UNIFACCAMP
Centro Universitário Campo Limpo Paulista

# *Acelerando a Fase de Busca do Algoritmo k-NN*

**Raphael Lopes de Souza**

Abril / 2023

Dissertação de Mestrado em Ciência da Computação

# Acelerando a Fase de Busca do Algoritmo *k*-NN

Esse documento corresponde à dissertação apresentado à Banca Examinadora para defesa de Mestrado em Ciência da Computação do Centro Universitário Campo Limpo Paulista.

Campo Limpo Paulista, 13 de abril de 2023.

Raphael Lopes de Souza

Prof. Dr. Osvaldo Luiz de Oliveira (Orientador)

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

## FICHA CATALOGRÁFICA

Ficha catalográfica elaborada pela
Biblioteca Central da Unifaccamp

S718a

Souza, Raphael Lopes de

Acelerando a fase de busca do algoritmo K-NN / Raphael Lopes de Souza. Campo Limpo Paulista, SP: Unifaccamp, 2023.

Orientador: Prof. Dr. Osvaldo Luiz de Oliveira

Dissertação (Programa de Mestrado Profissional em Ciência da Computação) – Centro Universitário Campo Limpo Paulista – Unifaccamp.

1. K-NN. 2. K vizinhos mais próximos. 3. Busca rápida. 4. Desigualdade triangular. I. Oliveira, Osvaldo Luiz de. II. Centro Universitário Campo Limpo Paulista. III. Título.

CDD – 005.1

***Resumo:*** *O k-NN – k-Nearest Neighbors – é um algoritmo utilizado em aplicações de Aprendizado de Máquina que envolvem, entre outras, reconhecimento de padrões, mineração de dados e raciocínio automatizado. Tendo como entradas uma instância x e um inteiro k ≥ 1, o k-NN envolve um processo de busca de k instâncias mais similares à instância x em um conjunto T de N ≥ 1 instâncias definidas por D ≥ 1 dimensões. Quando T possui grande número de instâncias e/ou grande dimensionalidade tal processo de busca tende a consumir muito tempo devido, principalmente a cálculos de funções de similaridade entre instâncias, geralmente utilizando a métrica descrita pela distância euclidiana entre instâncias. Diversas propostas têm sido feitas para reduzir o tempo de busca do algoritmo k-NN. Este trabalho relata os esforços de investigar algoritmos que objetivam reduzir o tempo de busca do algoritmo k-NN e apresentar uma proposta de um novo algoritmo, denominado kM++kNN, que utiliza o conceito de desigualdade triangular para reduzir o tempo de busca do k-NN. O algoritmo kM++kNN é comparado, em experimentos de medida da economia do número de cálculos funções de similaridade entre instâncias e de tempo de busca, com algoritmos atualmente considerados rápidos e que também utilizam o conceito de desigualdade triangular para acelerar a busca do algoritmo k-NN.*

***Palavras-chave:*** *k-NN, k vizinhos mais próximos, busca rápida, desigualdade triangular.*

***Abstract**: The k-NN – k-Nearest Neighbors – is an algorithm used in Machine Learning applications such as pattern recognition, data mining and automated reasoning. Having as inputs an instance x and an integer k ≥ 1, the k-NN searches for k similar instances to the instance x in a set T of N ≥ 1 instances with D ≥ 1 dimensions. When T has many instances and/or with large dimensionality, this search process tends to consume a lot of time, due to calculations of similarity functions between instances. Several proposals have been made to reduce the search time of the k-NN algorithm. This work reports the efforts to investigate algorithms, that aim to reduce the search time of the k-NN algorithm, and present a proposal for a new algorithm, called kM++kNN, which uses the concept of triangular inequality to reduce the k-NN search time. The kM++kNN algorithm is compared, in experiments to measure the economy of calculations of similarity functions between instances and search time, with algorithms currently considered fast and which also use the concept of triangular inequality to speed up the search of the algorithm k-NN.*

***Keywords:** k-NN, k-nearest neighbors, fast search, triangle inequality.*

**Dedicatória**

Dedico essa dissertação a minha esposa, filhos, família e aos professores que me apoiaram e me deram sustento durante este período de aprofundamento nos estudos.
Dedico também a Deus, que sempre me fortalece nos momentos de dificuldade.

**Agradecimentos**

Agradeço primeiramente a Deus por me dar discernimento, me sustentar e conduzir para a realização deste trabalho.

Ao meu orientador professor Dr. Osvaldo Luiz de Oliveira pela dedicação, paciência, compreensão, insistência, motivação e ensinamentos durante todo o processo de desenvolvimento deste trabalho, um verdadeiro professor, amigo e orientador, obrigado por ajudar a melhorar sempre, agradecimentos eternos.

À minha esposa e filhos, pelo sustento durante todo o processo de estudos até aqui, pela perseverança, motivação que me trouxeram, minha eterna gratidão e amor a vocês.

Aos meus familiares que sempre me fortaleceram durante esse tempo de estudo.

Aos professores e funcionários do programa de mestrado em Ciência da Computação do Centro Universitário Campo Limpo Paulista.

A todos que contribuíram para a realização deste trabalho.

“Aqueles que passam por nós não vão sós. Deixam um pouco de si, levam um pouco de nós.”

(Antoine de Saint-Exupéry)

# Sumário

**CAPÍTULO 1 INTRODUÇÃO ..... 14**

1.1 – CONTEXTO DA PESQUISA ..... 15
1.2 – PROBLEMA E MOTIVAÇÃO ..... 17
1.3 – OBJETIVOS E CONTRIBUIÇÕES ..... 18
1.4 – ORGANIZAÇÃO DESTE DOCUMENTO ..... 19

**CAPÍTULO 2 CONCEITOS FUNDAMENTAIS E NOTAÇÕES ..... 21**

2.1 – CONJUNTO DE INSTÂNCIAS ..... 22
2.2 – ESPAÇOS MÉTRICOS ..... 23
2.3 – DISTÂNCIA EUCLIDIANA ENTRE INSTÂNCIAS ..... 26
2.4 – SIMILARIDADE E DISSIMILARIDADE ENTRE INSTÂNCIAS ..... 27
2.5 – DESIGUALDADE TRIANGULAR ..... 28

**CAPÍTULO 3 REVISÃO BIBLIOGRÁFICA ..... 31**

3.1 – ALGORITMO *K*-NN ..... 32
3.2 – COMPLEXIDADE COMPUTACIONAL DO *K*-NN E A PROBLEMÁTICA ..... 33
3.3 – ALGORITMO *K*M*K*NN ..... 34
3.4 – ALGORITMO KMC-OTI-FS ..... 42
3.4 – ALGORITMO KMC-EOTI-FS ..... 49

**CAPÍTULO 4 ALGORITMO *K*M++*K*NN ..... 56**

4.1 – IDEIA PARA UM NOVO ALGORITMO ..... 57
4.2 – O ALGORITMO *K*M++*K*NN ..... 57

**CAPÍTULO 5 METODOLOGIA DA PESQUISA ..... 61**

5.1 – METODOLOGIA DE CONDUÇÃO DA PESQUISA E DE EXECUÇÃO DOS EXPERIMENTOS ..... 62
5.2 – AMBIENTE PARA EXPERIMENTAÇÃO ..... 63
5.3 – CONJUNTOS DE INSTÂNCIAS SELECIONADOS PARA OS EXPERIMENTOS ..... 64
5.4 – ESTRATÉGIA DE VALIDAÇÃO CRUZADA ..... 65
5.5 – ÍNDICES DE DESEMPENHO ..... 67
5.6 – MÉTODO DE EXPERIMENTAÇÃO ..... 67

**CAPÍTULO 6 RESULTADOS E DISCUSSÃO ..... 70**

6.1 – RESULTADOS ..... 71

6.2 – DISCUSSÃO ..... 72

**CAPÍTULO 7 CONCLUSÕES E TRABALHOS FUTUROS..... 75**

7.1 – CONCLUSÕES ..... 76

7.2 – TRABALHOS FUTUROS ..... 76

**APÊNDICE I – REPOSITÓRIO PARA O CÓDIGO FONTE DO ALGORITMO *K*M++*K*NN, ESCRITO EM PYTHON, E CONJUNTO DE INSTÂNCIAS UTILIZADO.....81**

# Glossário

| Expressão | Descrição |
|---|---|
| ANN | *ANN (Approximate Nearest Neighbor),* refere-se a algoritmos que buscam acelerar a fase de busca do algoritmo *k*-NN utilizando métodos aproximados, isto é, aceleram a fase de busca do *k*-NN renunciando à busca exata de *k* instâncias em um conjunto de instâncias mais similares a uma instância *x*. |
| ENN | ENN (*Exact Nearest Neighbor)*, refere-se a algoritmos que buscam acelerar a fase de busca do *k*-NN entregando os mesmos resultados que o algoritmo *k*-NN entregaria. |
| IDE | IDE *(Integrated Development Environment)*, é uma expressão utilizada para descrever um ambiente de desenvolvimento de software que integra diversas ferramentas tais como editor de textos, compilador, depurador, etc.. |
| *T* | Conjunto de treinamento. |
| *D* | Dimensão do conjunto *T* (ou seja, número de atributos que descrevem as instâncias). |
| *N* | Número de instâncias de *T*. |
| *M* | Número de grupos em *T*. |
| *x* | Instância de busca. |

# Lista de Tabelas

**Tabela 3.1:** Distâncias calculadas e ordenadas, de *x* a cada centro de grupo, onde *G1* é o primeiro grupo, *G2* o segundo grupo e *G3* o terceiro grupo. ..... 39

**Tabela 3.2:** Distâncias computadas e ordenadas de *x* a cada centro de grupo..... 46

**Tabela 5.1:** Conjuntos de Instâncias selecionados para execução dos experimentos, extraídos do repositório da UCI (Dua & Graff, 2021). ..... 64

**Tabela 5.2:** Relação dos algoritmos utilizados nos experimentos e os capítulos onde foram descritos. ..... 65

**Tabela 6.1:** Resultados dos Experimentos. ..... 71

# Lista de Figuras

**Figura 2.1:** O conjunto de dados Iris, disponível no *UCI Machine Learning Repository* (Dua & Graff, 2021), como um exemplo de conjunto de instâncias, nos termos da Definição 2.1. O conjunto de instâncias Iris possui 150 instâncias, mas por razões ligadas a espaço e suficiência explanatória, a Figura 2.1 apresenta apenas três instâncias. ..... 23

**Figura 2.2:** Representação gráfica dos dois primeiros espaços métricos do Exemplo 2.2: (a) espaço métrico unidimensional; (b) espaço métrico bidimensional. ..... 26

**Figura 2.3:** Desigualdades triangulares: (a) em um triângulo; (b) em distâncias euclidianas entre instâncias de um conjunto de instâncias no espaço real *D*-dimensional. ..... 30

**Figura 3.1:** Pseudocódigo do algoritmo *k*-NN. ..... 33

**Figura 3.2:** Uso do conceito de desigualdade triangular no contexto do algoritmo *k*M*k*NN. ..... 35

**Figura 3.3:** Pseudocódigo algoritmo *k*M*k*NN. ..... 38

**Figura 3.4:** Conjunto de instâncias utilizado para ilustrar o funcionamento do algoritmo *k*M*k*NN. ..... 38

**Figura 3.5:** Agrupamento induzido pelo algoritmo *k*-Means com três grupos..... 39

**Figura 3.6:** A figura apresenta a estratégia utilizada do algoritmo KMC-OTI-FS, a fim de, desprezar as instâncias mais distantes do centro de um grupo $Cj*$. (adaptado de Pan et al., 2020). ..... 44

**Figura 3.7:** Pseudocódigo algoritmo KMC-OTI-FS. ..... 46

**Figura 3.8:** Pseudocódigo do algoritmo KMC-EOTI-FS. ..... 51

**Figura 4.1:** Pseudocódigo do algoritmo *k*M++*k*NN ..... 60

**Figura 5.1:** Etapas da metodologia de pesquisa. ..... 63

**Figura 5.2:** Fluxograma do método de experimentação para implementação, execução e obtenção de resultados dos algoritmos. ..... 69

# Capítulo 1
# Introdução

*Este capítulo descreve o objeto de estudo deste trabalho e o situa em relação ao corpo de conhecimento ao qual ele se insere. Este trabalho é sobre o algoritmo k-NN (Cover & Hart, 1967), especificamente no que diz respeito a contribuições para a melhoria de desempenho de tempo durante a fase de busca realizada pelo algoritmo. Muitas contribuições têm sido feitas para isto, mas estas contribuições podem ser organizadas em algumas poucas categorias no que se refere às técnicas utilizadas para a melhoria do desempenho da fase de busca. Assim, a Seção 1.1 contém uma revisão literária de contribuições e uma categorização destas contribuições. A "região" dentro do "mapa de contribuições" onde se situa o interesse deste trabalho é estabelecida ao final da Seção 1.1. O problema que motiva este trabalho é o da alta complexidade computacional da fase de busca do algoritmo k-NN. A Seção 1.2 analisa a principal causa desta complexidade computacional e argumenta que esta complexidade, embora seja polinomial, é inaceitável em aplicações envolvendo grande número de instâncias. Os objetivos e as contribuições deste trabalho são declarados na Seção 1.3. Por fim, a Seção 1.4 informa sobre como esta dissertação de mestrado está organizada.*

## 1.1 – Contexto da Pesquisa

O **algoritmo *k*-NN** (Cover & Hart, 1967) – do inglês *k-nearest neighbor* – é amplamente empregado na área de Aprendizado de Máquina pela sua simplicidade, efetividade e por inúmeras aplicações práticas em diversas áreas do conhecimento, tais como reconhecimento de padrões (Glowacz & Glowacz, 2016), raciocínio automatizado (Gerhana *et al*., 2017), detecção de ruído em dados (Angiulli, Basta &. Pizzuti, 2006), enfim, a todo um universo de aplicações que necessitam, genericamente, de processos de classificação ou de regressão. A notoriedade deste algoritmo extrapola os limites da pesquisa científica e da engenharia de produtos inteligentes, frequentando artigos dirigidos ao público geral, e sendo considerado um dos dez principais algoritmos da área de Aprendizado de Máquina deve conhecer (GeeksforGeeks, 2021; Le, 2021; ProjectPro, 2021).

**Sobre este trabalho, cabe uma importante ADVERTÊNCIA**. Este trabalho não é sobre aprendizado supervisionado ou aprendizado não supervisionado. Este trabalho é sobre complexidade de algoritmos. O foco deste trabalho está na melhoraria da complexidade de tempo da busca do algoritmo *k*-NN.

O algoritmo *k*-NN é descrito na literatura, a exemplo do que faz Mitchell (1997), como um algoritmo da categoria denominada "baseada em instâncias". Tal algoritmo não demanda nenhum procedimento para treinamento a fim de induzir um modelo genérico para a busca de uma instância[1] $x$ em um conjunto de instâncias $T$, de modo que o próprio conjunto $T$ pode ser considerado o conjunto de treinamento [2]do algoritmo. A inclusão de uma nova instância no conjunto $T$ não requer nada além da própria inclusão da instância em $T$. Tendo por referência uma função de similaridade entre instâncias e, como entradas uma instância $x$ e um inteiro $k \geq 1$, o algoritmo *k*-NN realiza uma busca exaustiva das $k$ instâncias mais similares à instância $x$ em um conjunto de instâncias $T$, contendo $N \geq 1$ instâncias em um espaço de $D \geq 1$ dimensões[3].

[1] A definição de instância será abordada no Capítulo 2.
[2] Em aprendizado supervisionado, um conjunto de treinamento é um subconjunto de instâncias utilizado para induzir um modelo. Não sendo o aprendizado supervisionado foco deste trabalho, basta o leitor se atentar que o conjunto de instâncias será o próprio conjunto de treinamento para fins do entendimento do que este trabalho propõe.
[3] Dimensão refere-se ao número de atributos de uma instância. Este e outros conceitos correlatos estão definidos no Capítulo 2.

Quando usado em processos de classificação, o mecanismo de funcionamento do *k*-NN é bastante simples e consiste apenas em computar o valor da similaridade de uma instância *x* a cada uma das instâncias em *T*, selecionar as *k* instâncias mais similares a *x* e, com base nas classes das instâncias selecionadas, calcular uma classe como resposta ao processo de classificação, geralmente pela regra da maioria dos votos. Contrapõe-se a esta simplicidade o alto custo computacional do processo de busca exaustiva no conjunto de instâncias *T*, principalmente em aplicações com grande número de instâncias com alta dimensionalidade.

Muitas propostas para acelerar o mecanismo de busca do algoritmo *k*-NN têm sido feitas. De forma geral tais propostas podem ser classificadas em duas categorias, aqui denominadas pelas siglas ANN e ENN.

A **categoria ANN** (*Approximate Nearest Neighbor)* engloba algoritmos (e.g., Wang, Pan & Li, 2019; Andoni & Indyk, 2006) que propõem acelerar o processo de busca do *k*-NN sacrificando a precisão dos resultados. Algoritmos desta categoria buscam um compromisso entre a eficiência e a precisão dos resultados da busca. Tais algoritmos assumem, por princípio, que as instâncias mais similares computadas podem não ser as mesmas identificadas pelo algoritmo *k*-NN, mas que os resultados obtidos compensam na prática o ganho de tempo no processo de busca.

Algoritmos da **categoria ENN** (*Exact Nearest Neighbor)* (e.g., Xueyi, 2012) objetivam acelerar o processo de busca do *k*-NN sem perder a precisão dos resultados, entendendo-se por "precisão" que tais algoritmos propõem processos de cálculo que resultam nos mesmos resultados que seriam obtidos com o algoritmo *k*-NN. Muitas propostas de algoritmos desta categoria realizam um pré-processamento do conjunto de instâncias como meio de melhorar a eficiência do processo de busca. Duas subcategorias de algoritmos ENN se destacam: a subcategoria dos algoritmos baseados em árvores e a subcategoria dos algoritmos baseados em grupos.

Os algoritmos da **subcategoria baseada em árvores** constroem árvores na fase de pré-processamento para mapear o espaço do conjunto de instâncias e, com isto, melhorar a eficiência da busca. A ideia é a de que o percurso na árvore construída na fase de pré-processamento acelere o processo de busca exata de instâncias na fase de busca. KD-tree (Sproull, 1991), Ball-tree (Uhlmann, 1991) e VP-tree (Yianilos, 1993) são exemplos de algoritmos pertencentes a esta subcategoria de algoritmos ENN.

Algoritmos da **subcategoria baseada em grupos** empregam métodos de agrupamento para particionar o espaço do conjunto de instâncias. Estes algoritmos funcionam em dois estágios: o estágio de agrupamento e o estágio de busca propriamente. No estágio de agrupamento, algoritmos de agrupamento – e.g., *k*-Means (MacQueen, 1967) – são utilizados para induzir grupos de instâncias dentro do conjunto de instâncias. A ideia é a de que o processo de busca possa ser acelerado via alguma **estratégia de exclusão** que objetiva limitar o número de grupos ou o número de instâncias dentro de grupos em que as instâncias devem ser buscadas, em vez de considerar todo o espaço compreendido pelo conjunto de instâncias. O conceito de **desigualdade triangular** (Lima, 2013) vem sendo utilizado para desenvolver estratégias de exclusão em propostas de algoritmos não hierárquicos da categoria ENN.

Este trabalho tem como foco algoritmos da subcategoria baseada em grupos da categoria ENN e que empregam o conceito de desigualdade triangular para estabelecer estratégias de exclusão para limitar o espaço de busca e, com isto, acelerar a fase de busca do algoritmo *k*-NN, a fase de busca para este trabalho se refere propriamente a busca do *k*-NN.

## 1.2 – Problema e Motivação

Na raiz do problema que motiva este trabalho está a alta complexidade computacional da fase de busca do algoritmo *k*-NN. Tal alta complexidade computacional é devida, principalmente, aos cálculos de função de similaridade entre instâncias.

A distância euclidiana (Anton, 1994; Theodoridis, & Koutroumbas, 2009) é comumente utilizada como uma métrica (Lima, 2013) para a função de similaridade empregada em diversas versões do algoritmo *k*-NN e, dada a sua prevalência, é empregada neste estudo. Quando ela é usada como medida de similaridade, quanto maior for a distância euclidiana entre duas instâncias, mais dissimilares essas instâncias são consideradas serem e, inversamente, quanto menor for a distância euclidiana entre duas instâncias, mais similares elas são consideradas serem. Para duas instâncias $p$ e $q$ no espaço $D$ dimensional a distância euclidiana entre elas é definida pela expressão (1.1)

$$d(p,q) = \sqrt{\sum_{d=1}^{D}(p_d - q_d)^2}\,, \qquad (1.1)$$

Em que $p_d$ e $q_d$ denotam a $d$-ésima dimensão das instâncias $p$ e $q$.

O cálculo da distância euclidiana entre duas instâncias requer $D$ subtrações, $D$ multiplicações e 1 (um) cálculo de raiz quadrada. Em um conjunto de instâncias de treinamento $T$, com $N$ instâncias de $D$ dimensões, dado como entrada do algoritmo $k$-NN, o cálculo da distância euclidiana entre uma instância $x$ e todas as instâncias do conjunto $T$ requer $DN$ subtrações, $(D-1)\,N$ somas e $D$ cálculos de raiz quadrada, o que conduz a uma complexidade computacional $O(DN)$.

Em um primeiro cenário em que o valor de $D$ é pequeno e $D$ é muito menor do que $N$ a complexidade computacional pode ser considerada, na prática, linear em relação a $N$, ou seja, $O(N)$. Em um segundo cenário em que $D = O(N)$ a complexidade computacional tende a ser $O(N^2)$. Em aplicações popularmente conhecidas como *big data*, em que $N$ é grande e, pior ainda, em que $N$ e $D$ são grandes, os dois cenários de complexidade computacional são inaceitáveis em termos de tempo de processamento. Este problema fomenta o desafio do desenvolvimento de propostas de algoritmos que visam acelerar a fase de busca do algoritmo $k$-NN.

## 1.3 – Objetivos e Contribuições

Este trabalho é contextualizado nos esforços de pesquisa que visam acelerar a fase de busca do algoritmo $k$-NN de forma exata, tendo como ideia a proposta de soluções que utilizam técnicas de agrupamento e o conceito de desigualdade triangular para desenvolver estratégias de redução de cálculos de funções de similaridade entre instâncias. Ou seja, utilizando a terminologia descrita na Seção 1.1, este trabalho é sobre pesquisas, que envolvem como soluções, algoritmos da categoria ENN, subcategoria baseada em grupos e o conceito de desigualdade triangular para desenvolver estratégias de exclusão de grupos ou de instâncias dentro de grupos.

Foram objetivos do trabalho de pesquisa conduzido e descrito nesta dissertação de mestrado:

- Estudar propostas de algoritmos da categoria ENN, subcategoria baseada em grupos, e que empregam o conceito de desigualdade triangular para estabelecer estratégias de exclusão de grupos ou de instâncias dentro de grupos.
- Propor um algoritmo deste mesmo contexto de contribuições de pesquisa, como contribuição alternativa às propostas existentes.

- Realizar experimentos com os algoritmos estudados e com o algoritmo proposto afim de verificar a eficiência destes algoritmos no que se refere ao aceleramento da fase de busca do algoritmo *k*-NN.

Como contribuições do trabalho de pesquisa realizado podem ser elencadas:

- A proposta do algoritmo *k*M++*k*NN, como alternativa às contribuições de pesquisa evidenciadas na literatura.
- Uma discussão crítica das propostas existentes em relação a elas mesmas e em relação à proposta do algoritmo *k*M++*k*NN.
- Disponibilização ao domínio público do software resultante do algoritmo *k*M++*k*NN neste trabalho.

## 1.4 – Organização deste Documento

Além deste capítulo introdutório, este documento está organizado como segue.

Os conceitos e as notações empregados neste trabalho são definidos no Capítulo 2, que objetiva organizar semanticamente os conceitos, via definições, e sintaticamente, via notações, os principais conteúdos tratados neste trabalho. O capítulo visita e adapta ao contexto deste trabalho vários fundamentos da área de Aprendizado de Máquina tais como instância, conjunto de instâncias, métrica e similaridade entre instâncias. O conceito de desigualdade triangular, utilizado por todas as propostas que serão apresentadas, é também discutido neste capítulo.

O Capítulo 3 objetiva apresentar e explicar os algoritmos que foram estudados e que forneceram sustentação para alcançar o objetivo deste trabalho. Os algoritmos estudados têm como objetivo a redução da quantidade de cálculos de distância, a fim de, acelerar o processo de busca do algoritmo *k*-NN que também é apresentado no Capítulo 3.

O Capítulo 4 apresenta a proposta de algoritmo desenvolvido como contribuição deste trabalho, o algoritmo *k*M++*k*NN. O texto aborda métodos de agrupamento para particionar o espaço do conjunto de instâncias e as hipóteses que a partir de estudos experimentais aceleram a fase de busca do *k*-NN.

O Capítulo 5 define a metodologia de pesquisa empregada neste trabalho. No capitulo são abordados o ambiente para experimentação que foi utilizado, os conjuntos de instâncias

selecionados para realização dos experimentos, os índices de desempenho[4] dos algoritmos e o método de experimentação.

O Capítulo 6 descreve os resultados obtidos dos experimentos realizados, e faz uma discussão entre os resultados do algoritmo proposto nesse trabalho e algoritmos considerados rápidos que se situam da mesma categoria, isto é, algoritmos ENN, subcategoria baseada em grupos e que utilizam o conceito de desigualdade triangular para acelerar a fase de busca do algoritmo *k*-NN.

O Capítulo 7 apresenta as conclusões deste trabalho e identifica trabalhos futuros.

O Apêndice I contém informações sobre como é possível obter o código fonte, em Python, do algoritmo *k*M++*k*NN.

[4] Desempenho faz referencia exclusivamente a complexidade de tempo do algoritmo.

# Capítulo 2
# Conceitos Fundamentais e Notações

*Este capítulo descreve uma série de conceitos fundamentais a este trabalho, bem como estabelece as notações utilizadas na formalização destes conceitos. O capítulo inicia com a definição e o estabelecimento da notação a ser utilizada para a descrição conjunto de instâncias (Seção 2.1). Medidas de similaridade entre instâncias são necessárias para que, entre outras coisas, se possa discriminar, no conjunto de instâncias, instâncias mais semelhantes e menos semelhantes umas às outras. Medidas de similaridade são, normalmente, definidas a partir de métricas em um espaço métrico. A Seção 2.2 discute os conceitos de espaço métrico e de métrica. Uma métrica especial, a distância euclidiana, é definida na Seção 2.3. O conceito de medida de similaridade entre instâncias empregado neste trabalho é definido na Seção 2.4. Um conceito central a este trabalho é o de desigualdade triangular detalhado na Seção 2.5, e utilizado nos algoritmos estudados e no algoritmo proposto neste trabalho.*

## 2.1 – Conjunto de Instâncias

Em Aprendizado de Máquina o conceito de instância tem sido utilizado para se referir a algo utilizado para representar, sobre uma certa perspectiva, um objeto ou um fenômeno no mundo. Frequentemente uma instância é descrita por um vetor de atributos, sendo que cada atributo[5] representa uma propriedade ou característica mensurável do objeto ou fenômeno representado pela instância. Os conceitos de conjunto de instâncias, instância e atributo são os mais básicos empregados neste trabalho, sendo expressos formalmente pela Definição 2.1.

> **Definição 2.1 (Conjunto de Instâncias, Instância e Atributo)**: um conjunto de instâncias é um conjunto finito e não vazio de $N$ instâncias, $T = \{ I_1, I_2, ... I_N \}$. Cada instância $I_i$, $1 \leq i \leq N$, é descrita por uma *n-upla* ordenada finita e não vazia de $D \geq 1$ dimensões, $I_i = ( Ai_1, Ai_2, ..., Ai_D )$. Cada A$ij$, $1 \leq j \leq D$ é um atributo cujo valor é descrito por um número do conjunto dos números reais.

A Definição 2.1 utiliza conceitos e notações de conjunto e de *n-uplas* ordenadas da forma como usualmente são empregados na Matemática. Assim, o termo **conjunto** refere-se a uma coleção de elementos distintos, sendo os elementos do conjunto separados por vírgula entre chaves. A ordem dos elementos em um conjunto não é importante, i.e., o conjunto {a, t, o, r} é igual ao conjunto {r, o, t, a}. O termo ***n-upla* ordenada** refere-se a uma coleção ordenada de elementos, sendo os elementos da *n-upla* ordenada separados por vírgula entre parêntesis ou parêntesis angulares, i.e., "( )" ou "< >". Por exemplo, mesmo tendo os mesmos elementos, a *n-upla* ordenada (1, 2, 1, 3) é diferente da *n-upla* ordenada (1, 1, 2, 3) porque a ordem dos elementos nas duas coleções é diferente. Pode haver repetição de elementos em uma *n-upla* ordenada.

Um atributo tem por objetivo mensurar algum aspecto do objeto ou do fenômeno representado por uma instância. Este trabalho está restrito a atributos que podem ser mensurados e expressos por valores do conjunto dos números reais. A Definição 2.1, acima, formaliza esta restrição.

O Exemplo 2.1 ilustra um conjunto de instâncias nos termos da Definição 2.1.

> **Exemplo 2.1**. O conjunto de dados *Iris,* disponível no *UCI Machine Learning Repository* (Dua & Graff, 2021) é um exemplo de Conjunto de Instâncias. O conjunto

[5] Alguns textos utilizam o termo "característica" ou o termo "variável" no lugar do termo "atributo".

de dados *Iris* refere-se a uma família de plantas floríferas e tem sido amplamente utilizado na área de Aprendizado de Máquina em experimentos sobre aprendizado supervisionado, agrupamento, redução de dimensionalidade, entre outros. O conjunto todo possui 150 instâncias e cada instância é descrita pelos quatro atributos reais correspondentes, em centímetros, ao comprimento da sépala (*sepal length*), largura da sépala (*sepal width*), comprimento da pétala (*petal length*) e largura da pétala (*petal width*) de plantas da família *Iris*. Cada instância do conjunto está associada a uma das seguintes três classes, que se refere a tipos diferentes de *Iris*: *Iris Setosa*, *Iris Versicolour* e *Iris Virginica*. Neste exemplo, a classe está sendo representada por um atributo real utilizando a seguinte correspondência[6]: 1.0 para *Iris Setosa*, 2.0 para *Iris Versicolour* e 3.0 para *Iris Virginica.* Nos termos da Definição 2.1 pode-se dizer que o conjunto de instâncias exemplificado possui número de instâncias $N = 150$, cada uma delas descritas por $D = 5$ atributos. A primeira *n-upla* ordenada ilustrada na Figura 2.1, descreve uma instância com os seguintes atributos: comprimento da sépala igual a 5.1 cm, largura da sépala igual a 3.5 cm, comprimento da pétala igual a 1.4 cm, largura da pétala igual a 0.2 cm e a classe a que se refere a instância igual a 1.0 (*Iris-setosa*). As demais instâncias ilustradas na Figura 2.1 podem ser interpretadas de forma análoga.

$T$ = { (5.1, 3.5, 1.4, 0.2, 1.0 ), (4.9, 3.0, 1.4, 0.2, 1.0 ), ... , (5.9, 3.0, 5.1, 1.8, 3.0) }

**Figura 2.1:** O conjunto de dados Iris, disponível no *UCI Machine Learning Repository* (Dua & Graff, 2021), como um exemplo de conjunto de instâncias, nos termos da Definição 2.1. O conjunto de instâncias Iris possui 150 instâncias, mas por razões ligadas a espaço e suficiência explanatória, a Figura 2.1 apresenta apenas três instâncias.

## 2.2 – Espaços Métricos

Em Matemática, o conceito de espaço métrico reúne o conceito de conjunto e o conceito de métrica sobre os elementos do conjunto. A métrica é uma função que estabelece um valor real entre dois quaisquer elementos de um conjunto. O conceito de espaço métrico é muito útil em Aprendizado de Máquina e especialmente útil para este trabalho para estabelecer uma noção de similaridade ou dissimilaridade entre duas instâncias.

[6] Os números reais estão sendo apresentados neste trabalho tendo o ponto (não a vírgula) como separador entre a parte inteira e a fracionária.

A Definição 2.2 é uma formalização do conceito de espaço métrico adaptada de Lima (2013) aos propósitos deste trabalho.

**Definição 2.2** (**Espaço Métrico**): um espaço métrico é um par ordenado $< X, d >$, em que $X$ é um conjunto e $d$ é uma métrica sobre os elementos de $X$, i.e., $d$ é uma função $d: X \times X \to \mathbb{R}$, tal que, para quaisquer três elementos $x_1, x_2, x_3 \in X$, os seguintes axiomas são válidos, em que : $\mathbb{R}$ é o conjunto dos números reais.

1. *Reflexividade ou identidade*: $d(x_1, x_2) = 0 \leftrightarrow x_1 = x_2$.
2. *Simetria*: $d(x_1, x_2) = d(x_2, x_1)$.
3. *Desigualdade triangular:* $d(x_1, x_2) \leq d(x_1, x_3) + d(x_2, x_3)$.

Decorre desta definição que uma métrica é sempre positiva ou nula[7], i.e., para quaisquer $x_1, x_2 \in X$ ocorre que $d(x_1, x_2) \geq 0$. A métrica $d$ é comumente chamada de "função distância" ou, simplesmente de "distância".

O Axioma 1 determina que, por um lado, se a distância entre os elementos $x_1$ e $x_2$ é igual a zero então $x_1$ é igual a $x_2$ e, por outro lado, se $x_1$ é igual a $x_2$ então a distância entre $x_1$ e $x_2$ é igual a zero. Ou seja, nestas condições, $x_1$ é idêntico a $x_2$, daí o termo identidade ser utilizado para qualificar esse axioma.

O Axioma 2 estabelece que, para quaisquer dois elementos $x_1$ e $x_2$, a distância de $x_1$ até $x_2$ é igual à distância de $x_2$ até $x_1$. Ou seja, a função distância admite a propriedade simétrica da relação entre conjuntos.

O Axioma 3, conhecido como desigualdade triangular, estabelece que a distância entre dois elementos $x_1$ e $x_2$ é sempre menor ou igual à soma da distância deles para um terceiro elemento $x_3$. Informalmente, pode-se dizer que o axioma da desigualdade triangular estabelece que $x_3$ não é um atalho para o caminho que conecta $x_1$ e $x_2$. Pela importância da desigualdade triangular para este trabalho, a Seção 2.6 aborda o axioma com mais detalhes.

[7] O argumento de que $d(x_1, x_2) \geq 0$. decorre dos seguintes fatos: **(1)** $d(x_1, x_3) \leq d(x_1, x_2) + d(x_2, x_3)$, o que implica que $d(x_1, x_2) \geq d(x_1, x_3) - d(x_2, x_3)$. **(2)** $d(x_2, x_3) \leq d(x_1, x_2) + d(x_1, x_3)$, o que implica que $d(x_1, x_2) \geq d(x_2, x_3) - d(x_1, x_3)$. As implicações dos fatos **(1)** e **(2)** permite concluir que $d(x_1, x_2) \geq | d(x_1, x_3) - d(x_2, x_3) |$ e, finalmente, que $d(x_1, x_2) \geq 0$.

**Exemplo 2.2**. *Exemplos de espaços métricos*. Os exemplos a seguir constituem espaços métricos envolvendo o conjunto dos números reais e vetores de elementos do conjunto dos números reais, em que a métrica de distância utilizada é a euclidiana:

(1) $< \mathbb{R}, d >$, em que a distância (métrica) *d* entre dois elementos de $\mathbb{R}$ é dada por $d(v, w) = \sqrt{(v - w)^2} = |v - w|$.

(2) $< \mathbb{R}^2, d >$, em que a distância *d* entre dois elementos de $\mathbb{R}$ é dada por $d\big((v_1, v_2), (w_1, w_2)\big) = \sqrt{(v_1 - w_1)^2 + (v_2 - w_2)^2}$ .

(3) $< \mathbb{R}^D, d >$, $D \geq 1$, em que a distância *d* entre dois elementos de $\mathbb{R}$ é dada por $d\big((v_1, \ldots, v_D), (w_1, \ldots, w_D)\big) = \sqrt{(v_1 - w_1)^2 + \cdots + (v_D - w_D)^2}$.

Os espaços métricos exemplificados envolvem diferentes dimensões do conjunto dos números reais, sendo os espaços (1), (2) e (3), respectivamente, unidimensional, bidimensional e *D*-dimensional – note que (1) e (2) são casos particulares de (3).

Informalmente, o gráfico da Figura 2.2(a) representa o espaço métrico (1). Sendo unidimensional, o espaço métrico (1) está sendo representado graficamente como uma linha com infinitos números reais. A figura também ilustra a distância entre os elementos $v = 2.5$ e $w = 3.6$, calculada como $d(2.5, 3.6) = \sqrt{(2.5 - 3.6)^2} = |2.5 - 3.6| = 1.1$. O gráfico da Figura 2.2(b) representa o espaço métrico (2) via um plano de infinitos números reais em cada uma das duas dimensões, nomeadas no gráfico pelos números **1** e **2**. A figura também representa a distância entre os vetores (1.0, 3.0) e (2.0, 1.0), ou seja, $d\big((1.0, 3.0), (2.0, 1.0)\big) = \sqrt{(1.0 - 2.0)^2 + (3.0 - 1.0)^2} \cong 2.24$ . Os vetores estão representados como pontos no plano e a distância entre eles via uma a linha contínua que os conecta. As linhas pontilhadas fazem alusão aos catetos do triângulo retângulo cuja hipotenusa é a distância *d*.

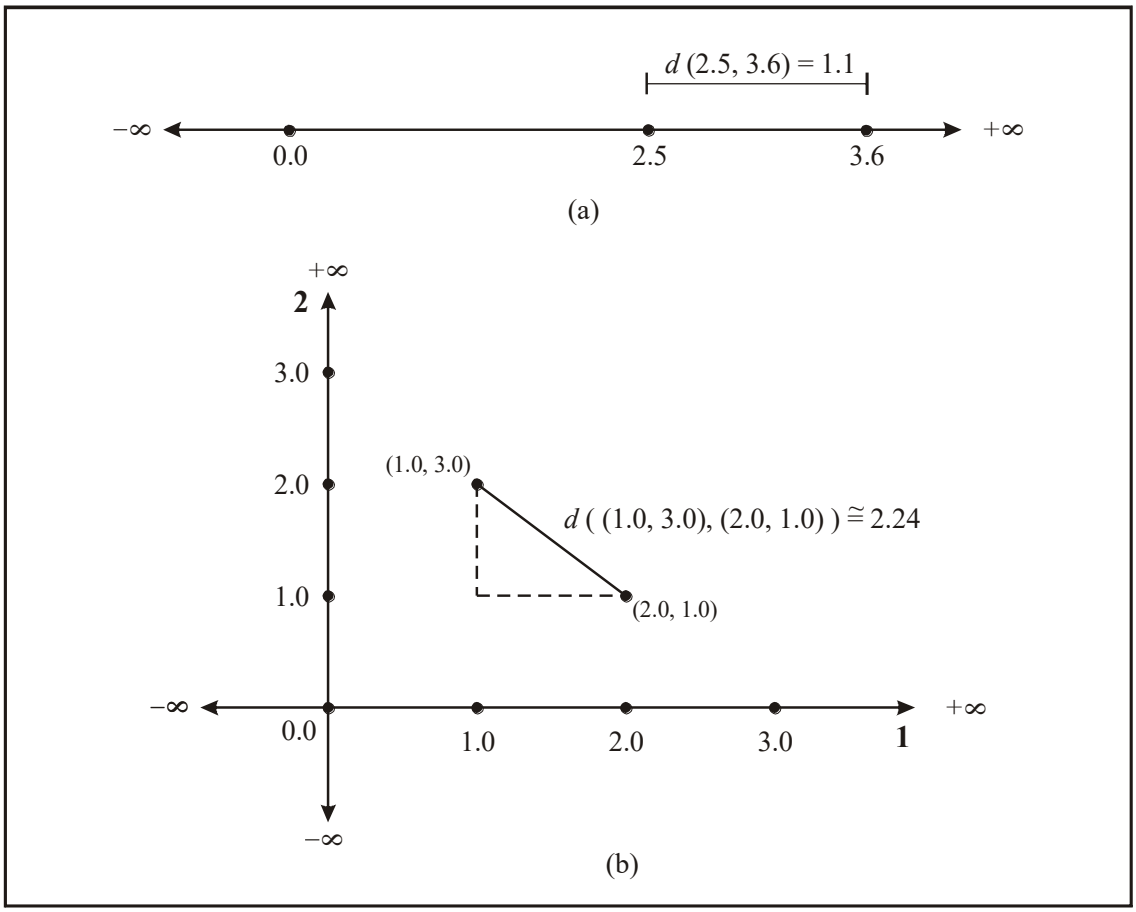



**Figura 2.2:** Representação gráfica dos dois primeiros espaços métricos do Exemplo 2.2: (a) espaço métrico unidimensional; (b) espaço métrico bidimensional.

Existem representações gráficas intuitivas para espaços métricos de até três dimensões. Espaços métricos *D*-dimensionais em que $D \geq 4$ podem ser representados graficamente, mas as pessoas não interpretam tais gráficos com a mesma naturalidade com que interpretam representações de até três dimensões. A título de exemplo, independentemente de representação gráfica para o espaço métrico (3), a distância *d* entre os vetores (1.1, 2.2, 1.0, 0.0) e (-1.1, 3.0, 1.8, 0.5) no espaço métrico $< \mathbb{R}^4, d >$ é dada por $d\big((1.1, 2.2, 1.0, 0.0), (-1.1, 3.0, 1.8, 0.5)\big) = \sqrt{(1.1 - (-1.1))^2 + (2.2 - 3.0)^2 + (1.0 - 1.8)^2 + (0.0 - 0.5)^2} \cong 2.52$ .

## 2.3 – Distância Euclidiana entre Instâncias

As distâncias dos espaços métricos do Exemplo 2.2 são conhecidas como **distâncias euclidianas** para espaços de dimensões $D \geq 1$. Estas distâncias representam o tamanho em linha reta entre dois vetores do espaço métrico. Embora outras definições de distância

possam ser utilizadas − e.g. distância Manhattan (Black, 2019), distância Mahalanobis (Maesschalck, Jouan-Rimbaud & Massart, 2000) − este trabalho usa apenas o conceito de distância euclidiana, que está estabelecido na Definição 2.3.

**Definição 2.3** (**Distância Euclidiana entre Instâncias**): Seja $T$ um conjunto de instâncias e duas instâncias $v \in T$ e $w \in T$, $v = (v_1, v_2, \ldots, v_D)$, $w = < (w_1, w_2, \ldots, w_D), D \geq 1$. A distância euclidiana entre as instâncias $v$ e $w$ é igual a $d(v, w) = \sqrt{(v_1 - w_1)^2 + \cdots + (v_D - w_D)^2} = \sqrt{\sum_{j=1}^{D}(v_j - w_j)^2}$.

Na notação utilizada na Definição 2.3, $v$ e $w$ são instâncias pertencentes a um conjunto de instâncias $T$. Sendo instâncias, $v$ e $w$ são descritos, respectivamente, por *n-uplas* ordenadas ($v_1$, $v_2$, ..., $v_D$) e ($w_1$, $w_2$, ..., $w_D$)

**Exemplo 2.3**. *Um exemplo de distância euclidiana entre instâncias.* Sejam $v$ = (5.1, 3.5, 1.4, 0.2) e $w$ = (4.9, 3.0, 1.4, 0.2). Nesse exemplo, a quantidade de atributos ou dimensões é $D$ = 4

A distância euclidiana entre as instâncias $v$ e $w$ é igual a $d(v, w) = \sqrt{(v_1 - w_1)^2 + \cdots + (v_4 - w_4)^2} = \sqrt{\sum_{j=1}^{4}(v_j - w_j)^2}$.

$$d(v, w) = \sqrt{(5.1 - 4.9)^2 + (3.5 - 3.0)^2 + (1.4 - 1.4)^2 + (0.2 - 0.2)^2}$$
$$= \sqrt{0.29} \cong 0.539$$

Ou seja, a distância euclidiana entre as instâncias $v$ = (5.1, 3.5, 1.4, 0.2) e $w$ = (4.9, 3.0, 1.4, 0.2) é, aproximadamente, igual a 0.539.

## 2.4 – Similaridade e Dissimilaridade entre Instâncias

Medidas de similaridade são utilizadas para indicar o grau de semelhança entre instâncias. Geralmente funções de similaridade são desenvolvidas para estabelecer a similaridade entre instâncias de um conjunto de instâncias, sendo que a função de similaridade empregada neste trabalho é estabelecida a partir do conceito de distância euclidiana entre instâncias. A escolha da distância euclidiana como medida de similaridade entre instâncias a ser utilizada neste trabalho tem a ver com a prevalência do uso desta medida em diversas versões do algoritmo *k*-NN.

A expressão 2.1 estabelece que a função de similaridade *sim* a ser empregada neste trabalho é dada por

$$sim(v, w) = d(v, w), \qquad (2.1)$$

em que $d(v, w)$ é a distância euclidiana entre as instâncias $v$ e $w$, calculada de acordo com a Definição 2.3. Neste trabalho, ambas funções $sim(v, w)$ e $d(v, w)$ denotam a similaridade entre duas instâncias $v$ e $w$, de tal forma que $d(v, w)$ denota, além da própria distância, também a similaridade entre as instâncias $v$ e $w$. Sendo assim definido, quanto menor for a distância euclidiana entre duas instâncias, mais similares essas instâncias são consideradas serem e, inversamente, quanto maior for a distância euclidiana entre duas instâncias, mais dissimilares elas são consideradas serem.

## 2.5 – Desigualdade Triangular

O conceito de desigualdade triangular tem origem na geometria plana euclidiana. Em qualquer triângulo, o comprimento de qualquer um de seus lados é menor ou igual à soma dos comprimentos dos outros dois lados. Ou seja, sendo $a$, $b$ e $c$ os comprimentos dos lados de um triângulo, $c \leq a + b$ (Figura 2.3-a) [8], $a \leq b +$ c e $b \leq b +$ c. Esse conceito é a formalização da noção intuitiva de que, sendo $x$, $v$ e $w$ os vértices de um triângulo (Figura 2.3-a), é mais curto ou igual o tamanho do caminho reto de $x$ a $w$ do que o tamanho do caminho de $x$ a $v$ somado com o tamanho do caminho de $v$ a $w$.

A distância euclidiana $d$ (Definição 2.3) entre instâncias de um conjunto de instâncias ($T$) $D$-dimensional é uma métrica no espaço métrico $< \mathbb{R}^D, d >$,. Logo $d$ satisfaz a desigualdade triangular entre instâncias $x$, $v$ e $w$ pertencentes a $T$ mostrada na expressão (2.2)

$$d(x, w) \leq d(x, v) + d(v, w)\,, \qquad (2.2)$$

ou seja, a distância euclidiana entre a instância $x$ e a instância $w$ é menor ou igual à distância euclidiana da instância $x$ à instância $v$ somada com a distância euclidiana da instância $v$ à instância $w$ (Figura 2.3-b).

[8]Alguns matemáticos, especialmente quando estão trabalhando com geometria elementar, preferem não considerar na formulação desse conceito, a igualdade entre $c$ e a soma de $a$ com $b$, porque nessa condição o triângulo degenera-se a uma linha. Assim, tais matemáticos optam por escrever que $c < a + b$, excluindo o caso da igualdade. Diferentemente deles, neste trabalho, considera-se a igualdade na formulação do conceito.

É também verdade que

$$d(x,v) \leq d(x,w) + d(v,w) \text{ e que}$$

$$d(v,w) \leq d(x,v) + d(x,w) ,$$

o que conduz a

$$d(x,w) \geq d(x,v) - d(v,w) \text{ e}$$

$$d(x,w) \geq d(v,w) - d(x,v).$$

Resumindo as duas últimas expressões em uma única expressão, obtêm-se

$$d(x,w) \geq | d(x,v) - d(v,w) | , \qquad (2.3)$$

ou seja, a distância euclidiana entre a instância *x* e a instância *w* é maior ou igual ao valor absoluto da diferença entre a distância euclidiana da instância *x* à instância *v* e a distância euclidiana entre a instância *v* e a instância *w*.

As expressões (2.2) e (2.3) estabelecem um limite inferior e um limite superior para a distância euclidiana entre as instâncias *x* e *w*, desde que se conheça a distância euclidiana entre as instâncias *x* e *v* e a distância euclidiana entre as instâncias *v* e *w*. Resumidamente, estes limites são:

- *Limite inferior para a distância euclidiana* $d(x,w)$:

$$| d(x,v) - d(v,w) | .$$

- *Limite superior para a distância euclidiana* $d(x,w)$:

$$d(x,v) + d(v,w) .$$

Estes limites são úteis para o desenvolvimento de algoritmos que objetivam diminuir o número de cálculos de distâncias euclidianas entre instâncias. Os algoritmos estudados neste trabalho utilizam estes limites ou alguma relação derivada destes limites.

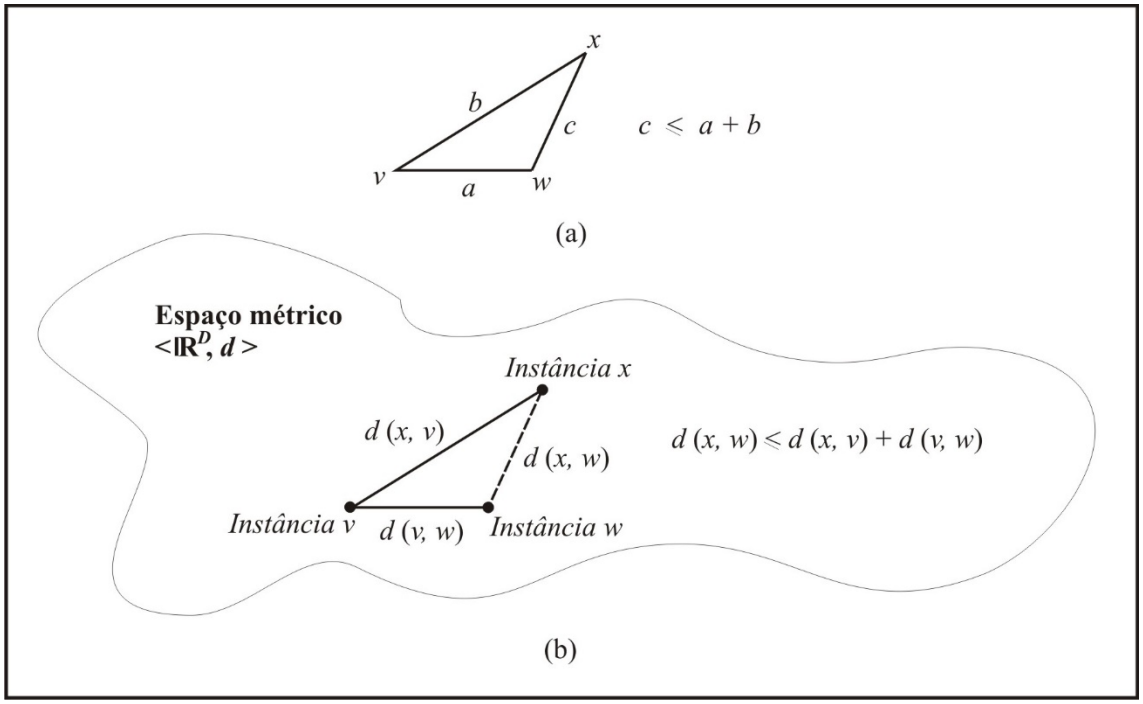



**Figura 2.3:** Desigualdades triangulares: (a) em um triângulo; (b) em distâncias euclidianas entre instâncias de um conjunto de instâncias no espaço real *D*-dimensional.

# Capítulo 3
# Revisão Bibliográfica

*Este capítulo descreve a revisão bibliográfica referente aos algoritmos que objetivam acelerar o processo de busca do algoritmo k-NN. A Seção 3.1 apresenta o algoritmo k-NN e seu funcionamento para o processo de busca das instâncias em um conjunto de instâncias. A Seção 3.2 tem por objetivo apresentar a complexidade computacional e a problemática apresentada para o algoritmo k-NN quando utilizando um conjunto com alto número de instâncias. A Seção 3.3 apresenta o algoritmo kMkNN, uma das propostas desenvolvidas por Xueyi* (2012)*, a fim de acelerar o processo de busca do algoritmo k-NN, para isto é empregado o conceito de desigualdade triangular, objetivando reduzir a quantidade de cálculos de distância realizados pelo k-NN. A Seção 3.4 apresenta um algoritmo, proposto em Pan et al.* (2020)*, denominado KMC-OTI-FS, que utiliza outra estratégia de emprego da desigualdade triangular. A Seção 3.5 apresenta uma segunda proposta, também apresentada em Pan et al.* (2020)*, o algoritmo KMC-EOTI-FS, o qual visa corrigir uma deficiência na complexidade computacional de espaço do algoritmo KMC-OTI-FS.*

## 3.1 – Algoritmo *k*-NN

O algoritmo *k*-NN *(k-Nearest Neighbors),* também conhecido como *k* vizinhos mais próximos, é comumente utilizado em diversas áreas como: reconhecimento de padrões, aprendizado de máquina e mineração de dados (Mitchell, 1997). O algoritmo *k*-NN, neste texto também referenciado como, tradicional, pode ser chamado também de *"aprendiz preguiçoso",* isto por que o estágio de inicialização tem um baixo custo computacional.

O estágio de inicialização tem por objetivo preparar o conjunto de instâncias, de tal forma que, as instâncias estejam melhor organizadas, de maneira que otimize o estágio de busca, neste caso, o *k*-NN tradicional não realiza uma preparação do conjunto de instâncias, a fim de otimizar o estágio de busca.

O *k*-NN envolve um processo de busca de *k* instâncias mais similares à instância *x* em um conjunto *T* de $N \geq 1$ instâncias definidas por $D \geq 1$ dimensões.

O estágio de pesquisa, ou, busca propriamente dito, é utilizado para buscar as *k* instâncias mais similares em relação a instância de busca fornecida como entrada, com isto, o algoritmo irá percorrer cada uma das instâncias do conjunto de instâncias, a fim de, verificar a similaridade entre cada uma das instâncias em relação à instância de busca.

O estágio de busca deste algoritmo possuí alto custo computacional, uma vez que, o algoritmo irá acessar todas as instâncias do conjunto de instâncias para computar suas medidas de distância em relação à instância de busca, que de modo geral utiliza-se euclidiana como medida de distância.

Tendo como entrada um conjunto de instâncias *T*, uma instância *x* e um inteiro $k \geq 1$, este algoritmo tem como objetivo buscar às *k* instâncias mais próximas de *x* em *T*.

A Figura 3.1 ilustra isto, assim, o algoritmo irá executar a busca a fim de encontrar as *k* instâncias mais próximas de *x* em *T*, promovendo assim como saída os índices das *k* instâncias próximas de *x,* temos o pseudocódigo do algoritmo *k*-NN, que subsidiara a compreensão do seu funcionamento na seção seguinte.

```
Algoritmo k-NN (T, N, x, k)
    Entrada: T, um vetor armazenando um conjunto de instâncias no espaço
    D dimensional, N, o número de instâncias em T, x, uma instância de
    pesquisa e o número de k ≥ 1 vizinhos mais próximos de x em T.
    Saída: R, sequência dos índices das k instâncias mais próximas de x
    em T, ordenada a partir da mais próxima.
1   {
2       para i := 1 até N faça
3       {
4           // V é um vetor de pares (i, d) em que i indica o índice de
            // uma instância i em T e d a distância da
5           // instância T[i] à instância x. O algoritmo
            // Distância_Euclidiana calcula a distância Euclidiana
6           // da instância T[i] à instância x no espaço D dimensional.
7           V[i] := (i, Distância_Eucliciada (T[i], x, D) );
8       }
9       Ordenar em ordem crescente, pelo campo d, as distâncias
        armazenadas em V;
10      R := o índice das k primeiras instâncias em V;
11      retornar R;
12  }
```

**Figura 3.1:** Pseudocódigo do algoritmo *k*-NN.

## 3.2 – Complexidade Computacional do *k*-NN e a Problemática

Com base no estudo e compreensão do algoritmo *k*-NN, um problema deste algoritmo é a sua execução para um conjunto com grande número de instâncias ou instâncias com grande dimensão.

Em um conjunto de instâncias de treinamento $T$, com $N$ instâncias de $D$ dimensões, dado como entrada ao algoritmo *k*-NN, o cálculo da distância euclidiana entre uma instância $x$ e todas as instâncias do conjunto $T$ requer $DN$ subtrações, $(D-1)\,N$ somas e $D$ cálculos de raiz quadrada, o que conduz a uma complexidade computacional $O(DN)$. Em um primeiro cenário onde $D$ é pequeno e $D$ é muito menor do que $N$ a complexidade computacional pode ser considerada, na prática, linear em relação a $N$, ou seja, $O(N)$. Em um segundo cenário onde $D = O(N)$ a complexidade computacional tende a ser $O(N^2)$. Em aplicações popularmente conhecidas como *big data*, onde $N$ é grande e, pior ainda, onde $N$ e $D$ são grandes, os dois cenários de complexidade computacional são inaceitáveis em termos de tempo de processamento.

Compreendendo a grande complexidade computacional que o *k*-NN pode obter quando lida com um conjunto com grande número de instâncias, na literatura podem ser encontradas propostas com foco na redução do número de cálculos de distâncias durante o processo de busca do *k*-NN. As próximas seções apresentaram três destas propostas.

### 3.3 – Algoritmo *k*M*k*NN

A ideia básica do algoritmo *k*M*k*NN[9] proposto em Xueyi (2012) é pré-processar um conjunto de instâncias *T* com o objetivo de acelerar o processo de busca. Assim o *k*M*k*NN possui duas fases denominadas de Fase 1, fase *offline* ou de pré-processamento, e a Fase 2, fase *online* ou fase de busca propriamente. Na Fase 1 o algoritmo de agrupamento *k*-Means (MacQueen, 1967) é utilizado para induzir um agrupamento com $M = \sqrt{N}$ grupos[10] sobre o conjunto de instâncias *T*, onde *N* é o número de instâncias de *T*.

Na Fase 2 o conceito de desigualdade triangular (Lima, 2013) é utilizado como estratégia para economizar cálculos de distâncias euclidianas e, com isto, acelerar o processo de busca de *k* instâncias em *T* mais similares a uma instância *x* dada como entrada na Fase 2. A Figura 3.2 ilustra o uso da desigualdade triangular no contexto do algoritmo *k*M*k*NN. A figura apresenta um grupo de instâncias $C_j$ cujo centro é $c_j$ e o raio é $r_j$, raio é a distancia do centro do grupo até a instância mais distante do grupo. A figura também apresenta o conjunto $y_k^{NN}$ de *k* instâncias correntemente[11] mais similares (mais próximas) à instância *x* e o raio $d_k$ deste conjunto.

[9] O algoritmo *k*M*k*NN, recebe um nome diferente por Pan *et al*. (2020), sendo chamado de KMC-TI-FS.
[10] Pan *et al*. (2020) afirmam que *M* foi definido como sendo igual $\sqrt{N}$ a partir de estudos experimentais, embora não apresentem os resultados destes estudos.
[11] O termo "correntemente" está sendo utilizado aqui porque o algoritmo *k*M*k*NN compõe o conjunto $y_k^{NN}$ de forma iterativa.

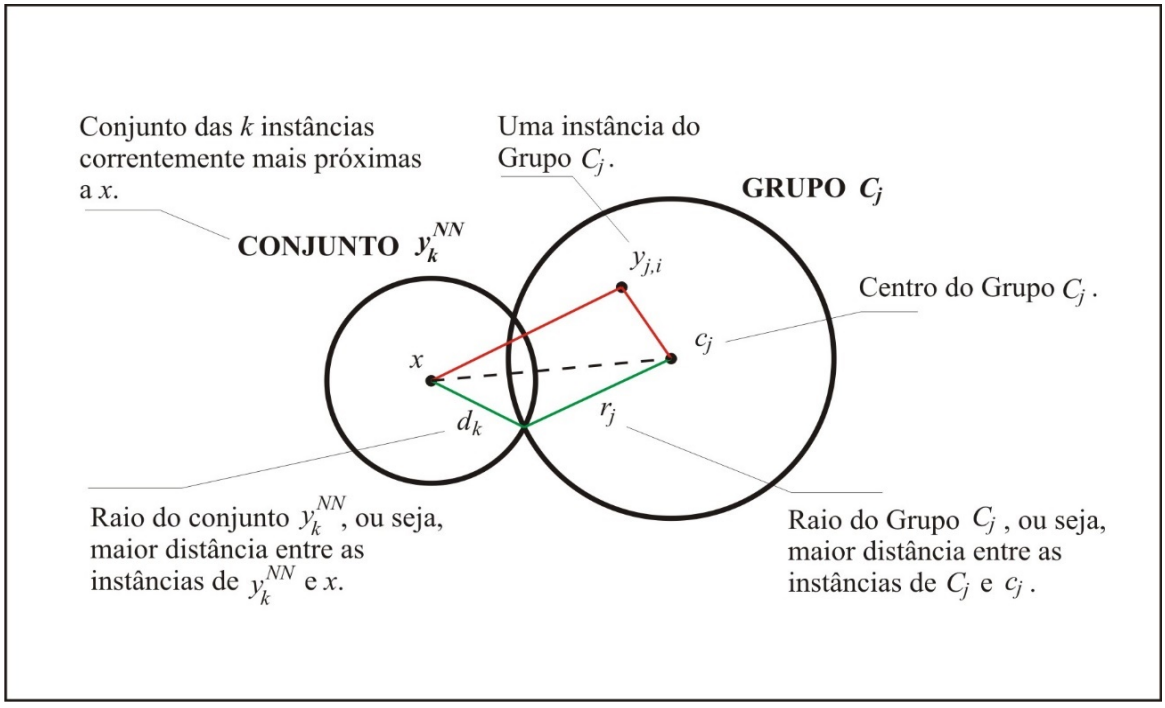



**Figura 3.2:** Uso do conceito de desigualdade triangular no contexto do algoritmo *k*M*k*NN.

O conceito de desigualdade triangular permite afirmar que

$$d_k + r_j \geq d(x, c_j),$$

o que implica em

$$d(x, c_j) - r_j \leq d_k. \qquad (3.1)$$

Se para um Grupo $C_j$ a expressão (3.1) for falsa, então não é necessário cálculos de distâncias entre $x$ e as instâncias do Grupo $C_j$, o que conduz à primeira fonte de economia de cálculos de distâncias empregada pelo algoritmo *k*M*k*NN.

Mesmo que a expressão (3.1) seja verdade para um Grupo $C_j$, ainda é possível uma segunda economia de cálculos de distância. Utilizando novamente o conceito de desigualdade triangular é possível afirmar que

$$d(x, y_{j,i}) \geq |d(x, c_j) - d(y_{j,i}, c_j)|. \qquad (3.2)$$

Se para uma certa instância $y_{j,i} \in C_j$ ocorrer de $|d(x, c_j) - d(y_{j,i}, c_j)| > d_k$ então pela expressão (3.2) $d(x, y_{j,i}) > d_k$ e, também nesta condição, a distância $d(x, y_{j,i})$ não necessita ser calculada. Esta é a segunda fonte de economia de cálculos de distâncias.

A Figura 2 apresenta o pseudocódigo do algoritmo *k*M*k*NN. Na Fase 1 o algoritmo *k*-Means é utilizado para induzir um agrupamento com $M$ grupos, $\{C_j\}_{j=1}^{M}$, em que cada grupo contém $N_j$ ($j = 1, ..., M$) instâncias, e armazenar dados pré-processados para a Fase 2, relativos aos centros dos grupos, $\{c_j\}_{j=1}^{M}$, aos raios dos grupos, $\{r_j\}_{j=1}^{M}$, e à distância de cada instância $y$ ao centro do grupo que a contém, $\{\{d(y_{j,i}, c_j)\}_{i=1}^{N_j}\}_{j=1}^{M}$.

A Fase 2, fase de busca de $k$ instâncias mais similares (mais próximas) de uma instância $x$, dada como entrada, é realizada da seguinte forma. Computa-se o Grupo $C_1^*$ mais próximo à instância $x$ (linhas 2 e 3 da Figura 3.3) e, exaustivamente, computa-se a distância de $x$ a cada uma das instâncias do Grupo $C_1^*$, com o objetivo de calcular provisoriamente o conjunto $\{y_s^{NN}\}_{s=1}^{k}$ das $k$ instâncias mais próximas a $x$ (linhas 4 a 7 da Figura 3.3). Até aqui, nenhuma economia de cálculos de distância é realizada. A economia virá com a iteração sobre os outros grupos utilizando a estratégia da desigualdade triangular descrita anteriormente (linhas 9 a 31 da Figura 3.3). Ao final da Fase 2 o algoritmo retorna como saída o conjunto $\{y_s^{NN}\}_{s=1}^{k}$ das $k$ instâncias mais similares a $x$.

**Algoritmo** *k*M*k*NN

**Fase 1: fase de pré-processamento (*offline*)**

**Entrada**: ***T***: conjunto de instancias; ***N***: número de instâncias em *T*; ***M***: número de grupos a serem formados em *T*, onde $M = \sqrt{N}$.

**Saída**: $\{C_j\}_{j=1}^{M}$.: *M* grupos de instâncias em *T*; $\{c_j\}_{j=1}^{M}$: *M* respectivos centros dos grupos de instâncias em *T*; $\{r_j\}_{j=1}^{M}$: *M* respectivos raios dos grupos de instâncias em *T*, onde o raio $r_j$ de um grupo $C_j$ é definido pela maior distância entre o centro do grupo $C_j$ e uma instância dentro do grupo $C_j$; $\{\{d(y_{j,i}, c_j)\}_{i=1}^{N_j}\}_{j=1}^{M}$:distância de cada instância *y* ao centro do grupo que a contém. Aqui $N_j$ representa o número de instâncias do grupo $C_j$.

**Procedimento**: algoritmo de agrupamento *k*-Means (MacQueen, 1967).

**Fase 2: fase de busca propriamente (*online*)**

**Entrada**: ***x***: instância à qual se busca, em *T*, instâncias mais próximas a ela; ***k***: número de instâncias similares que se quer buscar.

**Usa**: saídas da Fase 1 (*offline*).

**Saída**: $\{y_s^{NN}\}_{s=1}^{k}$: conjunto das *k* instâncias de *T* mais similares a *x*.

1 {

2 Computar as distâncias euclidianas $\{d(x, c_j)\}_{j=1}^{M}$.

```
3   Ordenar {d(x,c_j)}_{j=1}^{M} para obter {d(x,c_j^*)}_{j=1}^{M}, onde d(x,c_1^*) < d(x,c_2^*) < ··· < d(x,c_M^*).
4   Para o grupo mais próximo a x, C_1^*, computar as distâncias euclidianas
5   {d(x,y_{1,i})}_{i=1}^{N_1^*}, isto é, as distâncias de x a cada instância y do Grupo C_1^*.
6   Ordenar {d(x,y_{1,i})}_{i=1}^{N_1^*} para localizar as k instâncias correntes mais similares a x;
7   Inserir essas instâncias em {y_s^{NN}}_{s=1}^{n}, n = k, e fazer d_k = y_k^{NN}.
8   para j := 2 até M faça
9   {
10      se d(x,c_j^*) − r_j^* < d_k então // Primeira fonte de economia de cálculos
11        // de distância (Expressão 3.1). Aqui r_j^* é o raio do grupo C_j^*. Se
12        // este teste é falso, então não é necessário cálculos de
13        // distâncias entre x e as instâncias do Grupo C_j^*.
14      {
15         para i := 1 até N_j^* faça
16         {
17            se |d(x,c_j^*) − d(y_{j,i},c_j^*)| > d_k então
18              // Segunda fonte de economia de cálculos de distâncias
19              //(Expressão 3.2). Não é necessário o cálculo de d(x,y_{j,i}).
20            senão
21            {
22               Calcular a distância euclidiana d(x,y_{j,i}).
23               se d(x,y_{j,i}) < d_k então
24                  Fazer n = n + 1 e inserir y_{j,i} em {y_s^{NN}}_{s=1}^{n}.
25            }
26         }
27      }
28      Ordenar {d(x,y_s^{NN})}_{s=1}^{n}.
29      Reduzir {y_s^{NN}}_{s=1}^{n} para n = k instâncias mais próximas (mais similares).
30      Fazer d_k = (x,y_k^{NN}).
31  }
```

```
32      Retornar como saída {y_s^NN}_{s=1}^k.
33  }
```

**Figura 3.3:** Pseudocódigo algoritmo *k*M*k*NN.

O Exemplo 3.1 ilustra o funcionamento do algoritmo *k*M*k*NN.

**Exemplo 3.1.** Considere o conjunto de instâncias (*T*) mostrado na Figura 3.4, que possui 9 instâncias descritas por dois atributos numéricos, e será feita a busca das $k = 2$ instâncias mais similares à instância *x* de entrada.

A Figura 3.4 apresenta as instâncias utilizadas para ilustrar o funcionamento do *k*M*k*NN.

*T* = { (1.0, 2.0), (2.0, 3.0), (5.0, 1.0), (9.0, 5.0), (8.0, 6.0), (9.0,7.0), (1.0, 6.0), (2.0, 7.0), (3.0, 5.0) }

**Figura 3.4:** Conjunto de instâncias utilizado para ilustrar o funcionamento do algoritmo *k*M*k*NN.

Compreendendo o conjunto de instâncias que será utilizado neste exemplo, considera-se uma instância *x,* da qual será utilizada para buscar as *k* instâncias mais similares a ela,

$$x = (3.0, 3.0).$$

Na Fase 1 de execução do *k*M*k*NN, considere que o algoritmo *k*-Means induziu o agrupamento ilustrados na Figura 3.5. A quantidade de grupos que serão gerados é igual a $M = \sqrt{N}$, ou seja, $M = \sqrt{9} = 3$,

A Figura 3.5 mostra o agrupamento induzido pelo *k*-Means, com $M = 3$ grupos.

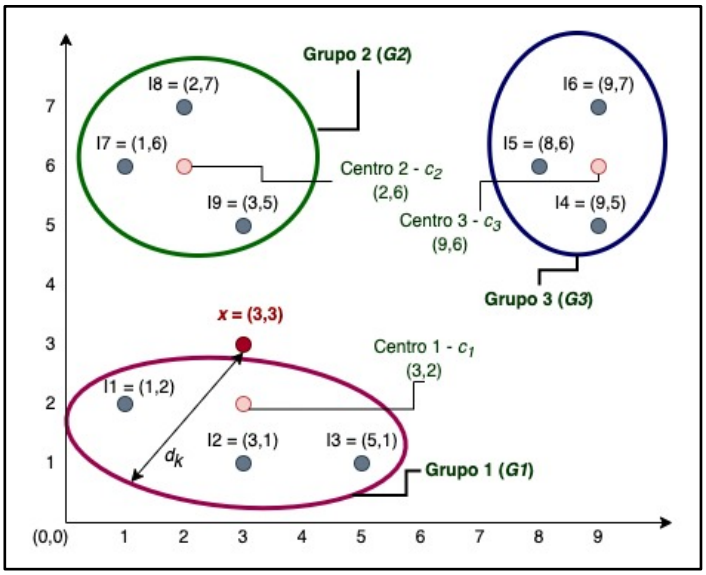



**Figura 3.5:** Agrupamento induzido pelo algoritmo *k*-Means com três grupos.

No início da Fase 2 do *k*M*k*NN, será computado a distância de *x* para o centro de cada um dos grupos, as distâncias computadas são ordenadas de forma ascendente, desta forma, via de exemplo tem-se os grupos mais próximos, considerando a seguinte ordem aos centros mais próximos da instância *x* (linhas 2 e 3).

$$\{d(x, c_j^*)\}_{j=1}^{M} = \{[1.0, 3.16, 6.7]\}$$

Na sequencia é apresentada a Tabela 3.1 com as distâncias de *x* aos centros dos grupos que compõem o agrupamento.

**Tabela 3.1:** Distâncias calculadas e ordenadas, de *x* a cada centro de grupo, onde *G1* é o primeiro grupo, *G2* o segundo grupo e *G3* o terceiro grupo.

| **Distâncias para o centro do grupo** | ***G1*** | ***G2*** | ***G3*** |
|---|---|---|---|
| ***x*** | 1.0 | 3.16 | 6.7 |

Deste modo, se tem os grupos mais similares à instância *x*, o que propicia identificar as instâncias mais próximas, sem a necessidade de percorrer todos os grupos.

Deve-se computar as distâncias de *x* para cada instância de *G1* (linha 4 e 5), e ordenar as distâncias de forma ascendente para localizar as instâncias mais similares à *x* (linha 6). Em seguida armazenar a instâncias mais similares e considerar $d_k$ como a distância à sendo a instância mais distante de *x* (linha 7).

Possuindo ou não as *k* instâncias mais similares, o algoritmo percorrerá os demais grupos, a fim de encontrar outras possíveis instâncias mais similares à instância *x*.

Neste exemplo, obtém-se através da instrução escrita na linha 7, o conjunto das instâncias mais próximas sendo: $\{y_s^{NN}\}_{s=1}^{n} = \{[(3.0, 1.0), (1.0, 2.0), (5.0, 1.0)]\}$ e $d_k = d(x, \mathrm{I1}) = 2.23$.

Deve-se agora, percorrer as demais instâncias dos demais grupos, neste exemplo, o próximo grupo é o *G2* (linha 8).

Através da desigualdade triangular (Expressão 3.1) de *x* para o centro de *G2*, é verificado se, $d(x, c_2) - r_2$ é menor que $d_k$,

$$3.16 - 7.07 < 2.23$$

logo,

$$-3.91 < 2.23$$

portanto, a expressão é verdadeira. Desta forma, é necessário percorrer cada uma das instâncias do grupo *G2*, a fim de verificar a Expressão (3.2) através da linha 17, ou seja,

$$|d(x, c_2) - d(\mathrm{I7}, c_2)| > 2.23,$$

então,

$$2.16 > 2.23,$$

obtém-se, resultado falso para a Expressão (3.2), isto significa que, será necessário computar cálculo de distância, a fim de, verificar se a instância I7 é uma instância candidata a uma das *k* mais similares a *x*. Com isto, é calculado a distância de *x* para a instância I7 (linha 22), obtendo a distância $d(x, \mathrm{I7}) = 3.6$.

É verificado então, se a instância I7 deve ser inserida no conjunto $\{y_s^{NN}\}_{s=1}^{n}$ (linha 23), onde,

$$3.6 < 2.23,$$

a partir deste resultado, despreza-se a instância I7.

A próxima instância a ser verificada é a I8, através da Expressão (3.2) (linha 17), considerando

$$|d(x, c_2) - d(\mathrm{I8}, c_2)| > 2.23,$$

então,

$$2.16 > 2.23,$$

para a instância I8, obtém-se resultado falso, isto conduz a verificar se esta instância pode ser candidata as instâncias mais similares, através da linha 22, computando a distância $d(x, \mathrm{I8}) = 4.12$.

Com isso, pode-se concluir que a instância I8, também, deve ser desprezado uma vez que $4.12 < 2.23$ (linha 23).

A última instância a ser verificada para o grupo *G2* é a instância I9, através da Expressão (3.2), obtemos,

$$|d(x, c_2) - d(\mathrm{I9}, c_2)| > 2.23,$$

então,

$$1.75 > 2.23,$$

resultado falso para a instância I9, conduzindo a computar cálculo de distância e verificação através das linhas 22 e 23, obtendo, $d(x, \mathrm{I9}) = 2.0$, o que conduz a verificação,

$$2.0 < 2.23,$$

que produz um resultado verdadeiro, isto é, I9 é uma instância que deverá ser inserida no conjunto $\{y_s^{NN}\}_{s=1}^{n}$.

O algoritmo agora irá ordenar de forma ascendente as distâncias de *x* para as instâncias do conjunto $\{y_s^{NN}\}_{s=1}^{n}$ (linha 28), obtendo

$$\{y_s^{NN}\}_{s=1}^{n} = \{[(3.0, 1.0), (3.0, 5.0), (1.0, 2.0)]\}.$$

A seguir o algoritmo reduz as instâncias do conjunto de instâncias mais similares, para armazenar apenas as *k* mais similares, logo, neste exemplo, buscamos $k = 2$ (linha 29), logo

$$\{y_s^{NN}\}_{s=1}^{n} = \{[(3.0, 1.0), (3.0, 5.0)]\}.$$

É atualizado $d_k$ para a instância mais distante de *x*, a partir do conjunto das instancias mais similares, logo neste exemplo, $d_k = 2.0$ (linha 30).

O algoritmo retorna para a linha 10, onde, se verificará a necessidade de percorrer as instancias do grupo *G3*, caso, $d(x, c_3) - r_3$ for menor que $d_k$, uma vez que,

$$6.7 - 8.0 < 2.0,$$

logo,

$$-1.3 < 2.0.$$

Obtém-se resultado verdadeiro para a expressão, com isto, é necessário percorrer cada uma das instâncias do grupo *G3*, para este exemplo, a verificação pela Expressão (3.2) para as instâncias I4, I5 e I6 produzirão os mesmos resultados, ou seja,

$$\left|d\left(x, c_j^*\right) - d(y_{j,i}, c_j^*)\right| > d_k,$$

então,

$$5.7 > 2.0.$$

Sendo o resultado verdadeiro para a Expressão (3.2), aqui ocorre a segunda fonte de economia de cálculos de distância, apresentada no algoritmo, uma vez que, não é necessário computar cálculos de distância para as instâncias do grupo *G3*.

Por fim, são executadas novamente as linhas 28, 29 e 30.

A saída deste algoritmo será o conjunto das *k* instâncias mais similares a *x*,

$$\{y_s^{NN}\}_{s=1}^{n} = \{[(3.0, 1.0), (3.0, 5.0)]\}.$$

### 3.4 – Algoritmo KMC-OTI-FS

O algoritmo KMC-OTI-FS (Pan *et al.,* 2020), foi desenvolvido a partir de adaptações do algoritmo *k*M*k*NN, propondo melhorias no processo de busca por meio de um processo de otimização baseado no uso da desigualdade triangular.

A melhoria no processo de busca proposto tem como objetivo otimizar a estratégia de desigualdade triangular apresentada no *k*M*k*NN, de modo que o *k*M*k*NN não possuí eficiência em desprezar as instâncias mais distantes do centro de um grupo $C_j^*$. Com isso, o KMC-OTI-FS, propõe uma estratégia que consiste em buscar uma desigualdade triangular que seja mais eficiente para a estratégia de busca.

Segundo Pan e seus colegas se $\left|d\left(x, c_j^*\right) - d(y_{j,i}, c_j^*)\right|$ obtiver valor suficientemente pequeno, a expressão $\left|d\left(x, c_j^*\right) - d(y_{j,i}, c_j^*)\right| < d_k$ terá grande possibilidade de ser verdadeira. Desta forma, a estratégia de desigualdade triangular tem grande possibilidade

de desprezar as instâncias $y_{j,i}$, presentes nas áreas mais distantes do agrupamento, ou, áreas marginais, mesmo que não sejam as *k* instâncias mais próximas de *x*.

De acordo com a desigualdade apresentada acima, pode-se entender que, se o termo $\left|d\left(x, c_j^*\right) - d(y_{j,i}, c_j^*)\right|$ obtiver um valor suficientemente grande, a expressão $\left|d\left(x, c_j^*\right) - d(y_{j,i}, c_j^*)\right| > d_k$ terá grande possibilidade de ser verdadeira. Consequentemente, de acordo com Pan e seus colegas, se faz necessário selecionar um novo centro do grupo, a fim de, através da desigualdade triangular tornar o resultado do termo $\left|d\left(x, c_j^*\right) - d(y_{j,i}, c_j^*)\right|$ grande o suficiente. Para isto, o novo centro do grupo selecionado, $c_{opt}$, deve satisfazer duas condições:

1. $d(y_{j,i}, c_{opt})$, a distância da instância $y_{j,i}$ do grupo até o centro, $c_{opt}$, deve ser conhecida antes da fase de busca.
2. O centro do grupo selecionado, $c_{opt}$, deve garantir que a distância, $\left|d\left(x, c_{opt}\right) - d(y_{j,i}, c_{opt})\right|$ é a maior entre os *M* grupos de $\{C_j\}_{j=1}^{M}$.

A proposta de otimização na busca das *k* instâncias mais próximas através do KMC-OTI-FS, se dá através da escolha de um novo centro do grupo, esta estratégia de desigualdade triangular é definida pela Expressão (3.3).

$$|d(x, c_{opt}) - d\left(y_{j,i}, c_{opt}\right)| = arg\ max_{s=1,2,..M} |d(x, c_s^*) - d\left(y_{j,i}, c_s^*\right)| \qquad (3.3)$$

A execução da expressão acima é conduzida durante a fase de busca do algoritmo KMC-OTI-FS, sem computar nenhum cálculo de distância, uma vez que ambas as distâncias $d(x, c_s^*)$ e $d\left(y_{j,i}, c_s^*\right)$ são conhecidas, devido $d(x, c_s^*)$ ser obtida no primeiro passo de execução do algoritmo que será apresentado adiante, e $d\left(y_{j,i}, c_s^*\right)$ é calculada e armazenada na fase de agrupamento (offline).

Por fim, se a Expressão (3.4),

$$|d(x, c_{opt}) - d\left(y_{j,i}, c_{opt}\right)| > d_k \qquad (3.4)$$

for verdadeira, pode-se deduzir que $d(x, y_{j,i}) > d_k$ é verdadeira de acordo com a Expressão (3.2), ou seja, podendo-se desprezas as instâncias $y_{j,i^*}$.

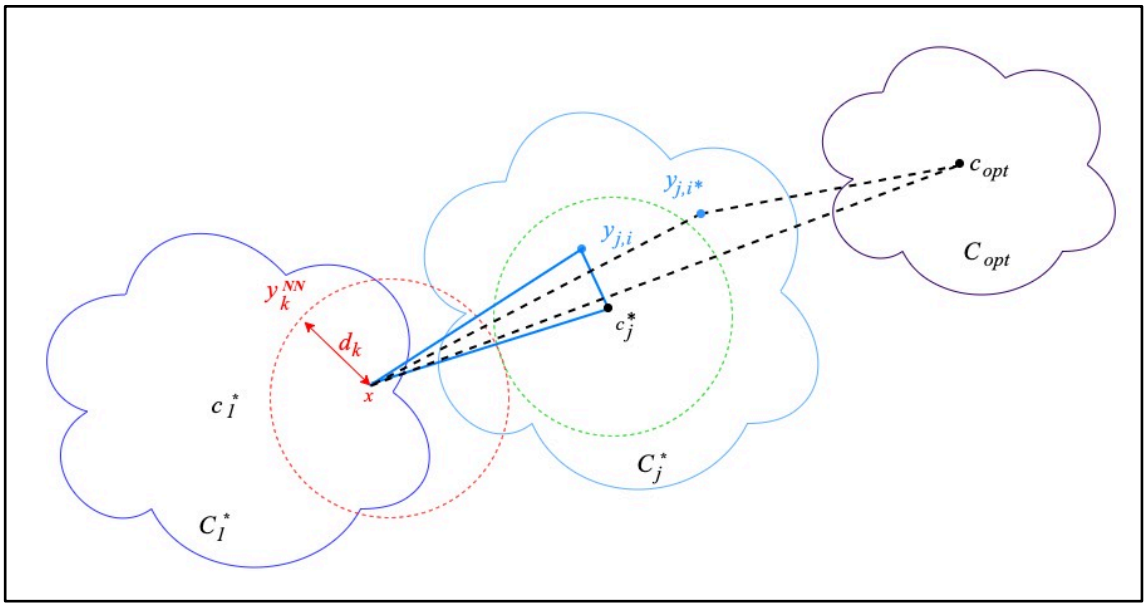



**Figura 3.6:** A figura apresenta a estratégia utilizada do algoritmo KMC-OTI-FS, a fim de, desprezar as instâncias mais distantes do centro de um grupo $C_j^*$. (adaptado de Pan *et al.*, 2020).

**Algoritmo** KMC-OTI-FS

**Fase 1: fase de agrupamento (*offline*)**

**Entrada**: $\boldsymbol{T}$: conjunto de instancias; $\boldsymbol{N}$: número de instâncias em $T$; $\boldsymbol{M}$: número de grupos a serem formados em $T$, onde $M = \sqrt{N}$.

**Saída**: $\{\boldsymbol{C_j}\}_{\boldsymbol{j=1}}^{\boldsymbol{M}}$.: $M$ grupos de instâncias em $T$; $\{\boldsymbol{c_j}\}_{\boldsymbol{j=1}}^{\boldsymbol{M}}$: $M$ respectivos centros dos grupos de instâncias em $T$; $\{\boldsymbol{r_j}\}_{\boldsymbol{j=1}}^{\boldsymbol{M}}$: $M$ respectivos raios dos grupos de instâncias em $T$, onde o raio $r_j$ de um grupo $C_j$ é definido pela maior distância entre o centro do grupo $C_j$ e uma instância dentro do grupo $C_j$; $\{\{\{\boldsymbol{d(y_{j,i}, c_p)}\}_{\boldsymbol{p=1}}^{\boldsymbol{M}}\}_{\boldsymbol{i=1}}^{\boldsymbol{N_j}}\}_{\boldsymbol{j=1}}^{\boldsymbol{M}}$: distância de cada instância $y$ ao centro ótimo do grupo que a contém, estes centros são definidos por definidos por $\boldsymbol{c_p}$. Aqui $N_j$ representa o número de instâncias do grupo $C_j$.

**Procedimento**: algoritmo de agrupamento $k$-Means (MacQueen, 1967).

**Fase 2: fase de busca propriamente (*online*)**

**Entrada**: $\boldsymbol{x}$: instância à qual se busca, em $T$, instâncias mais próximas a ela; $\boldsymbol{k}$: número de instâncias similares que se quer buscar.

**Usa**: saídas da Fase 1 (*offline*).

**Saída**: $\{y_s^{NN}\}_{s=1}^{k}$: conjunto das $k$ instâncias de $T$ mais similares a $x$.

1 {

2 Computar as distâncias euclidianas $\{d(x, c_j)\}_{j=1}^{M}$.

3 Ordenar $\{d(x, c_j)\}_{j=1}^{M}$ para obter $\{d(x, c_j^*)\}_{j=1}^{M}$, onde $d(x, c_1^*) < d(x, c_2^*) < \cdots < d(x, c_M^*)$.

4 Para o grupo mais próximo a $x$, $C_1^*$, computar as distâncias euclidianas

5 $\{d(x, y_{1,i})\}_{i=1}^{N_1^*}$, isto é, as distâncias de $x$ a cada instância $y$ do Grupo $C_1^*$.

```
6     Ordenar {d(x, y_{1,i})}_{i=1}^{N_1^*} para localizar as k instâncias correntes mais similares a x;
7     Inserir essas instâncias em {y_s^{NN}}_{s=1}^{n}, n = k, e fazer d_k = y_k^{NN}.
8     para j := 2 até M faça
9     {
10        se d(x, c_j^*) − r_j^* < d_k então // Primeira fonte de economia de cálculos
11          // de distância (Expressão 3.1). Aqui r_j^* é o raio do grupo C_j^*. Se
12          // este teste é falso, então não é necessário cálculos de
13          // distâncias entre x e as instâncias do Grupo C_j^*.
14        {
15           para i := 1 até N_j^* faça
16           {
17              |d(x, c_opt) − d(y_{j,i}, c_opt)| = arg max_{s=1,2,..M} |d(x, c_s^*) − d(y_{j,i}, c_s^*)|
18              se |d(x, c_opt) − d(y_{j,i}, c_opt)| > d_k então
19                // Segunda fonte de economia de cálculos de distâncias
20                //(Expressão 3.4). Não é necessário o cálculo de d(x, y_{j,i}).
21              senão
22              {
23                 Calcular a distância euclidiana d(x, y_{j,i}).
24                 se d(x, y_{j,i}) < d_k então
25                    Fazer n = n + 1 e inserir y_{j,i} em {y_s^{NN}}_{s=1}^{n}.
26              }
27           }
28        }
29        Ordenar {d(x, y_s^{NN})}_{s=1}^{n}.
30        Reduzir {y_s^{NN}}_{s=1}^{n} para n = k instâncias mais próximas (mais similares).
31        Fazer d_k = (x, y_k^{NN}).
32    }
33    Retornar como saída {y_s^{NN}}_{s=1}^{k}.
34 }
```

**Figura 3.7:** Pseudocódigo algoritmo KMC-OTI-FS.

**Exemplo 3.2.** Para este exemplo será considerado o mesmo conjunto de instâncias (*T*) apresentado na Seção 3.2 Figura 3.4.

Será utilizado para este exemplo, também, a mesma instância *x* de entrada em relação a seção anterior, ou seja,

$$x = (3.0,\ 3.0).$$

Na fase de agrupamento do KMC-OTI-FS, considera-se que o *k*-Means induziu o mesmo agrupamento citados na Seção 3.2 Exemplo 3.1, e a quantidade de grupos sendo $M = 3$.

Neste exemplo será realizada a busca das $k = 2$ instâncias mais similares à instância de entrada.

A fase de busca do algoritmo, inicia-se computando as distâncias de *x* para o centro de cada um dos grupos (linha 2) e deve-se ordenar as distâncias computadas de forma ascendente (linha 3), os resultados são apresentados na Tabela 3.2.

**Tabela 3.2:** Distâncias computadas e ordenadas de *x* a cada centro de grupo.

| **Distâncias para o centro do grupo** | ***G1*** | ***G2*** | ***G3*** |
|---|---|---|---|
| ***x*** | 1.0 | 3.16 | 6.7 |

Assim como é realizado pelo *k*M*k*NN, o KMC-OTI-FS verifica o grupo mais próximo, a fim de, encontrar as instâncias mais similares à *x,* computando as distâncias de *x* para cada instância do grupo *G1* (linhas 4 e 5) e as ordena de forma ascendente (linha 6), obtendo as distâncias,

$$\{d(x, y_{1,i})\}_{i=1}^{N_1^*} = \{[2.0,\ 2.23,\ 2.82]\},$$

respectivamente as instâncias, I2, I1 e I3.

Deve-se inserir as instâncias no conjunto de instâncias mais similares, obtendo,

$$\{y_s^{NN}\}_{s=1}^{n} = \{[(3.0, 1.0), (1.0, 2.0), (5.0, 1.0)]\},$$

e atualizar $d_k = 2.23$ (linha 7).

O algoritmo deverá percorrer os demais grupos, a fim de, encontrar outras instâncias similares, para isto, neste exemplo o próximo grupo a ser percorrido é o *G2* (linha 10).

Desta forma, através da Expressão (3.1) é verificado se, $d(x, c_2) - r_2$, é menor em relação à $d_k$, ou seja,

$$3.16 - 7.07 < 2.23$$

logo,

$$-3.91 < 2.23.$$

Obtendo resultado verdadeiro pela Expressão (3.1), deve-se percorrer cada uma das instâncias do grupo *G2*, sendo I7 a primeira instância.

Aqui ocorre a proposta empregada por Pan *et al.* (2020), objetivando acelerar a fase de busca do *k*-NN, onde, é utilizado a Expressão (3.3) com objetivo de desprezar as instâncias mais distantes do centro do grupo (linha 17), com isso,

$$|d(x, c_{opt}) - d(y_{j,i}, c_{opt})| = 2.16.$$

Com base no resultado obtido, o algoritmo verificará se o novo centro do grupo encontrado é maior em relação à $d_k$, pela Expressão (3.4) (linha 18),

$$2.16 > 2.23,$$

Neste exemplo, a partir do resultado obtido, deverá ser computado o cálculo de distância para a instância I7, em caso de resultado positivo, aqui ocorre uma fonte de economia proposta por Pan *et al.* (2020).

É computado e verificado então, se a instância I7 deve ser inserida no conjunto $\{y_s^{NN}\}_{s=1}^{n}$ (linha 23 e 24), onde,

$$3.6 < 2.23,$$

a partir deste resultado, despreza-se a instância I7.

A próxima instância do grupo *G2* é a I8, pela Expressão (3.3),

$$|d(x, c_{opt}) - d(y_{j,i}, c_{opt})| = 2.16.$$

Para a instância I8, compreende-se que possuindo resultado semelhante, em relação a instância I7, deve ser desprezada, uma vez que, $d(x, \text{I8}) = 4.12$ e $4.12 < 2.23$ (linhas 23 e 24).

Por fim, a última instância de *G2* é a I9, temos pela Expressão (3.3),

$$|d(x, c_{opt}) - d(y_{j,i}, c_{opt})| = 1.75,$$

que, para este exemplo também será computado cálculo de distância e verificado se esta instância é candidata ao conjunto $\{y_s^{NN}\}_{s=1}^{n}$. Pelas linhas 23 e 24 obtemos, $d(x, \text{I9}) = 2.0 \; e \; 2.0 < 2.23$.

Portanto, a instância I9 deve ser inserida no conjunto $\{y_s^{NN}\}_{s=1}^{n}$.

O algoritmo deverá ordenar, reduzir e atualizar o valor de $d_k$, através das linhas 29, 30 e 31. Obtendo, $\{y_s^{NN}\}_{s=1}^{n} = \{[(3.0, 1.0), (3.0, 5.0)]\}$ e $d_k = 2.0$.

O último grupo a ser verificado é o *G3*, através da Expressão (3.1) (linha 10), sendo,

$$6.7 - 8.0 < 2.0$$

logo,

$$-1.3 < 2.0.$$

Obtém-se resultado verdadeiro para a expressão, com isto, é necessário percorrer cada uma das instâncias do grupo *G3*, para este exemplo, assim como no exemplo anterior, os resultados para a Expressão (3.3) e a verificação pela Expressão (3.4) para as instâncias I4, I5 e I6 produzirão os mesmos resultados, ou seja,

$$|d(x, c_{opt}) - d(y_{j,i}, c_{opt})| = 5.7,$$

e,

$$5.7 > 2.0.$$

Resultado verdadeiro para a Expressão (3.4), aqui ocorre a segunda fonte de economia de cálculos de distância, uma vez que, não é necessário computar cálculos de distância para as instâncias do grupo *G3*.

Portanto, o algoritmo deverá ordenar, reduzir e atualizar o valor de $d_k$, através das linhas 29, 30 e 31. Obtendo, $\{y_s^{NN}\}_{s=1}^{n} = \{[(3.0, 1.0), (3.0, 5.0)]\}$ e $d_k = 2.23$.

Ao seu termino, o algoritmo retornará o conjunto das instâncias mais similares à *x,*

$$\{y_s^{NN}\}_{s=1}^{n} = \{[(3.0, 1.0), (3.0, 5.0)]\}.$$

### 3.4 – Algoritmo KMC-EOTI-FS

O KMC-EOTI-FS, visa corrigir uma deficiência na complexidade computacional de espaço do algoritmo KMC-OTI-FS.

Segundo Pan e seus colegas, a proposta do KMC-EOTI-FS é manter a preocupação entre desempenho de busca e a complexidade computacional de tempo e espaço, com base nesta proposta os autores propuseram um modelo no qual busca maior eficiência na desigualdade triangular, com o propósito de constituir um centro secundário $c_{eopt}$, que possa minimizar a complexidade computacional de tempo e espaço do algoritmo.

A estratégia utilizada neste algoritmo consiste que para cada instância $y_{j,i}$, e são armazenados somente duas distâncias, $d(y_{j,i}, c_j)$ e $d(y_{j,i}, c_{max})$. Entre todos os grupos $\{c_j\}_{j=1}^{M}$, $c_j$ é o centro do grupo mais próximo da instância $y_{j,i}$, e $c_{max}$ é o centro de grupo mais distante da instância $y_{j,i}$. Com base nisso, a distância $d(y_{j,i}, c_{max})$ é definida pela Expressão (3.5).

$$d(y_{j,i}, c_{max}) = arg\ max_{s=1,2,..M} |d(y_{j,i}, c_s^*)| \quad (3.5)$$

Segundo Pan e seus colegas, a razão de usar $d(x, c_{max})$ é que $|d(x, c_{max}) - d(y_{j,i}, c_{max})|$ pode ser máximo quando $d(x, c_{max})$ for máximo. Logo, $c_{eopt}$ é definido pela Expressão (3.6).

$$|d(x, c_{eopt}) - d(y_{j,i}, c_{eopt})| = max\begin{Bmatrix} |d(x, c_j) - d(y_{j,i}, c_j)|, \\ |d(x, c_{max}) - d(y_{j,i}, c_{max})| \end{Bmatrix} \quad (3.6)$$

A expressão (3.7) pode ser compreendida a partir das expressões (3.5) e (3.6). A expressão a seguir é demonstrado que $|d(x, c_{eopt}) - d(y_{j,i}, c_{eopt})|$ será sempre maior ou igual em relação à $d(x, c_j) - d(y_{j,i}, c_j)|$. Pan e seus colegas afirmam que em uma comparação com o *k*M*k*NN, o KMC-EOTI-FS pode não só desprezar as instâncias menos relevantes, próximas do centro do grupo mais próximos, quanto desprezar as instâncias menos relevantes mais distantes do centro dos grupos mais próximos.

$$\left|d(x, c_{opt}) - d(y_{j,i}, c_{opt})\right| \geq \left|d(x, c_{eopt}) - d(y_{j,i}, c_{eopt})\right| \geq \left|d(x, c_j) - d(y_{j,i}, c_j)\right| \quad (3.7)$$

Por fim, com base no KMC-EOTI-FS conclui-se que se a Expressão (3.8) for verdadeira, pode-se desprezar as instâncias $y_{j,i}$.

$$\left|d(x, c_{eopt}) - d(y_{j,i}, c_{eopt})\right| > d_k \quad (3.8)$$

**Algoritmo** KMC-EOTI-FS

**Fase 1: fase de agrupamento (*offline)***

**Entrada**: ***T***: conjunto de instancias; ***N***: número de instâncias em *T*; ***M***: número de grupos a serem formados em *T*, onde $M = \sqrt{N}$.

**Saída**: $\{\boldsymbol{C_j}\}_{\boldsymbol{j=1}}^{\boldsymbol{M}}$.: *M* grupos de instâncias em *T*; $\{\boldsymbol{c_j}\}_{\boldsymbol{j=1}}^{\boldsymbol{M}}$: *M* respectivos centros dos grupos de instâncias em *T*; $\{\boldsymbol{r_j}\}_{\boldsymbol{j=1}}^{\boldsymbol{M}}$: *M* respectivos raios dos grupos de instâncias em *T*, onde o raio $r_j$ de um grupo $C_j$ é definido pela maior distância entre o centro do grupo $C_j$ e uma instância dentro do grupo $C_j$; $\{\{\boldsymbol{d(y_{j,i}, c_j), d(y_{j,i}, c_{max})}\}_{\boldsymbol{i=1}}^{\boldsymbol{N_j}}\}_{\boldsymbol{j=1}}^{\boldsymbol{M}}$:distância de cada instância *y* ao centro do grupo que a contém, estes centros são definidos por definidos por $\boldsymbol{c_j}$ e a distância de cada instância *y* para o agrupamento mais distante. Aqui $N_j$ representa o número de instâncias do grupo $C_j$.

**Procedimento**: algoritmo de agrupamento *k*-Means (MacQueen, 1967).

**Fase 2: fase de busca propriamente (*online*)**

**Entrada**: ***x:*** instância à qual se busca, em *T*, instâncias mais próximas a ela; ***k***: número de instâncias similares que se quer buscar.

**Usa**: saídas da Fase 1 (*offline*).

**Saída**: $\{y_s^{NN}\}_{s=1}^{k}$: conjunto das *k* instâncias de *T* mais similares a *x*.

```
1  {
2     Computar as distâncias euclidianas {d(x, c_j)}_{j=1}^{M}.
3     Ordenar {d(x, c_j)}_{j=1}^{M} para obter {d(x, c_j^*)}_{j=1}^{M}, onde d(x, c_1^*) < d(x, c_2^*) < ··· < d(x, c_M^*).
4     Para o grupo mais próximo a x, C_1^*, computar as distâncias euclidianas
5     {d(x, y_{1,i})}_{i=1}^{N_1^*}, isto é, as distâncias de x a cada instância y do Grupo C_1^*.
6     Ordenar {d(x, y_{1,i})}_{i=1}^{N_1^*} para localizar as k instâncias correntes mais similares a x;
7     Inserir essas instâncias em {y_s^{NN}}_{s=1}^{n}, n = k, e fazer d_k = y_k^{NN}.
8     para j := 2 até M faça
9     {
10       se d(x, c_j^*) − r_j^* < d_k então // Primeira fonte de economia de cálculos
11          // de distância (Expressão 3.1). Aqui r_j^* é o raio do grupo C_j^*. Se
12          // este teste é falso, então não é necessário cálculos de
13          // distâncias entre x e as instâncias do Grupo C_j^*.
14       {
15          para i := 1 até N_j^* faça
```

16 {

17 $|d(x, c_{eopt}) - d(y_{j,i}, c_{eopt})| = max\begin{Bmatrix} |d(x, c_j) - d(y_{j,i}, c_j)|, \\ |d(x, c_{max}) - d(y_{j,i}, c_{max})| \end{Bmatrix}$

18 **se** $|d(x, c_{eopt}) - d(y_{j,i}, c_{eopt})| > d_k$ **então**

19 // Segunda fonte de economia de cálculos de distâncias

20 //(Expressão 3.8). Não é necessário o cálculo de $d(x, y_{j,i})$.

21 **senão**

22 {

23 Calcular a distância euclidiana $d(x, y_{j,i})$.

24 **se** $d(x, y_{j,i}) < d_k$ **então**

25 Fazer $n = n + 1$ e inserir $y_{j,i}$ em $\{y_s^{NN}\}_{s=1}^{n}$.

26 }

27 }

28 }

29 Ordenar $\{d(x, y_s^{NN})\}_{s=1}^{n}$.

30 Reduzir $\{y_s^{NN}\}_{s=1}^{n}$ para $n = k$ instâncias mais próximas (mais similares).

31 Fazer $d_k = (x, y_k^{NN})$.

32 }

33 Retornar como saída $\{y_s^{NN}\}_{s=1}^{k}$.

34 }

**Figura 3.8:** Pseudocódigo do algoritmo KMC-EOTI-FS.

**Exemplo 3.3.** Mantendo a conformidade dos exemplos anteriores, será utilizado o mesmo conjunto de instâncias apresentado na Seção 3.2 Figura 3.4.

Considera-se a instância de entrada para busca, sendo,

$$x = (3.0, 3.0).$$

Na fase de agrupamento do KMC-EOTI-FS, faz o uso do algoritmo *k*-Means, o qual induziu um agrupamento com os mesmos $M = 3$ grupos mostrados na Seção 3.2 Exemplo 3.1.

Neste exemplo será realizada a busca das $k = 2$ instâncias mais similares à instância de entrada *x*.

A fase de busca do algoritmo, inicia-se computando as distâncias de *x* para o centro de cada um dos grupos (linha 2) e deve-se ordenar as distâncias computadas de forma ascendente (linha 3), os resultados para esta execução são os mesmos em relação aos apresentados na Seção 3.2 Tabela 3.1.

A linhas iniciais do KMC-EOTI-FS são semelhantes aos demais algoritmos apresentados, logo, através das linhas 4, 5 e 6, obtém-se as distâncias das instâncias que tendem a as mais similares à instância *x,* portanto, deve-se inserir as instâncias no conjunto de instâncias mais similares (linha 7),

$$\{y_s^{NN}\}_{s=1}^{n} = \{[(3.0, 1.0), (1.0, 2.0), (5.0, 1.0)]\},$$

e atualizar $d_k = 2.23$.

O próximo grupo que deve ser percorrido é o grupo *G2*, a fim de, verificar se existem outras instâncias mais similares à instância *x* (linha 10).

Desta forma, através da Expressão (3.1) é verificado, $d(x, c_2) - r_2$, é menor em relação à $d_k$, ou seja,

$$3.16 - 7.07 < 2.23,$$

logo,

$$-3.91 < 2.23.$$

Obtendo resultado verdadeiro pela Expressão (3.1), deve-se percorrer cada uma das instâncias do grupo *G2*, sendo I7 a primeira instância.

Aqui ocorre a proposta empregada para o KMC-EOTI-FS, por Pan *et al.* (2020), objetivando acelerar a fase de busca do *k*-NN, onde, é utilizado a Expressão (3.6) com objetivo de desprezar as instâncias mais distantes do centro do grupo (linha 17), com isso, primeiramente sabe-se que para a instância I7, $d(y_{j,i}, c_{max}) = arg\ max_{s=1,2,..M} |d(y_{j,i}, c_s^*)|$ é definido pelas Expressões (3.5) e (3.6), sendo,

$$d(y_{2,1}, c_{max}) = 8.0,$$

logo,

$$|d(x, c_{eopt}) - d(y_{j,i}, c_{eopt})| = max \begin{Bmatrix} |3.16 - 1.0|, \\ |6.7 - 8.0| \end{Bmatrix},$$

portanto,

$$\left|d\left(x, c_{eopt}\right) - d\left(y_{j,i}, c_{eopt}\right)\right| = 2.16.$$

Com isso, pela Expressão (3.8) $\left|d\left(x, c_{eopt}\right) - d\left(y_{j,i}, c_{eopt}\right)\right|$ é menor em relação a $d_k$, deve ser computado cálculo de distância de *x* para I7, e, verificar se I7 pode ser inserido no conjunto $\{y_s^{NN}\}_{s=1}^{n}$ (linhas 23 e 24), computando $d(x, \text{I7})$ = 3.6, logo, $3.16 > 2.23$. Despreza-se a instância I7.

A próxima instância a se verificar é a instância I8, sendo que pelas Expressões (3.5) e (3.6), obtém-se,

$$d\left(y_{2,2}, c_{max}\right) = 7.07,$$

logo,

$$\left|d\left(x, c_{eopt}\right) - d\left(y_{j,i}, c_{eopt}\right)\right| = max\left\{\begin{matrix}|3.16 - 1.0|,\\ |6.7 - 7.07|\end{matrix}\right\},$$

portanto,

$$\left|d\left(x, c_{eopt}\right) - d\left(y_{j,i}, c_{eopt}\right)\right| = 2.16.$$

Pela Expressão (3.8), $2.16 < 2.23$, ou seja, assim como a instância I7, deverá ser computado cálculo de distância de *x* para I8, e, verificar se a instância I8 deve ou não ser desprezada.

Computando $d(x, \text{I8})$ = 4.12, logo, $4.12 > 2.23$, deve-se desprezar a instância I8.

Para a última instância do grupo *G2*, pelas Expressões (3.5) e (3.6), temos,

$$d\left(y_{2,3}, c_{max}\right) = 6.08,$$

logo,

$$\left|d\left(x, c_{eopt}\right) - d\left(y_{j,i}, c_{eopt}\right)\right| = max\left\{\begin{matrix}|3.16 - 1.41|,\\ |6.7 - 6.08|\end{matrix}\right\},$$

portanto,

$$\left|d\left(x, c_{eopt}\right) - d\left(y_{j,i}, c_{eopt}\right)\right| = 1.41.$$

Pela Expressão (3.8), $1.41 < 2.23$, com isso, deve-se computar cálculo de distância para I9 (linha 23) e verificar sua inclusão no conjunto $\{y_s^{NN}\}_{s=1}^{n}$.

Para I9, computando $d(x, \text{I9})$ = 2.0, logo, $2.0 < 2.23$, portanto, deve-se incluir I9 no conjunto $\{y_s^{NN}\}_{s=1}^{n}$, logo, $\{y_s^{NN}\}_{s=1}^{n} = \{[(3.0, 1.0), (1.0, 2.0), (5.0, 1.0), (3.0, 5.0)]\}$.

O algoritmo ordenará e reduzirá o conjunto das instâncias mais próximas para obter as *k* instâncias mais similares à instância *x* (linhas 29 e 30), obtendo,

$$\{y_s^{NN}\}_{s=1}^{n} = \{[(3.0, 1.0), (3.0, 5.0)]\},$$

e redefinir o valor para $d_k = 2.0$ (linha 31).

O último grupo a se verificar pela Expressão (3.1) é o *G3* (linha 10), considerando,

$$6.7 - 8.0 < 2.0,$$

então,

$$-1.3 < 2.0.$$

A expressão sendo verdadeira, deve-se percorrer cada uma das instâncias de *G3*, sendo I4 a primeira instância.

Pelas Expressões (3.5) e (3.6),

$$d(y_{3,1}, c_{max}) = 7.07,$$

logo,

$$|d(x, c_{eopt}) - d(y_{j,i}, c_{eopt})| = max\begin{Bmatrix} |6.7 - 1.0|, \\ |6.7 - 7.07| \end{Bmatrix},$$

portanto,

$$\left|d(x, c_{eopt}) - d(y_{j,i}, c_{eopt})\right| = 5.7.$$

Com o resultado da Expressão (3.6), verifica-se pela Expressão (3.8),

$$5.7 > 2.0,$$

logo, obtendo resultado verdadeiro, deve-se computar cálculo de distância para a instância I4, e, verificando a distância de *x* para a instância I4, onde, $6.32 > 2.0$ (linhas 23 e 24). Despreza-se a instância I4.

Para a instância I5, pelas Expressões (3.5) e (3.6), temos,

$$d(y_{3,2}, c_{max}) = 6.4,$$

logo,

$$|d(x, c_{eopt}) - d(y_{j,i}, c_{eopt})| = max\begin{Bmatrix} |6.7 - 1.0|, \\ |6.7 - 6.4| \end{Bmatrix},$$

portanto,

$$\left|d\left(x, c_{eopt}\right) - d\left(y_{j,i}, c_{eopt}\right)\right| = 5.7.$$

Com o resultado da Expressão (3.6), verifica-se pela Expressão (3.8), o resultado produzido é igual em relação a instância I4, logo, considerando o cálculo de distância de *x* para I5, onde, $5.85 > 2.0$ (linhas 23 e 24). Despreza-se a instância I5.

A última instância deste grupo, I6, uma vez que pelas Expressões (3.5) e (3.6), produzirá o mesmo resultado das instâncias I4 e I5, computa-se cálculo de distância de *x* para I6, onde, $7.21 > 2.0$ (linhas 23 e 24). Despreza-se, também, a instância I6.

Por fim, o algoritmo deverá ordenar, reduzir e atualizar o valor de $d_k$, através das linhas 29, 30 e 31. Obtendo, $\{y_s^{NN}\}_{s=1}^{n} = \{[(3.0, 1.0), (3.0, 5.0)]\}$ e $d_k = 2.0$.

Ao seu termino, o algoritmo retornará o conjunto das instâncias mais similares à *x,*

$$\{y_s^{NN}\}_{s=1}^{n} = \{[(3.0, 1.0), (3.0, 5.0)]\}.$$

# Capítulo 4
# Algoritmo *k*M++*k*NN

*Este capítulo descreve o algoritmo proposto, denominado kM++kNN. A Seção 4.1 discute a principal ideia que inspirou o desenvolvimento do algoritmo. A Seção 4.2 apresenta o algoritmo kM++kNN dando foco para as diferenças entre ele e o algoritmo kMkNN.*

## 4.1 – Ideia para um Novo Algoritmo

O *k*M*k*NN (Xueyi, 2012) vem sendo citado como um algoritmo rápido para buscar *k* instancias mais similares a uma instância *x* em um conjunto de instâncias *T* (Pan *et al*., 2020). Em sua fase de pré-processamento o algoritmo induz um agrupamento (conjunto de grupos) no conjunto de instâncias dado como entrada. Na fase de busca o *k*M*k*NN usa o conceito de desigualdade triangular como estratégia de estratégia de exclusão no sentido de limitar o número de grupos ou o número de instâncias dentro de grupos em que as instâncias devem ser buscadas, em vez de considerar todo o espaço compreendido pelo conjunto de instâncias.

O *k*M*k*NN emprega o algoritmo *k*-Means (MacQueen, 1967) para formar o agrupamento da fase de pré-processamento. O agrupamento formado pelo *k*-Means depende de um processo de escolha aleatória dos centroides iniciais dos grupos que formarão o agrupamento final. Isto pode produzir situações nem sempre desejáveis como, por exemplo, escolher para centroides instâncias com valores extremos ou instâncias muito próximas (similares) uma das outras. O agrupamento formado afeta a fase de busca na medida em que grupos muito próximos uns dos outros tendem a não serem excluídos pelas estratégias de economia de cálculos de distância empregadas, quando da busca de uma instância *x* dentro destes grupos.

O algoritmo *k*-Means++ (Arthur & Vassilvitskii, 2007), funciona de forma similar ao *k*-Means, mas utiliza uma estratégia de escolha de centroides iniciais que tende a eleger instâncias mais distantes unas das outras e, por fim, tende a induzir agrupamentos contendo grupos mais distantes uns dos outros. É possível que isto otimize o uso das estratégias de economia de cálculos de distâncias na fase de busca.

A principal ideia do algoritmo proposto é utilizar o algoritmo *k*-Means++ para formar um agrupamento na fase de pré-processamento. O novo algoritmo se diferencia do algoritmo *k*M*k*NN em mais alguns detalhes, conforme está escrito na próxima seção.

## 4.2 – O algoritmo *k*M++*k*NN

A principal diferença entre o algoritmo que está sendo proposto e o algoritmo *k*M*k*NN é o emprego do algoritmo *k*-Means++ na fase de pré-processamento. O nome dado a este algoritmo, *k*M++*k*NN foi composto para lembrar deste importante fato. O nome é uma

junção de "*k*M++", uma contração do termo "*k*-Means++*",* com "*k*-NN", uma referência ao algoritmo *k*-NN.

O pseudocódigo do algoritmo *k*M++*k*NN está apresentado na Figura 4.1. O funcionamento do algoritmo *k*M++*k*NN é muito parecido com o do *k*M*k*NN, de modo que muito do que foi explicado no Capítulo 3 sobre o funcionamento do algoritmo *k*M*k*NN aplica-se também ao algoritmo *k*M++*k*NN. No entanto, o algoritmo *k*M++*k*NN se diferencia do algoritmo *k*M*k*NN nos seguintes aspectos:

- Fase de pré-processamento:
    - Emprega, como já foi mencionado, o algoritmo de agrupamento *k*-Means++ para formar o agrupamento.
- Fase de busca propriamente:
    - Emprega como a expressão $d(x, y_{j,i}) \leq d_k$, para economia de cálculos de distância em vez da expressão $d(x, y_{j,i}) < d_k$ usada no algoritmo *k*M*k*NN (Linha 23, Figura 4.1).
    - Não prescindir, para seu correto funcionamento, que o algoritmo de agrupamento induza $k$ ou mais instâncias para o grupo mais próximo à instância que se quer buscar as $k$ instâncias mais próximas. O algoritmo *k*M*k*NN necessita, para seu correto funcionamento, que o grupo mais próximo contenha, pelo menos, $k$ instâncias (Linhas 4 a 7, Figura 4.1).

**Algoritmo** *k*M++*k*NN

**Fase 1: fase de pré-processamento (*offline*)**

**Entrada**: ***T***: conjunto de instancias; ***N***: número de instâncias em *T*; ***M***: número de grupos a serem formados em *T*, onde $M = \sqrt{N}$.

**Saída**: $\{C_j\}_{j=1}^{M}$.: *M* grupos de instâncias em *T*; $\{c_j\}_{j=1}^{M}$: *M* respectivos centros dos grupos de instâncias em *T*; $\{r_j\}_{j=1}^{M}$: *M* respectivos raios dos grupos de instâncias em *T*, onde o raio $r_j$ de um grupo $C_j$ é definido pela maior distância entre o centro do grupo $C_j$ e uma instância dentro do grupo $C_j$; $\{\{d(y_{j,i}, c_j)\}_{i=1}^{N_j}\}_{j=1}^{M}$:distância de cada instância *y* ao centro do grupo que a contém. Aqui $N_j$ representa o número de instâncias do grupo $C_j$.

**Procedimento**: algoritmo de agrupamento ***k*-Means++** (Arthur & Vassilvitskii, 2007).

**Fase 2: fase de busca propriamente (*online*)**

**Entrada**: ***x***: instância à qual se busca, em *T*, instâncias mais próximas a ela;
***k***: número de instâncias similares que se quer buscar.

**Usa**: saídas da Fase 1 (*offline*).

**Saída**: $\{y_s^{NN}\}_{s=1}^{k}$: conjunto das *k* instâncias de *T* mais similares a *x*.

```
{
    Computar as distâncias euclidianas {d(x,c_j)}_{j=1}^{M}.
    Ordenar {d(x,c_j)}_{j=1}^{M} para obter {d(x,c_j^*)}_{j=1}^{M}, onde d(x,c_1^*) < d(x,c_2^*) < ... < d(x,c_M^*).
    Para o grupo mais próximo a x, C_1^*, computar as distâncias euclidianas
    {d(x,y_{1,i})}_{i=1}^{N_1^*}, isto é, as distâncias de x a cada instância y do Grupo C_1^*.
    Ordenar {d(x,y_{1,i})}_{i=1}^{N_1^*} para achar u ≤ k instâncias correntes mais similares a x;
    Inserir essas instâncias em {y_s^{NN}}_{s=1}^{n}, n = u, e fazer d_k = y_u^{NN}.
    para j := 2 até M faça
    {
        se d(x,c_j^*) − r_j^* < d_k então // Primeira fonte de economia de cálculos
          // de distância (Expressão 3.1). Aqui r_j^* é o raio do grupo C_j^*. Se
          // este teste é falso, então não é necessário cálculos de
          // distâncias entre x e as instâncias do Grupo C_j^*.
        {
            para i := 1 até N_j^* faça
            {
                se |d(x,c_j^*) − d(y_{j,i},c_j^*)| > d_k então
                  // Segunda fonte de economia de cálculos de distâncias
                  //(Expressão 3.2). Não é necessário o cálculo de d(x,y_{j,i}).
                senão
                {
                    Calcular a distância euclidiana d(x,y_{j,i}).
                    se d(x,y_{j,i}) ≤ d_k então
                        Fazer n = n + 1 e inserir y_{j,i} em {y_s^{NN}}_{s=1}^{n}.
                }
            }
```

```
27          }
28          Ordenar {d(x, y_s^NN)}_{s=1}^n.
29          Reduzir {y_s^NN}_{s=1}^n para n = k instâncias mais próximas (mais similares).
30          Fazer d_k = (x, y_k^NN).
31      }
32      Retornar como saída {y_s^NN}_{s=1}^k.
33  }
```

**Figura 4.1:** Pseudocódigo do algoritmo *k*M++*k*NN

# Capítulo 5
# Metodologia da Pesquisa

*Este capítulo descreve a metodologia de pesquisa adotada nesta dissertação. A Seção 5.1 apresenta a metodologia adotada para a condução da pesquisa e desenvolvimento dos experimentos que foram realizados. A Seção 5.2 aborda o ambiente de experimentação, apresentando dados sobre o hardware utilizado, bem como dados sobre o IDE (Integrated Development Environment) de desenvolvimento, linguagem de programação e bibliotecas utilizadas. A Seção 5.3, descreve os conjuntos de instâncias utilizados nos experimentos, apresentando suas características e também os algoritmos utilizados para a experimentação. A Seção 5.4 apresenta a estratégia de validação cruzada utilizada na implementação dos experimentos. A Seção 5.5 faz referência aos índices de desempenho utilizados na análise dos experimentos, a fim de comparar resultados entre os algoritmos. Por fim, a Seção 5.6 apresenta detalhadamente o método utilizado na realização dos experimentos.*

## 5.1 – Metodologia de Condução da Pesquisa e de Execução dos Experimentos

A metodologia utilizada nesta pesquisa envolve as seguintes etapas:

(1) Pesquisa (Levantamento) bibliográfica de algoritmos que objetivam acelerar o processo de busca exata do algoritmo *k*-NN. A pesquisa concentrou-se em algoritmos da classe ENN, subcategoria baseada em grupos e que utilizam o conceito de desigualdade triangular (vide Capítulo 1). A pesquisa resultou nos três algoritmos descritos no Capítulo 3.

(2) Estudo e entendimento dos algoritmos identificados na literatura. O resultado do estudo está relatado no Capítulo 3.

(3) Proposta de um novo algoritmo como contribuição original deste trabalho aos esforços de redução do tempo de busca do algoritmo *k*-NN. O algoritmo proposto foi descrito no Capítulo 4.

(4) Estudo e a proposta de um ambiente para experimentação (hardware, sistema operacional, linguagem, IDE, compilador/interpretador).

(5) Implementação dos algoritmos que são abordados neste estudo.

(6) Seleção dos conjuntos de instâncias a serem utilizados em experimentos com os algoritmos estudados.

(7) Proposta de uma metodologia para experimentação, tendo em vista a comparação entre os algoritmos estudados, no que se refere à redução de cálculos de distância e de tempo de execução.

(8) Discussão dos resultados dos experimentos e a inferência de conclusões.

A Figura 5.1 ilustra as etapas que integram esta metodologia de pesquisa.

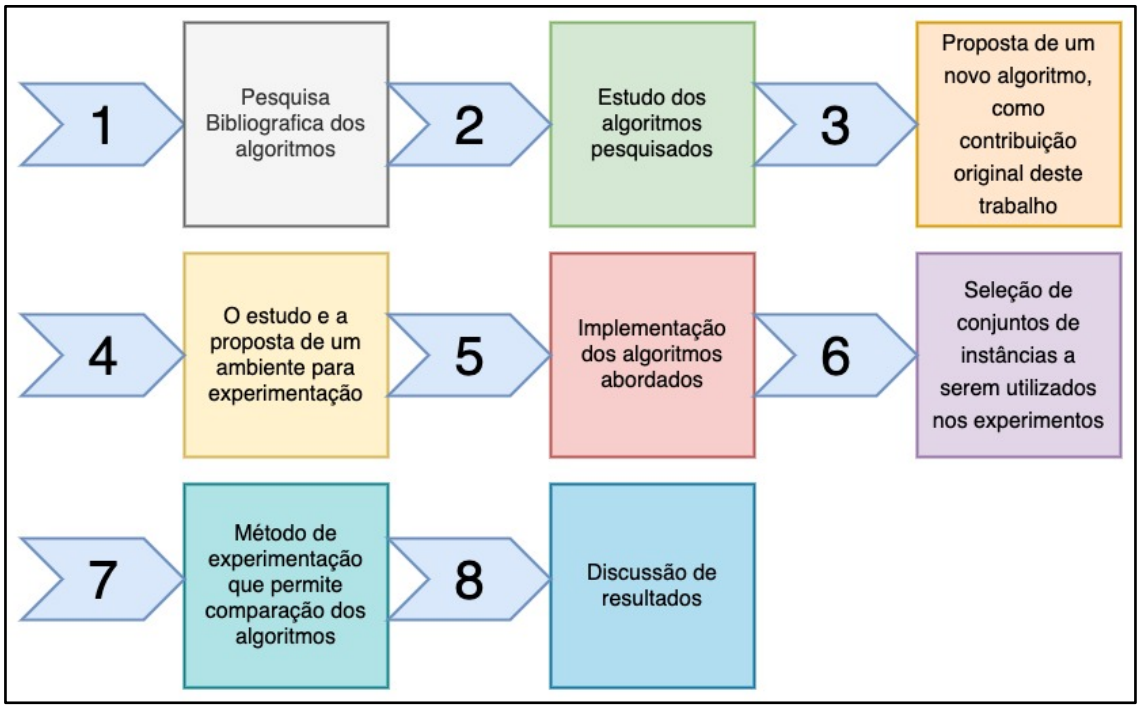



**Figura 5.1:** Etapas da metodologia de pesquisa.

As etapas 1, 2 e 3 foram abordadas anteriormente e estão relatadas nos capítulos 2, 3 e 4 respectivamente. As etapas 4, 6 e 7 serão tratadas neste capítulo. A Seção 5.2 descreve o ambiente de software proposto e a Seção 5.3 os conjuntos de instâncias selecionados para os experimentos desta pesquisa (etapas 4 e 6). A Etapa 7 da metodologia está detalhada na Seção 5.4. A Etapa 8 está descrita no Capítulo 6.

## 5.2 – Ambiente para Experimentação

Para a implementação dos algoritmos foi utilizada a linguagem Python, fazendo uso da IDE de programação PyCharm (JetBrains, 2022). A escolha desta IDE para implementação dos algoritmos, tem por motivo ser uma IDE desenvolvida genuinamente para linguagem Python e por possuir características que auxiliam o processo de depuração do algoritmo, como por exemplo: pontos de parada (*Break Points*) e painéis que permitem análise em tempo de execução. A IDE PyCharm (JetBrains, 2022) também possui fácil integração com o gerenciador de bibliotecas da linguagem Python, o que auxilia no processo de desenvolvimento de modo que a própria IDE faz sugestões de importação de bibliotecas. As bibliotecas utilizadas nas implementações dos algoritmos são *numpy* (NumPy, 2022), *math* (Python, 2022), *scikit-learn* (Scikit Learn, 2022), *csv* (Python, 2022), *pandas* (NumFocus, 2022) e *datetime* (Python, 2022). Os algoritmos são executados na IDE PyCharm (JetBrains, 2022) apenas na fase de implementação e

depuração. Na fase de experimentação os algoritmos são executados via Python Shell (Python, 2022) a fim de garantir que processos que executam na IDE PyCharm (JetBrains, 2021) não interfiram na apuração dos resultados, especialmente na medida dos tempos de processamento. O *hardware* utilizado conta com uma CPU i5-580, operando a 2.5 GHz, memória DDR3 de 8 GB executando o sistema operacional Windows 10 Pro.

## 5.3 – Conjuntos de Instâncias Selecionados para os Experimentos

Nos experimentos foram utilizados 8 conjuntos de instâncias de diferentes tamanhos e dimensões (Tabela 5.1). Tais conjuntos de instâncias foram escolhidos para facilitar a comparação com os experimentos relatados em Pan *et al.* (2020) que fazem uso destes mesmos conjuntos de instâncias.

Todos os conjuntos de instâncias foram obtidos do repositório UCI (Dua & Graff, 2021).

A Tabela 5.1 lista os conjuntos de instâncias utilizados, informando os seus nomes, número de instâncias, número de atributos e uma breve descrição do domínio a que pertencem.

**Tabela 5.1:** Conjuntos de Instâncias selecionados para execução dos experimentos, extraídos do repositório da UCI (Dua & Graff, 2021).

| Nome | Número de Instâncias | Número de Atributos | Domínio |
|---|---|---|---|
| Arcene | 900 | 10000 | Tipos de Câncer |
| Multiplefeatures | 2000 | 649 | Números Manuscritos |
| GasSensors | 13910 | 128 | Sensores Químicos |
| Spambase | 4601 | 57 | E-mails |
| Waveform | 5000 | 40 | Física |
| DUWWTP | 527 | 38 | Sensores de Água |
| Shuttle | 58000 | 9 | Transportadoras |
| Slide | 53500 | 384 | Imagens de Tomografia |

A Tabela 5.2 apresenta os algoritmos utilizados nos experimentos, estes algoritmos foram apresentados no Capítulo 3 e no Capítulo 4.

**Tabela 5.2:** Relação dos algoritmos utilizados nos experimentos e os capítulos onde foram descritos.

| Algoritmo | Capítulo |
|---|---|
| ***k*-NN** | 3 |
| ***k*M*k*NN** | 3 |
| **KMC-OTI-FS** | 3 |
| **KMC-EOTI-FS** | 3 |
| ***k*M++*k*NN** | 4 |

## 5.4 – Estratégia de Validação Cruzada

O processo de experimentação utiliza a estratégia de validação cruzada. Especificamente a estratégia utilizada é conhecida como *s-fold* (Bishop, 2011) e será realizada com 10 dobras. O método de validação cruzada *s-fold* não foi usado para validação propriamente, como é caso de muitos trabalhos em aprendizado supervisionado. O método foi escolhido pois é empregado em trabalhos correlatos a este, e também por que permite gerar um número maior de testes do seria possível apenas com, por exemplo, uma busca sobre a número de instâncias dos conjuntos de instâncias utilizados.

Nestas condições, sendo $N$ o número de instâncias de um conjunto de instâncias, cada dobra contém $N/10$ instâncias de teste e $N' = N - N/10$ instâncias de treinamento. Assim, o número de buscas em cada experimento é dado por $N/10$ instâncias de teste vezes 10 dobras, ou seja, é igual a $Q = N$ buscas. Para um algoritmo que realiza busca exaustiva de cada instância de teste nas instâncias de treinamento, como é o caso do algoritmo *k*-NN, o número de cálculos de distância (denotado por $c_{k\text{-NN}}$) a ser realizado é igual a

$$c_{k-\text{NN}} = Q\,N' = N(N - N/10) = N^2 - N^2/10. \quad (5.1)$$

Por exemplo, considerando o conjunto de instâncias Arcene, que possui $N$ = 900 instâncias, cada experimento com este conjunto realiza $Q$ = 900 buscas em conjuntos de treinamento de $N' = 900 - 900/10 =$ 810 instâncias e o número de cálculos de distância, realizado por um algoritmo que faz buscas exaustivas como o *k*-NN, é igual a $c_{k-\text{NN}} = 900^2 - 900^2/10$ = 7.29 x $10^5$ cálculos de distâncias.

A codificação utiliza um algoritmo da biblioteca *scikit-learn* (Scikit Learn, 2022) fazendo implementando a estratégia *s-fold*.

## 5.5 – Índices de Desempenho

A comparação entre algoritmos é realizada via índices de desempenho definidos em relação ao desempenho do algoritmo *k*-NN. Dois índices são definidos, um deles para indicar o desempenho de um algoritmo relativo a cálculos de distância e o outro em relação à redução do tempo de computação. Estes índices são os seguintes:

- RRDC (Pan *et al*., 2020), taxa de redução de cálculos de distância, dada em termos percentuais por RRDC = 100 (1 – $c/c_{k\text{-NN}}$), onde $c$ é o número de cálculos de distância realizados por um algoritmo em um experimento com um conjunto de instâncias e $c_{k\text{-NN}}$ , dado pela Equação 5.1, é o número de cálculos de distância realizado pelo algoritmo *k*-NN em um experimento com o mesmo conjunto de instâncias.
- RRST (Pan *et al*., 2020), taxa de redução do tempo de busca, dada em termos percentuais por RRST = 100 (1 – $t/ t_{k\text{-NN}}$), onde $t$ é o tempo gasto por um algoritmo no experimento com um conjunto de instâncias e $t_{k\text{-NN}}$ é o tempo gasto pelo algoritmo *k*-NN em um experimento com mesmo conjunto de instâncias.

Por exemplo, consideremos experimentos o conjunto de instâncias Arcene, que possui $N$ = 900 instâncias, e os seguintes dados experimentais envolvendo um Algoritmo A e o Algoritmo *k*-NN:

- Algoritmo A: $c = 2.92 \times 10^5$ cálculos de distância computados em $t$ = 97 segundos.
- Algoritmo *k*-NN: $c_{k-\text{NN}} = 900^2 - 900^2/10 = 7.29 \times 10^5$ cálculos de distâncias computados em $t$ = 250 segundos.

Neste cenário experimental o índice RRDC = 100 (1 – $2.92 \times 10^5$ / $7.29 \times 10^5$) = 60 %, indica que o Algoritmo A reduziu em sessenta porcento o número de cálculos de distância em relação ao algoritmo *k*-NN. Já o índice RRST = 100 (1 – 97 / 250) = 61 %, indica que o Algoritmo A reduziu em sessenta e um porcento o tempo de computação em relação ao algoritmo *k*-NN.

## 5.6 – Método de Experimentação

Na experimentação realizou-se seguintes etapas:

(1) Obtenção dos conjuntos de instâncias no repositório UCI (Dua & Graff, 2021).

Os conjuntos de instâncias utilizados são de uso livre.

(2) Execução do algoritmo *k*-NN via interpretador Python Shell e obtenção do número de cálculos de distância e do tempo de computação em cada experimento envolvendo cada um dos conjuntos de instâncias. Para cada conjunto de instância serão realizados quatro experimentos, cada um deles para o valor de *k* igual a 10, 50, 100 e 200, em que *k* é o número de instâncias mais similares a uma instância *x* fornecida como entrada, a serem computadas.

(3) Execução dos algoritmos *k*M*k*NN, KCM-OTI-FS, KMC-EOTI-FS e *k*M++*k*NN através do Python Shell e obtenção do número de cálculos de distância e do tempo de computação em cada experimento envolvendo cada um dos conjuntos de instâncias. Para cada conjunto de instância serão realizados quatro experimentos, cada um deles para o valor de *k* igual a 10, 50, 100 e 200, em que *k* é o número de mais similares a serem computadas.

(4) Cálculo do índice *RRDC* para cada um dos algoritmos *k*M*k*NN, KCM-OTI-FS, KMC-EOTI-FS e *k*M++*k*NN.

(5) Cálculo do índice RRST para cada um dos algoritmos *k*M*k*NN, KCM-OTI-FS, KMC-EOTI-FS e *k*M++*k*NN.

(6) Tabulação dos resultados obtidos nas etapas (2), (3), (4) e (5).

Na tabulação dos resultados utiliza-se aplicação de planilha eletrônica, com o intuito de apresentar lado a lado os resultados obtidos.

(7) Comparação dos índices RRDC e RRST entre algoritmos, a fim de verificar as taxas de redução em relação ao algoritmo *k*-NN.

A Figura 5.2 ilustra os passos deste método de experimentação em forma de fluxograma, e como pode ser reproduzido.

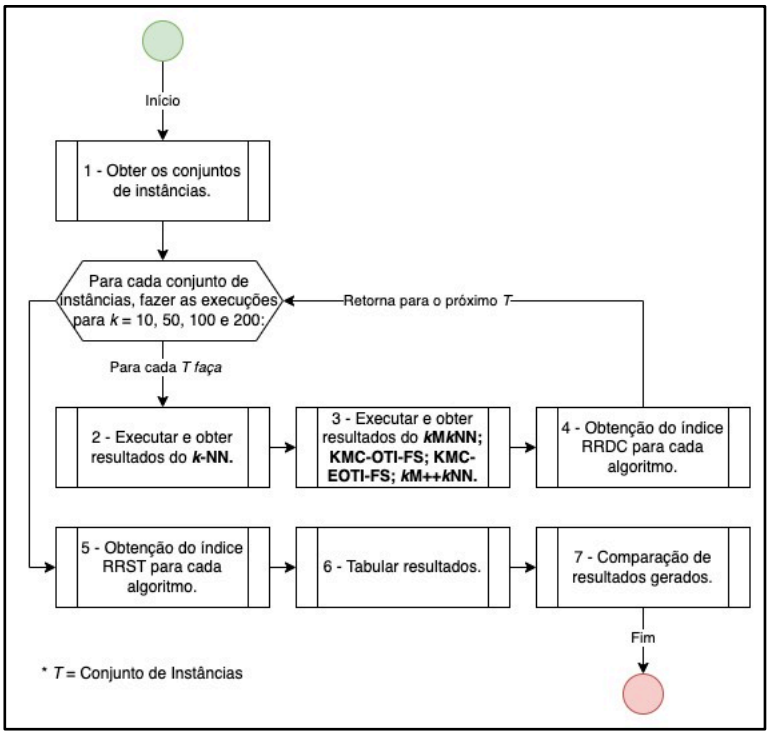



**Figura 5.2:** Fluxograma do método de experimentação para implementação, execução e obtenção de resultados dos algoritmos.

Na execução dos experimentos, após a realização da Etapa 1, há uma repetição das Etapas 2, 3 e 4, é importante ressaltar de que, para cada um dos conjuntos de instâncias obtidos, estas instâncias serão embaralhadas antes de serem fornecidas como entrada dos algoritmos para suas execuções.

# Capítulo 6
# Resultados e Discussão

*Este capítulo apresenta os resultados dos experimentos realizados de acordo com o método de experimentação detalhado no Capítulo 5. A Seção 6.1 apresenta os resultados dos experimentos realizados e a Seção 6.2 discute estes resultados.*

## 6.1 – Resultados

Os resultados apresentados nesta seção foram obtidos empregando o método de experimentação detalhado no Capítulo 5. A Tabela 6.1 sintetiza os resultados, utilizando os seguintes símbolos e siglas: $T$ – Conjunto de instâncias; $N$ − número de instâncias de $T$; $D$ − número de dimensões das instâncias de $T$; $k$ − número de instâncias mais similares buscadas; $c_{k\text{-NN}}$ − número de cálculos de distâncias; $t_{k\text{-NN}}$ − tempo de execução; RRDC − taxa de redução de cálculos de distância; RRST − taxa de redução de tempo de busca. O tempo de execução do algoritmo $k$-NN está estabelecido em segundos e os índices RRDC e RRST estão descritos em termos percentuais.

Os melhores índices RRDC e RRST, na comparação geral entre todos os algoritmos experimentados, estão destacados em negrito. Especificamente, na comparação particular entre os algoritmos $k$M++$k$NN e $k$M$k$NN, os melhores índices RRDC e RRST estão destacados entre parênteses.

**Tabela 6.1:** Resultados dos Experimentos.

| ***T***: *N*, *D* | ***k*** | ***k*-NN: $c_{k\text{-NN}}$ \| $t_{k\text{-NN}}$ (s)** | ***k*M*k*NN: RRDC (%) \| RRST (%)** | **KMC-OTI-FS: RRDC (%) \| RRST (%)** | **KMC-EOTI-FS: RRDC (%) \| RRST (%)** | ***k*M++*k*NN: RRDC (%) \| RRST (%)** |
|---|---|---|---|---|---|---|
| Arcene: 900, 10000 | 10 | $7.29 \times 10^5$ \| 3773 | 64 \| 59 | 70 \| 62 | 58 \| **65** | **(75)** \| (61) |
| | 50 | | 61 \| 60 | 71 \| 56 | 59 \| **61** | **(74)** \| (61) |
| | 100 | | 60 \| 58 | 58 \| 52 | 55 \| 56 | **(71)** \| (60) |
| | 200 | | 59 \| 56 | 48 \| 41 | 44 \| 45 | **(71)** \| **(58)** |
| DUWWTP: 527, 38 | 10 | $2.50 \times 10^5$ \| 237 | 86 \| 76 | 89 \| 74 | 89 \| **85** | **(90)** \| **(78)** |
| | 50 | | 71 \| 56 | 77 \| 56 | 73 \| 59 | **(78)** \| **(61)** |
| | 100 | | 58 \| 40 | 66 \| 43 | 64 \| 48 | **(72)** \| **(65)** |
| | 200 | | 42 \| 24 | 46 \| 24 | 45 \| 28 | **(72)** \| **(59)** |
| GasSensors: 13910, 128 | 10 | $1.74 \times 10^8$ \| 923 | 93 \| 92 | **99** \| 92 | **99** \| 94 | (97) \| **(93)** |
| | 50 | | 94 \| 88 | **99** \| 88 | 98 \| 91 | (96) \| **(93)** |
| | 100 | | 90 \| 86 | **98** \| 89 | 97 \| **91** | (95) \| **(91)** |
| | 200 | | 90 \| 78 | **94** \| 80 | 91 \| **84** | **(94)** \| (83) |
| Multiplefeatures: 2000, 649 | 10 | $3.60 \times 10^6$ \| 1584 | 85 \| 80 | **93** \| 83 | 90 \| **86** | **(93)** \| (83) |
| | 50 | | 78 \| 72 | 81 \| **78** | 80 \| 73 | **(88)** \| (76) |
| | 100 | | 73 \| 66 | 77 \| **72** | 75 \| **72** | **(82)** \| (70) |
| | 200 | | 68 \| 58 | 70 \| 60 | 68 \| **62** | **(74)** \| (61) |
| Slice: 53500, 384 | 10 | $2.58 \times 10^9$ \| 106272 | 95 \| 92 | **99** \| 91 | **99** \| 93 | **(99)** \| **(98)** |
| | 50 | | 94 \| 85 | **99** \| 86 | 98 \| 89 | (98) \| **(92)** |
| | 100 | | 91 \| 81 | **99** \| 77 | 98 \| 80 | (98) \| **(89)** |
| | 200 | | 90 \| 75 | **98** \| 68 | 96 \| 71 | (96) \| **(83)** |
| Shuttle: 58000, 9 | 10 | $3.03 \times 10^9$ \| 4128 | 97 \| 84 | **99** \| 78 | **99** \| **92** | **(99)** \| (89) |
| | 50 | | 96 \| 83 | **99** \| 68 | 97 \| **90** | (98) \| (89) |
| | 100 | | 95 \| 82 | **99** \| 56 | 97 \| 87 | (98) \| **(90)** |
| | 200 | | 93 \| 77 | **99** \| 47 | 96 \| **87** | (97) \| **(87)** |
| Spambase: 4601, 57 | 10 | $1.91 \times 10^7$ \| 800 | 88 \| 79 | **93** \| 84 | **93** \| **90** | (92) \| (84) |
| | 50 | | 86 \| 77 | **92** \| 83 | 91 \| **89** | (90) \| (81) |
| | 100 | | 83 \| 73 | **91** \| 76 | 90 \| **87** | (89) \| (79) |
| | 200 | | 76 \| 61 | **89** \| 73 | 88 \| **77** | (82) \| (70) |
| Waveform: 5000, 40 | 10 | $2.25 \times 10^7$ \| 440 | 11 \| 8 | **23** \| -14 | 19 \| **9** | **(23)** \| **(9)** |
| | 50 | | 7 \| 3 | 13 \| -31 | 10 \| **5** | **(19)** \| **(5)** |
| | 100 | | 5 \| -1 | 1 \| -35 | 0 \| -7 | **(12)** \| **(1)** |
| | 200 | | 3 \| -9 | 0 \| -49 | 0 \| -16 | **(12)** \| **(-7)** |
| | | **Média** | 71 \| 62 | 76 \| 57 | 74 \| 66 | **(79)** \| **(69)** |

## 6.2 – Discussão

### *Experimentos deste Trabalho versus Experimentos Publicados*

Pan *et al*. (2020) relata experimentos com os algoritmos *k*-NN, *k*M*k*NN, KMC-OTI-FS, KMC-EOTI-FS utilizando sobre os mesmos conjuntos de instâncias utilizados nos experimentos descritos neste trabalho. Xueyi (2012) também relata experimentos com os algoritmos *k*-NN e *k*M*k*NN utilizando os conjuntos de instâncias Arcene, Multiplefeatures e Spambase.

Os resultados dos experimentos com os algoritmos *k*-NN, *k*M*k*NN, KMC-OTI-FS, KMC-EOTI-FS apresentados nesse trabalho, são consistentes com os resultados relatados em Pan *et al.* (2020) e Xueyi (2012) tanto em relação à taxa RRDC quanto em relação à taxa RRST. Diferenças entre os experimentos apresentados nesse trabalho e os relatados nos trabalhos citados são compreensíveis, considerando que as instâncias de cada conjunto de instâncias são embaralhadas no início de cada experimento e a ordem em que elas são tomadas pelos algoritmos afeta a taxa de redução de cálculos de distância e a taxa de redução do tempo de busca.

### ***k*M++*k*NN *Versus k*M*k*NN**

Em todos os experimentos realizados, o algoritmo *k*M++*k*NN apresentou maior taxa de redução de cálculos de distância e maior taxa de redução de tempo de busca do que o algoritmo *k*M*k*NN. A Tabela 6.1 destaca entre parênteses os maiores valores, experimento por experimento, das taxas RRDC e RRST. Para os experimentos realizados, o algoritmo *k*M++*k*NN apresentou a média de 79% da taxa RRDC contra a média de 71% apresentada pelo algoritmo *k*M*k*NN. Em média, a redução do tempo de busca também é favorável ao algoritmo *k*M++*k*NN, sendo de 69% da taxa RRST contra 62% apresentada pelo algoritmo *k*M*k*NN.

### ***k*M++*k*NN *Versus Outros Algoritmos Rápidos Estudados***

Para os experimentos realizados, o desempenho médio do algoritmo *k*M++*k*NN, medido pelos índices RRDC e RRST, é melhor do que todos os demais algoritmos estudados. A última linha da Tabela 6.1 apresenta a média dos índices RRDC e RRST obtido nos experimentos realizados. O algoritmo *k*M++*k*NN apresentou índice RRDC médio igual 79% contra 76% do algoritmo KMC-OTI-FS e 74% do algoritmo KMC-EOTI-FS. O índice médio RRST igual a 69%, apresentado pelo algoritmo *k*M++*k*NN, também supera

os índices médios RRST apresentados pelos algoritmos KMC-OTI-FS (57%) e KMC-EOTI-FS (66%).

A complexidade de espaço do algoritmo *k*M++*k*NN é igual à complexidade de espaço do algoritmo *k*M*k*NN, afinal ambos utilizam as mesmas estruturas de dados. Denominando S(*N*) a complexidade de espaço do algoritmo *k*M*k*NN, Pan *et al*. (2020) calcula a complexidade de espaço do algoritmo KMC-OTI-FS em *M* S(*N*) e a complexidade de espaço do algoritmo KMC-EOTI-FS em 2 S(*N*), em que *N* é o número de instâncias do conjunto de instâncias, dado como entrada dos algoritmos, e *M* é o número de grupos formados na fase de pré-processamento dos algoritmos. Em aplicações envolvendo conjunto de instâncias com grande número de instâncias e/ou instâncias com alta dimensionalidade, em que o consumo de memória é um fator crítico, a opção por algoritmos com complexidade de espaço mais baixa pode ser definitiva, o que torna o algoritmo KMC-OTI-FS apenas teórico, inviável na prática, e pode fazer o algoritmo KMC-EOTI-FS ser rejeitado em função da possibilidade de usar um outro algoritmo que consome a metade da memória que ele requer, como é o caso do algoritmo *k*M++*k*NN.

Além disso, é preciso destacar que os algoritmos KMC-OTI-FS e KMC-EOTI-FS utilizam estratégias de economia de cálculo de distâncias mais complicadas no sentido de serem programadas do que os algoritmos *k*M++*k*NN.

***Existem Explicações para estes Resultados?***

Existem explicações para estes resultados? Uma resposta conclusiva a esta pergunta ainda requer investigações, mas é possível que os resultados sejam explicados pela principal diferença entre os algoritmos *k*M++*k*NN e os algoritmos *k*M*k*NN, KMC-OTI-FS, KMC-EOTI-FS: enquanto o primeiro usa na fase de pré-processamento o algoritmo *k*-Means++ (Arthur, D. & Vassilvitskii, 2007), todos os demais algoritmos utilizam o algoritmo *k*-Means (MacQueen, 1967). O ponto de partida dos algoritmos *k*-Means++ e *k*-Means é a escolha dos centroides iniciais dos grupos que comporão o agrupamento final que será induzido pelos algoritmos. Ressalta-se que o agrupamento final induzido pelos algoritmos depende da escolha dos centroides iniciais dos grupos. O algoritmo *k*-Means sorteia de forma aleatória os centroides iniciais dos grupos e isto pode produzir situações nem sempre desejáveis como, por exemplo, escolher para centroides instâncias com valores extremos ou instâncias muito próximas (similares) uma das outras. Já o *k*-Means++ utiliza uma estratégia de escolha de centroides iniciais que tende a eleger instâncias mais distantes umas das outras. Isto faz com que o agrupamento formado na fase de pré-

processamento do algoritmo *k*M++*k*NN contenham grupos mais distantes uns dos outros, otimizando assim o uso das estratégias de economia de cálculos de distâncias.

# Capítulo 7
# Conclusões e Trabalhos Futuros

*Este capítulo descreve em sua Seção 7.1 as conclusões deste trabalho. Trabalhos futuros são descritos na Seção 7.2.*

## 7.1 – Conclusões

Este trabalho propõe o algoritmo *k*M++*k*NN como alternativa para acelerar a fase de busca do algoritmo *k*-NN e o compara experimentalmente com os algoritmos *k*M*k*NN (Xueyi, 2012), KMC-OTI-FS e KMC-EOTI-FS (Pan *et al.,* 2020), atualmente considerados rápidos para o cálculo exato das *k* instâncias mais similares a uma instância *x* em um conjunto de instâncias *T*. Comum a estes algoritmos está o fato de que eles utilizam técnicas que os classificam, como definido no Capítulo 1, na categoria ENN, subcategoria baseada em grupos, e que empregam o conceito de desigualdade triangular para acelerar a fase de busca do algoritmo *k*-NN.

Considerando os experimentos realizados, os resultados sugerem que o desempenho médio do algoritmo *k*M++*k*NN é superior ao dos outros algoritmos rápidos estudados, tanto em redução de cálculos de distâncias entre instâncias (cálculos de funções de similaridade), quanto em redução de tempo de busca. Ademais é importante mencionar que o algoritmo *k*M++*k*NN tem complexidade de espaço igual à do algoritmo *k*M*k*NN, mas inferior às complexidades de espaço dos algoritmos KMC-OTI-FS e KMC-EOTI-FS.

Embora mais estudos e experimentos precisem ser realizados no futuro para compreender melhor a relação de desempenho entre os algoritmos *k*M++*k*NN e os demais algoritmos estudados, estes resultados são animadores e inspiram novos trabalhos.

## 7.2 – Trabalhos Futuros

Experimentos com mais conjuntos de instâncias podem contribuir para entender melhor a relação de desempenho entre o algoritmo proposto e os demais algoritmos estudados.

O algoritmo *k*M++*k*NN, assim como os demais algoritmos estudados, formam na fase de pré-processamento $M = \sqrt{N}$ grupos. Xueyi (2012) e Pan *et al*. (2020) afirmam que este número foi escolhido empiricamente. No entanto, é razoável hipotetizar que o valor de *M* afeta o desempenho do algoritmo *k*M++*k*NN e tenha relação não só com o número *N* de instâncias, mas também com outras propriedades do conjunto de instâncias, dado como entrada do algoritmo. Estudos que investiguem a relação entre o número *M* e o desempenho do algoritmo *k*M++*k*NN podem contribuir para a sua melhoria.

Os algoritmos *k*M*k*NN, *k*M++*k*NN, KMC-OTI-FS e KMC-EOTI-FS empregam diferentes estratégias de desigualdade triangular, mas evidentemente não esgotam a possibilidade

de uso do conceito de desigualdade triangular para acelerar a fase de busca do algoritmo *k*-NN. Novas estratégias de uso do conceito de desigualdade triangular podem ser propostas.

Os algoritmos estudados empregam métodos de agrupamento para particionar o espaço do conjunto de instâncias e, com isto, estabelecem estratégias de exclusão de grupos e de instâncias dentro de grupos, visando a otimização do tempo de busca. Por outro lado, algoritmos hierárquicos, utilizam a ideia de mapeamento do espaço do conjunto de instâncias para melhorar a eficiência da busca. Sem deixar de lado métodos de agrupamento, é possível que a hierarquização dentro de cada grupo possa também oferecer melhoria de desempenho da fase de busca do algoritmo *k*-NN.

Por fim, aplicações práticas frequentemente exigem inclusões, exclusões e alterações de instâncias de um conjunto de instâncias de forma dinâmica. O estudo de métodos eficientes para alterar dinamicamente as estruturas de dados pré-processadas utilizadas pelo algoritmo *k*M++*k*NN é um outro importante campo de investigação.

**Referências**


Andoni, A., & Indyk, P. (2008). Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1), 117-122.

Angiulli, F., Basta, S., & Pizzuti, C. (2005). Distance-based detection and prediction of outliers. *IEEE transactions on knowledge and data engineering*, 18(2), 145-160.

Anton, H. (1994). *Elementary Linear Algebra* (7th ed.). Edited by Nadia Magnenat-Thalmann and Daniel Thalmann, John Wiley & Sons ltd.

Arthur, D., & Vassilvitskii, S. (2007). K-Means++: The Advantages of Careful Seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms,* 1027–1035.

Berthold, M. R.; Borgelt, C.; Höppner, F.; Klawonn, F. (2010). Guide to Intelligent Data Analysis, Texts in Computer Science, D. Gries & F. B. Schneider (Eds.), Springer-Verlag.

Bishop, C. M. (2011). Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag.

Black, P. (2019), Manhattan distance, in Dictionary of Algorithms and Data Structures. On-line in 2021/06/12. Available in https://www.nist.gov/dads/HTML/manhattanDistance.html.

Chakraborty, S., & Yeh, C.H. (2007). A Simulation Based Comparative Study of Normalization Procedures in Multiattribute Decision Making. In *Proceedings of the 6th Conference on 6th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases,* 6*,* 102–109.

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1), 21-27.

Dua, D. & Graff, C. (2021). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml].

GeeksforGeeks (2021). Top 10 Algorithms every Machine Learning Engineer should know. On-line in 2021/09/22. Available in https://www.geeksforgeeks.org/top-10-algorithms-every-machine-learning-engineer-should-know/.

Gerhana, Y. A., Atmadja, A. R., Zulfikar, W. B., & Ashanti, N. (2017). The implementation of K-nearest neighbor algorithm in case-based reasoning model for forming automatic answer identity and searching answer similarity of algorithm case. In *2017 5th International Conference on Cyber and IT Service Management (CITSM). 2017 5th International Conference on Cyber and IT Service Management (CITSM)*, 1-5.

Glowacz, A., & Glowacz, Z. (2016). Recognition of images of finger skin with application of histogram, image filtration and K-NN classifier. *Biocybernetics and biomedical engineering*, 36(1), 95-101.

JetBrains (2022). JetBrains: Essential tools for software developers and teams. On-line in 2022/04/22. Available in https://www.jetbrains.com/.

Kotsiantis, S. B., Zaharakis, I. D., & Pintelas, P. E. (2006). Machine learning: a review of classification and combining techniques. *Artificial Intelligence Review*, 26(3), 159-190.

Le, James (2021). The Top 10 Machine Learning Algorithms Every Beginner Should Know. On-line in 2021/09/22. Available in https://builtin.com/data-science/tour-top-10-algorithms-machine-learning-newbies.

Lima, E. L. (2013). Espaços métricos. 5ª ed. IMPA.

MacQueen, J. B. (1967). Classification and analysis of multivariate observations. In *5th Berkeley Symposium on Mathematical Statistics and Probability*, 1, 281-297.

Maesschalck, R., Jouan-Rimbaud, D., & Massart, D. L. (2000). The mahalanobis distance. *Chemometrics and intelligent laboratory systems*, 50(1), 1-18.

Mitchell, T. M. (1997) Machine Learning. McGraw-Hill Education.

NumFocus (2022). NumFOCUS: A Nonprofit Supporting Open Code for Better Science. On-line in 2022/04/22. Available in https://numfocus.org/.

NumPy (2022). The fundamental package for scientific computing with Python. On-line in 2022/04/22. Available in https://numpy.org/.

Pan, Y., Pan, Z., Wang, Y., & Wang, W. (2020). A new fast search algorithm for exact k-nearest neighbors based on optimal triangle-inequality-based check strategy. *Knowledge-Based Systems*, 189, 105088.

ProjectPro (2021). Common Machine Learning Algorithms for Beginners. On-line in 2021/09/22. Available in https://www.projectpro.io/article/common-machine-learning-algorithms-for-beginners/202.

Python (2022). Python: Programming Language. On-line in 2022/04/22. Available in https://www.python.org/.

Scikit Learn (2022). Scikit-learn – Machine Learning in Python. On-line in 2022/04/22. Available in https://scikit-learn.org/stable/.

Sproull, R. F. (1991). Refinements to nearest-neighbor searching in k-dimensional trees. *Algorithmica*, 6(1), 579-589.

Theodoridis, S., & Koutroumbas, K. (2009). Pattern recognition. Academic Press.

Uhlmann, J.K. (1991). Satisfying General Proximity/Similarity Queries with Metric Trees. *Information Processing. Letters,* 40(4), 175-179.

Wang, Y., Pan, Z., & Li, R. (2019). A new cell-level search based non-exhaustive approximate nearest neighbor (ann) search algorithm in the framework of product quantization. *IEEE Access*, 7, 37059-37070.

Xueyi, W. (2012). A Fast Exact k-Nearest Neighbors Algorithm for High Dimensional Search Using k-Means Clustering and Triangle Inequality. *Proceedings of International Joint Conference on Neural Networks / co-sponsored by Japanese Neural Network Society (JNNS) [et al.]. International Joint Conference on Neural Networks,* 43(6), 2351-2358.

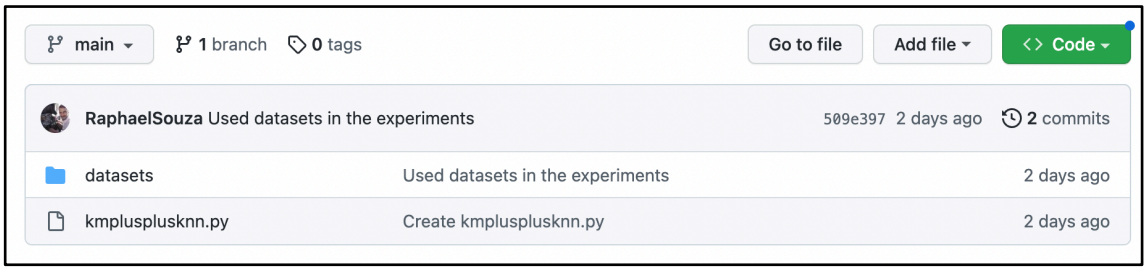
Yianilos, P. N. (1993). Data structures and algorithms for nearest neighbor. In *Proceedings of the ACM-SIAM Symposium on Discrete algorithms*, 66, 311.

# Apêndice I
# Repositório para o Código Fonte do algoritmo *k*M++*k*NN, Escrito em Phyton, e Conjuntos de Instâncias Utilizados

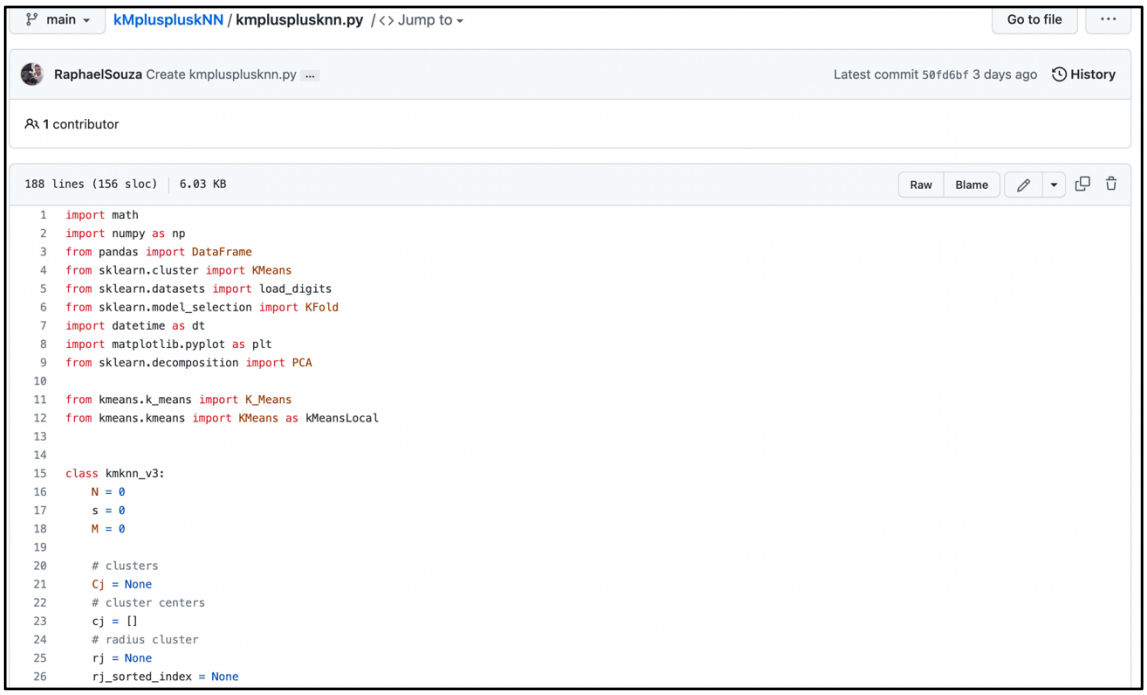
*Este apêndice apresenta o repositório para o código fonte do algoritmo kM++kNN, escrito em Phyton. O repositório também contém os conjuntos de instâncias utilizados nos experimentos descritos nesta dissertação.*

**Como acessar o código fonte do *k*M++*k*NN**

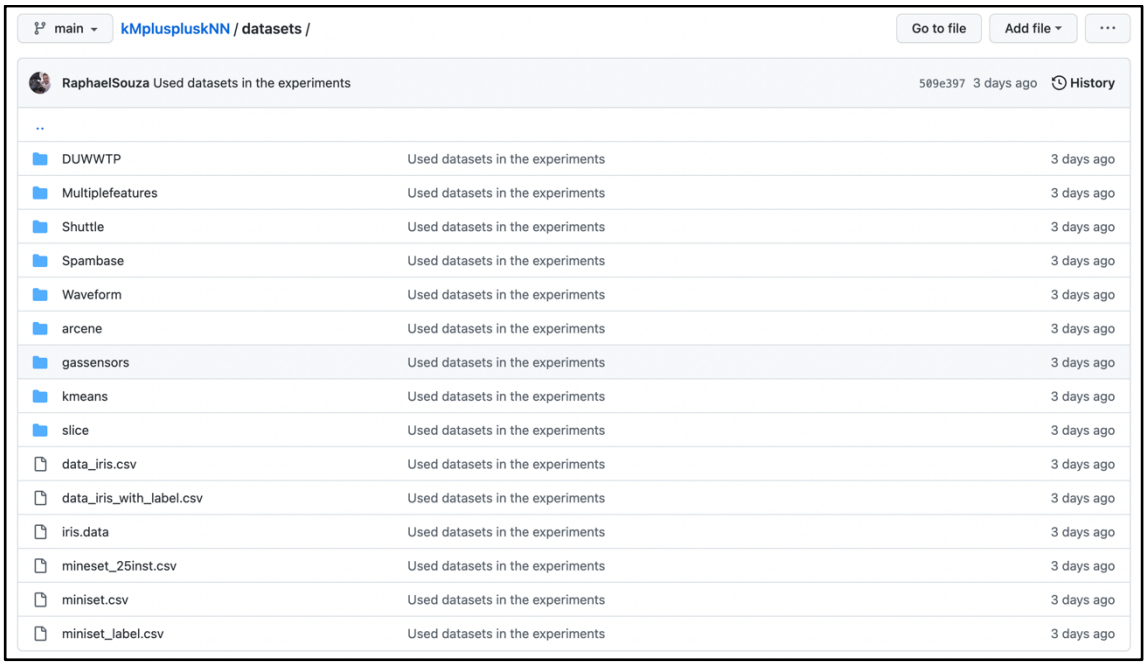
O código fonte do algoritmo *k*M++*k*NN esta disponibilizado no repositório https://github.com/RaphaelSouza/kMplusplusKNN. Acessando o repositório, o usuário terá condição de baixar o código fonte do algoritmo *k*M++*k*NN e os conjuntos de instâncias utilizados nos experimentos deste trabalho. A imagem abaixo apresenta o arquivo para o algoritmo escrito em linguagem Python e a pasta com os conjuntos de instâncias.




Ao clicar no arquivo do algoritmo *k*M++*k*NN, é possível visualizar o código fonte. Conforme citado durante este trabalho, é recomendado a utilização da IDE PyCharm para a execução de testes deste algoritmo.




Os conjuntos de instâncias utilizados também foram disponibilizados e encontra-se na pasta datasets do repositório, adicionalmente foi disponibilizado neste repositório conjuntos de instâncias clássicos, a fim de, serem utilizados para testes aos que se interessarem, como exemplo, o conjunto *íris*.


main
kMpluspluskNN / datasets /
Go to file
Add file
RaphaelSouza Used datasets in the experiments
509e397 3 days ago
History
..
DUWWTP Used datasets in the experiments 3 days ago
Multiplefeatures Used datasets in the experiments 3 days ago
Shuttle Used datasets in the experiments 3 days ago
Spambase Used datasets in the experiments 3 days ago
Waveform Used datasets in the experiments 3 days ago
arcene Used datasets in the experiments 3 days ago
gassensors Used datasets in the experiments 3 days ago
kmeans Used datasets in the experiments 3 days ago
slice Used datasets in the experiments 3 days ago
data_iris.csv Used datasets in the experiments 3 days ago
data_iris_with_label.csv Used datasets in the experiments 3 days ago
iris.data Used datasets in the experiments 3 days ago
mineset_25inst.csv Used datasets in the experiments 3 days ago
miniset.csv Used datasets in the experiments 3 days ago
miniset_label.csv Used datasets in the experiments 3 days ago