UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

RAFAEL FELIPE TASAKA DE MELO

REDUÇÃO DE DIMENSIONALIDADE EM BASE DE DADOS DE MICROARRANJO USANDO REDE NEURAL AUTOCODIFICADORA

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA

2021

RAFAEL FELIPE TASAKA DE MELO

REDUÇÃO DE DIMENSIONALIDADE EM BASE DE DADOS DE MICROARRANJO USANDO REDE NEURAL AUTOCODIFICADORA

Dimensionality reduction in microarray database using autoencoder neural network

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação, do Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná.

Orientador: Profª Drª Helyane Bronoski Borges

PONTA GROSSA

2021

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Ponta Grossa
Diretoria de Graduação e Educação Profissional
Departamento Acadêmico de Informática
Bacharelado em Ciência da Computação

TERMO DE APROVAÇÃO

REDUÇÃO DE DIMENSIONALIDADE EM BASE DE DADOS DE MICROARRANJO USANDO REDE NEURAL AUTOCODIFICADORA

por

RAFAEL FELIPE TASAKA DE MELO

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 24 de Agosto de 2021 como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof.ª Drª Helyane Bonoski Borges
Orientadora

_____
Prof.ª Drª Simone Nasser Matos
Membro titular

_____
Prof. MSc Geraldo Ranthum
Membro titular

_____
Prof. MSc. Geraldo Ranthum
Responsável pelo Trabalho de Conclusão de Curso

_____
Prof. Dr. André Pinz Borges
Coordenador do curso

Dedico este trabalho a minha família e aos meus amigos pela motivação e apoio nos momentos mais difíceis.

AGRADECIMENTOS

Agradeço, primeiramente, à minha família pelo apoio e paciência para poder concluir este trabalho.

À minha orientadora que me mostrou o caminho que deveria seguir e que fez-me iluminar às ideias nos momentos de escuridão.

A todos os professores, colegas de sala e departamento e todo o corpo docente da UTFPR que possibilitou a estrutura para poder realizar meu trabalho.

Também agradeço a todos que, de alguma forma, contribuíram para a realização deste trabalho.

Você não consegue ligar os pontos olhando pra frente; você só consegue ligá-los olhando pra trás. Então você tem que confiar que os pontos se ligarão algum dia no futuro. Você tem que confiar em algo – seu instinto, destino, vida, carma, o que for. Esta abordagem nunca me desapontou, e fez toda diferença na minha vida.

(JOBS, Steve)

RESUMO

Algoritmos de Aprendizagem de Máquina vem sendo cada vez mais utilizados pela sua capacidade de aprender a partir de grandes volumes de dados como, por exemplo, dados de expressão gênica obtidos pela técnica de microarranjo. Uma característica das bases de dados de microarranjos é que, geralmente, ela é formada por uma grande quantidade de atributos e um pequeno número de amostras. Bases de dados com alta dimensionalidade podem possuir atributos redundantes e muitas vezes irrelevantes, podendo atrapalhar o processo de aprendizagem e o desempenho das predições. Métodos de redução de dimensionalidade são utilizados para reduzir a quantidade de atributos das bases de dados. Redes neurais autocodificadoras podem ser adaptadas e utilizadas para a extração de atributos e, consequentemente, a redução da dimensionalidade. Este trabalho tem como objetivo utilizar uma rede neural autocodificadora combinada a uma rede neural classificadora para realizar uma redução de dimensionalidade. Para isso, serão realizados experimentos em cinco bases de dados. Os resultados foram avaliados por meio da taxa de acerto dos classificadores KNN, SVM, Naive Bayes e Árvore de Decisão. Os resultados mostram que o método criou base de dados menores e que apresentam uma boa representação.

Palavras-chave: Mineração de dados. Redução de dimensionalidade. Microarranjo. Autocodificador.

ABSTRACT

Machine Learning Algorithms have been increasingly used by its ability to learn from large volumes of data, such as gene expression data obtained by the microarray technique. A characteristic of the microarray data is that, generally, it is formed by a large amount of attributes and a few samples. It is known that data with high dimensionality can have redundant and often irrelevant attributes, it can hinder the learning process and the performance of predictions. Methods of dimensionality reduction are used to reduce the amount of attributes of the database. Autoencoder Neural Networks can be adapted and used for attribute extraction and, consequently, dimensionality reduction. This job aims to use an autoencoder neural network combined with a classifying neural network to perform a dimensionality reduction. For this, experiments will be carried out in five databases. The results were evaluated through the hit rate of the classifiers KNN, SVM, Naive Bayes and Decision Tree. The results show that the method created very representative smaller databases.

Keywords: Data mining. Dimensionality reduction. Microarray. Autoencoder.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

AM	Aprendizagem de Máquina
API	*Application Programming Interface*
BR	*Bayesian Regularization*
EPS	*Encapsulated PostScript*
FEA-PTC	*Feature Extraction based on Audoencoders using Pre Training with Classification*
GMM	*Gaussian Mixture Model*
IA	Inteligência Artificial
K-NN	*K-Nearest Neighbor*
LDA	*Linear Discriminant Analysis*
MLP	*Multi-Layer Perceptron*
MSE	*Mean Squared Error*
NA	*Neighborhood Analysis*
NPV	*Negative predictive value*
PCA	*Principal Component Analysis*
PPV	*Positive predictive value*
PS	*PostScript*
RNA	Redes Neurais Artificiais
SAM	*Significance analysis of microarray*
SIMLR	*Single-cell Interpretation via Multi-kernel LeaRning*
SVM	*Support Vector Machine*

SUMÁRIO

1 INTRODUÇÃO

No campo da Aprendizagem de Máquina (AM) diversos algoritmos utilizam de bases de dados que possuem milhares de atributos para o treinamento (MENG *et al.*, 2017). Utilizar essa quantidade de atributos faz com que, além de deixar o processo de treinamento lento, o algoritmo tenha dificuldades em encontrar padrões nos dados, dificultando uma possível classificação entre eles.

Algumas dessas bases possuem, também, a característica de muitos atributos e poucos exemplos. Muitas vezes, isso se deve pela dificuldade em coletar amostras em largas quantidades. Esse problema é conhecido como maldição da dimensionalidade (GÉRON, 2017). Devida essa característica muitos dos métodos de AM não conseguem criar um modelo de classificação eficiente o suficiente para prever exemplos futuros por conta da dificuldade em generalizar tamanha quantidade de atributos.

Visando diminuir o problema da maldição da dimensionalidade e outros problemas gerados pela alta dimensionalidade, técnicas de redução de dimensionalidade podem ser aplicadas para retirar da base de dados atributos irrelevantes e/ou redundantes (GÉRON, 2017). Além disso, representar uma grande quantidade de atributos em um número reduzido de dados pode ajudar a biólogos a identificarem quais genes estão diretamente ligados aos problemas neles identificados.

As técnicas de redução de dimensionalidade podem ser divididas em duas grandes abordagens: extração de atributos e seleção de atributos (CASTRO; FERRARI, 2016). Métodos de seleção de atributos selecionam os atributos mais relevantes sem alterar a base de dados, enquanto os métodos de extração de atributos modificam a base de dados para representar os dados, como por exemplo, o método *Principal Component Analysis* (PCA) e o *Linear Discriminant Analysis* (LDA).

Contudo a projeção feita por esses métodos não leva em consideração as relações entre as bases originais e as bases reduzidas, fazendo com que uma futura representação não represente bem seus dados originais (MENG *et al.*, 2017).

Como alternativa para a tarefa de redução, pode-se utilizar uma rede autocodificadora que pode realizar a extração de atributos em suas camadas ocultas. Utilizar redes autocodificadoras por si só pode apresentar problemas, uma vez em que ela foca em reduzir dados e, depois, reconstrui-los, criando-se assim uma rede preparada para redução e reconstrução dos dados mas

desconsiderando as classes pertencentes aos dados.

Tendo em vista a preocupação de se observar as classes o método proposto neste trabalho busca reduzir a dimensionalidade da base a partir da extração de atributos onde, dado um conjunto de atributos *X*, busca-se a criação de um novo conjunto de atributos *Y*, que são mais expressivos e melhor representem a variedade dos atributos originais. A estrutura responsável por essa redução se chama codificador. O codificador é uma estrutura interna da autocodificadora que consiste nas camadas internas entre a camada de entrada, composta pelo número total de atributos a serem reduzidos, e a menor camada da rede (também chamada de *bottleneck*), composta pelo número desejado de redução.

Devida a importância da estrutura codificadora para a redução da base, gerando uma base de dados reduzida, o método proposto visa aprimorar a rede neural autocodificadora realizando um pré treinamento no codificador utilizando uma *Multi-Layer Perceptron* (MLP) para classificação e, com os pesos utilizados nesse treinamento, é realizado um novo treinamento com o decodificador, finalizando assim o treino completo. Esse pré treinamento faz com que o codificador crie uma relação entre base a original e sua classe bem como com o decodificador, para que esse também esteja otimizado para uma possível reconstrução dos atributos.

1.1 DESCRIÇÃO DO PROBLEMA E MOTIVAÇÃO

Bases de microarranjo vem sendo amplamente estudadas no campo da biomedicina (ADIWIJAYA *et al.*, 2018). Essas bases vêm se mostrando importantes para um melhor entendimento tanto da caracterização genética de uma célula quanto para identificar sequenciamentos problemáticos que possam gerar, entre outras doenças, diversos tipos de câncer (ADIWIJAYA *et al.*, 2018). As bases, contudo, possuem a característica de possuírem uma quantidade de atributos exponencialmente maior do que o número de exemplares. Esse problema é conhecido como "maldição da dimensionalidade"(GÉRON, 2017). Essa alta dimensão dificulta análises posteriores de como os genes interagem, suas características e quais são os mais relevantes para geração de doenças no organismo.

No momento em que a análise se dados médicos se tornou possível e relevante, métodos de Inteligência Artificial (IA) se mostraram igualmente relevantes, logo utilizar um método de IA para análise desses dados é uma boa escolha - impedindo que processamentos lentos e redundantes aconteçam.

Um dos métodos de redução aplicado nessas bases são as redes neurais autocodifica-

doras. Essas redes possibilitam a redução das bases - podendo assim entender quais atributos são mais relevantes e, depois, reconstruí-las. Essa arquitetura permite que redução seja mais representativa e útil para a construção de classificadores posteriores.

A redução de dimensionalidade facilita a classificação, visualização e armazenamento de grandes bases de dados. A proposta de um novo método não só visa realizar as vantagens de um algoritmo de redução de dimensionalidade como também procura otimizar algoritmos já existentes utilizando as estruturas conhecidas combinadas com outras estruturas e modificações voltadas para a base de experimentos.

1.2 OBJETIVOS

Esta Seção apresenta o objetivo geral e os específicos deste trabalho.

1.2.1 Objetivo geral

Desenvolver um método de extração de atributos utilizando a combinação de uma MLP classificadora com uma rede neural autocodificadora.

1.2.2 Objetivos específicos

Como objetivos específicos deste trabalho têm-se:

- Realizar um mapeamento da literatura sobre redes autocodificadoras, bem como métodos de redução de dimensionalidade já existentes;
- Implementar o método proposto utilizando bibliotecas e ferramentas disponíveis na linguagem de programação Python;
- Realizar experimentos em base de dados de microarranjo, observando a acurácia e os erros e cada base de dados de teste;
- Fazer uma análise comparativa dos resultados obtidos.

1.3 ESTRUTURA DO TRABALHO

Este trabalho está organizado em 6 capítulos. O Capítulo 1 apresenta a introdução e contextualização ao tema do trabalho, também apresenta os objetivos gerais e específicos do trabalho. O Capítulo 2 aborda conceitos de redução de dimensionalidade, bem como os principais métodos de redução. O Capítulo 3 apresenta e aplica a metodologia de revisão sistemática da literatura. O Capítulo 4 apresenta a proposta de um método de redução de dimensionalidade utilizando a combinação de uma MLP classificadora com uma rede neural autocodificadora, o Capítulo também exemplifica o funcionamento do método utilizando uma base de dados fictícia. O Capítulo 5 explica a metodologia para a realização dos experimentos (bem como as ferramentas utilizadas, os métodos de avaliação e as bases de dados utilizadas) e apresenta os resultados após os experimentos. Por fim, o Capítulo 6 conclui o trabalho e sugere possíveis trabalhos futuros.

2 REDUÇÃO DE DIMENSIONALIDADE

Este Capítulo apresenta conceitos de redução de dimensionalidade. A Seção 2.1 apresenta a motivação e as principais técnicas de redução. A Seção 2.2 faz uma explicação sobre a seleção de atributos e suas principais abordagens. A Seção 2.3 mostra como funciona a extração de atributos e suas principais frentes. A Seção 2.4 apresenta um breve conceito de redes neurais e, por fim, a Seção 2.5 apresenta o que é uma rede neural autocodificadora e, por fim, a Seção 2.6 relata as considerções finais do Capítulo.

2.1 CONCEITOS DE REDUÇÃO DE DIMENSIONALIDADE

A tecnologia vem avançando e possibilitou com que uma grande massa de dados pudesse ser coletada de maneira mais rápida, contudo isso gerou um problema de superabundância dos dados, fazendo com que a capacidade de analisar os dados seja menor que a capacidade de armazená-los (CASTRO; FERRARI, 2016). Logo viu-se a necessidade de técnicas que buscassem retirar informações relevantes desses dados.

Segundo Maaten *et al.* (2009), a redução de dimensionalidade é a transformação de dados de grande dimensão em uma representação mais significativa dentro de uma dimensão reduzida. Idealmente, essa dimensão reduzida deve conter o mínimo de parâmetros necessários para caracterizar as propriedades observadas nos dados originais.

As técnicas de redução de dimensionalidade podem ser divididas em dois grandes grupos: seleção de atributos e extração de atributos (CASTRO; FERRARI, 2016).

- Seleção de atributos: visa reduzir a base de dados sem alterar os dados originais selecionando os atributos mais relevantes;
- Extração de atributos: visa reduzir a base de dados alterando os dados originais, criando atributos novos que representem os dados originais.

Utilizar esses métodos faz com que a nova base de dados possua somente dados relevantes, facilitando possíveis processamentos em IA, analisar informações relevantes em imagens, dentre outros.

2.2 SELEÇÃO DE ATRIBUTOS

A seleção de atributos consiste em selecionar um subconjunto das variáveis de entrada enquanto reduz efeitos da base de ruído ou variáveis irrelevantes (aquelas que, ao comparar com as demais variáveis dos dados, não caracterizam de maneira significativa um exemplar da base de dados), podendo assim fazer predições melhores (ZEBARI *et al.*, 2020).

Segundo Venkatesh e Anuradha (2019) existem vários processos na seleção de atributos, sendo estes:

1. Direção de busca: É o primeiro estágio do método, onde é determinado o ponto inicial e para que direção se dará a busca. As direções são classificadas em busca retilínea, busca retrógrada e busca randômica.

2. Determinar a estrategia de busca: as estratégias de busca podem ser randomizadas, sequenciais ou exponenciais. Uma boa estratégia de busca pode obter o ótimo local, um custo computacional eficiente e uma boa habilidade de busca local.

3. Critério de avaliação: nesta etapa os melhores atributos são selecionados. Os critérios são classificados como: métodos de filtro, métodos *wrapper*, métodos *embedded* e métodos híbridos.

4. Critério de parada: estágio onde é determinado quando o processo de seleção de atributos deve parar. Os critérios podem ser, por exemplo, um número de iterações pré-definido ou um número de atributos finais pré-definidos.

5. Validação: Última etapa onde é validado os resultados do processo.

Algoritmos de seleção de atributos utilizam, de maneira geral, diferentes abordagens para o critério de avaliação, sendo esses a abordagem filtro, *embedded* e *wrapper*.

2.2.1 Abordagem filtro, *Wrapper* e *Embedding*

A abordagem filtro utiliza de métodos de *ranking* para as variáveis onde variáveis menores do que dado limite são eliminadas (ZEBARI *et al.*, 2020). Uma das vantagens dessa abordagem são o baixo custo computacional e a prevenção de sobre ajustes nos dados. Como desvantagem a seleção dos atributos pode não ser a mais otimizada e esta pode conter redundâncias.

Diferentemente da abordagem de filtro, a abordagem *wrapper* utiliza de algoritmos de indução para determinar quais atributos selecionar e criar uma acurácia a partir do novo *subset*. Dependendo da acurácia obtida, o método pode adicionar ou remover atributos desse subconjunto (ZEBARI *et al.*, 2020).

Algoritmos que utilizam a abordagem *wrapper* são ótimos para obter boas classificações, contudo possuem uma alta complexidade computacional - tornando-os menos eficientes computacionalmente.

A abordagem *embedding* funciona de uma maneira onde os melhores atributos são selecionados durante o processo de aprendizagem do algoritmo, ou seja, os atributos selecionados para um subconjunto e o modelo de aprendizagem são realizados em simultâneo (ZEBARI *et al.*, 2020).

Por conta dessa propriedade, algoritmos com a abordagem *embedding* possuem um custo computacional mais baixo se comparado à abordagem *wrapper*, além disso, esse método também evita que um novo treinamento seja realizado para cada vez que uma nova combinação de atributos é explorada.

2.3 EXTRAÇÃO DE ATRIBUTOS

Enquanto a seleção de atributos separa os atributos mais significativos de uma base de dados, a extração de atributos visa construir um novo subconjunto com atributos transformados, ou seja, ele transforma os atributos originais em outros atributos mais significativos e em menor quantidade. Essa técnica é utilizada para reduzir a complexidade dos atributos e também criar uma representação mais simples dos dados (ZEBARI *et al.*, 2020). A extração de atributos pode ser dividida em duas grandes categorias: métodos lineares e não lineares.

De maneira formal a extração de atributos transforma uma entrada X com n atributos, sendo $X = \{x_1, x_2, ..., x_n\}$ em um subconjunto Z com m atributos, onde $Z = \{z_1, z_2, ..., z_m\}$ e $m < n$

Os métodos lineares são utilizados para resolução de problemas lineares, não modificando assim a estrutura espacial dos dados. Um dos principais métodos lineares é o PCA que realiza a extração de atributos preservando a linearidade dos dados, porém a linearidade também se apresenta como um fator limitante ao método (SIQUEIRA, 2019).

Os problemas não lineares exigem com que os métodos de extração de atributos apliquem transformações não lineares, por conta dessa necessidade esses métodos podem ser mais

custosos computacionalmente, porém mais robustos. Utilizar esses métodos, segundo Siqueira (2019), possibilita um melhor controle de *overfitting* (quando o método possui alta taxa de acerto em uma base de treino, porém essa taxa é baixa na base de teste) e possui um alto poder de discriminação dos atributos extraídos.

2.4 REDES NEURAIS

As Redes Neurais Artificiais (RNA) foram introduzidas por McCulloch e Pitts (1943). No artigo, os autores propõem uma estrutura matemática baseada na lógica proposital, com a inspiração no funcionamento de neurônios de animais, no entanto os estudos das RNAs não se desenvolveram, voltando a ter seu grande "estouro"nos anos 1990 (GÉRON, 2017). Alguns dos fatores que impulsionaram a volta dos estudos dessa estruturam foram:

- A quantidade de dados disponível aumentou;
- Algumas limitações teóricas (como o ótimo local, por exemplo) foram derrubadas na prática;
- O poder computacional aumentou e;
- Os algoritmos de treino foram melhorados.

Desde então RNAs são largamente utilizadas em diversas áreas: biomedicina, processamento de imagem, *marketing*, vendas, entre outras.

Para entender melhor o funcionamento de uma RNA é necessário, primeiramente, entender o modelo proposto por McCulloch e Pitts (1943).

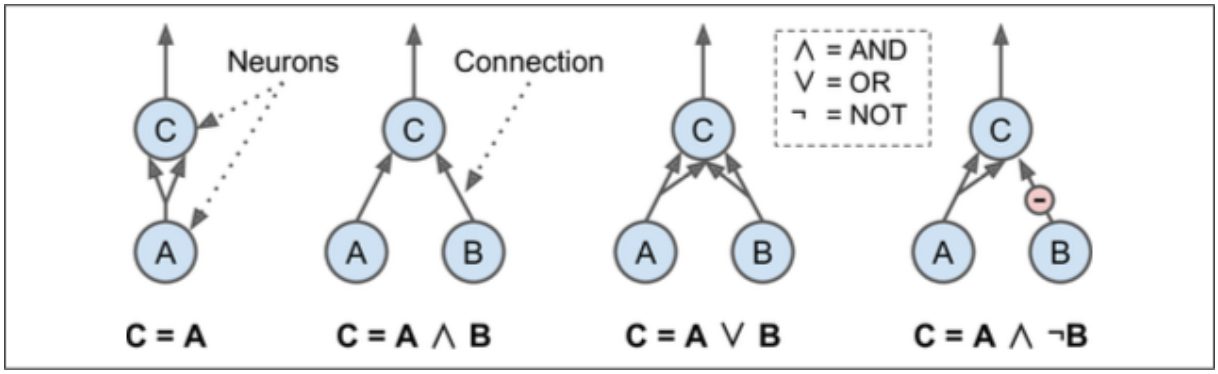
O modelo tinha como objetivo realizar cálculos lógicos (operadores E e OU), logo a saída deste modelo era definida se as entradas estavam "ligadas"(valor 1) ou "desligadas"(valor 0), conforme exemplificado na Tabela 1

Tabela 1 – Tabela verdade do operador lógico "E".

Valor da primeira entrada	Valor da segunda entrada	Saída do modelo
0	0	0
0	1	0
1	0	0
1	1	1

Fonte: Autoria própria.

Figura 1 – Exemplo de diversas aplicações do modelo proposto por McCulloch e Pitts (1943)



Fonte: Géron (2017)

2.4.1 O *perceptron*

O *perceptron* é o modelo de RNA mais simples que existe. Fazendo uma analogia ao sistema nervoso animal, o *perceptron* é comparável a somente um neurônio do sistema nervoso.

Dado um vetor $\bar{X}$ de n entradas, definido pela Equação 1

$$\bar{X} = \{x_1, x_2, ...,x_n\} \tag{1}$$

e um vetor de pesos $\bar{W}$, definido pela Equação 2

$$\bar{W} = \{w_1, w_2, ...,w_n\} \tag{2}$$

a saída resultante y do *perceptron* é dada Equação 3

$$y = f(\sum_{i=1}^{n} x_i w_i) = f(\bar{X} \cdot \bar{W}) \tag{3}$$

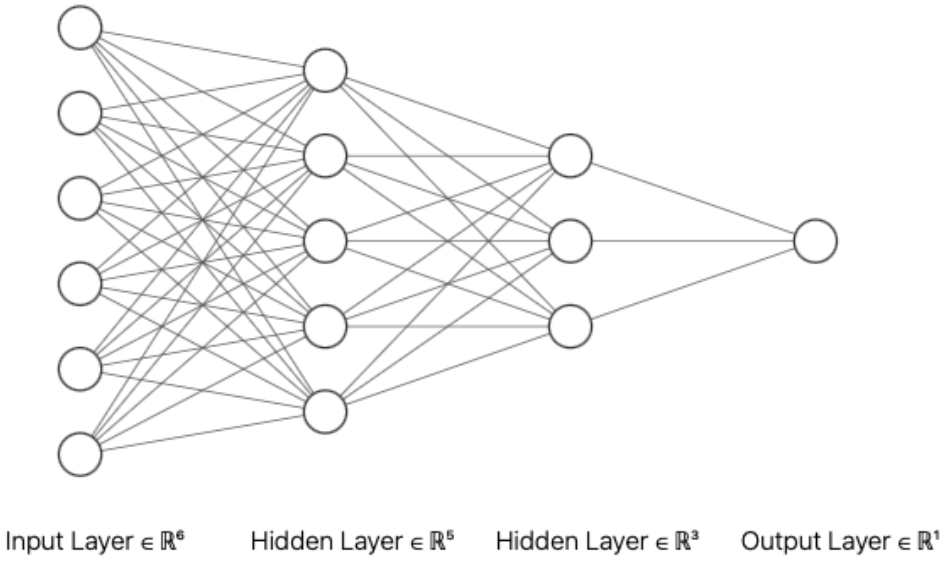
Os *perceptrons* são ótimas estruturas para resoluções de problemas linearmente separáveis, ou seja, é possível solucionar esses problemas com uma separação linear de uma reta em um hiperplano, contudo quanto se trata da resolução de problemas não-lineares *perceprons* não são capazes de obter uma resolução. Retomando aos exemplos de portas lógicas um *perceptron* não consegue resolver, por exemplo, a separação de uma porta lógica *XOR* (Ou exclusivo), demandando assim de uma construção mais complexa.

Para resolver problemas não-lineares buscou-se utilizar diversos *perceptrons* juntos. Essa junção de neurônios é chamada de MLP.

Uma MLP é composta por uma camada de entrada, uma ou mais camadas internas (chamadas camadas ocultas) e uma camada de saída (GÉRON, 2017). Quando esta rede possuir duas ou mais camadas ocultas, ela também tem o nome de Rede Neural Profunda.

Assim como no *perceptron* cada neurônio possui uma entrada, uma função de ativação (com exceção da camada de entrada, que transmite os dados sem alterações) e uma saída. Cada neurônio recebe a saída de todos os outros de sua camada antecessora. A Figura 2 ilustra uma MLP.

Figura 2 – Exemplo de uma MLP



Fonte: Autoria própria

Por conta da complexidade que a rede pode possuir torna-se igualmente complexo observar neurônios de maneira individual. Para tal um algoritmo é utilizado para treinar a rede, esse algoritmo é chamado *backpropagation*.

O *backpropagation* é idêntico ao cálculo de Gradiente Estocástico, assim o cálculo do novo peso é definido pela Equação 4

$$W' = W - \eta * \frac{\sigma E}{\sigma W} \tag{4}$$

onde W' é o novo peso calculado, W o peso atual, η uma taxa de aprendizagem pré definida, σE a derivada do erro e σW a derivada do peso atual.

2.5 REDES NEURAIS AUTOCODIFICADORAS

Uma Rede Neural Autocodificadora é uma espécie de IA não-supervisionada que possui um design onde é possível transformar a entrada da rede em uma representação compactada e significativa através de um codificador e, depois, utilizar de um decodificador para reconstruir a

entrada da maneira mais fiel possível ao original (PINAYA *et al.*, 2020). Em outras palavras a Rede Neural Autocodificadora é uma estrutura composta por duas funções: $f(x)$ e $g(h)$ (para fins explicativos) onde a função $f(x)$ é o codificador da rede definido pela Equação 5:

$$f(x) = h = Wx + b_x \tag{5}$$

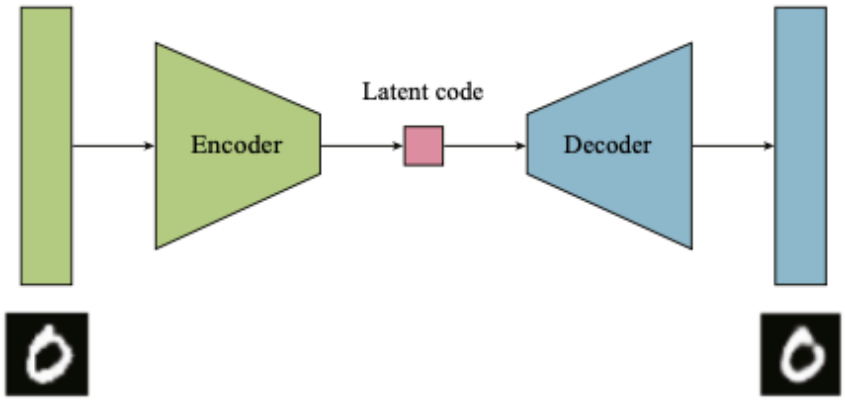
onde W é a matriz de pesos, x o mapeamento da entrada e b o bias. Com o resultado h a função $g(h)$ a transforma em uma réplica de x (aqui denominada de y), ou seja, essa função decodifica h, definida pela Equação 6:

$$g(h) = y = Wh + b_h \tag{6}$$

onde W é a matriz de pesos, h a base compactada e b o bias.

Uma representação da rede autocodificadora é mostrada na Figura 3

Figura 3 – Modelo básico de uma rede neural autocodificadora



Fonte: Pinaya *et al.* (2020)

Por conta de sua característica de compactação, redes neurais autocodificadoras são boas escolhas para reduzir grandes bases de dados e depois reconstruí-las caso necessário, por conta disso essa categoria de rede neural se encaixa no objetivo de reduzir grandes dimensionalidades de bases.

Redes neurais autocodificadoras não conseguem predizer classes ou realizar classificações, são utilizadas para que sua saída replique a sua entrada. Para isso ocorrer uma rede neural autocodificadora deve conter:

- O mesmo número de neurônios na camada de entrada e de saída;
- A rede deve ser simétrica;

- Deve possuir duas estruturas: a estrutura codificadora e a estrutura decodificadora e;
- Possuir camada(s) interior(es) menor(es) que a dimensão da camada de entrada/saída. A camada com o menor número de neurônios da rede é chamada *bottleneck* (ou "gargalo").

Assim como uma MLP redes neurais autocodificadoras possuem um treino baseado em *backpropagation* e cálculo de erros utilizando, por exemplo, MSE.

2.6 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Este Capítulo apresentou os conceitos básicos de redução de dimensionalidade e suas principais técnicas: seleção de atributos e extração de atributos. Foi apresentado, também, as principais abordagens de seleção de atributos (filtro, *wrapper* e *embedding*) e a principal ideia da extração de atributos. Também apresentou-se um breve conceito de redes neurais e o funcionamento de uma rede neural autocodificadora.

A área de estudo da redução de dimensionalidade e de redes neurais é vasto, possuindo diversos métodos e técnicas, por isso este Capítulo somente englobou o necessário para o melhor entendimento deste trabalho.

3 MAPEAMENTO SISTEMÁTICO DA LITERATURA

Este Capítulo apresenta um mapeamento sistemático do tema visando estabelecer o atual estado da arte sobre redução de dimensionalidade de bases de dados de microarranjos. A Seção 3.1 trata sobre o método de pesquisa e os passos a serem seguidos, a Seção 3.2 mostra a aplicação do mapeamento. A Seção 3.3 mostra os resultados obtidos após a aplicação do mapeamento e, por fim, a Seção 3.4 apresenta as considerações finais do Capítulo.

3.1 DESCRIÇÃO DO MÉTODO DO MAPEAMENTO SISTEMÁTICO

Para encontrar resultados mais precisos e uma busca mais focada, um método de pesquisa foi utilizado com objetivo de obter um trabalho de melhor qualidade. O método de mapeamento escolhido é apresentado por Kitchenham e Chartes (2007). Para este trabalho, o método proposto foi adaptado e apresenta as seguintes etapas:

1. Questão de pesquisa: elaboração de questionamentos que buscam respostas relacionadas ao tema do trabalho;
2. Seleção das bases de pequisa: selecionar as bases que contenham trabalhos relevantes;
3. Definição dos termos de busca: elaboração de palavras-chave que auxiliem na busca de trabalhos relacionados;
4. Procedimento de filtragem: aplicação de filtros que eliminem trabalhos não relacionados ao tema e;
5. Resultados: demonstração dos trabalhos filtrados e respostas às questões de pesquisa.

3.2 APLICAÇÃO DO MÉTODO

Esta seção apresenta o método escolhido para a realização do mapeamento sistemático, além da realização das etapas antes descritas.

3.2.1 Questão de pesquisa

O mapeamento sistemático busca identificar características dos métodos que buscam seja um aprimoramento de métodos já existentes ou por métodos novos, para isso é necessário elaborar perguntas que visam atingir este objetivo, visto que estas ajudam a levantar as principais características dos trabalhos encontrados pela busca. O Quadro 1 apresenta essas perguntas.

Quadro 1 – Questões de pesquisa.

ID	Pergunta
Q1	Houve utilização de redes autocodificadoras? Se não, qual abordagem de redução de dimensionalidade foi utilizada?
Q2	Que tipo de bases foram utilizadas? Estas eram de microarranjo?
Q3	O método utilizado pelo autor foi novo ou uma aprimoração de um já existente?
Q4	Quais as medidas de avaliação e/ou classificadores foram utilizadas?

Fonte: Autoria prórpia.

3.2.2 Seleção das bases de pesquisa

Para realizar a busca o primeiro passo é definir em quais bases de dados serão retirados esses artigos. Assim as bases selecionadas devem atender os seguintes requisitos: seleção de publicações na área de Ciência da Computação, a possibilidade de utilizar palavras-chave como filtro e ser na língua inglesa.

As bases utilizadas neste processo foram selecionadas no sistema CAPES, e estes são apresentados no Quadro 2

Quadro 2 – Base de pesquisa.

Base de pesquisa	Site
ArXiv.org	<https://arxiv.ogr/>
IEEE	<https://ieeexplore.ieee.org>
InderScience	<https://www.inderscienceonline.com>
Science Direct	<https://www.sciencedirect.com>
Springer	<https://link.springer.com/>

Fonte: Autoria prórpia.

3.2.3 Definição dos termos de busca

Para retornar resultados relacionados ao tema a ser explorado, foram definidos termos de busca para que as pesquisas busquem resultados mais precisos e estejam mais próximos possíveis ao tema proposto. Os termos definidos são apresentados no Quadro 3

Quadro 3 – Termos de busca.

ID	Termo na língua inglesa	Termo na língua portuguesa
1	*Dimensionality reduction*	Redução de dimensionalidade
2	*Autoencoder*	Autocodificadora
3	*Deep Learning*	Aprendizagem profunda
4	*Microarray*	Microarranjo
5	*Feature Extraction*	Extração de atributos

Fonte: Autoria prórpia.

Com os termos de busca utilizou-se de combinações dos mesmos para definir *strings* de busca utilizando o operador lógico AND, apresentados no Quadro 4

Quadro 4 – Termos de busca.

ID	String de busca
S1	*"Microarray" AND "Autoencoder"*
S2	*"Microarray" AND "Dimensionality Reduction"*
S3	*"Microarray" AND "Deep Learning"*
S4	*"Microarray" AND "Feature Extraction"*

Fonte: Autoria prórpia.

3.2.4 Realização das buscas

Os resultados da busca utilizando as *strings* definidas no Quadro 3 se apresentam na Tabela 2

3.2.5 Procedimento de filtragem

O processo de filtragem dos artigos encontrados nas bases seguem os seguintes critérios:

Tabela 2 – Aplicação das etapas 1, 2 e 3 da filtragem

Base de pesquisa	S1	S2	S3	S4	Busca Inicial
ArXiv.org	0	5	13	39	57
IEEE	7	40	85	404	536
InderScience	0	4	0	3	7
Science Direct	142	62	251	116	571
Springer	13	343	303	1982	2641

Fonte: Autoria própria.

1. Exclusão de duplicatas pela junção dos resultados;
2. Eliminação de todas as publicações de conferências, livros ou capítulos de livros;
3. Eliminação de quaisquer trabalhos antecessores à 2018 e;
4. Exclusão de trabalhos não relacionados ao tema deste trabalho considerando título, resumo e palavras-chave;

Para a realização dos filtros 1, 2 e 3 foi criado um repositório na ferramenta *Zotero* onde, também, houve uma junção dos resultados e os filtros devidamente aplicados. A Tabela 3 mostra o resultado após aplicação desses filtros por base de dados.

Tabela 3 – Aplicação das etapas 1, 2 e 3 da filtragem

Base de dados	Busca Inicial	Filtragem 1, 2 e 3
ArXiv	57	2
Inder Science	*7*	0
IEE	*536*	28
Science Direct	*571*	424
Springer	2641	363

Fonte: Autoria própria.

No filtro 4 foi feita a leitura do título, resumo e palavras-chave. A Tabela 4 mostra a quantidade de artigos selecionados após a realização desse filtro.

Tabela 4 – Aplicação das etapa 4 da filtragem

Base de dados	Filtragem 1, 2 e 3	Filtragem 4
ArXiv	2	0
Inder Science	0	0
IEE	28	0
Science Direct	424	5
Springer	363	3
Total	**815**	**8**

Fonte: Autoria própria.

Após essa filtragem os artigos selecionados são mostrados no Quadro 5, sendo este o resultado após todos os filtros aplicados.

Quadro 5 – Artigos finais após todo o processo de filtragem.

ID Publicação	Autor	Ano	Título
1	Patil, S.; Naik, G.; Pai, R.; Gad, R.	2018	*Stacked Autoencoder for classification of glioma grade III and grade IV*
2	Adem, K.; Kiliçarslan, S.; Comert, O.	2019	*Classification and diagnosis of cervical cancer with stacked autoencoder and softmax classification*
3	Kinalis, S.; Nielsen, F.; Winther, O.; Bagger, F. O.	2019	*Deconvolution of autoencoders to learn biological regulatory modules from single cell mRNA sequencing data*
4	Geddes, T. A.; Kim, T.; Nan, L.; Burchfield, J.; Yang, J.; Tao, D.; Yang, P.	2019	*Autoencoder-based cluster ensembles for single-cell RNA-seq data analysis*
5	Sheet, S.; Ghosh, A.; Gosh, R.; Chakrabarti, A.	2020	*Identification of Cancer Mediating Biomarkers using Stacked Denoising Autoencoder Model - An Application on Human Lung Data*
6	Adem, K.	2020	*Diagnosis of breast cancer with Stacked autoencoder and Subspace kNN*
7	Wang, J.; Xie, X.; Shi, J.; He, W.; Chen, Q.; Chen, L.; Gu, W.; Zhou, T.	2020	*Denoising Autoencoder, A Deep Learning Algorithm, Aids the Identification of A Novel Molecular Signature of Lung Adenocarcinoma*
8	Tangherloni, A.; Ricciuti, F.; Besozzi, D.; Liò, P.; Cveji, A.	2021	*Analysis of single-cell RNA sequencing data based on autoencoders*

Fonte: Autoria prórpia.

3.3 RESULTADOS

Nesta etapa do mapeamento sistemático são apresentadas as respostas obtidas para as perguntas anteriormente definidas, para isso as perguntas são respondidas de acordo com seus identificadores (Q1 a Q4).

Q1 *- Houve utilização de redes neurais autocodificadoras? Se não, qual abordagem de redução de dimensionalidade foi utilizada?*

Em Patil *et al.* (2018) os autores sugerem um método onde utilizam a técnica de limiar combinada com o método *Ratio* (método híbrido) para selecionar os atributos e, em seguida, utilizar, dentre os algoritmos propostos, para classificação uma rede neural autocodificadora. Uma vez com os dados reduzidos, os autores utilizam da rede neural para criar uma representação da base de dados de entrada e a classifica.

Adem *et al.* (2019) utiliza uma rede neural autocodificadora para classificar (utilizando função *softmax*) câncer de colo de útero, ou seja, os autores utilizam um método já existente para estudar sua eficiência nos casos desse câncer.

Kinalis *et al.* (2019) utiliza uma rede neural autocodificadora para extrair o perfil de um único sequenciamento genético, com o objetivo de criar possíveis perfis problemáticos desse

sequenciamento.

Geddes *et al.* (2019) criam um *framework* para a redução de um sequenciamento genético. Primeiro é realizada uma projeção aleatória das bases de dados em sub-espaços para criar diversidade, logo após cada sub-espaço é inserido em uma rede neural autocodificadora para gerar uma base de dados reduzida e esta sofre uma clusterização.

Sheet *et al.* (2020) propõe um método que combina MLP com rede neurais autocodificadoras com ruído (MLP-SDAE) com o objetivo de construir um modelo preditivo. A MLP fará o papel de aprender as entradas e as saídas do modelo, enquanto a rede neural autocodificadora fará o papel de realizar a redução e a reconstrução da base de dados.

Adem (2020) propõe um modelo de rede neural autocodificadora que reduz e classifica base de dados de microarranjo, contudo, diferentemente de uma rede convencional, o autor testa diversos classificadores para o treino quando a rede é aplicada pela primeira vez, tornando assim o sucesso da rede maior - independente da base de dados aplicada.

Wang *et al.* (2020) visa estudar a aplicação de redes neurais autocodificadoras com ruído para a redução de base de dados de câncer de pulmão. O estudo também procura entender como a rede se comporta na redução dessas bases.

Tangherloni *et al.* (2021) visa analisar o desempenho de diversos tipos de redes neurais autocodificadoras para clusterização de diversas bases de sequenciamento genético, para isso uma ferramenta foi desenvolvida: a *scAEspy*. Também os autores também desenvolveram duas redes neurais autocodificadoras que exploram várias categorias de dristribuições Gausianas.

Q2 *- Que tipo de bases de dados foram utilizadas? Estas eram de microarranjo?*

As bases de dados utilizadas por cada autor são apresentadas no Quadro 6

Quadro 6 – Base de dados utilizadas pelos autores.

Autor	Base de dados
Patil, S. *et al* (2018)	Glioma de grau III e IV
Adem *et al* (2019)	Cancêr de colo de útero
Kinalis *et al* (2019)	Sequenciamento de RNA
Geddes *et al* (2019)	Sequenciamento de RNA
Sheet *et al* (2020)	Câncer de pulmão
Adem (2020)	Câncer de mama
Wang *et al*. (2020)	Câncer de pulmão

Fonte: Autoria prórpia.

É possível observar que todos os autores utilizam base de dados de microarranjo em

seus testes, apesar destes serem de tipos de sequenciamentos diferentes.

Q3 *- O método utilizado pelo autor foi novo ou uma aprimoração de um já existente?*

Os artigos encontrados podem ser divididos em dois grupos:

- Métodos novos: métodos propostos que utilizam uma nova técnica de redução, seja utilizando combinações de métodos existentes ou criação de um método completamente novo;
- Aprimoração dos métodos: trabalhos onde há uma implementação simples do método e/ou que contenha um aprimoramento de um método já existente na literatura.

O Quadro 7 apresenta a separação dos trabalhos em grupos através de seus respectivos identificadores.

Quadro 7 – Tipos de trabalhos.

ID do trabalho	Tipo do método
1	Novo
4	
5	
6	
8	
2	Aprimoração
3	
7	

Fonte: Autoria prórpia.

Q4 *- Quais as medidas de avaliação e/ou classificadores foram utilizadas?*

Para validar seu método Patil *et al.* (2018) utiliza acurácia e testa vários *subsets* com tamanhos diferentes. Os resultados obtidos são comparados com outros trabalhos que utilizam as mesmas bases de dados onde, nesta comparação, o método proposto se apresentou melhor.

Para a rede neural, Adem *et al.* (2019) utiliza acurácia e o tempo de processamento para a classificação em cada base de dados.

Para avaliar a representatividade do *subset*, Kinalis *et al.* (2019) compara sua base de dados reduzida pela rede neural com uma base de dados reduzida pelo método Seurat (um método de clusterização para genes). A base original e o subconjunto da rede neural foram clusterizados com o método *k-means*. Os resultados mostraram que os três métodos resultam em subconjuntos idênticos, porém utilizando redes neurais autocodificadoras os autores concluíram que sua estrutura lida bem com base de dados com ruído, conseguem reconstruir as vizinhanças

eliminadas dos atributos e que cada neurônio na rede representa um módulo do sequenciamento genético.

Em Geddes *et al.* (2019) os autores buscam utilizar os resultados do *framework* proposto e os clusterizam no método PCA, *K-means* e *Single-cell Interpretation via Multi-kernel LeaRning* (SIMLR) (um algoritmo de clusterização específico para a base de dados utilizada). Após essa etapa cada resultado é comparado com um "padrão ouro"(o que seria um clusterização ideal para essa base de dados). Os resultados mostram que, para todas as clusterizações, o *framework* apresentou ótimas representações mostrando que, além de eficiente, sua utilização também é viável para outros algoritmos de clusterização.

Sheet *et al.* (2020) comparam seu modelo com diversas técnicas de seleção de genes: *Support Vector Machine* (SVM), *Significance analysis of microarray* (SAM), *Bayesian Regularization* (BR), *Neighborhood Analysis* (NA) e *Gaussian Mixture Model* (GMM). Para validar os resultados das classificações os autores utilizam teste-t pareado e, também, quantos atributos de otonlogia genética foram gerados. Em todos os casos, o método proposto obteve resultados melhores.

Os classificadores utilizados por Adem (2020) para os testes do modelo são *softmax*, *K-Nearest Neighbor* (K-NN), SVM, *subspace* K-NN e Floresta Aleatória. Para a classificação da base de dados de câncer de mama o autor comparou seu modelo com o método PCA e sem utilizar nenhum método de redução (base original). Também houve posterior comparação com outros trabalhos. Para validar os resultados foi utilizada acurácia, *Positive predictive value* (PPV), *Negative predictive value* (NPV) e valores Kappa. Em todos os casos o modelo proposto apresentou resultados melhores, porém o tempo de processamento e custo computacional foi maior.

Wang *et al.* (2020) comparam a redução da rede neural com o método PCA e conclui que a rede consegue lidar com entradas de alta complexidade e alto nível, ou seja, a rede pode transformar dados complexos e não-lineares em atributos de baixo nível e mais simples.

Tangherloni *et al.* (2021) utilizam da ferramenta desenvolvida para testar as diversas variações de redes neurais autocodificadoras e compará-las com suas redes propostas. A rede proposta se mostrou ser a melhor opção quando se utiliza a função de perda Poisson ou Binominal Negativo.

3.4 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Neste Capítulo foram apresentados os trabalhos correlatos ao tema deste trabalho utilizando-se de uma metodologia de pesquisa. A metodologia e o mapeamento permitiram que fosse possível encontrar periódicos relevantes e com qual importância o tema está sendo estudado e aplicado.

Após o mapeamento sistemático é possível observar que redes neurais autocodificadoras são ótimas para a redução de bases de microarranjo, e que essa redução teve como objetivo tanto a clusterização dos dados quanto sua classificação.

Em relação à utilização da rede autocodificadora todos os trabalhos levantados no mapeamento a utilizam: seja ela sozinha ou unida à algum outro método. Em especial o método apresentado por Sheet *et al.* (2020) também utiliza uma MLP combinada a uma rede neural autocodificadora, contudo a MLP é utilizada para o treinamento de cada camada da rede neural — definindo assim seus pesos. O trabalho mostra que combinar essas duas estruturas cria uma base reduzida bem representada, logo adotar essa combinação é um bom caminho a ser adotado.

Também é possível observar que utilizar a rede neural autocodificadora para a tarefa de classificação apresenta resultados bem satisfatórios se combinado com outro algoritmo de IA, logo pretende-se criar um método utilizando a abordagem já utilizada no estado-da-arte do tema combinado redes neurais autocodificadoras com uma rede MLP de classificação.

4 EXTRAÇÃO DE ATRIBUTOS BASEADO EM AUTOCODIFICADORES UTILIZANDO CLASSIFICAÇÃO COMO PRÉ-TREINO (FEA-PTC)

Este Capítulo apresenta o método proposto denominado de *Feature Extraction based on Audoencoders using Pre Training with Classification* (FEA-PTC) para a redução de base de dados utilizando autocodificadores com um pré-treino de classificação. A Seção 4.1 descreve o funcionamento geral do método, a Seção 4.2 traz uma simulação do método em funcionamento e, por fim, a Seção 4.3 apresenta as considerações finais do capítulo.

4.1 DESCRIÇÃO DO MÉTODO

O método procura reduzir uma base de dados visando obter uma nova base de dados (sendo esta, de maneira significativa, menor e represente a base original) através da extração de atributos.

Com a rede autocodificadora a base de dados é reduzida através de um codificador e, de maneira optativa, é reconstruída através de um decodificador, em adição há uma etapa anterior ao autocodificador: uma rede classificadora utilizando o codificador.

O método possui três estruturas principais: o codificador, o classificador e o decodificador. O codificador possui um mapeamento de entrada X e uma representação reduzida Y definida pela Equação 7.

$$Y = f(X) = s_f(WX + b_x) \tag{7}$$

em que s_f é uma função de ativação para a rede, W uma matriz de pesos para parametrização e um vetor bias $b \in R^n$. O classificador recebe a representação reduzida Y para a classificação y definida pela Equação 8.

$$y = g(Y) = s_g(WY + b_y) \tag{8}$$

em que s_g é uma função de ativação para a rede, W uma matriz de pesos para parametrização e um vetor bias $b \in R^n$. O decodificador é responsável por mapear a representação Y para fazer a reconstrução Xi definido pela Equação 9.

$$Xi = h(Y) = s_h(W'Y + b_y) \tag{9}$$

onde s_h é uma função de ativação para a rede, W' uma matriz de pesos para parametrização e um vetor bias $b \in R^n$.

A representação da rede autocodificadora é apresentada na Figura 4. A rede é totalmente conectada, possuindo uma camada de entrada e duas camadas de saída. Para determinar o número de neurônios em cada camada e a quantidade de camadas ocultas para a estrutura autocodificadora, utilizou-se o critério de Siqueira (2019) com adaptações para comportar base de dados de microarranjo.

Segundo este critério, primeiro, é necessário determinar um Fator de Redução (Fr) e uma Porcentagem de Redução (Pr) definidas pela Equação 10:

$$Fr = Pr/6 \tag{10}$$

em que Pr é a porcentagem de redução desejada.

A Figura 4 apresenta a estrutura do método proposto. Nele: 1) Um mapeamento de entrada X é processado na estrutura codificadora (A) até o gargalo (D) e, logo após, continua por uma MLP de classificação (C) que gerará uma saída y representando a classe da entrada. 2) O mesmo codificador (A), agora com os pesos treinados pela classificação, segue até o gargalo (D) e passa pelo decodificador (E) até gerar uma saída réplica da entrada X, com isso tem-se uma estrutura codificadora que reduzirá uma entrada X.

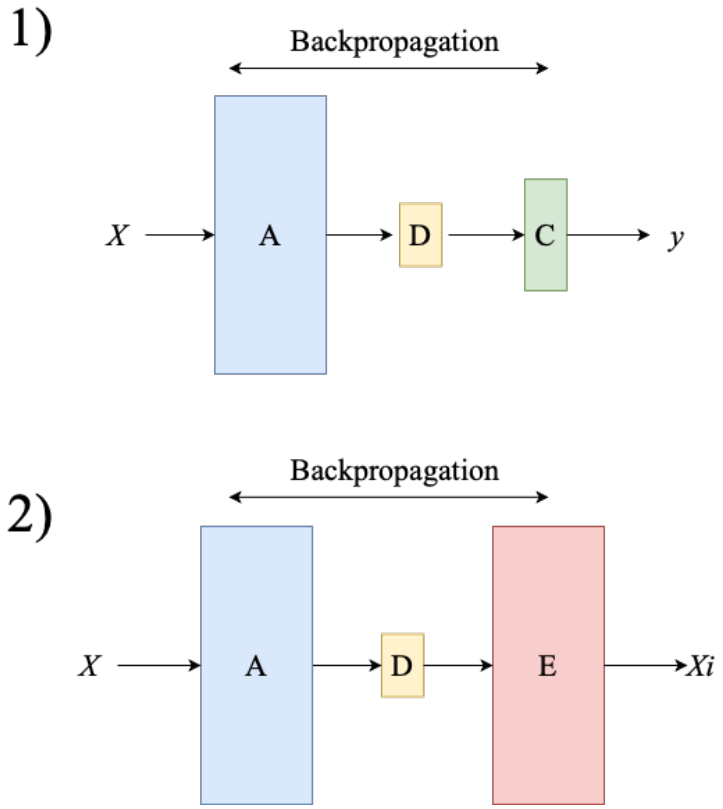
4.2 FUNCIONAMENTO

Para melhor entendimento do método, esta seção apresenta uma simulação de um treinamento utilizando-se de uma base de dados fictícia.

4.2.1 Base de dados fictícia

A base de dados está apresentada na Tabela 5, onde X_n representa um atributo (característica) e En representa cada instância da base (exemplos de amostras). A coluna Classe representa a classe de cada instância, logo esta base possui 9 características com 10 exemplos e 2 classes.

Figura 4 – Estrutura do modelo FEA-PTC.



Fonte: Autoria própria

Tabela 5 – Base de dados fictícia para melhor entendimento do método

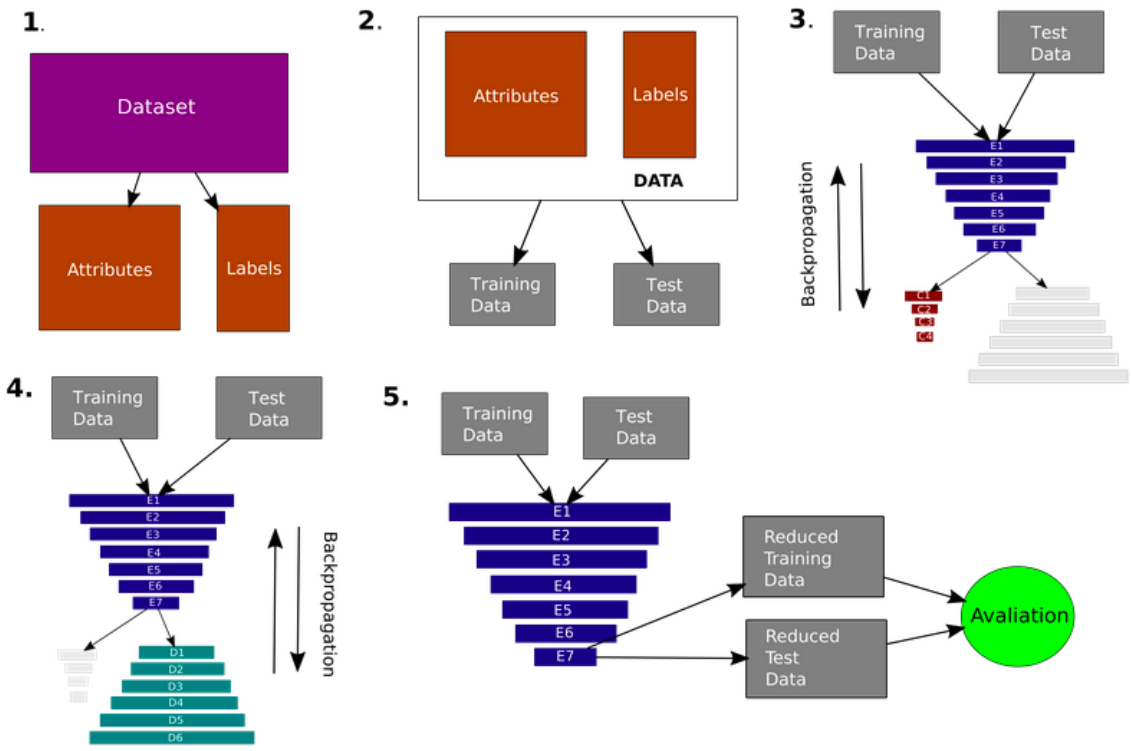
	x1	x2	x3	x4	x5	x6	x7	x8	x9	CLASSE
E1	0,45	0,42	0,56	0,92	0,95	0,34	0,21	0,88	0,16	A
E2	0,84	0,37	0,10	0,35	0,47	0,78	0,92	0,81	0,01	A
E3	0,27	0,49	0,04	0,41	0,83	0,19	0,36	0,67	0,91	A
E4	0,85	0,11	0,95	0,89	0,69	0,47	0,16	0,35	0,67	A
E5	0,26	0,82	0,53	0,95	0,74	0,89	0,33	0,41	0,49	A
E6	0,36	0,76	0,88	0,04	0,11	0,61	0,77	0,58	0,24	B
E7	0,67	0,49	0,06	0,54	0,35	0,01	0,63	0,19	0,16	B
E8	0,08	0,93	0,20	0,36	0,76	0,05	0,58	0,88	0,90	B
E9	0,79	0,89	0,26	0,22	0,44	0,57	0,21	0,75	0,46	B
E10	0,75	0,32	0,35	0,74	0,07	0,28	0,42	0,24	0,18	B

Fonte: Autoria própria.

4.2.2 Metodologia para criação da rede neural

Para o treinamento da rede utilizando a base de dados fictícia, a Figura 5 apresenta os passos utilizados pelo método desde o tratamento de dados até a validação final da rede.

Figura 5 – Passo-a-passo do funcionamento do método proposto



Fonte: Autoria própria.

4.2.2.1 Etapa 1: preparação dos dados

Em um primeiro momento é necessário o tratamento dos dados. Para um bom funcionamento da rede é necessário garantir que a base de dados:

- Não possua elementos faltantes;
- Não possua elementos com caracteres (palavras, letras, caracteres especias dentre outros);
- Tenha características e classes bem distintas e;
- As classes sejam vetorizadas para uma codificação *One-Hot*.

Para a base de dados fictícia é necessário vetorizar as classes, para tal a Equação 11 foi utilizada:

$$\vec{y} = \begin{cases} [0,1], & \text{se } Classe = A \\ [1,0], & \text{senão} \end{cases} \tag{11}$$

Uma vez com a base de dados tratada, é necessário separar as características das classes, tendo-se duas matrizes: a matriz de características X e a matriz de classes y, onde a classe y_i pertencerá às características X_i.

4.2.2.2 Etapa 2: separação dos dados em treino e teste

Para que não ocorra um vício na rede no treino, a base precisa ser separada em duas: base de treino e base de teste. Para este exemplo, a base de treino é composta por, cerca de, 66% da base de dados original, enquanto os demais dados pertencem à base de dados de teste. Os dados para cada base são selecionados de maneira aleatória.

4.2.2.3 Etapa 3: criação da rede, inicialização dos pesos e treinamento da rede classificadora

Nesta etapa ocorre a criação da rede e o treino de uma parte dela: a parte classificatória.

No primeiro momento é necessário definir o número de neurônios em cada camada. Seguindo os cálculos da Figura 4, arredondando os resultados para cima (função teto) e admitindo, para fins de simulação, $Pr = 0{,}95$ e $Fr = Pr/6 \approx 0{,}15$ tem-se a Tabela 6.

Tabela 6 – Números de neurônios para cada camada para este exemplo

Camada	Quantidade de neurônios
E1	$n = 9$
E2	$n * (1 - Fr) \approx 8$
E3	$n * (1^{\lor} Fr * 2) \approx 7$
E4	$n * (1^{\lor} Fr * 3) \approx 5$
E5	$n * (1^{\lor} Fr * 4) \approx 4$
E6	$n * (1^{\lor} Fr * 5) \approx 2$
E7	$n * (1^{\lor} Pr) \approx 1$
D1	$n * (1^{\lor} Fr * 5) \approx 2$
D2	$n * (1^{\lor} Fr * 4) \approx 4$
D3	$n * (1^{\lor} Fr * 3) \approx 5$
D4	$n * (1^{\lor} Fr * 2) \approx 7$
D5	$n * (1^{\lor} Fr) \approx 8$
D6	$n = 9$
C1	$(n * (1^{\lor} Pr))/2 \approx 1$
C2	$(n * (1^{\lor} Pr))/4 \approx 1$
C3	$(n * (1^{\lor} Pr))/6 \approx 1$
C4	Nº de classes = 2

Fonte: Autoria própria.

A caracterização da rede neural para a simulação da-se pela:

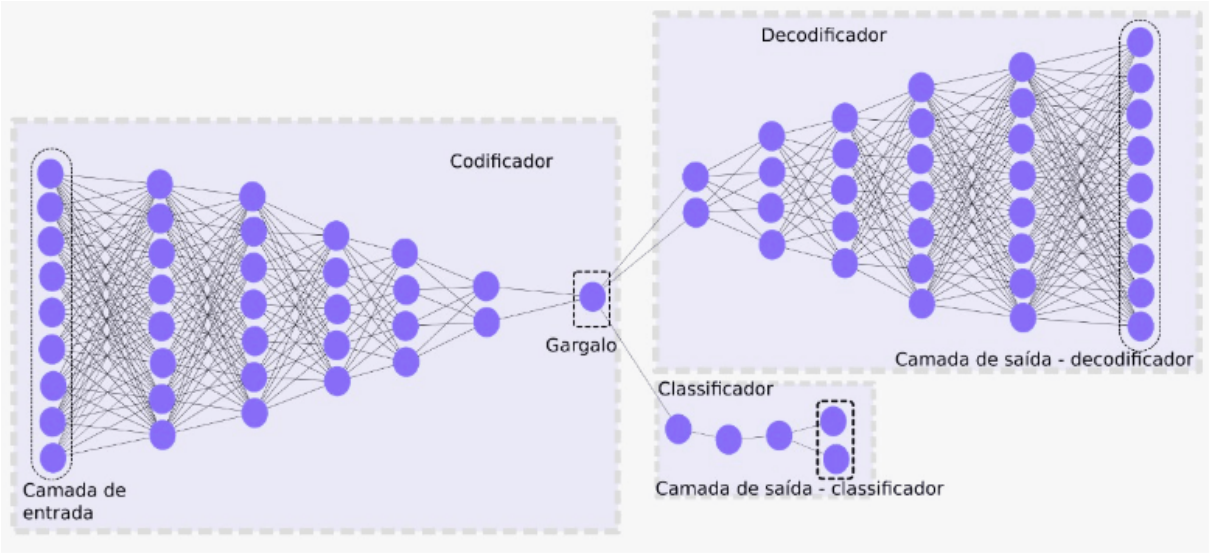
- **Topologia**: a rede é totalmente conectada, sem retroalimentação e com múltiplas saídas (duas);

- **Quantidade de camadas**: conforme adaptado de Siqueira (2019) a quantidade de camadas ocultas é descrita na Tabela 6 sendo: uma camada de entrada (E1), quatorze camadas (sendo uma delas o bottleneck(E7)) ocultas e duas camadas de saída (C4 e D6).
- **Quantidade de neurônios**: conforme descrito na Tabela 6
- **Neurônios**: Cada neurônio está associado a um peso aleatório entre [-1,1] que se atualizará conforme cada época do treino da rede.
- **Função de ativação**: todas as camadas — com exceção das camadas de saída - utilizam uma função ReLU por conta de sua continuidade. A camada de saída da classificação utiliza uma função *softmax* e a camada de saída de decodificação utiliza uma função *sigmoid*.
- **Estrutura codificadora**: Formada da camada de entrada até o gargalo (camada com a menor quantidade de neurônios da rede neural autocodificadora).
- **Estrutura decodificadora**: formada do gargalo até a camada de saída da decodificação.
- **Estrutura classificadora**: Formada da estrutura codificadora até a saída de classificação.
- **Função de otimização**: função do gradiente estocástico.
- **Função de custo**: na estrutura classificadora é utilizada a acurácia, enquanto na estrutura decodificadora é utilizada o MSE.
- **Treinamento**: Algoritmo de retropropagação (*backpropagation*).
- **Taxa de aprendizagem**: 0,05 para a classificação e 0,07 para a decodificação.
- **Critério de parada**: para este exemplo, somente uma época de treinamento.

A construção da rede será igual à rede apresentada na Figura 6. Nela também é mostrado com mais detalhes a divisão de cada estrutura de dentro da rede.

Após a criação da rede, a estrutura classificadora é treinada primeiro utilizando a base X_{treino} e, para mensurar sua eficácia, a base X_{teste} é utilizada para tirar o parâmetro da acurácia.

Figura 6 – Rede gerada seguindo os critérios da simulação



Fonte: Autoria própria.

4.2.2.4 Etapa 4: Treinamento da estrutura decodificadora

Uma vez a rede classificadora devidamente treinada agora a estrutura decodificadora é treinada, contudo a estrutura codificadora se mantém a mesma do treino da classificadora, ou seja, a matriz de pesos da estrutura codificadora é a resultante após o treino de classificação. Agora o treino é utilizado o MSE como critério de análise. As mesmas bases de treino e teste são usadas.

4.2.2.5 Etapa 5: Análise da base reduzida e validação

Após todos os treinos a estrutura codificadora é salva. As mesmas bases de teste e treino são utilizadas para obter bases de dados reduzidas. Estas, agora, são submetidas à validação para analisar se estas bases são suficientemente representativas ou não.

4.3 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Neste Capítulo foi apresentado o método FEA-PTC cujo objetivo é construir uma rede neural autocodificadora com uma estrutura codificadora pré-treinada com um classificador. Também uma simulação do funcionamento do método foi apresentada. Salienta-se ainda que as estruturas e parâmetros apresentados na simulação serão os mesmos para a implementação do método.

5 EXPERIMENTOS E RESULTADOS

Este Capítulo apresenta a metodologia utilizada para a validação do método FEA-PTC, bem como os experimentos realizados e os resultados obtidos. A Seção 5.1 especifica as ferramentas utilizadas para o desenvolvimento do algoritmo. A seção 5.2 apresenta as bases de dados utilizadas para o experimento. A Seção 5.3 descreve a metodologia que será aplicada para avaliação do método. Na seção 5.4 são apresentados os resultados obtidos após a aplicação da metodologia e, por fim, a Seção 5.5 relata as considerações finais do Capítulo.

5.1 FERRAMENTAS PARA IMPLEMENTAÇÃO

Para a implementação do código optou-se pela linguagem de programação *Python* (versão 3.7) tanto por sua simplicidade em programar quanto por possuir compatibilidade com a biblioteca *Tensorflow*.

A biblioteca *Tensorflow* é *open-source* e sua utilização focada na implementação de algoritmos de aprendizagem profunda faz com que seja boa para os experimentos. Além disso, a biblioteca abstrai detalhes das redes (tornando a lógica de programação mais simples) e possui duas *Application Programming Interface* (API)s essenciais para implementação e visualização da rede: *Keras* (utilizada para realizar uma implementação mais simples e mais fácil de realizar manutenções) e *Tensorboard* (que possibilita a visualização do comportamento dos treinos ao decorrer do tempo).

5.2 BASES DE DADOS

Para treinar e testar o método FEA-PTC utilizou-se das mesmas bases de dados utilizadas por Borges (2006). Todas elas são bases de dados do tipo microarranjo cuja principal característica são muitas características, porém poucos exemplares.

A Tabela 7 apresenta as características das bases de dados utilizadas. Para preencher as bases com elementos faltantes, utilizou-se o método da média (soma dos elementos de uma coluna dividido pela quantidade) para criar os valores faltantes.

É importante ressaltar que cada base será treinada separadamente e um modelo será criado para cada uma das bases.

Tabela 7 – Características das bases de dados

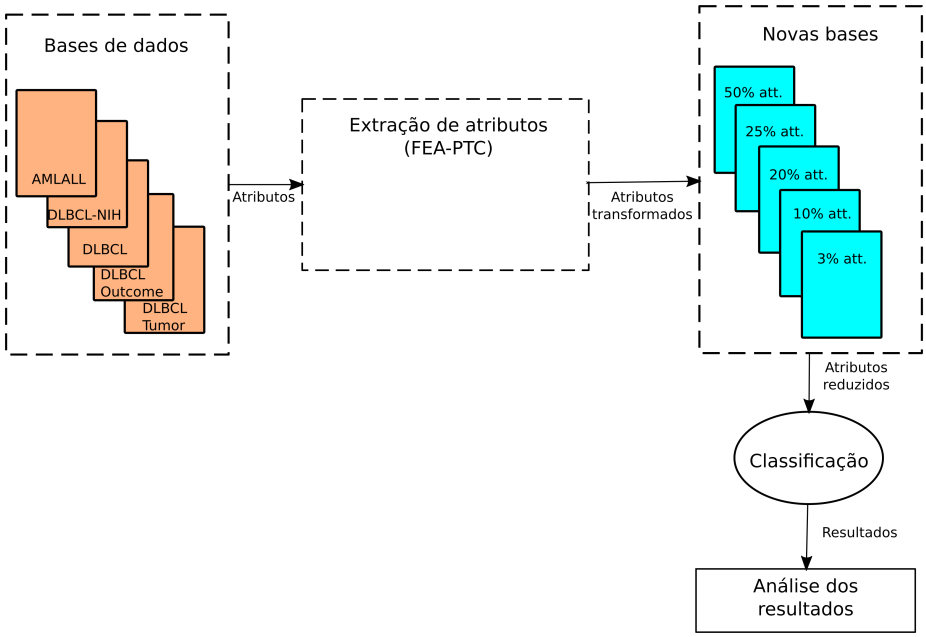
Bases de Dados	Quantidade de atributos	Quantidade de exemplos
AMLALL	7129	72
DLBCL-NIH	7399	240
DLBCL	4026	47
DLBCLOutcome	7129	58
DLBCLTumor	7129	77

Fonte: Autoria própria.

5.3 METODOLOGIA DOS EXPERIMENTOS

A metodologia utilizada para a realização dos experimentos é composta por, basicamente, três etapas: limpeza e separação das bases de dados, treinamento do método FEA-PTC e avaliação das bases reduzidas. A metodologia dos experimentos é mostrado na Figura 7

Figura 7 – Metodologia dos experimentos



Fonte: Autoria própria

- **Preparação das bases de dados**: assim como demonstrado na simulação (ver Capítulo 4) o mesmo será aplicado para as cinco bases de dados. As bases de dados foram devidamente trabalhadas para a utilização do método. Foi separado, para testes, 33% da base de dados original e o restante para o treino.

- **Método FEA-PTC**: nesta etapa o método é utilizado para a redução das bases. Os critérios adotados seguem os mesmos realizados na simulação - utilizando-se 50 épocas para a estrutura classificadora e 100 épocas para a estrutura decodificadora.

- **Avaliação das bases reduzidas**: para a avaliação dos resultados fizeram-se experimentos reduzindo as bases de cinco maneiras: com quantidades de atributos relativos à 3%, 10%, 20%, 25% e 50% da base original. Para a avaliação as bases reduzidas são separadas em teste e treino, em seguida são aplicados os algoritmos de classificação K-NN(com *K* igual a 1, 3, 5 e 7), SVM, Naive Bayes e Árvore de Decisão. Esse processo se repete vinte vezes para cada base, obtendo-se assim uma média de acurácias bem como seus desvios padrão.

5.4 RESULTADOS

Esta seção apresenta os resultados obtidos após a realização da metodologia.

5.4.1 Avaliação das bases reduzidas

A Tabela 8 apresenta os resultados obtidos na base de dados AML/ALL conforme os algoritmos de classificação utilizado e o percentual de redução atributos. Essa base de dados obteve as melhores acurácias dentre todas as outras.

Tabela 8 – Resultados obtidos na base de dados AML/ALL

	AML/ALL				
Algoritmo	3%	10%	20%	25%	50%
k-NN (k=1)	0.952 ± 0.043	0.970 ± 0.046	0.958 ± 0.035	0.966 ± 0.038	0.958 ± 0.038
k-NN (k=3)	0.945 ± 0.043	0.960 ± 0.046	0.943 ± 0.035	0.966 ± 0.038	0.960 ± 0.038
k-NN (k=5)	0.935 ± 0.046	0.966 ± 0.038	0.945 ± 0.041	0.958 ± 0.051	0.952 ± 0.035
k-NN (k=7)	0.927 ± 0.045	0.958 ± 0.047	0.935 ± 0.038	0.954 ± 0.057	0.933 ± 0.048
SVM	0.95 ± 0.040	0.977 ± 0.020	0.958 ± 0.034	0.970 ± 0.043	0.931 ± 0.042
Naive Bayes	0.95 ± 0.033	0.964 ± 0.042	0.962 ± 0.045	0.970 ± 0.032	0.962 ± 0.034
Árvore de Decisão	0.920 ± 0.057	0.95 ± 0.048	0.943 ± 0.051	0.958 ± 0.052	0.945 ± 0.069

Fonte: Autoria própria.

A Tabela 9 apresenta os resultados obtidos na base de dados DLBCL-NIH conforme os algoritmos de classificação utilizado e o percentual de redução atributos. Dentre todas as reduções, reduzir a 50% apresentou as piores acurácias dentre todas as outras na mesma base.

A Tabela 10 apresenta os resultados obtidos na base de dados DLBCL conforme os algoritmos de classificação utilizado e o percentual de redução atributos. A base apresentou resultados com altas acurácias - sendo muito idêntica à base AML/ALL, porém apresentou desvios padrões maiores.

A Tabela 11 apresenta os resultados obtidos na base de dados DLBCL-Outcome conforme os algoritmos de classificação utilizado e o percentual de redução atributos. Esta base de dados foi a que apresentou os piores resultados se comparada com as demais, ela também

Tabela 9 – Resultados obtidos na base de dados DLBCL-NIH

DLBCL-NIH					
Algoritmo	3%	10%	20%	25%	50%
k-NN (k=1)	0.838 ± 0.030	0.825 ± 0.042	0.819 ± 0.037	0.801 ± 0.032	0.729 ± 0.040
k-NN (k=3)	0.866 ± 0.030	0.845 ± 0.042	0.849 ± 0.037	0.826 ± 0.032	0.744 ± 0.040
k-NN (k=5)	0.876 ± 0.028	0.863 ± 0.044	0.848 ± 0.041	0.824 ± 0.037	0.710 ± 0.041
k-NN (k=7)	0.873 ± 0.029	0.870 ± 0.040	0.846 ± 0.032	0.823 ± 0.030	0.686 ± 0.033
SVM	0.861 ± 0.029	0.869 ± 0.044	0.863 ± 0.025	0.842 ± 0.031	0.666 ± 0.052
Naive Bayes	0.865 ± 0.027	0.825 ± 0.047	0.822 ± 0.047	0.805 ± 0.041	0.658 ± 0.055
Árvore de Decisão	0.84 ± 0.036	0.817 ± 0.0398	0.833 ± 0.043	0.818 ± 0.040	0.733 ± 0.043

Fonte: Autoria própria.

Tabela 10 – Resultados obtidos na base de dados DLBCL

DLBCL					
Algoritmo	3%	10%	20%	25%	50%
k-NN (k=1)	0.937 ± 0.061	0.943 ± 0.045	0.909 ± 0.063	0.925 ± 0.068	0.931 ± 0.055
k-NN (k=3)	0.95 ± 0.061	0.946 ± 0.045	0.896 ± 0.063	0.921 ± 0.068	0.937 ± 0.055
k-NN (k=5)	0.943 ± 0.065	0.931 ± 0.058	0.881 ± 0.058	0.931 ± 0.062	0.937 ± 0.044
k-NN (k=7)	0.943 ± 0.065	0.921 ± 0.065	0.9 ± 0.05	0.943 ± 0.062	0.940 ± 0.041
SVM	0.940 ± 0.066	0.940 ± 0.054	0.909 ± 0.046	0.934 ± 0.063	0.956 ± 0.028
Naive Bayes	0.962 ± 0.041	0.940 ± 0.060	0.903 ± 0.050	0.943 ± 0.065	0.928 ± 0.045
Árvore de Decisão	0.95 ± 0.061	0.915 ± 0.066	0.871 ± 0.082	0.906 ± 0.064	0.928 ± 0.049

Fonte: Autoria própria.

apresenta a pior acurácia obtida dentre todos os resultados do estudo (somente 42% de acurácia para 50% dos atributos para o Algoritmo k-NN para k igual à 7). Os desvios-padrão também foram os maiores neste estudo, mostrando que o método se mostra impreciso ao analisar essa base.

Tabela 11 – Resultados obtidos na base de dados DLBCL-Outcome

DLBCL-Outcome					
Algoritmo	3%	10%	20%	25%	50%
k-NN (k=1)	0.817 ± 0.063	0.807 ± 0.074	0.815 ± 0.065	0.777 ± 0.073	0.532 ± 0.114
k-NN (k=3)	0.815 ± 0.063	0.847 ± 0.074	0.842 ± 0.065	0.837 ± 0.073	0.475 ± 0.114
k-NN (k=5)	0.837 ± 0.056	0.845 ± 0.052	0.830 ± 0.064	0.855 ± 0.058	0.447 ± 0.108
k-NN (k=7)	0.84 ± 0.062	0.845 ± 0.056	0.815 ± 0.054	0.855 ± 0.063	0.427 ± 0.108
SVM	0.83 ± 0.059	0.85 ± 0.063	0.825 ± 0.062	0.847 ± 0.062	0.455 ± 0.110
Naive Bayes	0.807 ± 0.067	0.857 ± 0.063	0.822 ± 0.055	0.837 ± 0.066	0.527 ± 0.116
Árvore de Decisão	0.785 ± 0.098	0.84 ± 0.069	0.8 ± 0.083	0.815 ± 0.080	0.5 ± 0.088

Fonte: Autoria própria.

A Tabela 12 apresenta os resultados obtidos na base de dados DLBCL-Tumor conforme os algoritmos de classificação utilizado e o percentual de redução atributos. A base se apresentou satisfatória para as acurácias, porém em diversas situações apresentou desvios padrões maiores que as outras bases.

Ao observar todos os resultados juntos, é possível observar que algumas bases obtiveram resultados bem satisfatórios, porém outras apresentaram resultados ruins e desvio padrões altos.

Tabela 12 – Resultados obtidos na base de dados DLBCL-Tumor

	DLBCL-Tumor				
Algoritmo	3%	10%	20%	25%	50%
k-NN (k=1)	0.909 ± 0.044	0.911 ± 0.040	0.875 ± 0.063	0.932 ± 0.051	0.917 ± 0.054
k-NN (k=3)	0.907 ± 0.044	0.923 ± 0.040	0.876 ± 0.063	0.913 ± 0.051	0.890 ± 0.054
k-NN (k=5)	0.896 ± 0.045	0.882 ± 0.057	0.869 ± 0.068	0.9 ± 0.042	0.855 ± 0.056
k-NN (k=7)	0.890 ± 0.044	0.863 ± 0.069	0.834 ± 0.085	0.898 ± 0.057	0.836 ± 0.072
SVM	0.905 ± 0.046	0.907 ± 0.063	0.878 ± 0.071	0.930 ± 0.051	0.825 ± 0.093
Naive Bayes	0.934 ± 0.047	0.913 ± 0.054	0.896 ± 0.045	0.946 ± 0.039	0.938 ± 0.037
Árvore de Decisão	0.907 ± 0.071	0.905 ± 0.072	0.882 ± 0.047	0.959 ± 0.046	0.923 ± 0.053

Fonte: Autoria própria.

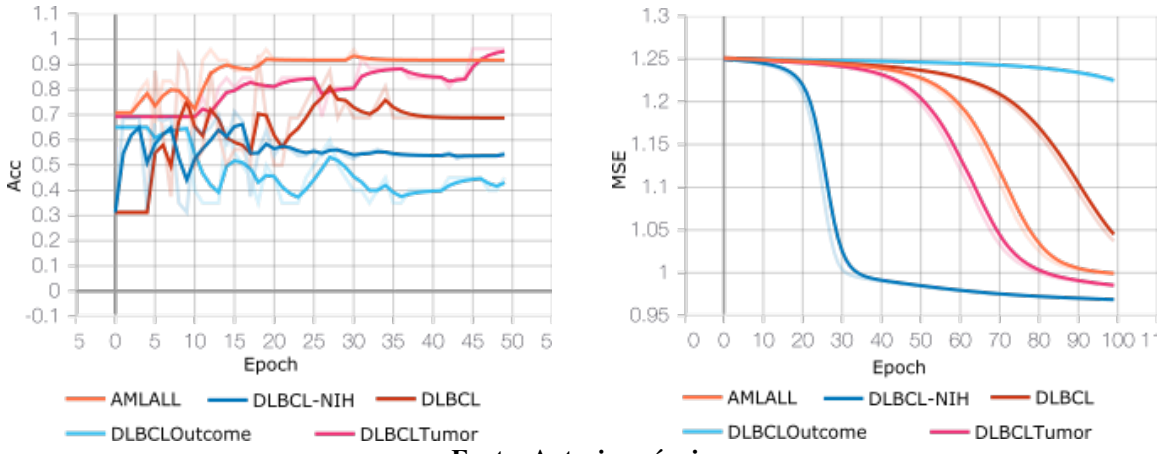
5.4.2 Evolução do treino das bases

Apesar de resultados promissores após as análises de acurácia, também é importante analisar como ocorreu o treino da rede neural para observar como ela se comportou e se os parâmetros utilizados são favoráveis. Para isso as imagens a seguir mostram a evolução dos treinos por época de treino observando dois parâmetros: a acurácia no treino da estrutura classificadora e o MSE da estrutura decodificadora.

A Figura 8 mostra a evolução do treino através das épocas observado a acurácia e o MSE para 3% dos atributos. Nota-se que há uma relação entre a acurácia e a redução: quanto melhor a acurácia, menor o erro quadrático na reconstrução.

Ao observar individualmente as bases, é possivel notar que as bases AMLALL, DLBCL, DLBCL-NIH e DLBCL-Tumor obtiveram taxas de acurária acima dos 50% e valores de MSE convergindo para o mínimo global (com valores inferiores à 1), contudo a base de dados DLBCLOutcome apresentou os piores resultados com acurárias abaixo dos 50% e valores de MSE que não convergiram. Isso significa que a rede apresentou dificuldades em encontrar padrões nessa base em específico.

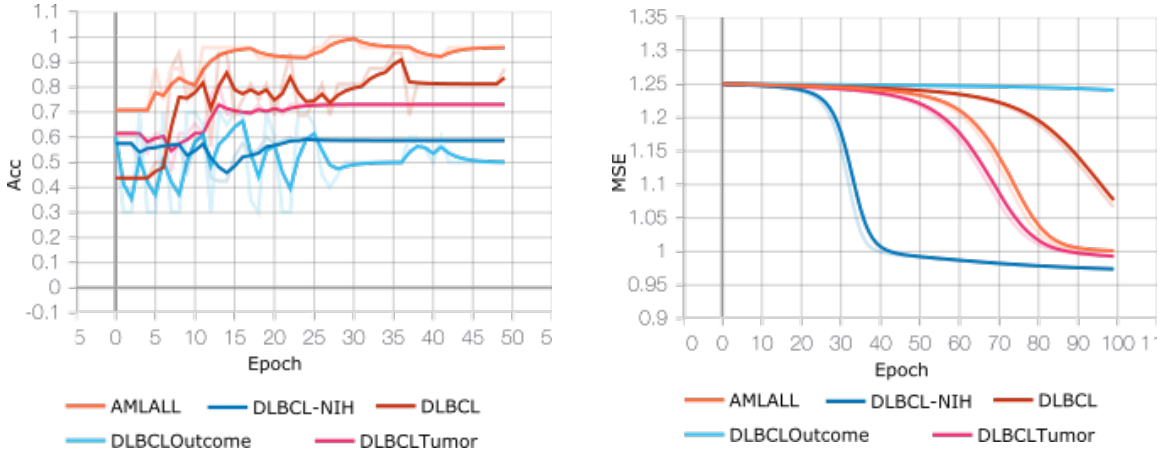
Figura 8 – Acurácia e MSE para 3% dos atributos por base de dados



Fonte: Autoria própria

A Figura 9 mostra a evolução do treino através das épocas observado a acurácia e o MSE para 10% dos atributos. Aqui vale-se das observações realizadas anteriormente para 3% dos atributos.

Figura 9 – Acurácia e MSE para 10% dos atributos por base de dados

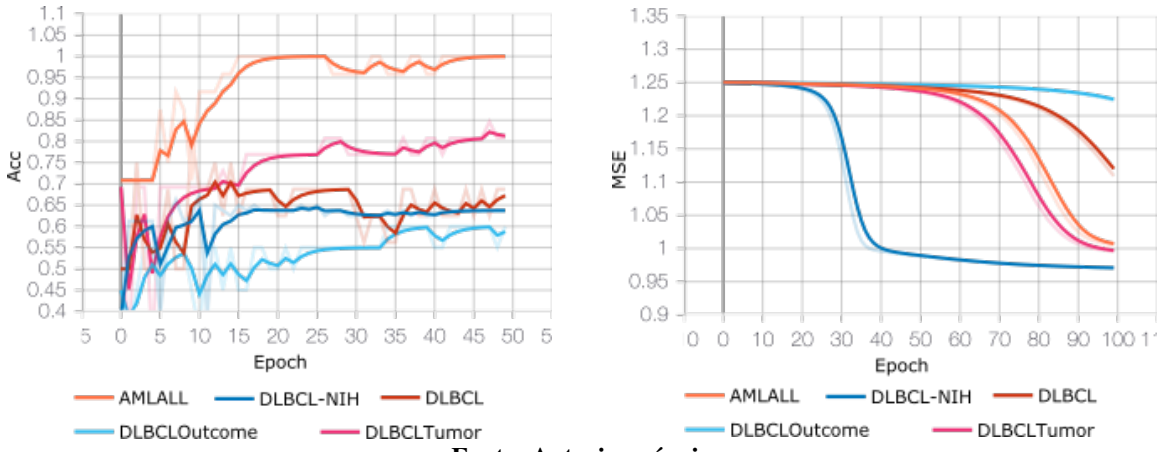


Fonte: Autoria própria

A Figura 10 mostra a evolução do treino através das épocas observado a acurácia e o MSE para 20% dos atributos. Nesta configuração as bases de dados AMLALL apresentou acurácias de 100%, ou seja, a rede acertou todas as classificações da base de teste.

Apesar de algumas variações em algumas bases as observações ainda se mantém as mesmas: MSEs convergindo ao mínimo local e acurácias acima de 50% (exceto a base DLBCLOutcome)

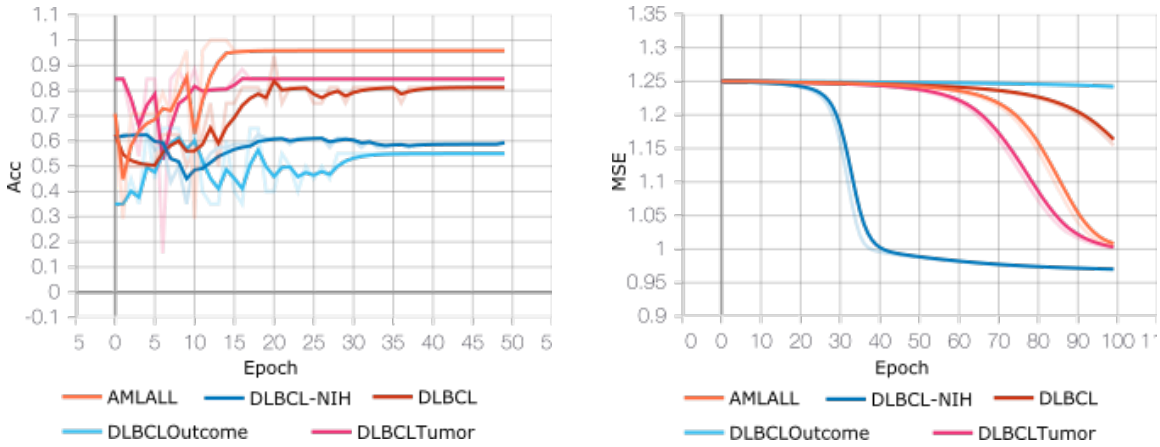
Figura 10 – Acurácia e MSE para 20% dos atributos por base de dados



Fonte: Autoria própria

A Figura 11 mostra a evolução do treino através das épocas observado a acurácia e o MSE para 25% dos atributos. Aqui é possível notar que as bases de dados começam a apresentar uma maior demora para convergir os erros. Observa-se que há uma constante convergência dos erros, contudo ela é mais lenta (essa demora é bem perceptível se observada a base DLBCL ao longo dos experimentos).

Figura 11 – Acurácia e MSE para 25% dos atributos por base de dados

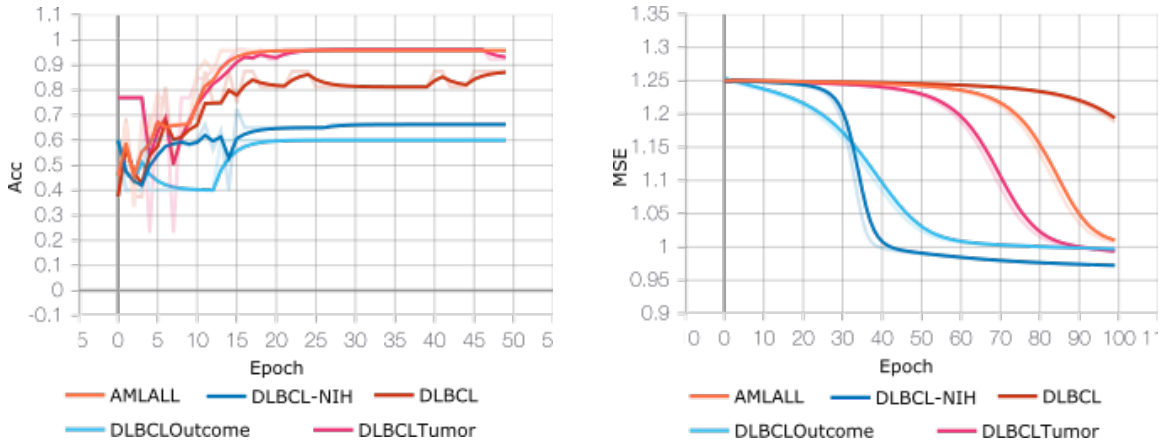


Fonte: Autoria própria

A Figura 12 mostra a evolução do treino através das épocas observado a acurácia e o MSE para 50% dos atributos. Dentre todos os experimentos apresentados este foi o que apresentou resultados mais distintos.

Neste a base de dados DLBCLOutcome (a que apresentava os treinos mais inferiores) apresentou uma acurácia com uma constância de 60% e apresentou uma convergência no MSE. As demais bases não apresentaram comportamentos distintos ao já vistos até então.

Figura 12 – Acurácia e MSE para 50% dos atributos por base de dados



Fonte: Autoria própria

5.4.3 Comparação dos resultados

A comparação completa das acurácias obtidas por este trabalho e por Borges (2006) é apresentada no Apêndice A.

De maneira geral o método FEA-PTC apresentou uma acurácia maior na maior parte dos casos, independentemente das bases e do método de classificação. Na base de dados DLBCL-Outcome (a que apresentou resultados inferiores) o método proposto mostrou resultados com acurácias maiores em todos os casos.

Apesar das acurácias maiores os experimentos também apresentaram que os desvios padrão também foram maiores. Isso indica que a precisão nas acurácias é menor dos que as obtidas por Borges (2006).

5.5 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Este Capítulo mostrou os resultados do método FEA-PTC após a aplicação da metodologia proposta. Os resultados mostraram que o método conseguiu obter boas classificações e que a base de dados reduzida é bem representativa.

Apesar dos bons resultados o treinamento das bases foi lento. O custo computacional se mostrou alto tanto no treino da estrutura classificadora quanto na estrutura decodificadora, contudo nenhum parâmetro foi utilizado para realizar essa medição.

A base de dados DLBCL-Outcome apresentou resultados inferiores se comparada às outras bases em todos os experimentos. Uma possível causa para esse problema são os muitos valores faltantes na base antes da etapa de limpeza, fazendo com que o método de imputação

utilizado inserisse diversos números idênticos por ela — tornando assim a base cheia de valores repetidos.

Também é importante salientar que os resultados foram somente comparados com os de Borges (2006) utilizando os mesmos algoritmos de classificação e o parâmetro do desvio padrão.

6 CONCLUSÃO

Métodos de redução de dimensionalidade visam reduzir bases de dados que possuam o problema da "maldição da dimensionalidade". Essa redução é de alta relevância visto que ela elimina atributos irrelevante e/ou redundantes.

A redução de dimensionalidade pode melhorar o desempenho na tarefa de classificação, visualização e armazenamento de grandes bases de dados. Este trabalho teve como objetivo utilizar um método para a redução treinando, primeiro, a estrutura codificadora com classificação e, em seguida, utilizar a estrutura codificadora numa rede autocodificadora. A proposta de otimização de um método existente não só visa realizar as vantagens de um algoritmo de redução de dimensionalidade como também procura melhorar algoritmos já existentes.

O método FEA-PTC cumpre sua função de reduzir as bases de dados de microarranjo, contudo seu custo computacional se mostrou elevado.

O método realizou boas reduções nas cinco bases de dados testadas, contudo, em especial, a base de dados DLBC-Outcome foi problemática ao método - apresentando resultados insatisfatórios em diversos momentos.

Como possíveis continuações a este trabalho sugerem-se:

- Outras bases de dados: utilizar o método para o teste em outros tipos de bases de dados;
- Comparação de métodos: comparar o método deste trabalho com outros existentes na literatura (tanto de seleção de atributos quanto de extração de atributos) e;
- Aperfeiçoamento do método: buscar maneiras de aperfeiçoar o método utilizando, por exemplo, um número diferente de neurônios, uma quantidade diferente de camadas internas, etc.

REFERÊNCIAS

ADEM, K. Diagnosis of breast cancer with stacked autoencoder and subspace knn. **Physica A: Statistical Mechanics and its Applications**, v. 551, 2020.

ADEM, K.; KILICARSLAN, S.; COMERT, O. Classification and diagnosis of cervical cancer with stacked autoencoder and softmax classification. **Expert Systems with Applications**, v. 115, p. 557–564, 2019.

ADIWIJAYA; WISESTY, Untari N.; LISNAWATI, E.; ADITSANIA, A.; KUSUMO, Dana S. Dimensionality reduction using principal component analysis for cancer detection based on microarray data classification. **Journal of Computer Science**, p. 1521 – 1530, 2018.

BORGES, H. B. **Redução de dimensionalidade em bases de dados de expressão gênica**. 2006. — Pontifícia Universidade Católica do Paraná., Curitiba, Brasil, 2006.

CASTRO, L. N.; FERRARI, D. G. **Introdução à mineração de dados: conceitos básicos, algoritmos e aplicações**. São Paulo, BR: Saraiva, 2016. 351 p.

GEDDES, T. A.; KIM, T.; NAN, L.; BURCHFIELD, J.; YANG, J.; TAO, D.; YANG, P. Autoencoder-based cluster ensembles for single-cell rna-seq data analysis. **BMC Bioinformatics**, v. 20, 2019.

GÉRON, Aurélien. **Hands-On Machine Learning with Scikit-Learn & TensorFlow**. Sebastopol, USA: O'Reilly Media, Inc., 2017. 541 p.

KINALIS, S.; NIELSEN, F.; WINTHER, O.; BAGGER, F. O. Deconvolution of autoencoders to learn biological regulatory modules from single cell mrna sequencing data. **BMC Bioinformatics**, v. 20, 2019.

KITCHENHAM, Barbara; CHARTES, Stuart. Guidelines for performing systematic literature reviews in software engineering. **School of Computer Science and Mathematics**, p. 1–65, 2007.

MAATEN, Laurens Van Der; POSTMA, Eric; HERIK, Jaap Van Den. Dimensionality reduction: A comparative review. **J Mach Learn Res**, v. 10, 2009.

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

MENG, Qinxue; CATCHPOOLE, Daniel; SKILLICORN, David; KENNEDY, Paul. Relational Autoencoder for Feature Extraction. **IEEE Xplore**, 2017.

PATIL, S.; NAIK, G.; PAI, R.; GAD, R. Stacked autoencoder for classification of glioma grade iii and grade iv. **Biomedical Signal Processing and Control**, v. 46, p. 67–75, 2018.

PINAYA, W. H. L.; VIEIRA, S.; DIAS, R. G.; MECHELLI, A. **Machine Learning: Methods and Applications to Brain Disorders**. *[S.l.]*: Academic Press, 2020. 193–208 p.

SHEET, S.; GHOSH, A.; GOSH, R.; CHAKRABARTI, A. Identification of cancer mediating biomarkers using stacked denoising autoencoder model - an application on human lung data. **Procedia Computer Science**, v. 167, p. 686–695, 2020.

SIQUEIRA, R. F. **Redução de dimensionalidade em bases de dados de classificação hiarárquica multirrótulo usando autoencoders**. 2019. — Universidade Tecnológica Federal do Paraná, Ponta Grossa, Brasil, 2019.

TANGHERLONI, A.; RICCIUTI, F.; BESOZZI, D.; LIò, P.; CVEJI, A. Analysis of single-cell rna sequencing data based on autoencoders. **BMC Bioinformatics**, v. 22, 2021.

VENKATESH, B.; ANURADHA, J. A review of feature selection and its methods. **Cybernetics and Information Technologies**, v. 19, 2019.

WANG, J.; XIE, X.; SHI, J.; HE, W; CHEN, Q; CHEN, L.; GU, W.; ZHOU, T. Denoising autoencoder, a deep learning algorithm, aids the identification of a novel molecular signature of lung adenocarcinoma. **Genomics, Proteomics & Bioinformatics**, v. 18, p. 468–480, 2020.

ZEBARI, Rizgar R.; ABDULAZEEZ, Adnan Mohsin; ZEEBAREE, Diyar Qader; ZEBARI, Dilovan Asaad; SAEED, Jwan Najeeb. A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction. **Journal of Applied Science and Technology Trends**, p. 56–70, 2020.

APÊNDICE A – COMPARAÇÃO DOS RESULTADOS

APÊNDICE A – COMPARAÇÃO DOS RESULTADOS

Este Apêndice apresenta as tabelas comparativas entre os resultados apresentados no Capítulo 5 e Borges (2006) por base de dados.

A Tabela 13 mostra a comparação entre os métodos na base de dados AML/ALL. Ao analisar os resultados é possível notar que, geralmente, o método deste trabalho obteve acurácias maiores, porém não muito distantes das de Borges (2006), contudo os desvios-padrão foram maiores, indicando que o método deste trabalho é menos preciso.

A Tabela 14 mostra a comparação entre os métodos na base de dados DLBCL-NIH. Nesta base os resultados obtidos por esse trabalho se mostraram expressivamente maiores, indicando que o método deste trabalho se adaptou melhor a esta base criando, assim, uma base de dados reduzida que representa bem a base original.

A Tabela 15 mostra a comparação entre os métodos na base de dados DLBCL. Os resultados apresentados neste trabalho se apresentaram ligeiramente melhores, contudo para a classificação do algoritmo SVM Borges (2006) apresentou resultados melhores e para o algoritmo Naive Bayes os resultados foram próximos.

A Tabela 16 mostra a comparação entre os métodos na base de dados DLBCL-Outcome. A base se apresentou problemática para ambos os métodos (especialmente quando a base foi reduzida para 50% dos atributos), no entanto, os resultados deste trabalho foram melhores. Também é importante salientar que Borges (2006) teve desvios-padrão menores, sugerindo assim que esse método é mais preciso.

A Tabela 17 mostra a comparação entre os métodos na base de dados DLBCL-Tumor. Aqui cabe as mesmas observações referentes à Tabela 13

Tabela 13 – Comparação das acurácias na base de dados AML/ALL

AML/ALL						
Algoritmo	**Método**	**Número de atributos**				
		3,00%	10,00%	20,00%	25,00%	50,00%
k-NN (k = 1)	Este trabalho	0.9520 ± 0.0438	0.9708 ± 0.0465	0.9583 ± 0.0355	0.9666 ± 0.0386	0.9583 ± 0.0383
	(BORGES, 2006)	0.8929 ± 0.0231	0.9188 ± 0.0177	0.9186 ± 0.0154	0.9184 ± 0.0126	0.9254 ± 0.0158
k-NN (k = 3)	Este trabalho	0.9458 ± 0.0438	0.9604 ± 0.0465	0.9437 ± 0.0355	0.9666 ± 0.0386	0.9604 ± 0.0383
	(BORGES, 2006)	0.8947 ± 0.0184	0.9195 ± 0.0231	0.9304 ± 0.0100	0.9290 ± 0.0112	0.9315 ± 0.0060
k-NN (k = 5)	Este trabalho	0.9354 ± 0.0465	0.9666 ± 0.0386	0.9458 ± 0.0418	0.9583 ± 0.0510	0.9520 ± 0.0355
	(BORGES, 2006)	0.8877 ± 0.0265	0.9082 ± 0.0162	0.9111 ± 0.0159	0.9129 ± 0.0188	0.9220 ± 0.0121
k-NN (k = 7)	Este trabalho	0.9270 ± 0.0454	0.9583 ± 0.0475	0.9354 ± 0.0383	0.9541 ± 0.0572	0.9333 ± 0.0482
	(BORGES, 2006)	0.8807 ± 0.0191	0.9006 ± 0.0165	0.9063 ± 0.0157	0.9040 ± 0.0136	0.9013 ± 0.0166
SVM	Este trabalho	0.95 ± 0.0408	0.9770 ± 0.0207	0.9583 ± 0.0348	0.9708 ± 0.0438	0.9312 ± 0.0422
	(BORGES, 2006)	0.9630 ± 0.0180	0.9714 ± 0.0117	0.9786 ± 0.0101	0.9786 ± 0.0075	0.9786 ± 0.0075
Naive Bayes	Este trabalho	0.95 ± 0.0338	0.9645 ± 0.0422	0.9625 ± 0.0454	0.9708 ± 0.0325	0.9625 ± 0.0346
	(BORGES, 2006)	0.9145 ± 0.0242	0.9300 ± 2,65	0.9343 ± 0.0215	0.9300 ± 0.0207	0.9243 ± 0.0166
Árvore de Decisão	Este trabalho	0.9208 ± 0.0572	0.95 ± 0.0485	0.9437 ± 0.0514	0.9583 ± 0.0527	0.9458 ± 0.0698
	(BORGES, 2006)	0.8048 ± 0.0597	0.8316 ± 0.0540	0.8114 ± 0.0662	0.8084 ± 0.0732	0.8234 ± 0.0692

Fonte: Autoria própria

Tabela 14 – Comparação das acurácias na base de dados DLBCL-NIH

DLBCL-NIH						
Algoritmo	**Método**	**Número de atributos**				
		3,00%	10,00%	20,00%	25,00%	50,00%
k-NN (k = 1)	Este trabalho	0.8387 ± 0.0306	0.8256 ± 0.0424	0.8193 ± 0.0375	0.8018 ± 0.0321	0.7293 ± 0.0409
	(BORGES, 2006)	0.5671 ± 0.0245	0.5650 ± 0.0167	0.5663 ± 0.0122	0.5746 ± 0.0176	0.5717 ± 0.0136
k-NN (k = 3)	Este trabalho	0.8662 ± 0.0306	0.8456 ± 0.0424	0.8493 ± 0.0375	0.8268 ± 0.0321	0.7443 ± 0.0409
	(BORGES, 2006)	0.5871 ± 0.0191	0.5954 ± 0.0320	0.6033 ± 0.0237	0.6033 ± 0.0211	0.6112 ± 0.0220
k-NN (k = 5)	Este trabalho	0.8768 ± 0.0283	0.8631 ± 0.0441	0.8487 ± 0.0416	0.8243 ± 0.0371	0.7106 ± 0.0413
	(BORGES, 2006)	0.5721 ± 0.0157	0.5900 ± 0.0179	0.5900 ± 0.0221	0.5884 ± 0.0265	0.5775 ± 0.0199
k-NN (k = 7)	Este trabalho	0.8737 ± 0.0298	0.8706 ± 0.0407	0.8468 ± 0.0328	0.8237 ± 0.0305	0.6868 ± 0.0336
	(BORGES, 2006)	0.5879 ± 0.0181	0.5821 ± 0.0231	0.5787 ± 0.0212	0.5763 ± 0.0295	0.5775 ± 0.0191
SVM	Este trabalho	0.8618 ± 0.0294	0.8693 ± 0.0441	0.8631 ± 0.0257	0.8425 ± 0.0319	0.6662 ± 0.0524
	(BORGES, 2006)	0.5988 ± 0.0343	0.6229 ± 0.0396	0.6250 ± 0.0390	0.6338 ± 0.0323	0.6417 ± 0.0242
Naive Bayes	Este trabalho	0.8656 ± 0.0279	0.825 ± 0.0472	0.8225 ± 0.0475	0.8056 ± 0.0411	0.6581 ± 0.0553
	(BORGES, 2006)	0.5983 ± 0.0198	0.5979 ± 0.0176	0.6050 ± 0.0246	0.5987 ± 0.0155	0.5987 ± 0.0108
Árvore de Decisão	Este trabalho	0.84 ± 0.0367	0.8175 ± 0.0398	0.8337 ± 0.0431	0.8181 ± 0.0400	0.7331 ± 0.0437
	(BORGES, 2006)	0.5533 ± 0.0461	0.5208 ± 0.0261	0.5487 ± 0.0431	0.5458 ± 0.0248	0.5546 ± 0.0252

Fonte: Autoria própria

Tabela 15 – Comparação das acurácias na base de dados DLBCL-NIH

DLBCL-NIH						
Algoritmo	**Método**	**Número de atributos**				
		3,00%	10,00%	20,00%	25,00%	50,00%
k-NN (k = 1)	Este trabalho	0.9375 ± 0.0612	0.9437 ± 0.0453	0.9093 ± 0.0633	0.925 ± 0.0681	0.9312 ± 0.0559
	(BORGES, 2006)	0.7600 ± 0.0557	0.8100 ± 0.0533	0.8070 ± 0.0414	0.8030 ± 0.0350	0.8200 ± 0.0225
k-NN (k = 3)	Este trabalho	0.95 ± 0.0612	0.9468 ± 0.0453	0.8968 ± 0.0633	0.9218 ± 0.0681	0.9375 ± 0.0559
	(BORGES, 2006)	0.7910 ± 0.0450	0.8635 ± 0.0350	0.8655 ± 0.0395	0.8550 ± 0.0252	0.8545 ± 0.0219
k-NN (k = 5)	Este trabalho	0.9437 ± 0.0652	0.9312 ± 0.0589	0.8812 ± 0.0589	0.9312 ± 0.0621	0.9375 ± 0.0441
	(BORGES, 2006)	0.8080 ± 0.0693	0.8695 ± 0.0300	0.8585 ± 0.0189	0.8695 ± 0.0215	0.8735 ± 0.0172
k-NN (k = 7)	Este trabalho	0.9437 ± 0.0652	0.9218 ± 0.0651	0.9 ± 0.05	0.9437 ± 0.0621	0.9406 ± 0.0418
	(BORGES, 2006)	0.8070 ± 0.0925	0.8400 ± 0.0588	0.8780 ± 0.0317	0.8625 ± 0.0333	0.8700 ± 0.0248
SVM	Este trabalho	0.9406 ± 0.0669	0.9406 ± 0.0540	0.9093 ± 0.0462	0.9343 ± 0.0639	0.9562 ± 0.0286
	(BORGES, 2006)	0.9060 ± 0.0561	0.9295 ± 0.0231	0.9400 ± 0.0222	0.9320 ± 0.0324	0.9435 ± 0.0306
Naive Bayes	Este trabalho	0.9625 ± 0.0414	0.9406 ± 0.0608	0.9031 ± 0.0502	0.9437 ± 0.0652	0.9281 ± 0.0453
	(BORGES, 2006)	0.9035 ± 0.0412	0.9405 ± 0.0285	0.9370 ± 0.0357	0.9435 ± 0.0316	0.9480 ± 0.0187
Árvore de Decisão	Este trabalho	0.95 ± 0.0612	0.9156 ± 0.0663	0.8718 ± 0.0826	0.9062 ± 0.0640	0.9281 ± 0.0495
	(BORGES, 2006)	0.7970 ± 0.0508	0.7260 ± 0.0699	0.7205 ± 0.0754	0.7325 ± 0.1059	0.7890 ± 0.1233

Fonte: Autoria própria

Tabela 16 – Comparação das acurácias na base de dados DLBCL-Outcome

DLBCL-Outcome						
Algoritmo	**Método**	**Número de atributos**				
		3,00%	10,00%	20,00%	25,00%	50,00%
k-NN (k = 1)	Este trabalho	0.8175 ± 0.0634	0.8075 ± 0.0749	0.815 ± 0.0657	0.7775 ± 0.0739	0.5325 ± 0.1145
	(BORGES, 2006)	0.5710 ± 0.0545	0.5450 ± 0.0171	0.5370 ± 0.0268	0.5310 ± 0.0290	0.5163 ± 0.0210
k-NN (k = 3)	Este trabalho	0.8150 ± 0.0634	0.8475 ± 0.0749	0.8425 ± 0.0657	0.8375 ± 0.0739	0.4750 ± 0.1145
	(BORGES, 2006)	0.5554 ± 0.0489	0.5657 ± 0.0480	0.5790 ± 0.0304	0.5757 ± 0.0234	0.5673 ± 0.0252
k-NN (k = 5)	Este trabalho	0.8375 ± 0.0567	0.845 ± 0.0522	0.8300 ± 0.0640	0.855 ± 0.0589	0.4475 ± 0.1089
	(BORGES, 2006)	0.5537 ± 0.0699	0.5314 ± 0.0386	0.5424 ± 0.0287	0.5447 ± 0.0392	0.5223 ± 0.0273
k-NN (k = 7)	Este trabalho	0.84 ± 0.0624	0.845 ± 0.0567	0.8150 ± 0.0549	0.855 ± 0.0630	0.4275 ± 0.1089
	(BORGES, 2006)	0.5223 ± 0.0566	0.4983 ± 0.0454	0.5157 ± 0.0247	0.5010 ± 0.0314	0.5050 ± 0.0330
SVM	Este trabalho	0.83 ± 0.0599	0.85 ± 0.0632	0.825 ± 0.0622	0.8475 ± 0.0621	0.455 ± 0.1105
	(BORGES, 2006)	0.5277 ± 0.0715	0.5100 ± 0.0281	0.5200 ± 0.0161	0.5230 ± 0.0274	0.5200 ± 0.0225
Naive Bayes	Este trabalho	0.8075 ± 0.0675	0.8575 ± 0.0637	0.8225 ± 0.0558	0.8375 ± 0.0668	0.5275 ± 0.1166
	(BORGES, 2006)	0.3973 ± 0.0653	0.3880 ± 0.0463	0.3907 ± 0.0356	0.3763 ± 0.0446	0.3660 ± 0.0292
Árvore de Decisão	Este trabalho	0.785 ± 0.0988	0.84 ± 0.0699	0.8 ± 0.0836	0.8150 ± 0.0807	0.5 ± 0.0880
	(BORGES, 2006)	0.4570 ± 0.0859	0.4837 ± 0.0933	0.4897 ± 0.0846	0.4683 ± 0.1491	0.4770 ± 0.0885

Fonte: Autoria própria

Tabela 17 – Comparação das acurácias na base de dados DLBCL-Tumor

DLBCL-Tumor						
Algoritmo	**Método**	**Número de atributos**				
		3,00%	10,00%	20,00%	25,00%	50,00%
k-NN (k = 1)	Este trabalho	0.9096 ± 0.0445	0.9115 ± 0.0403	0.875 ± 0.0638	0.9326 ± 0.0514	0.9173 ± 0.0547
	(BORGES, 2006)	0.8784 ± 0.0230	0.8773 ± 0.0221	0.8786 ± 0.0157	0.8798 ± 0.0210	0.8817 ± 0.0138
k-NN (k = 3)	Este trabalho	0.9076 ± 0.0445	0.9230 ± 0.0403	0.8769 ± 0.0638	0.9134 ± 0.0514	0.8903 ± 0.0547
	(BORGES, 2006)	0.8687 ± 0.0179	0.8628 ± 0.0177	0.8689 ± 0.0151	0.8741 ± 0.0234	0.8651 ± 0.0093
k-NN (k = 5)	Este trabalho	0.8961 ± 0.0456	0.8826 ± 0.0576	0.8692 ± 0.0681	0.9 ± 0.0428	0.8557 ± 0.0568
	(BORGES, 2006)	0.8853 ± 0.0178	0.9037 ± 0.0162	0.9048 ± 0.0172	0.9012 ± 0.0124	0.9097 ± 0.0105
k-NN (k = 7)	Este trabalho	0.8903 ± 0.0443	0.8634 ± 0.0693	0.8346 ± 0.0852	0.8980 ± 0.0574	0.8365 ± 0.0728
	(BORGES, 2006)	0.9032 ± 0.0223	0.8998 ± 0.0193	0.9198 ± 0.0233	0.9152 ± 0.0218	0.9113 ± 0.0155
SVM	Este trabalho	0.9057 ± 0.0462	0.9076 ± 0.0636	0.8788 ± 0.0712	0.9307 ± 0.0510	0.825 ± 0.0937
	(BORGES, 2006)	0.9587 ± 0.0140	0.9611 ± 0.0127	0.9679 ± 0.0112	0.9650 ± 0.0129	0.9684 ± 0.0099
Naive Bayes	Este trabalho	0.9346 ± 0.0472	0.9134 ± 0.0542	0.8961 ± 0.0456	0.9461 ± 0.0392	0.9384 ± 0.0372
	(BORGES, 2006)	0.8855 ± 0.0197	0.8857 ± 0.0231	0.88,84 ± 0.0169	0.8793 ± 0.0310	0.8877 ± 0.0141
Árvore de Decisão	Este trabalho	0.9076 ± 0.0713	0.9057 ± 0.0724	0.8826 ± 0.0478	0.9596 ± 0.0462	0.9230 ± 0.0530
	(BORGES, 2006)	0.8336 ± 0.0566	0.8584 ± 0.0630	0.8521 ± 0.0627	0.8286 ± 0.0686	0.8661 ± 0.0446

Fonte: Autoria própria