no tempo. Para isso se realiza a integração das posições mediado por algoritmos de integração. Os integradores são muito importantes no que diz respeito à precisão do modelo assim como tempo computacional (GRIEBEL; KNAPEK; ZUMBUSCH, 2007; RAPAPORT, 1995b).

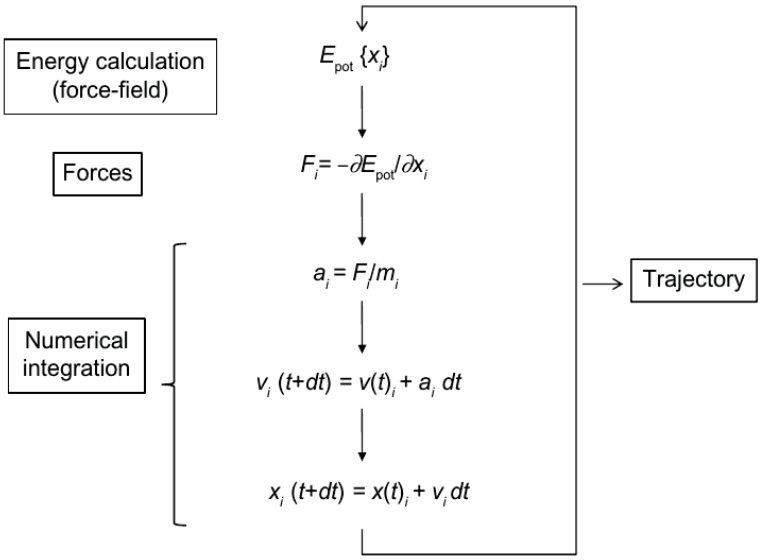
O integrador utlizado foi o Verlet [8], que é bastante veloz, e possui um erro aceitável para o presente objetivo. Este integrador pode ser descrito como,

$$\vec{x}_{t+\Delta t} = -\vec{x}_{t-\Delta t} + 2\vec{x}_t + \vec{a}\Delta t^2, \tag{5.7}$$

onde $\vec{x}$ é a posição em termos de um tempo t, com um passo de tempo Δt, e $\vec{a}$ uma aceleração. Verlet não envolve cálculo de velocidades.

[8] Verlet é um integrador de ordem 2, e a ordem do integrador é proporcionalmente inversa a seu erro.

FIGURA 5.4: ALGORITMO BÁSICO DE DINÂMICA MOLECULAR.



FONTE:Hospital *et al.* (2015).
LEGENDA: A partir da energia do sistema, calculam-se as forças atuantes sobre cada partícula do sistema, e deduz-se a aceleração. Com o valor da aceleração, define-se pela integração numérica a próxima posição da partícula após uma fração de tempo definida por dt. Neste diagrama, E_{pot} refere-se à energia potencial, t ao tempo atual, dt ao tempo de integração, i à i-ésima partícula que possui x, posição, v, velocidade, a, aceleração, F, força e m, massa, vinculados.

O átomo de um único elétron é um sistema governado pela ação atrativa entre as cargas positiva nuclear e negativa eletrônica. A força de Coulomb, ou força de cargas estáticas, descreve a interação entre as cargas (EISBERG; RESNICK, 1979, p.307-313) e é dada pela igualdade:

$$\vec{F_C} = \frac{q_1 q_2}{4\pi\varepsilon_0 r^2}\hat{r} \tag{5.8}$$

onde as cargas q_1 e q_2 correspondem à carga das partículas interagentes, o núcleo (que possui carga $+e$, por possuir apenas um próton) e o elétron (carga $-e$). Cada carga corresponde a $e = 1.60 \cdot 10^{-19}$ Coulomb. Existem outras forças atuantes na interação elétron-núcleo que serão desconsideradas para simplificar esta breve demonstração.

Integramos a posição do núcleo no tempo decorrente da interação Coulombiana com o elétron orbitante. As posições do elétron foram fornecidas pela rede por meio de iteração. Um núcleo com posição inicial (x_0) na origem, utilizando um passo de tempo de 10^{-14} s. A aceleração foi deduzida de F_C,

$$\vec{a} = \frac{F_C}{m_n}\hat{r}$$

onde m_n é a massa do núcleo e $\hat{r}$ é o versor. Para cada passo de integração do núcleo, foram gerados 100 posições do elétrons em iteração, e a sua posição média foi utilizada para se obter

a força atuante sobre o núcleo, representando o efeito da nuvem eletrônica sobre o núcleo no decorrer do tempo.

Foi utilizado o padrão SI para as unidades. Todas as constantes foram obtidas das referências Mohr, Newell e Taylor (2016) e Linstrom e Mallard (2020).

6 RESULTADOS

Como primeiro resultado, em vista das dificuldades em se encontrar um modelo de aprendizado global sem supervisão local, tivemos o desenvolvimento do novo modelo de rede denominado *Feature Resonance Neural Network* (FRes), já apresentado na Seção 4.3, em conjunto com a proposta.

Após o desenvolvimento do novo modelo de rede, foi realizada uma série de testes para sua validação. Inferindo sua capacidade de aprendizado, inicialmente propomos problemas simples: o problema do ou exclusivo (XOR) na Subseção 6.1.1, e o de classificação na Subseção 6.1.2. Um teste adicional foi realizado com relação à capacidade da rede aprender comportamentos com base em distribuições globais, aplicado-a a um problema de geração de dados fictícios, como apresentado na Seção 6.2.

Finalizada a fase de testes, e atestada a capacidade da rede de aprender comportamentos globais, a rede foi implementada para o problema objetivo (Seção 6.3), visando com que a rede aprendesse o comportamento do elétron ligado a um núcleo em seu estado fundamental. Por fim, são realizadas avaliações sobre as redes ensinadas que apresentaram melhores resultados, na Seção 6.4.

6.1 VALIDAÇÃO DA REDE

Dada a arquitetura da FRes, foi necessário verificar sua funcionalidade. O problema do XOR permitiu demonstrar que a ressonância é essencial na solução de problemas não lineares. Os testes seguintes permitiram validar a FRes para executar classificação, o que demonstra sua qualidade como rede neural. O gerador de instâncias mostrou que a rede é capaz de otimizar funções de custo complexas e realizar aprendizado global não supervisionado.

6.1.1 O ‘OU exclusivo’ (XOR)

Este problema nos foi útil para avaliar as diferenças da rede com e sem ressonância. O treinamento da rede para classificação demonstrou que a rede sem ressonância não consegue solucionar o problema do XOR, alcançando uma acurácia de 75% (acertou 3 dos 4 padrões apresentados) para todas as 1000 redes treinadas. Por outro lado, a rede com ressonância atingiu os 100% de acertos na classificação em 54.4% dos casos, não obtendo sucesso nos demais 45.6% dos casos. A rede com ressonância conseguiu aprender o padrão XOR apresentado (detalhe dos parâmetros na Subseção 5.2.1).

As outras redes treinadas utilizando o dobro da população ($n_{pop} = 100$ indivíduos) obtiveram melhora em sua performance alcançando 100% de acertos na classificação em 80.6%

TABELA 6.1: COMPARAÇÃO DA TAXA DE ACERTOS PARA O PROBLEMA XOR ENTRE REDES COM E SEM RESSONÂNCIA.

		Taxa de acertos (%)			
r_{eson}	n_{pop}	2/4	3/4	4/4	$\bar{t}$ (s)
0	50	0	100	0	0,7073
1	50	14,9	30,7	54,4	1,4293
1	100	2,8	16,6	80,6	2,5235

FONTE: O autor (2020).
LEGENDA: A tabela torna clara a incapacidade da rede sem ressonância de aprender o padrão XOR apresentado. Também demonstra que o ajuste de parâmetros pode melhorar o desempenho da rede em aprender este padrão. $\bar{t}$ remete ao tempo médio dispendido no treinamento das redes.

dos treinos. A Tabela 6.1 mostra comparativamente o resultado obtido entre as redes com ressonância e a sem ressonância.

Portanto, nenhuma das redes sem ressonância conseguiu realizar a classificação correta para os 4 estados, enquanto que a maioria dos treinamentos da rede com ressonância obtiveram sucesso. Esses resultados também indicam que o ajuste de parâmetros, como o aumento do tamanho da população da GA, pode melhorar o desempenho da rede.

6.1.2 Classificação

Os treinos para a classificação dos dados da *Iris* (FISHER, 1936; DUA; GRAFF, 2017) e do Câncer de Mama (WOLBERG; MANGASARIAN, 1990; DUA; GRAFF, 2017) permitiram averiguar e confirmar a capacidade da rede.

A rede tipo MLP obteve um sucesso médio de acertos de $(94.93 \pm 4.83)\%$ para a *Iris* e de $(95.87 \pm 1.61)\%$ para o Câncer de Mama. Utilizando a nossa rede, obtivemos o melhor resultado com a rede sem ressonância (FResC pela Tabela 6.2), obtendo acurácia média de $(93.99 \pm 2.96)\%$ a *Iris* e de $(96.69 \pm 1.12)\%$ para o Câncer de Mama para a melhor configuração, com um amostral de 1000 réplicas cada. A rede sem ressonância (FResC) obteve valores equiparáveis às da MLP, validando a rede para este fim.

A Tabela 6.2 apresenta a relação completa de todos os parâmetros e o respectivo tempo de treinamento e acurácia obtidos. Os resultados de acurácia foram estatisticamente iguais para todas as redes, entretanto a rede sem ressonância foi a que obteve menor desvio. A Figura A.1 em anexo mostra a frequência das acurácias obtida por cada uma das 4 configurações de rede, permitindo observar melhor a consistência dos dados.

A rede sem ressonância obteve melhor performance que em relação às com ressonância. Apesar das redes com ressonância possuírem maior capacidade de aprendizado, por permitir aprendizado não linear, elas possuem três vezes mais pesos a otimizar que a rede sem ressonância, e portanto maior dificuldade em alcançar os mesmos resultados, com os mesmos hiperparâmetros.

TABELA 6.2: PARÂMETROS E ACURÁCIA OBTIDOS PARA OS TESTES DE CLASSIFICAÇÃO.

Iris Dataset							
	camadas	r_{eson}	n_{iter}	n_{pop}	n_{ger}	t (s)	Acurácia (%)
MLP	[3 2]	–	–	–	–	0,37	94,93 ± 4,83
FResA	[3 2]	1	10	100	25	11,69	89,09 ± 10,62
FResB	[3 2]	1	5	500	10	11,61	90,19 ± 14,12
FResC	[3 2]	0	10	100	25	3,14	93,99 ± 2,96

Breast Cancer Dataset							
	camadas	r_{eson}	n_{iter}	n_{pop}	n_{ger}	t (s)	Acurácia (%)
MLP	[5 2]	–	–	–	–	0,39	95,87 ± 1,61
FResA	[5 2]	1	10	100	25	47,93	93,97 ± 2,24
FResB	[5 2]	1	5	500	10	40,06	91,98 ± 11,43
FResC	[5 2]	0	10	100	25	3,89	96,69 ± 1,12

Abreviações: r_{eson} - presença de ressonância, n_{iter} - número de iterações da rede, n_{ger} - número de gerações da GA, n_{pop} - tamanho da população da GA, t – tempo de treino.

FONTE: O autor (2020).
LEGENDA: Utilizando 2 conjuntos de dados, comparamos a acurácia, e seu respectivo desvio padrão, entre uma MLP comum e 3 configurações diferentes da rede. Em todos os casos, as redes obtiveram acurácias que se sobrepõem. Dentre as 3 configurações, para classificação, a sem ressonância (FResC) obteve melhor resultado, com valores equiparáveis às da MLP.

A rede MLP teve um tempo de treinamento cerca de 10 vezes menor que a FRes sem ressonância, e a FRes sem ressonância obteve um tempo 10 vezes menor que a com ressonância, aproximadamente. Isso se justifica pelo fato da MLP utilizar o algoritmo *Levenberg-Marquardt*, enquanto que nossa rede utiliza GA, um algoritmo de qualidade mas que dispende muito mais tempo por utilizar otimização de uma população. A ressonância também envolve uma quantidade muito maior de cálculos por possuir maior número de pesos a otimizar.

Verificada a funcionalidade da rede como tal, podemos avançar e explorar a adaptabilidade da rede para funções de custo mais complexas e não supervisionadas.

6.2 GERADOR DE INPUT (*AGUMENTATION*)

A ideia de se gerar informação visando aumentar o *pool* de dados para algum fim, como a classificação, não é nova (WANG; PEREZ, 2017; GUPTA, 2019). Esta técnica vem se mostrando bastante útil no que diz respeito a redução de overfitting[1] Alguns usos anteriores

[1]Overfitting é um fenômeno que ocorre quando uma rede é treinada em excesso, isto é, quando ela perde a capacidade de generalizar para dados similares ao que ela foi treinada, se tornando especialista com relação a seus dados de treinamento (HAYKIN, 2008a).

TABELA 6.3: ACURÁCIA DA CLASSIFICAÇÃO SOBRE OS DADOS GERADOS.

#	Ac. média Treino (%)	Ac. média Teste (%)
1	82,63 ± 12,88	87,75 ± 6,90
2	96,75 ± 2,38	95,68 ± 3,37
3	97,24 ± 1,82	92,70 ± 0,89
4	92,77 ± 2,85	89,74 ± 3,52
5	94,41 ± 1,86	96,32 ± 0,84

FONTE: O autor (2020).
LEGENDA: Para cada uma das cinco réplicas de redes geradoras, foram treinadas 100 MLPs classificadoras, e calculada a acurácia média (Ac. média) para os dados originais do grupo de treino e teste. Com divisão dos dados 1:1, o melhor resultado obtido para o grupo teste foram as redes da réplica #5.

TABELA 6.4: ERROS DOS DADOS GERADOS PELAS REDES.

Classe	E.C. (%)	Erro Perc. da Média (%)				Erro Perc. do Desvio (%)			
		C.S.	L.S.	C.P.	L.P.	C.S.	L.S.	C.P.	L.P.
I. setosa	67.57	0,03	2,26	0,76	11,85	23,66	32,28	24,54	27,09
I. versicolor	29.43	0,09	1,06	1,26	2,92	20,52	45,08	32,85	33,54
I. virginica	31.29	2,37	4,82	2,22	0,14	40,88	30,44	34,37	6,23

Abreviações: E.C.: erro percentual das correlações, C.S - comprimento da sépala, C.P - comprimento da pétala, L.S - largura da sépala, L.P - largura da pétala.

FONTE: O autor (2020).
LEGENDA: Utilizando as redes geradoras treinadas da réplica #5 (Tabela 6.3), foram calculados os erros percentuais entre as correlações, médias e desvios de cada atributos entre os dados originais completos e os gerados em apenas um passo (sem iterações).

foram feitos com redes GAN (WANG; PEREZ, 2017; GUPTA, 2019), já que possuem como característica inerente a presença de uma rede geradora de dados, como abordado na Seção 3.4.

Havido treinado 5 réplicas de redes FRes geradoras, e treinado as 100 réplicas de MLP classificadoras, obtivemos as acurácias médias, apresentadas na Tabela 6.3. A quinta réplica (#5) foi a que obteve melhor resultado com (96.32 ± 0.84)% de acurácia no grupo teste. Esta rede, utilizando proporção 50:50, obteve resultado estatisticamente igual ao treino de classificação com MLP realizado com o conjunto original, e proporção 70:30, apresentado na Seção anterior (Tabela 6.2).

Do treinamento das redes FRes geradoras, obtivemos uma distribuição dos atributos para cada classe. Na Figura A.2 em anexo, podemos comparar a distribuição real total dos 4 atributos de cada indivíduo para as 3 classes e a distribuição produzida pela rede em um único

TABELA 6.5: COMPARAÇÃO DE PARÂMETROS DE TREINO PARA DISTRIBUIÇÃO RADIAL.

	Parâmetros				Valores médios			Melhor resultado	
Teste	r_{eson}	$\bar{r}$	b_{ias}	n_{input}	$\bar{t}$ (min)	Hr (%)	$\bar{r}$ (%)	Hr (%)	$\bar{r}$ (%)
FRes01A	0	1	1	1	10,90	91,19 ± 0,37	0,90 ± 0,09	91,11	0,69
FRes02A	1	1	1	1	25,35	24,59 ± 6,82	3,51 ± 3,43	11,66	1,93
FRes02B	1	1	1	1	1,97	32,81 ± 21,95	10,93 ± 25,36	10,49	0,41
FRes03A	1	0	1	1	25,43	25,96 ± 6,83	3,82 ± 5,01	12,68	0,47
FRes03B	1	0	1	1	2,05	35,74 ± 25,89	9,6 ± 210,8	13,15	2,70
FRes04A	1	1	0	1	29,02	25,51 ± 7,39	2,46 ± 2,55	11,96	6,08
FRes04B	1	1	0	1	1,90	35,25 ± 24,96	16,83 ± 66,69	9,74	1,24
FRes08A	1	1	1	2	34,89	21,52 ± 6,11	3,48 ± 4,15	10,83	1,09
FRes08B	1	1	1	2	3,41	27,83 ± 20,95	6,06 ± 9,84	7,13	0,85

Abreviações: $\bar{t}$ - tempo médio, $\bar{r}$ - raio médio (3/2 do raio de Bohr), e seu respectivo erro, Hr - Erro percental médio do histograma, $\bar{r}$ - Erro percental médio do raio médio

FONTE: O autor (2020).
LEGENDA: Utilizando 50 réplicas cada. Os testes 'A' utilizam poucos pontos (15) com muitas iterações (100) e avaliação individualizada, enquanto que os 'B' utilizam muitos pontos (200), poucas iterações (5) e avaliação comum. Para todos os testes, foi utilizada uma camada de entrada, uma de saída e [6 6] ocultas. A estatística é realizada sobre uma iteração de $t_{sim} = 10^4$.

passo (sem iteração). As redes utilizadas foram da réplica #5, com melhores resultados. Os erros entre as distribuições foram, em média, de (16.50 ± 10.17)%.

Sobre esses dados gerados, foram avaliados os parâmetros o valor médio, o desvio padrão e as correlações, como apresentado na Tabela 6.4, entre os valores previstos e os dados originais completos da *Iris*. Com exceção das médias, que obtiveram valores muito similares aos originais, as correlações e desvios tiveram diferença de (29.29 ± 9.65)% em média. É curioso que, mesmo com diferenças consideráveis entre os dados originais e os gerados, tenha sido possível obter uma acurácia de 96% na classificação do grupo teste.

Levando em conta os histogramas da Figura A.2 em anexo com os erros percentuais da Tabela 6.4, temos um indicativo de que os fatores mais relevantes são centro da distribuição e a coincidência das "fronteiras", de forma a conseguir separar os grupos entre si. Contudo não podemos generalizar esta conclusão.

Este gerador de instâncias é uma tentativa mais rebuscada de avaliação da rede. Demonstramos que, baseado em distribuições conhecidas, é possível obter dados similares aos originais utilizando-se de uma função de custo adequada. E mais importante, conseguimos fazer isso utilizando apenas supervisão global. Baseado nesse primeiro resultado, podemos também utilizar treinamento baseado em distribuições para obter a densidade de probabilidade de Schrödinger, convertendo-a em um histograma de maneira similar.

6.3 APLICAÇÃO A MODELAGEM DO ÁTOMO DE HIDROGÊNIO

A rede se mostrou eficiente para a geração de dados, como apresentado na Seção 6.2, utilizando-se de histogramas de distribuição. Utilizamos, então, deste mesmo formato de treinamento para o elétron no átomo de hidrogênio. Da mesma forma que o gerador de *Iris* não possui ordem ou saída esperada (supervisão), não se conhece a trajetória do elétron, mas sabemos como eles devem se comportar. Obviamente, o movimento do elétron é bem mais complexo que a distribuição dos dados da Iris, o que requer uma rede e função de custo mais complexas.

Verificamos a capacidade da rede de obter como saída uma distribuição de coordenadas que obedecesse a curva da densidade de probabilidade de Schrödinger para o nível fundamental do átomo de hidrogênio. Primeiramente, consideramos *apenas a distribuição radial*, isto é, ignoramos as distribuições angulares θ e φ. Depois, o conjunto completo de informações foi considerado e, sobre os melhores modelos treinados, foram realizadas avaliações.

6.3.1 Testes Preliminares

Foram realizados 7 testes preliminares, utilizando as funções de custo detalhadas no Algoritmo B.3.

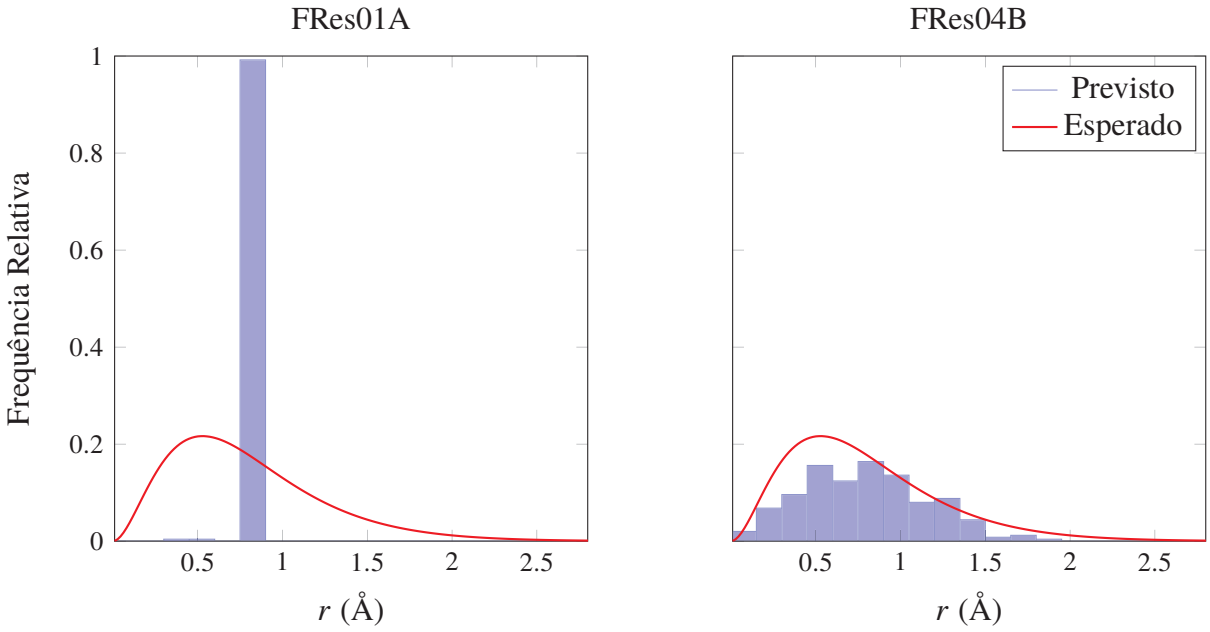
Parâmetros

Analisando os dados obtidos nos testes apresentados na Tabela 6.5, podemos perceber que o uso de ressonância é totalmente necessário, uma vez que não se consegue alcançar a distribuição (nem de longe) deixando de utilizar a ressonância (FRes01). Como apresentado na Seção 6.1 de validação da rede, a rede apesar de ser mais eficiente para solucionar problemas simples como os de classificação apresentados, problemas não lineares como o XOR não são possíveis de resolver sem a ressonância. A Figura 6.1 traz a comparação dos resultados de um treinamento com e sem ressonância, dos testes denominados FRes01 e FRes04 (Tabela 6.5), demonstrando que o problema do comportamento do elétron não é linear. Tendo sido observada a incapacidade de aprendizado por redes sem ressonância, todos os treinamentos posteriores foram realizados com ressonância.

A presença de bias é totalmente opcional, pois os resultados de FRes02 e FRes04 são estatisticamente iguais, inclusive em seu desvio. A presença do raio de médio ($\bar{r}$) pouco influencia no resultado, o que pode ser explicado pelo fato da distribuição já garantir o raio médio próximo ao de Bohr.

A função de custo tipo B certamente é muito mais rápida, uma vez que possui menor número de iterações. Contudo, tanto o erro quanto o desvio médio em todos os casos foi superior ao da função A. O desvio foi 20% superior em todos os casos, indicando ser um treinamento instável e menos confiável. Apesar isso, em muitos casos a melhor rede gerada pela função B foi melhor que a gerada pela função A.

FIGURA 6.1: COMPARAÇÃO DO RESULTADO DO USO DE RESSONÂNCIA.



FONTE: O autor (2020).
LEGENDA: À esquerda temos a melhor rede de FRes01A, sem ressonância, que não obteve a distribuição esperada. À direita o resultado da iteração da melhor rede de FRes04B com ressonância, que alcançou resultado próximo à curva esperada. Foram utilizados 1000 pontos em cada iteração da trajetória (detalhes tab. 6.5).

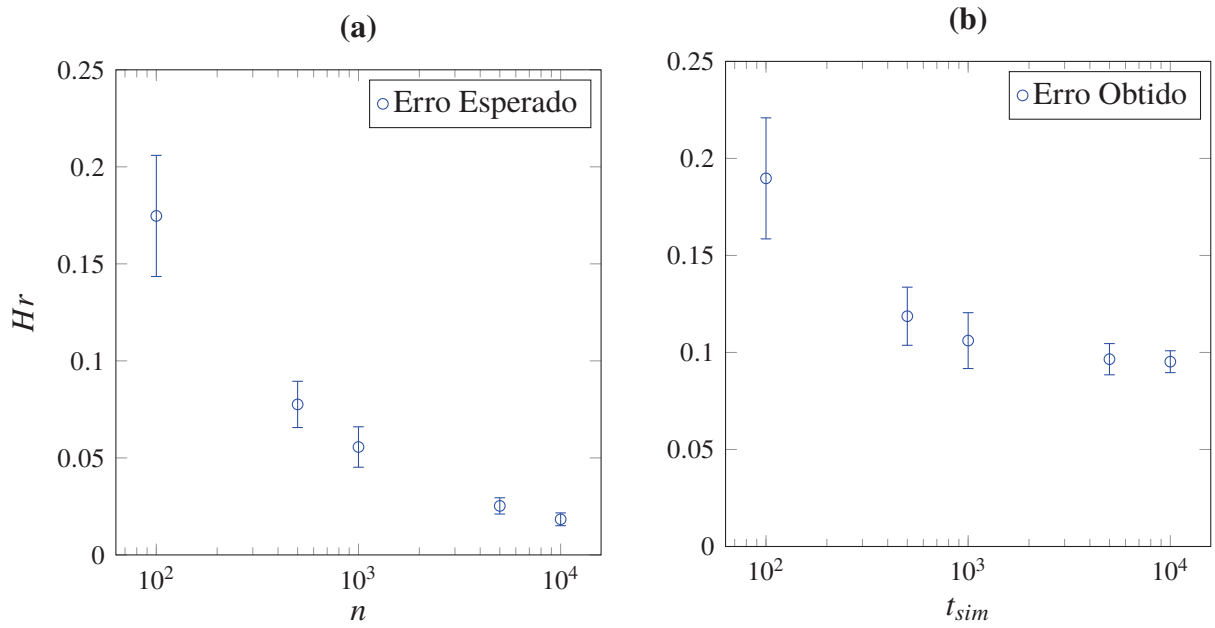
Chegamos a um impasse: a função de custo A é mais estável com melhores resultados estatisticamente, mas tem um tempo de treinamento maior. Enquanto que a função de custo B é estatisticamente inferior, mas com um tempo muito inferior de treinamento permite realizar muito mais treinos, dentre os quais um pode resultar em uma rede com baixo erro. Por este motivo seguimos utilizando ambos modelos para treinos posteriores.

Quando utilizamos histogramas para verificar a distribuição coincidente entre dois conjuntos de pontos, devemos esperar um erro médio. Quando se geram pontos utilizando Monte Carlo e compara seus histogramas, haverá um erro relacionado a randomicidade dos pontos. Conforme mais pontos são gerados, menor o erro em relação a um histograma de referência com vários pontos (10^5 pontos). A Figura 6.2(a) mostra um comparativo entre o número de pontos gerados e seu erro. O mesmo pode ser pensado com relação ao número de iterações da rede. Quanto mais iterações, mais confiável a curva de distribuição, como na Figura 6.2(b), utilizando a melhor rede do teste FRes04B (tab. 6.5).

Camadas

Os testes do número de camadas nos foi norteador. Observando a Figura 6.3, para o teste unidimensional, é possível perceber que a função tipo B possui real instabilidade, com um desvio grande para quaisquer combinações de camadas e nós. Na função tipo A, com desvio menor, é

FIGURA 6.2: ERRO MÉDIO ESPERADO BASEADO EM t_{sim}.

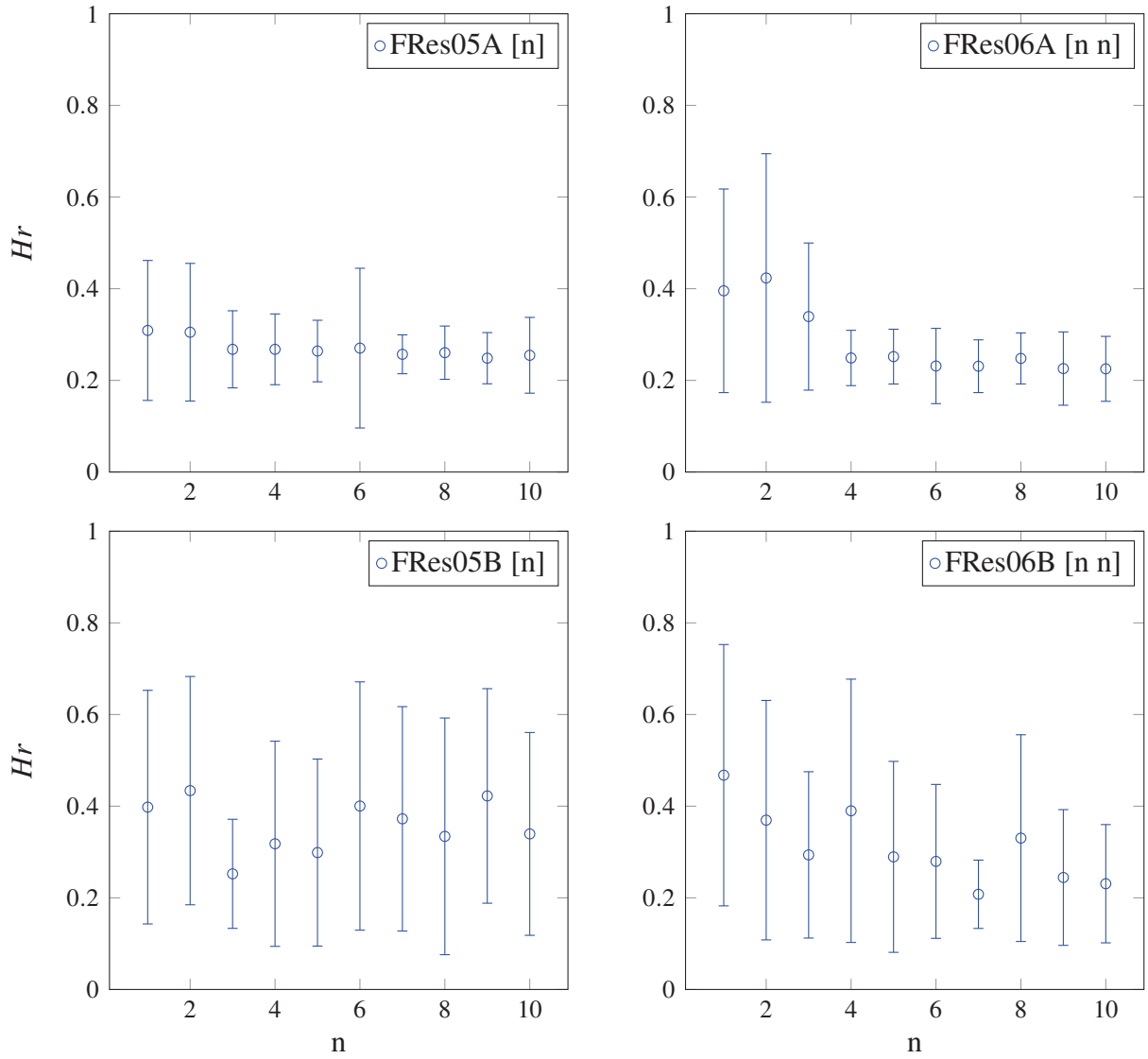


FONTE: O autor (2020).
LEGENDA: **(a)** dados gerados por Monte Carlo, obtemos uma frequência de pontos consistente com a distribuição alvo (densidade de probabilidade de Schrödinger). Quando realizamos um comparativo (erro entre histogramas) entre uma população de 10^5 pontos de referência contra um amostral de n pontos, obtemos um erro entre os histogramas, devido à aleatoriedade. Conforme aumentamos o amostral de pontos, menor o erro aparente. **(b)** dados gerados por iteração da melhor rede de FRes04B (tab. 6.5) foram calculados os erros entre histogramas para um t_{sim} variado, demonstrando que a maneira como foi construída a função de custo permite alcançar a bacia de atração com poucos pontos. Para cada ponto no gráfico, foram realizadas 100 réplicas. Hr – Erro médio entre os histogramas radiais previsto e de referência.

possível observar que o número de nós para camada única (FRes05A) pouco interfere no erro médio dos histogramas radiais. Para camada única, quando a camada oculta possui apenas 1 ou 2 nós, o desvio foi maior, o que anormalmente também ocorreu com 6 nós. Para a camada dupla, as camadas de menor número de nós tiveram maior erro e desvio, que decresceu com o aumento do número de nós, se estabilizando em 4.

Esperávamos obter um padrão decrescente mais evidente de erro conforme aumentasse o número de nós e camadas, pois mais pesos permitem aprender melhor o padrão apresentado. E um aumento do erro conforme o número de nós e camadas aumentasse muito, uma vez que se tornaria mais difícil de otimizar um número muito grande de pesos. Provavelmente esse efeito não foi observado pois o número de nós não foi grande o suficiente e devido ao fato de utilizarmos apenas um ponto de saída.

FIGURA 6.3: COMPARAÇÃO ENTRE ACURÁCIA E NÚMERO DE NÓS NAS CAMADAS EM UMA DIMENSÃO.

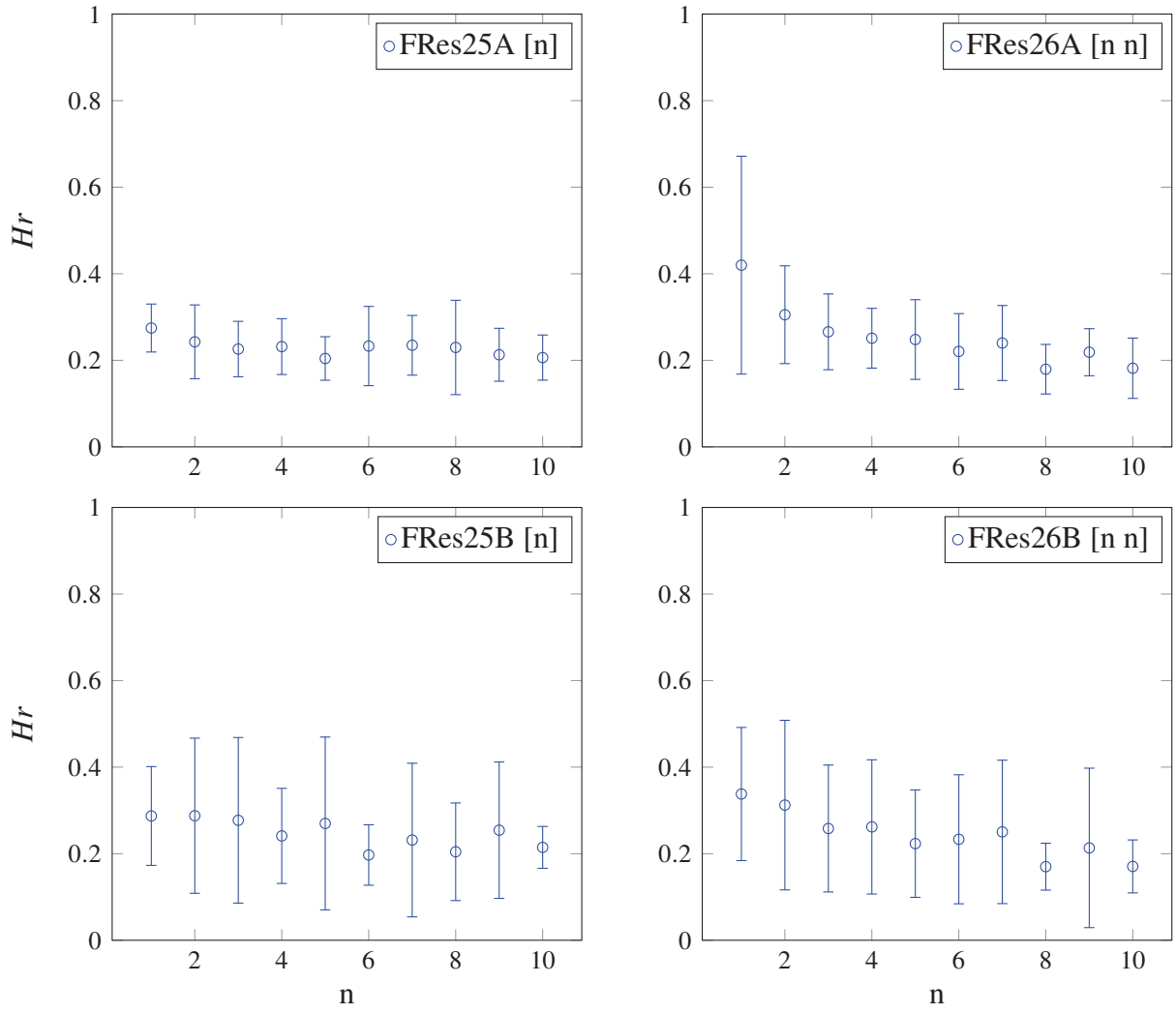


FONTE: O autor (2020).

LEGENDA: Onde *n* corresponde ao número de neurônios por camada, (a) com camada única (padrão [n]), e (b) duas camadas com igual número de neurônios por camada (padrão [n n]). Com 20 réplicas para cada *n*. Nos testes FRes05 e FRes06 *n* corresponde ao número de nós ocultos que vai de 1 a 20, com 20 réplicas cada. Os testes 'A' utilizam poucos pontos (15) com muitas iterações (100) e avaliação individualizada, enquanto que os 'B' utilizam muitos pontos (200), poucas iterações (5) e avaliação comum. *Hr* – Erro Médio Histograma Radial

Agora, observando a Figura 6.4, para o teste tridimensional, essa queda se torna mais evidente. O desvio padrão para o teste utilizando `FCustoB` foi menor do que o observado utilizando apenas uma dimensão, mas ainda maior que a utilizando a `FCustoA`. Para os 4 casos, houve queda sutil no erro conforme aumentava o número de nós no FRes25 usando camada oculta única, e queda mais evidente utilizando duas camadas ocultas em FRes26.

FIGURA 6.4: COMPARAÇÃO ENTRE ACURÁCIA E NÚMERO DE NÓS NAS CAMADAS EM TRÊS DIMENSÕES.

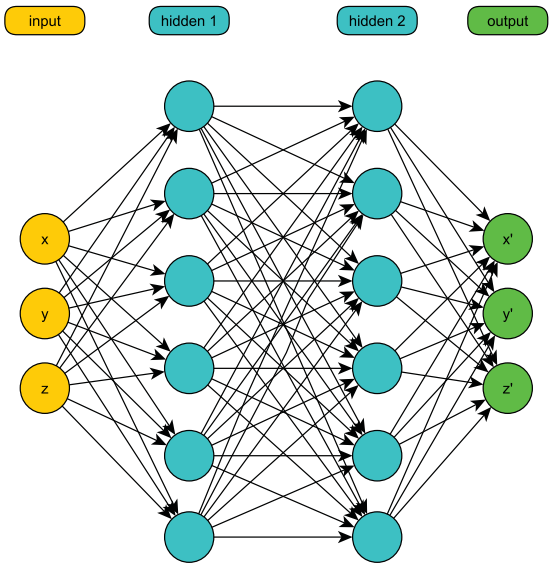


FONTE: O autor (2020).

LEGENDA: Onde n corresponde ao número de neurônios por camada, (a) com camada única (padrão [n]), e (b) duas camadas com igual número de neurônios por camada (padrão [n n]). Com 20 réplicas para cada n. Nos testes FRes25 e FRes26 n corresponde ao número de nós ocultos que vai de 1 a 20, com 20 réplicas cada. Os testes 'A' utilizam poucos pontos (15) com muitas iterações (100) e avaliação individualizada, enquanto que os 'B' utilizam muitos pontos (200), poucas iterações (5) e avaliação comum. Hr – Erro Médio Histograma Radial.

O uso de 2 camadas parece ser mais proveitoso, pois a melhoria é evidente com o aumento do número de nós. Como a queda a partir de $n = 6$ é sutil, optamos por escolher as camadas [6 6] e [8 8] para os próximos testes tridimensionais. A Figura 6.5 apresenta a configuração correspondente a [3 6 6 3] utilizada para as próximas fases de treinamento, como exemplificação.

FIGURA 6.5: ESTRUTURA DA FRES USADA PARA TREINO.



FONTE: O autor (2020).
LEGENDA: A rede possui o formato básico 3 *inputs* (x, y, z) e 3 *outputs* (x', y', z'), com 2 camadas ocultas com 6 neurônios cada, correspondente a [3 6 6 3].

6.3.2 Testes Finais

Foram realizados 6 testes finais para definir qual o melhor descritor e o melhor conjunto de avaliações, utilizando a disposição das camadas ocultas [6 6] e [8 8] com bons resultados (Figura 6.4).

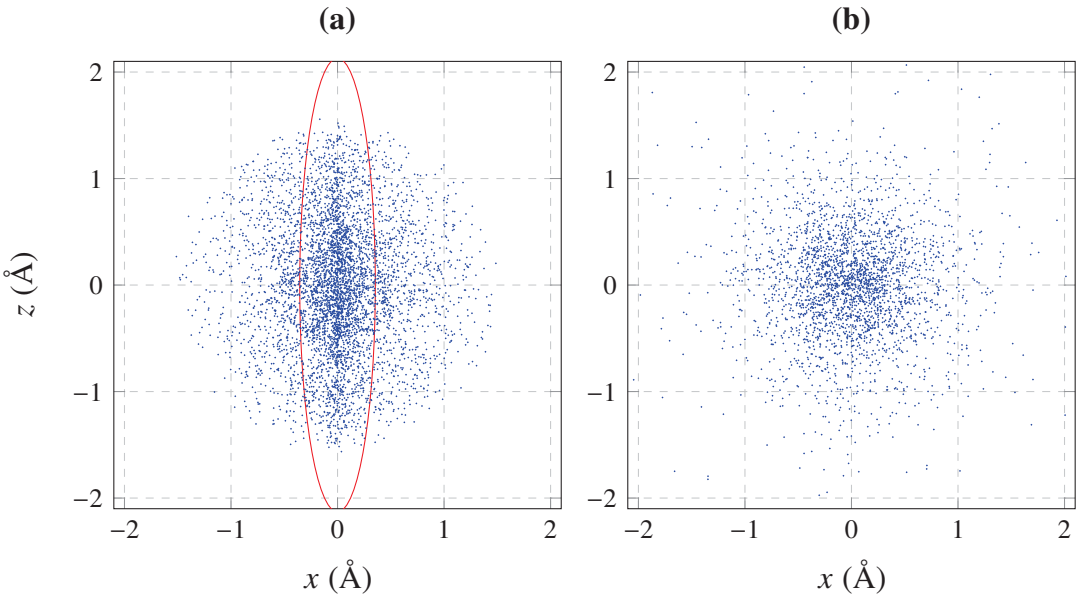
Geração de dados

Apesar de inicialmente parecer bastante simples a ideia de ter uma distribuição aleatória de ângulos, isso se mostrou verdadeiro apenas para o ângulo θ azimutal. A elevação φ foi um tanto mais problemática e exigiu mudanças na forma de se gerar os pontos iniciais e de se avaliar. Utilizamos inicialmente o método abaixo para gerar a distribuição angular,

$$\begin{aligned} \theta &= (rand(0,1) * 2\pi) - \pi, \\ \varphi &= (rand(0,1) * \pi) - \frac{\pi}{2}. \end{aligned} \tag{6.1}$$

A distribuição aleatória de θ é simples, linear, igualmente distribuída. Entretanto, se φ for gerado da mesma forma acarreta uma tendência sobre o eixo z, como na Figura 6.6(a). Devido a este problema, os ângulos não foram mais gerados diretamente em coordenadas esféricas, mas sim criando um cubo randômico em coordenadas cartesianas, garantindo a aleatoriedade dos pontos, e então convertendo para coordenadas esféricas, como apresentado no Algoritmo B.4. Com isso,

FIGURA 6.6: TENDÊNCIA SOBRE O EIXO Z.



FONTE: O autor (2020).
LEGENDA: O *plot* da posição do elétron em iteração (a) mostra uma tendência sobre o eixo z ao gerar os pontos diretamente em coordenadas esféricas. Em (b) observa-se a ausência de tendência sobre o eixo z com a implementação do Algoritmo B.4.

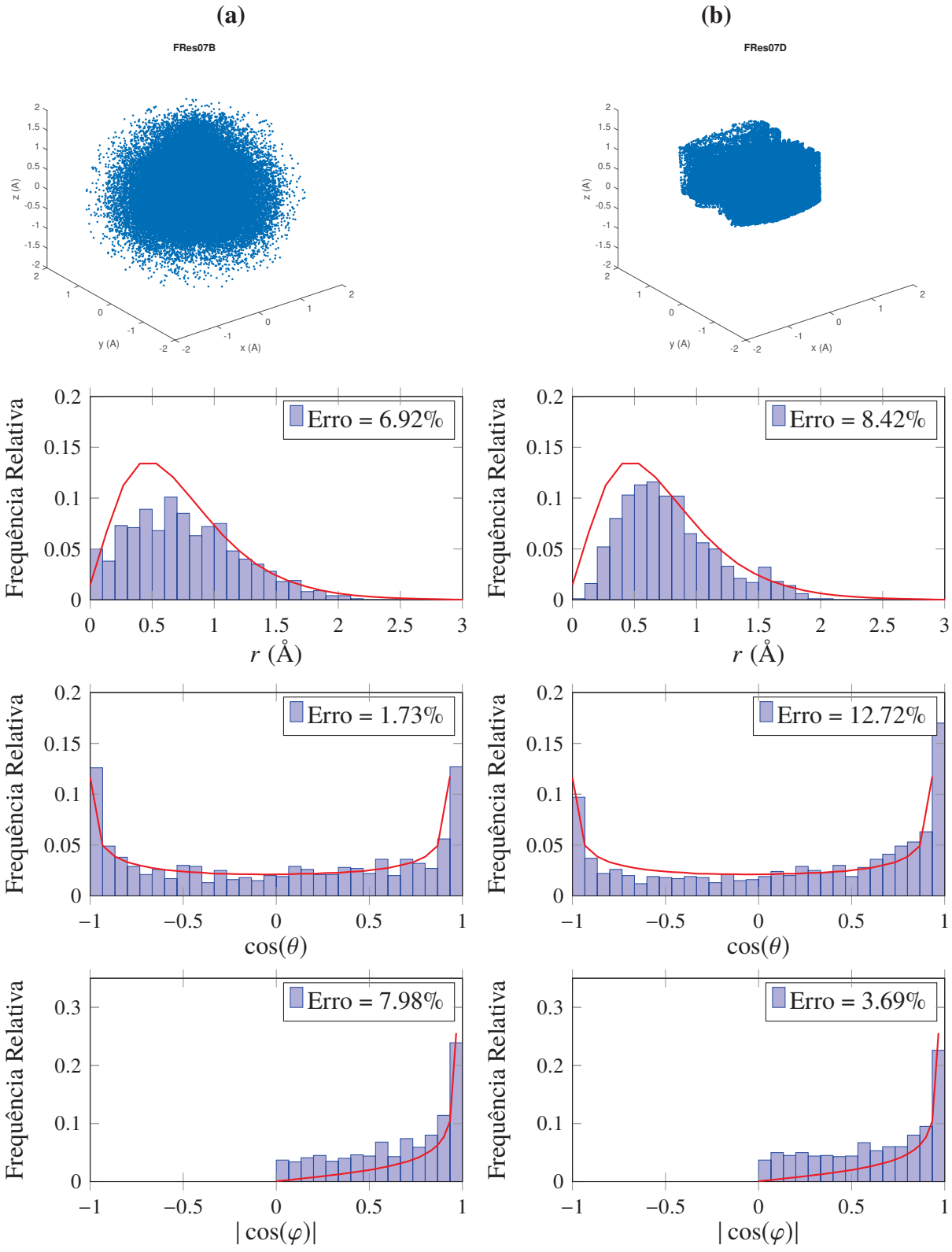
conseguimos obter uma distribuição sem tendência sobre o eixo z (Figura 6.6 (b)). Este mesmo princípio foi utilizado para gerar a distribuição φ de referência para a função de custo.

Formato de Entrada

A Figura 6.7 torna evidente o motivo de utilizarmos apenas coordenadas esféricas nos próximos passos de teste. Comparamos o melhor resultado fornecido pelos testes FRes07B e FRes07D, o primeiro em coordenadas esféricas e o segundo em cartesianas.

Neste teste utilizamos 3 coordenadas de entradas e 3 de saída, e verificamos que quando usamos coordenadas cartesianas há uma tendência em aparecer padrões ondulatórios visíveis. Como a rede é baseada em senóides, a saída também ganha esse padrão, que é perdido quando convertemos as coordenadas esféricas para cartesianas, mas não quando a saída é diretamente cartesiana. O padrão, na verdade, faz com que apesar de respeitar as distribuições angulares, o resultado seja na verdade quadrada (Fig. 6.7(b)). O uso de coordenadas cartesianas não ofereceu vantagem, como apresentado na Tabela 6.6, comparando os resultados de FRes07A com FRes07C, e FRes07B com FRes07D – lembrando que os pares de funções de custo A-C e B-D são iguais em conceito, variando apenas que em C e D são realizadas conversões das coordenadas cartesianas para esféricas para a realização das avaliações.

FIGURA 6.7: COMPARAÇÃO DOS RESULTADOS DE ENTRADA CARTESIANA E ESFÉRICA.



FONTE: O autor (2020).

LEGENDA: **(a)** coordenadas esféricas (FRes07B) **(b)** coordenadas cartesianas (FRes07D). Nota-se que em (a) o resultado é a obtenção de uma distribuição esférica, enquanto que em (b), mesmo obtendo resultados razoáveis quanto a distribuição radial e angulares, o resultado foi algo quadrangular.

Função de Custo

Agrupamos os resultados dessa fase final de testes na Tabela 6.6. Agora, com 3 coordenadas, focamos a análise em 5 erros: distribuição radial, distribuições angulares de θ e φ, no raio médio e no centro de massa.

Um cuidado que foi tomado, que pode parecer bastante óbvio, é o de que a rede não aprende se não for ensinada. Como descrito nos métodos, utilizamos o Algoritmo B.4 "radialmente distribuído" para garantir que a distribuição radial fosse igual para todos os ângulos. Esta avaliação foi inclusa em todos os testes (tab. 6.6) pois quando não era incluso acarretava no surgimento de padrões secundários e na não homogeneidade da distribuição.

Comparando diretamente FRes07 e FRes09 (tab. 6.6), nota-se que FRes09 tem quase todos os percentuais médios menores. É notável que o aumento do tamanho da população, n_{pop}, impacta diretamente na capacidade do GA em encontrar os mínimos locais. Como consequência, observamos que justamente os 3 testes realizados com maior tamanho populacional, FRes09, FRes12 e FRes14, foram os testes que geraram os melhores resultados médios, indicando que um grande tamanho da população é imprescindível para melhor explorar o espaço da função. Observe que para `FCustoA` utilizamos $n_{pop} = 250$ e para `FCustoB` $n_{pop} = 500$, isso se deve ao fato de que `FCustoB` é mais veloz, e aproveitamos essa vantagem para aumentar ainda mais o tamanho da população.

Observado a Tabela 6.6, notamos que os melhores valores médios (negrito) se concentram no FRes12, que possui maior n_{pop} e todas as funções avaliatativas inclusas, exceto o CM. Já em FRes14, idêntico ao FRes12, mas com CM incluso, possui valores médios do erro maiores que FRes12. Com isso podemos deduzir que a avaliação de CM melhora certamente o resultado do desvio do CM, contudo acaba por tornar a otimização mais complicada e piorando os demais parâmetros.

Apesar desta desvantagem, a inclusão de um centro de massa foi necessária. Ao treinar as redes, observou-se uma tendência do centro da distribuição para fora da origem, mesmo com uma distribuição homogênea de ângulos (fig.6.8). Com a intenção de posterior aplicação da rede em dinâmica molecular clássica, o deslocamento do centro de massa gera um efeito indesejado similar ao de uma força externa, puxando o núcleo de sua posição. Ainda, apesar dessa dificuldade, o treino FRes14 forneceu tantas redes quanto a FRes12 com percentual inferior a 10% de erro, classificadas como "melhores redes".

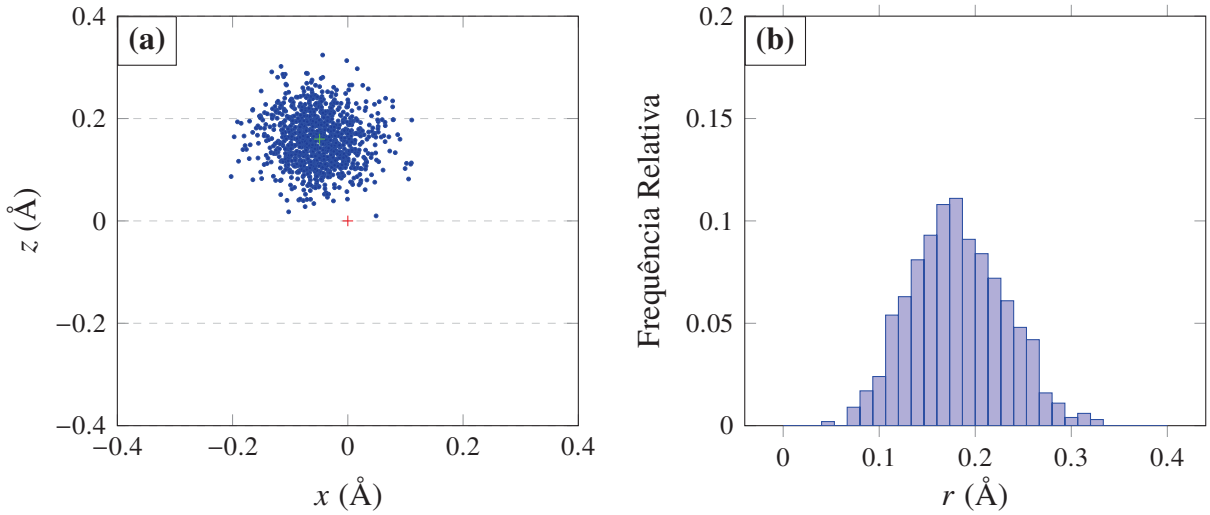
Na maioria dos casos, a melhor rede de cada teste (e quase todas as melhores redes treinadas, como resumido na Tabela 6.7) pertencia ao grupo dos FCustoB. Acreditamos que a "instabilidade" (grande desvio) desse tipo de função pode gerar tanto redes muito ruins, quanto redes muito boas. Mas é fato que a FCustoA tem uma otimização muito mais lenta que a FCustoB, podendo levar até 10 vezes mais tempo.

TABELA 6.6: COMPARAÇÃO DE PARÂMETROS DE TREINO PARA DISTRIBUIÇÃO RADIAL E ANGULAR.

	Parâmetros							Erro médio					Erro da melhor rede				
	n_{layers}	n_{pop}	CM	$\bar{r}$	H$\varphi\theta$	RD	σ	t (min)	Hr (%)	$H\varphi$ (%)	$H\theta$ (%)	Bohr (%)	d_{CM} (Å)	Hr (%)	$H\varphi$ (%)	$H\theta$ (%)	$\bar{r}$(%)	d_{CM}(Å)
FRes07A	[6 6]	100	0	0	1	1	0	36,67	17,92 ± 5,88	17,91 ± 6,21	9,40 ± 5,73	14,97 ± 8,52	0,16 ± 0,11	19,56	5,14	**0,73**	0,63	0,17
FRes07B	[6 6]	100	0	0	1	1	0	**4,65**	15,46 ± 8,27	18,17 ± 9,23	14,95 ± 12,33	26,71 ± 86,48	0,30 ± 0,55	6,92	7,98	1,73	4,96	0,19
FRes07C	[6 6]	100	0	0	1	1	0	49,48	35,21 ± 32,04	18,74 ± 7,00	18,05 ± 5,82	174,75 ± 319,8	0,16 ± 0,17	10,90	14,73	12,67	12,69	0,08
FRes07D	[6 6]	100	0	0	1	1	0	**4,42**	21,48 ± 25,56	21,53 ± 14,15	24,80 ± 14,14	54,39 ± 160,1	0,16 ± 0,13	8,42	**3,69**	12,72	1,59	0,12
FRes09A	[6 6]	250	0	0	1	1	0	98,16	15,77 ± 4,67	18,28 ± 7,05	8,29 ± 5,15	17,59 ± 7,48	0,17 ± 0,10	14,34	4,14	1,12	21,80	0,07
FRes09B	[6 6]	500	0	0	1	1	0	20,34	**10,79 ± 4,84**	19,43 ± 12,16	10,34 ± 7,56	13,52 ± 6,78	0,21 ± 0,12	**4,30**	8,14	2,59	3,05	0,06
FRes10A	[6 6]	100	1	1	1	1	0	37,77	21,02 ± 7,47	17,49 ± 6,10	12,23 ± 8,16	4,70 ± 9,00	**0,13 ± 0,11**	14,59	8,43	2,69	1,48	**0,03**
FRes10B	[6 6]	100	1	1	1	1	0	**4,23**	22,41 ± 13,24	19,51 ± 8,93	11,76 ± 11,23	8,67 ± 13,61	**0,13 ± 0,14**	7,03	11,18	2,01	3,98	0,08
FRes11A	[8 8]	100	0	1	1	1	0	38,30	19,80 ± 4,84	18,48 ± 6,82	8,48 ± 5,55	2,85 ± 4,17	0,18 ± 0,13	14,36	12,30	4,41	1,42	0,10
FRes11B	[8 8]	100	0	1	1	1	0	5,45	20,37 ± 9,30	18,26 ± 10,46	12,38 ± 12,86	9,34 ± 26,08	0,21 ± 0,16	12,88	8,54	4,63	0,33	0,29
FRes12A	[8 8]	250	0	1	1	1	1	136,39	14,23 ± 3,42	**15,90 ± 6,73**	**7,55 ± 4,58**	**1,97 ± 1,79**	0,19 ± 0,11	13,82	11,11	1,40	**0,05**	0,10
FRes12B	[8 8]	500	0	1	1	1	1	38,70	**10,87 ± 7,07**	**15,10 ± 7,02**	11,19 ± 6,77	5,26 ± 18,53	0,22 ± 0,11	8,69	8,28	1,20	0,42	0,05
FRes13A	[8 6]	100	1	1	0	1	0	93,55	16,22 ± 4,34	29,41 ± 18,77	40,23 ± 33,58	**1,94 ± 2,43**	0,19 ± 0,22	5,78	8,57	8,25	1,65	0,07
FRes13B	[8 6]	100	1	1	0	1	0	23,86	13,20 ± 6,10	30,94 ± 16,33	43,68 ± 34,72	3,40 ± 4,28	0,14 ± 0,17	11,07	15,25	4,31	1,36	**0,03**
FRes14A	[8 8]	250	1	1	1	1	1	147,26	15,01 ± 3,24	16,83 ± 8,33	**7,96 ± 5,88**	2,27 ± 3,29	0,15 ± 0,13	9,85	6,37	3,75	1,44	0,17
FRes14B	[8 8]	500	1	1	1	1	1	39,06	11,86 ± 11,09	17,54 ± 11,53	11,74 ± 14,63	2,82 ± 5,56	**0,13 ± 0,14**	**4,64**	**3,70**	6,27	2,24	0,18

Legenda: CM - centro de massa, d_{CM} - desvio do centro de massa, $\bar{r}$ - raio médio (3/2 de a_0), H$\varphi\theta$ - histogramas angulares, RD - radialmente distribuído, σ - desvio padrão.
FONTE: O autor (2020).
LEGENDA: FRes07 C e D utilizam coordenadas cartesianas, no qual a função 'C' é equivalente a 'A', e a 'D' é equivalente a 'B'. Os parâmetros comuns foram um amostral de 50 redes cada, $n_{iter} = 7$, $r_{eson} = 1$, $b_{ias} = 0$, $n_{ger} = 25$, $m_{rate} = 0.15$, $n_{elite} = 5$. Verificação do erro do histograma radial (HR) está presente em todos os testes. Em negrito os melhores valores por característica avaliada.

FIGURA 6.8: DESVIO DO CENTRO DE MASSA.



FONTE: O autor (2020).
LEGENDA: Após realizar iteração 10^5 vezes com uma das redes FRes14, foi calculado o centro de massa a cada 100 pontos. (a) O centro de massa possui um desvio de 0.18 Å, o '+' vermelho indica a origem e o verde o centro de massa. Em (b) apresentamos a distribuição dos centros de massa que formam uma distribuição quase gaussiana entorno deste ponto.

Notou-se uma repetitiva dificuldade em uniformizar o ângulo φ da elevação, mais complicada que o ângulo θ. As consequências de seus erros também foram mais evidentes, tendendo a formar vazios nos ângulos próximos ao eixo z.

6.4 ANÁLISE DAS MELHORES REDES

O critério para as melhores redes, que foram selecionadas dentre a coleção de treinamentos dos testes finais (tab. 6.6, com 50 redes cada teste FRes#), foi um erro inferior a 10% nos parâmetros de avaliação: erro nos histogramas radial e angulares (Hr,$H\varphi$ e $H\theta$) e no raio médio ($\bar{r}$). Seguindo este critério, percebemos que os treinamentos utilizando a FCustoB têm taxa de sucesso de cerca de 3 vezes maior que a com FCustoA. As melhores redes são apresentadas na Tabela 6.7).

O treinamento da rede originou múltiplos modelos, todos eles plausíveis com base nas informações fornecidas para seu treinamento. Os gráficos são trazidos em anexo para comparação com maiores detalhes, Capítulo C. Dentre as redes de melhor desempenho (tab. 6.7), a melhor de todas foi a FRes14B #34, e por isso aqui iremos detalhar seus resultados, enquanto as demais são apresentadas em forma de gráfico no Capítulo C em anexo.

Como primeiro passo, obtivemos uma sequência de pontos iterada pela FRes14B#34, como apresentado em corte na Figura 6.9(c). A escala de cores tem único propósito de evidenciar as camadas, o que permite visualizar a distribuição dos pontos na esfera, que apresenta

TABELA 6.7: COMPARAÇÃO DOS RESULTADOS DAS MELHORES REDES.

config.	#	Hr (%)	Hφ (%)	Hθ (%)	$\bar{r}$(%)	d_{CM}(Å)	RD
FRes07B	49	7,29 ± 0,42	8,55 ± 0,47	**2,81 ± 0,41**	9,71 ± 1,01	0,1983 ± 0,0048	2,21 ± 0,26
FRes09B	28	**4,76 ± 0,43**	8,48 ± 0,45	7,03 ± 0,54	5,85 ± 0,84	0,0620 ± 0,0055	2,10 ± 1,08
FRes09B	47	9,40 ± 0,43	8,19 ± 0,30	4,45 ± 0,38	18,69 ± 0,93	0,1243 ± 0,0042	1,40 ± 0,22
FRes12B	11	9,85 ± 0,40	**4,46 ± 0,44**	9,59 ± 0,45	**0,45 ± 0,35**	0,1811 ± 0,0048	1,15 ± 0,11
FRes12B	29	9,91 ± 0,48	5,66 ± 0,33	4,62 ± 0,42	3,85 ± 0,97	0,0696 ± 0,0048	1,14 ± 0,48
FRes12B	31	8,78 ± 0,43	7,27 ± 0,41	6,15 ± 0,45	2,37 ± 1,12	0,0680 ± 0,0075	3,49 ± 0,53
FRes12B	34	8,91 ± 0,40	7,96 ± 0,50	**2,44 ± 0,34**	1,28 ± 0,89	**0,0556 ± 0,0052**	1,28 ± 0,38
FRes13A	31	6,07 ± 0,43	8,89 ± 0,43	9,92 ± 0,55	3,00 ± 1,29	0,0821 ± 0,0041	1,81 ± 1,30
FRes14A	29	9,88 ± 0,42	6,67 ± 0,44	**3,08 ± 0,35**	2,92 ± 1,00	0,1669 ± 0,0055	2,06 ± 0,90
FRes14B	4	**4,77 ± 0,41**	**4,48 ± 0,43**	7,62 ± 0,36	4,65 ± 0,52	0,1773 ± 0,0056	2,53 ± 0,34
FRes14B	13	5,74 ± 0,31	10,23 ± 0,46	**3,62 ± 0,40**	1,75 ± 0,79	0,0771 ± 0,0032	1,25 ± 0,31
FRes14B	34	**3,51 ± 0,44**	6,91 ± 0,39	5,36 ± 0,43	**0,72 ± 0,42**	**0,0597 ± 0,0054**	**0,77 ± 1,19**

Legenda: d_{CM} - desvio do centro de massa, $\bar{r}$ - raio médio (3/2 de a_0), H$\varphi\theta$ - histogramas angulares, RD - radialmente distribuído, σ - desvio padrão.
FONTE: O autor (2020).
LEGENDA: Foram realizadas 50 réplicas, com pontos iniciais distintos, de iterações de 10^5 passos, sobre o qual foi obtida a média e desvio padrão. Em negrito os melhores valores por característica avaliada. RD em unidades arbitrárias, sendo zero um valor ideal pois indica distribuição totalmente homogênea.

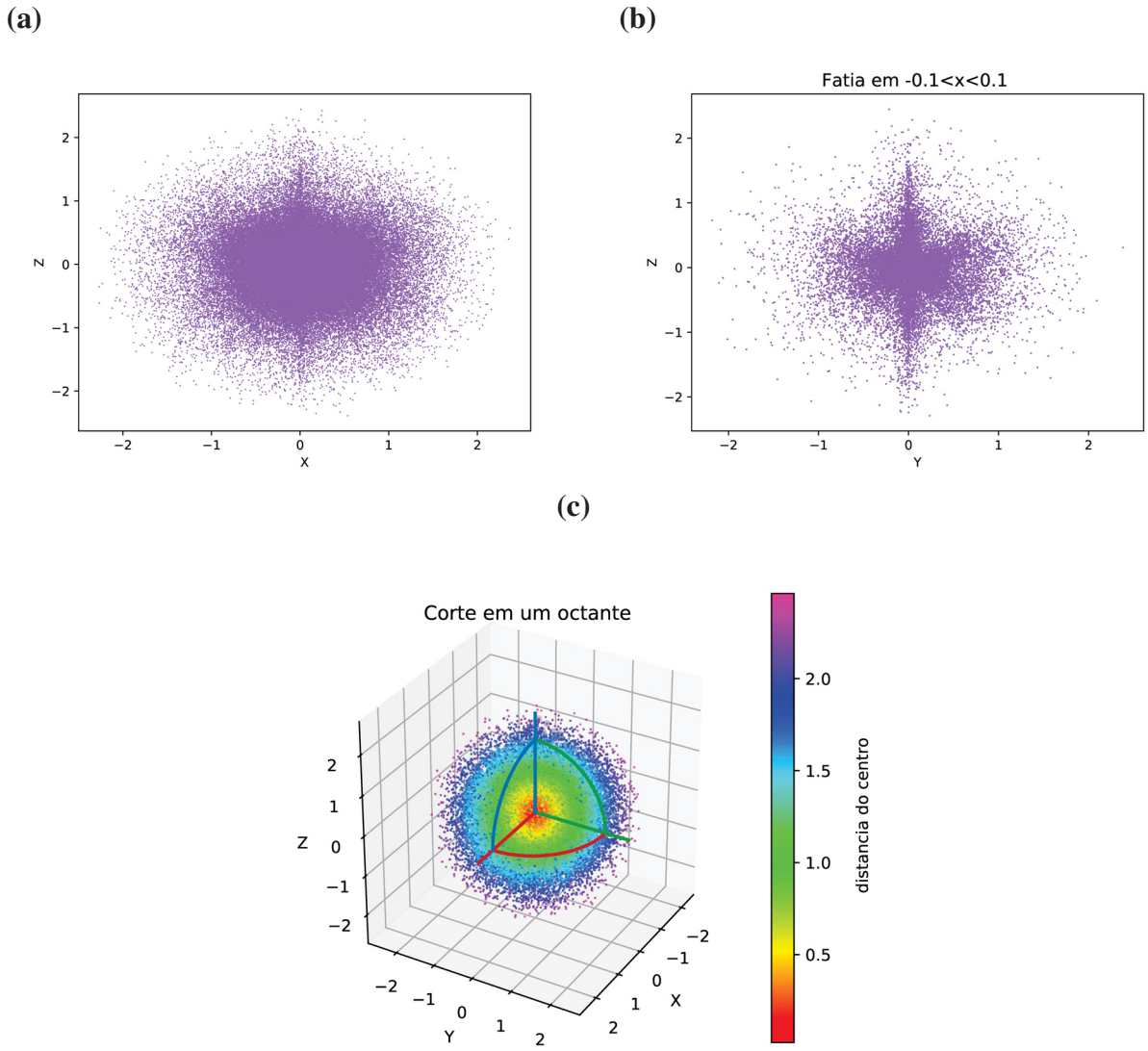
uniformidade numa visão mais geral. Observando a Figura 6.10, também a sequência de pontos iterada utlizando FRes14B#34, percebemos que não existe um padrão de repetição dos pontos, mas temos a distribuição ao lado que demonstra ser um atrator. Isso configura a construção de um atrator caótico.

Uma observação sobre esta distribuição iterada (como detalhada na Figura 6.10) é com relação a alguns padrões ocultos observados nas distribuições. Apesar de a entrada fornecida não possuir tendência sobre o eixo z (fig. 6.6), o treinamento das redes parece formar padrões secundários, como na Figura 6.9(a-b).

A uniformização da entrada não resultou em mudanças no treinamento, uma vez que o amostral gerado foi de 15 a 200 pontos, e o restante dos pontos foi obtido por iteração. A função radialmente distribuída também não conseguiu eliminar essas tendências, pois como na Tabela 6.7 o erro RD foi mínimo, indicando que esses padrões secundários ainda obedecem a distribuição proposta e não podem ser descartados como modelos válidos.

A propriedade atratora da rede não era esperada, mas demonstra a grande capacidade de aprendizado da rede. Fornecendo variados padrões de dados para a rede, obtemos sempre um padrão similar (mas não idêntico) de distribuição. As Figuras 6.11 e 6.12 trazem a comparação de diferentes conjuntos de dados de entrada e o resultado. A figura traz na primeira coluna os padrões de entrada em azul, então o resultado de após uma iteração em coordenadas esféricas. Observa-se que há pouca distinção entre os resultados provenientes da transformação dos diferentes conjuntos de entradas – os conjuntos apresentados foram cúbicos e esféricos.

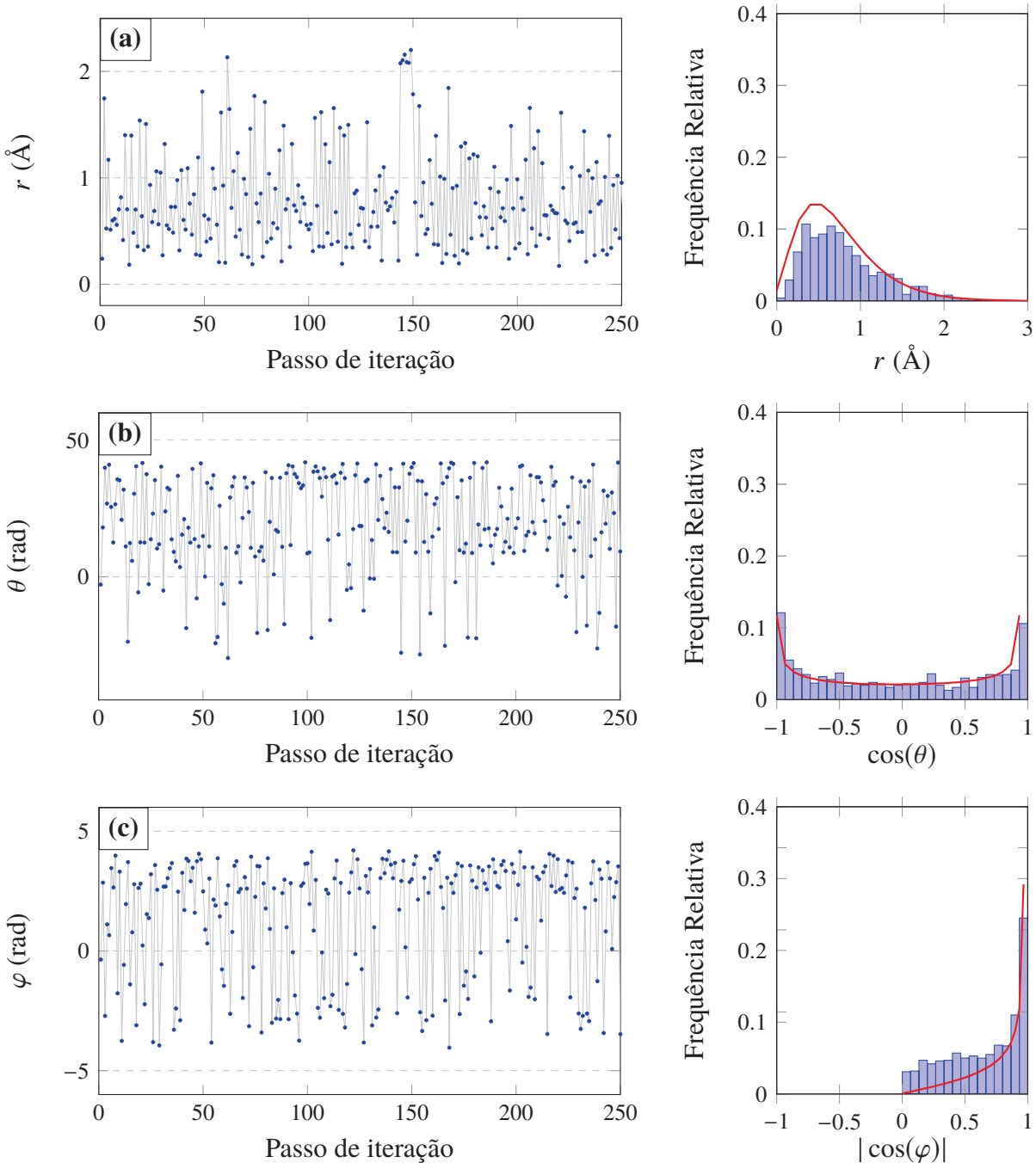
FIGURA 6.9: PLOT TRIDIMENSIONAL DO ELÉTRON ITERADO.



FONTE: O autor (2020).

LEGENDA: A formação de padrões secundários, como a tendência sobre o eixo z, persiste mesmo com a apresentação de dados sem tendência e o uso da função radialmente distribuído. Em (a) o *plot* suavizado dos eixos xz, e em (b) um *plot* dos eixos yz da fatia do eixo x, entre $-0.1 < x < 0.1$ provenientes da rede FRes14B#34. (c) Os pontos do octante positivo foram removidos para visualização da distribuição interior. A escala de cores refere-se a distância do elétron ao núcleo.

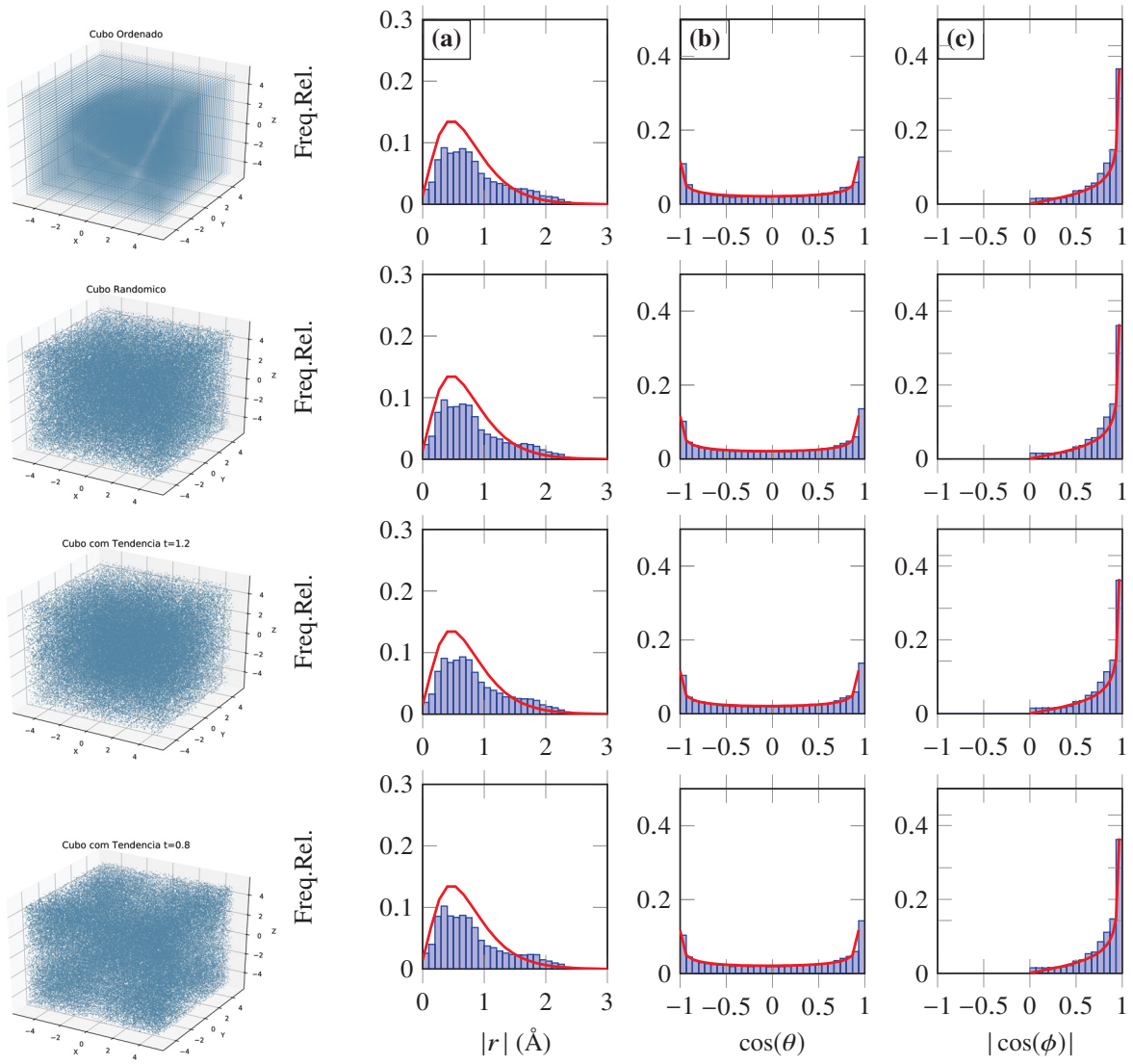
FIGURA 6.10: DISTRIBUIÇÃO CAÓTICA DOS PONTOS EM ITERAÇÃO.



FONTE: O autor (2020).

LEGENDA: Utilizando a rede FRes14B, #34, obtemos a sequência de pontos iterada e um padrão caótico de pontos. À direita, a distribuição que demonstra se tratar da construção de um atrator caótico.

FIGURA 6.11: CARÁTER ATRATOR OBSERVADO A PARTIR DE ENTRADAS CÚBICAS.



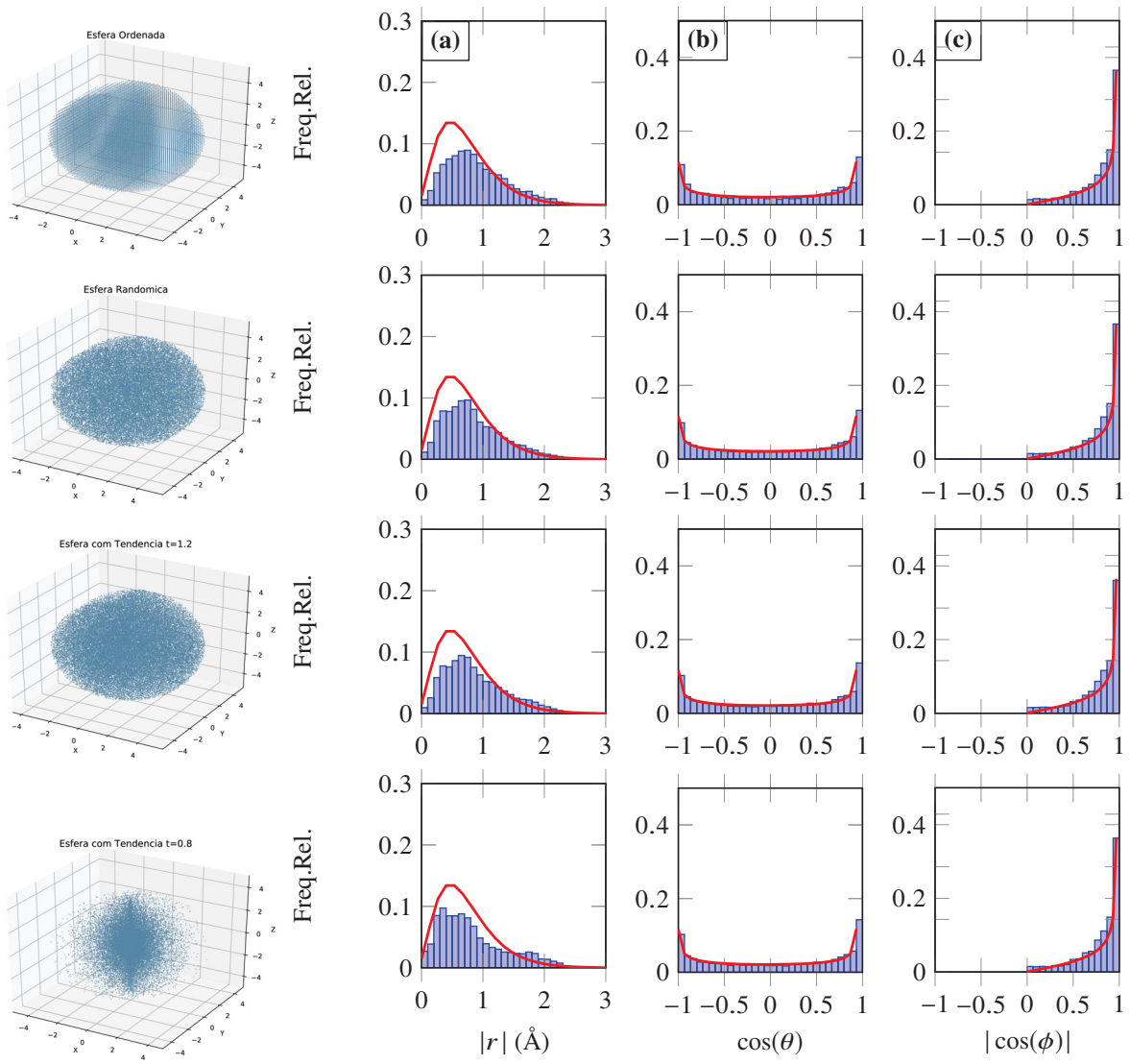
FONTE: O autor (2020).

LEGENDA: Cada linha corresponde a (1) pontos ordenado, (2) pontos randômicos, (3) com tendência para fora (t=1.2) e (4) tendência para dentro (t=0.8). As colunas correspondem a comparação com as distribuições referências de (a) raio, (b) ângulo θ e (c) ϕ.

De maneira similar, a Figura C.24 (em anexo) apresenta a capacidade atratora da rede quando apresentada a esferas ocas concêntricas com raio de 0.5 Å(em vermelho). Objetivamos aqui compreender se havia algum padrão interno da rede na geração dos pontos das camadas mais internas e externas. Ao que pudemos observar, a partir de certas camadas de entrada (em azul), as saídas tiveram tendência a se agrupar em certas regiões, ora mais internas, ora mais externas, mas sem um padrão claro. A ausência de padrão pode ser verificada nos gráficos originados a partir das demais redes (cap. C), que não apresentam consenso quanto ao comportamento. O que

observa-se é que a maioria consegue se aproximar da distribuição esperada já no segundo passo de iteração, e algumas já no primeiro passo, independente da entrada.

FIGURA 6.12: CARÁTER ATRATOR OBSERVADO A PARTIR DE ENTRADAS ESFÉRICAS.



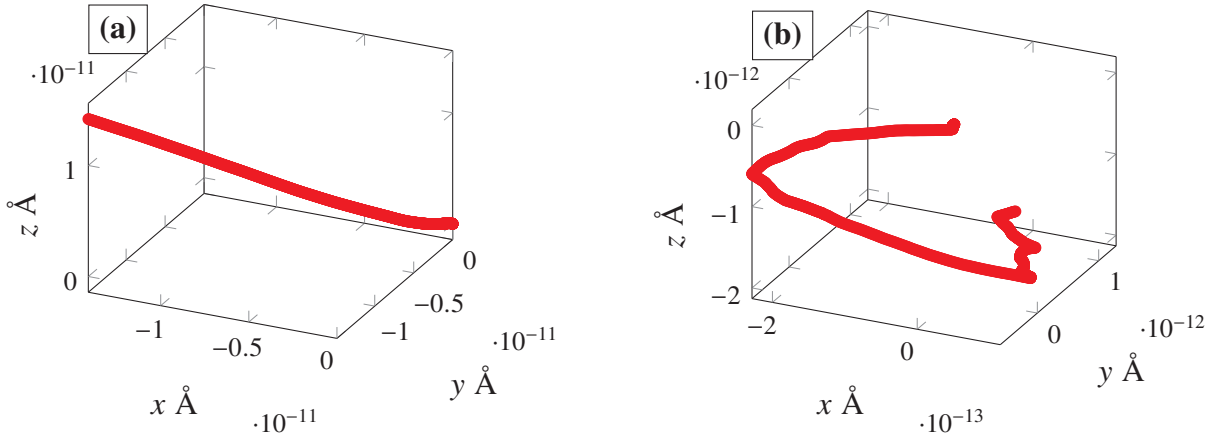
FONTE: O autor (2020).

LEGENDA: Cada linha corresponde a (1) pontos ordenado, (2) pontos randômicos, (3) com tendência para fora (t=1.2) e (4) tendência para dentro (t=0.8). As colunas correspondem a comparação com as distribuições referências de (a) raio, (b) ângulo θ e (c) ϕ.

6.4.1 Integração a um sistema dinâmico

A simulação semiclássica foi testada e comprovou a estabilidade do sistema, nas condiçoes utilizadas. Tomando o núcleo partindo da origem, foi obtida a sua trajetória (fig. 6.13(a))

FIGURA 6.13: TRAJETÓRIA DO NÚCLEO EM INTERAÇÃO COM O ELÉTRON PREVISTO PELA REDE FRES14B#34.



FONTE: O autor (2020).

LEGENDA: Em (a) utilizando a FRes como estimador da posição do elétron e em (b) utilizando Monte Carlo como comparativo.

em 10^4 passos de tempo com intervalo de 10^{-14} s, e com 100 iterações da rede (FRes14B#34) a cada passo. Das 100 iterações obtivemos a posição média do elétron.

A Figura 6.13(a), com a trajetória do núcleo mostra que existe uma tendência na movimentação decorrente do desvio de centro de massa da rede. Contudo, este desvio é muito pequeno na escala de 10^{-11} Å(para os 10^4 passos), o que é explicado pela diferença de massa entre o elétron e o núcleo. Realizamos o mesmo experimento, mas utilizando um Monte Carlo como fonte das posições do elétron, o qual não possui desvios – Figura 6.13(b). Neste o desvio obtido foi na escala de 10^{-12} Å, dez vezes menor.

Nota-se observando os gráficos que a rede gera uma tendência para apenas uma direção, enquanto que no experimento utilizando os pontos gerados por MC tem direção oscilante. Como consequência, a rede faz com que o núcleo tenda a caminhar numa direção, tal como a ação de uma força externa, que não existe, enquanto que o MC ao oscilar tende a se mover em torno da origem, sem tender em nenhuma direção preferencialmente.

O passo de tempo utilizado foi de 10^{-14} s, enquanto que um elétron leva aproximadamente 10^{-15} s para completar uma volta em torno do núcleo. Isto significa que entre um ponto e outro do movimento do núcleo, o elétron realizou 10 voltas completas.

7 DISCUSSÃO

Partimos da necessidade de ver o modelo atômico com outros olhos. Participamos aqui da busca por maneiras diferentes de se reproduzir computacionalmente o comportamento do elétron ligado. Nos afastando do uso mais convencional das soluções exatas e das suas aproximações para N-corpos, que são muito custosas computacionalmente e também em termos de desenvolvimento, para um modelo baseado exclusivamente em dados estatísticos da densidade de probabilidade do hidrogênio.

Desenvolvemos a rede FRes para preencher uma lacuna metodológica, mas obtivemos uma rede também inovadora em termos de inteligência computacional. A rede desenvolvida não pode ser completamente classificada em nenhuma categoria pré-existente de redes e métodos de otimização. A FRes apresenta-se como uma rede de otimização – além da otimização interna, ela é capaz de realizar otimização para a solução de problemas. E dentre suas características mas marcantes, e provavelmente a mais importante, é o uso de uma função de custo externa e maleável, customizável para variados problemas tal qual um algoritmo evolutivo.

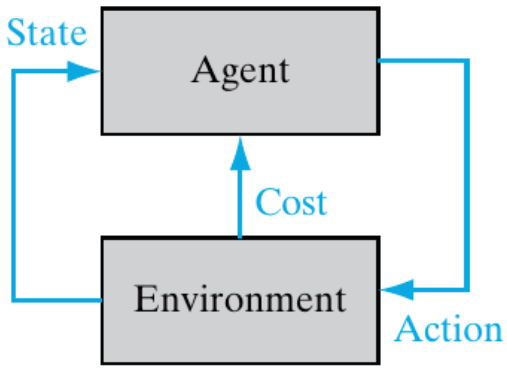
7.1 SOBRE A ESTRUTURA DA FRES

A rede proposta pode ser classificada como uma "*Evolving Neural Network*" por possuir otimização dos pesos mediada por algoritmo evolutivo, mais precisamente, um algoritmo genético. Sua otimização pode ser realizada por qualquer outro tipo de algoritmo evolutivo compatível, como o *simulated annealing*, por exemplo. O SA foi incluso com esse objetivo de demonstrar a versatilidade e sua modularidade.

A rede proposta não corresponde a nenhuma das classes de métodos de aprendizado apresentadas no Capítulo 3 de revisão. Entretanto, seu formato melhor se enquadra com o aprendizado por reforço *model-free* de estados contínuos, o qual não utiliza cadeias de Markov como demais métodos de RL uma vez que este modelo é utilizado para casos onde não se conhece a probabilidade de transição entre os estados.

É possível encontrar vários paralelos conceituais entre o *Q-learning* associado a uma rede *feedfoward* e a FRes, características que também preenchem os requisitos definidos em Gaskett, Wettergreen e Zelinsky (1999) para que um sistema *Q-learning* consiga lidar com estados e ações contínuos, como apresentado na Subseção 3.4.1. A FRes é, por conceito, *model-free* por não ter uma estrutura de supervisão implícita, ou em termos de RL, não necessita conhecer a probabilidade de transição associada.A rede permite naturalmente o uso de valores contínuos (estados não discretos), configurando a continuidade da ação. A função de custo confere a avaliação do estado, e a GA direciona o treinamento para regiões de menor custo por tentativa e erro, desempenhando o papel de seleção da ação. O formato em que a FRes usa a função de custo garante a generalização do modelo em estado e ação. Adicionalmente a FRes traz a

FIGURA 7.1: ESTRUTURA DE TREINAMENTO DE UMA *Q-LEARNING*.



FONTE: Haykin (2008a).

possibilidade de avaliação local e global (similar mas não idêntico ao conceito de recompensa - local - e reforço esparso - global).

Apesar de grande a semelhança, este modelo de RL difere substancialmente da FRes. Diferentemente da *Q-learning*, a FRes não ajusta seus pesos por meio de interpolação, mas sim utilizando a função de busca da GA (aspecto que foi explorado em trabalhos posteriores a Gaskett, Wettergreen e Zelinsky (1999)) (HASSELT, 2012). Ainda, o RL *model-free* como o *Q-learning*, são fortemente vinculados a ideia de um estado atual, que pode corresponder a entrada do sistema de aprendizado (fig. 7.1). A FRes não possui essa dependência implícita, pois o que importa é o custo, e como ele é obtido é totalmente a critério do usuário. Adicionalmente, a rede no formato apresentado permite performar iteração em sua função de custo, utilizando a própria rede que está sendo otimizada, o que não é descrito nos modelos investigados. O modelo apresentado por Gaskett, Wettergreen e Zelinsky (1999) realiza a otimização enquanto é realizada a iteração, passo a passo, o que pode ser um tanto limitador. Além disso, a FRes possui a função de ressonância (alg. B.6), um aspecto indispensável para a solução de problemas complexos não lineares. Trabalhos mais recentes que citam Gaskett, Wettergreen e Zelinsky (1999) demonstram que obter um RL de políticas e ações contínuas é um desafio, e quanto maior o espaço de ação, mais complicado de torna, forçando muitas vezes a recorrer a aproximações (MOLINEAUX; AHA; MOORE, 2008; PAZIS; LAGOUDAKIS, 2009; BUSONIU *et al.*, 2010; ZHONG *et al.*, 2019).

Agora, realizando uma comparação da FRes com as redes profundas, uma característica que se torna evidente é a sua capacidade de executar tarefas razoavelmente complexas com um número pequeno de nós e camadas. Nossos modelos limitaram-se a duas camadas com até 10 nós, enquanto que na literatura as redes de *deep learning* alcançam facilmente centenas de neurônios.

Neste trabalho, realizamos uma busca exaustiva pela função de custo ideal para melhorar o treinamento de nossa rede. Isso vai contra a tendência da linha de trabalhos que utilizam redes neurais profundas. Cada abordagem possui suas vantagens e desvantagens. As redes profundas

eximem o usuário da necessidade de buscar a entrada adequada de atributos físicos que são relevantes para o treinamento da rede, pois toda a informação é fornecida permitindo à rede descobrir quais informações são úteis – denominado de aprendizado inexpressivo – porém exige redes de enormes dimensões e treinamento demorado (MILLS; SPANNER; TAMBLYN, 2017). Por outro lado, nossa abordagem toma tempo para investigar a melhor combinação de entradas (ou no nosso caso, de avaliações), contudo, o tempo de treinamento é bem inferior e a dimensão das redes bastante reduzida.

Dado esses comparativos, pudemos observar em nossos resultados algumas características interessantes sobre a estrutura e o uso da FRes. A primeira e mais evidente é a sua função de ressonância. O problema do XOR nos permitiu verificar que a rede com ressonância é essencial para a solução de problemas não lineares. Por outro lado, a classificação da *Iris* demonstrou que para problemas lineares a rede sem ressonância é mais eficiente. Isso talvez seja justificado pelo fato da rede sem ressonância possuir um número de nós 3 vezes menor, o que facilita a busca do valor ótimo. A ressonância também se mostrou essencial para obter as distribuições do átomo de hidrogênio e do gerador de instâncias, problemas mais complexos e não lineares

É importante reforçar que, diferentemente das redes ART, a ressonância aqui não é um fenômeno observável, mas sim uma inspiração na construção das funções internas, assim como o uso de senóides possuem similaridade com a série de Fourier.

A segunda característica da FRes está no uso de uma função de custo à semelhança de uma GA, o que dá grande maleabilidade na construção dos modelos. O uso dessa função de custo variável permite todos os três tipos de aprendizado, sendo que dois desses foram aqui apresentados com sucesso: o aprendizado com supervisão, e por reforço. A separação da função de custo da estrutura da rede a torna generalista.

Antes de aplicá-la diretamente ao problema do átomo de hidrogênio, validamos a rede com sucesso para classificação e para a aplicação em *augmentation* com o uso de distribuições de frequências na função de custo. Esse passo era necessário antes de integrar o uso de histogramas de frequência no problema objetivo, já que o uso de histogramas não é usual na literatura, e pelo fato de se tratar de um aprendizado mediado apenas por supervisão global.

A arquitetura da rede, assim como a função de custo desenvolvida, se mostraram capazes e eficientes em replicar as instâncias, considerando que utilizamos apenas 25 pontos de cada classe para o treino da rede geradora, um resultado bastante promissor. Esse sucesso inicial pode resultar em trabalhos futuros com aplicações em outros dados reais que necessitem de geração de pseudodados para o treinamento de redes para classificação e regressão. Esse método tem uso certo onde a disponibilidade de dados é limitada.

É preciso atentar, entretanto, para o fato de que a diferença percentual entre histogramas pode não ser uma boa medida para poucos dados. Percebemos que a escolha dos intervalos das barras dos histogramas – *bins* – pode alterar e muito a distribuição. Isso ocorre especialmente quando há poucos dados, como no caso da *Iris*, acarretando variação do erro entre os histogramas e impossibilitando obter uma medida adequada e confiável dentro da função de custo. Por este

motivo o uso de médias, desvios e correlações – e outras métricas que se mostrarem úteis – são importantes para garantir a geração de dados similares aos originais. Esse mesmo princípio é válido para o caso do hidrogênio. Contudo, os histogramas são mais confiáveis nesse caso pois são provenientes de um conjunto de dados suficientemente grande.

Esta rede se comporta como um atrator de um único estado, isto é, foi treinada para convergir para apenas um padrão esperado – uma rede por classe. Da mesma forma, e utilizando o mesmo princípio, a FRes treinada para simular o comportamento do elétron age como atrator. Diferentes formatos de entrada tendem a formar o mesmo perfil de saída, seja em um único passo, ou após N iterações.

7.2 APLICAÇÃO SOBRE O COMPORTAMENTO DE PARTÍCULAS

A nossa incapacidade de medir e prever analiticamente a posição do elétron a cada instante exigiu a elaboração de um modelo de aprendizado capaz de realizar predição na ausência de supervisão.

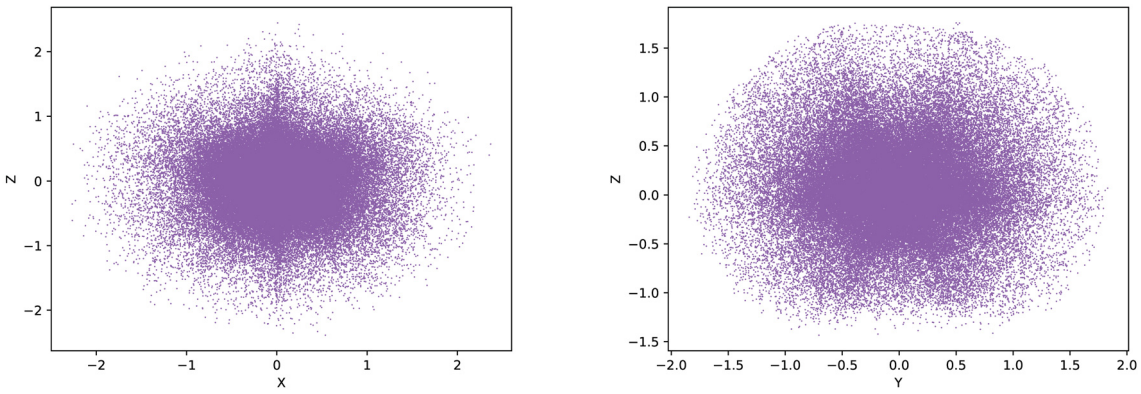
Os resultados para o problema do átomo de hidrogênio foram obtidos com o uso de uma rede neural, que fornece em recursão consistentemente a mesma distribuição para uma série temporal desconhecida. Nosso grande diferencial não foi ter conseguido uma distribuição aleatória de ângulos e densidade de pontos similar à de Schrödinger radialmente, mas conseguir isso com uma rede, treinando-a de forma a obedecer parâmetros específicos globais, sem fornecer uma saída precisa desejada (supervisão). Isso abre caminho para explorar novas maneiras de se realizar dinâmica molecular, diferentemente dos modelos baseados em potenciais de energia (RUPP *et al.*, 2012; BEHLER, 2017). Conseguimos demonstrar a possiblidade de integração do sistema a MD de modo introdutório (fig. 6.13(a)).

O que obtivemos aqui não pode ser considerado um modelo para a trajetória do elétron, mas sim um modelo do comportamento do elétron. A rede aprendeu como representar o elétron tanto como partícula quanto como onda. Já que não é sabido como representar a dualidade onda-partícula do elétron, a rede é que fará esse trabalho por nós. O que estamos fazendo aqui não é apenas ensinar padrões à rede, mas sim como a natureza se comporta.

Como mencionado, a FRes permite o uso dos dois tipos de verificação: local e global. Ou seja, podemos verificar o erro ponto a ponto, seja em um passo (local, como numa classificação, ou *time series*) ou em iteração (verificação do comportamento global, como a densidade e frequência). Se soubéssemos a trajetória de um elétron, ponto a ponto, poderíamos realizar as duas verificações.

Vinculado ao conceito de atração há também o conceito de atraso, que é o tempo (número de iterações) necessário para alcançar a bacia de atração. Nos testes realizados utilizando pontos iniciais já inseridos na bacia de atração, o resultado imediato está na bacia de atração em uma única iteração. Contudo, testes com outros padrões como cubos e esferas com variadas distribuições alcançaram também a bacia de atração em um único passo de iteração, o que é ideal.

FIGURA 7.2: PADRÕES DISTINTOS PROVENIENTES DE DUAS REDES TREINADAS COM OS MESMOS PARÂMETROS E SIMILAR ERRO.



FONTE: O autor (2020).
LEGENDA: À esquerda, FRes14#34, e à direita FRes14#13.

Tomamos o cuidado de não haver nenhum aprendizado não fundamentado (forçagens). Deixamos a rede livre, dentro do possível, para explorar os vários padrões, o que resultou em um atrator tipo caótico, que não possui um padrão evidente. Não sabemos como o elétron se comporta em detalhes, logo, todas as possibilidades de movimento, obedecendo às limitações impostas, são válidas como modelo de seu comportamento. Qual se aproxima do comportamento real é uma questão a ser discutida e pensada. Tínhamos em mente que padrões ocultos (não explícitos) podem ser aprendidos, e estes padrões ocultos podem ser justamente os que representam o comportamento físico, porém, como ocorre em qualquer aprendizado, infelizmente não temos garantias disso.

Apresentamos diversas combinações de parâmetros de avaliação e parâmetros para o átomo de hidrogênio. Observamos que utilizando os mesmos parâmetros e avaliações obtivemos padrões distintos, isto é, diversas distribuições possíveis de pontos no espaço surgiram e podem ser considerados válidos, pois se obedecem às exigências apresentadas (fig. 7.2). Como apresentado (tab. 6.7), nenhuma rede alcançou erro nulo, o que era esperado, uma vez que é impossível obter duas distribuições absolutamente idênticas, como demonstrado no gráfico da Figura 6.2. Ainda assim, obtivemos redes com resultados promissores com erros muito pequenos. Todos os padrões atratores gerados, com erro baixo, podem representar a nuvem eletrônica e não podem ser descartados como padrões válidos. Pela teoria física, ainda não somos capazes de definir se há uma estrutura única capaz de representar todos os átomos de hidrogênio, ou se cada átomo possui uma característica própria, individualizada. Queira dizer, as trajetórias dos elétrons seriam iguais para todos os hidrogênios, no mais fino detalhes, ou cada um teria características próprias, tornando-o único.

Observamos que as redes treinadas com a FCustoB tiveram melhor desempenho que as treinadas com FCustoA. Deduzimos que um dos principais fatores que determinaram essa vantagem seja o maior número de pontos inciais, que permitiram explorar melhor o espaço da

função. Depois, as redes que obtiveram melhores resultados pertenciam aos FRes09B, FRes12B e FRes14B, que tinham um n_{pop} duas vezes maior que as suas versões com FCustoA, o que certamente impactou positivamente na redução dos erros. Por fim, podemos supor que o alto desvio apresentado pelas redes treinadas com FCustoB permitiu obter tanto redes muito ruins quanto muito boas. O treinamento do modelo em nenhum dos casos (FCustoA e B) exigiu um número grande de iterações, alcançando bons resultados. Isso permitiu obtermos um tempo de treinamento bastante reduzido, indo de 5 minutos (com FCustoB) a duas horas no máximo (com FCustoA).

Realizando um comparativo da FRes, o MCMC possui um emprego bem mais direto da teoria na geração dos dados, uma vez que se utiliza de algoritmos de Monte Carlo. Este modelo estatístico satisfaz o requisito para avaliação global, mas ainda permite uma avaliação local por meio de cadeias de Markov. Para o presente problema, seria totalmente viável implementar o MCMC, que resultaria em um amostral coerente com as nossas especificações, e ainda possui como vantagem ser livre de otimização. Contudo, o MCMC é um gerador de amostral, não são armazenados pesos nem sequências, isto é, cada vez que é preciso utilizar o modelo, é necessário usar valores randomizados para gerar os pontos, que podem ser aceitos ou rejeitados. Essa necessidade de realizar várias tentativas que envolvem rejeições podem tornar a tarefa custosa em tempo computacional (especialmente quando utilizado para problemas de otimização) (KIM; PARK; LEE, 2009; HAMDI; HAJIZADEH; SOUSA, 2015). Pelo fato do MCMC ser baseado em curvas de distribuição, a inclusão de outros tipos de avaliações como o valor médio da distribuição ou algum tipo de avaliação mais complexa que não envolva especificamente as curvas de probabilidade, se tornam complicadas. Essas avaliações certamente serão necessárias para trabalhos futuros em sistemas com N-corpos.

O fato de não possuirmos um conjunto de dados de treinamento, exigiu que estes fossem gerados. Como mencionado, o método de Monte Carlo é simples e eficiente para a geração de pontos em uma distribuição, e por isso foi aqui utilizado. Essa aplicação traz enorme vantagem com relação a outros trabalhos, como os dependentes de simulações *ab initio*, que sofrem com a escassez de dados. Contudo, é preciso lembrar que isso somente foi possível pois o estado fundamental do hidrogênio possui resultado analítico conhecido, o que não é verdadeiro para os casos mais complexos, envolvendo sistemas de múltiplos elétrons.

Com este trabalho não propomos a substituição dos modelos já existentes pelo aqui desenvolvido. Aqui, nós propomos um novo modelo a ser explorado e melhor estudado. Com este, tivemos um vislumbre na exploração de métodos alternativos de representar as partículas e suas interações, utilizando inteligência artificial.

7.3 PERSPECTIVA PARA TRABALHOS FUTUROS

Como já afirmado, o modelo construído para o hidrogênio se trata de um modelo preliminar. Trabalhamos apenas sobre o sistema quântico mais simples, o átomo de um elétron

em estado fundamental. Tendo conseguido isso, podemos agora avançar para seus níveis excitados e a transição entre estes estados. Também não consideramos outras informações relevantes, como o momento e energia, que podem auxiliar na sua integração e dinâmica molecular, e melhorar a descrição do átomo. Seu objetivo maior é ser integrado a modelos de dinâmica molecular, envolvendo N-corpos. Para isso, será necessária a inclusão de avaliações que englobem a interação de N-partículas.

A entrada e saída utilizadas foram limitadas a posições, entretanto poderíamos utilizar a curva de probabilidade das velocidades, da mesma maneira que conhecemos estatisticamente as posições, podendo integrar outras avaliações sobre o sistema como a energia média. Conhecer a velocidade e posição simultaneamente acarreta a quebra do princípio da incerteza, mas pode-se argumentar que este princípio é decorrente do problema da medida, e pode não corresponder a uma verdade física absoluta.

Uma grande questão a ser futuramente investigada é a capacidade do modelo aprender a descrever o elétron para outras condições completamente diferentes. Foi necessário treinar muitas redes para obter bons resultados do comportamento de um único elétron. Para sistemas mais complexos com N-corpos e/ou condições ambientais variadas (presença de um campo eletromagnético por exemplo), provavelmente será necessário incorporar outros métodos de treinamento e de funções de custo mais rebuscadas.

Há ainda a possibilidade de incorporar na função de custo equações explícitas envolvendo as funções de onda, num formato totalmente diferente do proposto. As possibilidades a serem exploradas são muitas, e cabe uma revisão bibliográfica mais profunda para definir qual seria o caminho mais lógico a seguir.

Uma discussão corrente com relação às redes neurais é a obrigatoriedade de usar entradas de tamanho fixo (MILLS; SPANNER; TAMBLYN, 2017; BEHLER, 2017), compartilhada por este modelo. A inclusão de múltiplos elétrons deverá ser repensada.

Considerando, agora, a FRes como um produto deste trabalho com potencial, é possível ainda realizar melhorias relativas a sua função de otimização. O GA é um ótimo algoritmo que, por meio de busca, permite explorar o espaço e simultaneamente realizar a descida pelo gradiente. Entretanto, o GA é um gargalo de tempo e para o tamanho da rede. Foi observado que no treino de redes com mais de 20 neurônios, divididos em diferentes configurações de camadas, com ressonância, a rede apresentou dificuldade em alcançar resultados satisfatórios. Por este motivo, temos como perspectiva desenvolver um algoritmo de otimização específico de descida no gradiente, independente de busca.

Como alternativa há a possibilidade da incorporação de algoritmos de otimização tais como o descrito no trabalho de Goto, Tatsumura e Dixon (2019), que são muito mais velozes e que podem permitir a obtenção de resultados satisfatórios para redes e funções de custo mais complexas.

O desenvolvimento da *Feature Resonance* abre espaço para explorar não apenas aplicações para a descrição do mundo microscópico, mas também para diversos fins em

variadas áreas do conhecimento. A multiplicação de instâncias (*augmentation*) (WANG; PEREZ, 2017; GUPTA, 2019), como brevemente apresentado, é uma das possíveis aplicações da rede desenvolvida. Outra possibilidade está no uso da FRes aplicado à clusterização, como foi idealizada com o uso da função `pdist` (eq. 4.3).

8 CONCLUSÃO

O desenvolvimento constante de novas técnicas de modelagem molecular permite o avanço no estudo das interações moleculares, e assim o avanço no estudo e desenvolvimento de técnicas para reproduzir computacionalmente as propriedades físicas e bioquímicas da matéria, com variadas aplicações (MILLS; SPANNER; TAMBLYN, 2017; SELLIER *et al.*, 2019; CAPELLE, 2006).O aprendizado de máquina é um grande aliado nesses estudos, como já foi demonstrado em diversos trabalhos (MATER; COOTE, 2019; LOPEZ-BEZANILLA; LILIENFELD, 2014; BEHLER; PARRINELLO, 2007; RUPP *et al.*, 2012; HANSEN *et al.*, 2013; LI *et al.*, 2013; DOLGIREV; KRUGLOV; OGANOV, 2016; BEHLER, 2017; CHMIELA *et al.*, 2017; CHMIELA *et al.*, 2018; ZHANG *et al.*, 2018a; BUTLER *et al.*, 2018).

Aqui apresentamos uma nova de rede neural, *Feature Resonance Neural Network* (FRes), que abre caminho para novas possibilidades de exploração. A FRes é uma rede generalista – não especializada – com função de custo customizável e de otimização evolutiva (algoritmo genético). Sua arquitetura é baseada em ressonância, o que confere a capacidade de solucionar problemas não lineares. Esta rede se caracteriza por permitir a implementação aos vários tipos de aprendizado (supervisionado, não supervisionado, por reforço) aplicados a variados problemas, permitindo uma avaliação local (supervisionada) e global (reforço esparso).

O primeiro teste realizado, problema do 'ou exclusivo', permitiu verificar que a ressonância é essencial para a solução de problemas não lineares. Para validar a rede, foram realizados testes de classificação (aprendizado supervisionado) com os conjuntos de dados da *Iris* e câncer de mama (sec. A.1 em anexo), obtendo respectivamente (93.99 ± 2.96)% e (96.69 ± 1.12)% de acurácia para a melhor configuração de parâmetros (tab. 6.2), valores estatisticamente iguais a performance de uma MLP.

Para validar a rede com relação ao uso de histogramas de distribuições na função de custo, foi utilizado o problema da geração de instâncias (*augmentation*), um modelo sem supervisão, que utilizou apenas um custo global. Aplicado ao conjunto de dados da *Iris*, o melhor resultado obtido foi de (96.32 ± 0.84)% de acurácia para a classificação dos dados gerados (tab. 6.3), numa proporção de 50:50 entre treino e teste. Este resultado foi estatisticamente igual à performance da classificação sobre os dados originais da *Iris*.

Após a validação da FRes, iniciamos uma série de testes preliminares que nos permitiram definir os parâmetros adequados para o treinamento da rede para representar o comportamento do elétron do átomo de hidrogênio. Tais testes permitiram verificar que o descritor de coordenadas esféricas era mais eficiente que o de coordenadas cartesianas na representação do sistema (fig. 6.7). A distribuição de nós era mais efetiva com o uso de duas camadas ocultas, cada qual com 6 ou 8 nós (fig. 6.3 e 6.4). Também verificamos dois tipos de estrutura iterativa, FCustoA e FCustoB, (alg. B.3); a primeira apresentava vantagem por fornecer médias e desvio padrão menores, e a

segunda tinha menor tempo de treinamento e redes com resultados individuais melhores que a média, superando as melhores rede de FCustoA.

As funções de custo desenvolvidas buscavam avaliar o comportamento do elétron apenas globalmente, baseadas na densidade de probabilidade de Schrödinger. Desta teoria se derivaram várias funções avaliativas, que foram testadas em diversas combinações: diferença entre os histogramas radial (Hr) e angulares ($H\varphi\theta$), raio médio ($\bar{r}$), homogeneidade da distribuição(RD), centro de massa (CM), desvio padrão (σ). Os resultados demonstraram que o uso de todas essas avaliações em uma única função de custo, associado a um tamanho populacional grande ($n_{pop} \geq 250$), é a melhor configuração para o treinamento, resultando nas melhores redes treinadas (tab. 6.6).

Obtivemos 12 redes com erro médio inferior a 10% nos parâmetros avaliados (tab. 6.7, cap. C anexo). A rede de melhor desempenho foi a código FRes14B#34, que obteve erros percentuais de 3.51 ± 0.44 %, 6.91 ± 0.39 %, 5.36 ± 0.43 %, 0.72 ± 0.42 % e 0.0597 ± 0.0054 Åna comparação dos histogramas radial, angulares (φ e θ), no raio médio e desvio do centro de massa, respectivamente.

No processo de análise, descobrimos que as redes treinadas apresentavam propriedades atratoras, permitindo classificar o modelo como um atrator caótico (fig. 6.10). As redes conseguiram, a partir de variados padrões de entrada (ordenados e randômicos), obter em apenas uma iteração (fig. 6.11 e 6.12) ou em poucos passos (fig. C.24) o padrão similar ao da densidade de probabilidade do átomo de um elétron em estado fundamental. O mesmo padrão também era obtido em vários passos iterativos partindo de um único ponto de partida (fig. 6.10 e 6.9).

Os resultados nos permitiram concluir que o comportamento do átomo de hidrogênio, apesar de ser o sistema atômico mais simples existente, não é um problema trivial. A rede conseguiu reproduzir seu comportamento por meio do desenvolvimento da propriedade atratora.

O modelo não teve como objetivo fornecer trajetórias literais, mas sim uma representação pontual e temporal do comportamento do elétron a partir de informações puramente estatísticas. É preciso enfatizar que o modelo aprendido é capaz de representar apenas alguns aspectos da natureza do átomo de um elétron, que nos permite idealizar a sua futura aplicação em modelos de dinâmica molecular. Este trabalho é apenas o primeiro passo no desenvolvimento de um modelo de aprendizado capaz de representar o comportamento atômico e molecular, um modelo alternativo aos já existentes.

A rede FRes também é por si só um resultado deste trabalho. Esta distingue-se dos modelos de aprendizado pré-existentes, se aproximando mais do aprendizado por reforço *model-free* de estados contínuos (GASKETT; WETTERGREEN; ZELINSKY, 1999). Contudo, a FRes ainda diferencia-se por não utilizar sistema de recompensa com otimização local no decorrer do treinamento, e por utilizar algoritmos evolutivos para sua otimização. Adicionalmente, a FRes permite performar iteração em sua função de custo, o que não é descrito nos modelos investigados. O grande diferencial da arquitetura da FRes é a generalidade permitida pela função de custo, customizando as avaliações específicas para cada problema, tornando-a tão maleável quanto

um algoritmo genético. A rede apresenta também vantagem com relação à *deep learning* por conseguir executar tarefas razoavelmente complexas com um número pequeno de nós e camadas.

A estrutura da rede abre possibilidades de aplicações em inúmeras áreas, não apenas na modelagem molecular. A FRes é uma possível solução para quaisquer problemas que envolvam aprendizado global, como aqueles solucionados por aprendizado por reforço, ou que exijam funções de custo mais complexas – que fujam dos problemas classificação e regressão para os quais existem arquiteturas de redes mais simples e eficientes. Além dos treinos supervisionados e por reforço, também seriam possíveis os treinos não supervisionados, como na construção de *autoencoders*. Os resultados dos testes realizados na geração de instâncias foram promissores, e esta é uma aplicação a ser explorada futuramente.

REFERÊNCIAS

ANDRIEU, C. *et al.* An introduction to MCMC for machine learning. **Machine learning**, Springer, v. 50, n. 1-2, p. 5–43, 2003.

ARTRITH, N.; URBAN, A. An implementation of artificial neural-network potentials for atomistic materials simulations: Performance for TiO2. **Computational Materials Science**, Elsevier, v. 114, p. 135–150, 2016.

BACH, R. *et al.* Controlled double-slit electron diffraction. **New Journal of Physics**, IOP Publishing, v. 15, n. 3, p. 033018, 2013.

BEALE, M. H. *et al.* **Neural network toolbox™ user's guide**. Natick, MA, U.S.: MathWorks, 2010.

BEHLER, J. First principles neural network potentials for reactive simulations of large molecular and condensed systems. **Angewandte Chemie International Edition**, Wiley Online Library, v. 56, n. 42, p. 12828–12840, 2017.

BEHLER, J.; PARRINELLO, M. Generalized neural-network representation of high-dimensional potential-energy surfaces. **Physical review letters**, APS, v. 98, n. 14, p. 146401, 2007.

BELLMAN, R. A markovian decision process. **Journal of mathematics and mechanics**, JSTOR, p. 679–684, 1957.

BELLO, I. *et al.* Neural combinatorial optimization with reinforcement learning. **arXiv preprint**, 2016.

BÖHM, D. A suggested interpretation of the quantum theory in terms of"hidden"variables. i. **Physical Review**, v. 85, n. 2, p. 166–179, 1952.

BORN, M.; OPPENHEIMER, R. Zur quantentheorie der molekeln. **Annalen der physik**, Wiley Online Library, v. 389, n. 20, p. 457–484, 1927.

BREEN, P. G. *et al.* Newton vs the machine: solving the chaotic three-body problem using deep neural networks. **arXiv preprint**, 2019.

BRIGGS, G. *et al.* The oxford questions on the foundations of quantum physics. In: PROCEEDINGS OF THE ROYAL SOCIETY A: MATHEMATICAL, PHYSICAL AND ENGINEERING SCIENCES. [S.l.]: The Royal Society Publishing, 2013. v. 469, n. 2157, p. 20130299.

BROCKHERDE, F. *et al.* Bypassing the kohn-sham equations with machine learning. **Nature communications**, Nature Publishing Group, v. 8, n. 1, p. 1–10, 2017.

BROGLIE, L. de. Investigations on quantum theory. **Annales de Physique**, v. 3, p. 22, 1925.

BRUNK, E.; ROTHLISBERGER, U. Mixed quantum mechanical/molecular mechanical molecular dynamics simulations of biological systems in ground and electronically excited states. **Chemical reviews**, ACS Publications, v. 115, n. 12, p. 6217–6263, 2015.

BUSONIU, L. *et al.* **Reinforcement learning and dynamic programming using function approximators**. Boca Raton: CRC press, 2010. v. 39.

BUTKOV, E. **Mathematical physics, volume unico**. Massachusetts: Addison-Wesley, 1973.

BUTLER, K. T. *et al.* Machine learning for molecular and materials science. **Nature**, Nature Publishing Group, v. 559, n. 7715, p. 547–555, 2018.

CAPELLE, K. A bird's-eye view of density-functional theory. **Brazilian Journal of Physics**, SciELO Brasil, v. 36, n. 4A, p. 1318–1343, 2006.

CARPENTER, G. A. Distributed learning, recognition, and prediction by art and artmap neural networks. **Neural networks**, Elsevier, v. 10, n. 8, p. 1473–1494, 1997.

CARPENTER, G. A.; GROSSBERG, S. Adaptive resonance theory. In: SAMMUT, C.; WEBB, G. I. (Ed.). **Encyclopedia of machine learning and data mining**. 2. ed. New York: Springer Publishing Company, Incorporated, 2017.

CHEN, Y.-J. *et al.* Policy sharing using aggregation trees for q-learning in a continuous state and action spaces. **IEEE Transactions on Cognitive and Developmental Systems**, IEEE, 2019.

CHMIELA, S. *et al.* Towards exact molecular dynamics simulations with machine-learned force fields. **Nature communications**, Nature Publishing Group, v. 9, n. 1, p. 3887, 2018.

CHMIELA, S. *et al.* Machine learning of accurate energy-conserving molecular force fields. **Science advances**, American Association for the Advancement of Science, v. 3, n. 5, p. e1603015, 2017.

CLAESEN, M.; MOOR, B. D. Hyperparameter search in machine learning. **arXiv preprint**, 2015.

CLAPHAM, J. N. C. **The Concise Oxford Dictionary of Mathematics, Fourth Edition (Oxford Paperback Reference)**. 4. ed. Pondicherry: Oxford University Press, USA, 2009. (Oxford Paperback Reference). ISBN 0199235945,9780199235940.

COHEN, A. J. *et al.* Insights into current limitations of density functional theory. **Science**, v. 321, n. 5890, p. 792–794, 2008.

DEISENROTH, M. P. *et al.* A survey on policy search for robotics. **Foundations and Trends® in Robotics**, Now Publishers, Inc., v. 2, n. 1–2, p. 1–142, 2013.

DING, S. *et al.* Evolutionary artificial neural networks: a review. **Artificial Intelligence Review**, Springer, v. 39, n. 3, p. 251–260, 2013.

DOERR, S. *et al.* Htmd: high-throughput molecular dynamics for molecular discovery. **Journal of chemical theory and computation**, ACS Publications, v. 12, n. 4, p. 1845–1852, 2016.

DOLGIREV, P. E. *et al.* Machine learning scheme for fast extraction of chemically interpretable interatomic potentials. **AIP Advances**, AIP Publishing LLC, v. 6, n. 8, p. 085318, 2016.

DORIGO, M.; BIRATTARI, M. Ant colony optimization. In: SAMMUT, C.; WEBB, G. I. (Ed.). **Encyclopedia of machine learning and data mining**. 2. ed. New York: Springer Publishing Company, Incorporated, 2017.

DOYA, K. Reinforcement learning in continuous time and space. **Neural computation**, MIT Press, v. 12, n. 1, p. 219–245, 2000.

DUA, D.; GRAFF, C. **UCI Machine Learning Repository**. 2017. Disponível em: <http://archive.ics.uci.edu/ml>.

EISBERG, R.; RESNICK, R. **Física Quântica**. Rio de Janeiro: Editora Campus, 1979.

ELFWING, S. *et al.* Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. **Neural Networks**, Elsevier, v. 107, p. 3–11, 2018.

EVERITT, B.; SKRONDAL, A. **The Cambridge dictionary of statistics**. New York: Cambridge University Press Cambridge, 2002. v. 106.

FAIGL, J. Gsoa: growing self-organizing array-unsupervised learning for the close-enough traveling salesman problem and other routing problems. **Neurocomputing**, Elsevier, v. 312, p. 120–134, 2018.

FAULKNER, M. F. *et al.* All-atom computations with irreversible markov chains. **The Journal of chemical physics**, AIP Publishing LLC, v. 149, n. 6, p. 064113, 2018.

FEYNMAN, R. P. *et al.* Amplitudes de probabilidade. In: **Lições de física de Feynman: edição definitiva**. Porto Alegre: Bookman, 2008. v. 3, cap. 3.

FEYNMAN, R. P. *et al.* A dependência das amplitudes com a posição. In: **Lições de física de Feynman: edição definitiva**. Porto Alegre: Bookman, 2008. v. 3, cap. 16.

FISHER, R. A. The use of multiple measurements in taxonomic problems. **Annals of eugenics**, Wiley Online Library, v. 7, n. 2, p. 179–188, 1936.

GAMERMAN, D.; LOPES, H. F. Markov chains. In: **Markov chain Monte Carlo: stochastic simulation for Bayesian inference**. Boca Raton: Chapman and Hall/CRC, 2006. cap. 4.

GASKETT, C. *et al.* Q-learning in continuous state and action spaces. In: AUSTRALASIAN JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE. [S.l.], 1999. p. 417–428.

GERSTNER, W. Biological Learning: Synaptic Plasticity, Hebb Rule and Spike Timing Dependent Plasticity. In: SAMMUT, C.; WEBB, G. I. (Ed.). **Encyclopedia of machine learning and data mining**. 2. ed. New York: Springer Publishing Company, Incorporated, 2017.

GILKS, W. R. *et al.* **Markov chain Monte Carlo in practice**. Boca Raton: Chapman and Hall/CRC, 1995.

GOH, G. B. *et al.* Deep learning for computational chemistry. **Journal of computational chemistry**, Wiley Online Library, v. 38, n. 16, p. 1291–1307, 2017.

GOLDBERG, D. E. Genetic algorithms in search. **Optimization, and MachineLearning**, Addison Wesley Publishing Co. Inc., 1989.

GOODFELLOW, I. *et al.* Generative adversarial nets. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2014. p. 2672–2680.

GOTO, H. *et al.* Combinatorial optimization by simulating adiabatic bifurcations in nonlinear hamiltonian systems. **Science advances**, American Association for the Advancement of Science, v. 5, n. 4, p. eaav2372, 2019.

GREBOGI, C. *et al.* Chaos, strange attractors, and fractal basin boundaries in nonlinear dynamics. **Science**, American Association for the Advancement of Science, v. 238, n. 4827, p. 632–638, 1987.

GRIEBEL, M. *et al.* **Numerical Simulation in Molecular Dynamics**. Berlin, Heidelberg: Springer, 2007. ISBN 978-3-540-68094-9.

GUPTA, R. Data augmentation for low resource sentiment analysis using generative adversarial networks. In: ICASSP 2019-2019 IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING (ICASSP). Brighton, 2019. p. 7380–7384.

HAGAN, M. T.; MENHAJ, M. B. Training feedforward networks with the marquardt algorithm. **IEEE transactions on Neural Networks**, IEEE, v. 5, n. 6, p. 989–993, 1994.

HAMDI, H. *et al.* Population-based sampling methods for geological well testing. **Computational Geosciences**, Springer, v. 19, n. 5, p. 1089–1107, 2015.

HAMID, O. H.; BRAUN, J. Reinforcement learning and attractor neural network models of associative learning. In: INTERNATIONAL JOINT CONFERENCE ON COMPUTATIONAL INTELLIGENCE. Melbourne, 2017. p. 327–349.

HANSEN, K. *et al.* Assessment and validation of machine learning methods for predicting molecular atomization energies. **Journal of Chemical Theory and Computation**, ACS Publications, v. 9, n. 8, p. 3404–3419, 2013.

HANSSON, T. *et al.* Molecular dynamics simulations. **Current opinion in structural biology**, Elsevier, v. 12, n. 2, p. 190–196, 2002.

HARTIG, F. *et al.* Statistical inference for stochastic simulation models–theory and application. **Ecology letters**, Wiley Online Library, v. 14, n. 8, p. 816–827, 2011.

HASSELT, H. V. Reinforcement learning in continuous state and action spaces. In: **Reinforcement learning**. [S.l.]: Springer, 2012. p. 207–251.

HAYKIN, S. **Neural Networks and Learning Machines**. New Jersey: NJ, USA: Pearson, 2008.

HAYKIN, S. Self-Organizing Maps. In: **Neural Networks and Learning Machines**. New Jersey: NJ, USA: Pearson, 2008. cap. 9.

HEINEN, M. R.; ENGEL, P. M. An incremental probabilistic neural network for regression and reinforcement learning tasks. In: INTERNATIONAL CONFERENCE ON ARTIFICIAL NEURAL NETWORKS. Thessaloniki, Grécia, 2010. p. 170–179.

HEISENBERG, W. Über den anschaulichen inhalt der quantentheoretischen kinematik und mechanik. In: **Original Scientific Papers Wissenschaftliche Originalarbeiten**. [S.l.]: Springer, 1985. p. 478–504.

HERTZ, J. Attractor neural networks. In: SCOTT, A. (Ed.). **Encyclopedia of nonlinear science**. New York: Routledge, 2006.

HOFMANN, H. F. On the fundamental role of dynamics in quantum physics. **The European Physical Journal D**, Springer, v. 70, n. 5, p. 118, 2016.

HOHENBERG, P.; KOHN, W. Inhomogeneous electron gas. **Physical review**, APS, v. 136, n. 3B, p. B864, 1964.

HOPFIELD, J. J.; TANK, D. W. "neural" computation of decisions in optimization problems. **Biological cybernetics**, Springer, v. 52, n. 3, p. 141–152, 1985.

HOSPITAL, A. *et al.* Molecular dynamics simulations: advances and applications. **Advances and applications in bioinformatics and chemistry: AABC**, Dove Press, v. 8, p. 37, 2015.

HOWARD, R. A. **Dynamic programming and markov processes.** [S.l.]: John Wiley, 1960.

HUANG, T. *et al.* A review of combinatorial optimization with graph neural networks. In: 2019 5TH INTERNATIONAL CONFERENCE ON BIG DATA AND INFORMATION ANALYTICS (BIGDIA). Yunnan, China, 2019. p. 72–77.

ITEN, R. *et al.* Discovering physical concepts with neural networks. **Physical Review Letters**, APS, v. 124, n. 1, p. 010508, 2020.

JENNINGS, D.; LEIFER, M. No return to classical reality. **Contemporary Physics**, Taylor & Francis, v. 57, n. 1, p. 60–82, 2016.

JIN, F. *et al.* Corpuscular model of two-beam interference and double-slit experiments with single photons. **Journal of the Physical Society of Japan**, The Physical Society of Japan, v. 79, n. 7, p. 074401, 2010.

KAELBLING, L. P. *et al.* Reinforcement learning: A survey. **Journal of artificial intelligence research**, v. 4, p. 237–285, 1996.

KARPLUS, M.; MCCAMMON, J. A. Molecular dynamics simulations of biomolecules. **Nature Structural and Molecular Biology**, Nature Publishing Group, v. 9, n. 9, p. 646, 2002.

KASKI, S. Self-Organizing Maps. In: SAMMUT, C.; WEBB, G. I. (Ed.). **Encyclopedia of machine learning and data mining**. New York: Springer Publishing Company, Incorporated, 2017.

KIM, W. *et al.* Stereo matching using population-based mcmc. **International journal of computer vision**, Springer, v. 83, n. 2, p. 195–209, 2009.

KIREEVA, N. *et al.* Generative topographic mapping (GTM): universal tool for data visualization, structure-activity modeling and dataset comparison. **Molecular informatics**, Wiley Online Library, v. 31, n. 3-4, p. 301–312, 2012.

KOHN, W.; SHAM, L. J. Self-consistent equations including exchange and correlation effects. **Physical review**, APS, v. 140, n. 4A, p. A1133, 1965.

KOHONEN, T. Self-organized formation of topologically correct feature maps. **Biological cybernetics**, Springer, v. 43, n. 1, p. 59–69, 1982.

LANDAU, L. D.; LIFSHITZ, E. M. **Quantum Mechanics. Non-Relativistic Theory**. 3. ed. Oxford: Pergamon Press, 1977.

LEACH, A. R. **Molecular modelling: principles and applications**. Harlow, England: Pearson education, 2001.

LEI, Z. *et al.* Event-chain monte carlo with factor fields. **Physical Review E**, APS, v. 99, n. 4, p. 043301, 2019.

LENNARD-JONES, J. E. Cohesion. **Proceedings of the Physical Society**, IOP Publishing, v. 43, n. 5, p. 461, 1931.

LI, H. Z. *et al.* An accurate and efficient method to predict y-no bond homolysis bond dissociation energies. **Mathematical Problems in Engineering**, Hindawi, v. 2013, 2013.

LI, S. *et al.* Reinforcement learning neural network-based adaptive control for state and input time-delayed wheeled mobile robots. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, IEEE, 2018.

LILIENFELD, O. A. V. *et al.* Optimization of effective atom centered potentials for london dispersion forces in density functional theory. **Physical review letters**, APS, v. 93, n. 15, p. 153004, 2004.

LINSTROM, P.; MALLARD, W. (Ed.). **NIST Chemistry WebBook, NIST Standard Reference Database Number 69**. National Institute of Standards and Technology, 2020. Acesso em Dezembro de 2020. Disponível em: <https://physics.nist.gov/cuu/Constants/index.html>.

LITTMAN, M. L. Markov games as a framework for multi-agent reinforcement learning. In: **Machine learning proceedings 1994**. New Jersey: Elsevier, 1994. p. 157–163.

LOPEZ-BEZANILLA, A.; LILIENFELD, O. A. von. Modeling electronic quantum transport with machine learning. **Physical Review B**, APS, v. 89, n. 23, p. 235411, 2014.

MARQUARDT, D. W. An algorithm for least-squares estimation of nonlinear parameters. **Journal of the society for Industrial and Applied Mathematics**, SIAM, v. 11, n. 2, p. 431–441, 1963.

MARTIN, R. M. Density functional theory: foundations. In: **Electronic structure: basic theory and practical methods**. Cambridge: Cambridge university press, 2004. cap. 6.

MATER, A. C.; COOTE, M. L. Deep learning in chemistry. **Journal of chemical information and modeling**, ACS Publications, v. 59, n. 6, p. 2545–2559, 2019.

MCMAHON, J. M. *et al.* The properties of hydrogen and helium under extreme conditions. **Reviews of modern physics**, APS, v. 84, n. 4, p. 1607, 2012.

MERALI, Z. What is really real? **Nature**, Nature Publishing Group, v. 521, n. 7552, p. 278, 2015.

MILLS, K. *et al.* Deep learning and the schrödinger equation. **Physical Review A**, APS, v. 96, n. 4, p. 042113, 2017.

MILNOR, J. On the concept of attractor. In: **The theory of chaotic attractors**. [S.l.]: Springer, 1985. p. 243–264.

MOHR, P. J. *et al.* Codata recommended values of the fundamental physical constants: 2014. **Journal of Physical and Chemical Reference Data**, NIST, v. 45, n. 4, p. 043102, 2016.

MOLINEAUX, M. *et al.* Learning continuous action models in a real-time strategy environment. In: **FLAIRS Conference**. Florida: [s.n.], 2008. v. 8, p. 257–262.

MONTAVON, G. *et al.* Machine learning of molecular electronic properties in chemical compound space. **New Journal of Physics**, IOP Publishing, v. 15, n. 9, p. 095003, 2013.

MUNRO, P. Backpropagation. In: SAMMUT, C.; WEBB, G. I. (Ed.). **Encyclopedia of machine learning and data mining**. 2. ed. New York: Springer Publishing Company, Incorporated, 2017.

NETO, A. P. **Visualização Robusta de Atratores Estranhos**. Dissertação (Mestrado) — Instituto Nacional de Matemática Pura e Aplicada, Rio de Janeiro, 2004.

NORRIS, J. R. Discrete-time markov chains. In: **Markov chains**. Cambridge: Cambridge university press, 1998. cap. 1, p. 1–59.

PAZIS, J.; LAGOUDAKIS, M. G. Binary action search for learning continuous-action control policies. In: PROCEEDINGS OF THE 26TH ANNUAL INTERNATIONAL CONFERENCE ON MACHINE LEARNING. Montreal, 2009. p. 793–800.

PERILLA, J. R. *et al.* Molecular dynamics simulations of large macromolecular complexes. **Current opinion in structural biology**, Elsevier, v. 31, p. 64–74, 2015.

PERILLA, J. R. *et al.* All-atom molecular dynamics of virus capsids as drug targets. **The journal of physical chemistry letters**, ACS Publications, v. 7, n. 10, p. 1836–1844, 2016.

PEZZELLA, M. *et al.* Molecular oxygen formation in interstellar ices does not require tunneling. **The journal of physical chemistry letters**, ACS Publications, v. 9, n. 8, p. 1822–1826, 2018.

POPOVA, M. *et al.* Deep reinforcement learning for de novo drug design. **Science advances**, American Association for the Advancement of Science, v. 4, n. 7, p. eaap7885, 2018.

POROTTI, R. *et al.* Coherent transport of quantum states by deep reinforcement learning. **Communications Physics**, Nature Publishing Group, v. 2, n. 1, p. 1–9, 2019.

RAPAPORT, D. C. **The Art of Molecular Dynamics Simulation**. 1. ed. Cambridge: Cambridge University Press, 1995. ISBN 0-521-44561-2.

RAPAPORT, D. C. Simulating simple systems. In: **The Art of Molecular Dynamics Simulation**. 1. ed. Cambridge: Cambridge University Press, 1995. cap. 3. ISBN 0-521-44561-2.

RINGBAUER, M. *et al.* Measurements on the reality of the wavefunction. **Nature Physics**, Nature Publishing Group, v. 11, n. 3, p. 249, 2015.

ROBINSON, J. C. Attractors. In: SCOTT, A. (Ed.). **Encyclopedia of nonlinear science**. New York: Routledge, 2006.

ROSENBLATT, F. **The perceptron, a perceiving and recognizing automaton Project Para**. Bufalo: Cornell Aeronautical Laboratory, 1957.

RUBINSTEIN, R.; KROESE, D. **Simulation and the Monte Carlo Method**. New Jersey: John Wiley & Sons, Inc, Hoboken, 2017.

RUELLE, D. **Chaotic evolution and strange attractors**. Cambridge: Cambridge University Press, 1989. v. 1.

RUELLE, D.; TAKENS, F. On the nature of turbulence. **Les rencontres physiciens-mathématiciens de Strasbourg-RCP25**, v. 12, p. 1–44, 1971.

RUPP, M. *et al.* Fast and accurate modeling of molecular atomization energies with machine learning. **Physical review letters**, APS, v. 108, n. 5, p. 058301, 2012.

SAMMUT, C.; WEBB, G. I. **Encyclopedia of machine learning and data mining**. 2. ed. New York: Springer Publishing Company, Incorporated, 2017.

SANCHEZ-LENGELING, B. *et al.* Optimizing distributions over molecular space. an objective-reinforced generative adversarial network for inverse-design chemistry (organic). **ChemRxiv**, 2017.

SANDEFUR, J. T. Discrete dynamical modeling. **The College Mathematics Journal**, Taylor & Francis, v. 22, n. 1, p. 13–22, 1991.

SCHLICK, T. Theoretical and Computational Approaches to Biomolecular Structure. In: **Molecular modeling and simulation: an interdisciplinary guide**. 2. ed. New York: Springer Science & Business Media, 2010. v. 21, cap. 8.

SCHLOSSHAUER, M. Decoherence, the measurement problem, and interpretations of quantum mechanics. **Reviews of Modern physics**, APS, v. 76, n. 4, p. 1267, 2005.

SCHRÖDINGER, E. Annalen der physik. leipzig 79, 361 (1926); e. schrödinger. **Annalen der Physik. Leipzig**, v. 79, p. 489, 1926.

SEHGAL, A. *et al.* Deep reinforcement learning using genetic algorithm for parameter optimization. In: 2019 THIRD IEEE INTERNATIONAL CONFERENCE ON ROBOTIC COMPUTING (IRC). Napoles, Itália, 2019. p. 596–601.

SELLIER, J. M. *et al.* Machine learning and signed particles, an alternative and efficient way to simulate quantum systems. **International Journal of Quantum Chemistry**, Wiley Online Library, p. e26017, 2019.

SELVARAJ, C. *et al.* Molecular dynamics simulations and applications in computational toxicology and nanotoxicology. **Food and Chemical Toxicology**, Elsevier, v. 112, p. 495–506, 2018.

SIGMAN, M. E.; RIVES, S. S. Prediction of atomic ionization potentials i-iii using an artificial neural network. **Journal of Chemical Information and Computer Sciences**, ACS Publications, v. 34, n. 3, p. 617–620, 1994.

SILVA, L. E. B. da *et al.* A survey of adaptive resonance theory neural network models for engineering applications. **Neural Networks**, Elsevier, v. 120, p. 167–203, 2019.

SILVER, D. *et al.* Mastering the game of go with deep neural networks and tree search. **Nature**, Nature Publishing Group, v. 529, n. 7587, p. 484, 2016.

SNYDER, J. C. *et al.* Finding density functionals with machine learning. **Physical review letters**, APS, v. 108, n. 25, p. 253002, 2012.

SPRINGENBERG, J. Unsupervised and semi-supervised learning with categorical generative adversarial networks. corr. **arXiv preprint**, 2016.

STANLEY, K. O.; MIIKKULAINEN, R. Efficient reinforcement learning through evolving neural network topologies. In: PROCEEDINGS OF THE 4TH ANNUAL CONFERENCE ON GENETIC AND EVOLUTIONARY COMPUTATION. San Francisco, 2002. p. 569–577.

STEWART, J. **Cálculo, traduçao EZ2 Translate**. São Paulo: [s.n.], 2013. v. 2.

STONE, P. Reinforcement learning. In: SAMMUT, C.; WEBB, G. I. (Ed.). **Encyclopedia of machine learning and data mining**. 2. ed. New York: Springer Publishing Company, Incorporated, 2017.

SUTTON, R. S.; BARTO, A. G. **Reinforcement learning: An introduction**. London: MIT press, 2018.

TANK, D.; HOPFIELD, J. Simple'neural'optimization networks: An a/d converter, signal decision circuit, and a linear programming circuit. **IEEE transactions on circuits and systems**, IEEE, v. 33, n. 5, p. 533–541, 1986.

VATS, D. *et al.* Analyzing mcmc output. **arXiv preprint**, 2019.

VONDRÁŠEK, J. *et al.* Unexpectedly strong energy stabilization inside the hydrophobic core of small protein rubredoxin mediated by aromatic residues: correlated ab initio quantum chemical calculations. **Journal of the American Chemical Society**, ACS Publications, v. 127, n. 8, p. 2615–2619, 2005.

WANG, J.; PEREZ, L. The effectiveness of data augmentation in image classification using deep learning. **Convolutional Neural Networks Vis. Recognit**, p. 11, 2017.

WILSON, G.; PAWLEY, G. On the stability of the travelling salesman problem algorithm of hopfield and tank. **Biological Cybernetics**, Springer, v. 58, n. 1, p. 63–70, 1988.

WOLBERG, W. H.; MANGASARIAN, O. L. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. **Proceedings of the national academy of sciences**, National Acad Sciences, v. 87, n. 23, p. 9193–9196, 1990.

YOON, J. *et al.* Time-series generative adversarial networks. In: **Advances in Neural Information Processing Systems**. Vancouver: [s.n.], 2019. p. 5509–5519.

YOUNG, T. Lectures on natural philosophy, diffraction. **Philosophical Transactions of the Royal Society**, v. 92, p. 12–48, 1802.

ZHANG, J. *et al.* A comprehensive review on the molecular dynamics simulation of the novel thermal properties of graphene. **Rsc Advances**, Royal Society of Chemistry, v. 5, n. 109, p. 89415–89426, 2015.

ZHANG, L. *et al.* Deep potential molecular dynamics: a scalable model with the accuracy of quantum mechanics. **Physical review letters**, APS, v. 120, n. 14, p. 143001, 2018.

ZHANG, Q. *et al.* Review of recent achievements in self-healing conductive materials and their applications. **Journal of materials science**, Springer, v. 53, n. 1, p. 27–46, 2018.

ZHONG, S. *et al.* Efficient reinforcement learning in continuous state and action spaces with dyna and policy approximation. **Frontiers of Computer Science**, Springer, v. 13, n. 1, p. 106–126, 2019.

ZHOU, Z.-Y. *et al.* Quantum twisted double-slits experiments: confirming wavefunctions' physical reality. **Science bulletin**, Elsevier, v. 62, n. 17, p. 1185–1192, 2017.

ZIMMERLI, U. *et al.* Dispersion corrections to density functionals for water aromatic interactions. **The Journal of chemical physics**, American Institute of Physics, v. 120, n. 6, p. 2693–2699, 2004.

APÊNDICE A – BENCHMARKS E RESULTADOS

A.1 CONJUNTOS DE DADOS DE CLASSIFICAÇÃO

Nesta seção se encontram as descrições dos conjuntos de dados utilizados para validação da rede: *Iris* e o *Breast Cancer* (câncer de mama).

A.1.1 *Iris Data set*

O conjunto de dados Iris possui 150 instâncias, 4 atributos e 3 classes (FISHER, 1936; DUA; GRAFF, 2017). Os valores dos atributos são dados em centímetros.

Seus atributos são:

1. comprimento da sépala
2. largura da sépala
3. comprimento da pétala
4. largura da pétala

E as classes são:

1. *Iris setosa*
2. *Iris versicolor*
3. *Iris virginica*

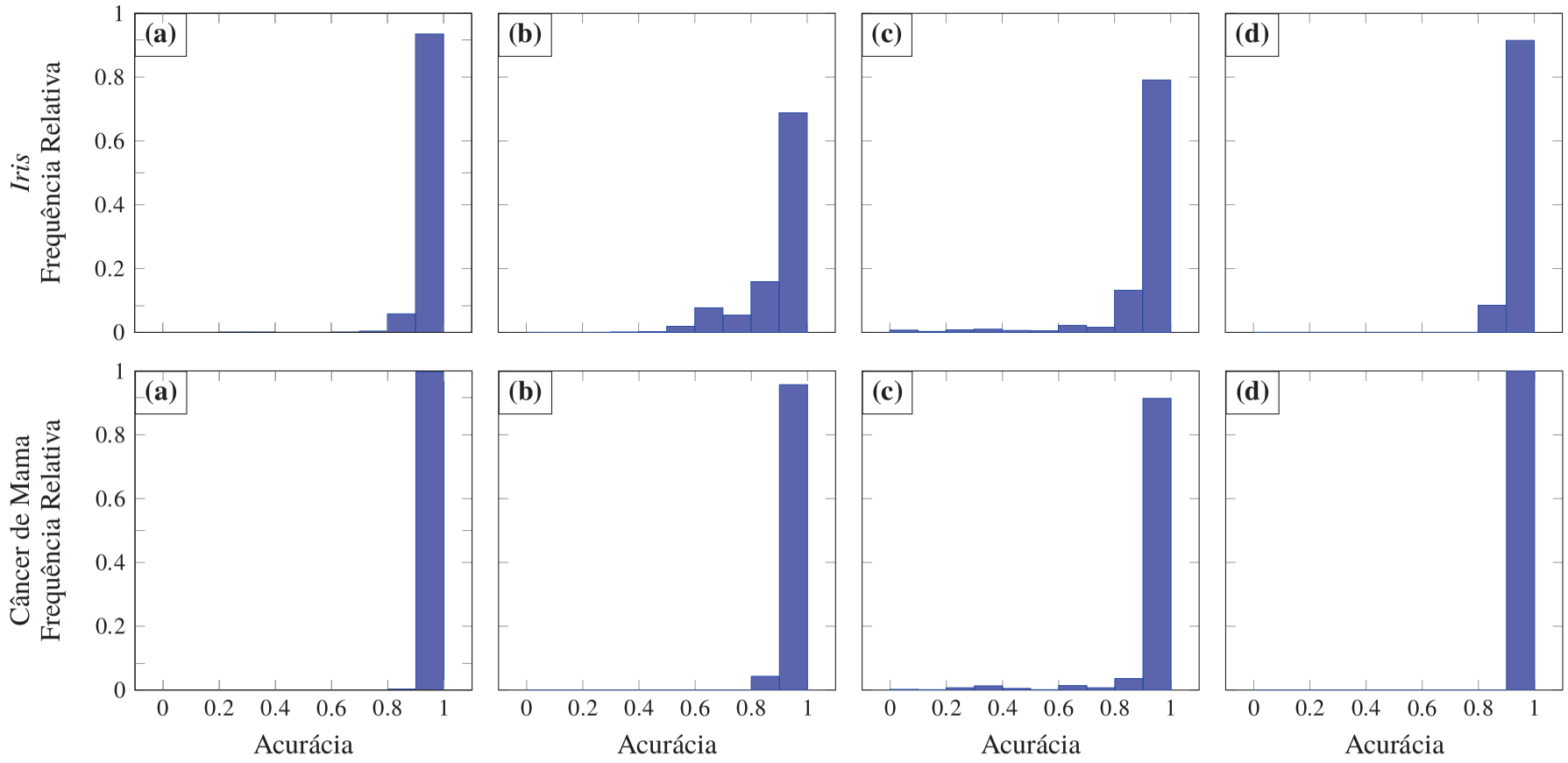
A.1.2 *Breast Cancer Wisconsin (Diagnostic) Data Set*

O conjunto de dados de Câncer de Mama possui 699 instâncias, das quais 683 completos (sem *"missing values"*), 9 atributos e 2 classes (WOLBERG; MANGASARIAN, 1990; DUA; GRAFF, 2017). As classes são (1) câncer benigno, e (2) câncer maligno. Seus atributos são:

1. Espessura do agrupamento (1-10)
2. Uniformidade do tamanho da célula (1-10)
3. Uniformidade da forma da célula (1-10)
4. Adesão marginal (1-10)
5. Tamanho de célula epitelial única (1-10)
6. Núcleos "desencapados" (1-10)
7. Cromatina suave (1-10)
8. Nucléolo normal (1-10)
9. Mitose (1-10)

A.2 DETALHAMENTO DOS RESULTADOS

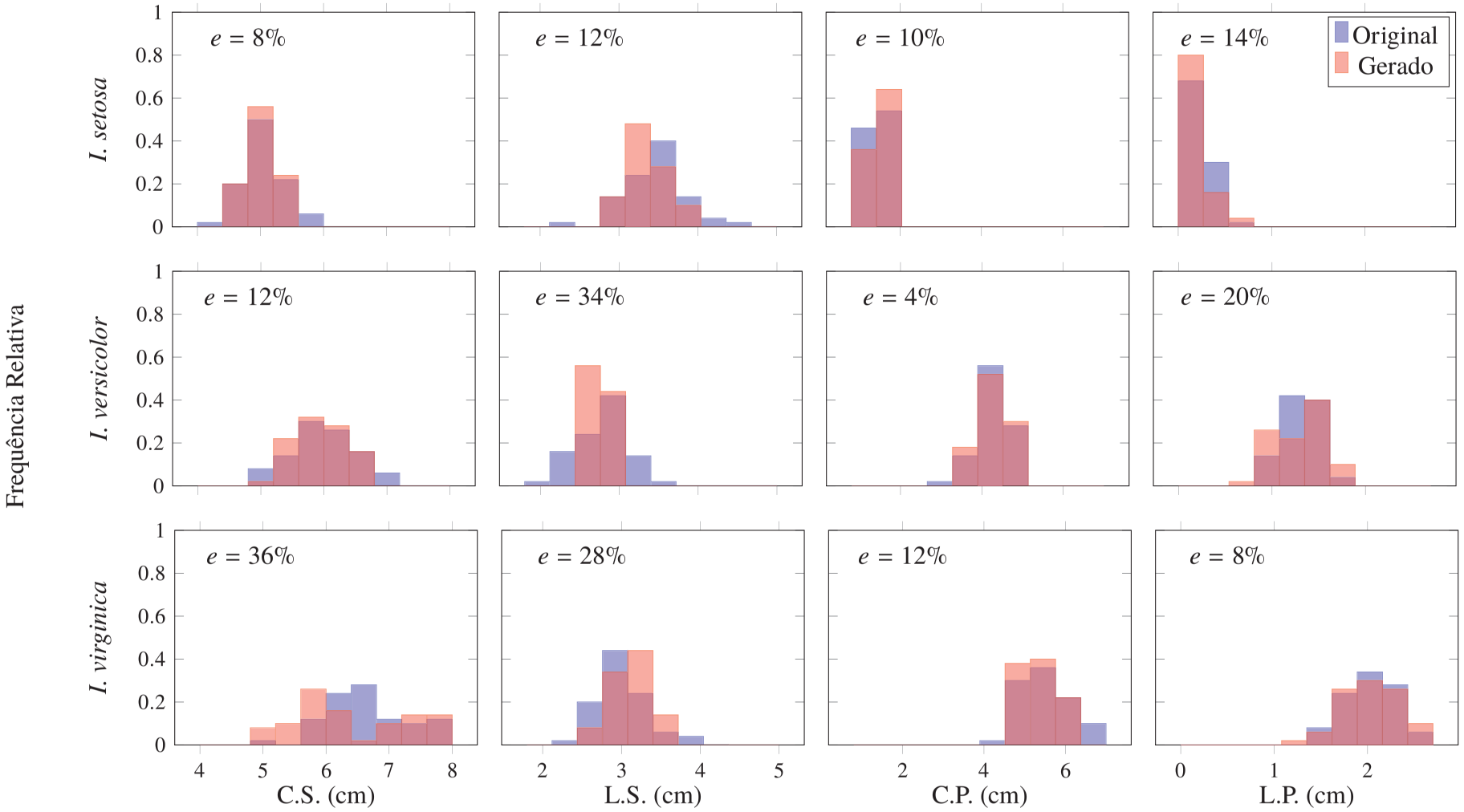
FIGURA A.1: ACURÁCIA DA CLASSIFICAÇÃO DO CONJUNTO DE DADOS *IRIS* E CÂNCER DE MAMA.



FONTE: O autor (2020).

LEGENDA: Os histogramas fornecem a frequência da taxa de acertos para a classificação da *Iris* e Câncer de Mama com 1000 réplicas cada, proporção 70:30. Na linhas de cima – dados da *Iris* – foram usadas a distribuição [3 2] de camadas ocultas, e na linha debaixo – dados de Câncer – distribuição [5 2]. (a) rede MLP, (b) FResA, (c) FResB, (d) FResC, conforme os hiperparâmetros apresentados na Tabela 6.2.

FIGURA A.2: COMPARAÇÃO ENTRE A DISTRIBUIÇÃO DOS DADOS ORIGINAIS DA *IRIS* E OS DADOS OBTIDOS PELA REDE FRES GERADORA DE INSTÂNCIAS.



FONTE: O autor (2020).

LEGENDA: Os dados foram obtidos com apenas uma iteração. Cada linha representa uma das espécies/classes de dados, e cada coluna um dos atributos. Os valores são dados em centímetros.

Abreviações: e - erro na sobreposição de histogramas, C.S - comprimento da sépala, C.P - comprimento da pétala, L.S - largura da sépala, L.P - largura da pétala.

APÊNDICE B – CÓDIGOS

ALGORITMO B.1 FUNÇÃO `GERAESFERARAND`.
Gera pontos aleatórios conforme a distribuição de Schrödinger no estado fundamental.

função GeraEsferaRand(N,raio)
    Geração de pontos aleatórios conforme a densidade de probabilidade de Schrödinger
    r = MonteCarlo(N,Schrodinger) // gero N raios conforme Schrödinger com raio máximo em `raio`. Vide Algoritmos 3.1 e B.2.
    x = rand(N, 3) · raio // gero N pontos, com 3 coordenadas, com limites em [-raio,raio]
    $y = |x| \leq$ raio
    tam = length(y) // quantos dos pontos gerados possuem raio inferior a `raio`
    enquanto tam < N **faça** // enquanto não atingir N pontos, continue
        x = rand(N − tam, 3) · raio // gero os pontos faltantes (N-tam)
        novos= $|x| \leq$ raio
        Concateno em y os novos pontos criados com raio menor que `raio`
        tam = length(y) // quantos dos pontos gerados?
    fim enquanto
    Converto y, com N pontos, em coordenadas angulares, e reservo os ângulos θ e ϕ.
    Crio matriz c_{sph} com os ângulos θ e ϕ e uno aos raios r gerados
    $c_{sph} = [\theta, \phi, r]$
    Converto c_{sph} para coordenadas cartesianas e armazeno em c_{cart}
    $[x, y, z]$ = sph2cart(θ, ϕ, r);
    $c_{cart} = [x, y, z]$
    output: c_{cart} e c_{sph} // devolve as N coordenadas cartesianas ou esféricas
fim função

ALGORITMO B.2 DENSIDADE DE PROBABILIDADE DE SCHRÖDINGER EM NÍVEL FUNDAMENTAL.

função Schrodinger(x)
    output: $p(x) = 4a_0x^2\left(\frac{1}{a_0}\right)^3 e^{(-2x/a_0)}$ // probabilidade p para o raio x
    onde $a_0 = 0.529$ é o raio de Bohr
fim função

ALGORITMO B.3 FUNÇÕES DE CUSTO A E B.

função FCustoA(inputdata,output,redeTS,param)
  para $i = 1$ até 15 **faça** // para um amostral de 15 pontos iniciais i
    para $t = 1$ até 100 **faça** // para 100 passos de iteração t
      $x_0 = \text{inputdata}_i$
      $x_t = \text{FResTS}(x_{t-1}, \text{redeTS})$ // Função de uso da rede
      onde x a trajetória prevista
    fim para
    $C_{H,r}$ – diferença entre os histogramas radiais de x e da referência
    C_{Bohr} – diferença das médias calculada de x e do referência
    $\text{custo}_i = C_{H,r} + C_{Bohr}$
  fim para
  output: $C = \sum_{i=1}^{5} \text{custo}_i$ – custo total
fim função

função FCustoB(inputdata,output,redeTS,param)
  para $t = 1$ até 5 **faça** // para 5 passos de iteração t
    $s_0 = \text{inputdata}$ // com todos os 200 pontos iniciais
    $s_t = \text{FResTS}(s_{t-1}, \text{redeTS})$ // Função de uso da rede
    onde s é a trajetória prevista para os 200 pontos por 5 iterações
  fim para
  $C_{H,r}$ – diferença entre os histogramas radiais de s e da referência
  C_{Bohr} – diferença das médias calculada de s e do referência
  output: $C = C_{H,r} + C_{Bohr}$ – custo total
fim função

ALGORITMO B.4 RADIALMENTE DISTRIBUÍDO.

função RadialmenteDistrib(r, θ, ϕ)
  erro = 0
  N_{part} = 20
  se $\sum |r| > 0$ **então**
    int = $[1.05 \cdot \max(r) - 0.95 \cdot \min(r)]/N_{part}$ // int = tamanho do intervalo
    ints = $[\min(r) \cdot 0.95 : int : \max(r) \cdot 1.05]$ // intervalos
  fim se
  para i = 1 até Nfatias **faça** // verificação de cada esfera concêntrica
    idx = find((r>ints(i))==(r<=ints(i+1))) // Tome o índice dos pontos no intervalo especificado
    N = length(idx)
    $H\theta$ = histogramaNormalizado$(\cos\theta(idx))$
    $H\phi$ = histogramaNormalizado$(\cos\phi(idx))$
    erro = erro + $\sum |H\theta - H\theta_0|$ + $\sum |H\phi - H\phi_0|$
  fim para
  output: erro
fim função

ALGORITMO B.5 FUNÇÃO DE CUSTO MODELO.
Considerando uma entrada tipo coordenadas esféricas, o pseudocódigo é apresentado.

função FunçãoCusto(inputdata,output,redeTS,param) // 'param' – outros parâmetros de entrada
  para $i = 1$ até N **faça** // para cada ponto inicial fornecido
    para $t = 1$ até t_{sim} **faça** // Itero t_{sim} para obter a trajetória, traj, do elétron
      traj_t = FResTS(PoiçãoAtual,FResAtual) // uso a rede com os pesos da atual geração FResAtual
      PoiçãoAtual = traj_t // atualizo a posição atual
    fim para
    Calculo histogramas radial e angulares normalizados
$H_{r,prev} = \text{hist}(r, [0 : .1 : 3])$ // utilizando os limites do histograma como [0:0.1:3].
$H_{\theta,prev} = \text{hist}(\cos(\theta), [-1 : 1/15 : 1])$ // utilizando os limites do histograma como [-1:1/15:1].
$H_{\phi,prev} = \text{hist}(|\cos(\phi)|, [-1 : 1/15 : 1])$

    Calculo erro dos histogramas radial e angulares normalizados obtidos H_{prev} e os de referência H_{ref}
$C_{Hr} = \frac{1}{2}\sum_{l=1}^{n} |H_{r,ref} - H_{r,prev}|$
$C_{H\theta} = \frac{1}{2}\sum_{l=1}^{n} |H_{\theta,ref} - H_{\theta,prev}|$
$C_{H\phi} = \frac{1}{2}\sum_{l=1}^{n} |H_{\phi,ref} - H_{\phi,prev}|$
onde l é cada barra das n barras dos histogramas.
    Cálculo do desvio padrão σ entre os histogramas conforme equação 5.6
$C_{\sigma} = \sum_{l=1}^{n} \sigma(H_{r,ref}, H_{r,prev}) + \sum_{l=1}^{n} \sigma(H_{\theta,ref}, H_{\theta,prev}) + \sum_{l=1}^{n} \sigma(H_{\phi,ref}, H_{\phi,prev})$
    Verificar a diferença entre o raio médio esperado $3/2a_0$ e o raio médio previsto $|\vec{v}|$
$\frac{3}{2}a_0 = 0.7935$ Å
$C_{Bohr} = \left|\frac{|\vec{v}|-0.7935}{0.7935}\right|$
    Calcular a uniformidade da distribuição conforme o Algoritmo B.4, e comparar com um valro de refêrencia γ_{ref} de uma amostra uniforme
$\gamma_{prev} = \text{RadialmenteDistrib}(r, \theta, \phi)$

    se $\gamma_{prev} > \gamma_{ref}$ **então** // somente se o previsto for menos uniforme que o valor de referência
$C_{uniform} = \gamma_{prev} - \gamma_{ref}$
    fim se
    Converto $\vec{v}$ de coordenadas esféricas para cartesianas
$(r, \theta, \phi) \rightarrow (x, y, z)$
    Calculo do desvio do centro de Massa
$C_{CM} = \frac{1}{3t_{sim}}\sum_{t=1}^{t_sim}(x_t + y_t + z_t)$ // normalizado
    Determino o custo total
$C_i = C_{CM} + C_{Hr} + C_{H\theta} + C_{H\phi} + C_{\sigma} + CBohr + C_{uniform}$
  fim para
  output: $\sum_{i=1}^{N} C_i$
fim função

ALGORITMO B.6 TREINAMENTO DA REDE `FRES`.
Detalhes no fluxograma da Figura 4.2.

options: parâmetros fornecidos à função de treinamento da rede FResTR

n_{iter}: número de iterações da rede

n_{layers}: camadas da rede

r_{esson}: com ou sem ressonância

b_{ias}: com ou sem bias

optionsGA: parâmetros fornecidos ao GA

n_{ger}: número de gerações da GA

m_{rate}: taxa de mutação

n_{elite}: tamanho da elite

n_{pop}: tamanho da população

optionsSA: parâmetros fornecidos ao SA alternativo ao GA

n_{iterSA}: número de gerações da SA

t_{emp}: algoritmo de temperatura

inputdata: Dados de entrada

função FResTR(inputdata,options,optionsGA)
Função de treinamento da rede FRes
    define o número de pesos p a otimizar, a partir do número de camadas
    para $i = 1$ até n_{iter} **faça** // otimiza os pesos n_{iter} vezes
        pesos = AlgoritmoGenético(p,optionsGA,FunçãoChamada) // vide Algoritmo 5.1
    fim para
    output: redeTreinada
fim função

---

função FunçãoChamada(NomeFunçãoCusto,inputdata,pesos)
    gera variável redeTS: pesos, camadas, r_{eson},
    output = propaganet(input,n_{layers},pesos,r_{eson},b_{ias})
    custo = FunçãoCusto(inputdata,output,redeTS,param) // vide Algoritmo B.5
    output: custo
fim função

---

função propaganet(input,n_{layers},pesos,r_{eson},b_{ias})
função de propagação – a rede em si
    para $j = 1$ até n_{layers} **faça**
        para $k = 1$ até n_j **faça** // onde n_j é o número de nós da camada j
            se reson = 1 **então**
                $x_{j,k} = \sum_{i=1}^{n_{j-1}} \gamma_{j,k}\,\mathrm{sen}(\alpha_{j,k} x_{j-1,i} + \beta_{j,k})$
                onde $x_{j-1,i}$ é o valor do i-ésimo nó da camada anterior, e α, β, γ são os pesos da camada j nó k atuais. O *input* corresponde à camada $j = 1$.
            senão
                $x_{j,k} = \sum_{i=1}^{n_{j-1}} \alpha_{j,k} x_{j-1,i}$
            fim se
            se bias = 1 **então** $x_{j,k} = b_{j,k} + x_{j,k}$
            fim se
        fim para
    fim para
    output: valor da camada de saída $l = n_{layers}$
fim função

APÊNDICE C – MELHORES REDES

Apresentam-se aqui os gráficos e informações das redes com melhor desempenho treinadas neste trabalho. Listam-se abaixo as redes de melhor desempenho:

1. FRes07B_49;
2. FRes09B_28;
3. FRes09B_47;
4. FRes12B_11;
5. FRes12B_29;
6. FRes12B_31;
7. FRes12B_34;
8. FRes13A_31;
9. FRes14A_29;
10. FRes14B_4;
11. FRes14B_13;
12. FRes14B_34;

Legenda das Figuras: , C.1 , C.2 , C.3 , C.5 , C.7 , C.6 , C.4 , C.8 , C.9 , C.10 , C.11 , C.12.

Após uma iteração de 10^4 pontos, calculamos os CM a cada 100 pontos da iteração e apresentamos seu *plot* nos planos (a) xy e (b) xz, e (c) a distribuição do módulo do CM evidenciando seu desvio da origem, também representado como um '+' vermelho em (a) e (b). Do elétron iterado obtivemos também a distribuição das coordenadas esféricas (d) raio r, (e) $\cos(\theta)$ e (f) $|\cos(\phi)|$, comparado às distribuições de referência representadas pelas curvas em vermelho.

Legenda das Figuras: , C.13 , C.14 , C.15 , C.17 , C.19 , C.18 , C.16 , C.20 , C.21 , C.22 , C.23 , C.24.

A partir de dados aleatórios fracionados em esferas ocas concêntricas de raio de 0.5 Å, foram obtidos em (a) uma iteração da posição do elétron, (b) duas iterações e (c) três iterações. Em vermelho estão os dados de entrada (fornecido apenas para a primeira iteração) e em azul a saída da rede em cada passo de iteração. Freq.Rel. – frequência relativa.

FIGURA C.1: DISTRIBUIÇÕES DAS MELHORES REDES – FRES07B#49..

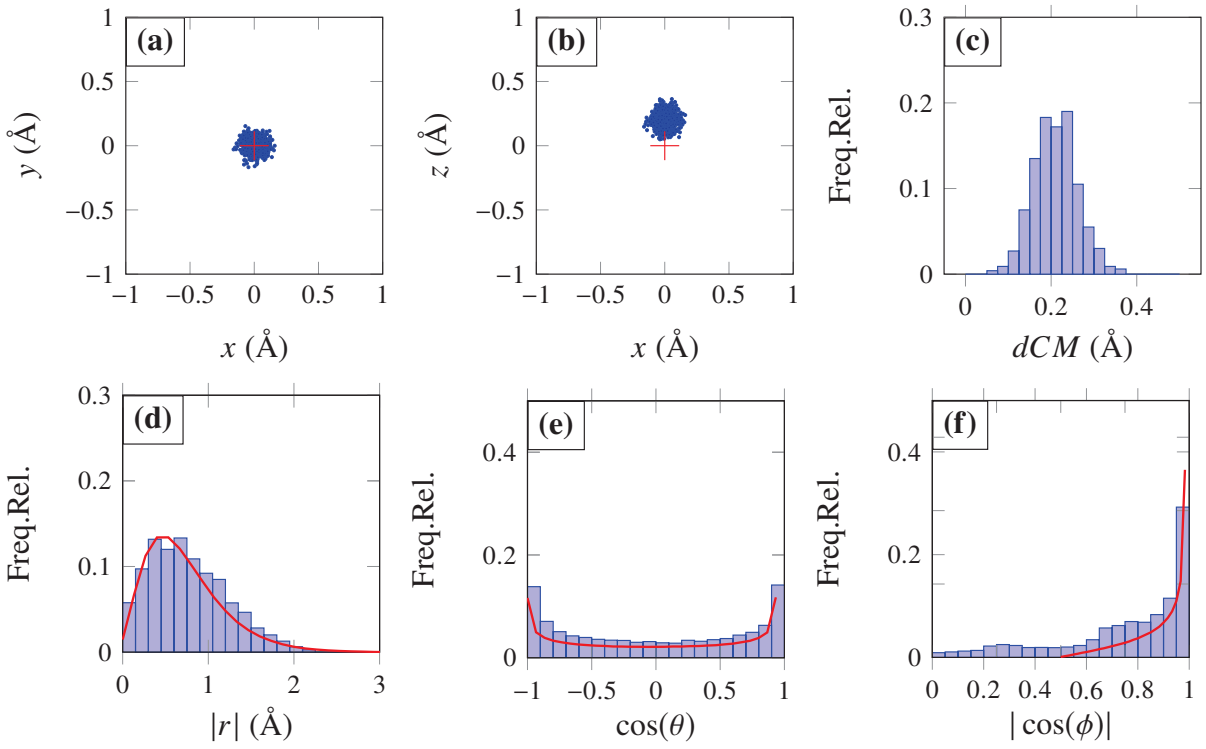


FIGURA C.2: DISTRIBUIÇÕES DAS MELHORES REDES – FRES09B#28.

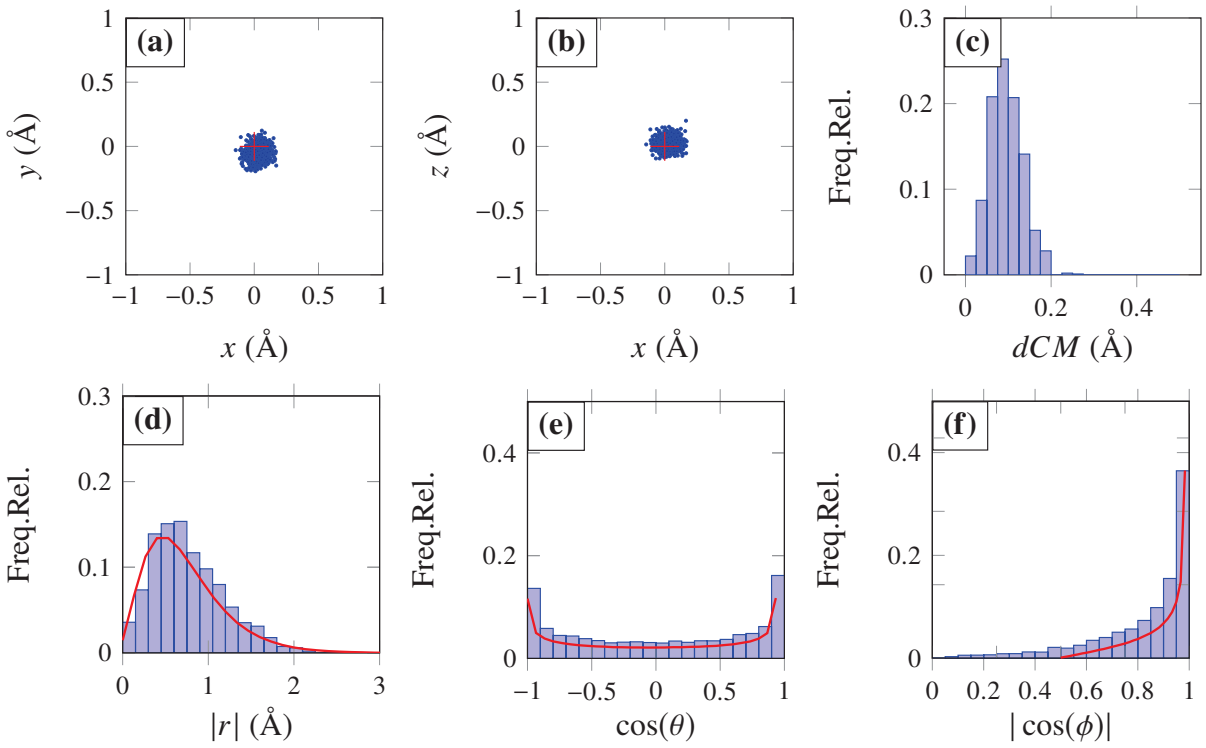


FIGURA C.3: DISTRIBUIÇÕES DAS MELHORES REDES – FRES09B#47.

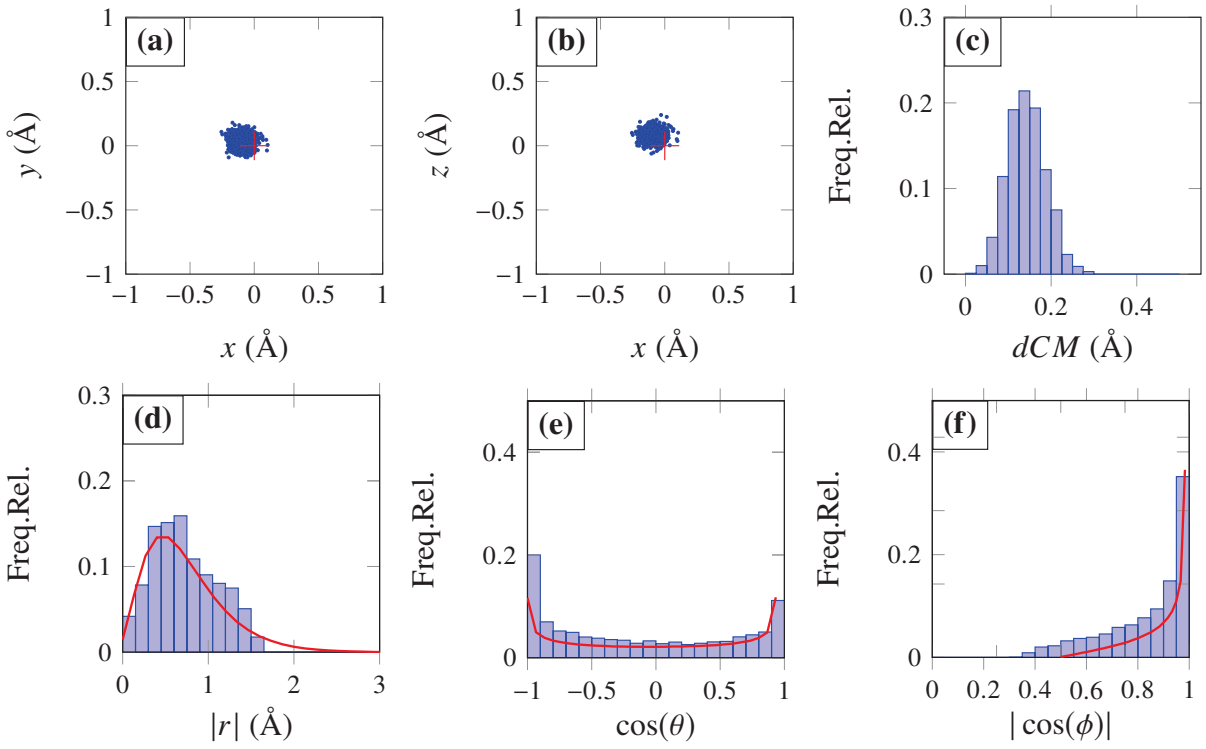


FIGURA C.4: DISTRIBUIÇÕES DAS MELHORES REDES – FRES12B#34.

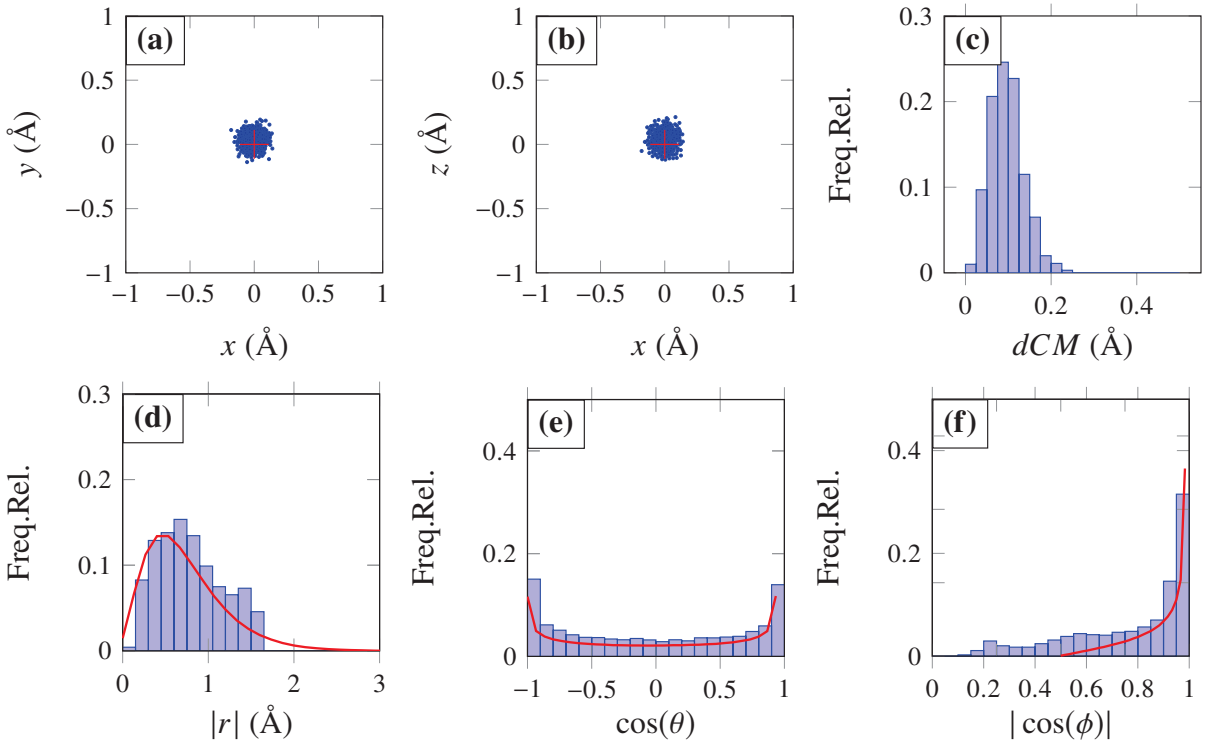


FIGURA C.5: DISTRIBUIÇÕES DAS MELHORES REDES – FRES12B#11.

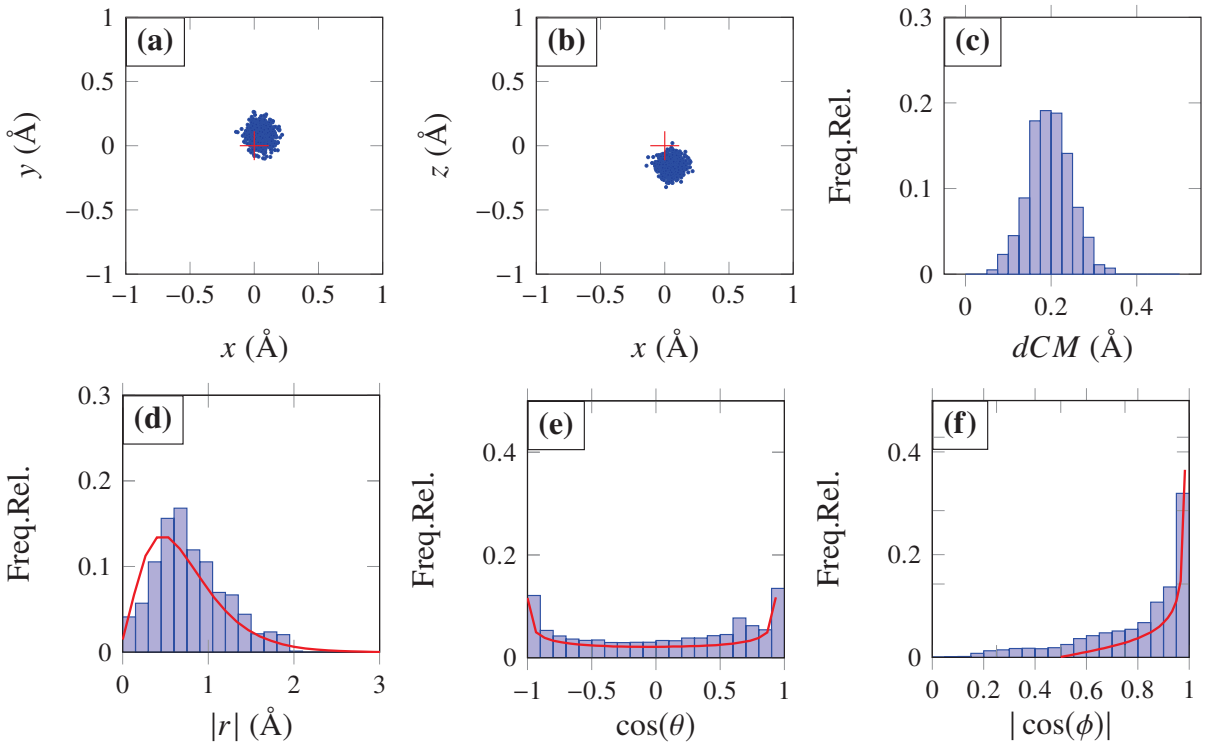


FIGURA C.6: DISTRIBUIÇÕES DAS MELHORES REDES – FRES12B#31.

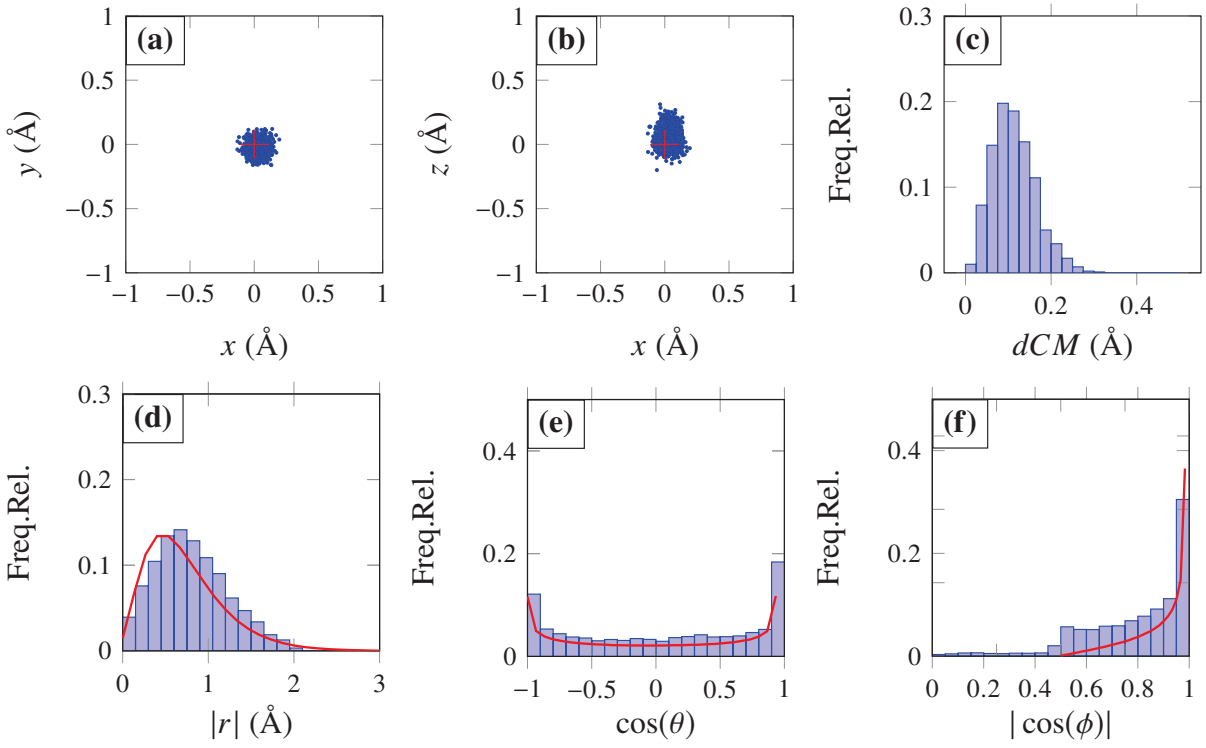


FIGURA C.7: DISTRIBUIÇÕES DAS MELHORES REDES – FRES12B#29.

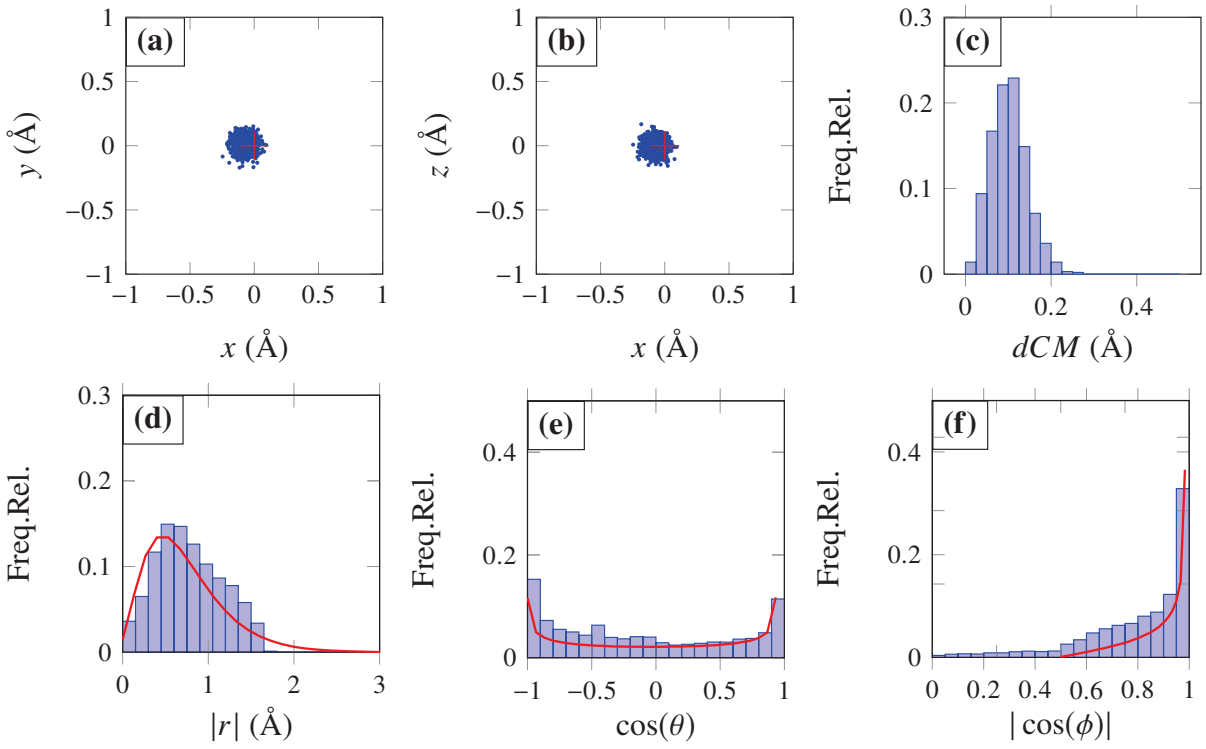


FIGURA C.8: DISTRIBUIÇÕES DAS MELHORES REDES – FRES13A#31.

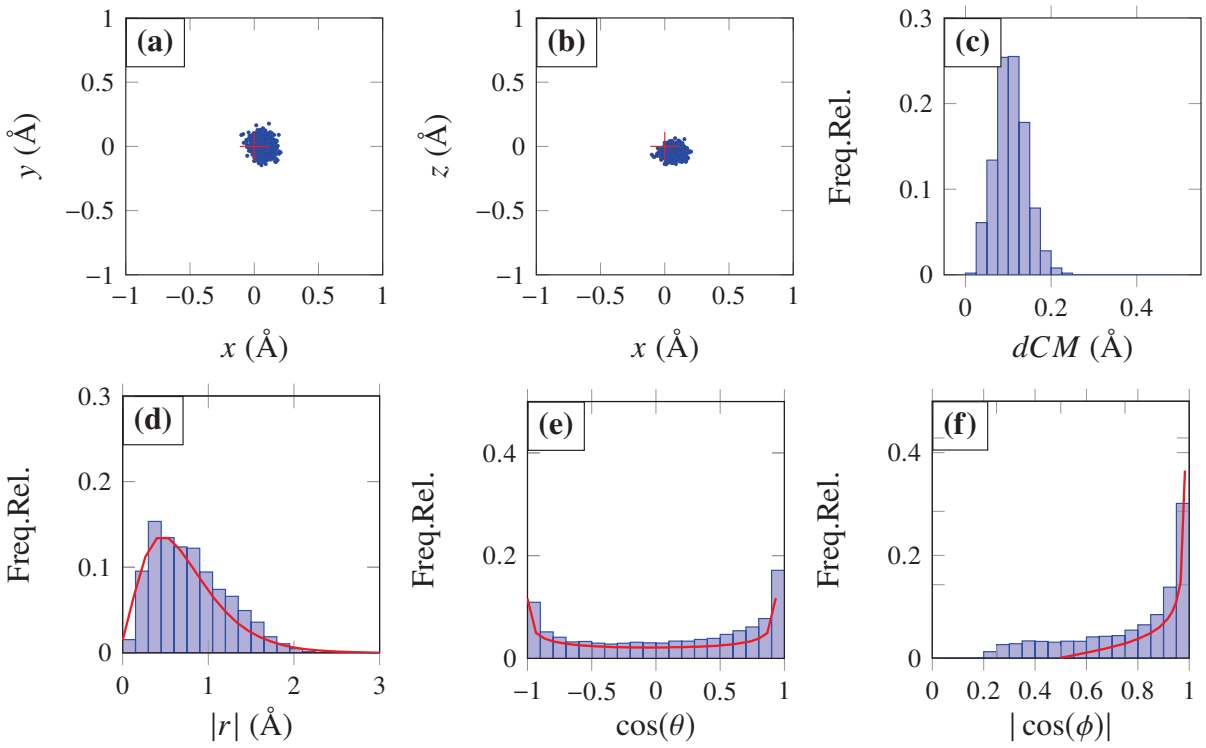


FIGURA C.9: DISTRIBUIÇÕES DAS MELHORES REDES – FRES14B#29.

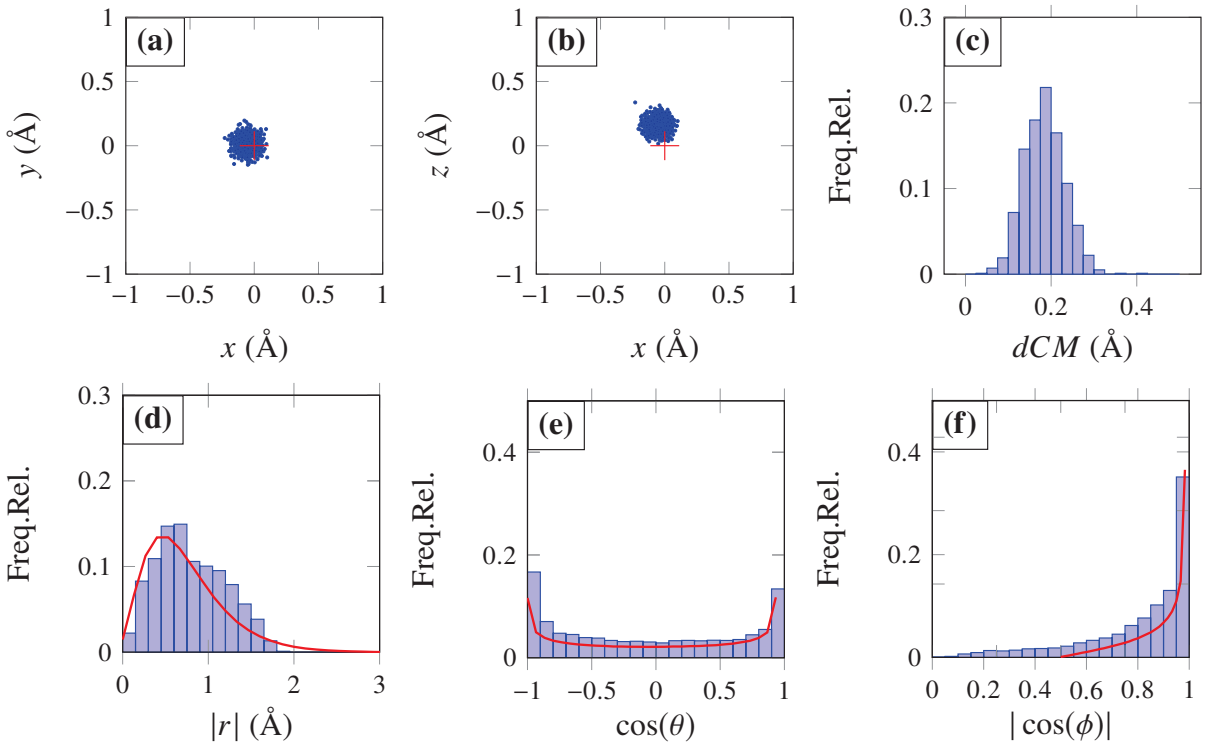


FIGURA C.10: DISTRIBUIÇÕES DAS MELHORES REDES – FRES14B#4.

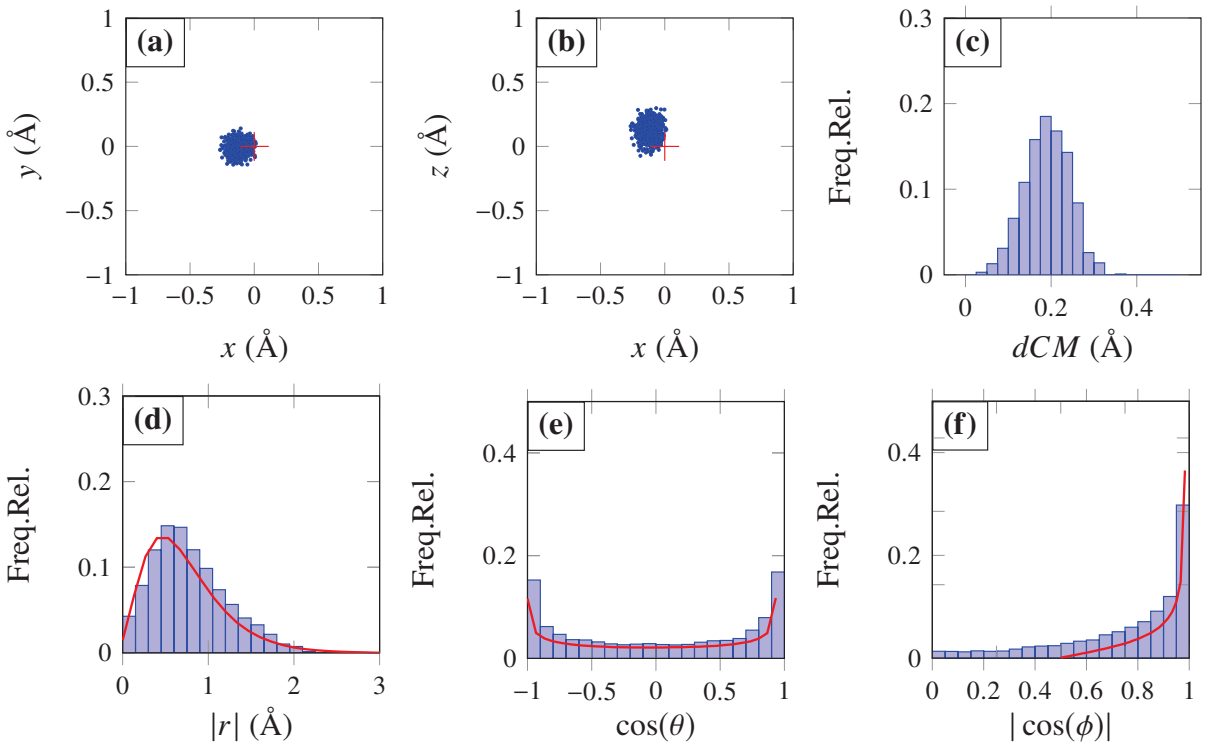


FIGURA C.11: DISTRIBUIÇÕES DAS MELHORES REDES – FRES14B#13.

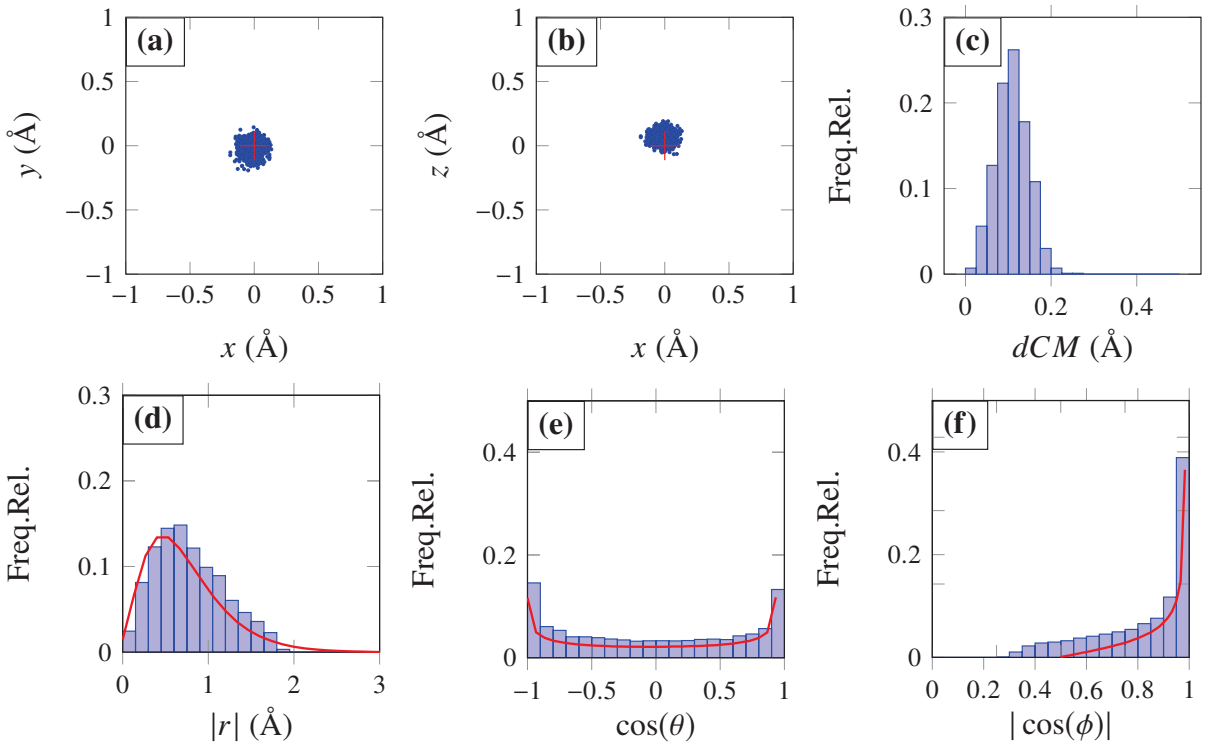


FIGURA C.12: DISTRIBUIÇÕES DAS MELHORES REDES – FRES14B#34.

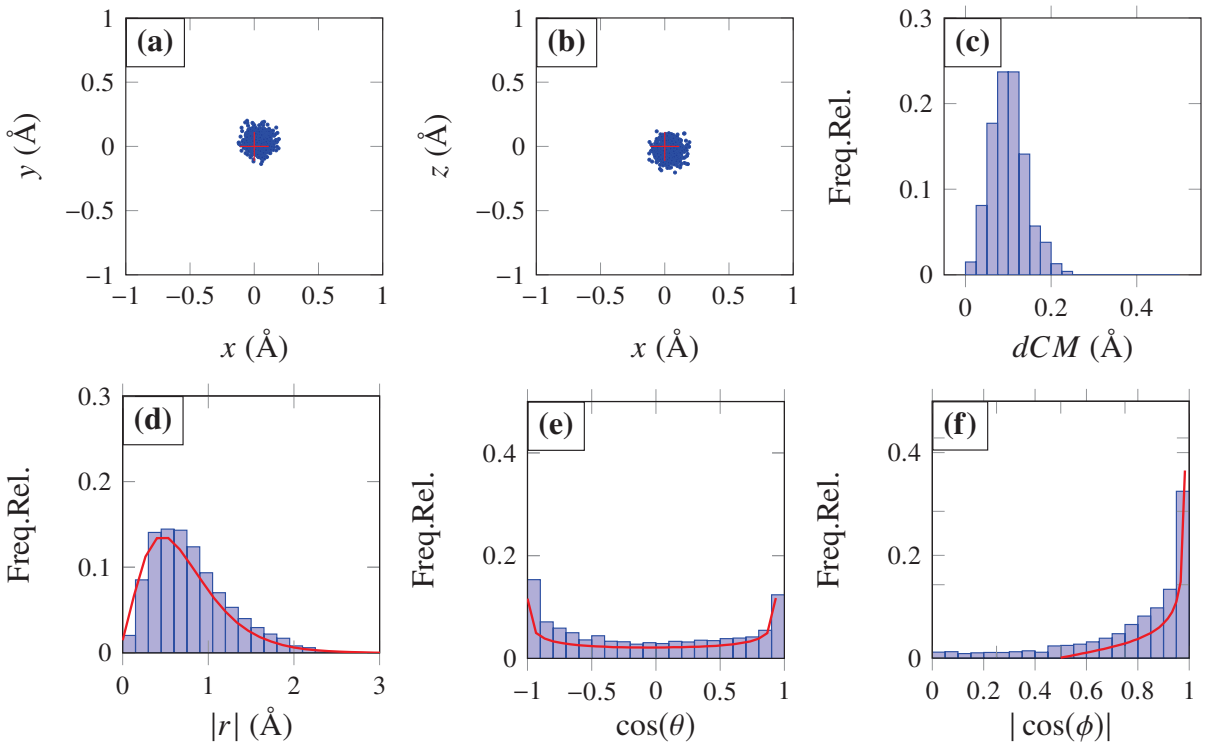


FIGURA C.13: ENTRADA DE ESFERAS CONCÊNTRICAS – FRES07B#49.

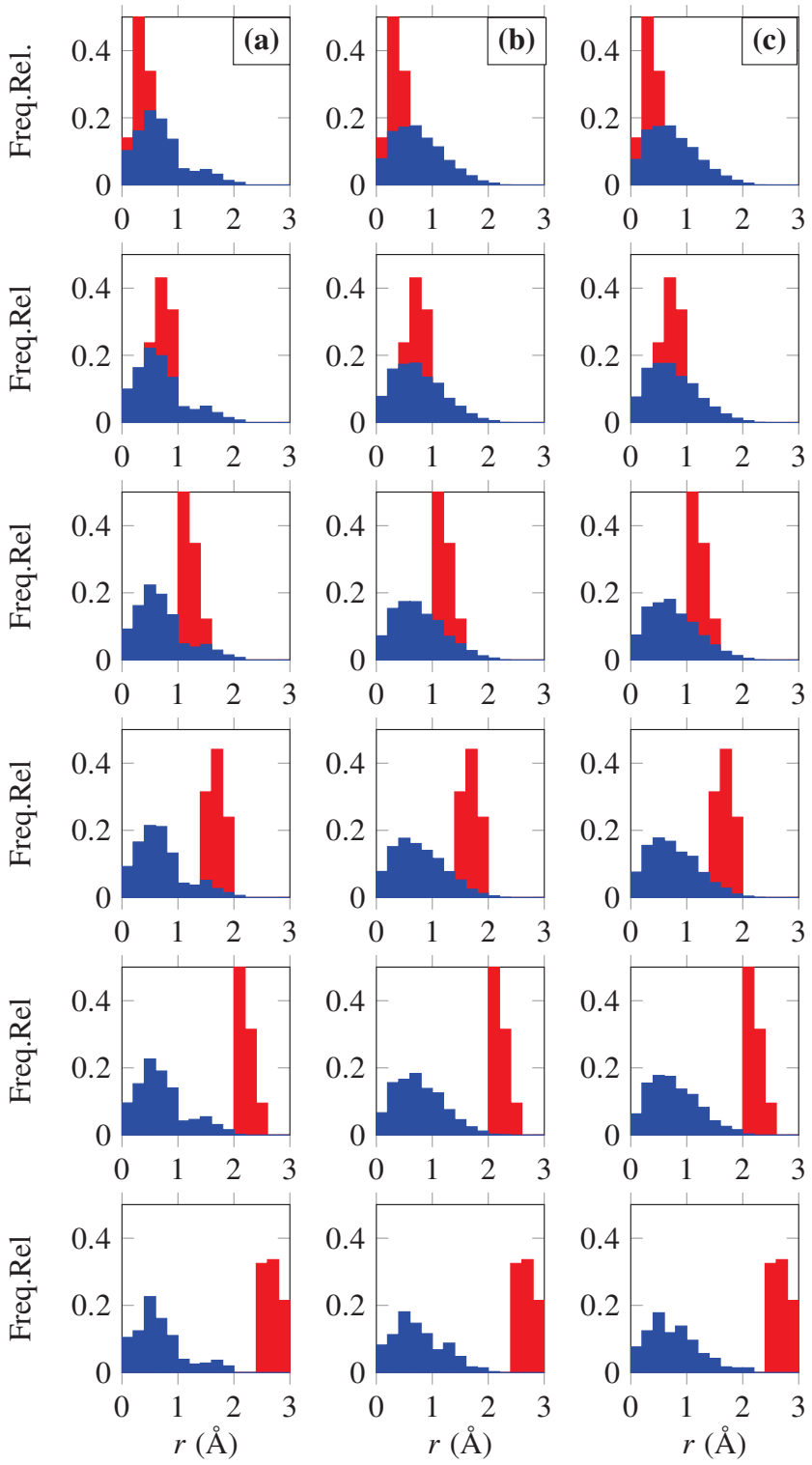


FIGURA C.14: ENTRADA DE ESFERAS CONCÊNTRICAS – FRES09B#28.

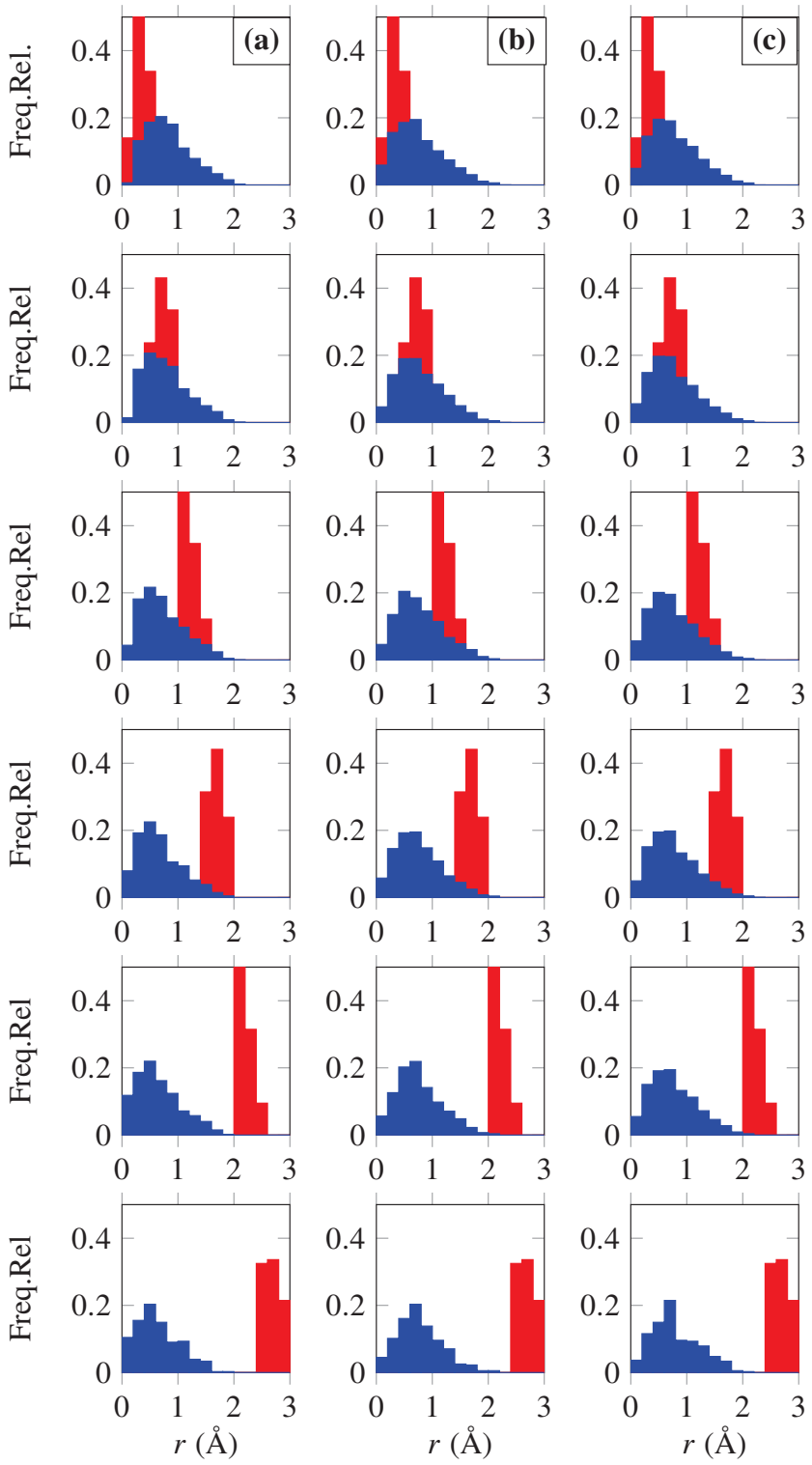


FIGURA C.15: ENTRADA DE ESFERAS CONCÊNTRICAS – FRES09B#47.

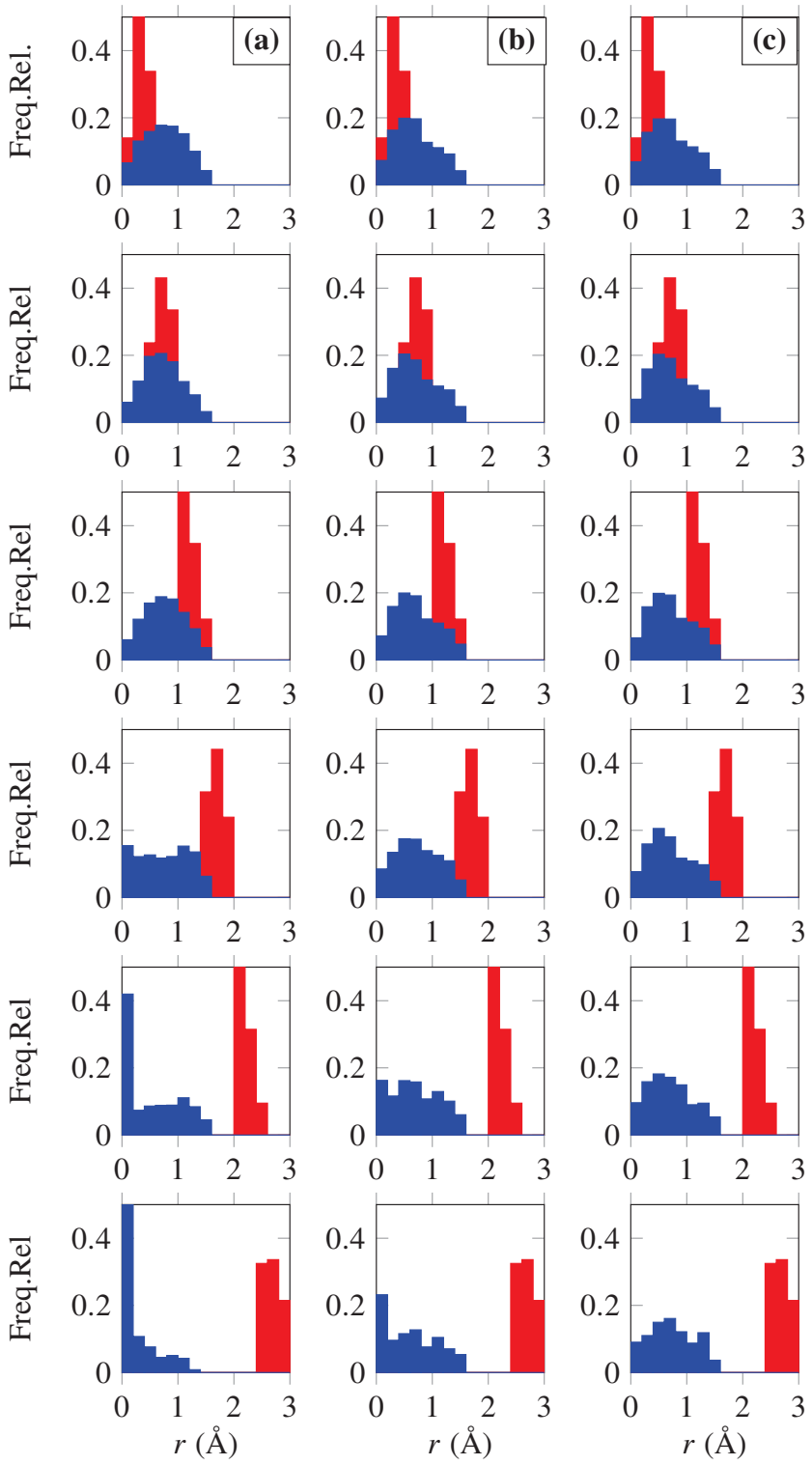


FIGURA C.16: ENTRADA DE ESFERAS CONCÊNTRICAS – FRES12B#34..

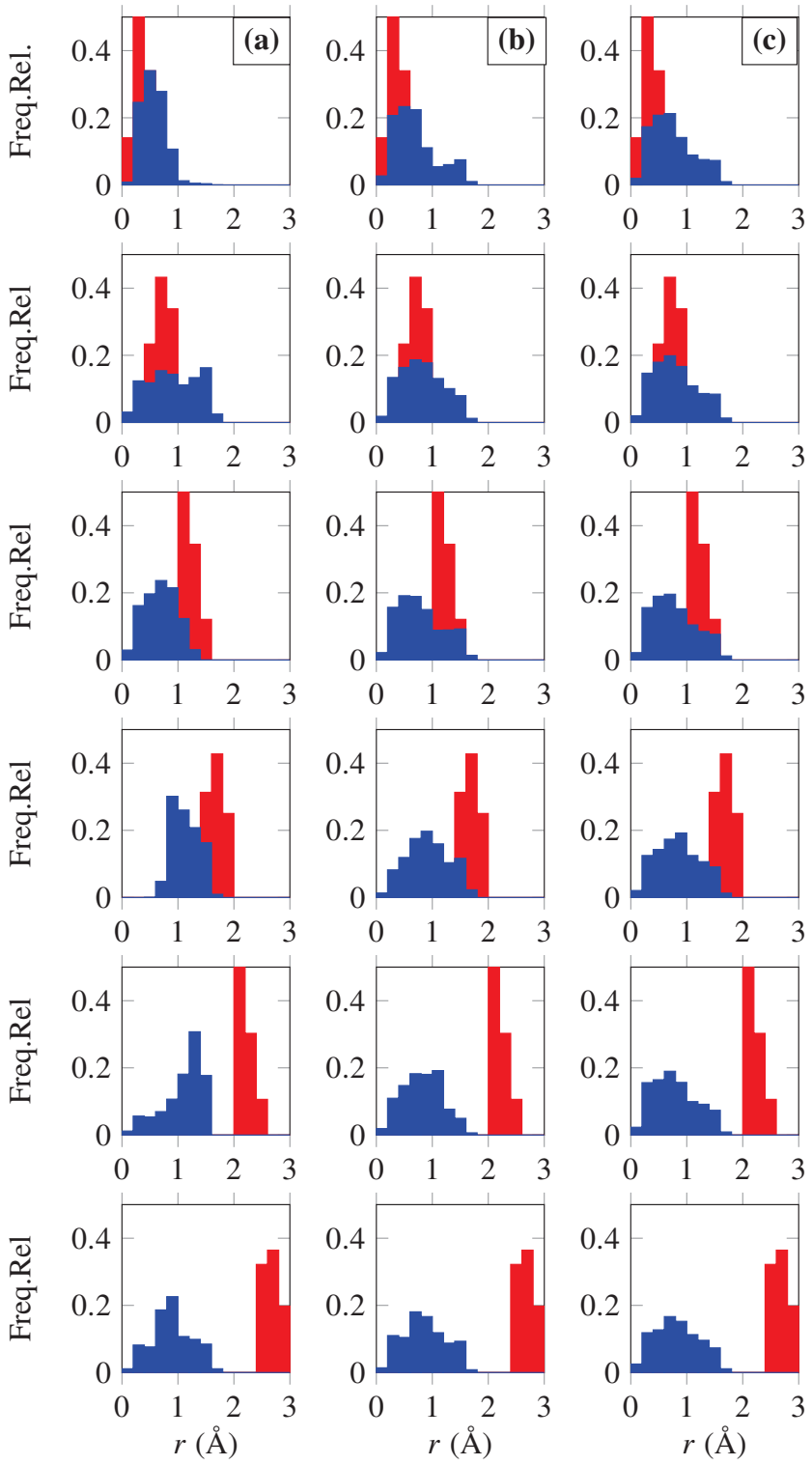


FIGURA C.17: ENTRADA DE ESFERAS CONCÊNTRICAS – FRES12B#11.

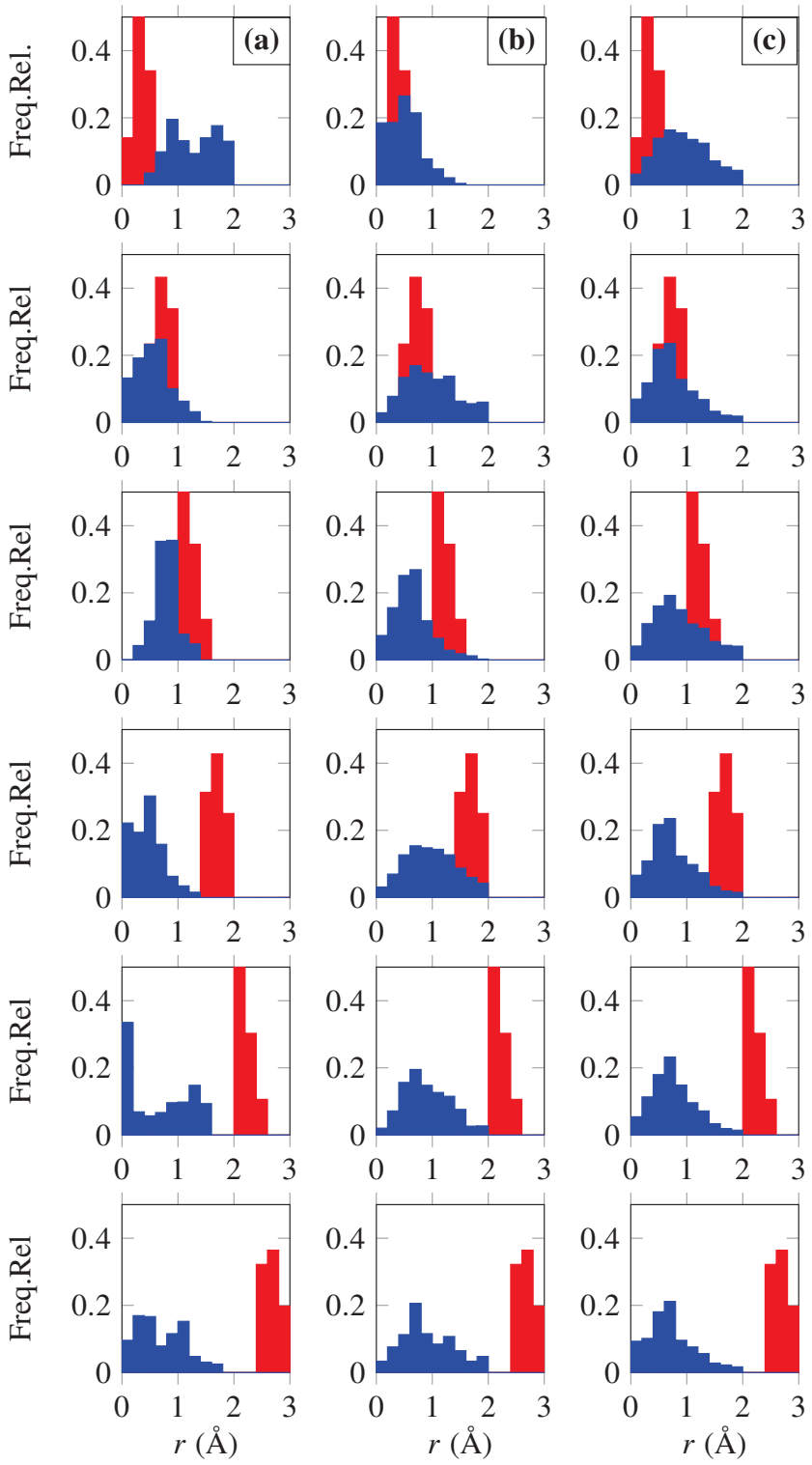


FIGURA C.18: ENTRADA DE ESFERAS CONCÊNTRICAS – FRES12B#31.

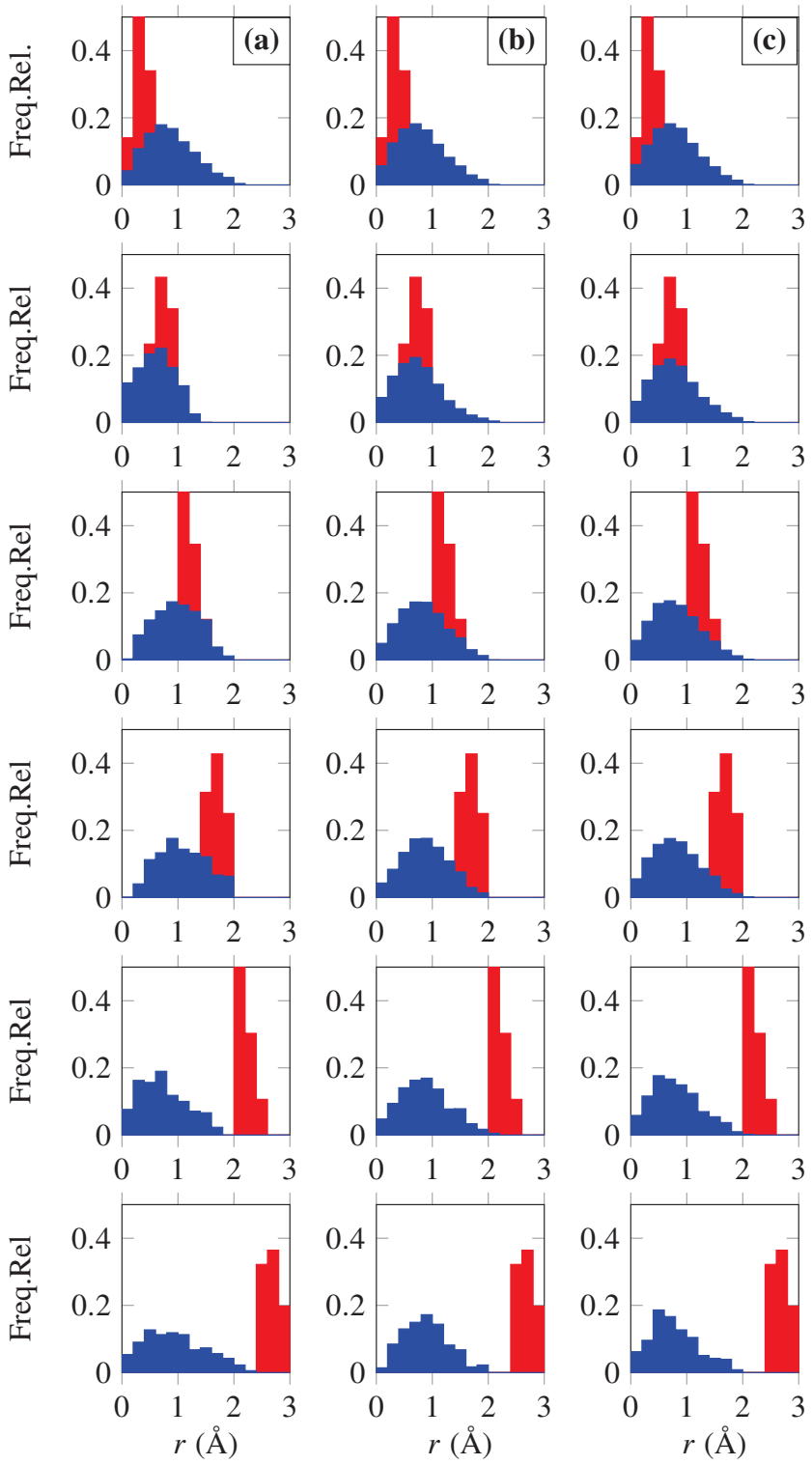


FIGURA C.19: ENTRADA DE ESFERAS CONCÊNTRICAS – FRES12B#29.

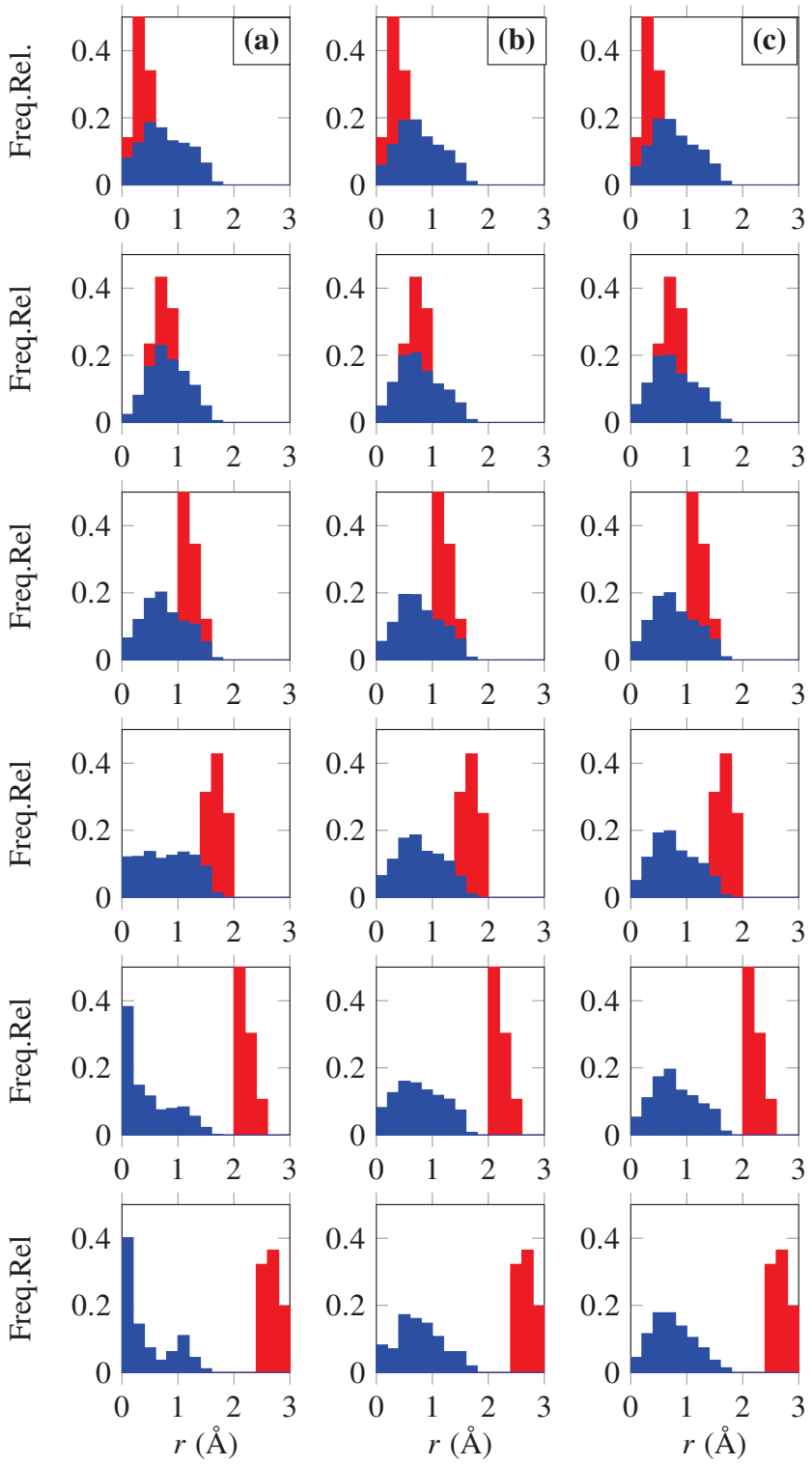


FIGURA C.20: ENTRADA DE ESFERAS CONCÊNTRICAS – FRES13A#31.

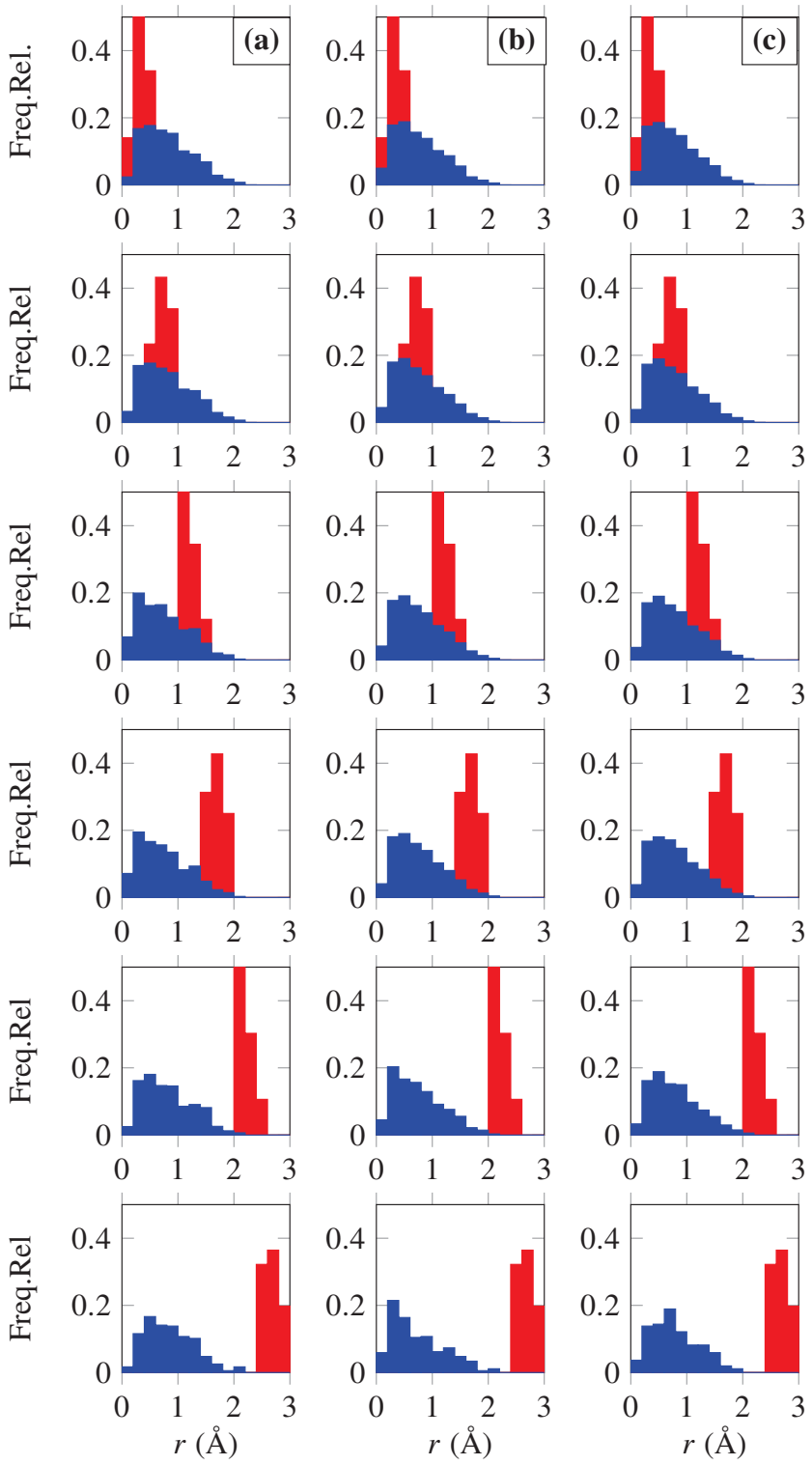


FIGURA C.21: ENTRADA DE ESFERAS CONCÊNTRICAS – FRES14A#29.

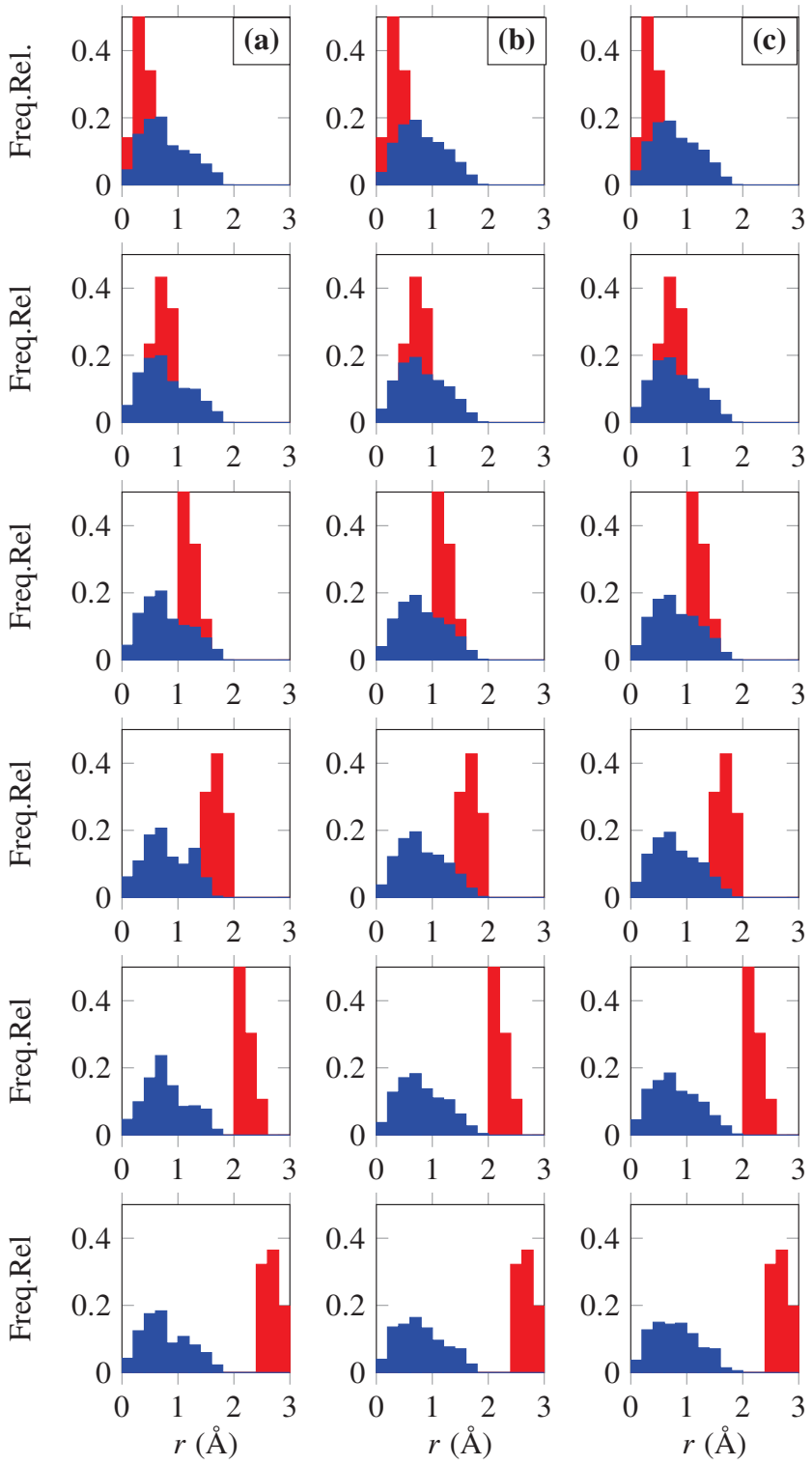


FIGURA C.22: ENTRADA DE ESFERAS CONCÊNTRICAS – FRES14B#4.

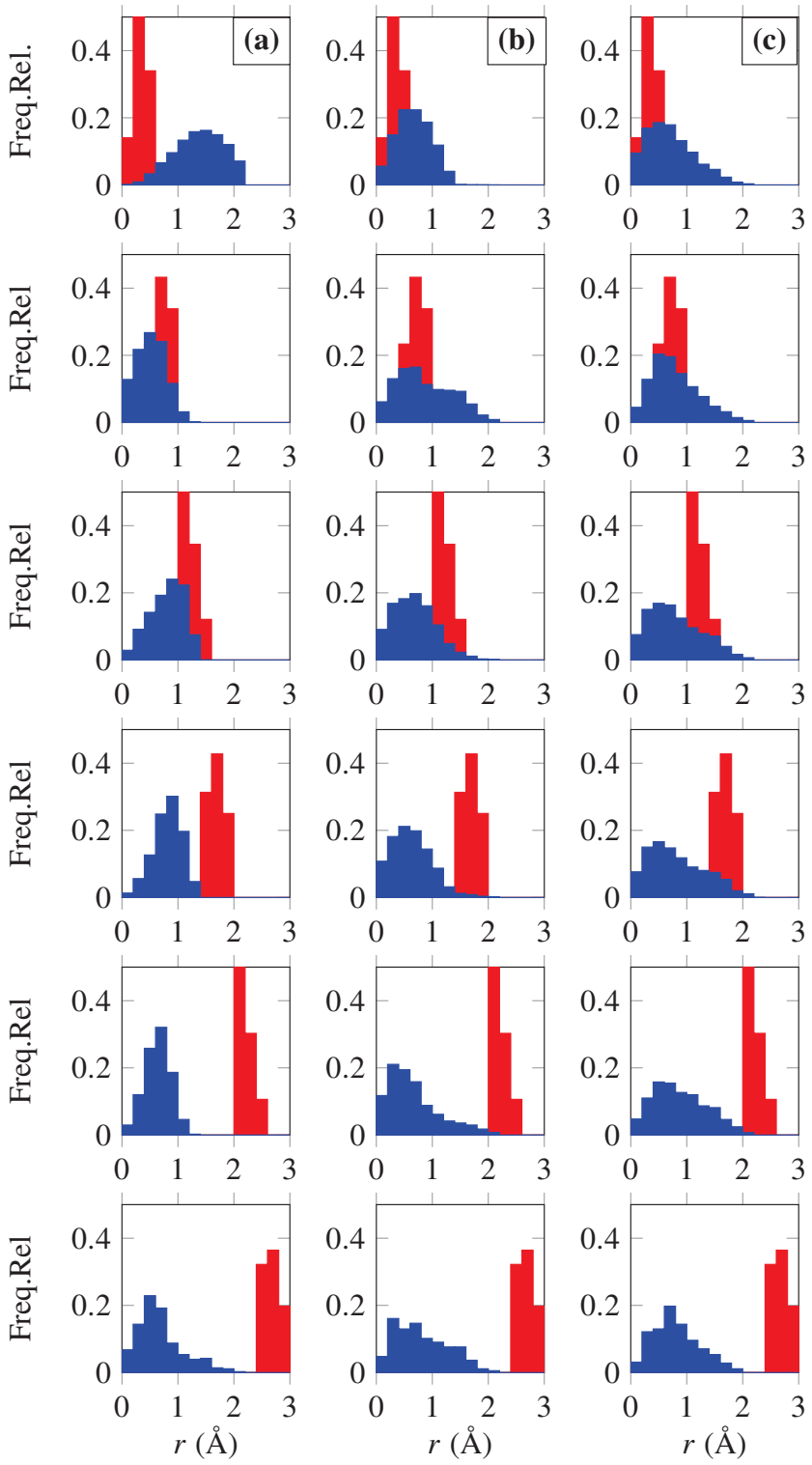


FIGURA C.23: ENTRADA DE ESFERAS CONCÊNTRICAS – FRES14B#13.

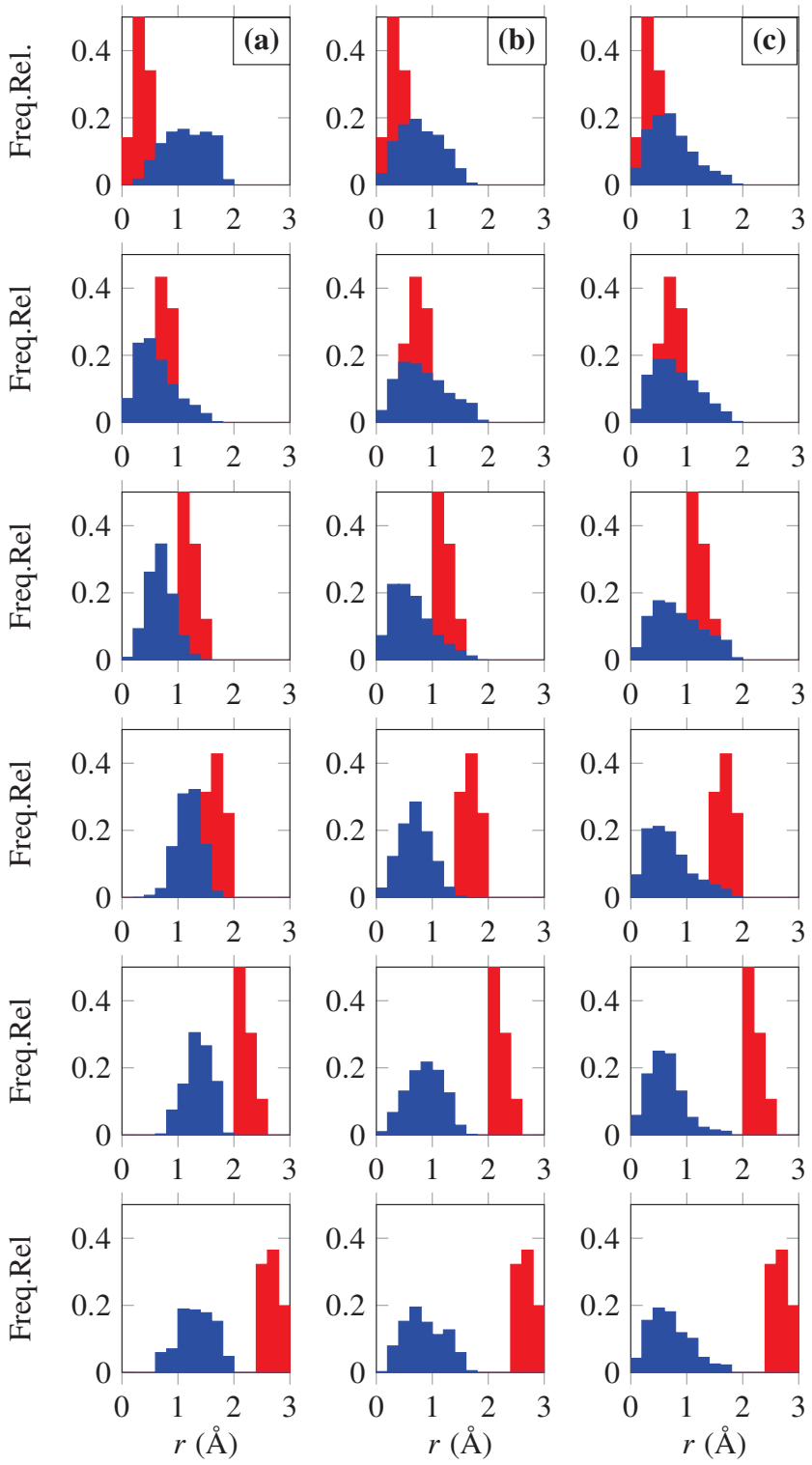


FIGURA C.24: ENTRADA DE ESFERAS CONCÊNTRICAS – FRES14B#34.