UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Raphael Giordano do Nascimento e Silva

DEEP LEARNING PARA SELEÇÃO DE CONFORMAÇÕES DE PROTEÍNAS CONSIDERANDO SUAS PROPRIEDADES TRIDIMENSIONAIS

Santa Maria, RS
2018

Raphael Giordano do Nascimento e Silva

DEEP LEARNING PARA SELEÇÃO DE CONFORMAÇÕES DE PROTEÍNAS CONSIDERANDO SUAS PROPRIEDADES TRIDIMENSIONAIS

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação, Área de Concentração em Computação Aplicada, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Mestre em Ciência da Computação**.

ORIENTADORA: Prof.ª Ana Trindade Winck

Santa Maria, RS
2018

Silva, Raphael Giordano do Nascimento e
DEEP LEARNING PARA SELEÇÃO DE CONFORMAÇÕES DE PROTEÍNAS CONSIDERANDO SUAS PROPRIEDADES TRIDIMENSIONAIS / Raphael Giordano do Nascimento e Silva.- 2018.
65 p.; 30 cm

Orientadora: Ana Trindade Winck
Dissertação (mestrado) - Universidade Federal de Santa Maria, Centro de Tecnologia, Programa de Pós-Graduação em Informática, RS, 2018

1. Aprendizagem Profunda 2. Redes Neurais Convolucionais 3. Docagem Molecular I. Winck, Ana Trindade II. Título.

Sistema de geração automática de ficha catalográfica da UFSM. Dados fornecidos pelo autor(a). Sob supervisão da Direção da Divisão de Processos Técnicos da Biblioteca Central. Bibliotecária responsável Paula Schoenfeldt Patta CRB 10/1728.

Raphael Giordano do Nascimento e Silva

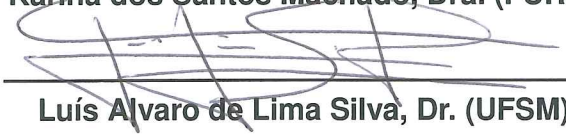
DEEP LEARNING PARA SELEÇÃO DE CONFORMAÇÕES DE PROTEÍNAS CONSIDERANDO SUAS PROPRIEDADES TRIDIMENSIONAIS

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação, Área de Concentração em Computação Aplicada, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Mestre em Ciência da Computação**.

Aprovado em 27 de agosto de 2018:

Ana Trindade Winck, Dra. (UFSM)
(Presidenta/Orientadora)

Karina dos Santos Machado, Dra. (FURG)



Luís Alvaro de Lima Silva, Dr. (UFSM)

Santa Maria, RS
2018

DEDICATÓRIA

Ao meu pai, João Onilso (in memorian), e meus avôs Antônio Laurindo (in memorian) e Benício (in memorian).

Ler fornece ao espírito materiais para o conhecimento, mas só o pensar faz nosso o que lemos.

(John Locke)

RESUMO

DEEP LEARNING PARA SELEÇÃO DE CONFORMAÇÕES DE PROTEÍNAS CONSIDERANDO SUAS PROPRIEDADES TRIDIMENSIONAIS

AUTOR: Raphael Giordano do Nascimento e Silva
ORIENTADORA: Ana Trindade Winck

A descoberta de novos fármacos é custosa, tanto financeiramente como em tempo. Para tratar essa questão, o processo de Desenho Racional de Fármacos investiga a interação entre moléculas receptoras e ligantes candidatos a fármacos. Por meio de experimentos de simulação de docagem molecular é possível avaliar a qualidade da ligação entre essas moléculas. Para simular a flexibilidade do receptor, podem ser executados simulações de dinâmica molecular, onde a proteína é representada por uma conformação distinta em cada instante de tempo. Assim, os experimentos de docagem molecular podem ser executados sobre essas diferentes conformações. Diversas técnicas de aprendizagem de máquina podem ser utilizadas para minerar esses dados. Esta dissertação apresenta uma abordagem para redes neurais *deep learning* que utiliza como dados de entrada diferentes conformações de uma proteína, geradas por meio de simulações de dinâmica molecular. Essas conformações são descritas em termos das coordenadas tridimensionais dos átomos que a compõem, rotuladas por um atributo alvo FEB (*Free Energy of Binding*), que relaciona a interação entre a conformação do receptor e o ligante em experimentos de docagem molecular. Nas abordagens de *deep learning* propostas neste trabalho, são considerados como entrada tanto esses dados brutos, como são considerados uma abordagem que utiliza resultados agrupamento de dados para gerar um bloco com intervalos ideais para cada átomo, em vez dos dados brutos. Essas propostas são implementadas em termos de redes neurais *deep feedforward* e convolucional. As estratégias são comparadas com redes neurais com a mesma arquitetura, sem o uso dos blocos. Para ambas arquiteturas, a estratégia de utilizar os blocos gerados por agrupamento mostrou resultados promissores. Entretanto os modelos gerados a partir da arquitetura *deep feedforward* apresentam os melhores resultados globais, sendo que aqueles com abordagem de agrupamento por DBSCAN destacam-se frente às outras duas abordagens.

Palavras-chave: Aprendizagem Profunda. Redes Neurais Convolucionais. Docagem Molecular.

ABSTRACT

DEEP LEARNING ON PROTEIN CONFORMATION SELECTION CONSIDERING THEIR TRIDIMENSIONAL PROPERTIES

AUTHOR: Raphael Giordano do Nascimento e Silva
ADVISOR: Ana Trindade Winck

A new drug discovery is financially costly and time-consuming. To deal this issue, the Rational Drug Design process investigates the macromolecular interaction between receptor molecules and drug-candidate ligands. Through molecular docking experiments it becomes possible to evaluate the binding quality between these molecules. To simulate the flexibility of the receptor, molecular dynamics simulations can be performed, where the protein is represented by a distinct conformation at each instant of time. Thus, molecular docking experiments can be performed on these different conformations. Several machine learning techniques can be used to mine these data. This work presents an approach for artificial neural networks deep learning, which uses as input several conformation about a given protein, generated through molecular dynamic simulations. These conformation are described in terms of its atoms tridimensional coordinates, labeled by a target attribute FEB, which points out the quality about each conformation in molecular docking experiments. In the deep learning approaches proposed in this work, we consider as input both these raw data, as well as we consider an approach that makes use of clustering experiments that generates a good parallelepiped surface for each atom, instead of the raw data. These strategies are implemented in terms deep feedforward networks and deep convolutional networks algorithms. For both architectures, the clustering-based strategy showed up promising results. However, models generated through deep feedforward showed up the best global results, being that those with the DBSCAN-Clustering-based approach highlight from the other ones.

Keywords: Deep Learning. Convolutional Neural Networks. Molecular Docking.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

$AcrB$	*Acriflavine resistance protein B*
AUC	*Area Under Curve*
BR	*Binary Relevance*
CPI	*Compound-Protein Interactions*
DBN	*Deep Belief Network*
$DCNN$	*Deep Convolutional Neural Network*
DFF	*Deep Feed Foward*
DL	*Deep Learning*
DM	Dinâmica Molecular
DT	*Decision Tree*
EBI	*European Bioinformatics Institute*
$EC2$	*Elastic Compute Cloud*
$EMBL$	*European Molecular Biology Laboratory*
FEB	*Free of Energy Binding*
IA	Inteligência Artificial
LE	*Lowest Energy*
$LSTM$	*Long-Short Term Memory*
MAE	*Mean Absolute Error*
MSE	*Mean Squared Error*
PDB	*Protein Data Bank*
RBM	*Restricted Boltzmann Machines*
RDD	*Rational Drug Design*
$ReLU$	*Rectified Linear Unit*
$RMSE$	*Root Mean Squared Error*
RNA	Redes Neurais Artificiais
SSD	*Solid-State Drive*
SVM	*Support Vector Machine*

SUMÁRIO

1 INTRODUÇÃO

Fármacos são fundamentais na prevenção e tratamento de doenças. Porém, desenvolver um novo fármaco é um grande desafio, mesmo para alvos conhecidos. Desenho Racional de Fármacos (RDD - *Rational Drug Design*) é o processo de descoberta de novos fármacos baseado no conhecimento de um alvo biológico com apoio do computador (KUNTZ, 1992). O princípio biológico fundamental desse processo diz respeito à interação entre uma macromolécula - chamadas de receptoras - e moléculas menores - chamadas de ligantes - em uma técnica chamada de docagem molecular (LYBRAND, 1995). Por meio de experimentos *in silico* de docagem molecular busca-se encontrar ligantes que inibem ou realizem funções específicas na molécula alvo. Uma das maneiras de avaliar a qualidade desse experimento é por meio de uma medida chamada energia livre de ligação (FEB - *Free Energy of Binding*), onde aqueles resultados que despendem mais energia tendem a ser os melhores. No processo de RDD, os ligantes que obtiverem os resultados mais promissores são sintetizados e testados em experimentos *in vitro* e *in vivo* e, em casos de resultados satisfatórios, um novo fármaco pode surgir.

Muitos algoritmos de docagem molecular consideram o receptor como uma estrutura rígida. Porém, a flexibilidade do receptor é um dos fatores que podem apresentar grande influência nos resultados. Uma forma de avaliar a flexibilidade do receptor é por meio de simulações de dinâmica molecular (DM). A DM gera uma conformação da proteína a cada intervalo de tempo. Para que a flexibilidade simulada por meio da DM seja incorporada em experimentos de docagem, cada conformação é submetida a uma simulação de docagem molecular. Assim, torna-se possível avaliar a interação entre um ligante e uma proteína, considerando as diferentes posições que essas proteínas exercem no ambiente celular.

Ao considerar a flexibilidade do receptor nesses experimentos, tem-se uma grande quantidade de dados a serem analisados para aferir sua qualidade. Alguns estudos, como os apresentados em (WINCK, 2012), (MACHADO et al., 2010) e (KRONBAUER; MACHADO; WINCK, 2016) abordam estratégias computacionais baseados em mineração de dados e algoritmos de aprendizado de máquina, para avaliar aquelas conformações que tem chance de serem mais promissoras para experimentos de docagem e, assim, contribuir: i) para o entendimento da flexibilidade da proteína e da interação da mesma com um dado ligante; ii)para a redução de futuros experimentos de docagem molecular, permitindo que tais experimentos sejam realizados apenas sobre aquelas conformações que tendem a produzir melhores resultados.

Mineração de dados é o processo de explorar um grande volume de dados com o objetivo de identificar padrões potencialmente úteis para o contexto em que estão sendo analisados (TAN et al., 2006). Para isso, são empregadas estratégias direcionadas de pré-

processamento de dados e, sobre esses dados pré-processados, são aplicados algoritmos de aprendizagem de máquina, os quais atuam basicamente com técnicas estatísticas para o mapeamento desses padrões. Dentre esses algoritmos, pode-se exemplificar: i) os não supervisionados, indicados para a análise descritiva de dados, tais como agrupamento de dados (*clustering*) e regras de associação; e ii) os supervisionados, indicados para a análise preditiva de dados, tais como árvores de decisão, redes bayesianas, métodos *ensemble* e, em especial, as redes neurais artificiais (ANN - *Artificial Neural Network*) e sua abordagem mais recentemente, as estratégias de *deep learning*.

Uma rede neural artificial é uma técnica de aprendizagem de máquina inspirada em redes neurais biológicas. As redes neurais simulam as conexões entre neurônios, por meio de uma estrutura topológica, cujo processo se adapta para computar uma ação desejada (NORVIG; RUSSELL, 2014). Com o aumento do poder de processamento, baseado nas ANN tradicionais, foram criadas técnicas mais avançadas, chamadas de aprendizagem profunda, ou *deep learning* (GOODFELLOW et al., 2016). Trata-se de um ramo da aprendizagem de máquina cujos algoritmos tentam modelar abstrações de alto nível de dados usando grafos em várias camadas, ou seja, redes neurais multi-camadas. Essas camadas, sejam nas redes neurais convencionais ou nas redes profundas, possuem funções de ativação, que podem ser lineares ou não-lineares. Por ser uma técnica preditiva de aprendizagem de máquina, possui um conjunto de dados rotulados como entrada, e aprende a como mapear a entrada para saída (ALPAYDIN, 2014).

O uso de aprendizagem profunda nos dados oriundos do processo de RDD é algo relativamente recente e com grande potencial, devido a sua capacidade de processar de forma eficiente os dados brutos e suas diferentes abordagens. Este trabalho investiga o uso dessas técnicas em dados obtidos das simulação de dinâmica molecular, rotulados pelos resultados de FEB obtidos dos experimentos de docagem molecular. Para tanto, propõe-se duas estratégias distintas de pré-processamento desses dados, as quais consideram como dados de entrada as propriedades tridimensionais das proteínas. Essas são submetidos a diferentes tipos de redes neurais *deep learning*, buscando encontrar o melhor subconjunto de conformações potencialmente favoráveis a futuros experimentos de docagem molecular.

1.1 MOTIVAÇÃO

Simulações de docagem molecular com receptor flexível são custosas em relação ao tempo e geram uma grande quantidade de dados. Minerar esses dados ajuda a criar modelos que auxiliam na seleção de conformações promissoras de um receptor e um dado ligante para experimentos futuros de docagem molecular, contribuindo para a redução no tempo desses experimentos, bem como para a compreensão de como a flexibilidade do

receptor sendo analisado pode influenciar na interação com algum determinado ligante. O trabalho desenvolvido por Winck (2012) trata dados provenientes de simulações de docagem molecular focando nas posições de cada átomo no espaço euclidiano, definindo cada átomo em relação a um intervalo considerado bom, utilizando árvores de decisão para regressão, para representar esses dados. Este algoritmo é denominado em Winck (2012) por 3D-Tri.

Buscando incorporar aspectos mais eficazes de implementação, o 3D-Tri foi reimplementado em linguagem Python por (SILVA, 2015), e avaliado usando dados de simulação de docagem molecular sobre a proteína AcrB (*Acriflavine resistance protein B*) (MURAKAMI et al., 2002) com o ligante NUNL02, conhecido como um inibidor da bomba de efluxo AcrB. Salienta-se que esses trabalhos valem-se das estruturas tridimensionais dos átomos das proteínas sendo analisadas como conjunto de dados de entrada para o algoritmo de árvore de decisão implementado.

Assim como no algoritmo proposto por Winck (2012) e testado por Silva (2015), esta dissertação busca tratar a natureza das propriedades tridimensionais dos átomos no espaço euclidiano, em vez de considerar simples atributos numéricos, como dados de entrada para um algoritmo de aprendizado de máquina. Entretanto, buscando atingir melhores resultados dos que os obtidos nos trabalhos anteriores, esta dissertação propõe a utilização de técnicas mais robustas de aprendizado de máquina, por meio de redes neurais artificiais de aprendizagem profunda (*deep learning*).

1.2 OBJETIVOS

1.2.1 Objetivo geral

O objetivo deste trabalho é apresentar uma nova abordagem na classificação de dados de conformações de proteínas, oriundos de simulação de dinâmica molecular (DM) e experimentos de docagem molecular. Para tanto, duas abordagens são utilizadas. A primeira, uma rede neural de aprendizagem profunda com alimentação para frente (*feed-forward*), e a segunda, uma rede neural convolucional. Ambas abordagens são capazes de interpretar os dados da posição no espaço euclidiano de cada um dos átomos da molécula utilizada na docagem, em cada instante de tempo.

1.2.2 Objetivos específicos

- Desenvolver uma estratégia para interpretar as características tridimensionais dos dados em redes de aprendizagem profunda;
- Implementar uma rede neural de aprendizagem profunda que utiliza a estratégia desenvolvida;
- Implementar uma rede neural convolucional que utiliza a estratégia desenvolvida;
- Comparar as duas redes neurais implementadas com redes neurais semelhantes, sem o uso da estratégia desenvolvida;
- Comparar as redes neurais implementadas com o algoritmo 3D-Tri.

1.3 ORGANIZAÇÃO DA DISSERTAÇÃO

A presente Dissertação está estruturada da seguinte forma:

- O Capítulo 2 trata dos conceitos de bioinformática, enfatizando o processo de docagem molecular e dinâmica molecular, e apresentando a proteína e o ligante utilizados nesta estratégia, os quais geraram os dados desta dissertação;
- No Capítulo 3 são apresentados os conceitos de aprendizagem de máquina, dando especial ênfase às redes neurais e *deep learning*;
- O Capítulo 4 apresenta os trabalhos relacionados no formato de revisão sistemática de literatura a respeito do uso de técnicas de *deep learning* no processo de docagem molecular;
- No Capítulo 5 é apresentada a metodologia utilizada no presente trabalho, para pré-processamento dos dados, utilização nos algoritmos de testes e na construção das redes neurais de aprendizagem profunda;
- O Capítulo 6 mostra e discute os resultados dos experimentos realizados;
- O Capítulo 7 apresenta as conclusões desta Dissertação, com sugestões para trabalhos futuros;
- Por fim, são apresentadas as referências bibliográficas.

2 BIOINFORMÁTICA E O PLANEJAMENTO RACIONAL DE FÁRMACOS

Bioinformática pode ser definida em termos de moléculas, onde se aplicam algoritmos e técnicas de informática para compreender e organizar informações associadas a essas moléculas em larga escala (LUSCOMBE et al., 2001). Os dados utilizados nas aplicações de bioinformática geralmente são dados brutos oriundos de sequências de DNA, sequência de proteínas, estrutura macromolecular, genomas e expressão genética.

Um dos grandes desafios de se trabalhar com dados de bioinformática consiste na quantidade desses dados, os quais são considerado muito grandes, e também o fato desses dados crescerem a uma taxa muito alta (LESK, 2013). O Instituto Europeu de Bioinformática (EMBL-EBI - *European Molecular Biology Laboratory - European Bioinformatics Institute*), o qual mantém e disponibiliza gratuitamente dados relacionados a ciências biológicas, em seu mais recente relatório (COOK et al., 2018) aponta a importância e desafios que os dados biológicos e o seu constante crescimento tem apresentado. Ainda, mencionam que o volume de armazenamento de dados em seus servidores atingiram 30% da sua capacidade total, chegando a aproximadamente 75 Petabytes de dados biológicos armazenados em 2015.

Essa quantidade de dados carrega consigo uma larga gama de informações que merecem ser analisadas, cujos significados biológico são de especial importância para as suas diferentes aplicações e profissionais envolvidos. Dessa forma, essa grande quantidade de dados biológicos, bem como as oportunidades de geração de dados biológicos, incentiva os cientistas a buscarem objetivos ambiciosos, como entender aspectos interativos da biologia dos organismos, relacionar sequência, estrutura tridimensional, interações e funções de proteínas individualmente, dentre outros objetivos.

2.1 PLANEJAMENTO RACIONAL DE FÁRMACOS

Uma das áreas cobertas pela bioinformática, diz respeito ao planejamento de fármacos. A descoberta de um novo fármaco é uma tarefa complexa. O tempo médio necessário para que um novo fármaco seja comercializado é, em média, de 11 anos, e os custos estimados estão em mais de 1 bilhão de dólares (DIMASI, 2014). Por esses motivos, esforços são aplicados para reduzir o custo e o tempo na descoberta de um novo fármaco, e, adicionalmente, aumentar a qualidade dos compostos candidatos a fármacos.

Desenho racional de fármacos é uma área da bioinformática que tem como princípio fundamental a análise da interação entre macromoléculas chamadas de receptores e moléculas menores chamadas de ligantes (LYBRAND, 1995), onde os receptores normalmente dizem respeito a proteínas alvo associadas a alguma doença, e os ligantes possíveis

inibidores dessa proteína (KUNTZ, 1992).

Na docagem molecular, se investiga e avalia o melhor encaixe entre o ligante e o receptor (KUNTZ, 1992). Essas ligações ocorrem em locais específicos, chamados de sítios ativos ou de ligação, em analogia a uma doca. Aqueles ligantes que melhor se encaixarem podem ser potenciais inibidores da proteína. A qualidade de uma ligação, entretanto, não diz respeito apenas ao encaixe das duas estruturas, mas especialmente à energia despendida nesse processo, sendo que quanto mais negativa, melhor (LYBRAND, 1995). Em (JEFFREY, 1997) é citado que a maior distância entre os átomos do ligante e do receptor, para que haja contato significante, é 4,00 Å(Angstroms). Mais recentemente, os estudos de (SILVEIRA et al., 2009) mostram que a distância para haver contato significante pode ser computada em 7,00 Å. Em outras palavras, quantos mais próximos estão os átomos das duas estruturas, mais energia é despendida em função dessa ligação. Uma das formas de se avaliar o resultado da docagem é por meio do valor da energia livre de ligação (FEB). Experimentos *in silico* de docagem molecular são executados em software específicos, tal como AutoDock Vina (TROTT; OLSON, 2010)

O processo de RDD consiste em um ciclo de quatro etapas principais definidas por Kuntz (1992), as quais são listados a seguir:

1. Primeiramente, isola-se uma estrutura alvo (ou receptor) para que sua estrutura tridimensional seja analisada computacionalmente a fim de identificar os sítios ativos de ligação. As informações a respeito dessas estruturas estão disponíveis em bancos de dados estruturais, como por exemplo o PDB (*Protein Data Bank*) (BERMAN et al., 2000);

2. O segundo passo consiste em selecionar ligantes candidatos a se encaixarem com o receptor selecionado no passo anterior. Para isso, as conformações e posições de um ligante dentro do sítio de ligação da proteína alvo podem ser simuladas usando um software de docagem molecular;

3. Os ligantes selecionados na etapa anterior são então ranqueados de acordo com seus resultados nas simulações. Os melhores são experimentalmente sintetizados e testados;

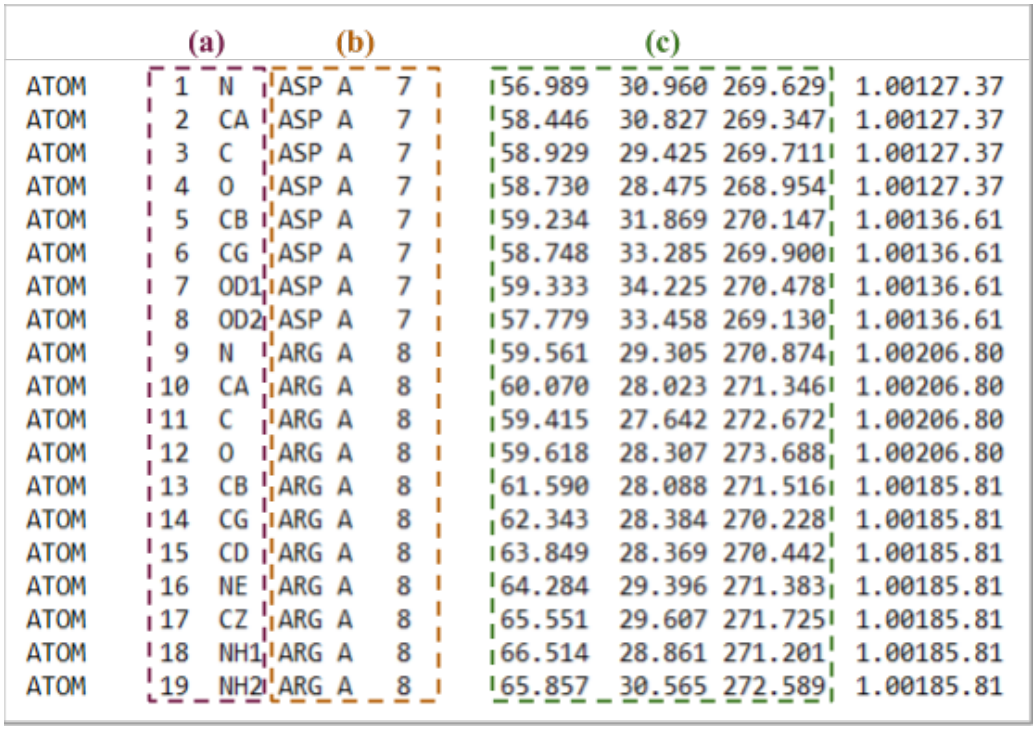
4. Por fim, com base nos resultados dos experimentos, um novo medicamento pode surgir. Caso contrário, volta-se para a primeira etapa.

Muitas abordagens de docagem molecular consideram apenas a flexibilidade do ligante, considerando portanto o receptor como uma estrutura rígida. Entretanto, em uma abordagem mais realista, deve-se considerar o receptor como sendo uma estrutura flexível, tal qual são as proteínas no ambiente celular (COZZINI et al., 2008). Há, na bioinformática, diferentes abordagens para considerar a flexibilidade do receptor. Dentre elas destaca-se

aquelas geradas por meio de uma técnica denominada simulação por dinâmica molecular (LIN et al., 2002; MACHADO et al., 2007; AMARO; BARON; MCCAMMON, 2008).

As informações acerca de uma estrutura 3D da proteína podem ser obtidas de bancos de dados estruturais, como por exemplo PDB (*Protein Data Bank*) (BERMAN et al., 2000). Cada proteína depositada possui um identificador único (PDBID), sendo disponibilizado um arquivo no formato PDB, o é composto por seções que trazem dados a respeito da proteína, tais como método, resolução e artigo que a descrevem, e por elementos técnicos que descrevem a proteína em termos dos átomos e resíduos que a compõem e das coordenadas cartesianas referentes a cada átomo. A Figura 2.1 ilustra o trecho de um arquivo PDB (PDBID: 1IWG), o qual contém as informações a respeito dos átomos e resíduos de uma proteína e suas respectivas coordenadas cartesianas.

Figura 2.1 – Exemplo de um trecho de um arquivo PDB (PDBID 1IWG), destacando-se a descrição dos átomos da proteína (a), dos resíduos da proteína (b) e das coordenadas cartesianas de cada átomo listado (c).



As simulações por dinâmica molecular buscam simular a flexibilidade da proteína em seu ambiente celular. Simulações por DM são executadas em um software específico, tais como Amber (PEARLMAN et al., 1995) e GROMACS (PRONK et al., 2013). Esses software ocupam-se de simular como as proteínas se comportam em um intervalo de tempo pré-definido, gerando uma conformação, aqui chamado de *snapshot*, da proteína nesse intervalo de tempo. Para cada *snapshot* pode ser gerado um arquivo no formato PDB.

Sendo assim, experimentos de docagem molecular que consideram a flexibilidade do receptor, ocupam-se de realizar um experimento de docagem sobre cada *snapshot* da

proteína obtida por meio de DM. Isso é, é executado um experimento de docagem sobre cada aquivo PDB oriundo da simulação por DM. Dependendo da trajetória sendo considerada na DM e do tempo de execução para cada experimento de docagem, o tempo para a execução dos experimentos sobre toda a trajetória da proteína pode ser muito elevado.

Esta dissertação faz uso de dados de simulação DM sobre a proteína AcrB (*Acriflavine resistance protein B*), e de experimentos de docagem molecular sobre as conformações desta proteína com o ligante NUNL02, conhecido como um inibidor da bomba de efluxo AcrB.

2.2 PROTEÍNA ACRB

Bombas de efluxo estão atualmente entre as proteínas mais importantes a serem investigadas no contexto de RDD. Essas proteínas estão presentes na célula bacteriana e são responsáveis pelo transporte ativo de compostos capturados na entrada da membrana para o exterior das células.

A proteína AcrB *Acriflavine resistance protein B* , ilustrada na Figura 2.2, é uma proteína relacionada à bomba de efluxo. Trata-se de uma proteína importante no que se refere aos estudos de resistência antimicrobianas (BRUNTON; KNOLLMANN, 2010). Este é um importante alvo de estudo em bioinformática, uma vez que as infecções acarretadas por bactérias resistentes necessitam de maiores cuidados, assim como necessitam de antimicrobianos de alto custo.

Segundo Prates (2014), a molécula dessa proteína é considerada como a principal transportadora de múltiplas drogas em *Escherichia Coli* (MURAKAMI, 2006) que trabalha em conjunto com o canal *TolC* situado na membrana externa, e a proteína *AcrA* localizada na membrana de fusão da célula (MURAKAMI, 2006).

Esta dissertação trabalha integralmente com os dados de dinâmica molecular gerados e descritos por (PRATES, 2014), o qual executou uma simulação por DM da proteína AcrB com objetivo de entender sua flexibilidade e regiões de maior mudança conformacional. A simulação de DM foi executada em 50 ns (nanossegundos, onde 1ns = 1.000ps e 1ps = 10^{-12}s), utilizando a ferramenta GROMACS 4.0.3 (PRONK et al., 2013), gerando um total de 1.001 conformações.

A partir dessa simulação, (PRATES, 2014) relata que foram executados experimentos de docagem molecular sobre o ligante *NUNL02*, usando o software AutoDockVina (TROTT; OLSON, 2010).

Figura 2.2 – Estrutura 3D da Proteína *Acriflavine resistance protein B* (AcrB).

Fonte: (PRATES, 2014)

2.2.1 Considerações do Capítulo

Este capítulo esclareceu conceitos relacionados à bioinformática e ao processo de RDD, destacando-se os métodos e ferramentas necessários para geração de dados biológicos neste contexto, em especial sobre as simulações por dinâmica molecular e os experimentos de docagem molecular.

Destacou-se o trabalho de (PRATES, 2014), o qual estudou o comportamento da flexibilidade da proteína AcrB, por meio de experimentos de agrupamento de dados sobre os dados gerados pela DM e pelos experimentos de docagem molecular.

Esta dissertação faz uso desses mesmos dados, com o intuito de minerá-los e selecionar conformações promissoras para futuros experimentos de docagem molecular. Assim, os dados utilizados nesta dissertação correspondem aos arquivos PDB referentes

às 1.001 conformações da proteína AcrB, e seus respectivos arquivos resultantes dos experimentos de docagem molecular. Assim, é possível destacar a importância de se minerar dados biológicos e de RDD, no caso deste trabalho. Dentre os fatores destacáveis desses dados está o seu grande volume e o custo do tempo necessário para o processo de docagem e dinâmica molecular. Para sanar ou reduzir esses problemas existe a mineração de dados que utiliza algoritmos de aprendizagem de máquina.

3 APRENDIZAGEM DE MÁQUINA

A Inteligência Artificial (IA) é um dos campos mais recentes em ciências e engenharia, responsável por tentar compreender e construir entidades inteligentes (NORVIG; RUSSELL, 2014). Para isso, a IA baseia-se em princípios teóricos e aplicados sólidos. Dentre esses princípios incluem estruturas de dados para representação do conhecimento (LUGER, 2013).

As definições de IA podem estar relacionadas a processos de pensamento e raciocínio ou ao comportamento. Além disso, o pensamento ou comportamento de um sistema de IA pode ser comparado ao pensamento/comportamento humano ou racional. É considerado um pensamento ou comportamento humano quando se permite falhas, e considerado um pensamento ou comportamento racional quando o sistema tem a ação correta dado o que ele sabe (NORVIG; RUSSELL, 2014).

Aprendizagem de máquina é parte da inteligência artificial responsável por modelar o processo de aprendizagem em suas múltiplas manifestações nos sistemas. O sistema pode aprender e se adaptar a essas mudanças (ALPAYDIN, 2014; MICHALSKI; CARBONELL; MITCHELL, 2013). Mitchell (1997) define aprendizado de máquina como "a capacidade de melhorar o desempenho na realização de alguma tarefa por meio da experiência". Assim, emprega-se o princípio da inferência denominado indução, obtendo-se conclusões genéricas a partir de um conjunto de exemplos (FACELI et al., 2011).

A aprendizagem de máquina é essencialmente uma forma aplicada das teorias de estatística em modelos matemáticos, fazendo inferências a partir de exemplos, estimando funções complexas e diminuindo a ênfase em prover intervalos de confiança em torno dessas funções (ALPAYDIN, 2014; GOODFELLOW et al., 2016).

Pode-se dividir a aprendizagem de máquina em supervisionadas, ou preditivas, e não supervisionadas, ou descritivas. Tem-se modelos definidos por alguns parâmetros, onde o aprendizado ocupa-se da execução de algoritmos que otimizem os parâmetros do modelo, utilizando dados de treinamento. No caso de aprendizado descritivo, o conhecimento é obtido a partir dos dados. Já no caso do aprendizado preditivo, fornece predições futuras usando basicamente um conjunto de dados de entrada x e uma saída y, onde a tarefa é aprender como mapear o conjunto de entrada para a saída. Esse mapeamento pode ser definido como uma função $y = g(x|\theta)$ onde $g(.)$ é o modelo e θ são os seus parâmetros (ALPAYDIN, 2014).

Na seção seguinte serão abordados os conceitos de aprendizagem supervisionada juntamente com os conceitos de árvores de decisão. A seção posterior abordará aprendizagem não-supervisionada e agrupamento de dados. Na sequência serão abordados os conceitos de redes neurais artificiais, seguido da discussão sobre aprendizagem profunda. Por fim, as considerações do capítulo.

3.1 APRENDIZAGEM SUPERVISIONADA

Algoritmos de aprendizagem supervisionada são algoritmos que aprendem a partir de uma associação de uma entrada e uma saída, dado um conjunto de exemplos de entrada x, representando os atributos ou características, e saída y, representando o atributo alvo ou classe (GOODFELLOW et al., 2016). Esse último se trata do atributo que se deseja classificar, ou seja, predizer após treinado.

Essa associação entre a entrada x e a saída y é dada a partir de uma função, também conhecida informalmente como modelo de classificação (TAN et al., 2006). Essa função é usada para predizer um valor desconhecido de x a partir de uma entrada x.

A dimensão de um conjunto de dados é dada pela quantidade de atributos (x) que ele possui. Por exemplo, uma entrada que possui os atributos *altura*, *largura* e *peso* é tridimensional (3D). Consequentemente para ser representado utiliza um gráfico de três dimensões, onde cada coordenada representa um atributo.

O atributo alvo pode ser discreto ou contínuo. Atributos discretos podem ser avaliados como *booleanos*, onde se verifica se o elemento pertence ou não a classe. Atributos contínuos se diferenciam dos atributos discretos por serem números contínuos, ou seja, sem que haja uma divisão em intervalos por exemplo. A avaliação do desempenho de um modelo de classificação é baseada na comparação entre a contagem de testes previstos corretamente e incorretamente pelo modelo. Essas contagens são representadas em uma tabela conhecida como matriz de confusão (TAN et al., 2006). No exemplo da Tabela 3.1, o número total de predições corretas feitas pelo modelo é (f_{11} + f_{00}) e o número total de predições incorretas é (f_{10} + f_{01}).

Tabela 3.1 – Matriz de confusão para um problema de 2 classes. Cada linha representa o valor real da classe, enquanto cada coluna representa o valor predito para a classe.

		Classe Predita	
		Classe = 1	Classe = 0
Classe Real	Classe = 1	f_{11}	f_{10}
	Classe = 0	f_{01}	f_{00}

Fonte: Adaptado de Tan et al. (2006).

Uma métrica de desempenho, como a acurácia usada para atributo alvo discreto, é usada para comparar o desempenho de diversos modelos. A Equação 3.1 define como a acurácia é calculada de acordo com os valores da Tabela 3.1. Basicamente, a acurácia é o número de predições corretas dividido pelo total de predições. Quanto maior a acurácia, melhor o resultado.

$$\text{Acurácia} = \frac{\text{Número de predições corretas}}{\text{Número total de predições}} = \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}} \tag{3.1}$$

O desempenho de um modelo com atributo alvo discreto também pode ser expressado em termos da sua taxa de erro. A Equação 3.2 a seguir define o cálculo da taxa de erro de acordo com os valores da Tabela 3.1.

$$\text{Taxa de erro} = \frac{\text{Número de predições erradas}}{\text{Número total de predições}} = \frac{f_{10} + f_{01}}{f_{11} + f_{10} + f_{01} + f_{00}} \tag{3.2}$$

Dessa forma, a maioria dos algoritmos de classificação procura modelos que atinjam a maior precisão ou a menor taxa de erro aplicados ao conjunto de teste (TAN et al., 2006). Para a taxa de erro, quanto menor, melhor.

Para os casos onde o atributo alvo é contínuo, em vez de o modelo classificar a saída em classes, $C(x) \in 0, 1$, o valor é uma função numérica (ALPAYDIN, 2014). Esses casos são chamados de Regressão.

Dentre as principais medidas de performance de predições numéricas (regressão) estão o Erro Médio Absoluto (*MAE - Mean Absolute Error*), o Erro Médio Quadrático (*MSE - Mean-Squared Error*), e a Raiz do Erro Médio Quadrático (*RMSE - Root Mean-Squared Error*) (WITTEN et al., 2016).

As Equações 3.3, 3.4 e 3.5 definem como calcular o *MAE*, *MSE* e *RMSE* respectivamente, onde p representa os valores da predição, a os valores reais e n o tamanho do conjunto de dados (número de elementos).

$$MAE = \frac{|p_1 - a_1| + \cdots + |p_n - a_n|}{n} \tag{3.3}$$

$$MSE = \frac{(p_1 - a_1)^2 + \cdots + (p_n - a_n)^2}{n} \tag{3.4}$$

$$RMSE = \sqrt{\frac{(p_1 - a_1)^2 + \cdots + (p_n - a_n)^2}{n}} \tag{3.5}$$

Dentre as várias técnicas disponíveis de aprendizagem supervisionada podemos citar:

- Árvores de decisão;
- Classificador baseado em regras;
- Classificadores de vizinho mais próximo;
- Classificadores bayesianos;
- Rede neural artificial (RNA);
- Regressão Linear;

- *Support Vector Machine* (SVM).

Redes Neurais Artificiais serão abordadas mais detalhadamente em uma seção a parte, seguido da também RNA, aprendizagem profunda. As Árvores de Decisão serão introduzidas na subseção a seguir.

3.1.1 Árvores de Decisão

Árvore de decisão é uma estrutura de dados hierárquica que implementa a estratégia de dividir para conquistar, a qual pode ser usada tanto para classificação quanto para regressão (ALPAYDIN, 2014). Uma das maiores vantagens de se utilizar Árvores de Decisão está na facilidade de leitura, ou visualização, por humanos, já que é fácil de visualizar todos os critérios tomados em cada decisão de divisão dos nodos (MITCHELL, 1997).

Uma Árvore de Decisão é composta por um nodo raiz, nodos internos e nodos folhas (terminais). O nodo raiz está localizado no topo da árvore e, portanto, não possui arestas chegando, apenas saindo. Os nodos internos, que possuem exatamente uma aresta chegando e duas ou mais saindo. Terminais, ou nodos folha, possuem exatamente uma aresta chegando e nenhuma saindo. Em Árvores de Decisão, cada nodo folha recebe um rótulo de classe (TAN et al., 2006).

Durante a indução de árvores de decisão, um atributo é selecionado como critério de divisão de acordo com o critério implementado pelo algoritmo escolhido. Isso gera uma função de teste $f_m(X)$ que testa o atributo e gera as arestas de acordo com os valores do atributo para cada instância. Dessa forma, o conjunto é dividido em n subconjuntos, sendo n o número de nodos gerados pela divisão do nodo da iteração. O processo é repetido para cada nodo até que a condição de parada, também estabelecida pelo algoritmo em questão, seja atingido e o nodo seja definido como folha, atribuindo um rótulo de classe.

A Equação 3.6 define a divisão de uma árvore de decisão para o nodo m, onde X_m é o subconjunto de X que chega ao nodo m; ou seja, X_m é o conjunto de todo o $x \in X$ que satisfaz a condição da aresta que chega ao nodo m (ALPAYDIN, 2014).

$$b_m = \begin{cases} 1 & \text{Se } x \in X_m : x \text{ chega ao nodo } m \\ 0 & \text{Caso contrário} \end{cases} \tag{3.6}$$

Caso a árvore de decisão tenha como atributo alvo, classe, um valor contínuo, essa árvore é também chamada de árvore de regressão. Nesses casos, a medida de impureza utilizada é substituída por uma medida apropriada para regressão. Um critério utilizado para a divisão dos nodos é a Redução de Desvio Padrão (RDP), usado no algoritmo M5P (QUINLAN et al., 1992; WANG; WITTEN, 1997). O principal objetivo deste algoritmo é

maximizar a RDP. A Equação 3.7 expressa o cálculo do RDP, onde $dp(X)$ é o desvio padrão do conjunto de dados X. O i iterado no somatório da equação representa cada subconjunto resultante da divisão do nodo. Em resumo, o RDP é dado pela diferença entre desvio padrão do conjunto, anterior a divisão, e o desvio padrão dos subconjuntos resultantes da divisão, buscando sempre reduzir o valor do desvio padrão.

$$RDP = dp(X) - \sum_{i} \frac{|X_i|}{|X|} \times dp(X_i) \tag{3.7}$$

3.2 APRENDIZAGEM NÃO-SUPERVISIONADA

Na aprendizagem não-supervisionada não se tem um atributo classe ou um supervisor que determine um rótulo, apenas os dados de entrada. Em estatística, chamamos de estimativa de densidade dado uma estrutura no espaço de entrada que contém certos padrões que ocorre com mais frequência do que outros e desejamos ver o que acontece e o que não (ALPAYDIN, 2014). Um dos métodos de estimativa de densidade é o agrupamento de dados, ou *clustering*, que busca por grupos a partir dos dados de entrada.

A subseção a seguir aborda o Agrupamento de Dados e dois algorítimos importantes para o presente trabalho: *K-means* e *DBSCAN*.

3.2.1 Agrupamento de Dados

Agrupamento de dados, também chamado de análise de grupos, é uma técnica de aprendizagem não-supervisionada que divide as instâncias em grupos (*clusters*) que tenham algum significado ou sejam úteis. Deve-se maximizar a similaridade entre elementos do mesmo grupo e minimizar a similaridade entre objetos de grupos distintos (HALKIDI; BATISTAKIS; VAZIRGIANNIS, 2001).

K-means e *DBSCAN* estão entre os principais algoritmos de agrupamento de dados, e, por terem relação com este trabalho, serão abordados a seguir.

3.2.1.1 K-means

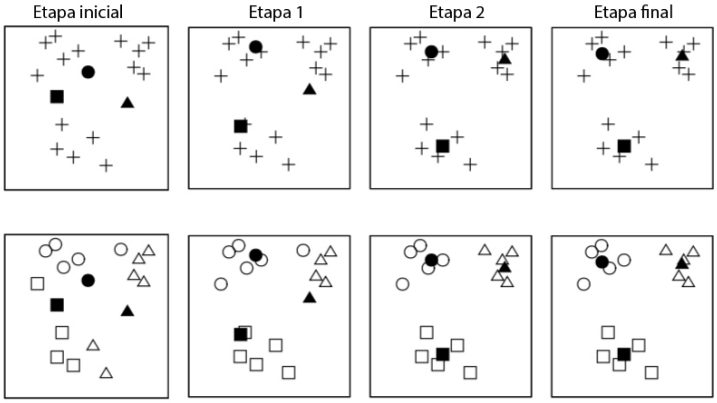
Em técnicas de agrupamento de dados baseado em protótipos, cada objeto está mais próximo (é mais semelhante) do protótipo que define o grupo do que do protótipo de qualquer outro grupo. *K-means* é um algoritmo que utiliza essa técnica, onde o protótipo é

definido em termos de um centroide, geralmente representando a média de um grupo de pontos e aplicada a objetos em um espaço n-dimensional contínuo (TAN et al., 2006).

Primeiramente, *k* pontos, onde *k* é definido como parâmetro, são escolhidos randomicamente como centro dos grupos. Todas as instâncias são atribuídas ao grupo mais próximo de acordo com a métrica de distância escolhida. Então, o centro de cada grupo é recalculado de acordo com a média das instâncias do mesmo. Por fim, o processo é repetido até que os grupos estejam estáveis e permaneçam iguais (WITTEN et al., 2016).

A Figura 3.1 mostra passo-a-passo a execução do algoritmo *K-means*. Na Etapa inicial são atribuídos aleatoriamente três grupos representados por um quadrado, circulo e triângulo. As instâncias são então atribuídas a esses três grupos. Na Etapa 1, os centros dos três grupos são recalculados, de acordo com a média de seus elementos. Os elementos são novamente atribuídos ao grupo de centro mais próximo. Na Etapa 2 o processo é repetido. Na Etapa final não houve mudança no centro, portanto, o algoritmo para e estão definidos os grupos.

Figura 3.1 – *K-means* passo-a-passo



Fonte: Adaptado de Witten et al. (2016).

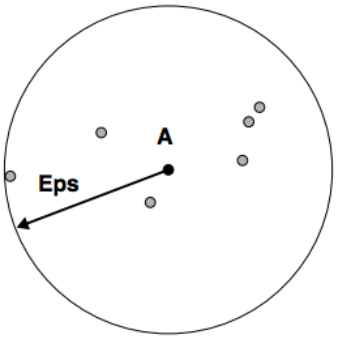
3.2.1.2 DBSCAN

DBSCAN é um algoritmo de agrupamento baseado em densidade simples. Agrupamentos baseados em densidade localizam regiões de alta concentração de pontos (regiões densas) separadas por regiões de baixa concentração de pontos (não densas) (TAN et al., 2006). O DBSCAN utiliza a abordagem baseada em centro para definir a densidade. A densidade de um ponto é dada pelo número de pontos dentro de um determinado raio, definido como Eps. A Figura 3.2 ilustra essa técnica.

A Figura 3.3 ilustra graficamente os conceitos de pontos centrais, pontos no limite e pontos de ruído. A seguir uma descrição sobre cada conceito segundo Tan et al. (2006):

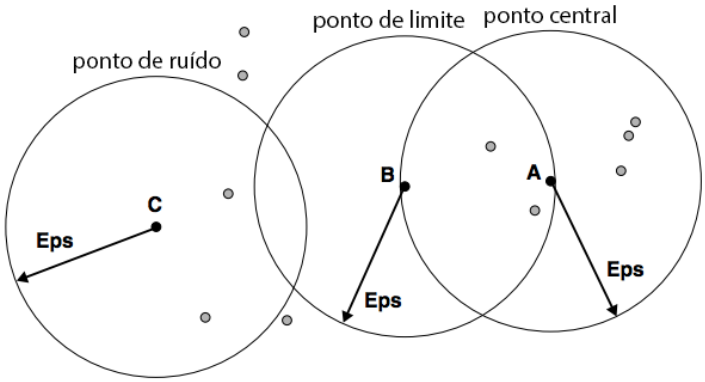
- Pontos Centrais: estão no interior de um grupo baseado em densidade. É considerado central se o número de pontos dentro de uma determinada vizinhança em torno do ponto exceder um determinado limite, MinPts, especificado como parâmetro pelo usuário, Na Figura 3.3, A é um ponto central para o raio indicado (Eps) se MinPts $<=$ 7.
- Pontos de Limite: não é um ponto central, mas está localizado dentro da vizinhança de um ponto central podendo estar dentro de vizinhança de mais de um ponto central. Na Figura 3.3, B é um ponto limite.
- Pontos de Ruído: qualquer ponto que não seja nem um ponto central nem um de limite. Na Figura 3.3, o ponto C é um ponto de ruído.

Figura 3.2 – Densidade baseada em centro



Fonte: Adaptado de Tan et al. (2006).

Figura 3.3 – Densidade baseada em centro



Fonte: Adaptado de Tan et al. (2006).

O algoritmo DBSCAN é definido por Tan et al. (2006) da seguinte forma:

1. Rotular todos os pontos como de centro, de limite ou de ruído.

2. Eliminar os pontos de ruído.
3. Colocar uma aresta entre todos os pontos de centro que estejam dentro da Eps uns dos outros.
4. Tornar cada grupo de pontos de centro conectados a um grupo separado.
5. Atribuir cada ponto de limite a um dos grupos dos seus pontos de centro associados.

3.2.1.3 Considerações sobre Agrupamento de Dados

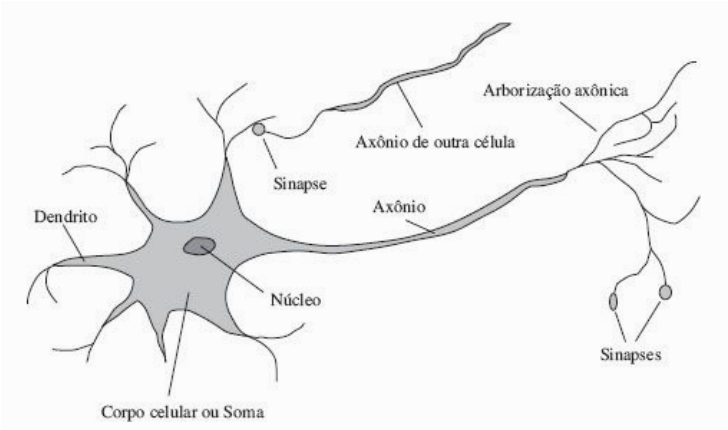
Ambos, K-means e DBSCAN, são algoritmos de agrupamento particional que atribuem cada objeto a um único grupo. A principal diferença está em que enquanto o K-means agrupa todos os objetos, o DBSCAN descarta objetos classificados como ruído. Além disso, o K-means pode encontrar grupos que não estejam bem separados, mesmo se houver intersecção entre eles, mas DBSCAN funde grupos que possuam intersecção. O DBSCAN sempre produz o mesmo conjunto de grupos de uma execução para outra, enquanto o K-means geralmente gera os centroides iniciais de forma aleatória. Por fim, o DBSCAN determina automaticamente o número de grupos, enquanto o K-means é definido por parâmetro (TAN et al., 2006).

3.3 REDES NEURAIS ARTIFICIAIS

Neurociência é a área do conhecimento responsável por estudar o sistema nervoso, em particular o cérebro. A partir do século XVIII o cérebro foi amplamente reconhecido como a sede da consciência. Em 1874, o médico e histologista Camillo Golgi (1843-1926) desenvolveu uma técnica de coloração que permitiu a observação de neurônios individuais no cérebro. Essa técnica foi usada para o estudo das estruturas de neurônios do cérebro primeiramente por Ramon y Cajal (1852-1934). Nicolas Rashevsky (1936, 1938) por sua vez, foi o primeiro a aplicar modelos matemáticos no estudo do sistema nervoso (NORVIG; RUSSELL, 2014). As partes de um neurônio podem ser observadas na Figura 3.4.

Inspirados na neurociência, alguns trabalhos na IA tiveram como objetivo criar redes neurais artificiais (RNA). McCulloch e Pitts (1943), em um trabalho intitulado "*A logical calculus of the ideas immanent in nervous activity*" apresentaram um modelo matemático simples do neurônio. Esse modelo de neurônio propaga uma saída quando uma combinação linear de suas entradas excede algum limiar. Uma rede neural artificial é portanto um conjunto dessas unidades conectadas, variando a sua topologia e as propriedades dos neurônios (NORVIG; RUSSELL, 2014).

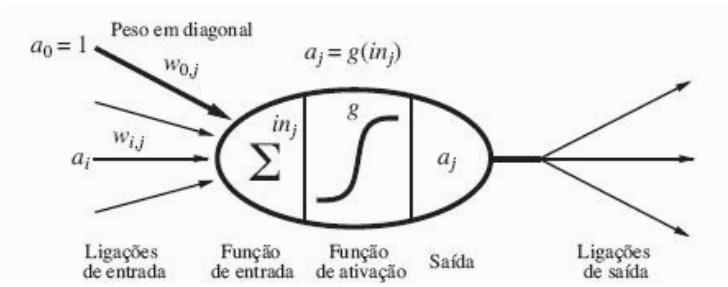
Figura 3.4 – Neurônio. Sua estrutura consiste em um corpo celular ou soma, contendo um núcleo, de onde se estende uma série de fibras chamadas dendritos e uma longa fibra chamada axônio. A conexão com outros neurônios ocorre em junções chamadas de sinapses.



Fonte: Adaptado de Norvig e Russell (2014).

A Figura 3.5 ilustra um modelo matemático simples de um neurônio j, que a partir da entrada vinda do neurônio i utiliza um peso $w_{i,j}$ para cada ligação de entrada $(0...n)$. O somatório de cada ligação de entrada multiplicado por seu respectivo peso é chamado de função de ativação, e seu resultado é propagado para as ligações de saída. RNAs são essencialmente sistemas distribuídos inspirados na estrutura e funcionamento do sistema nervoso, compostas por várias unidades de processamento (neurônios) e interligadas por uma grande quantidade de conexões (sinapses) (FACELI et al., 2011).

Figura 3.5 – Modelo matemático simples de um neurônio artificial. A ativação de saída é dada por $a_j = g(\sum_{i=0}^{n} w_{i,j}\, a_i)$, onde a_i é a ativação de saída da unidade i e $w_{i,j}$ é o peso sobre a ligação da unidade i com essa unidade.



Fonte: Adaptado de Norvig e Russell (2014).

Conforme Luger (LUGER, 2013), a base de uma rede neural artificial é o neurônio artificial o qual consiste em:

- sinais de entrada x_i, cujos dados podem ser obtidos do ambiente ou da ativação de outros neurônios;

- conjunto de pesos w_i, que descrevem as forças das conexões;
- nível de ativação $\sigma w_i x_i$, determinado pelas forças dos sinais de entrada;
- função de limiar f, que calcula o estado final ou saída do neurônio.

A Equação 3.8 representa a saída y de um neurônio, onde d é a quantidade de atributos de entrada, w_j o peso sináptico do atributo x_j, e w_0 o valor de viés (*bias*).

$$y = \sum_{j=1}^{d} w_j x_j + w_0 \tag{3.8}$$

Uma função de ativação é tipicamente um limiar rígido, unidade chamada de *perceptron*, ou uma função logística, chamada de *perceptron sigmoide* (NORVIG; RUSSELL, 2014). Outras funções de ativação serão abordadas a seguir.

Há duas maneiras de se conectar os neurônios artificiais. A primeira, rede com alimentação para frente (*feedforward*), que tem conexões somente em uma direção, formando um grafo acíclico dirigido. Dessa forma, um nó j recebe a entrada do nó i e propaga sua saída para o nó k, sem que haja laços. A segunda, rede recorrente, que alimenta suas saídas de volta às suas próprias entradas. Nesse caso, os níveis de ativação da rede formam um sistema dinâmico que pode atingir um estado estável, apresentar oscilações, ou um comportamento caótico (NORVIG; RUSSELL, 2014).

Uma RNA é formada por neurônios interligados, formando um grafo onde a primeira camada normalmente é a camada de entrada de dados, a última é a saída de dados, e as internas são ocultas (*hidden layers*). Uma rede com todas as entradas conectadas diretamente com as saídas é chamada de camada única ou perceptron.

Alpaydin (2014) define duas formas de calcular os pesos de uma rede perceptron: *offline* e *online*. Normalmente, o método *online* é utilizado. Nesse método, a entrada é feita instância por instância, uma a uma, e os parâmetros da rede são atualizados a cada instância, adaptando-se gradativamente. Os parâmetros são inicializados aleatoriamente e atualizados a cada iteração em pequenas variações de forma a reduzir o erro, sem esquecer o que foi aprendido anteriormente. Se a função for diferenciável pode-se usar gradiente descendente (ALPAYDIN, 2014).

A Equação 3.9 mostra a atualização do parâmetro w_i. O valor de Δw_i é dado pela Equação 3.10, onde t é a saída do exemplo atual, o a saída gerada pela rede perceptron, e η é uma constante positiva chamada de taxa de aprendizado. A constante de taxa de aprendizado é normalmente um valor muito baixo, permitindo que a aprendizagem seja gradual e evitando mudanças bruscas nos parâmetros (MITCHELL, 1997).

$$w_i \leftarrow w_i + \Delta w_i \tag{3.9}$$

$$\Delta w_i = \eta(t - o)x_i \tag{3.10}$$

Em resumo, os componentes básicos de uma RNA são:

- Neurônio: unidade computacional básica que compõe a rede;
- Arquitetura: estrutura que define a topologia de conexão dos neurônios;
- Aprendizagem: algoritmo que adapta a rede para processar uma função desejada.

3.3.1 Retropropagação

Uma RNA *Perceptron* tem apenas uma camada com pesos e utiliza apenas funções lineares a partir da entrada, não resolvendo problemas onde o discriminante a ser estimado é não-linear. Isso pode ser resolvido adicionando uma camada oculta (*hidden layer*) intermediária entre a camada de entrada e a de saída. Em caso de classificação, como *multilayer perceptrons* (MLP) pode implementar discriminantes não-lineares, e no caso de regressão, pode-se aproximar funções não-lineares a partir da entrada (ALPAYDIN, 2014).

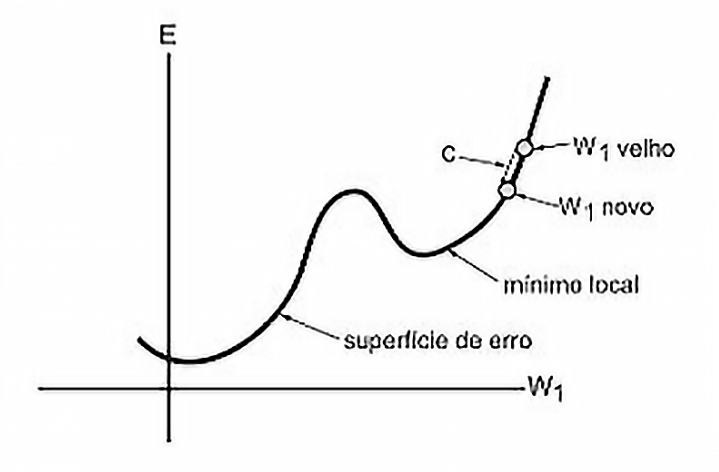
Pode-se generalizar o *perceptron* substituindo sua função de ativação com limiar abrupto por outros tipos. Funções de ativação contínuas, por exemplo, permitem a criação de algoritmos mais sofisticados de aprendizagem, além de permitir um ajuste mais fino na medida de erro (LUGER, 2013).

A regra delta é baseada na ideia de uma superfície de erro que representa o erro cumulativo sobre um conjunto de dados como uma função de pesos da rede, como demonstrado na Figura 3.6. Cada ponto sobre a superfície de erro n-dimensional representa uma configuração de pesos da rede (RUMELHART; HINTON; WILLIAMS, 1986). O algoritmo deseja então encontrar a direção sobre essa superfície que reduza o erro. Essa abordagem é chamada de aprendizado por gradiente descendente. Gradiente é a medida de inclinação a partir de um ponto da superfície (LUGER, 2013).

Com a regra delta ainda não é possível superar as limitações das redes de camada única. Porém, é a base para o algoritmo de retropropagação, também chamado de *back-propagation*. Nesse processo, é calculado uma função gradiente para calcular o erro das saídas (de acordo com os pesos sinápticos já calculados). Essa função é aplicada repetidas vezes para todos os módulos, a partir da saída na parte superior, onde a rede produz sua predição, até a parte inferior, onde a entrada externa é alimentada (LECUN; BENGIO; HINTON, 2015). A Figura 3.7 mostra como a retropropagação é realizada.

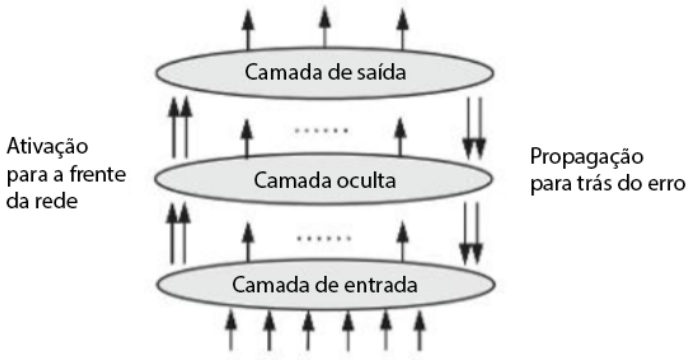
As redes *feedforward* ou MLP são consideradas são consideradas exemplos de redes *deep learning* (GOODFELLOW et al., 2016). A seguir serão abordados conceitos de aprendizagem profunda.

Figura 3.6 – Superfície de erro em duas dimensões. A constante c controla a taxa de aprendizado, ou seja, o quanto o valor do peso será acrescido ou decrescido a cada passo.



Fonte: Adaptado de Luger (2013).

Figura 3.7 – Retropropagação com uma camada oculta.



Fonte: Adaptado de Luger (2013).

3.4 APRENDIZAGEM PROFUNDA

Deep learning, ou aprendizagem profunda, permite modelos computacionais compostos de múltiplas camadas de processamento para aprender representações de dados com múltiplos níveis de abstração. Isso possibilita melhorar drasticamente o estado-da-arte de muitas áreas, como reconhecimento de imagens, fala e, como proposto neste trabalho, a descoberta de fármacos (LECUN; BENGIO; HINTON, 2015).

Há, no entanto, diversos algoritmos e formas de se projetar uma rede de *deep learning*. Isso porque uma rede neural pode ser *feedforward* ou recorrente. Uma rede *feedforward* segue um fluxo normal, partindo dos nodos de entrada, alimentando a camada seguinte até a saída. Há ainda a possibilidade de se usar redes de convoluções. Cada camada interna pode ainda ter função de custo e função de ativação diferentes.

A seguir serão abordadas as redes com alimentação para frente (*feedforward*) e as

redes convolucionais.

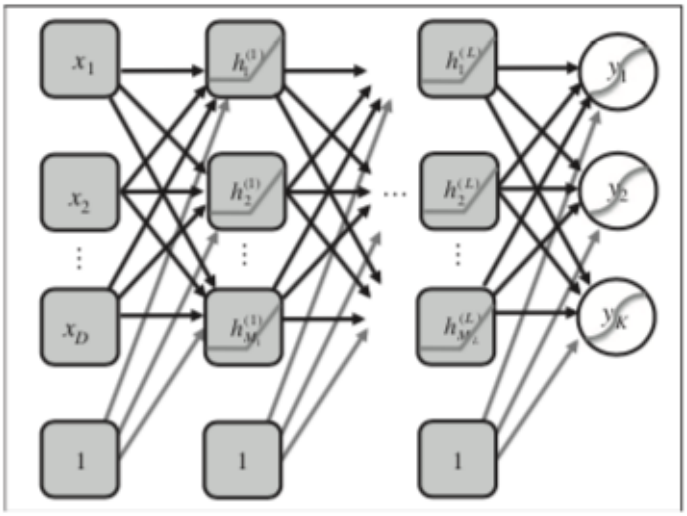
3.4.1 Deep Feedforward

As redes *Deep Feedforward* (DFF), também chamadas de *multilayer perceptrons* (MLPs), são os modelos de *deep learning* mais usados. O objetivo dessas redes é aproximar alguma função f, onde essa função mapeia uma classe y com entrada x e θ representa os parâmetros a serem treinados, conforme Equação 3.11 (GOODFELLOW et al., 2016).

$$y = f(x, \theta) \tag{3.11}$$

O fluxo das redes DFF segue apenas a frente. Não há um retorno onde a saída de um neurônio retorna para si mesmo. Nos casos em que isso ocorre são chamadas de redes neurais recorrentes. A Figura 3.8 mostra um exemplo de rede DFF, onde x representa a camada de entrada, h as camadas ocultas e y a camada de saída (classes).

Figura 3.8 – Exemplo de uma *Deep Feedforward*.



Fonte: Adaptado de Witten et al. (2016).

As redes DFF são de grande importância por serem amplamente utilizadas comercialmente, como, por exemplo, no reconhecimento de objetos a partir de imagens. Há também diferentes formas de se organizar uma rede DFF. A escolha do otimizador, função de ativação, função de custo e a forma das unidades de saída estão entre as decisões a serem tomadas ao se projetar uma DFF. O otimizador é usado para atualizar os valores dos pesos e viés. As funções de ativação são usadas para computar os valores das unidades ocultas. E a função de custo calcula basicamente a diferença entre o valor correto e o

valor predito no momento do treinamento para reduzir o erro (GOODFELLOW et al., 2016; WITTEN et al., 2016).

3.4.2 Deep Convolutional Neural Networks

Rede neural convolucional, ou *Deep Convolutional Neural Network* (DCNN), é um tipo de rede neural especializado em processar dados de topologia espacial. Em uma grade de 1D os elementos representam intervalos regulares de tempo, enquanto 2D representam imagens. O uso mais comum de redes convolucionais é feito em imagens (2D) para identificar objetos, pessoas ou padrões. Há ainda a possibilidade de se usar uma grade 3D. O nome convolucional vem do fato de que a rede utiliza uma operação matemática chamada de convolução (GOODFELLOW et al., 2016).

A Figura 3.9 mostra como uma camada de convolução é executada. É selecionada na entrada uma parte da grade do tamanho do *kernel* e multiplicado produzindo um único elemento na grade de saída. O processo é repetido após a seleção da entrada se deslocar em 1 coluna até acabar a linha, depois segue para as demais linhas. Isso gera uma grade de saída menor que a entrada, reduzindo a dimensão.

Uma camada convolucional típica consiste em 3 estágios. A primeira, executa as convoluções em paralelo para produzir um conjunto de ativações lineares. A segunda, em que cada uma dessas ativações lineares são executadas em uma função não-linear, como uma função de ativação linear retificada (*Rectified Linear Unit* - ReLU). E por último uma função de agrupamento (*pooling*) para modificar ainda mais a saída da camada (GOODFELLOW et al., 2016). Um exemplo de função *pooling* muito usada é a *max pooling* (ZHOU; CHELLAPPA, 1988).

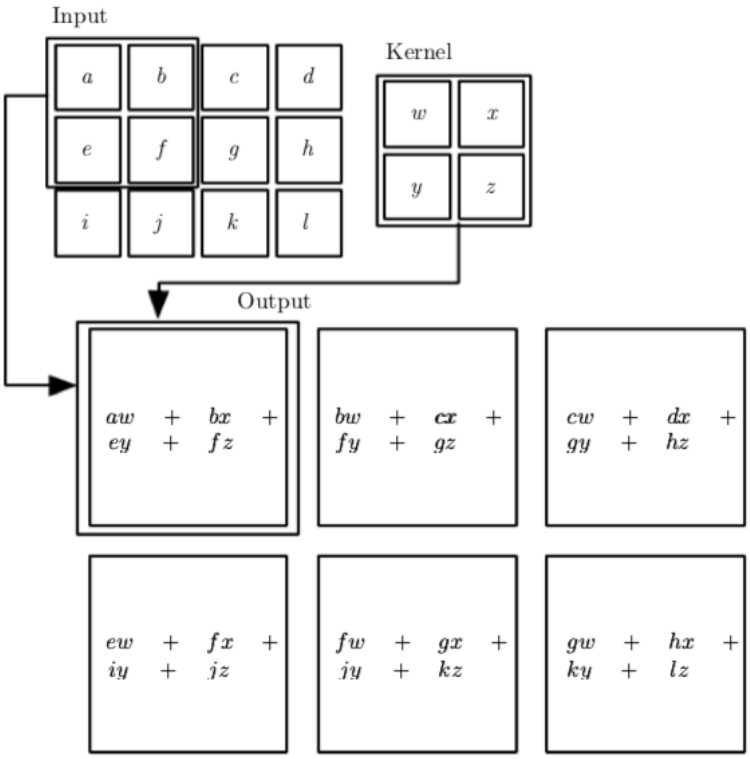
3.5 CONSIDERAÇÕES DO CAPÍTULO

Este capítulo apresentou o referencial teórico a respeito de aprendizado de máquina, em especial sobre as técnicas abordadas nesta dissertação.

Apresentou-se definições acerca de algoritmos de árvores de decisão e de agrupamento de dados, com o intuito de embasar os experimentos com o algoritmo 3D-tri, o qual serviu de base e inspiração para a desenvolvimento deste trabalho e cujos detalhes de implementação e seus experimentos são apresentados no Capítulo 5.

Foi dada especial ênfase à abordagem de redes neurais, principalmente àquelas de aprendizagem profunda, ou *deep learning*. Foram apresentados os conceitos a respeito das redes de aprendizagem profunda de alimentação para frente (*deep feedforward*)

Figura 3.9 – Exemplo de uma convolução 2D. O *kernel* é multiplicado por cada divisão da matriz, deslizando pela grade e produzindo como saída uma grade menor.



Fonte: Adaptado de Goodfellow et al. (2016).

e convolucionais (*deep convolucional neural networks*), as quais são implementadas no contexto desta dissertação e detalhadas na metodologia apresentada no Capítulo 5.

De fato, observa-se trabalhos recentes que exploram redes neurais de aprendizagem profunda aplicadas em problemas de bioinformática, como podem ser observadas as diferentes técnicas e aplicações relatadas no recente artigo de revisão de (LAN et al., 2018). De forma mais específica, o uso de *deep learning* também pode ser observado no contexto de triagem virtual de fármacos, conforme mostra os estudos de (CHENG et al., 2012) e (UNTERTHINER et al., 2014). Mais recentemente, a aplicação de redes neurais *deep learning* em desenho racional de fármacos, foi amplamente explorada e discutida no artigo de revisão de (RIFAIOGLU et al., 2018).

Esses trabalhos relatam a importância do planejamento racional de fármacos, e apresentam abordagens de trabalhar sobre esses dados a partir de redes neurais *deep learning*. Esta dissertação tem como objetivo investigar uma aplicação mais específica: o uso de *deep learning* em dados tridimensionais de docagem molecular considerando o receptor como uma estrutura flexível.

4 TRABALHOS RELACIONADOS

Para investigar o estado-da-arte em relação ao uso de *deep learning* em dados de docagem molecular, foi realizada uma revisão sistemática da literatura. Revisão sistemática é um termo utilizado para uma metodologia de pesquisa específica, desenvolvida para coletar e analisar evidências pertinentes a um tópico específico. Ao contrário dos métodos mais comuns de revisão de literatura, uma revisão sistemática segue um caminho formal e sistemático, conduzido por uma sequência metodológica bem definida de acordo com o protocolo escolhido (BIOLCHINI et al., 2005).

A revisão sistemática deste trabalho é baseada na metodologia definida por Kitchenham (2004) e Biolchini et al. (2005). Tais metodologias foram baseadas em outras metodologias já definidas, como as revisões Cochrane definidas por Clarke e Oxman (2000).

Esta seção aborda as etapas da revisão sistemática realizada, explicitando a formulação da pergunta, fontes selecionadas, seleção de estudos primários, critérios de inclusão e exclusão e extração de informação.

4.1 OBJETIVO E FORMULAÇÃO DA PERGUNTA

Uma revisão sistemática se inicia na fase de planejamento. É nessa etapa onde todo o protocolo é definido. Os primeiros passos na definição do protocolo é definir o objetivo e a formulação da pergunta. A pergunta ou questão deste trabalho norteiam a pesquisa para os seus objetivos.

O objetivo deste capítulo é investigar o estado da arte do uso de técnicas de *deep learning* em dados de docagem molecular. Ou seja, avaliar as abordagens de *deep learning* utilizadas na área pesquisada, e quais as possibilidades de uso das mesmas. Não foi definido um intervalo temporal para seleção dos estudos. A seguinte pergunta foi formulada: *"Quais os trabalhos realizados que utilizam técnicas de deep learning no contexto de docagem molecular e quais as possibilidades de uso dessas técnicas na área?"*

4.2 FONTES SELECIONADAS

Após definir o objetivo e formular a pergunta é necessário definir as palavras chave (*keywords*). Foram definidas como palavras-chave os termos a seguir:

- *deep learning*;

- *docking*.

Depois de definir as *keywords*, foram definidos os critérios de seleção na fonte, sendo estes critérios iniciais de busca dos estudos. Foi definido como critério a existência da *string de busca* no título, nas *keywords* ou no *abstract*. A *string de busca*, ou *search string*, ou ainda *descritor*, foi definida como: *"deep learning"AND "docking"*.

A pesquisa dos estudos primários foi realizada em uns dos repositórios digitais das editoras e organizações mais conhecidas na área da ciência da computação. A lista das fontes é a seguinte: *ACM Digital Library*, *IEEE Computer Society Digital Library*, *Science@Direct* e *Scopus*. Apenas estudos na língua inglesa foram considerados nesta revisão.

4.3 SELEÇÃO DE ESTUDOS PRIMÁRIOS

Após definir os repositórios a serem pesquisados e os termos de busca, a próxima etapa consiste em realizar a busca em si. Para cada repositório mencionado anteriormente foi executado o termo de busca que havia sido definido. A busca na *Scopus*, retornou 18 resultados. Para a *IEEE Computer Society Digital Library* e para *Science@Direct*, foram 5 resultados cada. E para *ACM Digital Library*, apenas 1 resultado foi encontrado. O total de resultados retornados por repositório está sumarizado na Tabela 4.1.

Tabela 4.1 – Número de estudos retornados de cada repositório.

Repositório	Total de resultados
Scopus	18
IEEE Computer Society Digital Library	5
Science@Direct	5
ACM Digital Library	1
Total	29

4.4 CRITÉRIOS DE INCLUSÃO E EXCLUSÃO

Foram definidos como critérios de inclusão apenas estudos primários escritos em inglês disponíveis para *download*. Dessa forma, se exclui estudos como revisões bibliográficas, por não serem estudos primários, estudos em outros idiomas, e estudos indisponíveis por completo (como, por exemplo, aqueles que não são de acesso aberto, *open access*).

Foi definido também como critério de exclusão o tema principal da pesquisa diferente do escopo da pesquisa.

A Tabela 4.2 mostra o número total de estudos retornados pela busca e o número de estudos primários selecionados de acordo com os critérios de inclusão e exclusão. Foram selecionados 6 estudos primários da *Scopus*, 3 da *IEEE Computer Society Digital Library*, nenhum da *Science@Direct* e da *ACM Digital Library*. Após a remoção de 2 estudos duplicados, sobraram 10 estudos, conforme apresentado na Tabela 4.3.

Tabela 4.2 – Número de resultados e estudos primários obtidos de cada repositório.

Repositório	Total de resultados	Estudos selecionados
Scopus	18	6
IEEE Computer Society Digital Library	5	3
Science@Direct	5	0
ACM Digital Library	1	0
Total	29	9

Tabela 4.3 – Número de estudos aceitos, rejeitados e duplicados.

Estudos aceitos	Estudos rejeitados	Estudos duplicados	Total
7	20	2	29

4.5 EXTRAÇÃO DE INFORMAÇÃO

Depois de selecionar os estudos primários, a etapa seguinte trata de extrair informação dos estudos. As informações analisadas nesta revisão foram baseadas em Biolchini et al. (2005). Os autores indicam como extração objetiva de resultados as seguintes informações:

- Identificação do estudo: título, autores e fonte
- Metodologia do estudo: métodos utilizados para conduzir o estudo
- Resultados do estudo: efeito obtido pela execução do estudo
- Problemas do estudo: limitações encontradas pelos autores

4.6 ARTIGOS SELECIONADOS

O estudo intitulado "*Multi-Label Classification Using Deep Belief Networks for Virtual Screening of Multi-Target Drug*" (FITRIAWAN et al., 2016), utilizou dados de digitais *fingerprint* para extrair os atributos e utilizá-los em uma *Deep Belief Network* (DBN) para realizar uma triagem virtual com múltiplos alvos. Uma triagem virtual com múltiplos alvos faz uso de classificação multi-rótulo, onde se tem mais de uma classe. As redes DBN foram desenvolvidas pela primeira vez por Hinton (2007) e utilizam a probabilidade máxima na inicialização dos parâmetros. Os autores também utilizaram a transformação *Binary Relevance* (BR) que transforma o problema multi-rótulo em um problema binário para cada rótulo e um número de classificador binário é aplicado a cada problema individualmente. Como medidas, foram utilizadas a acurácia e taxa de classificação. Também foram utilizadas as medidas de AUC (*Area Under Curve*). Os autores destacam o aumento significativo da acurácia ao aumentar as épocas, atingindo acurácias entre 97% e 98% para o método de melhor resultado.

Em "Molecular descriptors selection and machine learning approaches in protein-ligand binding affinity with applications to molecular docking", Hsieh et al. (2016) propõem um algoritmo para ranquear características e escolhe-las para criar um modelo de classificação. O algoritmo utiliza o coeficiente de correlação no valor do atributo alvo (LE - *Lowest Energy*) e ranqueia as características pelo valor absoluto. Por fim, são selecionados *k* características para gerar o modelo de classificação. Isso permite excluir algumas "caixas"(sub-espaços) com menores chances de obterem sucesso na docagem. Ao comparar SVM, *deep learning* e regressão logística, a abordagem com SVM obteve os melhores resultados.

Pereira, Caffarena e Santos (2016) apresentam o estudo intitulado "Boosting docking-based virtual screening with deep learning" apresenta a *DeepVS*, uma rede neural *deep learning* que usa dados provenientes de docagem molecular como um átomo e tipos de resíduos provenientes da docagem. Uma grande vantagem da *DeepVS* é de não precisar engenharia de características. O resultado foi capaz de reduzir o AUC (*Area Under the Curve*) usando o Autodock Vina para docagem após executar o *DeepVS*.

Zhao e Gong (2017) apresenta o estudo intitulado "Protein-protein interaction interface residue pair prediction based on deep learning architecture" no qual trata de uma abordagem que utiliza a arquitetura *deep learning* chamada *Long-Short Term Memory* (LSTM) para predizer os pares de resíduos de interface da proteína. Primeiro, foram criadas três novos atributos, adicionando-os a outros seis para descrever uma proteína. Esses três atributos são: área de contato interior, entre átomos em um resíduo; área de contato exterior, com outros resíduos; área exterior vazia, sem qualquer contato com outros resíduos. Depois foram usados dois limiares para selecionar os pares de resíduos mais propensos a formarem interface proteína-proteína. Essa etapa aumenta a proporção de pares de

resíduos de interface e reduz a influência de dados desbalanceados. Por último, foram construídas as redes LSTMs para organizar e refinar a predição de pares de resíduos de interface. Para o melhor modelo, a acurácia foi superior a 62%. Isso ajuda a entender os mecanismos de interação proteína-proteína e para orientação em experimentos biológicos.

No trabalho intitulado “CGBVS-DNN: prediction of compound-protein interactions based on deep learning“ de Hamanaka et al. (2017) é apresentado a CGBVS-DNN, uma evolução da CGBVS, que utiliza SVM. A principal vantagem da CGBVS é que ela possibilita CPIs (*Compound-Protein Interactions*) sem prever estruturas tridimensionais conhecidas. Apesar do bom desempenho da CGBVS, SVMs requerem um aumento exponencial de recursos computacionais conforme aumenta o tamanho dos *datasets*. Devido a isso, buscou-se estratégias de *deep learning*. *Deep learning* é considerada melhor que SVM porque requer uma menor quantidade de memória, pois o aprendizado pode ser realizado em lotes. O artigo cita ainda que geralmente um pré-treinamento não supervisionado melhora o desempenho da aprendizagem profunda. Isso porque os pesos são iniciados de forma randômica frenquentemente não transmite nenhuma informação, o que faz com que seja difícil de ocorrer a retropropagação. Foi usada então uma *Deep Belief Network* (DBN), que, para o chamado pré-treino, utiliza camadas de Restricted Boltzmann Machines (RBM). O resultado foi uma acurácia de 98.2%.

O trabalho de Sunseri et al. (2018) intitulado “Convolutional neural network scoring and minimization in the D3R 2017 community challenge“ trata de funções de *score* baseada em uma rede neural convolucional. Funções essas usadas em várias tarefas comuns no domínio da descoberta de fármacos. Dentre essas tarefas está a classificação da posição do ligante dado um conjunto de receptores de referência e classificando esses ligantes em ativos ou inativo usando informação estrutural. A rede neural convolucional foi usada para recalcular o *score* ou refinar as posições gerando uma função de *score* convencional, Autodock Vina, e comparar essa abordagem com função de *score* convencional sem esse recalculo do *score* ou refinamento de posições. O resultado foi superior ao comparado na maioria das tarefas, sem exigir uma inspeção manual por um especialista. Por outro lado, a predição de posição para o alvo escolhido, *Cathepsin S*, foi desafiador para a docagem *de novo*. Apesar disso, os autores ressaltam que a rede neural convolucional teve melhor desempenho em várias tarefas de triagem virtual, mostrando a relevância do uso de *deep learning* no processo de triagem virtual.

Costa et al. (2018) apresentam o trabalho “Perturbation theory/machine mearning model of ChEMBL data for dopamine targets: docking, synthesis, and assay of new l-prolyl-l-leucyl-glycinamide peptidomimetics“ que trata de modelos de teoria de pertubações, que permitem a predição de propriedades de uma consulta de compostos ou sistema molecular em ensaios experimentais. É criado, segundo os autores, o primeiro estudo que utiliza aprendizagem de máquina com teoria de pertubações de uma grande base de dados *ChEMBL* de ensaios pré-clinicos de compostos que tem como alvo proteínas das vias

dopaminérgicas. O melhor modelo encontrado teve precisão de 70% a 91% para 5.000 casos nas séries de treinamento com validação externa. O modelo proposto foi também comparado com modelos não-lineares (ANN, *random forest*, *deep learning*, etc). Alguns obtiveram melhores resultados que o trabalho, mas a custo de um incremento considerável na complexidade do modelo. Foi realizada também um estudo de docagem molecular para alguns dos compostos no software Vina AutoDock. O modelo preditivo em questão teve resultados melhores de novos compostos em um grande número de novos ensaios.

A Tabela 4.4 mostra um comparativo entre os trabalhos selecionados, apresentando nas considerações onde as técnicas de *deep learning* são usadas e a comparação com este trabalho.

4.7 OUTROS TRABALHOS RELACIONADOS

O trabalho desenvolvido por Winck (2012) apresenta o algoritmo 3D-Tri, que faz uso das propriedades tridimensionais para induzir uma árvore de regressão com dados de docagem com dinâmica molecular. O algoritmo utiliza a ideia de gerar blocos para cada átomo na escolha do atributo da divisão do nó da árvore. É escolhido o melhor grupo para cada átomo, e então o átomo com o melhor grupo geral é utilizado na divisão. Para gerar esses blocos é utilizado o algoritmo de agrupamento K-means. O algoritmo foi posteriormente implementado em Python por ser uma linguagem amplamente utilizada para aprendizagem de máquina (SILVA, 2015).

No trabalho de Kronbauer (2016) foi utilizada uma estratégia de densidade para gerar os grupos. Para isso, foi escolhido o algoritmo DBSCAN. O estudo mostrou ser possível compreender os átomos que influenciam positivamente no resultado da docagem. Mostrou também ser possível reduzir a busca por regiões promissoras em pelo menos 50% (KRONBAUER; MACHADO; WINCK, 2016).

4.8 CONSIDERAÇÕES DO CAPÍTULO

Este capítulo apresentou trabalhos relacionados usando uma revisão sistemática. Foi possível notar que o uso de técnicas *deep learning* em dados de docagem molecular é algo recente e em constante crescimento. Isso reforça a necessidade de se investigar mais estratégias possíveis na área.

Tabela 4.4 – Comparativo dos trabalhos selecionados.

Trabalho	Consideração
(FITRIAWAN et al., 2016)	São usadas DBN para ajudar no processo de triagem virtual. São usados dados de *molecular fingerprint*. Em comparação com este trabalho, os dados usados nos treinamentos são diferentes, além das técnicas de *deep learning*.
(HSIEH et al., 2016)	*Deep learning* foi uma das técnicas testadas no trabalho que utilizou dados de descritores moleculares. A técnica de *deep learning* utilizada foi baseada em regressão logística.
(PEREIRA; CAFFARENA; SANTOS, 2016)	Utiliza dados de docagem, assim como o presente trabalho. Também utiliza redes convolucionais. Como atributos utiliza informações do átomo, como tipo, carga, distância e tipo de aminoácido.
(ZHAO; GONG, 2017)	Faz uso de LSTMs para predizer resíduos de docagem. Objetivo e técnica diferente deste trabalho.
(HAMANAKA et al., 2017)	Utiliza dados estruturais e descritores moleculares para predizer interações usando redes convolucionais. Difere do presente trabalho nos tipos de dados usados.
(SUNSERI et al., 2018)	Trata de uma função de *score*, onde utiliza redes convolucionais para tal. Tem foco diferente deste trabalho, apesar de ser relacionado.
(COSTA et al., 2018)	Utilizou dados de diferentes proteínas, linhas celulares, organismos, etc, para treinar o classificador *deep learning* com uma única camada oculta, com 1.000 neurônios. Não utiliza dados semelhantes aos deste trabalho.

5 METODOLOGIA

A evolução das configurações de computadores tem permitido que experimentos de dinâmica molecular sejam executados com trajetórias cada vez mais extensas. Por consequência, experimento de docagem molecular que consideram a flexibilidade do receptor passam a ser executados com mais dados de conformações de proteínas. Assim, os avanços da era genômica, aliado à área de desenho racional de fármacos, tem produzido cada vez mais dados biológicos que demandam análises e processamento. Técnicas de aprendizado de máquina e mineração de dados que processam esses dados biológicos têm sido fundamental para que experimentos sejam realizados para reduzir tempo e custo, bem como para propor novas técnicas e descobertas com importantes e positivos impactos sociais.

Essa circunstância é corroborada pela revisão de (LAN et al., 2018), que descreve que o desafio está em como combinar as técnicas de bioinformática com os métodos avançados de análises de dados, com a finalidade de interligar sistematicamente as duas áreas e, assim, minerar dados biológicos com sucesso. Os autores dão ênfase a diferentes estratégias de pré-processamento e técnicas de mineração de dados sobre dados de bioinformática, em especial quando esses dados são submetidos a estratégias de redes neurais de aprendizagem profunda.

Esta dissertação apresenta abordagens baseadas em *deep learninig* para processar dados de dinâmica molecular, com o objetivo de apontar conformações de um dado receptor que sejam as mais promissoras para serem utilizados em experimentos de docagem molecular, para um determinado ligante. Para tanto, este trabalho propõe abordagens que utilizam técnicas de *deep learning* para criar modelos preditivos que consideram a representação tridimensional das conformações de proteínas, obtidas por meio de simulações de dinâmica molecular.

5.1 MOTIVAÇÃO

Para Rifaioglu et al. (2018), o desenvolvimento de novas drogas continua sendo um dos principais desafios para a bioinformática, onde métodos computacionais de inteligência artificial tem sido utilizados por quase três décadas para auxiliar a compreensão dos mecanismos moleculares e propor novos tratamentos para diferentes doenças. Os autores destacam que os avanços do poder computacional e da análise e inferência de dados com técnicas de aprendizagem de máquina, em especial aquelas de *deep learning*, oferecem diferentes oportunidades quando aplicadas no contexto de experimentos *in silico* de desenho racional de fármacos.

Alguns trabalhos que antecedem a abordagem implementada nesta dissertação foram propostos para selecionar conformações promissoras para novos experimentos de docagem molecular, ao considerar a flexibilidade do receptor utilizando-se de dados de trajetórias obtidas por meio de simulações de dinâmica molecular.

Em (WINCK et al., 2009), os autores apresentam uma metodologia de processamento dos dados para construção de um data set, baseado na distância em Angstrom (Å) de cada resíduo de amino ácido do receptor em relação ao ligante. Assim, produzia-se um data set $X_{n,m}$ mais o atributo alvo, onde:

- X diz respeito a um data set para um receptor e um ligante;
- n atributos, dos quais $n-1$ corresponde ao número de resíduos de amino ácido que compõem o receptor;
- m é o número de conformações resultantes da trajetória obtida pela simulação de dinâmica molecular, que convergiram em experimentos de docagem molecular;
- cada célula do data set corresponde à distância, em Å, do resíduo para o ligante;
- para cada conformação m é atribuído o valor de FEB correspondente, obtido dos experimentos de docagem molecular.

Diferentes abordagens de mineração de dados foram aplicadas sobre essas configurações de data set, onde exemplifica-se os trabalhos de (MACHADO et al., 2010), (WINCK et al., 2013) e (MACHADO et al., 2011). Dentre as abordagens desses trabalhos, destacam-se as árvores de decisão para classificação e para regressão.

Como a distância do ligante para um determinado resíduo só é possível de ser obtida a partir de experimentos de docagem molecular, foi proposto um algoritmo que desvinculava o pré-processamento dos atributos preditivos dos experimentos de docagem molecular, apresentando uma configuração de data set que apresenta a representação tridimensional da proteína (WINCK, 2012). Nessa estratégia, é proposto o processamento de um data set $X_{n\times 3,m}$ mais o atributo alvo, onde:

- X diz respeito a um data set para uma trajetória de dinâmica molecular de um receptor e os resultados de docagem deste receptor com um ligante;
- n corresponde ao número de átomos que compõem o receptor, sendo que cada átomo é representado por cada uma de suas coordenadas espaciais (x, y, z nessa ordem), tendo uma dimensão de $n \times 3$ colunas;
- m é o número de conformações resultantes da trajetória obtida pela simulação de dinâmica molecular, que convergiram em experimentos de docagem molecular;

- cada célula do data set representa o valor correspondente à posição da respectiva coordenada de cada átomo;
- para cada conformação m é atribuído o valor de FEB correspondente, obtido dos experimentos de docagem molecular.

A partir dessa abordagem de pré-processamento para geração do data set, propôs-se um algoritmo de indução de árvore de decisão baseado em agrupamento de dados por *k-means* para identificar quais são os átomos e suas respectivas regiões atômicas favoráveis a experimentos de docagem molecular (WINCK, 2012). Este algoritmo, que foi aperfeiçoado e reimplementado em (SILVA, 2015), induz árvores que mostram as regiões espaciais mais favoráveis dos átomos mais relevantes para predizer se uma determinada conformação de proteína é promissora para obter um bom experimento de docagem com um dado ligante.

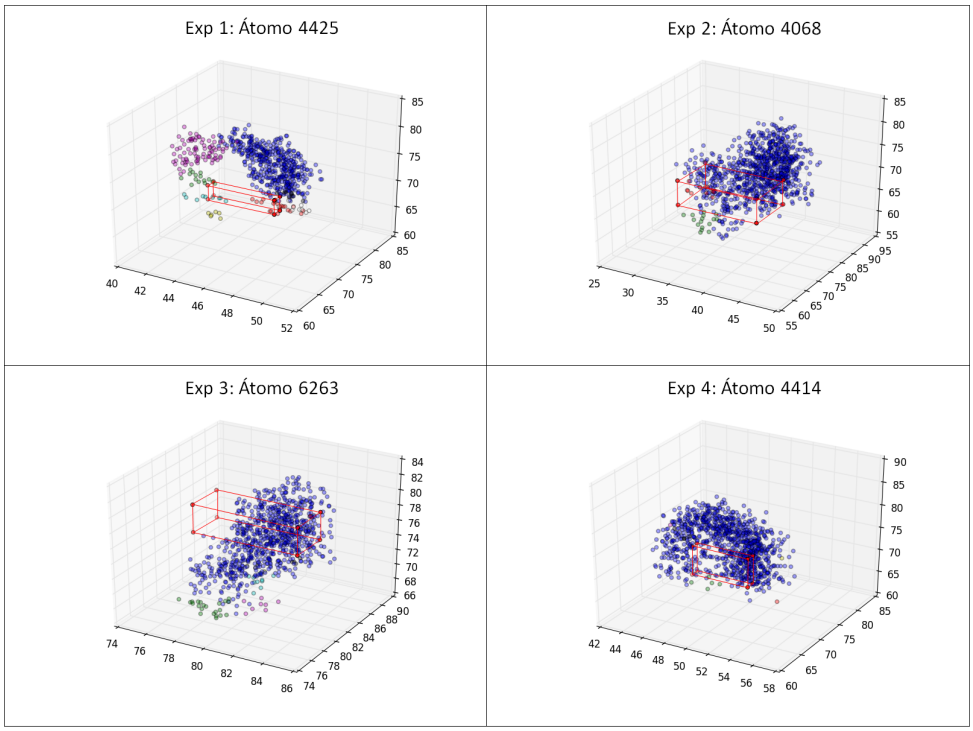
Posteriormente, a abordagem de agrupamento de regiões atômicas foi remodelada e reimplementada, fazendo-se uso de uma técnica baseada em densidade (KRONBAUER, 2016; KRONBAUER; MACHADO; WINCK, 2016). Esse estudo mostrou que é possível compreender os átomos que mais influenciam positivamente no resultado de docagem molecular, quando considerada a flexibilidade do receptor, reduzindo a busca por regiões promissoras em pelo menos 50%.

A Figura 5.1 ilustra o conceito das melhores regiões atômicas encontradas, por meio do algoritmo de agrupamento DBSCAN. Nessa figura é possível observar 4 átomos da proteína AcrB. Os pontos no espaço demonstram as coordenadas assumida por cada átomo, em cada configuração. As diferentes cores dos pontos representam os grupos encontrados pelo algoritmo. A caixa em vermelho representa o bloco em que a abordagem de (KRONBAUER; MACHADO; WINCK, 2016) identifica como sendo a melhor região da flexibilidade do átomo, para o ligante testado. Esses blocos são capturadas no formato $[(x_i, x_f), (y_i, y_f), (z_i, z_f))$, onde x, y e z representam os eixos, sendo que i e f representam os valores iniciais e finais, respectivamente, para cada eixo, formando o bloco.

A Figura 5.2 ilustra um recorte dos três primeiros níveis de uma árvore de decisão da proteína AcrB, induzida pelo algoritmo de (SILVA, 2015), utilizando o algoritmo de agrupamento DBSCAN para geração dos blocos. Observa-se que cada nodo é um átomo, representado pelo bloco identificado pelo algoritmo de agrupamento, sendo que as arestas indicam se o átomo está dentro ou fora do bloco. Por fim, as folhas representam o valor de médio de FEB para cada caminho da árvore.

Na busca por implementar soluções mais robustas de aprendizagem de máquina, esta dissertação agrega técnicas de *deep learning*, consideradas promissoras em diversos contextos, ao conhecimento adquirido nos trabalhos anteriores citados nesta seção, a fim de obter melhores resultados de seleção de conformações de proteínas, considerando suas propriedades tridimensionais.

Figura 5.1 – Agrupamentos por DBSCAN dos átomos número 4425, 4068, 6263 e 4414 da proteína AcrB, e seus respectivos blocos



Fonte: (KRONBAUER, 2016)

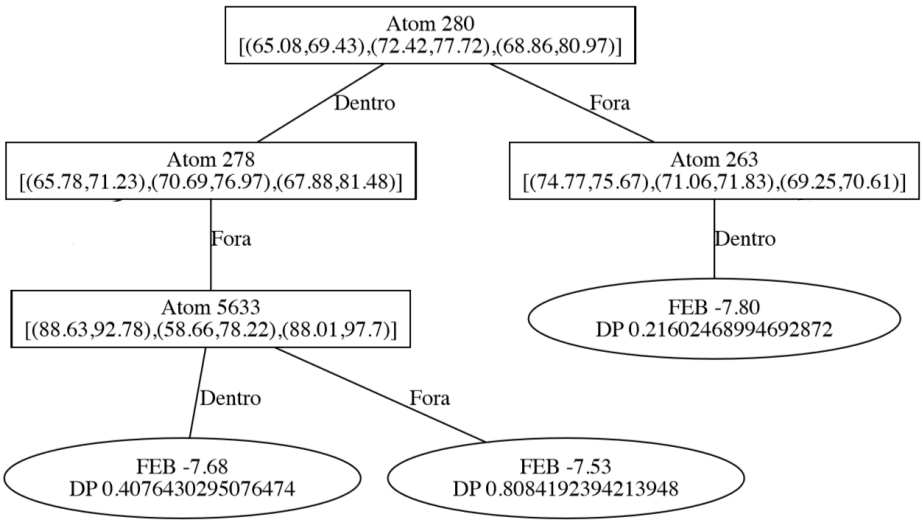
5.2 CONJUNTO DE DADOS

Esta dissertação segue a abordagem de utilização de um data set que represente as conformações de proteínas obtidas por simulações de dinâmica molecular, onde as mesmas são representadas pelas coordenadas espaciais de seus átomos, tal qual descrito na seção 5.1 e apresentado em (WINCK, 2012; SILVA, 2015; KRONBAUER, 2016; KRONBAUER; MACHADO; WINCK, 2016).

Para tanto, foi desenvolvido o módulo de pré-processamento dos dados, em linguagem Python. Esse código lê e processa todos os arquivos PDB provenientes da dinâmica molecular para extração das coordenadas espaciais de cada átomo, além de ler e processar todos os arquivos provenientes dos resultados de docagem, associando os mesmos a cada conformação da proteína, e extraindo o respectivo resultado de FEB.

Os dados utilizados nessa dissertação são da proteína AcrB (*Acriflavine resistance protein B*), gerados por (PRATES, 2014), cujas conformações foram obtidas a partir de uma simulação por dinâmica molecular de 50 ns. Dessas conformações foram selecionadas um total de 1001 conformações ao longo do total de conformações. Essa proteína contém um total de 1.029 resíduos de aminoácido, compostos por um total de 9.661 átomos. A simulação completa foi submetida a experimentos de docagem molecular, utilizando-se o ligante NUNL02, conhecido como um inibidor da bomba de efluxo AcrB. As simulações por dinâmica molecular, bem como os experimentos de docagem molecular foram planejados e

Figura 5.2 – Recorte dos três primeiros níveis de uma árvore gerada pelo 3D-Tri com o algoritmo de agrupamento DBSCAN, para a proteína AcrB



Fonte: O autor

executados pelo Grupo de Biologia Computacional da Universidade Federal do Rio Grande, e estão descritos em (PRATES, 2014).

Considerando a configuração do data set para os dados descritos, gerou-se um data set que, a partir de agora, será referenciado por D_{AcrB}. Esse possui um total de 1.001 linhas e 28.983 colunas, as quais representam as coordenadas espaciais dos átomos, além da coluna para o FEB, que é o atributo alvo.

A Tabela 5.1 ilustra uma representação reduzida dos dados contidos em D_{AcrB}, onde a primeira coluna refere-se à coordenada x do átomo *Átomo 1*, seguido pelas coordenadas y e z do mesmo átomo. As demais colunas seguem a mesma estrutura para a sequência dos demais átomos. Por fim, na última coluna, aparece o atributo alvo *FEB*.

Tabela 5.1 – Exemplo de coordenadas para o *Dataset* utilizado. A letra A representa um átomo, seguido pelo número do átomo e o eixo (x, y ou z).

A 1 x	A 1 y	A 1 z	...	A 9661 x	A 9661 y	A 9661 z	FEB
44,270	87,490	68,170	...	42,880	68,730	99,270	-7.6
28,280	62,050	68,620	...	23,220	96,380	61,910	-8,2
33,880	85,670	66,700	...	37,830	97,040	96,670	-8,1
...	...	...	...	...	...	...	...
28,580	82,820	50,830	...	26,760	90,990	93,180	-8,6
40,920	79,900	44,710	...	28,500	90,330	101,830	-8,5
38,820	85,320	59,210	...	25,440	87,790	95,080	-7,1

As duas seções seguintes abordam a metodologia de desenvolvimento desta dissertação para uma rede neural *feedforward* (seção 5.3) e para uma rede neural que utiliza convoluções (seção 5.4).

5.3 REDES NEURAIS FEEDFORWARD

Primeiramente, foi definida uma rede neural *feedforward* que utiliza o arquivo D_{AcrB} pré-processado como entrada e retorna como saída um valor de FEB. A arquitetura dessa rede neural é representada na Figura 5.3. A camada de entrada recebe as coordenadas. As camadas ocultas possuem 256, 128, 64, 32 e 16 unidades respectivamente. A camada de saída representa o valor de FEB. A essa rede é dado o nome de **Modelo DFF 1**.

Figura 5.3 – Arquitetura das redes neurais *feedforward* propostas. A rede neural possui uma camada de entrada, cinco camadas ocultas e uma camada de saída com o valor do FEB.

Em **Modelo DFF 1**, as camadas ocultas utilizam a função de ativação ReLU, dada pela Equação 5.1, onde $x = (i * w) + b$, sendo que i representa o valor do atributo, w representa o peso e b o invés.

O erro médio quadrático (MSE) foi usado como função de custo, e o algoritmo Adam foi utilizado como otimizador. Esse algoritmo é baseado em gradiente descendente estocástico e calcula taxas de aprendizagem adaptativa para cada parâmetro (KINGMA; BA, 2014; RUDER, 2016).

$$\text{ReLU}(x) = \begin{cases} x & \text{se} \quad x \geq 0 \\ 0 & \text{caso contrário} \end{cases} \tag{5.1}$$

A partir dessa primeira rede DFF (**Modelo DFF 1**), foram definidas mais duas redes que utilizam a definição de grupos proposta pelo algoritmo 3D-Tri (WINCK, 2012) e por (KRONBAUER, 2016; KRONBAUER; MACHADO; WINCK, 2016). A ideia consiste de, em vez de utilizar as coordenadas de cada átomo como entrada, são usadas as distâncias em relação ao centro do bloco de cada átomo. Isso reduz a dimensionalidade da camada de entrada para um terço da dimensionalidade original, o que acaba por reduzir o tempo de processamento do treinamento. A redução do tempo de processamento se acentua quanto maior for o número de épocas do treinamento. Isso porque a geração dos blocos consome um tempo adicional nessas abordagens. Essas duas outras redes DFF foram nomeadas **Modelo DFF 2** e **Modelo DFF 3**. A rede **Modelo DFF 2** utiliza o DBSCAN como algoritmo de agrupamento, e a rede **Modelo DFF 3**, o K-means.

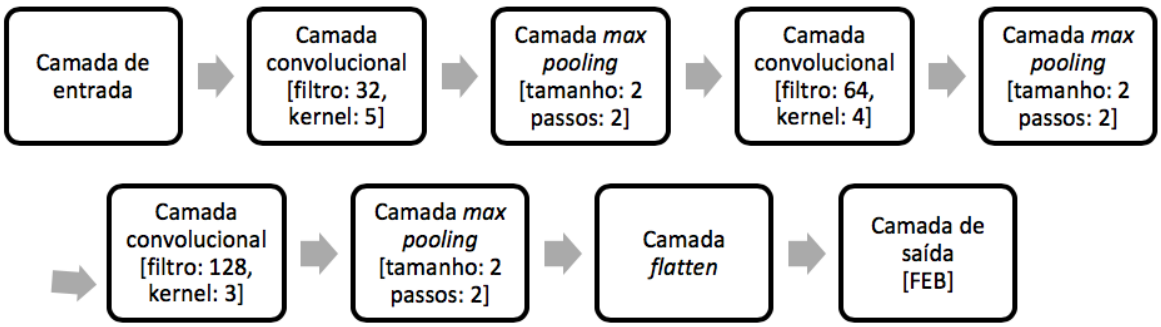
Vale ressaltar que nas abordagens dos Modelos **DFF 2** e **DFF 3**, quando o átomo está fora do bloco identificado pelo algoritmo, em vez da distância para o centro, é atribuído o valor 0.

5.4 REDES NEURAIS CONVOLUCIONAIS

Da mesma forma que as redes DFF, foi definida uma rede neural convolucional que não utiliza os blocos dos átomos para servir de referência. Primeiramente, o arquivo csv é carregado, sendo gerado uma grade 3D com os eixos escalados para um valor usado como parâmetro (tamanho da área). Essa conversão é dada pela Equação 5.2, onde y a saída, x a entrada, t o tamanho da área, min o menor valor de coordenada de todo *dataset* e max o maior valor. Essa transformação é necessária devido os limites de alocação de memória. A camada de entrada recebe essas grades 3D, que segue para a camada convolucional. Após a camada de convolução, há uma camada *max pooling* para reduzir a dimensão do grade. Essa sequência de camada convolucional seguida por uma camada de *max pooling* ocorre três vezes. Na sequência, uma camada *flatten* desfaz as grades para alimentar a camada de saída. A arquitetura das redes convolucionais é representada na Figura 5.4.

$$y = \frac{t(x - min)}{max - min} \tag{5.2}$$

Figura 5.4 – Arquitetura das redes neurais convolucionais propostas. A rede neural possui uma camada de entrada, três sequências compostas por uma camada convolucional e uma *max pooling* cada. Seguido por uma camada *flatten* e uma camada de saída com o valor do FEB.



O modelo de rede neural convolucional base, chamado de **Modelo DCNN 1**, utiliza um canal que representa a presença ou não de um átomo qualquer nessa posição da grade. Os modelos que utilizam os blocos utilizam dois canais. Sendo o primeiro canal, igual ao **Modelo DCNN 1**, e o segundo dado pela Equação 5.3, sendo x a distância euclidiana da posição do átomo em relação ao centro de seu bloco e y o valor atual do canal de distância. O **Modelo DCNN 2** utiliza os blocos gerados usando o DBSCAN como algoritmo de agrupamento, enquanto o **Modelo DCNN 3** utiliza o K-means.

$$f(x, y) = \begin{cases} x & \text{se} \quad x \leq y \quad \text{e o átomo estiver dentro do bloco} \\ y & \text{caso contrário} \end{cases} \tag{5.3}$$

5.5 IMPLEMENTAÇÃO DOS ALGORITMOS

Para implementar as redes neurais *deep learning* propostas neste trabalho, é utilizada a biblioteca *Keras*, uma *API* (*Application Programming Interface*) alto-nível escrita em Python que utiliza *TensorFlow*, *CNTK* e *Theano*. No caso deste trabalho em específico, o *Keras* foi usado como camada acima do *Tensorflow*. *Tensorflow* é uma biblioteca de código aberto para aprendizagem de máquina desenvolvido pela *Google Brain* utilizada para projetar e implementar redes neurais, permitindo modelar a estrutura da rede neural conforme a necessidade (ABADI et al., 2016). Cada nodo na rede neural, é chamado de tensor, criando um fluxo inciando do tensor de entrada e terminando pelo tensor de saída. A biblioteca *SciPy* foi utilizada para a execução do K-means, e a *scikit-learn* para o DBSCAN. Além disso, a biblioteca *NumPy* foi usada para manipulação de matrizes e carregamento de arquivos csv.

O treinamento é dado pelo Algoritmo 1, que recebe como argumentos o nome do arquivo csv que será carregado, o número de épocas a serem executadas, o tipo de RNA a ser utilizada, o método de agrupamento usado para gerar os blocos e a área da grade (para as redes convolucionais). Primeiramente, o *Dataset* é carregado retornando um objeto que contém os dados. A RNA é gerada de acordo com o tipo especificado, já recebendo no construtor o *Dataset*, método de agrupamento e, para as convolucionais, a área da grade. Na sequência o modelo é construído a partir da arquitetura definida pela classe da RNA. Esse modelo é um objeto da classe *Sequential* do *Keras*. É exibido um resumo da estrutura desse modelo na linha subsequente. O treinamento do modelo é realizado na linha 11. Após, um gráfico é gerado. Depois, o modelo e o histórico são salvos no disco. Por fim, são exibidos o tempo de processamento, MSE e MAE.

Algoritmo 1: Algoritmo de treinamento

```
1  def treina(arquivo: string, quantidadeÉpocas: int, tipo: string, método: string,
   áreaGrade: int):
2      Dataset ← Carrega o Dataset a partir do arquivo informado
3      RNA ← nova RNA do tipo especificado, informando o Dataset, método e
       áreaGrade
4      Modelo ← RNA.constróiModelo()
5      Exibe um resumo da estrutura da Modelo
6      Histórico ← Modelo.ajusta(RNA.x, RNA.y, quantidadeÉpocas, validação,
       tamanhoLote)
7      Gera o gráfico
8      Salva o modelo no disco
9      Salva o histórico no disco
10     Exibe o tempo de processamento
11     Exibe MSE e MAE
```

Como observado no algoritmo de treinamento, uma etapa fundamental não é exe-

cutada nesse algoritmo, mas sim em uma classe definida a parte. O Algoritmo 1 mostra como os dados dos blocos são tratados para serem usados nas redes neurais DFF. Primeiramente, os dados do csv são carregados. Depois, é chamado um método que gera o bloco (baseado no 3D-Tri). Na sequência, são calculadas as quantidades de instâncias e átomos. Então, para cada átomo de cada instância é verificado se o átomo em questão está dentro do bloco desse mesmo átomo. Se positivo, o valor do atributo desse átomo é a distância euclidiana entre o átomo e o centro do bloco. Caso contrário, será zero. São retornados o *Data_x* com os atributos para o treinamento, contendo um atributo por átomo, e uma outra matriz com os valores de FEB.

Algoritmo 2: Carrega e processa dados para as redes DFF que utilizam os blocos

```
1  def carregaDatasetDFF(arquivo: string, método: string):
2      Dados ← Carrega Dataset a partir do arquivo informado
3      Blocos ← geraBlocos(dados)
4      QuantidadeInstâncias ← contaInstancias(dados)
5      QuantidadeÁtomos ← contaAtomos(dados)
6      Para cada instância i = 1,...,QuantidadeInstâncias faça
7          Para cada atomo j = 1,...,QuantidadeÁtomos faça
8              se o átomo da instância está dentro do bloco então
9                  Data_x[i][j] ← distância euclidiana entre o átomo e o centro do bloco
10             senão
11                 Data_x[i][j] ← 0
12             fim
13         fim
14         Data_y[i][0] ← FEB da instância atual
15     fim
16     retorna Data_x, Data_y
```

Os dados das redes convolucionais são tratados de maneira diferente. O Algoritmo 1 mostra como os dados são tratados para serem usados nas redes convolucionais. O processo ocorre semelhante ao das redes DFF até a linha 5. Na linha 6 inicia-se uma grade 3D de dimensões dadas pelo parâmetro *tamanhoGrade* com valores 0. Para cada átomo de cada instância são calculadas as coordenadas transformadas, dentro da escala que vai de 0 até *tamanhoGrade*. Atribui-se o valor 1 para o canal 0, indicando que nesse ponto da grade existe um átomo. Testa-se então se o átomo está dentro do bloco gerado para o mesmo. Se positivo, é calculada a distância euclidiana entre o átomo e o centro do bloco. Dessa forma é possível testar se o valor do canal 1 daquela posição da grade é maior que o valor da distância ou ainda não foi definido. Caso positivo, esse valor é atualizado para o valor da distância calculada anteriormente. Com isso, o valor do canal 1 para essa posição será sempre a distância o átomo mais próximo do seu centro naquela posição.

Algoritmo 3: Carrega e processa dados para as redes DCNN que utilizam os blocos

```
1  def carregaDatasetDCNN(arquivo: string, método: string, tamanhoGrade: int):
2      Dados ← Carrega Dataset a partir do arquivo informado
3      Blocos ← geraBlocos(dados)
4      QuantidadeInstâncias ← contaInstancias(dados)
5      QuantidadeÁtomos ← contaAtomos(dados)
6      Data_x ← inicia a grade 3D
7      Para cada instância i = 1,...,QuantidadeInstâncias faça
8          Para cada atomo j = 1,...,QuantidadeÁtomos faça
9              coordX ← escalaValor(atomo.x)
10             coordY ← escalaValor(atomo.y)
11             coordZ ← escalaValor(atomo.z)
12             Data_x[i][coordX][coordY][coordZ][0] ← 1
13             se o átomo da instância está dentro do bloco então
14                 distância ← distância euclidiana entre o átomo e o centro do bloco
15                 se Data_x[i][coordX][coordY][coordZ][1] > distância ou
                   Data_x[i][coordX][coordY][coordZ][1] = 0 então
16                     Data_x[i][coordX][coordY][coordZ][1] ← distância
17                 fim
18             fim
19         fim
20         Data_y[i][0] ← FEB da instância atual
21     fim
22     retorna Data_x, Data_y
```

5.6 CONSIDERAÇÕES DO CAPÍTULO

Neste capítulo foi apresentada a metodologia deste trabalho. Foram definidas duas arquiteturas de redes neurais. A primeira, uma *Deep Feedforward*, e a segunda, uma rede neural convolucional. Para ambas foram criados 3 tipos de redes, uma que não utiliza os dados dos blocos, uma segunda que utiliza as distâncias em relação ao centro do bloco gerado por DBSCAN, e a última, que utiliza as distâncias em relação ao centro do bloco gerado por K-means. Foi também definido o algoritmo de treinamento que utiliza os dados dos blocos e como esses dados são processados. Por fim, foram definidos os testes a serem executados.

6 RESULTADOS

Neste capítulo são descritos os resultados obtidos das implementações dos algoritmos. Primeiramente é ilustrado como se dão os casos de testes, definindo os parâmetros utilizados. Posteriormente são apresentados os testes da abordagem utilizando redes *Deep FeedForward*, seguindo dos resultados dos testes que utilizam uma abordagem convolucional. Na sequência, traz-se um comparativo com os resultados do algoritmo 3D-Tri (WINCK, 2012).

Os testes dos algoritmos implementados neste trabalho foram executados na *Amazon AWS* usando o serviço EC2 (*Elastic Compute Cloud*). Foi utilizada uma instância do tipo "c5.large", uma instância otimizada para cargas de trabalho com uso intensivo de computação com alto desempenho. Essa instância possui 2 vCPU e 4GB de memória. Além disso, vale ressaltar que a configuração utilizou SSD (*Solid-State Drive*).

6.1 CASOS DE TESTES

Para testar o algoritmo são utilizados os dados relatados no Capítulo 5, com diferentes parâmetros. Também são realizados testes com uma rede neural simples.

Os parâmetros utilizados são:

- **Número de épocas.** Determina quantas vezes o processo de treinamento será executado;
- **Tamanho do lote.** Determina quantos elementos serão processados por vez (usado apenas nas redes convolucionais);
- **Tamanho da área.** Tamanho de cada eixo na grade (usado apenas na rede convolucional);
- **Função de custo.** Determina qual função de custo será utilizada pela rede neural;
- **Otimizador.** Determina qual função será usada para atualizar os valores de pesos da rede neural;
- **Taxa de aprendizagem.** Determina o quanto os pesos serão atualizados pelo otimizador a cada iteração.

Além dos parâmetros das redes neurais, cabe destacar alguns parâmetros utilizados para a geração dos grupos. Para o DBSCAN é calculado o *Eps* a partir de um parâmetro, com valor entre 0 e 1, multiplicado pela variância do eixo de maior variância do átomo.

Também é usado como parâmetro o **número mínimo de exemplos**. E para a definição do bloco dos átomos são utilizados dois parâmetros: **LimiteErroBloco** e **LimitePopBloco** definidos por Winck (2012).

As redes DFF serão comparadas entre si para avaliar a estratégia da utilização dos blocos dos átomos. As mesmas comparações são feitas entre as redes convolucionais visando avaliar a estratégia de uso dos blocos dos átomos. Por fim, os resultados são comparados com o algoritmo 3D-Tri (WINCK, 2012).

6.2 RESULTADOS COM REDES DEEP FEEDFORWARD

Nos primeiros testes foram executados 3 tipos de redes neurais DFF (*Deep Feed-Forward*):

- Modelo DFF 1, abordagem simples onde os dados pré-processados foram inseridos na rede;
- Modelo DFF 2, com as distâncias em relação ao centro gerado por DBSCAN;
- Modelo DFF 3, DFF com as distâncias em relação ao centro gerado por K-means.

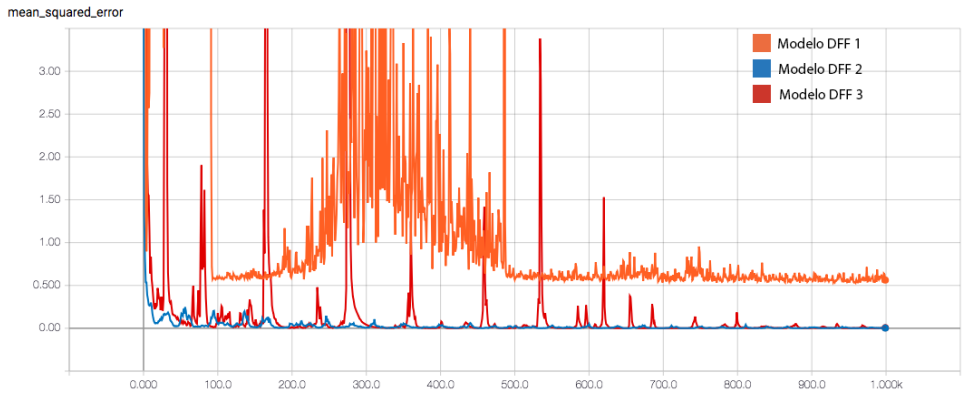
Cada modelo foi treinado com 1.000 épocas, usando MSE como função de perda e avaliado por MAE e MSE do conjunto total para que possam ser comparados entre si e com o modelo gerado pelo algoritmo 3D-Tri.

A Tabela 6.1 mostra o resultado da execução dos 3 modelos de redes neurais DFF. Apesar de o Modelo DFF 3 ter sido o mais rápido (~7 minutos), o Modelo DFF 2, que utiliza o DBSCAN para a geração do bloco, obteve melhores resultados, com 0,00231 de MSE e 0,03854 de MAE. O Modelo DFF 3, que utiliza o K-means, também obteve valores consideravelmente superiores ao Modelo DFF 1, que não utiliza a estratégia das distâncias em relação aos blocos dos átomos. A Figura 6.1 representa o histórico do MSE durante as épocas de treinamento enquanto a Figura 6.2 apresenta o histórico do MAE. Em ambas é possível notar a diferença entre o Modelo DFF 1, que não utiliza os blocos, e os demais.

Tabela 6.1 – Resultado das predições para redes neurais DFF especificando o tempo de execução, MAE e MSE para 1.000 épocas. Em negrito, destaca-se os melhores resultados para cada coluna.

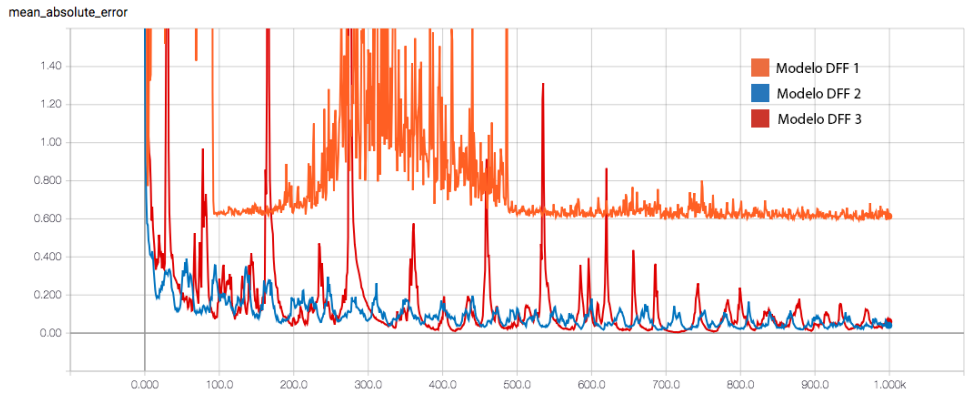
Modelo	Tempo de execução	MAE	MSE
Modelo DFF 1	00:17:55	0,59760	0,53427
Modelo DFF 2	00:09:25	**0,03854**	**0,00231**
Modelo DFF 3	**00:07:47**	0,06526	0,00667

Figura 6.1 – Gráfico do MSE dos modelos de rede DFF durante o processo de treinamento. Observa-se os valores menores dos modelos DFF 2 e 3 em relação ao DFF 1.



Fonte: O autor.

Figura 6.2 – Gráfico do MAE dos modelos de rede DFF durante o processo de treinamento. Observa-se os valores menores dos modelos DFF 2 e 3 em relação ao DFF 1.



Fonte: O autor.

6.3 RESULTADOS COM REDES DEEP CONVOLUTIONAL NEURAL NETWORK

Uma outra abordagem foi testada utilizando redes neurais convolucionais profundas (DCNN - *Deep Convolutional Neural Network*). Nessa abordagem, as posições dos átomos foram reduzidas a um espaço tridimensional com 32 unidades de medida em cada eixo do espaço 3D (32x32x32) devido a limitação de alocação de memória RAM. Essa redução de espaço é dado pela Equação 5.2. Para essa abordagem foram testados 3 modelos especificados a seguir:

- Modelo DCNN 1, abordagem simples onde foi atribuído o valor 1 para as posições onde havia algum átomo naquela conformação e 0 para as demais;
- Modelo DCNN 2, onde foram utilizados 2 canais, um representando ausência ou presença de átomo naquela posição, e outro com a distância em relação ao centro gerado por DBSCAN;

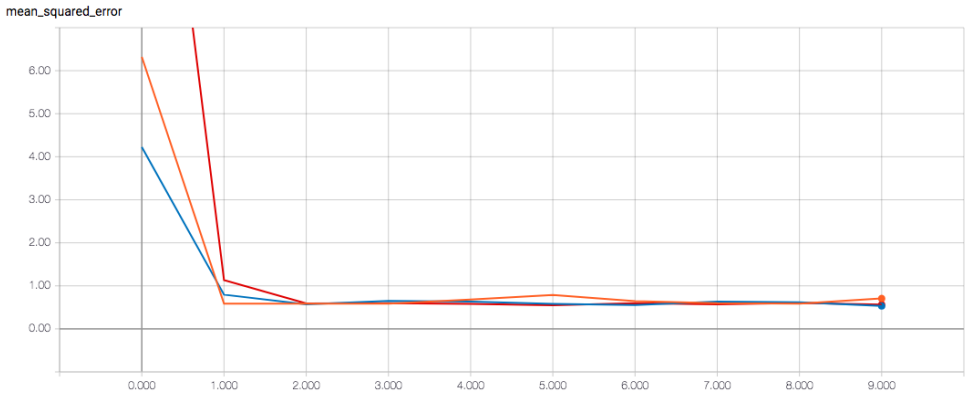
- Modelo DCNN 3, onde foram utilizados 2 canais, um representando ausência ou presença de átomo naquela posição, e outro com a distância em relação ao centro gerado por K-means;

A Tabela 6.2 apresenta os resultados das redes convolucionais, onde, apesar de o Modelo DCNN 1 ter sido o mais rápido (com ~3 horas), o Modelo DCNN 3, que utiliza o K-means apresenta os melhores resultados, com 0,62336 de MAE e 0,57951 de MSE. A Figura 6.3 representa o histórico do MSE durante as épocas de treinamento enquanto a Figura 6.4 apresenta o histórico do MAE. É possível notar que há pouca diferença entre os 3 modelos.

Tabela 6.2 – Resultado das predições para redes neurais DCNN especificando o tempo de execução, MAE e MSE para 10 épocas.

Modelo	Tempo de execução	MAE	MSE
Modelo DCNN 1	**03:00:17**	0,70741	0,76065
Modelo DCNN 2	03:09:52	0,73635	0,80790
Modelo DCNN 3	03:08:36	**0,62336**	**0,57951**

Figura 6.3 – Gráfico do MSE dos modelos de rede DCNN durante o treinamento. Nota-se apenas diferenças sutis entre os modelos.

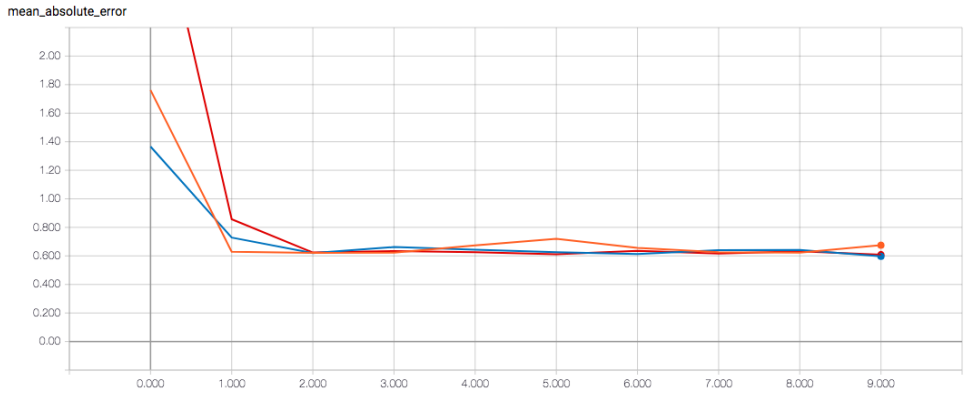


Fonte: O autor.

6.4 COMPARAÇÃO COM O 3D-TRI

O algoritmo 3D-Tri foi testado usando o DBSCAN e o K-means para ser comparado com os resultados deste trabalho. Para comparação com o 3D-Tri o tempo de execução foi desconsiderado, já que por se tratar de algoritmos diferentes, que não utiliza o conceito de épocas, não faz sentido compará-los. A Tabela 6.3 apresenta os resultados do teste. É

Figura 6.4 – Gráfico do MAE dos modelos de rede DCNN. Nota-se apenas diferenças sutis entre os modelos.



Fonte: O autor.

possível notar que os valores de MSE e RMSE são muito próximos com uma leve vantagem do K-means, mas o MAE do DBSCAN apresenta um valor aproximadamente 0,15 menor. A comparação entre todos os modelos mostra a vantagem dos modelos DFF que utilizam a estratégia do uso dos blocos: **Modelo DFF 2** e **Modelo DFF 3**.

Tabela 6.3 – Resultado do algoritmo 3D-Tri especificando o tempo de execução, MAE e MSE. Em negrito, destaca-se os melhores resultados para cada coluna.

Modelo	MAE	MSE	RMSE
3D-Tri com K-means	0,76307	0,49732	0,70521
3D-Tri com DBSCAN	0,61045	0,50611	0,71142
Modelo DFF 1	0,59760	0,53427	0,73093
Modelo DFF 2	**0,03854**	**0,00231**	**0,04806**
Modelo DFF 3	0,06526	0,00667	0,08167
Modelo DCNN 1	0,70741	0,76065	0,87215
Modelo DCNN 2	0,73635	0,80790	0,89883
Modelo DCNN 3	0,62336	0,57951	0,76125

6.5 CONSIDERAÇÕES DO CAPÍTULO

Este capítulo apresentou os testes para os dois tipos de redes neurais profundas avaliados neste trabalho. Para ambos, foram propostas abordagens que utilizam a estratégia proveniente do algoritmo 3D-Tri de usar agrupamento de dados para gerar blocos para cada átomo. Para cada tipo de rede neural foram testados 3 modelos: o primeiro sem a estratégia de gerar os blocos a partir do agrupamento, a segunda utilizando o DBSCAN como agrupamento, e a terceira utilizando o K-means como agrupamento. Além disso, o

algoritmo 3D-Tri também foi testado para ser comparado com as estratégias usadas neste trabalho. O resultado mostrou que a utilização de redes neurais de aprendizagem profunda são promissoras para dados de docagem molecular com dinâmica molecular. E mostra também que a utilização das distâncias em relação ao centro dos blocos gerados para os átomos a partir de um agrupamento de dados, em especial o DBSCAN, traz melhores resultados que a utilização dos dados brutos.

Os resultados obtidos evidenciam que a abordagem de utilizar *deep feedforward* apresentam os melhores resultados globais, isso é, tanto em relação ao tempo de execução quanto aos resultados das funções de custo. Ainda em relação à utilização de *deep feedforward*, nota-se que a abordagem de utilização de agrupamento de dados por DBSCAN apresenta-se mais favorável, com resultados de MAE e RMSE mais satisfatórios.

Ao se comparar o tipo de dados de entrada para cada abordagem, nota-se que a abordagem por DBSCAN apresenta os melhores resultados tanto para os modelos DFF quanto para os modelos obtidos com o 3D-Tri. Para os models DCNN, a abordagem a partir do algoritmo de agrupamento K-Means apresentou os melhores resultados.

7 CONCLUSÃO

Técnicas convencionais de aprendizagem de máquina são limitadas em processar dados em sua forma bruta. Técnicas de aprendizagem profunda, porém, utilizam múltiplas camadas com o intuito de representar o conhecimento em diferentes níveis de abstração. Isso permite aprender conceitos complexos a partir de dados brutos. Por esse motivo, técnicas de *deep learning* são aplicadas em dados considerados complexos como imagens, vídeos e áudio (LECUN; BENGIO; HINTON, 2015; SCHMIDHUBER, 2015).

Este trabalho está inserido no contexto de desenho racional de fármacos, e utilizou dados de docagem molecular sobre um modelo flexível do receptor AcrB, gerado por meio de simulação de dinâmica molecular. Apesar de crescente, a aplicação de técnicas de *deep learning* ainda é pouco explorada em dados de docagem molecular, conforme abordado o Capítulo 4, que trata da revisão sistemática realizada.

Foi apresentada neste trabalho uma abordagem *deep learning* para seleção de conformações de proteínas que considera suas propriedades tridimensionais. A abordagem faz uso da estratégia de geração de blocos para cada átomo proposta por Winck (2012). Os blocos, ao contrário do Algoritmo 3D-Tri, são gerados apenas uma vez, determinando assim um único bloco para cada átomo. Os modelos *Feedforward* utilizaram as distâncias de cada átomo em relação ao centro de seu bloco ao invés de suas coordenadas. Na abordagem convolucional foi adicionado um canal extra, com a distância em relação ao centro do bloco e 0 caso não houvesse átomo na posição. Em ambas arquiteturas foram testados dois algoritmos de agrupamento na geração dos blocos: K-means e DBSCAN. O primeiro, originalmente proposto por Winck (2012) e o segundo proposto posteriormente por Kronbauer (2016). Dentre os algoritmos de agrupamento, a estratégia baseada em densidade do DBSCAN apresentou melhores resultados.

Os testes foram realizados com dados de simulação de docagem molecular sobre a proteína AcrB (*Acriflavine resistance proteine B*) com o ligante NUNL02, conhecido como inibidor da bomba de efluxo AcrB (PRATES, 2014). Os testes mostraram que a utilização dos blocos gerou resultados melhores que a utilização dos dados brutos, em especial para os modelos *Feedforward*. A estratégia convolucional teve resultados significativamente inferiores a *Feedforward*. Cabe considerar que os modelos convolucionais foram testado com uma quantidade menor de épocas, devido o tempo de processamento que as camadas convolucionais demandam.

Por fim, conclui-se que a estratégia de utilizar blocos gerados por agrupamento de dados foi capaz de reduzir significativamente as taxas de erro para os dados testados. O resultado mostra também resultados melhores que os do Algoritmo 3D-Tri.

7.1 TRABALHOS FUTUROS

Este trabalho apresenta uma metodologia promissora para classificar dados de docagem molecular com dinâmica molecular. Porém, essa metodologia pode ser expandida e melhorada em trabalhos futuros. Dentre diferentes oportunidades relevantes para continuidade da pesquisa, pode-se citar:

- Testar a metodologia com diferentes experimentos de docagem molecular;
- Criar uma abordagem que acrescente mais informações da docagem, além das coordenadas dos átomos;
- Utilizar a abordagem aqui proposta para auxiliar em um processo de triagem virtual.

REFERÊNCIAS BIBLIOGRÁFICAS

ABADI, M. et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. **arXiv preprint arXiv:1603.04467**, 2016.

ALPAYDIN, E. **Introduction to machine learning**. [S.l.]: MIT press, 2014.

AMARO, R. E.; BARON, R.; MCCAMMON, J. A. An improved relaxed complex scheme for receptor flexibility in computer-aided drug design. **Journal of computer-aided molecular design**, Springer, v. 22, n. 9, p. 693–705, 2008.

BERMAN, H. et al. The protein data bank nucleic acids research, 28, 235-242. **View Article PubMed/NCBI Google Scholar**, 2000.

BIOLCHINI, J. et al. Systematic review in software engineering. **System Engineering and Computer Science Department COPPE/UFRJ, Technical Report ES**, v. 679, n. 05, p. 45, 2005.

BRUNTON, L. L.; KNOLLMANN, B. C. **As Bases Farmacológicas da Terapêutica de Goodman e Gilman**. 12. ed. [S.l.]: McGraw-Hill, Inc., 2010.

CHENG, T. et al. Structure-based virtual screening for drug discovery: a problem-centric review. **The AAPS journal**, Springer, v. 14, n. 1, p. 133–141, 2012.

CLARKE, M.; OXMAN, A. **Cochrane reviewers' handbook**. [S.l.]: Update Software, 2000.

COOK, C. E. et al. The european bioinformatics institute in 2017: data coordination and integration. **Nucleic Acids Research**, v. 46, n. D1, p. D21–D29, 2018. Disponível em: <http://dx.doi.org/10.1093/nar/gkx1154>.

COSTA, J. Ferreira da et al. Ptml model of chembl data for dopamine targets, docking, synthesis, and assay of new plg peptidomimetics. **ACS chemical neuroscience**, ACS Publications, 2018.

COZZINI, P. et al. Target flexibility: An emerging consideration in drug discovery and design. **Journal of Medicinal Chemistry**, v. 51, n. 20, p. 6237–6255, 2008.

DIMASI, J. A. Innovation in the pharmaceutical industry: New estimates of rd costs. **Tufts Center for the Study of Drug Delelopment**, 2014.

FACELI, K. et al. **Inteligência Artificial: uma abordagem de aprendizado de máquina**. 1. ed. Rio de Janeiro: LTC, 2011. ISBN 8521618808, 9788521618805.

FITRIAWAN, A. et al. Multi-label classification using deep belief networks for virtual screening of multi-target drug. In: IEEE. **Computer, Control, Informatics and its Applications (IC3INA), 2016 International Conference on**. [S.l.], 2016. p. 102–107.

GOODFELLOW, I. et al. **Deep learning**. [S.l.]: MIT press Cambridge, 2016. v. 1.

HALKIDI, M.; BATISTAKIS, Y.; VAZIRGIANNIS, M. On clustering validation techniques. **Journal of intelligent information systems**, Springer, v. 17, n. 2-3, p. 107–145, 2001.

HAMANAKA, M. et al. Cgbvs-dnn: Prediction of compound-protein interactions based on deep learning. **Molecular informatics**, Wiley Online Library, v. 36, n. 1-2, p. 1600045, 2017.

HINTON, G. E. Learning multiple layers of representation. **Trends in cognitive sciences**, Elsevier, v. 11, n. 10, p. 428–434, 2007.

HSIEH, C.-E. et al. Molecular descriptors selection and machine learning approaches in protein-ligand binding affinity with applications to molecular docking. In: IEEE. **2016 International Computer Symposium (ICS)**. [S.l.], 2016. p. 38–43.

JEFFREY, G. A. **IntroductionAn introduction to hydrogen bonding**. [S.l.]: Oxford University, 1997.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **arXiv preprint arXiv:1412.6980**, 2014.

KITCHENHAM, B. Procedures for performing systematic reviews. **Keele, UK, Keele University**, v. 33, n. 2004, p. 1–26, 2004.

KRONBAUER, A. M. M. **IDENTIFICAÇÃO DE REGIÕES DENSAS EM TRAJETÓRIAS ATÔMICAS POR MEIO DE AGRUPAMENTO DE DADOS EM SIMULAÇÃO DE DINÂMICA MOLECULAR**. [S.l.]: Universidade Federal de Santa Maria, 2016.

KRONBAUER, A. M. M.; MACHADO, K. S.; WINCK, A. T. Identicação de regiões densas de trajetórias atômicas em simulações de dinâmica molecular. In: **Symposium on Knowledge Discovery, Mining and Learning - KDMILE**. [S.l.: s.n.], 2016.

KUNTZ, I. D. Structure-based strategies for drug design and discovery. **Science**, American Association for the Advancement of Science, v. 257, n. 5073, p. 1078–1082, 1992. ISSN 00368075, 10959203. Disponível em: <http://www.jstor.org/stable/2879835>.

LAN, K. et al. A survey of data mining and deep learning in bioinformatics. **Journal of Medical Systems**, v. 42, n. 8, p. 139, Jun 2018. ISSN 1573-689X. Disponível em: <https://doi.org/10.1007/s10916-018-1003-9>.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, Nature Research, v. 521, n. 7553, p. 436–444, 2015.

LESK, A. **Introduction to bioinformatics**. [S.l.]: Oxford University Press, 2013.

LIN, J.-H. et al. Computational drug design accommodating receptor flexibility: the relaxed complex scheme. **Journal of the American Chemical Society**, ACS Publications, v. 124, n. 20, p. 5632–5633, 2002.

LUGER, G. F. **Inteligência Artificial**. 6. ed. São Paulo: Pearson, 2013. ISBN 8581435505, 9788581435503.

LUSCOMBE, N. M. et al. What is bioinformatics? a proposed definition and overview of the field. **Methods of information in medicine**, FK SCHATTAUER VERLAGSGESELLSCHAFT MBH, v. 40, n. 4, p. 346–358, 2001.

LYBRAND, T. P. Ligand—protein docking and rational drug design. **Current opinion in structural biology**, Elsevier, v. 5, n. 2, p. 224–228, 1995.

MACHADO, K. S. et al. Automating molecular docking with explicit receptor flexibility using scientific workflows. In: SPRINGER. **Brazilian Symposium On Bioinformatics**. [S.l.], 2007. p. 1–11.

____. Mining flexible-receptor docking experiments to select promising protein receptor snapshots. **BMC genomics**, v. 11 Suppl 5, p. S6, 2010.

____. Mining flexible-receptor molecular docking data. **WIRE Data Mining Knowl Discov**, v. 1, p. 532–541, 2011.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, Springer, v. 5, n. 4, p. 115–133, 1943.

MICHALSKI, R. S.; CARBONELL, J. G.; MITCHELL, T. M. **Machine learning: An artificial intelligence approach**. [S.l.]: Springer Science & Business Media, 2013.

MITCHELL, T. M. **Machine Learning**. 1. ed. New York, NY, USA: McGraw-Hill, Inc., 1997. ISBN 0070428077, 9780070428072.

MURAKAMI, S. Crystal structures of a multidrug transporter reveal a functionally rotating mechanism. **Nature**, v. 443, p. 101–038, 2006.

MURAKAMI, S. et al. Crystal structure of bacterial multidrug efflux transporter acrb. **Nature**, Nature Publishing Group, v. 419, n. 6907, p. 587, 2002.

NORVIG, P.; RUSSELL, S. **Inteligência Artificial: Tradução da 3a Edição**. [S.l.]: Elsevier Brasil, 2014. v. 1.

PEARLMAN, D. A. et al. Amber, a package of computer programs for applying molecular mechanics, normal mode analysis, molecular dynamics and free energy calculations to simulate the structural and energetic properties of molecules. **Computer Physics Communications**, Elsevier, v. 91, n. 1-3, p. 1–41, 1995.

PEREIRA, J. C.; CAFFARENA, E. R.; SANTOS, C. N. dos. Boosting docking-based virtual screening with deep learning. **Journal of chemical information and modeling**, ACS Publications, v. 56, n. 12, p. 2495–2506, 2016.

PRATES, N. S. Simulações por dinâmica molecular e agrupamento de estruturas da proteína da bomba de efluxo acrb. Universidade Federal do Rio Grande, 2014.

PRONK, S. et al. Gromacs 4.5: a high-throughput and highly parallel open source molecular simulation toolkit. **Bioinformatics**, v. 29, n. 7, p. 845–854, 2013. Disponível em: <http://dx.doi.org/10.1093/bioinformatics/btt055>.

QUINLAN, J. R. et al. Learning with continuous classes. In: SINGAPORE. **5th Australian joint conference on artificial intelligence**. [S.l.], 1992. v. 92, p. 343–348.

RIFAIOGLU, A. S. et al. Recent applications of deep learning and machine intelligence on in silico drug discovery: methods, tools and databases. **Briefings in Bioinformatics**, p. bby061, 2018. Disponível em: <http://dx.doi.org/10.1093/bib/bby061>.

RUDER, S. An overview of gradient descent optimization algorithms. **arXiv preprint arXiv:1609.04747**, 2016.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **nature**, Nature Publishing Group, v. 323, n. 6088, p. 533, 1986.

SCHMIDHUBER, J. Deep learning in neural networks: An overview. **Neural networks**, Elsevier, v. 61, p. 85–117, 2015.

SILVA, R. G. N. **Implementação de Árvores de Decisão para Propriedades Tridimensionais em Linguagem Python**. 2015. Monografia (Bacharel em Ciência da Computação), UFSM (Universidade Federal de Santa Maria), Santa Maria, Brazil.

SILVEIRA, C. H. da et al. Protein cutoff scanning: A comparative analysis of cutoff dependent and cutoff free methods for prospecting contacts in proteins. **Proteins**, v. 74, n. 3, p. 727—743, February 2009. ISSN 0887-3585.

SUNSERI, J. et al. Convolutional neural network scoring and minimization in the d3r 2017 community challenge. **Journal of computer-aided molecular design**, Springer, p. 1–16, 2018.

TAN, P.-N. et al. **Introduction to data mining**. [S.l.]: Pearson Education India, 2006.

TROTT, O.; OLSON, A. J. Autodock vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. **Journal of Computational Chemistry**, Wiley Subscription Services, Inc., A Wiley Company, v. 31, n. 2, p. 455–461, 2010. ISSN 1096-987X. Disponível em: <http://dx.doi.org/10.1002/jcc.21334>.

UNTERTHINER, T. et al. Deep learning as an opportunity in virtual screening. In: **Proceedings of the Deep Learning Workshop at NIPS**. [S.l.: s.n.], 2014.

WANG, Y.; WITTEN, I. Inducing model trees for continuous classes. In: **European Conference on Machine Learning**. [S.l.: s.n.], 1997. p. 128–137.

WINCK, A. T. 3d-tri: Um algoritmo de indução de árvore de regressão para propriedades tridimensionais-um estudo sobre dados de docagem molecular considerando a flexibilidade do receptor. Pontifícia Universidade Católica do Rio Grande do Sul, 2012.

WINCK, A. T. et al. Fredd: Supporting mining strategies through a flexible-receptor docking database. In: GUIMARÃES, K. S.; PANCHENKO, A.; PRZYTYCKA, T. M. (Ed.). **Advances in Bioinformatics and Computational Biology**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 143–146.

____. Context-based preprocessing of molecular docking data. **BMC genomics**, v. 14 Suppl 6, p. S6, 2013.

WITTEN, I. H. et al. **Data Mining: Practical machine learning tools and techniques**. [S.l.]: Morgan Kaufmann, 2016. 654 p.

ZHAO, Z.; GONG, X. Protein-protein interaction interface residue pair prediction based on deep learning architecture. **IEEE/ACM Transactions on Computational Biology and Bioinformatics**, IEEE, 2017.

ZHOU, Y.-T.; CHELLAPPA, R. Computation of optical flow using a neural network. In: **IEEE International Conference on Neural Networks**. [S.l.: s.n.], 1988. v. 1998, p. 71–78.