Victor De Cia Costa

Representação de Conjuntos de Dados utilizando Redes Neurais Artificiais

São José dos Campos, SP

Victor De Cia Costa

Representação de Conjuntos de Dados utilizando Redes Neurais Artificiais

Trabalho de conclusão de curso apresentado ao Instituto de Ciência e Tecnologia – UNIFESP, como parte das atividades para obtenção do título de Bacharel em Ciência da Computação.

Universidade Federal de São Paulo – UNIFESP

Instituto de Ciência de Tecnologia

Bacharelado em Ciência da Computação

Orientador: Prof. Dr. Marcos Gonçalves Quiles

São José dos Campos, SP

Junho de 2015

Victor De Cia Costa

Representação de Conjuntos de Dados utilizando Redes Neurais Artificiais

Trabalho de conclusão de curso apresentado ao Instituto de Ciência e Tecnologia – UNIFESP, como parte das atividades para obtenção do título de Bacharel em Ciência da Computação.

Trabalho aprovado em 22 de Dezembro de 2015:

Prof. Dr. Marcos Gonçalves Quiles
Orientador

Professor
Convidado 1

Professor
Convidado 2

Professor
Convidado 3

São José dos Campos, SP
Junho de 2015

Agradecimentos

Agradeço ao meu orientador Prof. Marcos Gonçalves Quiles pelo incentivo e atenção durante o desenvolvimento desse projeto. Ao meus colegas de turma, Léo e Amadeus, por sempre colaborarem com ideias e motivarem sempre. Ao meu orientador de Iniciação Científica, Prof. Vinícius Veloso, pelos dois anos de trabalhos constantes, conhecimento e paixão pela área. A minha companheira Daiane, por estar presente nos momentos mais difíceis e sempre mostrar motivação, a calma e o apoio para realizar qualquer tarefa. Aos meus pais, por sempre me apoiar nos meus estudos.

"Ciência da Computação está tão
relacionada aos computadores quanto
a Astronomia aos telescópios, Biologia
aos microscópios, ou Química aos tubos de ensaio.
A Ciência não estuda ferramentas. Ela estuda
como nós as utilizamos, e o que descobrimos com elas."
(Edsger Dijkstra)

Resumo

A geração de uma rede que represente fielmente dados de uma base é um dos passos principais ao se utilizar técnicas baseadas em redes. Podemos agrupar a forma de se gerar uma rede a partir de um conjunto de dados nas seguintes categorias: redes totalmente conectadas e redes com conexões esparsas. Porém, a quantidade e a dependência dos parâmetros utilizados para se gerar essas redes podem influenciar drasticamente o resultado do algoritmo de classificação. Além disso, muitos dos parâmetros são dependentes do conjuntos de dados e necessitam de ajustes customizados para a produção de bons resultados de classificação. Nesse contexto, visando reduzir a quantidade de parâmetros envolvidos na geração da rede, são investigados o uso de Autoencoders, uma rede neural artificial construída sobre a arquitetura de uma rede neural Multi-Layer Perceptron, que permite a construção de uma representação eficiente e compacta de um conjunto de dados. Essa representação pode ser utilizada para diversos propósitos, como por exemplo a redução de dimensionalidade, normalização dos atributos, obtenção de atributos não correlacionados, etc.

Palavras-chaves: autoencoder. geração de rede. representação de dados. multi-layer perceptron

Abstract

The generation of a network of data that faithfully represents a base is one of the main steps when using networks based techniques. We can group the way to generate a network from a dataset in the following categories: fully connected networks and networks with sparse connections. However, the number and the dependency of the parameters used to generate these networks can dramatically influence the results of the classification algorithm. Furthermore, many of the parameters are dependent on the data sets and require customized settings to produce good results classification. In this context, to reduce the number of parameters involved in the generation of the network are investigated using Autoencoders, a neural network constructed on a neural network architecture Multi-Layer Perceptron, which permits the construction of an efficient and compact representation of a data set. This representation can be used for various purposes, such as the dimensionality reduction, normalization of attributes, obtain non-correlated attributes, etc.

Key-words: autoencoder. network generation. data representation. multi-layer perceptron

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

PCA	Análise de Componentes Principais
*k*NN	k - vizinhos mais próximos
MLP	Multi layer perceptron
RNA	Redes neurais artificiais
PMC	Perceptron multicamadas
L1	Camada de entrada
HM	Função de similaridade Hein e Maier
LGC	Local global consistency
UCI	UCI Machine Learning Repository

Lista de símbolos

Γ	Letra grega Gama
Λ	Lambda
ζ	Letra grega minúscula zeta
∂	Derivada
η	Letra grega eta
δ	Letra grega minúscula delta
Δ	Letra grega maiúscula delta
∇	Nabla
α	Letra grega minúscula alpha
ϵ	Letra grega épsilon
$\in$	Pertence
κ	Letra grega kappa
χ	Letra grega chi
Ψ	Letra grega psi
$\bigcap$	Intersecção
σ	Sigma
μ	Letra grega mu

Sumário

1 Introdução

Dados do mundo real, tais como sinais de voz, fotografias digitais, conteúdo de redes sociais, fotografias de satélite, dentre muitas outras, geralmente possuem uma alta dimensionalidade. Essa dimensionalidade dos dados é expressa como o número de variáveis medidas a cada observação (FODOR, 2002), ou seja, a quantidade de atributos daquele conjunto de dados. A fim de lidar com dados do mundo real de forma adequada, a sua dimensionalidade precisa de ser reduzida. O processo da redução dessa dimensionalidade é a transformação de dados de alta dimensionalidade em uma representação significativa dos dados originais de dimensão reduzida. Idealmente, a representação reduzida deve ter uma dimensionalidade que corresponde à dimensão intrínseca dos dados. A dimensionalidade intrínseca dados é o número mínimo de parâmetros necessários para representar as propriedades observadas dos dados (FUKUNAGA, 2013).

A redução de dimensionalidade é importante em muitos domínios, uma vez que atenua a chamada maldição da dimensionalidade e outras propriedades indesejáveis dos espaços de elevadas dimensões (JIMENEZ; LANDGREBE et al., 1998). Um dos problemas de conjuntos de dados de elevada dimensão é que, em muitos casos, nem todas os atributos medidos são relevantes para entender o problema fundamental de uma determinada aplicação.

Como resultado, a redução de dimensionalidade facilita, entre outros, a classificação, visualização e compressão de dados de elevada dimensionalidade. Tradicionalmente, a redução de dimensionalidade é realizada utilizando algumas técnicas, tal como a linear Análise de Componentes Principais (PCA) (WOLD; ESBENSEN; GELADI, 1987).

A geração de uma rede que represente de forma fiel os dados de uma base é um dos passos fundamentais ao utilizar técnicas baseadas em redes (ZHU; LAFFERTY; ROSENFELD, 2005) como pode ser exemplificada na Figura 1.

Segundo Zhu (ZHU; LAFFERTY; ROSENFELD, 2005) existem diversas formas de se gerar uma rede a partir de um conjunto de dados, sendo que estas podem ser agrupas nas seguintes categorias: redes totalmente conectadas e redes com conexões esparsas.

Nas redes totalmente conectadas todos os vértices (exemplos da base) são conectados entre si e o peso das arestas é utilizado para representar a similaridade (distância/relação) entre os dados. Como vantagem desta forma de representação está a possibilidade da utilização da derivada dos pesos da rede na construção de técnicas de aprendizagem. Por outro lado, uma das desvantagens está no alto custo computacional envolvido no tratamento de redes densamente conectadas.

As redes com conexões esparsas podem ser obtidas através das seguintes metodolo-

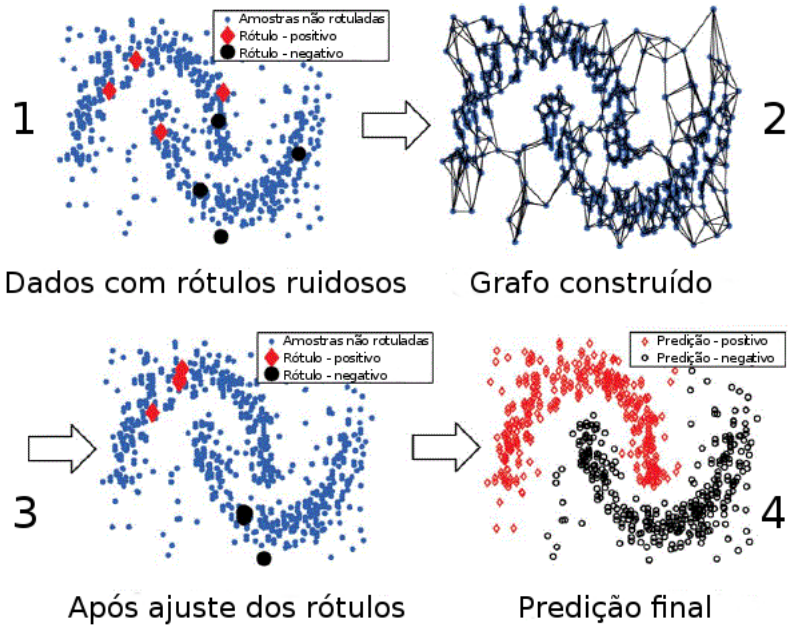

Figura 1 – Representação de dados através de redes. Figura retirada de (LIU; WANG; CHANG, 2012).

gias: 1) utilizar um limiar de corte epsilon no qual dois vértices são conectados apenas quando a similaridade entre eles é superior a este limiar; 2) conexão dos vértices a um número pré-estabelecido de vizinhos (denominado k-vizinhos mais próximos - *k*NN); e 3) utilização conjunta das metodologias 1) e 2). Uma alternativa à técnica dos *k*NN, denominada *b-matching*, foi proposta em (JEBARA; SHCHOGOLEV, 2006; JEBARA; WANG; CHANG, 2009). Esta técnica garante que os vértices da rede gerada terão exatamente k conexões, o que não é garantido pela técnica *k*NN. Outras formas para transformar a matriz de adjacência numa matriz simétrica é proposta em (LIU; CHANG, 2009). Por fim, é importante destacar que, na representação esparsa, as arestas podem ou não assumir pesos.

Além dos métodos apresentados acima, o conhecimento prévio do domínio também é um importante fator que pode ser levado em consideração durante o processo de construção da rede, pois, uma vez que algumas propriedades da base são conhecidas, estas podem colaborar na forma como os pontos são conectados. Além disso, com base nos vértices (dados) previamente rotulados, pode-se estabelecer novas conexões na rede ligando tais vértices (QUILES et al., 2010).

Em (SOUSA, 2013), foi mostrado que a quantidade e a dependência dos parâmetros utilizados para gerar uma rede podem influenciar drasticamente o resultado do algoritmo de classificação. Além disso, muitos dos parâmetros são dependentes do conjunto de dados e necessitam de ajustes customizados para a produção de bons resultados de classificação.

Nesse contexto, visando reduzir a quantidade de parâmetros envolvidos na geração da rede, representações alternativas via autoencoders (OLSHAUSEN et al., 1996; RUMELHART;

HINTON; WILLIAMS, 1985), foram investigadas. O autoencoder é uma rede neural, geralmente construído sobre a arquitetura de uma rede neural MLP (Multi-Layer Perceptron) (KUBAT, 1999), que permite a construção de uma representação eficiente e compacta de um conjunto de dados. Essa representação pode ser utilizada para diversos propósitos, como por exemplo: a redução dimensionalidade, normalização dos atributos, obtenção de atributos não correlacionados, etc. Neste trabalho, essa representação foi considerada como uma forma de pré-processamento dos dados visando a redução a quantidade de parâmetros necessários para a geração da rede.

O restante desse documento está organizado da seguinte maneira. O Capítulo 2 denota uma revisão bibliográfica sobre redes neurais artificiais e métodos de representação de dados através de grafos. O Capítulo 3 apresenta a metodologia aplicada no uso de redes neurais para a geração de novas representações dos dados, na geração da rede com as novas representações e nos experimentos computacionais realizados. O Capítulo 4 expõe os experimentos realizados, os resultados obtidos e sua análise. Por fim, as conclusões são apresentadas no Capítulo 5.

1.1 Objetivos

Neste trabalho uma técnica de redução de atributos, ou dimensionalidade, que utiliza uma rede neural artificial com uma arquitetura de Autoencoder é usada para indentificar correlações lineares ou não-lineares entre quaisquer atributos de uma determinada base de dados. O foco da aplicação é no pré-processamento dos dados.

Em específico, pretende-se eliminar a necessidade do refinamento de parâmetros utilizados por outros métodos encontrados na literatura e comumente empregados nessa etapa, reduzindo apenas a um parâmetro. Visando uma alta representatividade do conjunto de dados.

2 Revisão Bibliográfica

Foram realizadas revisões a respeito da construção e funcionamento das redes neurais artificiais, das técnicas utilizadas para gerar grafos a partir de conjuntos de dados utilizadas na comparação da abordagem utilizada neste trabalho e a técnica estudada neste trabalho com base em redes neurais.

2.1 Redes Neurais Artificiais

A criação, desenvolvimento e motivação das redes neurais arificiais (RNA) tem início nas áreas de pesquisas que simulam as capacidades cognitivas do ser humano, projetando assim máquinas capazes de exibir um comportamento similar como as reações de um ser humano. Assim, surgiu a tentativa de copiar a estrutura e o funcionamento do próprio cérebro em um ambiente computacional. A pesquisa então, tenta entender o funcionamento da inteligência residente nos neurônios e mapeá-la para uma estrutura artificial.

As pesquisas sobre as RNA começaram por (MCCULLOCH; PITTS, 1943) em um trabalho pioneiro, consistindo num estudo sobre o comportamento do neurônio biológico objetivando a criação do modelo matemático correspondente, com a interpretação do funcionamento do neurônio como sendo um circuito binário. O modelo matemático do neurônio de McCulloch e Pittts é capaz de separar duas entradas booleanas, todavia não é possível treinar os neurônios, pois não possuem parâmetros livres. O neurônio artificial criado representado pela Figura 2.

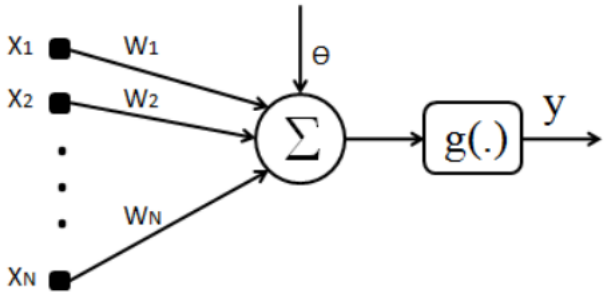

Figura 2 – Modelo do neurônio artificial.

Em uma análise matemática descrita por (SELLI; JR, 2007), as RNA podem ser definidas como um mapeamento não linear de um vetor de espaço de entrada para um vetor de espaço de saída, que pode ser feito através de camadas de funções de ativação ou neurônios, nos quais coordenadas de entrada são somadas de acordo com o valor de seus respectivos pesos e um

"bias" para produzir uma saída simples, ativada ou não, de acordo com o respectivo nível de disparo.

2.1.1 Perceptron

Uma grande importância nas pesquisas com redes neurais foi a concepção do modelo Perceptron, Figura 3, por (ROSENBLATT, 1958), que consistia em um modelo de entradas conectadas a uma única camada de neurônios tal como o modelo de McCulloch e Pitts, acrescidos de sinapses ajustáveis. (ROSENBLATT, 1958) demonstrou que essas redes poderiam ser treinadas para classificar padrões de classes linearmente separáveis.

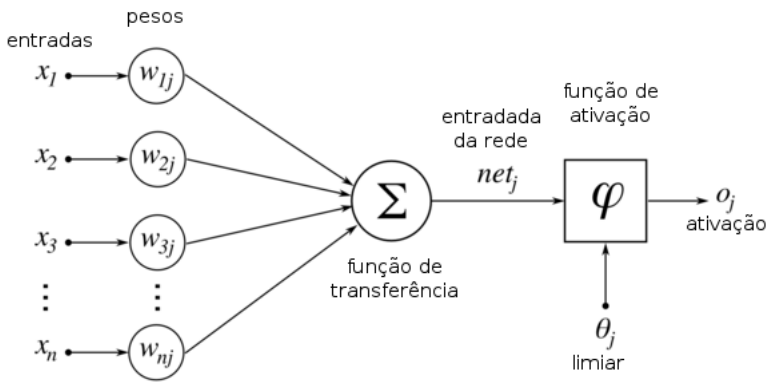

Figura 3 – Modelo do neurônio *Perceptron*.

2.1.2 Rede Perceptron Multicamadas e Algoritmo de Treinamento Back-propagation

As redes Perceptron Multicamadas (PMC), são caracterizadas pela arquitetura de multi camadas de neurônios de tipo Perceptron, Figura 4. O incremento de mais camadas neurais implica em um aumento na capacidade de processamento não linear e generalização da rede neural. Cada camada pode conter inúmeros neurônios.

A rede PMC é composta, em sua topologia, por uma camada de entrada e uma camada de saída, sendo que entre essas camadas podem existir inúmeras camadas intermediárias, chamadas de camadas ocultas. Sendo que as saídas dos neurônios de cada camada são as entradas dos neurônios da camada sub-sequente, desde a camada de entrada, até a camada de saída. Neste tipo de rede não há retro-alimentações, pois a propagação do processamento neural é unidirecional.

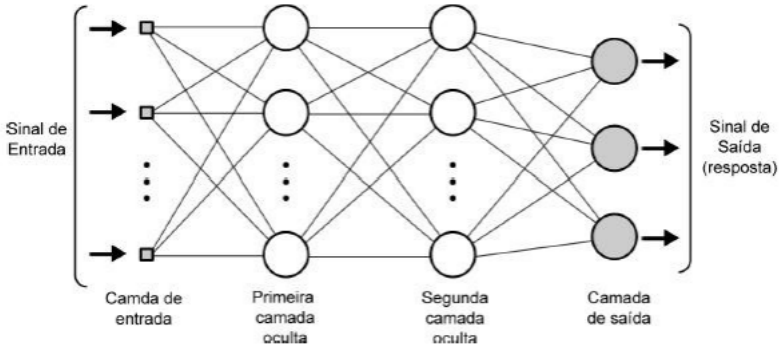

Figura 4 – Modelo da Rede Perceptron Multicamadas.

2.1.2.1 *Backpropagation*

As redes PMC possuem treinamento supervisionado, sendo que o algoritmo de treinamento visa ajustar os pesos da rede para, a partir de um conjunto conhecido de vetores de entrada para obter da rede um conjunto de saídas desejadas. Sendo assim, para treinamento de redes PMC é utilizado o treinamento supervisionado através do algoritmo *backpropagation*, onde é apresentado à rede um conjunto de treinamento da rede neural, com pares de entrada e saída conhecidos. O vetor de entrada é propagado camada a camada, até a camada de saída. O vetor de saída da rede é então comparado ao vetor de saída desejado, sendo a diferença entre ambas saídas (calculada e desejada) o erro de saída da rede. O funcionamento do algoritmo é explicado a seguir.

Na Figura 5 tem-se a seguinte simbologia:

- **N**: Número de pontos do vetor de entrada X;
- X: Vetor de entrada da rede: $X = [-1, x_1, x_2, ..., x_N]^T$;
- Wl_{ij}: Valor do peso sináptico conectado ao ***j**-ésimo* neurônio da camada **l** ao ***i**-ésimo* neurônio da camada (**l-1**); Ressalta-se que nesta convenção o valor do peso WI_{j0}, para todos os valores de **l** e **j**, tem-se o valor do *bias* (limiar) de cada neurônio;
- Il_j: Valor da entrada ponderada do ***j**-ésimo* neurônio da camada **l**.

Assim, generalizando para ***l*** camadas, teremos:

$$Il_j = \sum_{i=0}^{N[l-1]} Wl_{ji} \cdot y[l-1]_i;\ para\ j = 1..Nl \tag{2.1}$$

$$yl_j = g(Il_j);\ para\ j = 1..Nl \tag{2.2}$$

$$g():\ Função\ de\ ativação\ do\ neurônio. \tag{2.3}$$

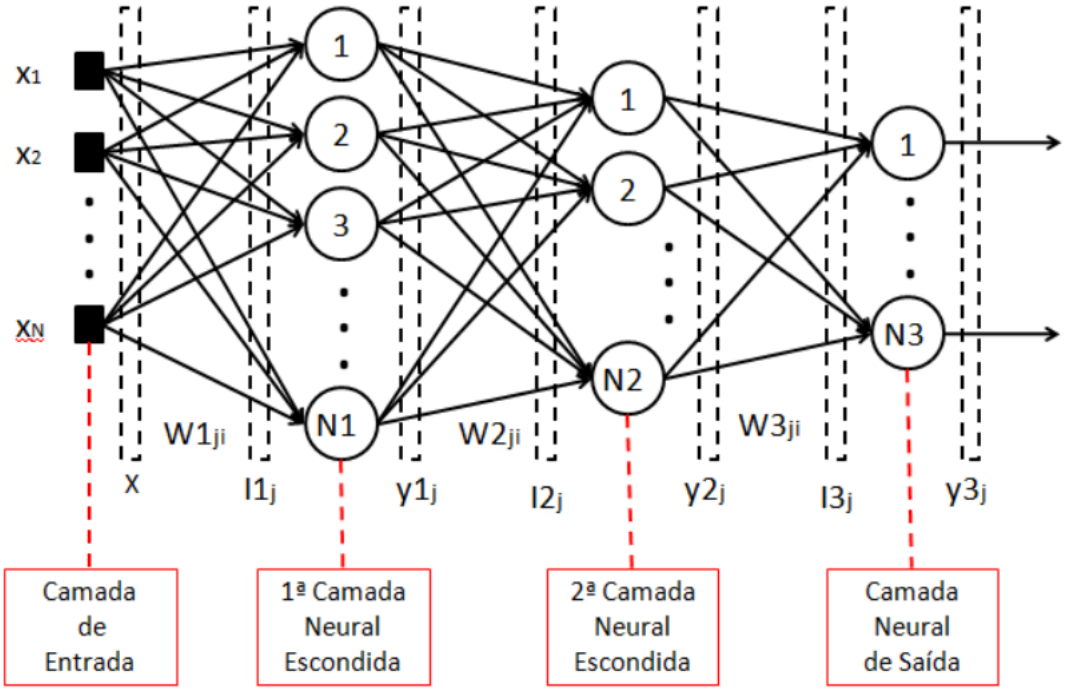

Figura 5 – Perceptron Multicamadas. Figura retirada de (AGUIAR, 2010).

Tem-se algumas funções utilizadas como critério de parada do processo de treinamendo da rede e durante a fase de ajustes de pesos dos neurônios, são elas:

1. Erro Quadrático: fornece o valor da soma dos erros quadráticos do padrão de treinamento de todos os neurônios da camada de saída. Descrito pela equação

$$E(k) = \frac{1}{2}\sum_{j=1}^{Nm}(d_j(k) - ym_j(k))^2, \textit{m é o número de camadas.} \tag{2.4}$$

2. Erro Quadrático Médio: obtido a partir da soma dos erros quadráticos relativos a todos os padrões de entrada utilizados no conjunto de treinamento. Descrito pela equação

$$E_M = \frac{1}{p}\sum_{k=1}^{p} E(k),\ p \textit{ são os pares de vetores de entrada e saída desejadas.} \tag{2.5}$$

Os pesos da camada de saída são ajustados a partir da minimização da função de erro quadrático médio em relação aos pesos $W3_{ij}$. Utilizando a regra de diferenciação em cadeia, tem-se:

$$\nabla E(W3_{ij}) = \frac{\partial E}{\partial W3_{ij}} = -(d_j - y3_j)\cdot g'(I3_j)\cdot y2_i \tag{2.6}$$

Para minimização, o ajuste deve ser realizado oposto ao gradiente do erro $\nabla E(W3_{ij})$.

$$\Delta W3_{ij} = -\eta \cdot \frac{\partial E}{\partial W3_{ij}} = \eta \cdot \delta 3_j \cdot y2_i \tag{2.7}$$

Onde

$$\delta 3_j = (d_j - y3_j) \cdot g'(I3_j) \tag{2.8}$$

. Sendo η a taxa de aprendizagem do algoritmo e $\delta 3_j$ o gradiente local. Assim, teremos:

$$W3_{ij}(t+1) = W3_{ij}(t) + \Delta W3_{ij}(t) = W3_{ij} + \eta \cdot \delta 3_j \cdot y2_i \tag{2.9}$$

Para calcular o ajuste das camadas escondidas, segue com a mesma intuição da camada de saída. Porém a minimização da função de erro quasdrático médio é feita em relação aos pesos $W2_{ij}$, os pesos das camadas escondidas. Sendo assim, teremos, após todas as deduções:

$$\nabla E(W2_{ij}) = \frac{\partial E}{\partial W2_{ij}} = (-\sum_{k=1}^{N3} \delta 3_k \cdot W3_{kj}) \cdot g'(I2_j) \cdot y1_i \tag{2.10}$$

$$\Delta W2_{ij} = -\eta \cdot \frac{\partial E}{\partial W2_{ij}} = \eta \cdot \delta 2_j \cdot y1_i \tag{2.11}$$

Onde:

$$\delta 2_j = (\sum_{k=1}^{N3} \delta 3_k \cdot W3_{kj}) \cdot g'(I2_j) \tag{2.12}$$

$$W2_{ij}(t+1) = W2_{ij}(t) + \Delta W2_{ij}(t) = W2_{ij} + \eta \cdot \delta 2_j \cdot y1_i \tag{2.13}$$

A primeira cada neural também deve ser ajustada, e seu ajuste é feito a partir da minimozação da função de erro quadrático médio em relação aos pesos $W1_{ij}$. Temos, então, após todas as deduções análogas à primeira dedução da camada de saída:

$$\nabla E(W1_{ij}) = \frac{\partial E}{\partial W1_{ij}} = (-\sum_{k=1}^{N2} \delta 2_k \cdot W2_{kj}) \cdot g'(I1_j) \cdot x_i \tag{2.14}$$

$$\Delta W1_{ij} = -\eta \cdot \frac{\partial E}{\partial W1_{ij}} = \eta \cdot \delta 1_j \cdot x_i \tag{2.15}$$

Onde:

$$\delta 1_j = (\sum_{k=1}^{N2} \delta 2_k \cdot W2_{kj}) \cdot g'(I1_j) \tag{2.16}$$

$$W1_{ij}(t+1) = W1_{ij}(t) + \Delta W1_{ij}(t) = W1_{ij} + \eta \cdot \delta 1_j \cdot x_i \tag{2.17}$$

Assim todas as camadas são ajustadas para uma nova entrada, e o processo continúa até o erro quadrático médio ser menor que um certo limiar desejável para a aplicação. O algoritmo *backpropagation* pode ser resumido nos seguintes passos:

Algoritmo 1 *Backpropagation*

função BACKPROPAGATION($X(k) = [-1, x_1, x_2, ..., x_N]^T$)) ▷ com N elementos - e saídas desejadas ($d(k) = [d_1, d_2, ..., d_M]^T$) - com M elementos
    Definir os parâmetros da rede (número de camadas neurais, de neurônios por camada, funções de ativação, parâmetros de aprendizagem, funções de erro;
    Inicializar as matrizes de pesos Wl_{ij};
    enquanto $|E_M(t) - E_M(t-1)| < \varepsilon$ **faça**
        para Para cada par $X(k)$, $d(k) \leftarrow$ **até faça**
            Passo *foward*
            Passo *backward*
        fim para
        Calcular $E_M(t)$
    fim enquanto
fim função

2.1.2.2 Fuções de ativação

Como mostrado anteriormente, cada neurônio que compõe a RNA possui uma função de ativação, no qual converte o valor do somatório dos pesos com as entradas do neurônio para uma faixa de valores que depende da função escolhida. A função de ativação dos neurônios utilizados é um dos parâmetros a ser definido pelo projetista e tem grande influência no desempenho da RNA.

A escolha de qual tipo de função será utilizada pelos nodos está intimamente relacionada à natureza do problema, podendo acarretar um aumento ou diminuição na velocidade de convergência (KORDOS; DUCH, 2004).

Existem muitos tipos de função de ativação, sendo uns mais simples e outros mais complexos. Geralmente, utiliza-se um mesmo tipo de função de ativação para todos os nodos de uma rede neural. Porém, existem outras abordagens, muito pouco exploradas, que definem diferentes funções de ativação para cada neurônio ou para camada da rede (DUCH; JANKOWSKI, 2001).

As funções de ativação mais conhecidas são a função linear (Figura 6) descrita pela equação (2.18), a função sigmóide logística (Figura 7) descrita pela equação (2.19) e a função sigmóide tangente hiperbólica (Figura 8) descrita pela equação (2.20). As duas últimas são as mais utilizadas como funções de ativação dos nodos, principalmente em redes treinadas com algoritmos baseados em gradiente descendente, que exigem que as funções sejam contínuas para o cálculo de derivadas utilizadas pelo algoritmo (KUBAT, 1999).

As funções de ativação apresentadas são formalizadas através das equações a seguir. Sendo α a ativação de um neurônio, dada por $\alpha_j = \sum_j W_{ij} O_i$, onde O_i é o sinal propagado pelo neurônio i, θ é um número real e b é o parâmetro de inclinação da curva sigmóide. Temos:

$$f(\alpha) = \theta\alpha \tag{2.18}$$

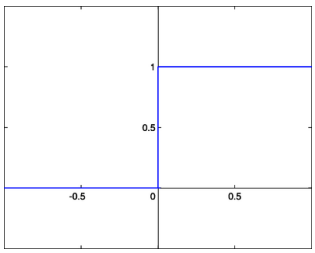

Figura 6 – Gráfico da função linear.

$$f(\alpha) = \frac{1}{1 + exp(-b\alpha)} \tag{2.19}$$

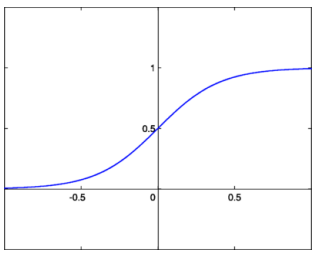

Figura 7 – Gráfico da função sigmóide logística.

$$f(\alpha) = \frac{1 - exp(-b\alpha)}{1 + exp(-b\alpha)} \tag{2.20}$$

2.1.3 Auto encoders

A arquitetura comum de um autoencoder consiste em uma rede neural com 3 camadas, sendo uma camada de entrada, uma camada oculta e uma camada de saída. A camada de entrada (L1) recebe os atributos que compõe cada exemplo do conjunto de dados, ou seja, o número de entradas é igual ao número de atributos do conjunto. Por exemplo, para a ilustração apresentada na Tabela 1, a camada de entrada é composta por 4 unidades.

A camada oculta representa o extrator de características, ou os novos atributos gerados para o conjunto de dados a partir do processo de treinamento. A camada de saída contem

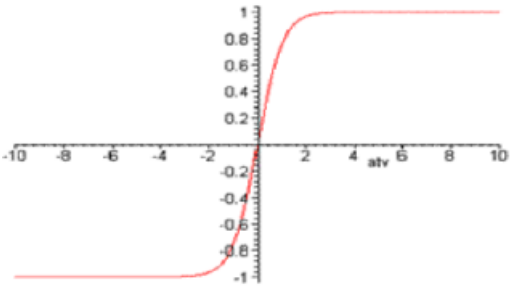

Figura 8 – Gráfico da função sigmóide tangente hiperbólica.

Tabela 1 – Exemplo de tabela atributo-valor com amostras do conjunto de dados Iris da UCI.

sepal lenght	sepal width	petal lenght	petal width	Class
5,1	3,5	1,4	0,2	Iris-setosa
7,0	3,2	4,7	1,4	Iris-versicolor
6,3	3,3	6,0	2,5	Iris-virginica

o mesmo número de neurônios da camada de entrada. Ou seja, a rede aprende a mapear os atributos em si mesmos. A Figura 9 ilustra a arquitetura básica de um autoencoder.

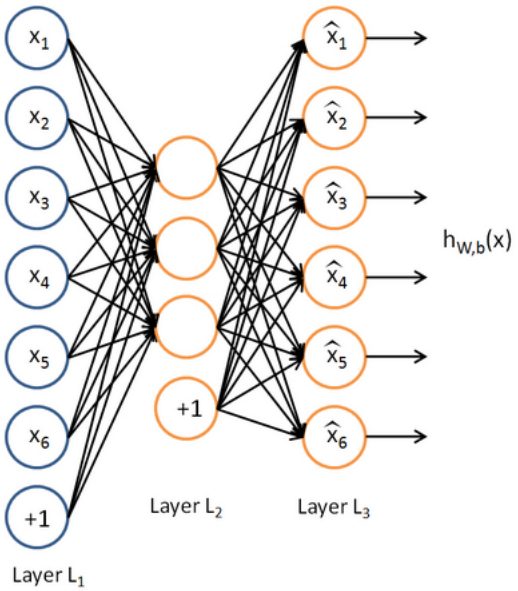

Figura 9 – Arquitetura de um autoencoder com uma única camada oculta. Figura retirada de (KUPREL,).

É importante observar que o autoencoder pode conter mais do que uma única camada oculta, inclusive, várias camadas ocultas podem ser empilhadas dando origem aos denominados *deep* autoencoders.

Autoencoders podem ser utilizados em diversas aplicações, (BETECHUOH; MARWALA; TETTEY, 2006) mostrou seu desempenho aplicado na classificação de vírus HIV combinando

autoencoders com algoritmos genéticos. O método proposto quando comparado com redes neurais *feedfoward* convencionais, resulta numa acurácia 8% maior.

Em (KAMYSHANSKA; MEMISEVIC, 2013), é mostrado como um autoencoder pode atribuir pontuações significativas para os dados de forma independente do processo de formação e sem referência a qualquer modelo probabilístico, interpretando-a como um sistema dinâmico. Certos autoencoders podem atribuir um *score* não normalizado aos dados. *Score* é a medida de quão bem o autoencoder pode representar o conjunto de dados. As pontuações são comumente calculada usando critérios de formação que se relacionam com o autoencoder para um modelo probabilístico, como a Máquina de Boltzmann Restrita, proposta por (SMOLENSKY, 1986).

Diferentemente da PMC, o autoencoder funciona com um aprendizado não-supervisionado, podendo assim ser aplicado em uma gama maior de aplicações.

2.2 Métodos de aprendizagem

Nesta seção são apresentados os métodos de aprendizagem utilizados no aprendizado de máquina, em contexto com a aplicação em técnicas de redes neurais. Esses métodos são escolhidos primeiramente de acordo com o conjunto de dados que será utilizado, o problema que se está tentando solucionar e a técnica de aprendizado de máquina utilizada para resolver o problema. Existem três tipos de aprendizagem, são elas, supervisionadas, não-supervisionadas e semi-supervisionadas.

2.2.1 Supervisionado

Neste tipo de aprendizagem existe um especialista no conjunto de dados que avalia a resposta da rede ao padrão atual de entradas, isto é, ele vai definir o rótulo do exemplo de entrada de acordo com sua experiência do conjunto de dados. As alterações dos pesos são calculadas de forma a que a resposta da rede tenda a coincidir com a do especialista. É o tipo de aprendizagem normalmente utilizada para treinar *feedforward networks*, além de que é relativamente fácil avaliar o desempenho da rede para um determinado estado do sistema, bastando verificar se a saída da rede é igual a saída esperada definida anteriormente.

2.2.2 Não-Supervisionado

Nesta forma de aprendizagem não existe especialista. A rede tem de descobrir sozinha relações, padrões, regularidades ou categorias nos dados que lhe vão sendo apresentados e codificá-las nas saídas.

2.2.3 Semi-Supervisionado

Esta forma combina os dois métodos anteriores da seguinte forma. Apenas uma parte dos dados é avaliada por um especialista e é rotulada, enquanto a outra parte não é. Portanto, pode ser utilizado tanto em tarefas de classificação, quando os exemplos rotulados são utilizados no processo de rotulação de novos exemplos, quanto em tarefas de agrupamento, sendo os exemplos rotulados responsáveis por auxiliar o processo de formação de grupos.

A ideia dessa forma de aprendizado é então utilizar os exemplos rotulados para se obter informações sobre o problema e utilizá-las para guiar o processo de aprendizagem a partir dos exemplos não rotulados.

2.3 Representação de dados através de grafos

Nesta seção são apresentadas formas da construção de grafos a partir de um conjunto de dados, possuindo dados rotulados e não-rotulados, que posteriormente serão utilzadas para a comparação e estudo com a técnica desenvolvida.

2.3.1 Notação

São definidas as notações da representação do conjunto de dados utilizados e da rede gerada a partir desses dados. Considere um exemplo de treinamento de algum conjunto de dados

$$X := \{x_i\}_{i=1}^{n} \subset \mathbb{R}^d \tag{2.21}$$

no qual os primeiros *l* exemplos são rotulados, isto é, x_i possui o rótulo $y_i \in \mathbb{N}_c$ onde

$$\mathbb{N}_p := \{i \in \mathbb{N}^* | 1 \leq i \leq p\} \tag{2.22}$$

com $p \in \mathbb{N}^*$ e c sendo o número de classes.

Seja $u := n - l$ a quantidade de exemplos não rotulados e $Y \in \mathbb{B}^{n \times c}$ a matriz de rótulos no qual $Y_{ij} = 1$ se e somente se, x_i possui um rótulo $y_i = j$.

Considerando um grafo não direcionado $G := (\chi, \epsilon)$ onde cada x_i é um nó de G. Então, para gerar uma matriz esparça de pesos $W \in \mathbb{R}^{n \times n}$ de G é usada uma função de similaridade

$$\kappa : \mathbb{R}^d \times \mathbb{R}^d \to R \tag{2.23}$$

para determinar os pesos entre x_i e cada $x_j \in \mathbb{N}_i$.

Seja $F \in \mathbb{R}^{n \times c}$ a saída de um algoritmo semi-supervisionado baseado em grafos. Todas as matrizes podem ser subdivididas em matrices de dados rotulados e dados não-rotulados. As matrizes F e Y são subdivididas em duas submatrizes, enquanto todas as outras são divididas em quatro submatrizes matrizes. Por exemplo:

$$W := \begin{bmatrix} W_{LL} & W_{LU} \\ W_{UL} & W_{UU} \end{bmatrix} F := \begin{bmatrix} F_L \\ F_U \end{bmatrix}$$

onde

$$W_{LL} \in \mathbb{R}^{l \times l} \tag{2.24}$$

$$e\, Y_L \in \mathbb{B}^{l \times c} \tag{2.25}$$

são as submatrizes de W e Y, respectivamente, para exemplos rotulados, e assim em diante. Por definição, Y_U é uma matriz nula de dimenções $u \times c$. O foco são problemas multi-classe; consequentemente, $Y_L 1_c = 1_l$.

A seguir são definidos os métodos de geração de grafos de adjacências a partir de um conjunto de dados. O processo de construção dos grafos de adjacência geram um grafo G (ou uma matriz de adjacência A) de χ usando uma função de distância

$$\Psi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R} \tag{2.26}$$

Seja $\Psi \in \mathbb{R}^{n \times n}$ uma matriz de distâncias no qual

$$\Psi_{ij} := \Psi(x_i, x_j) \tag{2.27}$$

$$e\, A \in \mathbb{B}^{n \times n} \tag{2.28}$$

sendo uma matriz de adjacência no qual $A_{ij} = 1$ se e somente se $x_j \in N_i$. Os métodos estudados nessa seção são utilizados frequentemente para a geração de grafos, encontrados na literatura.

2.3.2 *K-nearest neighbors*

***k*-nearest neighbors (*k*NN).** Foi descrito por (DUDANI, 1976) da seguinte forma, existirá uma aresta de x_i a x_j se e somente se x_j é um dos k exemplos mais próximos de x_i. A medida de distância para encontrar os exemplos mais próximos podem variar. A mais comum utilizada é a euclideana, porém a função de distância tem grande impacto na técnica, uma vez

que não é muito eficiente quanto se trata de dados multi-dimensionais. Matematicamente, o grafo *k*NN pode ser descrito como um problema de otimização:

$$\min_{\widehat{A}\in\mathbb{B}} tr\left(\widehat{A}^T\Psi\right) = \sum_{i=1}^{n}\sum_{j=1}^{n}\widehat{A}_{ij}\Psi(x_i,x_j) \tag{2.29}$$

$$s.t.\widehat{A}1_n = k1_n, tr\left(\widehat{A}\right) = 0 \tag{2.30}$$

no qual 1_n é um vetor *n*-dimensional de entrada, o sobrescrito ^{T} denota a transposição de matrizes ou vetores, e $tr(.)$ é utilizado como operador de traço da matriz, isto é,

$$tr(X) = \sum i = 1^p X_{ii}, \forall X \in \mathbb{R}^{p\times p} \tag{2.31}$$

2.3.3 Corte epsilon

Corte epsilon. Formulada por (HAUSSLER; WELZL, 1987), também é chamada de ϵ-net. Seja X um conjunto e R um subconjunto de X; tal par é chamado de um espaço ou intervalo hipergrafo, e os elementos de R são chamados intervalos ou hiperarestas. Um ϵ-net de um subconjunto P de X é um subconjunto N de P tal que qualquer intervalo $r \in R$ com $|r \bigcap P| \geqslant \epsilon|P|$ intersecta N (HAUSSLER; WELZL, 1987). Resumidamente, qualquer intervalo que intersepta pelo menos uma proporção ϵ dos elementos de P devem também interceptar a rede ϵ-net N.

2.3.4 Geração de matrizes ponderadas e funções de similaridade

Dada uma matriz de adjacência A, podemos gerar uma matriz esparsa ponderada W utilizando a função de similaridade $\kappa : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$. Existem várias abordagens para a geração da matriz W. Foram escolhidas duas formas, o RBF kernel e a função de similaridade de (HEIN; MAIER, 2007)

RBF kernel. O RBF kernel ou Gaussiano, computa a similaridade entre x_i e x_j atravéz de uma função:

$$\kappa_\sigma(x_i,x_j) = exp\left(-\frac{\Psi^2(x_i,x_j)}{2\sigma^2}\right) \tag{2.32}$$

onde $\sigma \in \mathbb{R}^*_+$ é o parâmetro do kernel.

Função de similaridade de Hein & Maier (HEIN; MAIER, 2007)**(HM).** A função de similaridade HM calcula a similaridade entre x_i e x_j atravéz da fórmula:

$$\kappa_\sigma(x_i, x_j) = exp\left(-\frac{\Psi^2(x_i, x_j)}{(max\Psi(x_i, x_{i_k}), \Psi(x_j, x_{j_k}))^2}\right) \tag{2.33}$$

Esse método pode ser visto como um kernel RBF com um tamanho adaptativo local de kernel, para cada par de vizinhos ordenados (x_i, x_j). Nós podemos calcular o tamanho do kernel adaptativo σ_{ij}^k como:

$$\sigma_{ij}^k = \frac{\sqrt{2}}{2} max\left\{\Psi(x_i, x_{i_k}), \Psi(x_j, x_{j_k})\right\} \tag{2.34}$$

2.3.5 Técnicas Híbridas

Além dos métodos apresentados acima, a rede também pode ser gerada combinando-se abordagens, por exemplo: pode-se utilizar o corte-epsilon para gerar uma rede inicial e o *k*NN como uma ferramenta de pós-processamento. Essa abordagem pode reduzir a quantidade de exemplos isolados facilmente observados ao utilizar a técnica corte epsilon isoladamente.

O conhecimento prévio do domínio também é um importante fator que deve ser levado em consideração durante o processo de construção da rede, pois, uma vez que algumas propriedades da base são conhecidas, estas podem colaborar na forma como os pontos são conectados. Além disso, com base nos vértices (dados) previamente rotulados, pode-se estabelecer novas conexões na rede ligando tais vértices.

3 Metodologia

3.1 Uso de autoencoders para geração de novas representações

Neste projeto, o autoencoder será utilizado como uma espécie de ferramenta de pré-processamento dos dados, ou seja, ao invés de utilizar os dados originais obtidos a partir de uma tabela atributo valor (ver Tabela 1), o primeiro passo para a geração da rede consistirá em treinar um autoencoder com esse conjunto gerando um novo conjunto de atributos.

Durante o processo de treinamento, a rede neural irá aprender uma função de mapeamento $h_W(x) \approx x$, onde W é o conjunto de pesos obtido durante a aprendizagem. Os neurônios da camada L2 representam um novo conjunto de características extraídas do conjunto original. Dentre as principais vantagens desse conjunto, as seguintes podem ser destacadas: correlação reduzida entre os novos atributos, valor normalizados entre 0 e 1 (ou -1 e 1, conforme a função de ativação utilizada); e número reduzido de características, além de outras.

Após o treinamento do autoencoder para um determinado conjunto de dados, uma rede (grafo) será montada seguindo as abordagens 1) corte *épsilon* e 2) *k*NN. Para isso, a similaridade entre os atributos obtidos na camada L2 serão utilizados.

3.2 Geração da rede com as novas representações

De uma forma geral, a geração de uma rede a partir de um conjunto de dados pode ser sumarizada nos seguintes passos (LIU; CHANG, 2009; ZHU; GOLDBERG, 2009; BREVE; ZHAO; QUILES, 2015):

1. Um vértice é gerado para cada exemplo do conjunto de dados;
2. Uma função de similaridade é definida;
3. Os parâmetros de similaridade são ajustados;
4. Se a similaridade entre um par de exemplos é superior a um determinado limiar (épsilon) ou se eles estão entre os k vizinhos mais próximos, uma aresta é criada entre seus respectivos vértices representantes;
5. As arestas podem ou não ser ponderadas;

Normalmente, para se estabelecer a similaridade entre os exemplos um Kernel RBF é utilizado:

$$V(X_i, X_j) = exp\left(-\frac{d(X_i, X_j)}{2\sigma^2}\right) \tag{3.1}$$

no qual X_i e X_j representam exemplos do conjunto (linhas da tabela atributo-valor excluindo-se o atributo classe); $d(X_i, X_j)$ representa a distância entre os exemplos, normalmente a distância euclidiana; σ define a abertura da gaussiana; e $V()$ define a similaridade entre os exemplos. Essa similaridade é utilizada para definir o corte, via corte *épsilon*, ou para encontrar os *k* vizinhos mais próximos.

Além de *k* ou *épsilon*, o valor de σ deve ser ajustado para cada conjunto de dados em particular. Assim, ao utilizar a representação gerada pelo autoencoder, espera-se encontrar uma melhor padronização e robustez a escolha dos parâmetros para a geração da rede.

3.3 Experimentos computacionais e avaliação dos resultados

Para avaliar a qualidade da rede gerada, técnicas clássicas de classificação semissupervisionada (*benchmarks*) serão comparadas, como por exemplo a técnica proposta por Zhu et al. baseada em Funções Harmônicas (ZHU et al., 2003), a técnica de Consistência Local e Global proposta por Zhou et al. (ZHOU et al., 2004) e a técnica de classificação baseada em Competição de Partículas proposta Breve et al. (BREVE; ZHAO; QUILES, 2015).

Testes serão realizados com dados sintéticos e com dados reais selecionados da UCI (ASUNCION; NEWMAN, 2007) amplamente utilizados para a validação de abordagens encontradas na literatura.

Para os testes e comparações serão comparadas as redes geradas com os novos atributos obtidos pelo autoencoder com as redes geradas a partir dos dados originais sem pré-processamento. Especificamente para a comparação das redes, utilizar-se-á um classificador semi-supervisionado LGC (*local global consistency*), por exemplo. A Figura 10 e Figura 11 fazem um resumo dos experimentos.

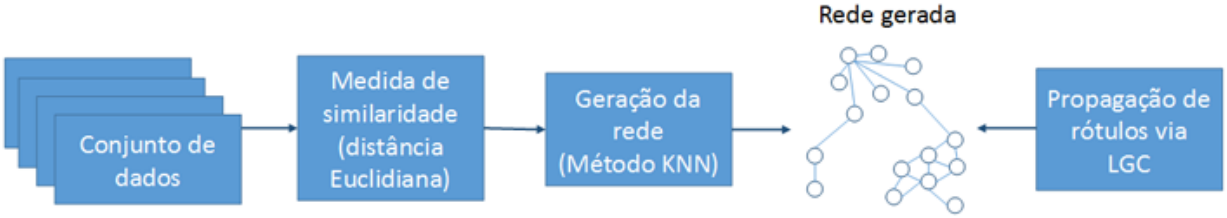

Figura 10 – Resumo da metodologia aplicada para o conjunto de dados original.

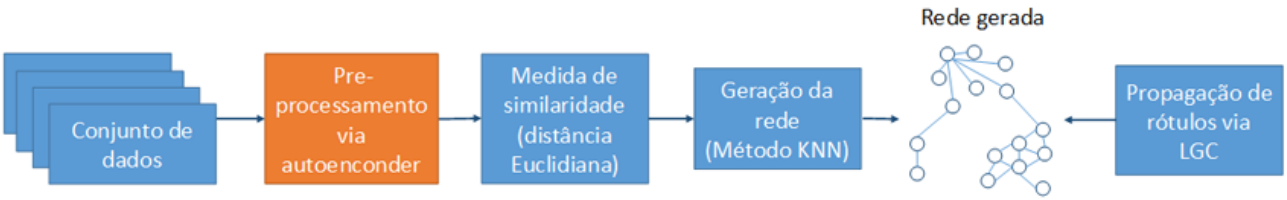

Figura 11 – Resumo da metodologia aplicada para as representações geradas através do pré-processamento via *autoencoder*.

4 Experimentos e Resultados

O objetivo deste trabalho foi explorar o uso de uma rede neural com uma arquitetura Autoencoder como forma de pré-processamento de um conjunto de dados, como forma de reduzir a quantidade de atributos, mantendo uma alta representatividade do mesmo.

Na Seção 4.1 são mostrados as bases de dados utilizadas. Na Seção 4.2 são mostrados os valores dos parâmetros testados. Por fim, na Seção 4.3 são explicados com mais detalhes os experimentos realizados e os resultados obtidos.

4.1 Bases de Dados

As bases de dados usadas neste trabalho foram todas retiradas do Repositório da Universidade da Califórnia (BACHE; LICHMAN, 2013). A Tabela 2 faz um resumo das características de cada base.

Procurou-se explorar base de dados com aspectos diferentes no que se refere a número de objetos, atributos e classes, respeitando o tipo de dados. Todas as bases possuem apenas dados numéricos, com excessão às classes. A base com maior número de objetos foi a *glass*, e a menor foi a *iris*. A base com maior número de atributos foi a *parkinsons*, a menor a base *iris*. A base de dados com maior número de classes foi a *glass*, a menor foi a *parkinsons*.

4.2 Parâmetros

Os parâmetros utilizados no trabalho são: Γ, utilizado no *autoencoder* para definir o máximo de épocas que a rede poderia alcançar em sua fase de treinamento; μ, o número de neurônios da camada oculta do *autoencoder*; δ, a quantidade de *folds* utilizados na validação cruzada; k, utilizados nos métodos de geração da rede k**NN**. A Tabela 3 mostra os valores utilizados para cada um dos parâmetros.

Tabela 2 – Bases de dados utilizadas nos experimentos e as quantidades de objetos, atributos e classes em cada uma delas.

Nome	Objetos	Atributos	Classes
glass	214	9	7
iris	150	4	3
wine	178	13	3
seeds	210	7	3
parkinsons	195	22	2

Tabela 3 – Valores utilizados para os parâmetros.

Parâmetro	Valor
k	1...10
Γ	100000
μ	2...(Quantidade de atributos do conjunto de dados) - 1
δ	10

4.3 Resultados

Os experimentos realizados tiveram como objetivo verificar até que ponto a redução de atributos de uma base de dados, atravéz do *autoencoder*, mantém uma alta representatividade. Além disso, verificar qual o parâmetro k do método k vizinhos mais próximos que gerará a rede dessa nova representação. Por fim, verificar se a utilização do autoencoder como um pré-processamento dos dados melhoraria os resultados. A Figura 12 faz um resumo dos experimentos.

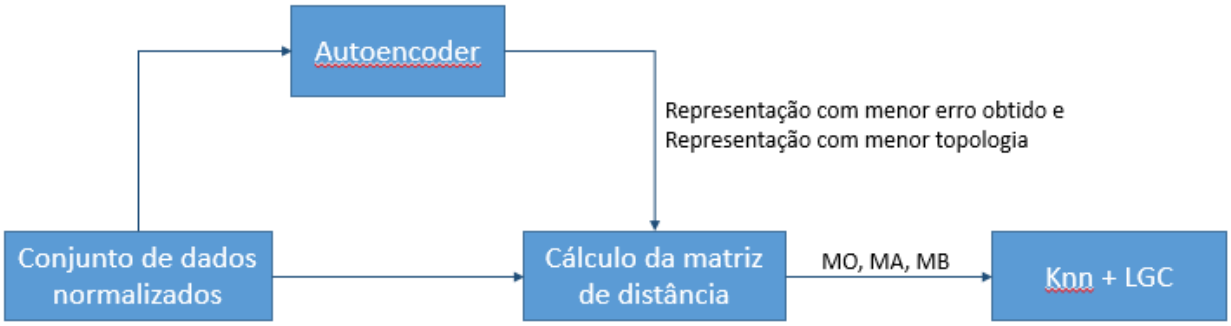

Figura 12 – Resumo dos experimentos realizados.

O primeiro passo dos experimentos é normalizar a base de dados. Feito isso os dados são submetidos a um autoencoder e são utilizados diretamente para a construção da matriz de distâncias original (**MO**).

O segundo passo é verificar qual o número de neurônios na camada oculta do *autoencoder*, tendo seu valor inicial igual ao número de atributos da base de dados menos 1 e chegando até 2, que possui o menor erro (**A**). Também é analisado qual a menor topologia gerada com erro tolerável (**B**).

Depois de encontrados os valores para **A** e **B**, para cada um deles o **autoencoder** é treinado, e após treinado, os dados originais são então aplicados à camada de entrada e à camada escondida, e a saída da camada escondida substitui os valores originais na base de dados. Os valores da base de dados agora vão possuir uma quantidade de atributos igual ao número de neurônios na camada escondida, e são utilizados na construção de suas matrizes de distância, **MA** e **MB**, respectivamente.

Tabela 4 – Estatísticas dos valores obtidos da execução do algoritmo LGC 100 vezes com k variando de 1...10, para cada representação do conjunto de dados *glass*.

		Conjunto de dados		
		GLASS		
k		**Original**	**Autoencoder Menor Erro**	**Autoencoder Menor Topologia**
1	**Média**	0.995259	0.992984	0.994842
	Des. Pad.	1.000274	0.998007	0.999854
	Mediana	0.994845	0.994845	0.994845
2	**Média**	0.995205	0.945255	0.969951
	Des. Pad.	1.000219	0.963155	0.979096
	Mediana	0.994845	0.994845	0.994845
3	**Média**	0.980913	0.967444	0.953305
	Des. Pad.	0.986862	0.974927	0.967270
	Mediana	0.994845	0.994845	0.994845
4	**Média**	0.979859	0.994843	0.968613
	Des. Pad.	0.985568	0.999855	0.979070
	Mediana	0.994845	0.994845	0.994845
5	**Média**	0.975427	0.984143	0.955600
	Des. Pad.	0.982306	0.989628	0.968569
	Mediana	0.994845	0.994845	0.994845
6	**Média**	0.975136	0.991657	0.972010
	Des. Pad.	0.981850	0.996721	0.980485
	Mediana	0.994845	0.994845	0.994845
7	**Média**	0.988391	0.940674	0.968572
	Des. Pad.	0.993614	0.956846	0.977562
	Mediana	0.994845	0.994845	0.994845
8	**Média**	0.993759	0.995152	0.994637
	Des. Pad.	0.998770	1.000166	0.999648
	Mediana	0.994845	0.994845	0.994845
9	**Média**	0.985238	0.993396	0.994843
	Des. Pad.	0.990679	0.998406	0.999854
	Mediana	0.994845	0.994819	0.994845
10	**Média**	0.994274	0.995616	0.993605
	Des. Pad.	0.999283	1.000633	0.998616
	Mediana	0.994819	0.994845	0.994845

Geradas as matrizes de distâncias, é aplicado o algoritmo dos Knn para k variando de 1 a 10, e para cada uma dessas novas representações é aplicado então, o algoritmo LGC 100 vezes para ser possível uma análise estatística.

Os resultados obtidos estão nas tabelas a seguir. Na Tabela 4 são mostrados os resultados para o conjunto de dados *glass*, na Tabela 5 para o conjunto *iris*, na Tabela 6 para o conjunto *parkinsons*, na Tabela 7 para o conjunto *seeds* e por fim, na Tabela 8 para o conjunto *wine*. Cada tabela representa um conjunto de dados. Para cada uma é mostrado os valores da média, mediana e desvio padrão dos valores de 100 execuções do algoritmo LGC para k variando de 1 a 10 e para cada representação **MO**, **MA**, **MB** de cada conjunto de dados utilizada.

Tabela 5 – Estatísticas dos valores obtidos da execução do algoritmo LGC 100 vezes com k variando de 1...10, para cada representação do conjunto de dados *iris*.

		Conjunto de dados		
		IRIS		
k		**Original**	**Autoencoder Menor Erro**	**Autoencoder Menor Topologia**
1	**Média**	0.989593	0.984785	0.992699
	Des. Pad.	0.994628	0.990066	0.997701
	Mediana	0.992593	0.992593	0.992593
2	**Média**	0.934571	0.963581	0.935606
	Des. Pad.	0.957434	0.973491	0.957819
	Mediana	0.992593	0.992647	0.992593
3	**Média**	0.941609	0.965203	0.944042
	Des. Pad.	0.962205	0.974566	0.963148
	Mediana	0.992647	0.992647	0.992647
4	**Média**	0.942199	0.939709	0.907034
	Des. Pad.	0.962427	0.952901	0.934339
	Mediana	0.992647	0.992593	0.992593
5	**Média**	0.898054	0.974878	0.856759
	Des. Pad.	0.930518	0.982874	0.920393
	Mediana	0.992593	0.992647	0.992593
6	**Média**	0.961267	0.975248	0.946327
	Des. Pad.	0.975833	0.983117	0.972486
	Mediana	0.992647	0.992647	0.992647
7	**Média**	0.961267	0.963445	0.910672
	Des. Pad.	0.975833	0.972795	0.952009
	Mediana	0.992647	0.992647	0.992647
8	**Média**	0.963670	0.981695	0.949371
	Des. Pad.	0.972943	0.987094	0.964132
	Mediana	0.992647	0.992593	0.992647
9	**Média**	0.947723	0.986420	0.924166
	Des. Pad.	0.960309	0.991538	0.947334
	Mediana	0.992593	0.992593	0.992593
10	**Média**	0.956851	0.988493	0.970646
	Des. Pad.	0.966594	0.993538	0.977371
	Mediana	0.992593	0.992593	0.992593

Tabela 6 – Estatísticas dos valores obtidos da execução do algoritmo LGC 100 vezes com k variando de 1...10, para cada representação do conjunto de dados *parkinsons*.

		Conjunto de dados		
		PARKINSONS		
k		**Original**	**Autoencoder Menor Erro**	**Autoencoder Menor Topologia**
1	**Média**	0.994856	0.861414	0.922460
	Des. Pad.	0.999870	0.865993	0.927828
	Mediana	0.994350	0.864407	0.943662
2	**Média**	0.980687	0.653287	0.735222
	Des. Pad.	0.986333	0.681652	0.769491
	Mediana	0.994350	0.723164	0.627119
3	**Média**	0.968116	0.594160	0.729614
	Des. Pad.	0.978575	0.630512	0.776687
	Mediana	0.994350	0.677966	0.795455
4	**Média**	0.969187	0.603274	0.766037
	Des. Pad.	0.977843	0.643237	0.801116
	Mediana	0.994350	0.681818	0.954545
5	**Média**	0.988148	0.603341	0.766099
	Des. Pad.	0.994082	0.641872	0.806275
	Mediana	0.994334	0.681818	0.954545
6	**Média**	0.969812	0.572492	0.687376
	Des. Pad.	0.977851	0.624064	0.731573
	Mediana	0.994350	0.680797	0.619318
7	**Média**	0.955838	0.573430	0.733096
	Des. Pad.	0.966363	0.621965	0.778736
	Mediana	0.994350	0.684505	0.801136
8	**Média**	0.979642	0.570112	0.717579
	Des. Pad.	0.986318	0.609790	0.757468
	Mediana	0.994350	0.677966	0.627119
9	**Média**	0.972793	0.614630	0.745882
	Des. Pad.	0.980452	0.641346	0.786133
	Mediana	0.994318	0.681564	0.803346
10	**Média**	0.964249	0.614250	0.779955
	Des. Pad.	0.972568	0.642903	0.809717
	Mediana	0.994350	0.680669	0.954545

Tabela 7 – Estatísticas dos valores obtidos da execução do algoritmo LGC 100 vezes com k variando de 1...10, para cada representação do conjunto de dados *seeds*.

		Conjunto de dados		
		SEEDS		
k		**Original**	**Autoencoder Menor Erro**	**Autoencoder Menor Topologia**
1	**Média**	0.991266	0.987995	0.993527
	Des. Pad.	0.996324	0.992979	0.998541
	Mediana	0.994737	0.989446	0.994737
2	**Média**	0.951833	0.866398	0.967237
	Des. Pad.	0.974918	0.901388	0.979548
	Mediana	0.994737	0.989418	0.994737
3	**Média**	0.956318	0.933308	0.943812
	Des. Pad.	0.979573	0.953179	0.962431
	Mediana	0.994737	0.989474	0.994737
4	**Média**	0.920261	0.918500	0.944051
	Des. Pad.	0.955031	0.941396	0.962539
	Mediana	0.994737	0.989474	0.994737
5	**Média**	0.943412	0.856063	0.961657
	Des. Pad.	0.969897	0.889735	0.975420
	Mediana	0.994737	0.989418	0.994737
6	**Média**	0.888065	0.938336	0.962628
	Des. Pad.	0.938572	0.951039	0.975879
	Mediana	0.994737	0.989474	0.994737
7	**Média**	0.870638	0.921865	0.949512
	Des. Pad.	0.928027	0.940325	0.960929
	Mediana	0.994737	0.989474	0.994737
8	**Média**	0.898272	0.926578	0.956644
	Des. Pad.	0.943981	0.944103	0.968302
	Mediana	0.994737	0.989418	0.994723
9	**Média**	0.898065	0.927151	0.994262
	Des. Pad.	0.943955	0.946061	0.999273
	Mediana	0.994737	0.989474	0.994737
10	**Média**	0.877321	0.938839	0.993467
	Des. Pad.	0.933093	0.955015	0.998478
	Mediana	0.994723	0.989418	0.994723

Tabela 8 – Estatísticas dos valores obtidos da execução do algoritmo LGC 100 vezes com k variando de 1...10, para cada representação do conjunto de dados *wine*.

		Conjunto de dados		
		WINE		
k		**Original**	**Autoencoder Menor Erro**	**Autoencoder Menor Topologia**
1	**Média**	0.992823	0.992017	0.992017
	Des. Pad.	0.997830	0.997034	0.997034
	Mediana	0.993827	0.993827	0.993827
2	**Média**	0.918900	0.918771	0.925339
	Des. Pad.	0.958177	0.958167	0.958841
	Mediana	0.993827	0.993789	0.993789
3	**Média**	0.924163	0.894448	0.889502
	Des. Pad.	0.958676	0.942770	0.942408
	Mediana	0.993789	0.993789	0.993789
4	**Média**	0.900879	0.936550	0.936861
	Des. Pad.	0.943595	0.968446	0.968461
	Mediana	0.993789	0.993827	0.993827
5	**Média**	0.935991	0.902551	0.896801
	Des. Pad.	0.968426	0.943881	0.943028
	Mediana	0.993827	0.993789	0.993789
6	**Média**	0.887711	0.938662	0.906632
	Des. Pad.	0.942338	0.964791	0.944678
	Mediana	0.993789	0.993827	0.993789
7	**Média**	0.926780	0.956164	0.941556
	Des. Pad.	0.963285	0.975041	0.965477
	Mediana	0.993827	0.993827	0.993827
8	**Média**	0.948450	0.954988	0.956597
	Des. Pad.	0.973729	0.974762	0.975150
	Mediana	0.993827	0.993827	0.993827
9	**Média**	0.947978	0.962019	0.953623
	Des. Pad.	0.973692	0.979467	0.974475
	Mediana	0.993827	0.993827	0.993827
10	**Média**	0.955560	0.913781	0.960901
	Des. Pad.	0.978676	0.949562	0.979252
	Mediana	0.993827	0.993789	0.993827

4.3.1 Análise dos Resultados

Fazendo uma análise dos resultados mostrados nas Tabela 4, Tabela 5, Tabela 6, Tabela 7, Tabela 8, Tabela 9, Tabela 10, percebe-se que em pelo menos em um dos valores de k o erro obtido pela representação compactada pelo *autoencoder*, tanto pela representação de menor erro ou a representação de menor topologia, obtiveram resultados melhores do que as representações originais. Em especial, analisando o valor de k, os melhores resultados foram obtidos com valores pequenos. No caso onde $k = 1$, vemos que para todos os conjuntos de dados, as representações geradas pelo *autoencoder* obtiveram os melhores resultados.

Independente do valor de k, nota-se que todas as bases tiveram bons resultados, onde, em maioria os melhores foram das bases geradas pela técnica abordada, e mesmo onde obtiveram resultados piores que as bases originais, esse resultado ficou muito próximo.

Analisando os resultados obtidos na base *glass* (Tabela 4), nota-se que a melhor representação obtida foi a gerada pelo *autoencoder* de menor topologia, mesmo esse tendo uma acurácia pior do que a outra gerada pelo mesmo método. Os resultados da base original só foram melhores para os valores de $k = 8$ e $k = 9$.

Para o conjunto de dados *iris* (Tabela 5), é visto um mesmo comportamento do que a base *glass*. A representação de melhores resultados foram as geradas pela técnica, no qual tem a menor topologia.

A base *parkinsons* (Tabela 6) foi onde foram obtidos os melhores resultados. Para todos os valores de k, as representações geradas pela técnica obtiveram resultados consideravelmente menores do que a base original.

Quando analisado o conjunto de dados *seeds* (Tabela 7), é visível o impacto que o valor de k tem sobre a base de dados, sendo que para valores pequenos, os melhores resultados são dados pelas representações geradas pela técnica. Já para valores de k mais altos, as representações originais obtiveram os melhores resultados.

Na Tabela 8 nota-se que as representações geradas pela técnica obtiveram melhores resultados para valores pequenos de k.

Na comparação entre usar a representação gerada pela técnica de maior acurácia, ou a representação gerada pela menor topologia com acurácia aceitável, observa-se que a de maior acurácia tem maior chances de se obter melhores resultados, porém, a menor representação, muitas vezes, também obteve resultados melhores ou equivalentes aos da base original. Portanto, a escolha do uso deverá levar em conta uma análise do conjunto de dados para melhores resultados.

Nas Tabela 9 e Tabela 10 podemos notar o impacto da escolha do valor de μ tanto na acurácia quanto no resultado final da rede gerada. Nota-se que as melhores acurácias são obtidas com um número pouco menor que a quantidade de atributos do conjunto de dados original.

Tabela 9 – Quantidade de atributos obtidos pelas representações geradas pelo *autoencoder* por conjunto de dados. MA: Maior Acurácia, MB: Menor topologia de acurácia aceitável.

	Número de atributos		
	Original	**Representação MA**	**Representação MB**
glass	9	3	2
iris	4	3	2
parkinsons	22	19	5
seeds	7	6	5
wine	13	9	7

Tabela 10 – Valores das acurácias obtidas ao treinar a rede *autoencoder* para as representações MA e MB. Foi utilizada a validação cruzada com método *k-fold* utilizando 10 *folds*. MA: Maior Acurácia, MB: Menor topologia de acurácia aceitável.

	Acurácias	
	MA	**MB**
glass	0.002566	0.002859
iris	0.003857	0.004241
parkinsons	0.010058	0.011330
seeds	0.008254	0.009627
wine	0.008728	0.011347

Percebe-se também a grande diferença entre o número de atributos entre a base original e a representação com menor topologia gerada pelo *autoencoder*. Mais especificamente, para o conjunto *parkinsons*, com o maior valor de atributos, a representação gerada com menor valor de μ obteve uma acurácia próxima a representação **MA** e seu resultado continuou sendo melhor do que a representação original, como podemos notar na Tabela 6.

Por fim, a comparação dos resultados obtidos sugere que a utilização da representação gerada pela técnica estudada não só obtém melhores resultados, mas como também é melhor generalizável, possui alta representatividade e gera melhores resultados.

5 Conclusões

Neste trabalho, um método de geração de redes a partir de uma base de dados, testando diferentes parâmetros da rede, como uma forma de pré-processamento foi testado e analisado.

As duas representações testadas geradas pela rede possuiam menos atributos do que a representação original, ver Tabela 9, e obtiveram no geral, os melhores resultados quando aplicado o algorítmo *k***NN** e LGC. Para a base de dados *parkinsons*, a base de maior número de atributos, obervou-se que os melhores resultados foram obtidos em maioria esmagadora através das representações geradas pela técnica. Reduzindo de 22 atributos até 5, maximizando os resultados.

Para as outras bases de dados, os resultados foram similares para valores de k pequenos, em especial $k = 1$, onde as representações geradas pela técnica foram as que alcançaram os melhores resultados.

Por fim, os resultados obtidos mostraram que foi possível utilizar um *autoencoder* como ferramenta de pré-processamento do conjunto de dados, gerando melhores resultados, ou equivalentes, do que da representação sem esse processamento. Notou-se também que em todos os conjuntos de dados, foi vantajoso ter seus atributos reduzidos, como já era esperado.

Referências

AGUIAR, F. G. *Utilização de redes neurais artificais para detecção de padrões de vazamento em dutos*. Tese (Doutorado) — Universidade de São Paulo, 2010. Citado 2 vezes nas páginas 13 e 30.

ASUNCION, A.; NEWMAN, D. *UCI machine learning repository*. 2007. Citado na página 42.

BACHE, K.; LICHMAN, M. *UCI machine learning repository*. 2013. Citado na página 45.

BETECHUOH, B. L.; MARWALA, T.; TETTEY, T. Autoencoder networks for hiv classification. *CURRENT SCIENCE-BANGALORE-*, CURRENT SCIENCE ASSOC/INDIAN ACADEMY OF SCIENCES, v. 91, n. 11, p. 1467, 2006. Citado na página 34.

BREVE, F. A.; ZHAO, L.; QUILES, M. G. Particle competition and cooperation for semi-supervised learning with label noise. *Neurocomputing*, Elsevier, v. 160, p. 63–72, 2015. Citado 2 vezes nas páginas 41 e 42.

DUCH, W.; JANKOWSKI, N. Transfer functions: hidden possibilities for better neural networks. In: CITESEER. *ESANN*. [S.l.], 2001. p. 81–94. Citado na página 32.

DUDANI, S. A. The distance-weighted k-nearest-neighbor rule. *Systems, Man and Cybernetics, IEEE Transactions on*, IEEE, n. 4, p. 325–327, 1976. Citado na página 37.

FODOR, I. K. *A survey of dimension reduction techniques*. [S.l.]: Technical Report UCRL-ID-148494, Lawrence Livermore National Laboratory, 2002. Citado na página 23.

FUKUNAGA, K. *Introduction to statistical pattern recognition*. [S.l.]: Academic press, 2013. Citado na página 23.

HAUSSLER, D.; WELZL, E. e-nets and simples range queries. *Discrete & Computational Geometry*, Springer, v. 2, n. 1, p. 127–151, 1987. Citado na página 38.

HEIN, M.; MAIER, M. Manifold denoising as preprocessing for finding natural representations of data. In: MENLO PARK, CA; CAMBRIDGE, MA; LONDON; AAAI PRESS; MIT PRESS; 1999. *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*. [S.l.], 2007. v. 22, n. 2, p. 1646. Citado na página 38.

JEBARA, T.; SHCHOGOLEV, V. B-matching for spectral clustering. In: *Machine learning: Ecml 2006*. [S.l.]: Springer Berlin Heidelberg, 2006. p. 679–686. Citado na página 24.

JEBARA, T.; WANG, J.; CHANG, S.-F. Graph construction and b-matching for semi-supervised learning. In: ACM. *Proceedings of the 26th Annual International Conference on Machine Learning*. [S.l.], 2009. p. 441–448. Citado na página 24.

JIMENEZ, L. O.; LANDGREBE, D. et al. Supervised classification in high-dimensional space: geometrical, statistical, and asymptotical properties of multivariate data. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, IEEE, v. 28, n. 1, p. 39–54, 1998. Citado na página 23.

KAMYSHANSKA, H.; MEMISEVIC, R. On autoencoder scoring. In: *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*. [S.l.: s.n.], 2013. p. 720–728. Citado na página 35.

KORDOS, M. law; DUCH, W. lodzis law. A survey of factors influencing mlp error surface. *Control and Cybernetics*, v. 33, n. 4, 2004. Citado na página 32.

KUBAT, M. *Neural networks: a comprehensive foundation by Simon Haykin, Macmillan, 1994, ISBN 0-02-352781-7*. [S.l.]: Cambridge Univ Press, 1999. Citado 2 vezes nas páginas 25 e 32.

KUPREL, B. Judging a movie by its poster using deep learning. Citado 2 vezes nas páginas 13 e 34.

LIU, W.; CHANG, S.-F. Robust multi-class transductive learning with graphs. In: IEEE. *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. [S.l.], 2009. p. 381–388. Citado 2 vezes nas páginas 24 e 41.

LIU, W.; WANG, J.; CHANG, S.-F. Robust and scalable graph-based semisupervised learning. *Proceedings of the IEEE*, IEEE, v. 100, n. 9, p. 2624–2638, 2012. Citado 2 vezes nas páginas 13 e 24.

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. Citado na página 27.

OLSHAUSEN, B. A. et al. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, v. 381, n. 6583, p. 607–609, 1996. Citado 2 vezes nas páginas 24 e 25.

QUILES, M. G. et al. Label propagation through neuronal synchrony. In: IEEE. *Neural Networks (IJCNN), The 2010 International Joint Conference on*. [S.l.], 2010. p. 1–8. Citado na página 24.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, American Psychological Association, v. 65, n. 6, p. 386, 1958. Citado na página 28.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. *Learning internal representations by error propagation*. [S.l.], 1985. Citado 2 vezes nas páginas 24 e 25.

SELLI, M. F.; JR, P. S. Online identification of horizontal two-phase flow regimes through gabor transform and neural network processing. *Heat transfer engineering*, Taylor & Francis, v. 28, n. 6, p. 541–548, 2007. Citado na página 27.

SMOLENSKY, P. Information processing in dynamical systems: Foundations of harmony theory. Department of Computer Science, University of Colorado, Boulder, 1986. Citado na página 35.

SOUSA, C. A. R. d. *Impacto da geração de grafos na classificação semissupervisionada*. Tese (Doutorado) — Universidade de São Paulo, 2013. Citado na página 24.

WOLD, S.; ESBENSEN, K.; GELADI, P. Principal component analysis. *Chemometrics and intelligent laboratory systems*, Elsevier, v. 2, n. 1, p. 37–52, 1987. Citado na página 23.

ZHOU, D. et al. Learning with local and global consistency. *Advances in neural information processing systems*, v. 16, n. 16, p. 321–328, 2004. Citado na página 42.

ZHU, X. et al. Semi-supervised learning using gaussian fields and harmonic functions. In: *ICML*. [S.l.: s.n.], 2003. v. 3, p. 912–919. Citado na página 42.

ZHU, X.; GOLDBERG, A. B. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, Morgan & Claypool Publishers, v. 3, n. 1, p. 1–130, 2009. Citado na página 41.

ZHU, X.; LAFFERTY, J.; ROSENFELD, R. *Semi-supervised learning with graphs*. [S.l.]: Carnegie Mellon University, Language Technologies Institute, School of Computer Science, 2005. Citado na página 23.