



*Impacto do Uso da Desigualdade Triangular
para Acelerar o Algoritmo k-Means*

Marcelo Kuchar Matte

Fevereiro / 2020

Dissertação de Mestrado em Ciência da
Computação

Impacto do Uso da Desigualdade Triangular para Acelerar o Algoritmo k-Means

Esse documento corresponde à dissertação apresentada à Banca Examinadora da Dissertação no curso de Mestrado em Ciência da Computação do UNIFACCAMP – Centro Universitário Campo Limpo Paulista.

Campo Limpo Paulista, 04 de fevereiro de 2020.

Marcelo Kuchar Matte

Prof. Dra. Maria do Carmo Nicoletti (Orientadora)

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

M387i

Matte, Marcelo Kuchar

Impacto do uso da desigualdade triangular para acelerar o algoritmo k-Means / Marcelo Kuchar Matte. Campo Limpo Paulista, SP: Unifaccamp, 2020.

Orientadora: Prof^a. Dr^a. Maria do Carmo Nicoletti

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

1. k-Means. 2. Otimização. 3. Desigualdade triangular. I. Nicoletti, Maria do Carmo. II. Campo Limpo Paulista. III. Título.

CDD – 005.1

Resumo O trabalho de pesquisa descrito nesta dissertação teve, como foco principal, a investigação de propostas de otimização de desempenho do algoritmo k-Means que utilizam a chamada desigualdade triangular. O k-Means pode ser descrito como um processo que busca particionar um conjunto de instâncias de dados em K grupos, em que K é um parâmetro geralmente fornecido pelo usuário. O algoritmo é considerado relativamente escalável e eficiente no processamento de conjuntos de instâncias volumosos; entretanto, pode ter um elevado custo computacional, dependendo do conjunto de instâncias fornecido. O algoritmo investe a maior parte de seu tempo de execução em cálculos de distância. Porém, muitos desses cálculos podem ser desnecessários e podem ser evitados utilizando estratégias adequadas baseadas em geometria. Neste trabalho foram investigadas cinco estratégias de aceleração propostas na literatura, em que o uso da desigualdade triangular foi empregado, de maneira distinta entre elas. As estratégias de aceleração investigadas foram: compare-Means, sort-Means, k-Means-Elkan, k-Means-Yinyang e k-Means-Fission-Fusion. Os experimentos conduzidos e discutidos neste trabalho evidenciam que o uso de tais estratégias, consome um tempo de execução inferior ao tempo de execução consumido pelo k-Means padrão. É importante observar que o k-Means utilizando tais estratégias produz, em geral, os mesmos resultados gerados pelo k-Means original, para as mesmas entradas i.e., conjunto de instâncias, o valor do parâmetro K e o conjunto inicial de centroides.

Palavras-chave: k-Means, otimização, desigualdade triangular.

Abstract The research work described in this dissertation had, as main focus, the investigation of proposals for the performance optimization of the k-Means algorithm, using the so-called triangular inequality. The k-Means can be described as a process that seeks to partition a set of data instances into K groups (clusters), where K is a user-supplied parameter. The algorithm is considered to be relatively scalable and efficient in processing large instance sets; however, it may have a high computational cost, depending on the given set of instances. The algorithm invests most of its runtime in distance calculations. However, many of the calculations may be unnecessary and can be avoided by using appropriate geometry-based strategies. In this work five acceleration strategies proposed in the literature, where the triangular inequality was used by all of them, with different purposes, were investigated. The acceleration strategies were: compare-Means, sort-Means, k-Means-Elkan, k-Means-Yinyang, and k-Means-Fission-Fusion. The experiments conducted and discussed in this dissertation show that by using such strategies the k-Means consumes a runtime less than the runtime consumed by the standard k-Means. It is important to remember that the k-Means having an embedded acceleration strategy generally produces the same results as those produced by the original k-Means, given that the same entries i.e., instance set, K parameter value, and initial set of centroid, are provided.

Keywords: k-Means, optimization, triangular inequality.

Dedicatória

Dedico essa dissertação à minha mãe Célia Kuchar Matte que sempre me apoiou na vida e nos estudos, e aos meus familiares e amigos por estarem ao meu lado, mesmo quando devido a necessidade de estudo, eu não consegui oferecer a mesma reciprocidade.

Dedico também a Deus e ao meu Mentor Espiritual, que sempre me fortalece nos momentos de dificuldade.

Agradecimentos

Agradeço primeiramente a Deus por me conduzir, sustentar e dar a sabedoria necessária para realização deste projeto.

À minha orientadora professora Dra. Maria do Carmo Nicoletti pela dedicação, paciência, motivação e ensinamentos durante todo o processo de desenvolvimento desta dissertação, uma pessoa a qual não terei como agradecer o suficiente nessa encarnação. Dessa forma, não poderia deixar de citar o seguinte proverbio chinês: “*One day as your teacher, like a father for a lifetime*”. Por isso, rogo a Deus que sempre a ampare e a fortaleça na jornada de educadora e pesquisadora, a qual faz com excelência.

Aos professores Dr. Osvaldo Luiz de Oliveira, Dr. José Hiroki Saito pela disposição e boa vontade em avaliar e sugerir melhorias nesta pesquisa.

Aos amigos Sandro Vieira Gonçalves, Yulle Gustavo Siqueira de Lima e José Chaves da Silva pelo companheirismo durante o mestrado.

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

Ao Instituto Federal de Mato Grosso do Sul - IFMS por permitir realizar esse aperfeiçoamento; e aos professores e funcionários do campus Jardim, por sempre apoiarem e atenderem as demandas relacionadas ao mestrado.

A todos que contribuíram para a realização deste trabalho.

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

Não existe luz no amanhã para o Espírito que dorme.

(Desconhecido)

Sumário

1.	Apresentação	1
2.	Considerações e Principais Aspectos Relacionados à Área de Aprendizado de Máquina	5
	2.1. Considerações Iniciais	5
	2.2. Aprendizado Indutivo de Máquina	6
	2.3. Pré-Processamento de Dados	9
	2.4. Conjunto de Treinamento e Teste	11
	2.5. Validação Cruzada	12
3.	Algoritmos de Agrupamentos	15
	3.1. Considerações Iniciais	15
	3.2. Notações e Definições Básicas	17
	3.3. Categorias dos Algoritmos de Agrupamento	20
	3.4. Medidas de Similaridade	21
	3.5. Índices de Validação de Agrupamento	22
	3.5.1. Índices de Validação Interna	23
	3.5.1.1. Índice Dunn	23
	3.5.1.2. Índice Silhouette	28
	3.5.2. Índices de Validação Externa	32
	3.5.2.1. Índice de Rand	34
	3.5.2.2. Índice de Jaccard	35
4.	Considerações Sobre o Algoritmo K-Means	36
	4.1. Introdução	36
	4.2. Notações e Definições Envolvidos no Algoritmo k-Means	37
	4.2.1. Método dos Mínimos Quadrados	37
	4.3. O algoritmo k-Means	38
	4.3.1. Um Exemplo Didático do k-Means	41
	4.4. Inicialização do k-Means (k-Means++)	44
5.	Revisão da Literatura com Foco em Estratégias para Acelerar o k-Means	47
	5.1. Breve Revisão Bibliográfica	47
6.	O Uso da Desigualdade Triangular para Acelerar o Algoritmo k-Means	53
	6.1. Considerações Iniciais	53
	6.2. Sobre a Desigualdade Triangular e Resultados Teóricos Associados ao Seu Emprego	53
7.	Estratégias de Aceleração do Algoritmo k-Means	58
	7.1. Considerações Iniciais	58
	7.2. Estratégias Para Acelerar o k-Means	58

7.3. O Algoritmo Compare-Means	59
7.3.1. Um Exemplo Completo do Uso do Compare-Means	61
7.4. O Algoritmo Sort-Means	65
7.4.1. Um Exemplo Completo do Uso do Sort-Means	68
7.5. O Algoritmo de Elkan	72
7.5.1. Um Exemplo Completo do Uso do k-Means-Elkan	77
7.6. O Algoritmo k-Means-Yinyang	82
7.6.1. Um Exemplo Completo do Uso do k-Means-Yinyang	87
7.6.2. Propostas de Modificação do k-Means-Yinyang	93
7.7. k-Means Fission-Fusion	95
7.7.1. Considerações Iniciais	95
7.7.2. Conceitos Básicos Utilizados no Algoritmo k-Means-Fission-Fusion	96
7.7.3 Algoritmo k-Means-Fission-Fusion	99
7.7.4 Um Exemplo Completo do Uso do k-Means-Fission-Fusion	102
8. Sistema <i>5Accelerated k-Means</i> , Experimentos e Análise dos Resultados	109
8.1. Considerações Iniciais	109
8.2. Sistema <i>5Accelerated k-Means (5ACCK-Means)</i>	109
8.2.1. Módulo de Leitura do Conjunto de Instâncias	111
8.2.2. Módulo de Pré-processamento	111
8.2.3. Módulo de Execução	113
8.2.4. Módulo de Índices de Validação	114
8.3. Dados, Metodologia, Experimentos e Análise de Resultados	116
8.3.1. Dados Utilizados nos Experimentos	116
8.3.2. Metodologia Adotada na Condução dos Experimentos	125
8.3.3. Experimentos e Análise dos Resultados Obtidos	129
8.3.3.1. Compare-Means	132
8.3.3.2. Sort-Means	134
8.3.3.3. k-Means-Elkan	136
8.3.3.4. k-Means-Yinyang	137
8.3.3.5. k-Means-Fission-Fusion	138
8.4. Considerações Finais sobre os Resultados dos Experimentos Realizados	139
9. Conclusões	141
9.1 Principais Pontos Investigados e Conclusões dos Experimentos	141
9.2 Possíveis Atividades para Continuidade do Trabalho	143
10. Referências	144

Glossário

AIM – Aprendizado Indutivo de Máquina.

AM – Aprendizado de Máquina.

IA – Inteligência Artificial.

MC – Matriz de Confusão.

MS – Matriz de Similaridade.

SQE – Soma dos Quadrados dos Erros

Lista de Tabelas

3.1.	Conjunto C contendo 40 instâncias geradas artificialmente, descritas por dois atributos numéricos.	19
3.2.	Agrupamento contendo 40 instâncias divididas em três grupos, conforme ilustrado na Figura 3.1.	26
3.3.	Valores calculados de a_i , conforme Eq. (3.6).	30
3.4.	Valores calculados de b_i , conforme Eq. (3.8).	31
3.5.	Valores do Silhouette das instâncias, conforme Eq. (3.9).	32
3.6.	Instâncias associadas aos grupos, conforme partições G e partição G'.	33
3.7.	Pares de instâncias do conjunto C, onde PG e PG' representam se os pares de instâncias pertencem ao mesmo grupo ou não, segundo as partições G e G'.	34
3.8.	Matriz de confusão gerada a partir das partições G e G' apresentadas na Tabela 3.6, onde 'sim' expressa que as duas instâncias participam do mesmo grupo na partição e 'não' expressa que não participam do mesmo grupo na partição.	34
7.1.	Conjunto C contendo 10 instâncias geradas artificialmente, descritas por dois atributos numéricos.	62
7.2.	Matriz de distância d da iteração 1 do algoritmo compare-Means.	63
7.3.	Cálculos de distância entre as instâncias e os centroides que foram evitados na iteração 1.	63
7.4.	Matriz de distância d da iteração 2 do algoritmo compare-Means.	63
7.5.	Cálculos de distância entre as instâncias e os centroides que foram evitados na iteração 2.	64
7.6.	Distâncias inter-centroides, considerando os quatro centroides da Figura 7.6.	66
7.7.	Matriz de distância d da iteração 1 do algoritmo sort-Means.	69
7.8.	Matriz de permutação O da iteração 1 do algoritmo sort-Means.	69
7.9.	Cálculos de distância entre as instâncias e centroides que foram evitados na iteração.	69
7.10.	Matriz de distância d da iteração 2 do algoritmo sort-Means.	70
7.11.	Matriz de permutação O da iteração 2 do algoritmo sort-Means.	70
7.12.	Cálculos de distância entre as instâncias e centroides que foram evitados na iteração 2.	71
7.13.	Matriz de distância d da iteração 1 do algoritmo k-Means-Elkan.	78
7.14.	Vetor contendo o valor de s_j , ou seja, a metade da menor distância entre o centroide c_j e qualquer outro centroide c_k , em que $j \neq k$.	78
7.15.	Cálculos de distância entre as instâncias e centroides que foram evitados na iteração 1.	79
7.16.	Vetor contendo o valor de δ_j , $j = 1, 2, 3$ ou seja, o deslocamento dos centroides da sua posição inicial para a atual.	79
7.17.	Valores do limite superior $u(x_i)$ ao final da primeira iteração.	80
7.18.	Valores dos limites inferiores $l_{im}(x_i, c_j)$ ao final da primeira iteração 1.	80

7.19.	Matriz de distância d da iteração 2 do algoritmo k-Means-Elkan.	80
7.20	Vetor contendo o valor de s_j , ou seja, a metade da menor distância entre o centroide c_j e qualquer outro centroide c_k , em que $j \neq k$.	80
7.21.	Cálculos de distância entre as instâncias e centroides que foram evitados na iteração.	81
7.22.	Conjunto C contendo 9 centroides gerados artificialmente, descritos por dois atributos numéricos.	82
7.23.	Conjunto CI contendo 20 instâncias geradas artificialmente, descritas por dois atributos numéricos.	87
7.24.	Valores do limite superior após a execução da primeira iteração.	89
7.25.	Valores dos limites inferiores após a execução da primeira iteração.	89
7.26.	Valores do limite superior após atualização no início da execução da segunda iteração.	90
7.27.	Valores dos limites inferiores após atualização no início da execução da segunda iteração.	90
7.28.	Valores do limite superior após a execução da iteração 3.	92
7.29.	Valores dos limites inferiores após a execução da iteração 3.	92
7.30.	Valores dos limites superiores dos grupos de instâncias após a execução da primeira.	103
7.31.	Valores dos limites inferiores dos grupos de instâncias após a execução da primeira iteração.	103
7.32.	Valores dos limites superiores dos grupos de instâncias após a iteração 2.	105
7.33.	Valores dos limites inferiores dos grupos de instâncias após a iteração 2.	106
7.34.	Valores dos limites inferiores dos subgrupos de instâncias pertencentes ao grupo G_3 após a iteração 2.	106
8.1.	Características dos 24 conjuntos de dados sintéticos criados, em que #Id é o identificador do conjunto, N é o número de instâncias de #Id e M é o número de atributos que descrevem as instâncias de #Id.	117
8.2.	Conjuntos de instâncias de dados reais, extraídos do UCI Repository, em que #Id: identificação do conjunto, N : número de instâncias em #Id e M : número de atributos que descrevem as instâncias em #Id, incluso classes.	118
8.3.	Conjuntos de instâncias de dados reais, extraídos do UCI Repository e submetidos ao processo de Pré-processamento, em que #Id: identificação do conjunto, N : número de instâncias em #Id e M : número de atributos que descrevem as instâncias em #Id.	126
8.4.	Resultado médio de 10 execuções de cada uma das estratégias de aceleração do k-Means nos conjuntos de instâncias reais, em que K é o número de grupos induzidos, W é o número médio de iterações, Calc. Dist. é a média do número de cálculos de distância realizados pelo k-Means padrão, Tempo é o tempo médio das execuções do k-Means padrão, % Calc é a porcentagem do número de cálculos realizada pelo k-Means acelerado em relação ao k-Means padrão e Speed (speed-up) é o tempo médio de execução do k-Means padrão dividido pelo tempo médio de execução do k-Means acelerado.	130
8.5.	Resultado médio de 10 execuções de cada uma das estratégias de aceleração do k-Means nos conjuntos de instâncias sintéticas, em que K é o número de grupos	131

induzidos, W é o número médio de iterações, Calc. Dist. é a média do número de cálculos de distância realizados pelo k-Means padrão, Tempo é o tempo médio das execuções do k-Means padrão, % Calc é a porcentagem do número de cálculos realizada pelo k-Means acelerado em relação ao k-Means padrão e Speed (speed-up) é o tempo médio de execução do k-Means padrão dividido pelo tempo médio de execução do k-Means acelerado.

8.6 Resultado médio de 10 execuções de cada uma das estratégias de aceleração do k-Means nos conjuntos de instâncias sintéticas, em que K é o número de grupos induzidos, W é o número médio de iterações, Calc. Dist. é a média do número de cálculos de distância realizados pelo k-Means padrão, Tempo é o tempo médio das execuções do k-Means padrão, % Calc é a porcentagem do número de cálculos realizada pelo k-Means acelerado em relação ao k-Means padrão e Speed (speed-up) é o tempo médio de execução do k-Means padrão dividido pelo tempo médio de execução do k-Means acelerado. 132

Lista de Figuras

2.1.	Esquema do processo de validação cruzada 5-fold ($K=5$), de forma que o conjunto de instâncias é dividido em 5 subconjuntos ou 5-fold, onde $K-1$ subconjuntos são usados como entrada para indução do classificador e o subconjunto restante é usado para avaliar a acurácia do classificador. Esse processo é repetido K vezes, de forma que cada subconjunto seja considerado como conjunto de teste uma vez.	14
3.1.	Exemplo de uma partição $G=\{G_1, G_2, G_3\}$, em que $ G_1 = 20$, $ G_2 = 12$ e $ G_3 = 8$, sendo $G_1 = \{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20\}$, $G_2 = \{21,23,24,25,27,28,30,33,34,35,36,40\}$ e $G_3 = \{22,26,29,31,32,37,38,39\}$.	18
3.2.	Exemplo de uma partição $G'=\{G_1, G_2\}$, em que $ G_1 = 20$ e $ G_2 = 20$, sendo $G_1 = \{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20\}$ e $G_2 = \{21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40\}$.	19
3.3.	Representação da distância euclidiana entre duas instâncias.	22
3.4.	Triângulo superior da matriz de similaridade gerada a partir agrupamento apresentado na Tabela 3.2.	25
3.5.	Menor distância entre duas instâncias que pertencem a grupos distintos, sendo destacado em verde a menor distância entre os grupos.	27
3.6.	Diâmetro dos grupos, sendo a maior distância ilustrada em cor verde.	28
4.1.	Representação de duas execuções do algoritmo k-Means, para $K=3$, indicadas por 1 e 2 na figura. Na execução 1, (1a) e (1b) indicam, respectivamente, a fase de inicialização randômica dos centroides e o agrupamento final obtido. Na execução 2, (2a) e (2b) indicam a fase de inicialização randômica dos centroides e o agrupamento final obtido.	39
4.2.	Representação das 40 instâncias do conjunto C da Tabela 3.1.	41
4.3.	Representação dos 3 centroides iniciais escolhidos, identificados como estrelas na figura.	42
4.4.	Representação da atribuição das instâncias aos centroides que lhes forem mais próximo.	42
4.5.	Representação da segunda iteração do algoritmo k-Means.	43
4.6.	Iteração final, melhor SQE mínimo local.	43
6.1.	Três situações que evidenciam a desigualdade triangular. Particularmente, em (III) a desigualdade se torna uma igualdade, considerando que o triângulo é um triângulo degenerado (<i>i.e.</i> , seus três vértices são colineares).	54
6.2.	Diagrama exibindo dois centroides (b e c) e uma instância de dado x em que $d(x,b)$ e $d(x,c)$ são as distâncias da instância x a cada um dos centroides e a linha pontilhada marca a distância entre os dois centroides.	55
6.3.	Diagrama exibindo dois centroides, $c = (5,8)$ e $c^* = (13,5)$ e uma instância de dado x que pertence ao grupo cujo centroide é $c = (5,8)$. Como $d(c,c^*) \geq 2d(x,c)$ <i>i.e.</i> , $8,544 \geq 2 \times 3,162 = 6,324$, é desnecessário calcular $d(x,c^*)$, uma vez que $d(x,c^*)$, garantidamente (com base no Lema 1), será maior que $d(x,c)$.	56
7.1.	Situação em que o uso da desigualdade triangular ajuda a diminuir o número de cálculos (para realocação de instâncias aos novos centroides obtidos) a serem realizados.	61

7.2.	Representação das 10 instâncias do conjunto C apresentadas na Tabela 7.1.	62
7.3.	Representação dos 3 centroides iniciais $c_1 = (2,5)$, $c_2 = (6,3)$ e $c_3 = (9,8)$ representados como estrelas azul, verde e vermelho, respectivamente. As instâncias estão atribuídas ao grupo G_1 , ilustrado como pontos de cor azul.	62
7.4.	Representação do agrupamento formado após a iteração 1 do algoritmo compare-Means, $G_1=\{1,2,10\}$, $G_2=\{3,4,5\}$ e $G_3=\{6,7,8,9\}$.	64
7.5.	Representação o agrupamento final gerado pelo algoritmo compare-Means, $G_1=\{1,2,10\}$, $G_2=\{3,4,5\}$ e $G_3=\{6,7,8,9\}$.	65
7.6.	Diagrama exibindo quatro centroides, $c_1 = (4,2)$, $c_2 = (6,7)$, $c_3 = (12,9)$ e $c_4 = (13,3)$ e as respectivas distâncias entre os centroides.	66
7.7.	Representação dos 3 centroides iniciais $c_1 = (2,5)$, $c_2 = (6,3)$ e $c_3 = (9,8)$ representados como estrelas (azul, verde e vermelho, respectivamente). As instâncias foram atribuídas ao grupo G_1 , ilustrado como pontos de cor azul.	68
7.8.	Representação o agrupamento formado após a iteração 1 do algoritmo sort-Means, $G_1=\{1,2,10\}$, $G_2=\{3,4,5\}$ e $G_3=\{6,7,8,9\}$.	70
7.9.	Representação o agrupamento final gerado pelo algoritmo compare-Means, $G_1=\{1,2,10\}$, $G_2=\{3,4,5\}$ e $G_3=\{6,7,8,9\}$.	71
7.10.	Ilustração do deslocamento do centroide c da iteração W-1 para a atual W.	74
7.11.	Representação dos 3 centroides iniciais $c_1=(2,5)$, $c_2=(6,3)$ e $c_3=(9,8)$ representados como estrelas (azul, verde e vermelho, respectivamente). As instâncias foram atribuídas ao grupo G_1 , ilustrado como pontos de cor azul.	77
7.12.	Representação o agrupamento formado após a iteração 1 do algoritmo k-Means-Elkan, $G_1=\{1,2,10\}$, $G_2=\{3,4,5\}$ e $G_3=\{6,7,8,9\}$.	79
7.13.	Representação o agrupamento final gerado pelo algoritmo k-Means-Elkan, $G_1=\{1,2,10\}$, $G_2=\{3,4,5\}$ e $G_3=\{6,7,8,9\}$.	81
7.14.	Diagrama exibindo um conjunto de centroides iniciais $C=\{c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9\}$ separados em três grupos $GC=\{GC_1, GC_2, GC_3\}$ em que $GC_1=\{c_1, c_2\}$, $GC_2=\{c_3, c_4, c_5\}$ e $GC_3=\{c_6, c_7, c_8, c_9\}$.	83
7.15.	Diagrama exemplificando os valores do limite superior e limites inferiores de x, onde $u(x)=1$ e $l_{im}(x, GC_1) = 2,82$, $l_{im}(x, GC_2) = 2,23$ e $l_{im}(x, GC_3) = 5$.	84
7.16.	Representação das 20 instâncias do conjunto CI apresentadas na Tabela 7.23.	88
7.17.	Conjunto CI com 20 instâncias e os 5 centroides iniciais identificados por estrelas <i>i.e.</i> , $c_1 = I_6$, $c_2 = I_4$, $c_3 = I_8$, $c_4 = I_{14}$, $c_5 = I_{10}$.	88
7.18.	Representação dos 2 grupos de centroides, $GC_1=\{I_4, I_{10}\}$ em verde e $GC_2=\{I_6, I_8, I_{14}\}$ em azul, gerados a partir dos 5 centroides iniciais.	89
7.19.	Agrupamento formado após a primeira iteração do algoritmo k-Means-Yinyang.	90
7.20.	Agrupamento formado após a segunda iteração do algoritmo k-Means-Yinyang.	92
7.21.	Agrupamento final induzido pelo algoritmo k-Means-Yinyang.	93
7.22.	Diagrama ilustrando o limite superior e os limites inferiores do grupo G_1 . O limite superior $u(G_1)$ é dado pela maior dentre as distâncias entre o centroide c_1 e as instâncias pertencente ao grupo G_1 . Os limites inferiores $l_{im}(G_1, c_j)$ são dados pela menor dentre as distâncias entre instâncias pertencente a G_1 e o centroide c_j , em que $j \neq 1$.	96
7.23.	Diagrama ilustrando o processo de fissão dos grupos em subgrupos. (a)	97

	agrupamento gerado ao final da primeira iteração. (b) duas instâncias que pertenciam a G_1 migraram para G_3 , dando origem ao subgrupo g_{31} , contendo as instâncias originais do grupo G_3 e o subgrupo g_{32} com as instâncias que migraram do grupo G_1 . (c) uma instância que pertencia a G_1 migrou para G_3 , dando origem ao subgrupo g_{33} . (d) uma instância que pertencia a G_2 migrou para G_1 , dando origem ao subgrupo g_{11} com as instâncias originais do grupo G_1 e ao g_{12} com a instância que migrou do grupo G_2 .	
7.24.	Diagrama ilustrando o limite superior e os limites inferiores do subgrupo g_{32} .	98
7.25.	As 3 instâncias do conjunto CI que foram escolhidas como centroides iniciais $c_1 = I_5$, $c_2 = I_{15}$, $c_3 = I_8$ são identificadas por estrelas.	102
7.26.	Diagrama ilustrando a primeira iteração do algoritmo k-Means-Fission-Fusion.	103
7.27.	Diagrama ilustrando o agrupamento formado ao final da primeira iteração do algoritmo k-Means-Fission-Fusion.	104
7.28.	Diagrama ilustrando o agrupamento formado ao final da primeira iteração do algoritmo k-Means-Fission-Fusion.	105
7.29.	Diagrama ilustrando o agrupamento formado ao final da iteração 3 do algoritmo k-Means-Fission-Fusion.	107
7.30.	Diagrama ilustrando o agrupamento formado ao final da iteração 4 do algoritmo k-Means-Fission-Fusion.	107
7.31.	Diagrama ilustrando o agrupamento formado ao final da iteração 4 do algoritmo k-Means-Fission-Fusion.	108
8.1.	Imagem do módulo de leitura após carregamento de arquivo do conjunto de instâncias.	112
8.2.	Imagem do módulo de pré-processamento após carregamento de arquivo, contendo as marcações sobre as funções disponíveis no módulo.	113
8.3.	Imagem do módulo de execução, contendo as marcações sobre as funções disponíveis no módulo.	114
8.4.	Gráfico gerado pelo módulo de execução do <i>5ACCK-Means</i> , considerando que o conjunto de dados carregado é o descrito em Tabela 7.1.	115
8.5.	Tela do módulo “Índices de Validação” do <i>5ACCK-Means</i> .	115
8.6.	Fluxograma alto-nível da metodologia empregada nos experimentos.	125
8.7.	Descrição do procedimento adotado na condução dos experimentos realizados.	128

Lista de Algoritmos

4.1.	Pseudocódigo simplificado do algoritmo k-Means.	40
4.2.	Pseudocódigo do procedimento de inicialização do k-Means++, adaptado de [Oliveira 2018].	45
7.1.	Pseudocódigo do k-Means implementando a estratégia compare-Means, proposto em [Phillip 2002].	60
7.2.	Pseudocódigo do k-Means implementando a estratégia sort-Means, proposta em [Phillip 2002].	67
7.3.	Pseudocódigo do k-Means implementando a estratégia proposta por Elkan, adaptado de [Hamerly & Drake 2015].	76
7.4.	Parte 1: pseudocódigo do algoritmo k-Means-Yinyang, adaptado de [Bottesch <i>et al.</i> 2016].	86
	Parte 2: pseudocódigo do algoritmo k-Means-Yinyang, adaptado de [Bottesch <i>et al.</i> 2016].	87
7.5.	Parte 1: pseudocódigo do algoritmo k-Means-Fission-Fusion, adaptado de [Yu & Dai 2017].	100
	Parte 2: pseudocódigo do algoritmo k-Means-Fission-Fusion, adaptado de [Yu & Dai 2017].	101

Capítulo 1

Apresentação

A pesquisa em nível de mestrado, descrita nesta Dissertação, aborda uma investigação sobre estratégias de aceleração do algoritmo de agrupamento conhecido como k-Means [MacQueen 1967]. A pesquisa está fundamentada na área de Aprendizado de Máquina (AM), uma subárea da Inteligência Artificial (IA) e tem como objetivo investigar propostas de otimização de desempenho do algoritmo k-Means utilizando a chamada desigualdade triangular. É importante lembrar que o algoritmo k-Means tem sido objeto de pesquisas e de tentativas de melhoria de seu desempenho desde que foi criado, como lembrado em [Jain 2010].

O algoritmo k-Means é caracterizado por ser um algoritmo *não-supervisionado*, isto é, um algoritmo que não requer informações das classes associadas às instâncias a serem agrupadas. O algoritmo k-Means é um algoritmo de agrupamento bastante popular em várias áreas de conhecimento e tem sido usado em um vasto número de aplicações computacionais. Algoritmos de agrupamento buscam descobrir grupos naturais em um conjunto de instâncias, por meio de técnicas estatísticas, realizando comparações de múltiplos atributos. O algoritmo k-Means é caracterizado como um algoritmo particional, isto é, o agrupamento (*clustering*) induzido pelo algoritmo é uma partição do conjunto original. Considerando a definição de partição, o agrupamento induzido pode ser descrito como um conjunto de subconjuntos do conjunto original de instâncias, cada um deles (nomeado de grupo ou *cluster*) não vazio, sendo que tais subconjuntos são dois-a-dois disjuntos e a união de todos eles resulta no conjunto original.

Abordado de uma maneira simplista, o algoritmo k-Means pode ser descrito como um processo que, dado um conjunto com N instâncias, representadas como um vetor de valores de atributos, busca particionar o conjunto dado em K grupos, em que K é um parâmetro geralmente fornecido pelo usuário. Cada instância é atribuída ao grupo cujo

centroide (geralmente definido como a média das instâncias do grupo) lhe for mais próximo.

Para atribuir uma instância ao grupo cujo centroide lhe for mais próximo é necessário que o algoritmo calcule a distância entre a instância e todos os K centroides, identifique o centroide mais próximo e, então, atribua a instância ao grupo associado a tal centroide. Devido a essa característica, o algoritmo k-Means investe a maior parte de seu tempo de execução em cálculos de distância. Porém, muitos desses cálculos podem ser desnecessários e podem ser evitados utilizando técnicas adequadas baseadas em geometria [Hamerly & Drake 2015].

Ao evitar cálculos de distâncias desnecessários, o tempo de execução do algoritmo tende a diminuir, provocando a sua aceleração. As estratégias de aceleração agregadas ao k-Means, objeto de investigação desta pesquisa, produzem, em geral, os mesmos resultados gerados pelo k-Means original, para as mesmas entradas e parâmetros, ou seja: o mesmo conjunto de instâncias, o mesmo número de grupos no agrupamento a ser induzido e o mesmo grupo de centroides iniciais. Tais estratégias, entretanto, consomem um tempo de execução inferior ao tempo de execução-consumido pelo k-Means padrão.

As estratégias de aceleração escolhidas para estudo e investigação foram: compare-Means e sort-Means, propostas por Phillips, em [Phillips 2002], a versão nomeada nesta Dissertação de k-Means-Elkan, proposta por Elkan, em [Elkan 2003], k-Means-Yinyang proposta por Ding e coautores, em [Ding *et al.* 2015] e, finalmente, k-Means-Fission-Fusion, proposta por Yu e Dai e descrita em [Yu & Dai 2017]. Cada uma dessas estratégias usa de forma distinta a desigualdade triangular, permitindo, assim, evitar cálculos de distâncias entre instâncias e centroides desnecessários.

Com o objetivo de introduzir o contexto formal em que a pesquisa se insere, esta Dissertação está organizada em mais oito capítulos, cujos conteúdos são brevemente descritos a seguir, por ordem de aparecimento no texto.

Capítulo 2: Introduz a área de Aprendizado de Máquina (AM), suas principais características e conceitos associados. O capítulo caracteriza conjunto de treinamento e conjunto de teste, além de abordar os conceitos de pré-processamento de instâncias e a técnica de validação cruzada.

Capítulo 3: Inicialmente contextualiza os algoritmos conhecidos como algoritmos de agrupamentos, formaliza as notações a serem utilizadas ao longo do texto e apresenta as definições básicas necessárias ao entendimento do texto da Dissertação. Apresenta uma das possíveis taxonomias de algoritmos de agrupamento e comenta brevemente sobre medidas de similaridade. O capítulo é finalizado com a apresentação e descrição dos chamados índices de validação de agrupamentos, que são utilizados para avaliar a qualidade do agrupamento induzido por um algoritmo.

Capítulo 4: Aborda o algoritmo k-Means, seu funcionamento e seus principais conceitos e apresenta um exemplo didático de uso do k-Means. O capítulo termina com a apresentação de um algoritmo que introduz uma melhoria no processo de inicialização do k-Means, que é conhecido como k-Means++ [Arthur & Vassilvitskii 2007]. O k-Means++ é, muitas vezes, o algoritmo escolhido para ser empregado em aplicações, uma vez que tende a apresentar melhores resultados, quando comparados a resultados obtidos pelo k-Means padrão; tal característica se deve à inicialização diferenciada que implementa.

Capítulo 5: Apresenta uma revisão da literatura com foco em estratégias para acelerar o algoritmo k-Means, enfatizando os principais aspectos que eventualmente contribuem para sua aceleração.

Capítulo 6: Discute, de forma teórica, o uso da desigualdade triangular, como maneira de acelerar o algoritmo k-Means. O uso da desigualdade triangular permite evitar cálculos de distâncias entre instâncias e centroides que são considerados desnecessários. O capítulo também apresenta exemplos de como a desigualdade triangular pode ser usada.

Capítulo 7: Tem por foco a apresentação detalhada das cinco estratégias para acelerar a execução do algoritmo k-Means, expõe a teoria que subsidia o funcionamento de cada uma delas. Além de apresentar um exemplo didático sobre cada estratégia.

Capítulo 8: Inicialmente o capítulo descreve, em linhas gerais, a arquitetura e as diferentes funcionalidades disponibilizadas pelo sistema computacional *5Accelerated k-Means* (*5ACCK-Means*). Tal sistema foi desenvolvido durante o desenvolvimento do Projeto de Pesquisa que subsidiou essa Dissertação. O *5ACCK-Means* está

disponibilizado para uso em experimentos com as estratégias de aceleração do algoritmo k-Means. O *5ACCK-Means* disponibiliza várias funcionalidades que dão suporte ao uso dos algoritmos implementados, incluindo: leitura das instâncias de dado a serem agrupadas, pré-processamento das instâncias, indução dos agrupamentos, visualização dos grupos induzidos e geração dos valores referentes a alguns índices de validação interna e externa. Na sequência, o capítulo apresenta e descreve os conjuntos de instâncias utilizados nos experimentos conduzidos, a metodologia criada para a execução dos experimentos e, também, discute os resultados de cada uma das estratégias investigadas. O capítulo finaliza com a apresentação dos resultados obtidos e uma análise comparativa do comportamento de cada uma das estratégias consideradas, subsidiada pelos resultados obtidos.

Capítulo 9: Resume os principais pontos levantados na pesquisa realizada bem como as conclusões derivadas dos experimentos conduzidos. Apresenta também possíveis linhas de pesquisa para a continuidade do trabalho realizado.

Capítulo 2

CONSIDERAÇÕES E PRINCIPAIS ASPECTOS RELACIONADOS À ÁREA DE APRENDIZADO DE MÁQUINA

2.1 Considerações Iniciais

O processo de aprendizado realizado por seres humanos é uma área de investigação sobre a qual a ciência tem se debruçado por um longo período, porém, ainda sem conclusões definitivas. E mesmo ainda sem entender completamente esse processo, mas vislumbrando sua potencialidade, há várias décadas surgiu o interesse em fomentar o desenvolvimento de um processo de aprendizado automático em computadores e/ou máquinas.

Apesar da dificuldade de definir exatamente o conceito de Inteligência Artificial (IA), pode ser verificado junto à literatura associada que IA, como área de conhecimento, não se apresenta como uma área confinada e é subsidiada por um vasto conjunto de áreas de conhecimento, entre elas Matemática, Estatística, Psicologia, Fisiologia, Educação, Reconhecimento de Padrões, Processamento de Língua Natural, entre outras. Nos dias de hoje já é possível vislumbrar algumas de suas áreas de aplicação que, via de regra, têm em comum o objetivo de fazer com que computadores aumentem sua performance na realização de determinada tarefa, sem a interferência humana.

A capacidade da máquina de não somente aumentar sua performance em uma tarefa, mas também, aprender durante o processo, é caracterizada como Aprendizado de Máquina (AM). Técnicas e algoritmos considerados como de AM buscam simular o processo de aprendizado, de forma automatizada, em computadores. Em [Michalski *et al.* 1983, pp. 3] os autores comentam, com relação ao AM, que “os processos de aprendizado incluem a aquisição de novos conhecimentos declarativos, o desenvolvimento de habilidades motoras e cognitivas por meio de instrução ou prática, a

organização de novos conhecimentos em representações gerais e eficazes e a descoberta de novos fatos e teorias por meio de observação e experimentação”.

O entendimento da habilidade do computador aprender vem ganhando crescente importância prática. Mitchell, em [Mitchell 1997, pp. 2], oferece uma definição sucinta do que consistiria o processo de aprendizado realizado por um sistema computacional: “Diz-se que um programa de computador aprende com base na experiência E , relativa a algum tipo de tarefa T na qual tem performance P , se sua performance P na realização da tarefa T , melhorar com a sua experiência E ”. Essa busca contínua da melhoria por meio da experiência, que se reflete em muitas das técnicas e algoritmos de AM, influencia diretamente o desenvolvimento dos modernos sistemas de IA.

Prati, em [Prati 2006, pp.14], sintetiza que “o aprendizado está relacionado à correta manipulação de conhecimento prévio e novas observações que possam levar a novos conhecimentos”. Tal manipulação está relacionada ao processo de inferência lógica, e pode ser abordada em três classes: dedução, indução e abdução.

A inferência lógica indutiva é o fundamento por trás do Aprendizado Indutivo de Máquina (AIM), que busca a partir de um conjunto de instâncias específicas, generalizar uma *expressão de conceito* que consiga caracterizar novas instâncias ou agrupá-las de forma mais homogênea possível, entre outros. Um tratamento formal do processo de inferência lógica indutiva pode ser encontrado em [Angluin 1980].

2.2 Aprendizado Indutivo de Máquina

Um grupo de algoritmos que vem se destacando na área de AM é formado por aqueles algoritmos comumente identificados como algoritmos de Aprendizado Indutivo de Máquina (AIM), que possuem como característica principal um processo de generalização, a partir de um conjunto de instâncias.

Aprendizado Indutivo [Michalski *et al.* 1983] pode ser entendido como uma heurística definida em um espaço de descrições simbólicas, geradas por meio da utilização de regras de inferência (por exemplo a generalização) sobre um conjunto de instâncias, em que um conjunto de instâncias C pode ser definido como $C = \{x_i\}_{i=1}^N$ em que cada x_i ($1 \leq i \leq N$) é um vetor M -dimensional de valores de atributos que descrevem os dados.

O conjunto de instâncias fornecido como entrada a um algoritmo indutivo de AM é usualmente chamado de *conjunto de treinamento*. A representação das instâncias do conjunto de treinamento, na dependência de ter (ou não) uma informação associada, identificada como *classe*, de certa forma colabora na identificação do tipo do algoritmo de AM adequado para um determinado conjunto de instâncias.

Algoritmos que fazem uso da informação fornecida pelas classes associadas às instâncias de dados são geralmente chamados de *algoritmos supervisionados*. Algoritmos que não utilizam a informação da classe ou, então, que foram propostos com o intuito de apenas serem agrupadores de subconjuntos de instâncias do conjunto original de instâncias fornecidos, são caracterizados como *algoritmos não-supervisionados* [Theodoridis & Koutroubas 2009] e, dentre os algoritmos não-supervisionados, os algoritmos de agrupamentos são o foco deste trabalho de pesquisa, que aborda o algoritmo de agrupamento conhecido como k-Means, inicialmente descrito por [MacQueen 1967] e abordado em detalhes no Capítulo 4.

Existem também os algoritmos conhecidos como *semisupervisionados* que, alternativamente, utilizam técnicas de aprendizado supervisionado e não supervisionado para conduzir o processo de aprendizado. O objetivo da pesquisa na área de aprendizado semisupervisionado é o de utilizar informações contidas em instâncias com classes associadas e em instâncias sem classes associadas, e investigar maneiras de como essa combinação de informações pode colaborar com o processo de aprendizado, bem como o de projetar algoritmos eficientes que implementem tal combinação [Chapelle *et al.* 2006] [Zhu & Goldberg 2009].

Em um ambiente de aprendizado supervisionado, o processo indutivo, codificado pela implementação de um algoritmo supervisionado, realiza a generalização das instâncias do conjunto de treinamento, em uma *expressão de conceito*, também conhecida como *representação formal do conceito*. Usualmente linguagens formais são empregadas com o propósito de representar formalmente a expressão.

Nicoletti, em [Nicoletti 1994], aponta dois tipos de linguagem para a representação de expressões formais simbólicas: a linguagem baseada em atributos, considerada uma linguagem baseada em lógica proposicional e a linguagem baseada em lógica de primeira ordem. Algoritmos de AM que utilizam lógica de primeira ordem para a

representação tanto do conjunto de treinamento quanto de expressões induzidas e outras informações relevantes ao aprendizado (*e.g.*, Teoria do Domínio), formam um grupo bastante específico de algoritmos que se caracterizam como algoritmos de Programação Lógica Indutiva. O processo indutivo conduzido pelo algoritmo envolve programação lógica e o resultado do aprendizado é expresso por um programa lógico, representado em linguagens baseadas em Prolog e, daí, tal área de pesquisa ser conhecida como Programação Lógica Indutiva.

Em sistemas que utilizam linguagens baseadas em atributos, o processo de aprendizado pode ser descrito como “dado um conjunto de instâncias de treinamento, representadas como vetores de pares atributos-valor, cujas classes são conhecidas, encontrar uma regra que prediga a classe de uma nova instância (sem classe associada) em função de seus atributos e valores”. Em [Nicoletti 1994] é apresentado um conjunto de características associadas a algoritmos de AM, que podem ser utilizadas para agrupar e/ou distinguir grupos de algoritmos que compartilham alguma(s) dela(s). Três das características são comentadas a seguir:

- *Incremental e Não Incremental*: algoritmos que se adequam à abordagem incremental são chamados de algoritmos incrementais. A incrementabilidade implementada por tais algoritmos consiste em 'corrigir' a expressão de conceito obtida a partir de instâncias anteriores apresentadas ao algoritmo, de maneira a adequá-la à representação, também, da nova instância. Em alguns casos pode ser mais eficiente revisar a expressão do conceito obtida até então, do que reiniciar todo o processo de aprendizado. Já algoritmos considerados não incrementais, exigem que todas as instâncias do conjunto de treinamento estejam disponíveis para que o processo de aprendizado possa ser inicializado.
- *Um Conceito ou Vários Conceitos*: essa característica está relacionada à capacidade de um sistema de AM aprender a representação de apenas um conceito ou de vários conceitos que estão instanciados no conjunto de treinamento.
- *Uso da Teoria do Domínio*: A representação do conhecimento caracterizado como Teoria de Domínio é crucial para viabilizar o aprendizado em domínios de dados que requerem uma representação mais elaborada do conhecimento, do que

simples regras proposicionais, que são utilizadas pela maioria dos algoritmos de AM correntemente utilizados.

2.3 Pré-Processamento de Dados

O pré-processamento de dados é uma fase fundamental no processo de aprendizado automático, pois os dados brutos disponíveis geralmente apresentam imperfeições, inconsistências, valores ausentes, ruídos (*i.e.*, dados incorretos ou inconsistentes) e/ou redundâncias. Como a baixa qualidade dos dados afeta o desempenho do algoritmo de aprendizado, é importante que estratégias que buscam eliminar tais tipos de problemas sejam consideradas em uma fase anterior a do aprendizado automático, fase essa identificada como pré-processamento de dados.

Em [Pyle 1999] é especificado um conjunto de aspectos que devem ser levados em consideração, com vistas ao tratamento inicial do conjunto de instâncias de dados disponibilizado ao aprendizado automático. No que segue são apresentados os aspectos que são relevantes ao trabalho de pesquisa que está sendo conduzido.

- *Representatividade do volume de instâncias disponibilizado*: Um problema comum no pré-processamento de dados é determinar o volume (*i.e.*, o número de instâncias) e a estrutura mais adequada para representar as instâncias disponibilizadas. Maiores detalhes sobre a importância que o volume de dados tem no aprendizado automático, bem como a representação adotada para codificá-los, podem ser encontrados em [Dash & Liu 1997], [Pyle 1999] e [Kotsiantis & Kanellopoulos 2006].
- *Tratamento de instâncias inconsistentes e incompletas*: Instâncias inconsistentes e incompletas tornam-se problemas quando utilizadas em um ambiente de aprendizado automático. Se não tratadas, tais instâncias podem introduzir correlações espúrias entre atributos, introduzir vieses, bem como diminuir a confiança nas instâncias de entrada; uma abordagem geral sobre viés e confiança pode ser encontrada em [Pyle 1999]. Como instâncias do mundo real, via de regra, apresentam inconsistências e muitas delas podem ter descrições incompletas, o tratamento de inconsistência e incompletude deve ser

implementado como parte do pré-processamento das instâncias. Em um ambiente de AM supervisionado, por exemplo, uma situação típica de instâncias inconsistentes pode ser a seguinte: X e Y são duas instâncias de treinamento, em que $X = (A_1 = 3, A_2 = 8, \text{classe}=A)$ e $Y = (A_1 = 3, A_2 = 8, \text{classe}=B)$. Note que X e Y são descritas pelos mesmos valores de atributos e, no entanto, pertencem a classes distintas, o que evidencia uma situação de inconsistência. Uma situação em que instâncias são descritas por 5 atributos, por exemplo, A_1, A_2, A_3, A_4 e A_5 , uma instância cuja descrição é dada por: $X = (A_1 = 3, A_2 = 8, A_3 = c, A_4 = 3.7, A_5 = ?, \text{classe}=A)$, em que ? representa ausência de valor de atributo, é uma instância incompleta.

Instâncias inconsistentes são comumente tratadas por meio de (1) métodos estatísticos (*e.g.*, técnicas baseadas em modelos das distribuições das instâncias), (2) técnicas de AM (*e.g.*, agrupamento, árvore decisão), (3) algoritmos genéticos, entre outros [Hodge & Austin 2004].

Instâncias com valor ausente de atributo podem ser tratadas por meio de (1) descartes, (2) uso do valor mais frequente (moda) do atributo para substituir o valor ausente, (3) uso do valor médio do atributo para substituir o valor ausente, entre outros [Kotsiantis & Kanellopoulos 2006] [Pyle 1999].

- **Normalização:** O objetivo da normalização é minimizar problemas provindos do uso de unidades e dispersões diferentes entre atributos. Os valores dos atributos podem ser normalizados segundo a amplitude ou a distribuição:

(1) Normalização de amplitude é uma técnica que tem como objetivo alterar os valores dos atributos para uma escala padronizada (comum), sem distorcer as diferenças de intervalos dos valores. Isto é, a técnica de normalização de amplitude propõe um remapeamento dos valores originais de entrada para um novo intervalo de saída. Manter os atributos em uma escala padronizada busca evitar que atributos com intervalos grandes (*e.g.*, renda), prevaleça sobre atributos com intervalos menores (*e.g.*, atributos binários); é comum o uso dos intervalos $[0,1]$ ou $[-1,1]$ como valores padrões de saída [Han *et al.* 2012].

(2) Normalização na distribuição consiste em remover distorções geradas por valores extremos e aumentar a previsibilidade linear [Pyle 1999] [Han *et al.* 2012].

- *Transformação de dados*: Seguindo o mesmo princípio da normalização, alguns algoritmos de aprendizado se beneficiam de certos formatos dos atributos (*e.g.*, discreto), dessa forma, pode ser conveniente, em certos casos, realizar a adequação, seguem algumas das possibilidades:

(1) Discretização de atributos quantitativos (contínuo e discreto): ocorre principalmente por meio da criação de faixas de valores. Na literatura existem propostas de algoritmos que realizam o processo de discretização de valores reais, como é o caso do algoritmo MDLP proposto por Fayyad e Irani, em [Fayyad & Irani 1993]. Também é possível encontrar uma abordagem geral em [Kotsiantis & Kanellopoulos 2006] e [Pyle 1999].

(2) Transformação de atributos qualitativos em quantitativos: consiste em transformar os atributos qualitativos (nominais e cardinais) em quantitativos (contínuo e discreto). Existem várias técnicas que implementam tal processo, mas, de forma geral, são atribuídos arbitrariamente valores numéricos a atributos qualitativos que não possuem ordem (*e.g.*, 1 - vermelho, 2 - verde, 3 - azul); caso os valores estejam ordenados, a ordem deve ser mantida após a transformação.

2.4 Conjunto de Treinamento e Teste

Como apresentado em seção anterior, o conjunto de treinamento é um conjunto de instâncias, a partir do qual o processo de aprendizado é realizado. Cada instância é representada por um conjunto de atributos (*i.e.*, vetor de valores associados a atributos) e que pode ter ou não uma classe associada. Um aspecto importante que precisa ser levado em conta em relação ao conjunto de treinamento, é que ele deve ser suficientemente volumoso para gerar resultados estatísticos significativos [Abu-Mostafa *et al.* 2012] [Elkan 2012].

O *conjunto de teste* tem como objetivo avaliar a representatividade da expressão de conceito que foi criada pelo algoritmo de aprendizado, a partir do conjunto de treinamento. Ou seja, avaliar o grau de efetividade do conceito aprendido. Dessa forma,

é possível inferir o comportamento da expressão de conceito em novas instâncias. O conjunto de teste, geralmente, é um conjunto independente do conjunto de treinamento, mas que busca seguir a mesma distribuição estatística.

Uma das técnicas de divisão do conjunto de instâncias é conhecida como *holdout*. Nessa técnica, comumente, se divide em uma proporção de 2/3 das instâncias para o conjunto de treinamento e 1/3 para o conjunto de teste [Kohavi 1995]. Essa abordagem somente é recomendada em conjunto de instâncias volumoso, pois, se o conjunto de instâncias for pequeno pode ocorrer grande variação na acurácia (probabilidade de se classificar corretamente uma instância selecionada de forma aleatória). Para tornar o resultado menos dependente da forma que se realiza a divisão do conjunto de instâncias, pode-se calcular a média de vários resultados, com isso, pode-se obter uma estimativa média do *holdout*.

Há casos em que a expressão de conceito induzida no conjunto de treinamento não apresenta boa avaliação no conjunto de teste, esse problema pode estar vinculado a um super ajustamento da expressão de conceito as instâncias de treinamento, ou seja, existe o risco dos ruídos das instâncias de treinamento sejam memorizados, e essa memorização influencie a indução. Esse problema é conhecido como *overfitting* [Dietterich 1995].

Real, em [Real 2014], esclarece, de forma simplista, que um alto *overfitting* expressa a plasticidade da expressão de conceito ‘se acomodar’ as instâncias de treinamento. Essa ‘acomodação’, de certa forma, torna a expressão de conceito menos eficiente em novas instâncias.

2.5 Validação Cruzada

A validação cruzada é uma técnica que ajuda a evitar ou então, que ajuda a reduzir o risco de *overfitting*, que é comumente utilizada associada a algoritmos de AM supervisionados, principalmente algoritmos de AM que induzem *classificadores*. Os métodos de validação cruzada têm como princípio dividir o conjunto original de instâncias em subconjuntos mutuamente exclusivos, permitindo, assim, estimar a acurácia da expressão de conceito e, por fim, escolher a expressão de conceito que apresenta a melhor avaliação. Kohavi, em [Kohavi 1995] aponta três desses métodos:

- k-fold: Consiste inicialmente em induzir uma partição G no conjunto X de instâncias de dados, em K subconjuntos *i.e.*, $G=\{G_1, G_2, \dots, G_K\}$. Como G é uma partição, (a) $\forall G_i \neq \emptyset, i = 1, \dots, K$; (b) $\cup G_i = X, i = 1, \dots, K$ e (c) $G_i \cap G_j = \emptyset, i, j = 1, \dots, K$ e $i \neq j$.

São então conduzidos K processos de treinamento seguido de teste, da seguinte maneira: dentre os K subconjuntos, $K-1$ deles são utilizados como conjunto de treinamento e o subconjunto que não participa do conjunto de treinamento, é considerado o conjunto de teste e usado para avaliar o classificador induzido pelas instâncias dos $K-1$ conjuntos. Cada um dos K conjuntos da partição participa $K-1$ vezes do conjunto de treinamento e 1 vez como conjunto de teste. A expressão induzida do conceito que apresentar melhor desempenho médio nos K conjuntos de teste é escolhida como o classificador [Arlot & Celisse 2010] [Abu-Mostafa *et al.* 2012] [Elkan 2012].

Elkan, em [Elkan 2012] afirma que em pesquisas recentes o valor mais comum escolhido para K é 10, e a complexidade de tempo de execução dos algoritmos que utilizam da técnica de validação cruzada k-fold é, em geral, K vezes maior que a do algoritmo que não utiliza.

A Figura 2.1, baseada em uma figura apresentada em [Nietto 2016], ilustra um classificador que utiliza a técnica k-fold, de forma que o conjunto de instâncias de dados seja dividido em $K=5$ subconjuntos com aproximadamente o mesmo tamanho (5-fold), onde 4 subconjuntos são utilizados como conjunto de treinamento (ilustrado com a cor verde) e o subconjunto restante como conjunto de teste (ilustrado com a cor amarela). Dessa forma, os subconjuntos definidos como conjunto de treinamento são utilizados como entrada do algoritmo de AM para indução do classificador e o subconjunto definido como conjunto de teste é utilizado para estimar os erros do classificador, isto é, sua acurácia. Esse processo é repetido K vezes de forma que cada subconjunto seja considerado como conjunto de teste uma vez. Por fim, é calculada a média da acurácia obtida no classificador nos K subconjuntos de testes.

- k-fold estratificado: Segue o mesmo princípio do k-fold, sendo considerado um complemento do mesmo. O k-fold estratificado busca manter a proporção de classes em cada fold (subconjunto) na mesma proporção do conjunto original de instâncias (*e.g.*, se no conjunto original 20% das instâncias estão associadas a classe ‘A’, em cada fold (subconjunto) deverá conter, aproximadamente, 20% de instâncias associadas a classe ‘A’).
- Leave-one-out: O leave-one-out é considerado uma variação do k-fold e é geralmente utilizado em conjunto de instâncias pequeno, pois é computacionalmente dispendioso. Em um conjunto com N instâncias, N–1 são utilizados para treinamento da expressão de conceito, sendo que a validação ocorre em apenas uma instância, isto é, aquela que não participou do conjunto de treinamento. Esse processo é repetido N vezes e a acurácia final é a média das acurácias calculadas em cada uma das N repetições.

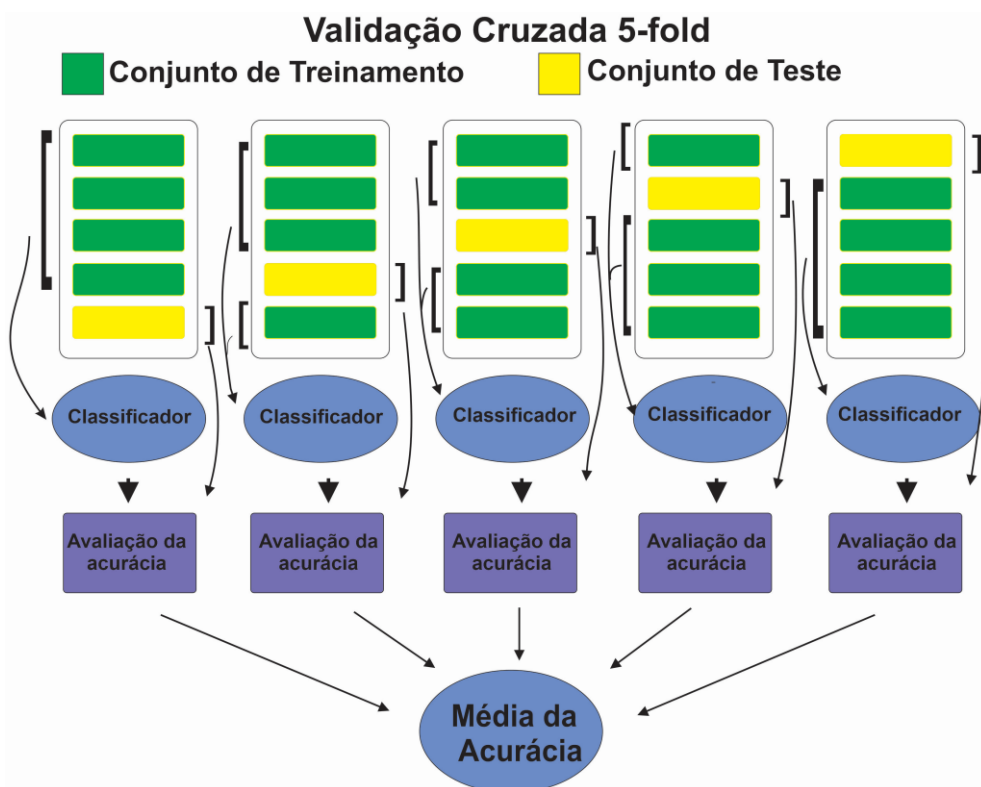


Figura 2.1 Esquema do processo de validação cruzada 5-fold ($K=5$), de forma que o conjunto de instâncias é dividido em 5 subconjuntos ou 5-fold, onde $K-1$ subconjuntos são usados como entrada para indução do classificador e o subconjunto restante é usado para avaliar a acurácia do classificador. Esse processo é repetido K vezes, de forma que cada subconjunto seja considerado como conjunto de teste uma vez.

Capítulo 3

ALGORITMOS DE AGRUPAMENTOS

3.1 Considerações Iniciais

Como brevemente apresentado no Capítulo 2, algoritmos de AM que não utilizam a informação da classe associada às instâncias ou, então, que foram propostos com o intuito de apenas serem agrupadores de subconjuntos de instâncias do conjunto original de instâncias fornecidas, são caracterizados como *algoritmos não-supervisionados* [Theodoridis & Koutroubas 2009]. Dentre os algoritmos não-supervisionados os chamados algoritmos de agrupamentos são os mais populares e um deles, o k-Means, abordado no Capítulo 4, é o foco deste trabalho de pesquisa.

O uso de algoritmos de agrupamento pode ser conveniente em várias aplicações do mundo real, particularmente naquelas que envolvem análise exploratória de relações em conjuntos de instâncias de dados, tomada de decisões, aprendizado de máquina, mineração de dados, recuperação de documentos, segmentação de imagens, etc. Agrupamentos são particularmente apropriados para exploração de inter-relações entre instâncias de dados que possuem poucas informações prévias e, basicamente, se fundamentam no uso apenas da informação contida nas descrições das instâncias a serem agrupadas em grupos; o conjunto de tais grupos é referenciado como um agrupamento [Jain 2010] [Jain *et. al.*, 1999].

Os algoritmos de agrupamentos buscam descobrir grupos naturais em um conjunto de instâncias, por meio de técnicas/medidas estatísticas, fazendo comparações quantitativas de múltiplos atributos. Jain, em [Jain 2010, pp. 652], oferece a seguinte definição operacional de agrupamento: "dado um conjunto contendo N instâncias de dados, encontrar K grupos de instâncias, tendo por base uma medida de similaridade entre instâncias, de maneira que instâncias que pertençam ao mesmo grupo sejam semelhantes, enquanto que instâncias que pertencem a grupos distintos não sejam semelhantes."

Uma das principais dificuldades para análise de agrupamento é determinar o número correto de grupos, pois raramente o número de grupos é conhecido na prática. Com isso, a maioria dos algoritmos de agrupamento é projetado para criar K grupos considerando um valor K pré-definido, que normalmente é determinado por um especialista do domínio das instâncias. Existem na literatura múltiplas propostas para determinar automaticamente o número de grupos em um conjunto de dados. Kodinariya e Makwana, em [Kodinariya & Makwana 2013], trazem detalhes sobre várias delas. No processo de busca por resultados satisfatórios na criação de agrupamentos Theodoridis e Koutroubas, em [Theodoridis & Koutroubas 2009], sugerem os seguintes passos:

- *Seleção de atributos*: considerando as instâncias (vetores de valores de atributos) disponibilizadas para o aprendizado automático, deve-se buscar selecionar atributos relevantes para a caracterização dos grupos, de forma a melhor representar o conjunto, buscando a menor redundância de informações. Se necessário, submeter o conjunto de instâncias a um pré-processamento.
- *Medida de Similaridade*: é uma medida que busca quantificar a similaridade ou dissimilaridade entre instâncias (*e.g.*, distância euclidiana).
- *CrITÉRIOS de Agrupamento*: dependendo do tipo de agrupamento que se pretende criar, os critérios de agrupamento podem variar, e essa variação depende da interpretação do especialista do domínio sobre o tipo de grupo que melhor modela o conjunto de instâncias fornecido. Dependendo dos critérios utilizados é possível gerar agrupamentos com características distintas, como por exemplo: variando o número de atributos que descrevem as instâncias pode-se obter agrupamentos como função de apenas determinados atributos, por exemplo. Também, pode-se adotar uma abordagem particional, com a indução de um único agrupamento ou, então, uma abordagem hierárquica, por meio da qual são obtidos vários agrupamentos ao longo de um processo que implementa a indução de uma hierarquia de agrupamentos.
- *Algoritmos de Agrupamento*: Essa etapa se refere à escolha de um algoritmo específico que melhor evidencie a estrutura dos grupos de instâncias existentes no conjunto original de instâncias disponibilizado.

- *Validação dos Resultados:* Uma vez executado o algoritmo de agrupamento tendo como *input* o conjunto de instâncias a serem agrupadas, é importante que o agrupamento induzido seja avaliado. A Seção 3.5 apresenta em detalhes quatro índices de validação utilizados na literatura. Dois deles são caracterizados como índices de validação interna e dois outros, são caracterizados como índices de validação externa.
- *Interpretação dos Resultados:* Em muitos casos é necessário que o especialista do domínio integre os resultados com outras evidências experimentais, para que, assim, seja possível chegar à uma maior compreensão das estruturas induzidas pelo algoritmo de agrupamento.

Diferentes seleções de atributos, medidas de similaridades, critérios de agrupamentos e algoritmos de agrupamento podem levar à indução de agrupamentos totalmente diferentes entre si. Outro fato importante que precisa ser analisado, é que não necessariamente os dados possuem uma estrutura de grupo, as instâncias podem ser de natureza completamente aleatória e tentar compreendê-las por meio do uso de algoritmos de agrupamento não é uma escolha recomendada.

3.2 Notações e Definições Básicas

Considere um conjunto de instâncias de dados C , contendo N instâncias, especificado e notado como $C = \{x_1, x_2, \dots, x_N\}$, em que cada instância, x_i , $1 \leq i \leq N$ é descrita por M atributos, $A_1, A_2, \dots, A_M$. Define-se como um *K-agrupamento* (*K-clustering*) de C uma partição de C em K conjuntos de instâncias (grupos), notadas por $G_1, G_2, \dots, G_K$. O K -agrupamento de C é então notado como: $G = \{G_1, G_2, \dots, G_K\}$ ou seja, é um conjunto de conjuntos *i.e.*, é um conjunto de K grupos (*clusters*).

Dado que um K -agrupamento de C é definido como uma partição de C em K grupos, as três condições a seguir devem ser satisfeitas, uma vez que as três condições caracterizam o que uma partição de um dado conjunto é, *i.e.*:

- (1) $G_i \neq \emptyset$, $i = 1, \dots, K$ (cada um dos grupos do agrupamento é não-vazio)
- (2) $\cup G_i = C$, $i = 1, \dots, K$ (ou seja, a união de todos os grupos que pertencem ao agrupamento resulta no conjunto C)

- (3) $G_i \cap G_j = \emptyset$, $i \neq j$ e $i, j = 1, \dots, K$ (ou seja, os grupos que pertencem ao agrupamento são dois-a-dois disjuntos)

Um atributo registra um valor ou uma expressão que representa uma informação e pode assumir valores quantitativos, qualitativos e atributos estruturados [Gowda & Diday 1992]:

- (1) Atributos quantitativos podem assumir intervalos numéricos contínuos (*e.g.*, tamanho, velocidade, peso) ou um conjunto finito discreto (*e.g.* {0,1,2});
- (2) Atributos qualitativos podem assumir valores nominais, que usualmente não apresentam uma ordem estabelecida (*e.g.*, vermelho, azul, verde); ou ordinais, que possuem uma ordem pré-definida (*e.g.*, pequeno, médio, grande); e também a combinação de nominais e ordinais (*e.g.*, Rua 1, Rua 2);
- (3) Atributos estruturados podem ser entendidos como dados complexos que armazenam uma estrutura informacional em um único atributo (*e.g.*, uma árvore de decisão, código fonte de um algoritmo).

A Tabela 3.1 apresenta o conjunto C, com 40 instâncias artificialmente geradas, em que cada instância x_i , $1 \leq i \leq 40$, é descrita por dois atributos numéricos, A_1 e A_2 . A Figura 3.1 ilustra em um gráfico uma das possíveis partições G, a partir do conjunto C de instâncias descritas na Tabela 3.1, com três subconjuntos (grupos), G_1 , G_2 , G_3 .

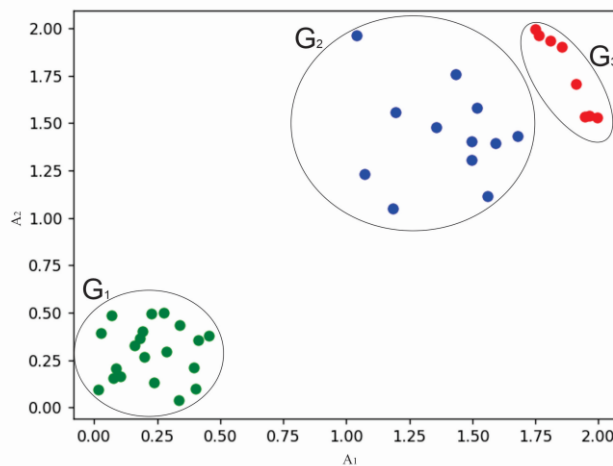


Figura 3.1 Exemplo de uma partição $G=\{G_1, G_2, G_3\}$, em que $|G_1| = 20$, $|G_2| = 12$ e $|G_3| = 8$, sendo $G_1 = \{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20\}$, $G_2 = \{21,23,24,25,27,28,30,33,34,35,36,40\}$ e $G_3 = \{22,26,29,31,32,37,38,39\}$.

A Figura 3.2 ilustra a partição G' formada a partir do conjunto C de instâncias descritas na Tabela 3.1, com dois subconjuntos, G_1 e G_2 . Ou seja, uma nova partição formada a partir do mesmo conjunto C de instâncias.

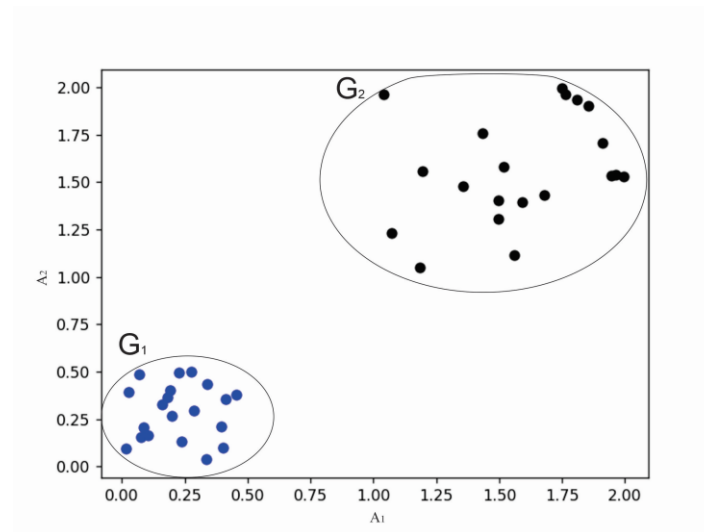


Figura 3.2 Exemplo de uma partição $G'=\{G_1,G_2\}$, em que $|G_1| = 20$ e $|G_2| = 20$, sendo $G_1 = \{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20\}$ e $G_2 = \{21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40\}$.

Tabela 3.1 Conjunto C contendo 40 instâncias geradas artificialmente, descritas por dois atributos numéricos.

#ID	A_1	A_2	#ID	A_1	A_2
1	0.1967	0.2662	21	1.4983	1.4035
2	0.4133	0.3558	22	1.8089	1.9349
3	0.3384	0.4357	23	1.3565	1.4794
4	0.1038	0.1643	24	1.0732	1.2317
5	0.1590	0.3250	25	1.5909	1.3962
6	0.0669	0.4874	26	1.9101	1.7050
7	0.3357	0.0379	27	1.1937	1.5585
8	0.2854	0.2935	28	1.4323	1.7566
9	0.0848	0.2069	29	1.7491	1.995
10	0.0738	0.1545	30	1.0391	1.9624
11	0.2380	0.1319	31	1.9463	1.5350
12	0.4540	0.3793	32	1.7636	1.9638
13	0.2760	0.4976	33	1.5588	1.1156
14	0.0164	0.0932	34	1.1838	1.0514
15	0.0269	0.3905	35	1.4979	1.3043
16	0.4025	0.0978	36	1.5178	1.5801
17	0.2256	0.4961	37	1.9942	1.5309
18	0.1913	0.4011	38	1.8548	1.9012
19	0.3948	0.2121	39	1.9624	1.5405
20	0.1821	0.3644	40	1.6789	1.4319

3.3 Categorias dos Algoritmos de Agrupamento

As categorias que agrupam diferentes tipos de algoritmos de agrupamento, dependendo dos critérios utilizados, podem ser bem diferenciadas de autor para autor. Autores como Jain, em [Jain *et. al.* 1999], Berkhin, em [Berkhin 2006], Theodoridis e Koutroubas, em [Theodoridis & Koutroubas 2009] apresentam suas próprias categorizações, não sendo, dessa forma, possível definir uma estrutura única de categorização, ficando a critério do usuário escolher aquela que melhor se adeque às suas necessidades. Para o trabalho descrito neste documento foi adotada a categorização proposta por Theodoridis e Koutroubas, que agrupa os algoritmos em:

- *Sequencial*: os algoritmos sequenciais têm como característica a criação de um único agrupamento e, via de regra, a ordem em que as instâncias são apresentadas interferem no resultado final. As instâncias podem ser apresentadas uma ou algumas vezes (normalmente não mais de cinco ou seis vezes). Os algoritmos sequenciais são considerados simples e rápidos e tendem a gerar agrupamentos compactos com formas esféricas ou elipsoidais, dependendo da medida de similaridade utilizada. O trabalho descrito em [Real 2014] investiga o algoritmo sequencial de agrupamento, como apresentado em [Theodoridis & Koutroubas 2009], bem como as duas variações desse algoritmo, também propostas pelos autores.
- *Hierárquicos*: Os algoritmos hierárquicos são caracterizados por induzirem um conjunto de agrupamentos que refletem uma estrutura em forma de árvore (dendograma) e podem ser caracterizados como aglomerativos ou divisivos,

Nos algoritmos hierárquicos aglomerativos, inicialmente, cada instância é considerada um grupo, em seguida, os dois grupos com maior similaridade são agrupados formando um novo grupo. Esse processo continua até que o número desejado de grupos seja atingido, criando, assim, uma árvore hierárquica. Nos algoritmos hierárquicos divisivos todas as instâncias são adicionadas a um único grupo e gradualmente divididas até chegar à um agrupamento em que o número de grupos produzidos é o número desejado [Kameshwaran & Malarvizhi 2014]. Detalhes sobre algoritmos divisivos e aglomerativos podem ser vistos em [Nietto 2016] e [Camargos 2016], respectivamente.

- *Baseados na otimização de uma função custo*: Essa categoria contém algoritmos em que a ‘sensibilidade’ é quantificada por uma função de custo J , pela qual o agrupamento é avaliado. Em geral, esses algoritmos fazem uso de uma função J e geram grupos sucessivos ao tentar otimizar J , e encerram quando um ótimo local de J é encontrado. Essa categoria inclui várias subcategorias, tais como: algoritmos *hard* (ou *crisp*), algoritmos probabilísticos, algoritmos nebulosos (*fuzzy*), entre outros.
- *Outros*: nessa categoria são agrupados algoritmos que se utilizam de técnicas especiais de agrupamento que não foram consideradas em qualquer das categorias anteriores (*e.g.*, *Branch and bound*, algoritmos de agrupamento baseados em algoritmos genéticos, algoritmos de agrupamento baseados na teoria dos grafos).

3.4 Medidas de Similaridade

A maioria dos algoritmos de agrupamento usa medidas de similaridade (a mais comum sendo a distância euclidiana) entre instâncias, como um critério para identificar instâncias similares. Dessa forma, instâncias próximas são consideradas similares e tendem a participar de um mesmo grupo, no agrupamento, enquanto instâncias distantes entre si, tendem a participar de grupos diferentes, no agrupamento sendo construído, uma vez que são consideradas não similares. A medida de similaridade deve ser escolhida com cuidado, de forma a melhor medir a similaridade entre as instâncias; sua escolha é fortemente direcionada pelas características do domínio de dado ao qual as instâncias a serem agrupadas pertencem.

Como comentado anteriormente, a distância *euclidiana*, via de regra, é uma das principais medidas para medir a similaridade entre instâncias de dados que participam de um processo de agrupamento. A distância euclidiana entre duas instâncias de dados, x_i e x_j , descritas por M atributos numéricos, é dada pela na Eq. (3.1).

$$d(x_i, x_j) = \sqrt{\sum_{m=1}^M (x_{i_m} - x_{j_m})^2} \quad (3.1)$$

O exemplo que segue, cujo diagrama está na Figura 3.3, mostra como a Eq. (3.1) é utilizada no cálculo da distância euclidiana entre as instâncias $x_1 = (A_1 = 2, A_2 = 1)$ e $x_2 = (A_1 = 5, A_2 = 6)$.

$$d(x_1, x_2) = \sqrt{(2 - 5)^2 + (1 - 6)^2} \cong 5.83$$

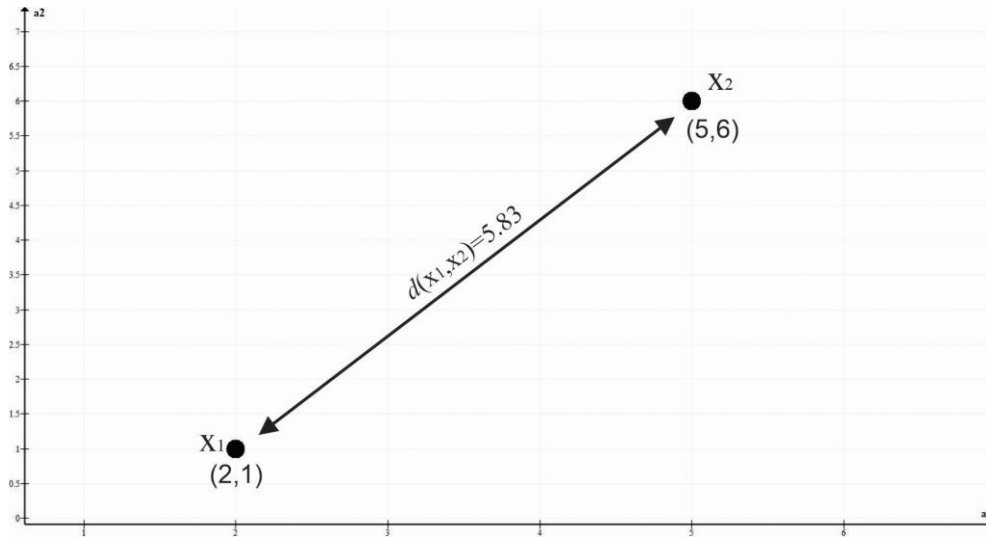


Figura 3.3 Representação da distância euclidiana entre duas instâncias.

Inúmeras outras medidas podem ser usadas para o cálculo de similaridade entre instâncias (*e.g.*, *Manhattan* e *Mahalanobis*); detalhes sobre tais medidas são encontrados em [Jain & Dubes 1988], [Berkhin 2006] e [Theodoridis & Koutroubas 2009].

3.5 Índices de Validação de Agrupamento

Índices de validação de agrupamento ajudam a validar agrupamentos que melhor reflitam uma partição natural das instâncias de dados, isto é, verifica quão representativo é o agrupamento induzido por um algoritmo de AM.

Métricas que implementam processos de validação de agrupamentos são fundamentais para a garantia de indução de agrupamentos que reflitam uma organização factível e, se possível, natural das instâncias. A avaliação da qualidade do agrupamento induzido é uma questão vital na área de aprendizado não-supervisionado quando algoritmos de agrupamento são utilizados.

Xiong e Li, em [Xiong & Li 2013], informam que até o momento, não se tem uma solução consistente e conclusiva sobre validação de agrupamentos e que existem vários

aspectos associados ao problema de indução de agrupamentos, que não foram ainda totalmente abordados na literatura. Dentre tais problemas, os autores consideram: (1) a importância de normalizar medidas de validação, (2) a definição e especificação das relações (se existentes) entre os diferentes índices de validação propostos na literatura, (3) a caracterização das propriedades relevantes na seleção das medidas de validação, entre outros.

Métricas que se caracterizam como de validação de agrupamentos são normalmente abordadas em dois grupos principais: aquelas voltadas à validação interna e as voltadas à validação externa. A validação interna é baseada em informações intrínsecas das instâncias (*e.g.*, densidade), enquanto a validação externa, geralmente, é baseada em um conhecimento prévio sobre as instâncias (*e.g.*, número de classes associadas às instâncias).

3.5.1 Índices de Validação Interna

Os índices de validação interna geralmente seguem os critérios de compactação e separação para melhor validar os grupos gerados. O conceito de compactação está ligado à proximidade das instâncias em um mesmo grupo, que é avaliada com base na variação, normalmente, da distância entre as instâncias (quanto menor for a distância entre as instâncias participantes de um grupo, maior é a compactação do grupo). A separação mede quão distinto ou bem separado um grupo é dos demais grupos [Liu *et al.* 2013] [Desgraupes 2003].

As duas subseções na sequência descrevem índices de validação interna de qualidade e buscam um agrupamento ótimo. Dessa forma, considere um conjunto C contendo N instâncias de dados, $C = \{x_1, x_2, \dots, x_N\}$, agrupadas em K grupos, $G_1, G_2, \dots, G_K$, formando o agrupamento $G = \{G_1, G_2, \dots, G_K\}$, em que n_k é o número de instâncias pertencente ao grupo G_k *i.e.*, $n_k = |G_k|$, $k = 1, \dots, K$.

3.5.1.1 Índice Dunn

O índice Dunn, proposto por Dunn, em [Dunn 1974], é baseado em medidas geométricas dos grupos e tem como objetivo comparar as distâncias entre instâncias de um mesmo grupo (*i.e.*, compactação) com as distâncias entre os grupos (*i.e.*, separação). Quanto

maior o valor do índice Dunn, melhor é o agrupamento induzido, tal que os valores do índice Dunn $\in [0, \infty)$.

A distância entre os grupos G_k e $G_{k'}$ é calculada como a distância entre suas duas instâncias mais próximas. Dessa forma, denotando $d_{\min}$ como a menor distância entre duas instâncias de grupos distintos, quanto maior for esse valor, mais separados os grupos estão [Desgraupes 2003]. A Eq. (3.2) mostra o cálculo de $d_{\min}$.

$$d_{\min} = \min_{\substack{x_i \in G_k \\ x_j \in G_{k'} \\ i \neq j}} d(x_i, x_j) \quad (3.2)$$

Considere diam_k ser o diâmetro do grupo G_k *i.e.*, a maior distância entre duas instâncias do grupo G_k . Quanto menor for o valor de diam_k , mais compacto é o grupo G_k , $k = 1, \dots, K$. A Eq. (3.3) mostra o cálculo de diam_k , para $k = 1, \dots, K$.

$$\text{diam}_k = \max_{\substack{i, j \in G_k \\ i \neq j}} d(x_i, x_j) \quad (3.3)$$

A métrica $d_{\max}$ é definida como o valor do maior diâmetro diam_k , para $k = 1, \dots, K$ e sua definição está formalizada pela Eq. (3.4).

$$d_{\max} = \max_{1 \leq k \leq K} \text{diam}_k \quad (3.4)$$

O Índice Dunn é definido como o quociente da divisão de $d_{\min}$ por $d_{\max}$, mostrado na Eq. (3.5).

$$\text{Dunn} = \frac{d_{\min}}{d_{\max}} \quad (3.5)$$

Para exemplificar o funcionamento do índice Dunn, é utilizado o agrupamento detalhado na Tabela 3.2.

A Figura 3.4 ilustra a *matriz de similaridade* (MS) de dimensão 40×40. Para facilitar a visualização da matriz, foi exposto somente o triângulo superior. As distâncias entre as instâncias foram calculadas utilizando a distância euclidiana Eq. (3.1).

#ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	0,000	0,234	0,221	0,138	0,070	0,257	0,267	0,093	0,127	0,166	0,140	0,281	0,245	0,250	0,210	0,266	0,232	0,135	0,205	0,099	1,729	2,320	1,678	1,304	1,795	2,238	1,632	1,936	2,324	1,894	2,161	2,310	1,605	1,261	1,665	1,863	2,198	2,329	2,178	1,886
2		0,000	0,109	0,364	0,256	0,371	0,327	0,142	0,361	0,395	0,284	0,047	0,197	0,476	0,388	0,258	0,234	0,227	0,145	0,231	1,508	2,108	1,467	1,097	1,571	2,015	1,434	1,732	2,115	1,724	1,934	2,100	1,375	1,038	1,441	1,649	1,970	2,113	1,950	1,661
3			0,000	0,359	0,211	0,276	0,398	0,152	0,342	0,386	0,320	0,129	0,088	0,470	0,315	0,344	0,128	0,151	0,231	0,172	1,511	2,100	1,458	1,083	1,578	2,020	1,411	1,715	2,103	1,680	1,948	2,090	1,397	1,046	1,449	1,643	1,985	2,109	1,964	1,670
4				0,000	0,170	0,325	0,264	0,223	0,047	0,032	0,138	0,411	0,375	0,113	0,239	0,306	0,354	0,252	0,295	0,215	1,866	2,458	1,816	1,442	1,931	2,374	1,770	2,074	2,462	2,027	2,296	2,448	1,738	1,398	1,801	2,001	2,333	2,466	2,313	2,022
5					0,000	0,187	0,337	0,130	0,139	0,191	0,209	0,300	0,208	0,272	0,147	0,333	0,184	0,083	0,261	0,046	1,720	2,305	1,663	1,288	1,788	2,230	1,610	1,916	2,306	1,859	2,158	2,294	1,608	1,256	1,659	1,850	2,196	2,315	2,175	1,880
6						0,000	0,524	0,292	0,281	0,333	0,395	0,402	0,209	0,397	0,105	0,514	0,159	0,151	0,428	0,169	1,699	2,265	1,627	1,252	1,775	2,209	1,555	1,864	2,259	1,767	2,152	2,249	1,619	1,251	1,648	1,816	2,192	2,279	2,168	1,868
7							0,000	0,260	0,302	0,287	0,136	0,361	0,463	0,324	0,469	0,090	0,471	0,391	0,184	0,361	1,793	2,402	1,766	1,403	1,850	2,293	1,746	2,039	2,414	2,049	2,199	2,398	1,630	1,322	1,719	1,943	2,232	2,404	2,214	1,936
8								0,000	0,218	0,253	0,168	0,189	0,204	0,335	0,276	0,228	0,211	0,143	0,136	0,125	1,644	2,240	1,598	1,225	1,709	2,152	1,557	1,859	2,245	1,831	2,074	2,230	1,516	1,175	1,579	1,782	2,110	2,247	2,090	1,799
9									0,000	0,054	0,171	0,407	0,348	0,133	0,193	0,336	0,322	0,221	0,310	0,185	1,852	2,441	1,799	1,424	1,919	2,361	1,748	2,054	2,443	1,998	2,287	2,430	1,732	1,386	1,789	1,985	2,324	2,450	2,303	2,010
10										0,000	0,166	0,442	0,398	0,084	0,241	0,334	0,374	0,273	0,326	0,236	1,894	2,486	1,844	1,469	1,961	2,403	1,796	2,101	2,489	2,049	2,326	2,476	1,769	1,427	1,830	2,029	2,363	2,495	2,343	2,051
11											0,000	0,328	0,368	0,225	0,334	0,168	0,364	0,273	0,176	0,239	1,790	2,391	1,751	1,381	1,852	2,296	1,717	2,016	2,399	1,998	2,211	2,384	1,647	1,319	1,721	1,933	2,245	2,397	2,227	1,941
12												0,000	0,214	0,523	0,427	0,286	0,256	0,264	0,177	0,272	1,463	2,063	1,423	1,054	1,525	1,969	1,392	1,689	2,071	1,688	1,887	2,056	1,328	0,992	1,395	1,604	1,923	2,068	1,904	1,615
13													0,000	0,480	0,271	0,419	0,050	0,128	0,309	0,163	1,521	2,101	1,460	1,084	1,593	2,032	1,403	1,709	2,101	1,652	1,966	2,089	1,424	1,063	1,464	1,647	2,005	2,112	1,983	1,686
14														0,000	0,297	0,386	0,454	0,354	0,397	0,318	1,978	2,570	1,928	1,553	2,044	2,487	1,880	2,184	2,573	2,131	2,409	2,560	1,850	1,510	1,913	2,113	2,445	2,578	2,425	2,134
15															0,000	0,476	0,225	0,165	0,409	0,157	1,786	2,358	1,719	1,343	1,860	2,297	1,651	1,960	2,354	1,870	2,235	2,343	1,695	1,332	1,732	1,907	2,274	2,371	2,251	1,953
16																0,000	0,436	0,370	0,114	0,346	1,705	2,314	1,679	1,317	1,760	2,204	1,661	1,952	2,327	1,970	2,109	2,310	1,540	1,233	1,630	1,855	2,142	2,315	2,125	1,846
17																	0,000	0,101	0,331	0,139	1,563	2,139	1,499	1,122	1,635	2,073	1,437	1,745	2,137	1,677	2,010	2,126	1,470	1,107	1,507	1,687	2,049	2,151	2,027	1,728
18																		0,000	0,278	0,038	1,647	2,229	1,588	1,212	1,717	2,157	1,531	1,838	2,229	1,777	2,089	2,217	1,543	1,187	1,588	1,775	2,128	2,240	2,106	1,810
19																			0,000	0,262	1,624	2,229	1,591	1,225	1,683	2,127	1,566	1,861	2,239	1,865	2,039	2,223	1,474	1,152	1,552	1,770	2,073	2,233	2,055	1,771
20																				0,000	1,677	2,261	1,619	1,244	1,746	2,187	1,565	1,871	2,262	1,813	2,117	2,249	1,568	1,215	1,617	1,806	2,155	2,271	2,134	1,838
21																					0,000	0,616	0,161	0,458	0,093	0,510	0,342	0,359	0,643	0,723	0,467	0,620	0,294	0,472	0,099	0,178	0,512	0,612	0,484	0,183
22																						0,000	0,642	1,018	0,581	0,251	0,721	0,417	0,085	0,770	0,423	0,054	0,857	1,082	0,703	0,459	0,444	0,057	0,423	0,520
23																							0,000	0,376	0,249	0,598	0,181	0,287	0,648	0,578	0,592	0,633	0,416	0,462	0,225	0,190	0,640	0,653	0,609	0,326
24																								0,000	0,543	0,961	0,348	0,636	1,020	0,731	0,924	1,006	0,499	0,212	0,431	0,565	0,968	1,029	0,941	0,638
25																									0,000	0,444	0,429	0,394	0,620	0,791	0,381	0,593	0,283	0,534	0,131	0,198	0,425	0,570	0,398	0,095
26																										0,000	0,731	0,481	0,332	0,908	0,174	0,297	0,686	0,977	0,575	0,412	0,193	0,204	0,173	0,358
27																											0,000	0,310	0,707	0,432	0,753	0,699	0,574	0,507	0,396	0,325	0,801	0,745	0,769	0,501
28																												0,000	0,397	0,444	0,560	0,391	0,653	0,748	0,457	0,196	0,605	0,447	0,572	0,408
29																													0,000	0,711	0,501	0,035	0,900	1,100	0,735	0,475	0,525	0,142	0,502	0,568
30																														0,000	1,003	0,724	0,994	0,922	0,802	0,613	1,048	0,818	1,015	0,831
31																															0,000	0,466	0,571	0,903	0,504	0,431	0,048	0,377	0,017	0,287
32																																0,000	0,873	1,081	0,711	0,456	0,490	0,111	0,468	0,539
33																																	0,000	0,380	0,198	0,466	0,602	0,840	0,586	0,338
34																																		0,000	0,403	0,625	0,942	1,083	0,919	0,624
35																																			0,000	0,277	0,546	0,695	0,521	0,221
36																																				0,000	0,479	0,465	0,446	0,219
37																																					0,000	0,396	0,033	0,330
38																																						0,000	0,376	0,501
39																																							0,000	0,304
40																																								0,000

Figura 3.4 Triângulo superior da matriz de similaridade gerada a partir do agrupamento apresentado na Tabela 3.2.

Tabela 3.2 Agrupamento contendo 40 instâncias divididas em três grupos, conforme ilustrado na Figura 3.1.

#ID	A ₁	A ₂	G	#ID	A ₁	A ₂	G
1	0.1967	0.2662	1	21	1.4983	1.4035	2
2	0.4133	0.3558	1	22	1.8089	1.9349	3
3	0.3384	0.4357	1	23	1.3565	1.4794	2
4	0.1038	0.1643	1	24	1.0732	1.2317	2
5	0.1590	0.3250	1	25	1.5909	1.3962	2
6	0.0669	0.4874	1	26	1.9101	1.7050	3
7	0.3357	0.0379	1	27	1.1937	1.5585	2
8	0.2854	0.2935	1	28	1.4323	1.7566	2
9	0.0848	0.2069	1	29	1.7491	1.995	3
10	0.0738	0.1545	1	30	1.0391	1.9624	2
11	0.2380	0.1319	1	31	1.9463	1.5350	3
12	0.4540	0.3793	1	32	1.7636	1.9638	3
13	0.2760	0.4976	1	33	1.5588	1.1156	2
14	0.0164	0.0932	1	34	1.1838	1.0514	2
15	0.0269	0.3905	1	35	1.4979	1.3043	2
16	0.4025	0.0978	1	36	1.5178	1.5801	2
17	0.2256	0.4961	1	37	1.9942	1.5309	3
18	0.1913	0.4011	1	38	1.8548	1.9012	3
19	0.3948	0.2121	1	39	1.9624	1.5405	3
20	0.1821	0.3644	1	40	1.6789	1.4319	2

O exemplo do cálculo do índice Dunn foi dividido em 3 passos para maior clareza.

Passo 1: Calcular a menor distância entre os grupos $d_{\min}$.

$$d_{\min} = \min \begin{bmatrix} d(G_1, G_2) = d(12,34) = 0,992 \\ d(G_1, G_3) = d(12,40) = 1,615 \\ d(G_2, G_3) = d(31,40) = 0,287 \end{bmatrix} = 0,287$$

A Figura 3.5 ilustra a menor distância entre duas instâncias que pertençam a grupos distintos do agrupamento, sendo destacado em verde a menor distância entre os grupos.

Passo 2: Calcular o diâmetro $diam_k$ do grupo G_k , ou seja, encontrar a maior distância entre duas instâncias que pertençam ao grupo G_k , para cada grupo pertencente ao agrupamento.

$$diam_1 = d(6,7) = 0,524$$

$$diam_2 = d(30,33) = 0,994$$

$$diam_3 = d(29,37) = 0,525$$

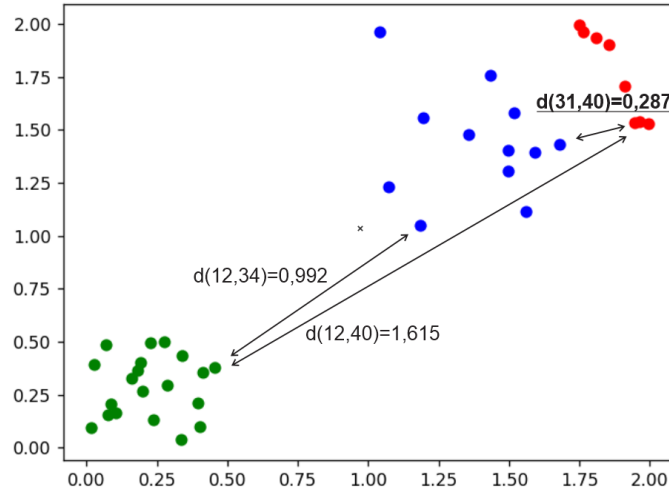


Figura 3.5 Menor distância entre duas instâncias que pertencem a grupos distintos, sendo destacado a menor distância entre os grupos.

Selecionar o maior diâmetro dos grupos, isto é, o maior $diam_k$.

$$d_{\max} = \max \begin{bmatrix} diam_1 = 0,524 \\ diam_2 = 0,994 \\ diam_3 = 0,525 \end{bmatrix} = 0,994$$

A Figura 3.6 ilustra o diâmetro dos grupos, sendo destacado em verde o maior diâmetro entre eles.

Passo 3: Calcular o índice Dunn a partir do quociente da divisão de $d_{\min}$ por $d_{\max}$.

$$Dunn = \frac{0,287}{0,994} \cong 0,2887$$

A complexidade de tempo de execução para o cálculo do índice Dunn com relação a um agrupamento é dada por $O(MN^2)$, sendo M o número de atributos das instâncias e N o número de instâncias no conjunto C . Assim, cada instância é comparada com todas as demais instâncias do conjunto, de forma que o número de grupos do agrupamento não influencia na complexidade do tempo de execução do algoritmo [Vendramin *et al.* 2010].

As principais desvantagens do índice Dunn estão ligadas ao tempo de execução em conjunto de instâncias volumoso e ao significativo impacto que ruídos no conjunto de instâncias exercem sobre o resultado final (o diâmetro máximo dos grupos e a separação

entre os grupos podem variar drasticamente em conjunto de instâncias que apresentem ruídos) [Nietto 2016].

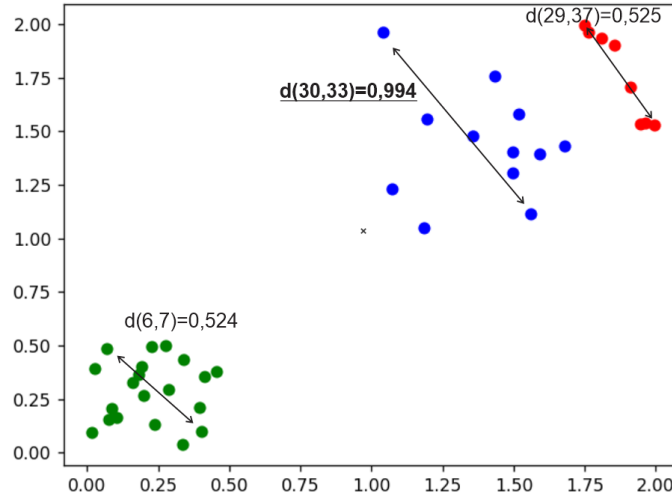


Figura 3.6 Diâmetro dos grupos, sendo destacado o maior diâmetro.

3.5.1.2 Índice Silhouette

O índice Silhouette, proposto por Rousseeuw, em [Rousseeuw 1987], normalmente é utilizado para medir a qualidade do resultado do agrupamento e, assim como o índice Dunn, é baseado em medidas geométricas dos grupos. O índice Silhouette é uma medida de como uma instância é semelhante às demais instâncias de seu grupo, comparada com as instâncias dos outros grupos. Ou seja, o Silhouette avalia a qualidade de uma solução levando em conta a compactação das instâncias no grupo e a separação entre os grupos.

Considere uma instância $x_i \in G_k$. Seja a_i a média das distâncias de x_i a cada uma das outras instâncias de G_k , como calculada pela Eq. (3.6), em que $n_k = |G_k|$. Pode-se entender o valor de a_i como uma medida de quão bem atribuída ao grupo G_k a instância x_i está.

$$a_i = \frac{1}{n_k - 1} \sum_{\substack{j \in G_k \\ j \neq i}} d(x_i, x_j) \quad (3.6)$$

Considere calcular a média das distâncias de $x_i \in G_k$ a todas as instâncias de cada um dos demais grupos $G_{k'}$, $k' \neq k$, $d_m(I_i, G_{k'})$, como estabelece a Eq. (3.7).

$$d_m(x_i, G_{k'}) = \frac{1}{n_{k'}} \sum_{j \in G_{k'}} d(x_i, x_j) \quad (3.7)$$

Ainda para $x_i \in G_k$, considere o cálculo de todos os valores $d_m(x_i, G_{k'})$, para todos os grupos $G_{k'}$, $k' \neq k$. Considere também o valor b_i , que identifica a menor distância do conjunto das $k-1$ distâncias, dado por $\{d_m(x_i, G_1), d_m(x_i, G_2), \dots, d_m(x_i, G_k)\}$, como especifica a Eq. (3.8).

$$b_i = \min_{k' \neq k} d_m(x_i, G_{k'}) \quad (3.8)$$

Para cada instância I_i é calculado um valor, pela Eq. (3.9), chamado de Silhouette da instância I_i , notado por s_i .

$$s_i = \frac{b_i - a_i}{\max\{a_i, b_i\}} \quad (3.9)$$

O valor de s_i é tal que $s_i \in [-1, 1]$. Valores de s_i próximos a 1 indicam que a instância I_i efetivamente pertence ao grupo G_k . Quanto mais próximos de -1 forem os valores de Silhouette de x_i , maior é a possibilidade que x_i pertença a outro grupo.

A média dos valores de Silhouette das instâncias de um grupo G_k é chamada de Silhouette do grupo G_k e notada por S_k , como mostra a Eq. (3.10).

$$S_k = \frac{1}{n_k} \sum_{i \in G_k} s_i \quad (3.10)$$

O índice Silhouette associado ao agrupamento como um todo é calculado como a média dos índices Silhouette dos grupos que o definem e formalmente seu cálculo é especificado pela Eq. (3.11).

$$\text{Silhouette} = \frac{1}{K} \sum_{k=1}^K S_k \quad (3.11)$$

Para exemplificar o cálculo do índice Silhouette o agrupamento descrito na Tabela 3.2 é considerado, bem como a MS ilustrada na Figura 3.4. O cálculo do índice foi dividido em 5 passos para melhor entendimento.

Passo 1: Calcular a_i para cada uma das instâncias pertencentes ao agrupamento apresentado na Tabela 3.2.

$$a_1 = \frac{0,234 + 0,221 + \dots + 0,205 + 0,099}{19} \cong 0,1913$$

$$a_2 = \frac{0,234 + 0,109 + \dots + 0,145 + 0,231}{19} \cong 0,2656$$

$$\dots$$

$$a_{22} = \frac{0,510 + 0,643 + 0,467 + 0,620 + 0,512 + 0,612 + 0,484}{7} \cong 0,2482$$

$$\dots$$

$$a_{40} = \frac{1,886 + 1,661 + \dots + 0,501 + ,304}{11} \cong 0,3986$$

Nos cálculos apresentados acima (e em outros a seguir) valores foram suprimidos de forma a permitir a ilustração, porém, sem comprometer o entendimento. A Tabela 3.3 apresenta os valores calculados de a_i .

Tabela 3.3 Valores calculados de a_i , conforme Eq. (3.6).

#ID	Valor	#ID	Valor	#ID	Valor	#ID	Valor
a₁	0,1913	a₁₁	0,2422	a₂₁	0,3057	a₃₁	0,2865
a₂	0,2656	a₁₂	0,3009	a₂₂	0,2482	a₃₂	0,2743
a₃	0,2526	a₁₃	0,2705	a₂₃	0,3137	a₃₃	0,4633
a₄	0,2430	a₁₄	0,3309	a₂₄	0,4943	a₃₄	0,5354
a₅	0,1965	a₁₅	0,2812	a₂₅	0,3398	a₃₅	0,3310
a₆	0,3039	a₁₆	0,3162	a₂₆	0,2320	a₃₆	0,3501
a₇	0,3271	a₁₇	0,2644	a₂₇	0,3951	a₃₇	0,3043
a₈	0,1990	a₁₈	0,2104	a₂₈	0,4447	a₃₈	0,2374
a₉	0,2366	a₁₉	0,2618	a₂₉	0,3031	a₃₉	0,2846
a₁₀	0,2614	a₂₀	0,1985	a₃₀	0,7146	a₄₀	0,3986

Passo 2: Calcular b_i para cada instância do agrupamento, isto é, calcular a menor distância média entre a instância e os grupos que ela não participa. Para melhor visualização foram unificadas as Eq. (3.7) e Eq. (3.8). A Tabela 3.4 apresenta os valores calculados de b_i .

$$b_1 = \min \left[\begin{array}{l} d_m(I_1, G_2) = \frac{1,7285 + 1,6784 + \dots + 1,8633 + 1,8857}{12} \cong \frac{20,2486}{12} \cong 1,6873 \\ d_m(I_1, G_3) = \frac{2,3204 + 2,2375 + \dots + 2,3286 + 2,1775}{8} \cong \frac{18,0576}{8} \cong 2,2572 \end{array} \right] \cong 1,6873$$

$$\begin{aligned}
b_2 &= \min \left[\begin{aligned} d_m(I_2, G_2) &= \frac{1,5083 + 1,4670 + \dots + 1,6489 + 1,6613}{12} \cong \frac{17,6977}{12} \cong 1,4748 \\ d_m(I_2, G_3) &= \frac{2,1075 + 2,0152 + \dots + 2,1133 + 1,9502}{8} \cong \frac{16,3051}{8} \cong 2,0381 \end{aligned} \right] \cong 1,4748 \\
&\dots \\
b_{22} &= \min \left[\begin{aligned} d_m(I_2, G_1) &= \frac{2,3204 + 2,1075 + \dots + 2,2289 + 2,2612}{20} \cong \frac{45,7814}{20} \cong 2,2890 \\ d_m(I_2, G_2) &= \frac{0,6156 + 0,6420 + \dots + 0,4589 + 0,5195}{12} \cong \frac{8,3854}{12} \cong 0,6987 \end{aligned} \right] \cong 0,6987 \\
&\dots \\
b_{40} &= \min \left[\begin{aligned} d_m(I_2, G_1) &= \frac{1,8857 + 1,6613 + \dots + 1,7711 + 1,8384}{20} \cong \frac{37,1077}{20} \cong 1,8553 \\ d_m(I_2, G_3) &= \frac{0,5195 + 0,3578 + \dots + 0,5011 + 0,3035}{8} \cong \frac{3,4055}{8} \cong 0,4256 \end{aligned} \right] \cong 0,4256
\end{aligned}$$

Tabela 3.4 Valores calculados de b_i , conforme Eq. (3.8).

#ID	Valor	#ID	Valor	#ID	Valor	#ID	Valor
b₁	1,6873	b₁₁	1,7555	b₂₁	0,5579	b₃₁	0,6146
b₂	1,4748	b₁₂	1,4306	b₂₂	0,6987	b₃₂	0,6938
b₃	1,4701	b₁₃	1,4754	b₂₃	0,6268	b₃₃	0,7392
b₄	1,8237	b₁₄	1,9349	b₂₄	0,9835	b₃₄	1,0109
b₅	1,6746	b₁₅	1,7338	b₂₅	0,5016	b₃₅	0,6238
b₆	1,6450	b₁₆	1,6790	b₂₆	0,6368	b₃₆	0,4529
b₇	1,7662	b₁₇	1,5148	b₂₇	0,7407	b₃₇	0,6581
b₈	1,6061	b₁₈	1,6009	b₂₈	0,4836	b₃₈	0,7047
b₉	1,8079	b₁₉	1,5944	b₂₉	0,7103	b₃₉	0,6304
b₁₀	1,8517	b₂₀	1,6316	b₃₀	0,8746	b₄₀	0,4256

Passo 3: Calcular o Silhouette da instância para cada instância pertencente ao agrupamento. A Tabela 3.5 apresenta os Silhouette das instâncias do agrupamento.

$$\begin{aligned}
s_1 &= \frac{b_1 - a_1}{\max \{a_1, b_1\}} = \frac{1,6873 - 0,1913}{1,6873} \cong 0,8865 \\
s_2 &= \frac{b_2 - a_2}{\max \{a_2, b_2\}} = \frac{1,4748 - 0,2656}{1,4748} \cong 0,8199 \\
&\dots \\
s_{22} &= \frac{b_{22} - a_{22}}{\max \{a_{22}, b_{22}\}} = \frac{0,6987 - 0,2482}{0,6987} \cong 0,6447 \\
&\dots \\
s_{40} &= \frac{b_{40} - a_{40}}{\max \{a_{40}, b_{40}\}} = \frac{0,4256 - 0,39863}{0,4256} \cong 0,0635
\end{aligned}$$

Tabela 3.5 Valores do Silhouette das instâncias, conforme Eq. (3.9)

#ID	Valor	#ID	Valor	#ID	Valor	#ID	Valor
s ₁	0,8865	s ₁₁	0,8620	s ₂₁	0,4521	s ₃₁	0,5337
s ₂	0,8199	s ₁₂	0,7896	s ₂₂	0,6447	s ₃₂	0,6045
s ₃	0,8281	s ₁₃	0,8166	s ₂₃	0,4995	s ₃₃	0,3732
s ₄	0,8667	s ₁₄	0,8289	s ₂₄	0,4973	s ₃₄	0,4703
s ₅	0,8826	s ₁₅	0,8377	s ₂₅	0,3225	s ₃₅	0,4694
s ₆	0,8152	s ₁₆	0,8116	s ₂₆	0,6356	s ₃₆	0,2269
s ₇	0,8147	s ₁₇	0,8254	s ₂₇	0,4665	s ₃₇	0,5375
s ₈	0,8760	s ₁₈	0,8685	s ₂₈	0,0804	s ₃₈	0,6630
s ₉	0,8691	s ₁₉	0,8358	s ₂₉	0,5732	s ₃₉	0,5484
s ₁₀	0,8587	s ₂₀	0,8783	s ₃₀	0,1829	s ₄₀	0,0635

Passo 4: Calcular o Silhouette do Grupo, para cada um dos grupos pertencentes ao agrupamento.

$$S_1 = \frac{1}{20} * (0,8865 + 0,8199 + \dots + 0,8358 + 0,8783) \cong 0,8436$$

$$S_2 = \frac{1}{12} * (0,4521 + 0,4995 + \dots + 0,2269 + 0,0635) \cong 0,3420$$

$$S_3 = \frac{1}{8} * (0,6447 + 0,6356 + \dots + 0,6630 + 0,5484) \cong 0,5926$$

Passo 5: Calcular o índice Silhouette associado ao agrupamento como um todo, como a média dos Silhouette dos grupos.

$$\text{Silhouette} = \frac{1}{3} (0,8436 + 0,3420 + 0,5926) = 0,5927$$

A complexidade de tempo de execução para o cálculo do valor do índice Silhouette de um agrupamento é dada por $O(MN^2)$, sendo M o número de atributos que descrevem as instâncias e N o número de instâncias do conjunto C [Vendramin *et.al.* 2010].

3.5.2 Índices de Validação Externa

Os índices de validação externa são projetados para medir a semelhança entre duas partições e levam em conta apenas a distribuição das instâncias nos diferentes grupos, não permitindo medir a qualidade dessa distribuição [Desgraupes 2003].

Os índices de validação externos abordados utilizam de uma *matriz de confusão* (MC) para alcançar seus objetivos. A MC utiliza de pares de instâncias, verificando se esses pares pertencem ao mesmo grupo ou não, de acordo com a partição P_1 ou de acordo com a partição P_2 . Sendo assim, existem quatro possibilidades:

- (1) a: As duas instâncias compartilham um mesmo grupo da partição P_1 e um mesmo grupo da partição P_2 ;
- (2) b: As duas instâncias compartilham um mesmo grupo da partição P_1 , mas estão em grupos diferentes na partição P_2 ;
- (3) c: As duas instâncias estão em grupos diferentes da partição P_1 e compartilham o mesmo grupo da partição P_2 ;
- (4) d: As duas instâncias estão em grupos diferentes da partição P_1 e estão em grupos diferentes da partição P_2 .

Os termos ‘a’ e ‘d’ indicam que as atribuições dos pares de instâncias aos grupos ocorreram de forma consistente, enquanto os termos ‘b’ e ‘c’ são considerados atribuições inconsistentes.

Considere um conjunto com N instâncias de dados. O total de número de pares de instâncias a ser considerado para a especificação da MC é dado por $N_t = \frac{N(N-1)}{2}$, que representa o número de possíveis combinações das N instâncias, consideradas duas a duas.

Para exemplificar a MC são utilizadas as partições G e G' ilustradas respectivamente nas Figura 3.1 e Figura 3.2. A Tabela 3.6 descreve os grupos e as instâncias a eles atribuídas, conforme partição G e G' .

Tabela 3.6 Instâncias associadas aos grupos, conforme partições G e partição G' .

#ID	G	G'	#ID	G	G'	#ID	G	G'	#ID	G	G'
1	1	1	11	1	1	21	2	2	31	3	2
2	1	1	12	1	1	22	3	2	32	3	2
3	1	1	13	1	1	23	2	2	33	2	2
4	1	1	14	1	1	24	2	2	34	2	2
5	1	1	15	1	1	25	2	2	35	2	2
6	1	1	16	1	1	26	3	2	36	2	2
7	1	1	17	1	1	27	2	2	37	3	2
8	1	1	18	1	1	28	2	2	38	3	2
9	1	1	19	1	1	29	3	2	39	3	2
10	1	1	20	1	1	30	2	2	40	2	2

Para o exemplo desenvolvido, um total de 780 pares de instâncias são formados e passam por essa verificação. A Tabela 3.7 apresenta os pares de instâncias e o termo a

qual é classificado. A MC apresentada na Tabela 3.8 é formada das verificações se os pares de instâncias participam do mesmo grupo ou não, conforme partição G e G’.

Tabela 3.7 Pares de instâncias do conjunto C, onde PG e PG’ representam se os pares de instâncias pertencem ao mesmo grupo ou não, segundo as partições G e G’.

#ID - #ID	PG	PG’	Termo	#ID - #ID	PG	PG’	Termo
1-2	sim	sim	a	22-1	não	não	d
1-3	sim	sim	a	22-2	não	não	d
...				22-3	não	não	d
1-21	não	não	d	...			
1-22	não	não	d	40-35	sim	sim	a
...				40-36	sim	sim	a
2-1	sim	sim	a	40-37	não	sim	c
2-3	sim	sim	a	40-38	não	sim	c
2-4	sim	sim	a	40-39	não	sim	c

Tabela 3.8 Matriz de confusão gerada a partir das partições G e G’ apresentadas na Tabela 3.6, onde ‘sim’ expressa que as duas instâncias participam do mesmo grupo na partição e ‘não’ expressa que não participam do mesmo grupo na partição.

	G’ (sim)	G’ (não)
G (sim)	a= 284	b=0
G (não)	c=96	d=400

3.5.2.1 Índice de Rand

O índice Rand, proposto por Rand, em [Rand 1971], tem como objetivo ser uma medida para avaliar um agrupamento induzido (partição) por algum algoritmo, em relação a uma outra partição independente, criada com base em algum critério (*e.g.*, classes associadas às instâncias). Considerando a MC já descrita, o índice Rand é dado pela Eq. (3.12).

$$I_{\text{Rand}} = \frac{a+d}{a+b+c+d} \quad (3.12)$$

O valor do índice Rand varia no intervalo [0,1]. O valor 0 indica que as partições são totalmente diferentes entre si (inconsistentes) e o valor 1 indica que as partições são iguais (semelhantes).

Uma crítica comum ao índice Rand é que ele dá a mesma importância aos termos ‘a’ e ‘d’ da MC, não diferenciando entre pares de instâncias que estão no mesmo grupo ou em grupos distintos. Essa consideração pode acarretar que o termo ‘d’ domine os demais

termos (a, b e c), fazendo o índice incapaz de distinguir adequadamente entre boas e más partições. Essa situação é particularmente crítica quando o número de grupos aumenta, uma vez que o valor do termo ‘d’ tende a aumentar [Desgraupes 2003] [Vendramin *et.al.* 2010].

O exemplo do índice de Rand calculado a seguir utiliza a MC apresentada na Tabela 3.7.

$$I_{\text{Rand}} = \frac{284+400}{284+0+96+400}$$

$$I_{\text{Rand}} \cong 0,8769$$

3.5.2.2 Índice de Jaccard

A lógica por trás do índice Jaccard, proposto inicialmente por Jaccard, em [Jaccard 1901], é essencialmente a mesma do índice Rand, exceto pela ausência do termo ‘d’, pois considera o termo ‘d’ ‘neutro’, por entender que o termo ‘d’ conta pares de instâncias que não são claramente indicativos de semelhança ou inconsistência [Desgraupes 2003] [Vendramin *et.al.* 2010]. O índice de Jaccard está definido na Eq. (3.13).

$$I_{\text{Jaccard}} = \frac{a}{a + b + c} \quad (3.13)$$

O exemplo do índice de Jaccard calculado a seguir utiliza a MC apresentada na Tabela 3.7.

$$I_{\text{Jaccard}} = \frac{284}{284 + 0 + 96}$$

$$I_{\text{Jaccard}} = 0,7473$$

Capítulo 4

CONSIDERAÇÕES SOBRE O ALGORITMO k-MEANS

4.1 Introdução

Dentre os inúmeros algoritmos de agrupamento disponíveis na literatura, aquele conhecido como k-Means é o objeto de estudo e investigação deste trabalho. O algoritmo k-Means é um dos mais utilizados dentre os algoritmos de agrupamento. O termo k-Means foi empregado pela primeira vez em 1967, por James MacQueen, em [MacQueen 1967], muito embora o procedimento descrito pelo algoritmo esteja ligado a Hugo Steinhaus desde 1957 (ver [Steinhaus 1957]).

O k-Means padrão (também referenciado como *standard* ou como *básico*) foi proposto em 1957 por Stuart Lloyd [Lloyd 1957], em um documento interno à Bell Telephone Laboratories, como uma técnica para a modulação *pulse-code*. O algoritmo foi publicado em um artigo escrito por MacQueen [MacQueen 1967], nos anais de um simpósio junto à Universidade de Berkeley e foi publicado em periódico por Lloyd apenas em 1982 [Lloyd 1982]. Em 1965, Forgy publicou essencialmente o mesmo método em [Forgy 1965], razão pela qual o algoritmo é também referenciado como algoritmo Lloyd-Forgy. Uma versão mais eficiente foi proposta e publicada em Fortran, por Hartigan e Wong em [Hartigan & Wong 1979].

O k-Means, abordado de uma maneira simplificada e objetiva, pode ser descrito como um processo que, dado um conjunto com N instâncias de dados (via de regra descritas como vetores de M valores, cada um deles associado a um atributo de um conjunto com M atributos), busca particionar o conjunto dado em K grupos, em que K é um parâmetro geralmente fornecido pelo usuário.

Cada instância de dado vai pertencer ao grupo cujo *centroide* (geralmente definido como a média das instâncias do grupo) lhe for mais próxima. Esse procedimento,

implementado como um processo iterativo, induz o particionamento do espaço de instâncias no que é conhecido como células de Voronoi [Reddy & Jana 2012].

É importante comentar que o k-Means tem sido objeto de pesquisa e tentativas de melhoramento de seu desempenho desde quando foi criado. A pesquisa descrita neste documento, contempla a investigação com vistas à melhoria de desempenho do k-Means por meio do uso da desigualdade triangular, com vistas a diminuir o número de cálculos realizado pelo algoritmo e, com isso, acelerar a sua execução.

4.2 Notações e Definições Envolvidos no Algoritmo k-Means

Considere um conjunto de instâncias de dados C , contendo N instâncias, especificado e notado como $C = \{x_1, x_2, \dots, x_N\}$, em que cada instância, x_i , $1 \leq i \leq N$ é descrita por M atributos, $A_1, A_2, \dots, A_M$. Define-se como um *K-agrupamento* (*K-clustering*) de C uma partição de C em K conjuntos de instâncias (grupos), notadas por $G_1, G_2, \dots, G_K$. O K -agrupamento de C é então notado como: $G = \{G_1, G_2, \dots, G_K\}$, e n_k é o número de instâncias pertencentes ao grupo G_k i.e., $n_k = |G_k|$.

Um grupo de um agrupamento pode ser representado pelas instâncias que dele fazem parte ou, então, por uma instância específica, denominada centroide que, como informado anteriormente, pode ser determinado como a média das instâncias que pertencem ao grupo em questão, como estabelece a Eq. (4.1). Um agrupamento com K grupos é, então, representado por um conjunto de K centroides. Note que o centroide não necessariamente é uma instância do conjunto de instâncias sendo considerado para o processo de agrupamento.

$$c_k = \frac{1}{n_k} \sum_{x_i \in G_k} x_i \quad (4.1)$$

4.2.1 Método dos Mínimos Quadrados

Técnicas de particionamento de instâncias de dados geralmente produzem agrupamentos ao otimizar uma função de critério, indicada por J , definida localmente (em um subconjunto de instâncias) ou globalmente (em todas as instâncias do conjunto).

Frequentemente, uma das formas de buscar essa otimização é por meio do uso do *método dos mínimos quadrados*, que consiste em otimizar a função J utilizando como

critério a minimização da *soma dos quadrados dos erros* (SQE). O método dos mínimos quadrados procura encontrar o melhor ajuste para o conjunto de instâncias, buscando minimizar o valor da SQE entre as instâncias e o centroide do grupo ao qual foram atribuídas, na tentativa de gerar um agrupamento ótimo [Jain *et. al.* 1999]. A função SQE é expressa formalmente pela Eq. (4.2).

$$SQE(G) = \sum_{k=1}^K \sum_{x_i \in G_k} d(x_i, c_k)^2 \quad (4.2)$$

A função SQE pode ser vista como uma avaliação da dispersão das instâncias dentro dos grupos aos quais pertencem. Melhorar a função SQE é conhecido como um problema exponencial (*i.e.*, NP-Difícil), mesmo para $K=2$ [Jain 2010] [Drineas *et al.* 2004].

4.3 O algoritmo k-Means

O algoritmo k-Means é um algoritmo de agrupamento bem popular que é caracterizado como um algoritmo particional. O algoritmo utiliza uma heurística que aplica uma abordagem gananciosa para otimizar a função SQE, convergindo, assim, para um agrupamento ótimo local (mínimo local) e não global. O algoritmo k-Means busca uma partição do conjunto de instâncias que tenha K grupos, tal que a soma dos quadrados dos erros entre os centroides e as instâncias a eles atribuídas seja minimizada; o algoritmo, porém, não garante que o agrupamento induzido é o melhor agrupamento possível [Jain 2010].

Na versão padrão (*standard*) do algoritmo k-Means dada por Lloyd, em [Lloyd 1982] os K centroides iniciais são escolhidos aleatoriamente entre as N instâncias pertencentes ao conjunto C , onde K representa o número de grupos a serem induzidos pelo algoritmo k-Means.

A escolha aleatória dos centroides iniciais no algoritmo k-Means pode acarretar alguns inconvenientes, tais como: (1) a cada execução do algoritmo k-Means o agrupamento induzido não necessariamente será o mesmo, mesmo utilizando conjunto de instâncias e número de grupos K iguais; (2) pode ocorrer a escolha de instância com valor extremo como centroide inicial, acarretando distorções no agrupamento induzido, não representando adequadamente os grupos naturais do conjunto; (3) duas instâncias muito

próximas podem ser escolhidas como centroides iniciais, levando o agrupamento a uma solução local que também não representa adequadamente os grupos naturais do conjunto de instâncias; entre outros [Khan & Ahmad 2004].

A escolha dos centroides iniciais é relevante no algoritmo k-Means, principalmente, por tal escolha interferir na indução do agrupamento final.

A Figura 4.1 ilustra 2 execuções do algoritmo k-Means (identificadas por 1 e 2 na figura), utilizando, em cada uma, o mesmo conjunto de 40 instâncias apresentado na Tabela 3.1 e tendo o valor de K estabelecido como 3. Em cada uma das execuções, entretanto, a inicialização do conjunto de centroides é feita randomicamente.

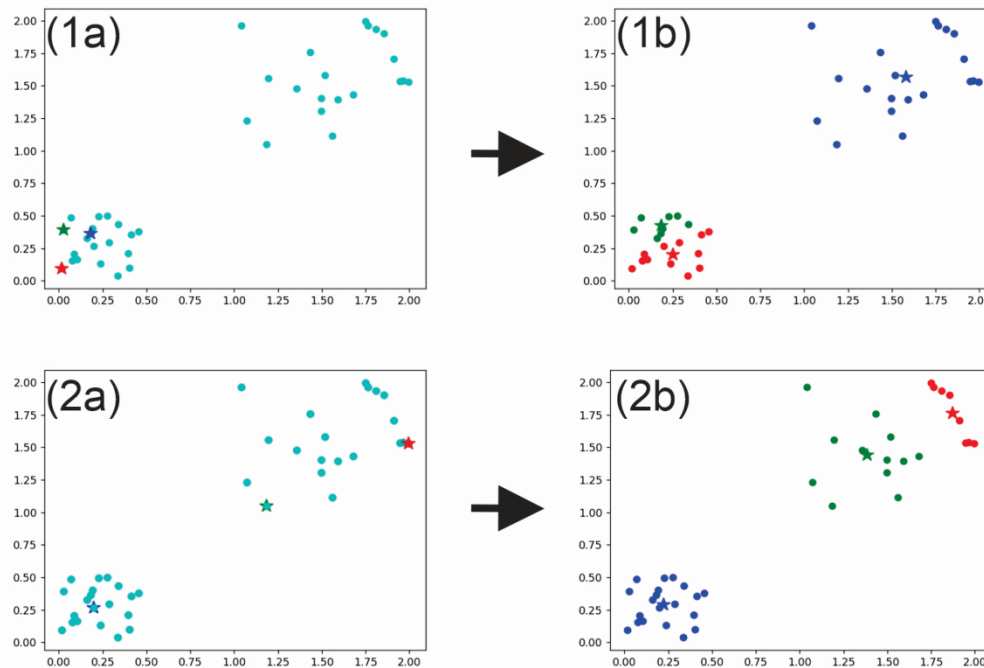


Figura 4.1 Representação de duas execuções do algoritmo k-Means, para K=3, indicadas por 1 e 2 na figura. Na execução 1, (1a) e (1b) indicam, respectivamente, a fase de inicialização randômica dos centroides e o agrupamento final obtido. Na execução 2, (2a) e (2b) indicam a fase de inicialização randômica dos centroides e o agrupamento final obtido.

O algoritmo k-Means é um algoritmo iterativo, que busca em cada iteração um melhor valor para a função SQE, encerrando a sua execução (condição de parada ou convergência), comumente, quando durante uma iteração, nenhuma ou poucas instâncias mudam dos grupos aos quais pertencem. Pode-se criar condições de parada levando em consideração as especificidades do conjunto de instâncias, cabendo ao usuário escolher a condição que melhor se adeque às suas necessidades (*e.g.*, número máximo de

iterações). O Algoritmo 3.1 apresenta um pseudocódigo simplificado do algoritmo k-Means.

```
procedure k-Means(C,K,G)
Input: C = {x1, x2, ..., xN}      %conjunto com N instâncias de dados a serem agrupadas
        K                          % número de grupos a serem criados
Output: G = {G1,G2,...GK} %agrupamento formado por K grupos induzidos a partir de C
begin
% Inicialização
% no passo (1) cada grupo é definido apenas pelo centroide
(1) escolha arbitrária de K instâncias do conjunto C, como centroides dos grupos G1,G2,...GK

% Indução do agrupamento G
(2) repeat
    (3) (re)atribuir cada instância xi ∈ C (i=1, ..., N) ao grupo cujo centroide que lhe
        seja mais próximo;
    (4) atualizar os centroides de cada grupo, como a média dos valores das suas instâncias
(5) until nenhuma alteração aconteça.
end.
return G = {G1,G2,...GK}
end_procedure
```

Algoritmo 4.1 Pseudocódigo simplificado do algoritmo k-Means.

O procedimento (1) é responsável por escolher de forma aleatória K instâncias para serem os centroides iniciais. O procedimento (4) é responsável por (re)calcular o centroide, baseado na média dos valores das instâncias atribuídas a cada um dos grupos G_k, k = 1, ..., K, conforme Eq. 4.1.

O processo iterativo do algoritmo k-Means continua até que haja a convergência. A convergência é um indicativo que os centroides estabilizaram, atingindo, assim, uma permanência. Ou seja, é alcançado o SQE mínimo local possível a partir dos centroides escolhidos inicialmente (*i.e.*, o melhor mínimo local).

O tempo execução do algoritmo k-Means sem otimização é da ordem de O(WNKM), em que W é o número de iterações necessárias para convergir, N é o número de instâncias do conjunto, K o número de grupos e M o número de atributos que descrevem cada instância. O número de iterações W pode variar dependendo da escolha dos centroides iniciais. Como, via de regra, K < N e W < N, o algoritmo k-Means é considerado relativamente escalável e eficiente no processamento de conjunto com grande volume de instâncias [Jain *et al.* 1999] [Jain 2010] [Oliveira 2018].

Como já informado anteriormente, a investigação conduzida nesta pesquisa tem como foco o uso da desigualdade triangular para melhorar o desempenho do algoritmo k-Means, de forma a diminuir o número de cálculos de distância entre instâncias e centroides realizados pelo algoritmo e, com isso, acelerar a sua execução.

4.3.1 Um Exemplo Didático do k-Means

No desenvolvimento deste exemplo foi utilizado o conjunto de instâncias C , demonstrados na Tabela 3.1, com 40 instâncias, em que cada instância é descrita por dois atributos numéricos, A_1 e A_2 , cujos valores foram artificialmente gerados. A Figura 4.2 ilustra as 40 instâncias, representadas como pontos, do conjunto em um gráfico de duas dimensões.

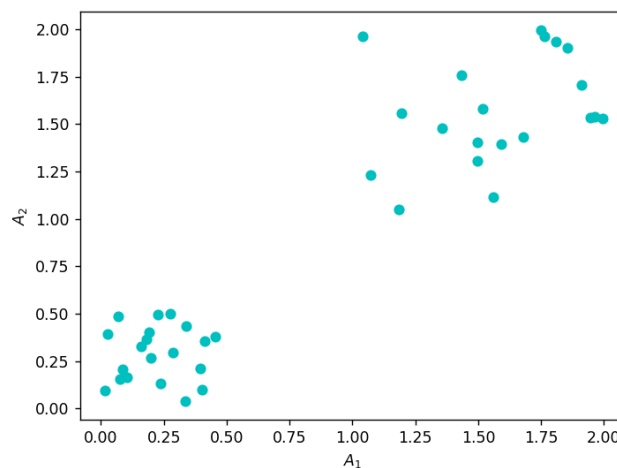


Figura 4.2 Representação das 40 instâncias do conjunto C da Tabela 3.1.

O código do algoritmo k-Means utilizado nesse experimento foi implementado na linguagem de programação Python 3.7 e, para a geração dos gráficos, foi utilizada a biblioteca matplotlib.

Iteração 1: Inicialmente, o k-Means escolhe aleatoriamente K ($K = 3$) instâncias do conjunto C como centroides iniciais, representados como estrelas no gráfico. A Figura 4.3 ilustra as instâncias escolhidas como centroides e para o *trace* de alto nível que segue, foi usada a distância euclidiana para calcular a similaridade entre instância e centroide.

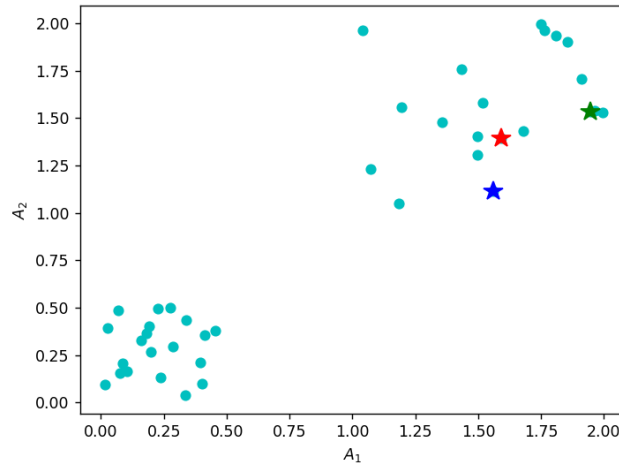


Figura 4.3 Representação dos 3 centroides iniciais escolhidos, identificados como estrelas na figura.

Iteração 2: Cada instância de C é atribuída ao grupo cujo centroide estiver mais próximo (maior similaridade) da instância, assumindo, assim, a mesma cor do centroide.

A Figura 4.4 ilustra os três centroides em suas posições iniciais e as instâncias que lhes foram atribuídas. As instâncias atribuídas a um centroide formam um grupo. Distingue-se pela cor (azul, verde e vermelho) o grupo ao qual pertencem as instâncias e os centroides.

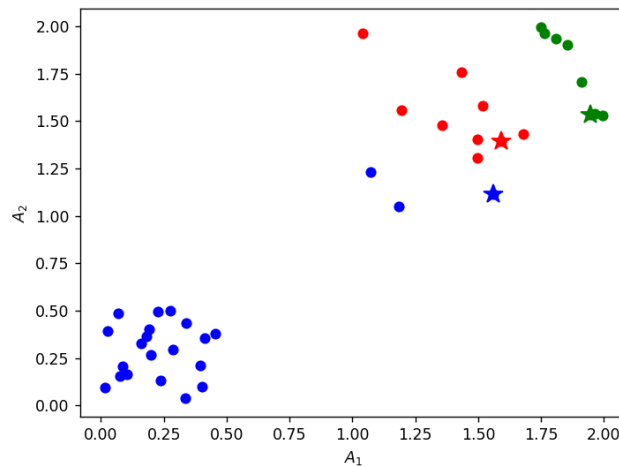


Figura 4.4 Representação da atribuição das instâncias aos centroides que lhes forem mais próximo.

Iteração 3: A atualização dos centroides é realizada recalculando, para cada grupo, a média dos valores das instâncias pertencentes ao grupo, conforme Eq. 4.1 e o processo

de atribuição das instâncias aos centroides que lhes são mais próximos é repetido, como pode ser visto na Figura 4.5.

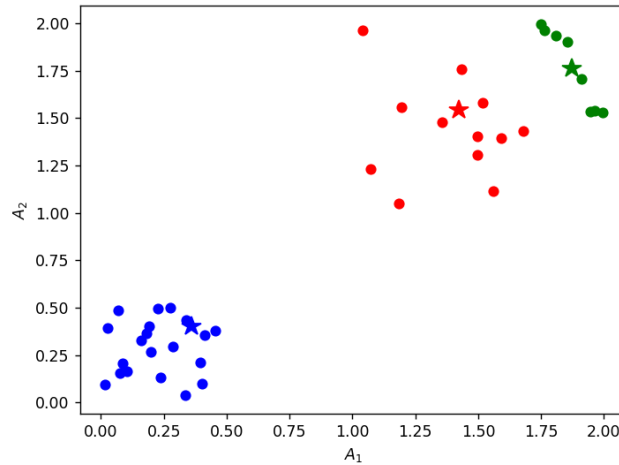


Figura 4.5 Representação da segunda iteração do algoritmo k-Means.

Iteração 4: As iterações do algoritmo k-Means (atualização dos centroides e atribuição das instâncias ao centroide mais próximo) ocorrem até que a SQE seja minimizada, que se reflete na situação em que as instâncias permanecem nos grupos em que estavam e não há recálculo de centroides, encerrando assim a execução do algoritmo, em que o agrupamento induzido é representado pelo conjunto dos últimos centroides obtidos. A Figura 4.6 ilustra o agrupamento final com a menor SQE possível, considerando os centroides inicialmente escolhidos.

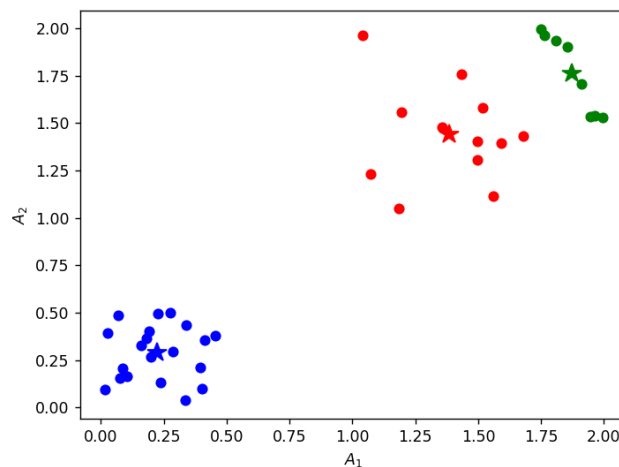


Figura 4.6 Iteração final, melhor SQE mínimo local.

4.4 Inicialização do k-Means (k-Means++)

Como comentado anteriormente, a performance dos algoritmos iterativos, inclusive a do k-Means, é altamente dependente dos centroides iniciais. Geralmente os centroides são escolhidos aleatoriamente, porém, escolher aleatoriamente pode gerar situações indesejáveis, como por exemplo, escolher instâncias com valores extremos ou centroides muito próximos um dos outros. Até o momento não se tem um método universalmente aceito para escolha dos centroides [Khan & Ahmad 2004]. Em [Celebi *et al.* 2013] é apresentado um estudo comparativo da eficiência dos principais métodos de inicialização para o k-Means. Como será abordado no Capítulo 5, em várias referências, algoritmos de inicialização de centroides, são também abordados como algoritmos para aceleração do k-Means.

Neste Projeto de Pesquisa o k-Means é abordado com duas inicializações distintas: (1) a inicialização *default* do k-Means original, que é feita via um processo de escolha aleatória dos K centroides iniciais e (2) a inicialização utilizada pelo algoritmo conhecido como k-Means++, proposto por Arthur e Vassilvitskii, em [Arthur & Vassilvitskii 2007]. O k-Means++ pode ser considerado como o k-Means original, com um processo de inicialização dos centroides diferenciado daquele utilizado pelo k-Means original, muito embora ainda envolva escolhas baseadas em aleatoriedade, como discutido em [Oliveira 2018].

O primeiro passo para inicialização dos centroides no k-Means++ ainda é escolher os centroides aleatoriamente, porém utiliza uma ponderação de acordo com o quadrado de sua distância ao centroide que lhe seja mais próximo. Algoritmo 4.2 apresenta um pseudocódigo do procedimento de inicialização do k-Means++ adaptado de [Oliveira 2018].

O processo de inicialização do k-Means++ escolhe aleatoriamente o primeiro centroide e, a partir do segundo centroide, os centroides são escolhidos a partir de instâncias com uma probabilidade proporcional à distância das instâncias aos centroides já definidos, essa probabilidade é dada pela função $\frac{d(x_i, c_k)^2}{\sum_{x_n \in CI} d(x_n, c_k)^2}$. Dessa forma, como o primeiro centroide ainda é escolhido de forma totalmente aleatória, deve-se realizar diferentes

execuções para obtenção de um bom conjunto de centroides e, conseqüentemente, um bom agrupamento.

```

procedure escolha_centroides_k-Means++(C,K,G)
input: C = {x1, x2, ..., xN} %conjunto com N instâncias de dados a serem agrupadas
      K % número de grupos a serem criados
output: G = {c1,c2,...,cK} % conjunto de centroides
begin
  c1 ← escolheCentroides(C,1) % escolhe aleatoriamente o primeiro centroide
  j ← 2
  while j ≤ K do
    soma ← 0
    for i ← 1 to N do
      begin
        d[i] ← (distanciaCentroideMaisProximo(xi,c))2
        soma ← soma+ dist[i] % soma das distâncias ao quadrado
      end
    for i ← 1 to N do
      begin
        p[i] ← d[i]/soma % p é um vetor de probabilidade
        pa[i] ← p[i] % pa é um vetor de probabilidade acumulada
      end
    i ← 2
    for i ← 1 to N do
      pa[i] ← pa[i-1] + p[i]
    s ← aleatorio([0,1]) % sorteio aleatório de um valor entre 0 e 1
    cj ← primeiraInstânciaComProbabilidadeAcumuladaMaiorQueS(pa,s)
    j ← j+1
  end while
return G
end_procedure

```

Algoritmo 4.2 Pseudocódigo do procedimento de inicialização do k-Means++, adaptado de [Oliveira 2018].

O procedimento *escolheCentroides* é responsável por escolher de forma aleatória uma instância para ser o centroide c_1 . O procedimento *distanciaCentroideMaisProximo* calcula a distância da instância até o centroide que lhe é mais próximo. O procedimento *aleatorio*([0,1]) lança de forma aleatória um valor entre 0 e 1. O procedimento *primeiraInstânciaComProbabilidadeAcumuladaMaiorQueS* varre o vetor pa que é um vetor de probabilidade acumulada das distâncias das instâncias ao centroide mais próximo, assim, os valores estão em ordem crescente, e escolhe a primeira instância que apresente valor de probabilidade acumulada maior que o valor lançado no procedimento *aleatorio*([0,1]).

Por fim, o procedimento retorna os centroides que foram escolhidos para serem usados como centroides iniciais do algoritmo k-Means.

Uma exemplificação didática do processo de escolha dos centroides iniciais utilizando a inicialização do k-Means++ pode ser encontrada em [Oliveira 2018].

Capítulo 5

Revisão da Literatura com Foco em Estratégias para Acelerar o k-Means

5.1 Considerações Iniciais

Na literatura podem ser evidenciados inúmeros trabalhos que propõem estratégias com o objetivo de acelerar o processamento do algoritmo k-Means. Entre alguns aspectos que eventualmente contribuem para o k-Means ter um processamento não muito rápido estão (1) número alto de instâncias a serem agrupadas; (2) número alto de cálculos de distâncias entre instâncias e centroides que o algoritmo deve realizar, um processo que é também influenciado pelo número de atributos que descrevem as instâncias e (3) número de iterações necessárias para a convergência.

Como comentado em [Hamerly & Drake 2015], os métodos básicos de aceleração de algoritmos de aprendizado de máquina podem ser categorizados em: (1) melhoramentos algorítmicos; (2) paralelização (incluindo *threading*, multiprocessamento e computação distribuída) e (3) aproximação.

Muitos pesquisadores têm investido na aceleração de algoritmos de agrupamento em geral. Como o trabalho descrito nesta dissertação tem por foco o algoritmo k-Means, o que segue é uma breve revisão bibliográfica de vários algoritmos encontrados na literatura, que foram propostos com o objetivo exclusivo de acelerar o processamento do k-Means e que podem ser caracterizados como melhoramentos algorítmicos. Parte do conteúdo deste capítulo encontra-se em [Matte & Nicoletti 2019].

5.2 Breve Revisão Bibliográfica

Hamerly e Drake, em [Hamerly & Drake 2015] comentam que para conjunto de instâncias descritas com um número pequeno de atributos (*i.e.*, tendo baixa dimensionalidade), a indexação das instâncias a serem agrupadas é uma maneira efetiva de acelerar o k-Means. Apesar dos trabalhos descritos em [Kanungo *et al.* 2002] e

[Pelleg & Moore 1999] terem sido feitos por pessoas distintas em épocas distintas, ambos são similares na maneira como propõem a adaptação de uma árvore k-d padrão [Bentley 1975] [Brown 2015] para promover uma aceleração do *k-Means*.

Como evidenciado nos experimentos descritos em [Pelleg & Moore 2001], o algoritmo conhecido como *Algoritmo Blacklisting* é efetivo para a indução de agrupamentos em conjuntos com um número elevado de instâncias. Os autores também comentam que o algoritmo se torna lento quando a dimensionalidade relativa ao número de atributos que descrevem as instâncias a serem agrupadas se torna maior do que 8. Em conjuntos de instâncias com alta dimensionalidade, instâncias e centroides tendem a ficar longe uns dos outros e pouca poda pode ser feita.

Outro aspecto importante a ser considerado, quando do uso de algoritmos que usam árvores k-d como estrutura para armazenamento das instâncias e de informações complementares, diz respeito aos custos envolvidos na construção e uso de tal estrutura. Para um conjunto com N instâncias de dados, o custo da construção de uma árvore k-d é da ordem de $O(N \log(N))$ e o custo de memória necessária é praticamente duplicado. Também, se o conjunto de instâncias sofre mudanças, a atualização da árvore k-d para refletir essas mudanças não é trivial.

Hamerly, em [Hamerly 2010], comenta que quando as instâncias a serem agrupadas têm alta dimensionalidade, esquemas de indexação, como o uso de árvores k-d, não funcionam bem, a ponto de o processo do exame de cada instância (*i.e.*, sem o uso de uma estrutura que favoreça a aceleração) ser bem mais rápido do que processos realizados por algoritmos que implementam aceleração para baixa dimensionalidade.

Em [Moore 2000] o autor descreve uma proposta de estrutura hierárquica, a hierarquia de âncoras, adequada para lidar com instâncias de dados com alta dimensionalidade, que satisfaz a desigualdade triangular. Similarmente ao uso de árvores k-d, a construção e manutenção dessa estrutura hierárquica podem ser complexas e demandam um investimento alto em tempo e memória.

Alguns trabalhos, ao invés de lidar diretamente com agrupamentos em conjuntos de instâncias com alta dimensionalidade, lidam com uma projeção desses conjuntos em um espaço com baixa dimensionalidade, e usam algoritmos adequados para agrupamento de

instâncias com baixa dimensionalidade. O método para redução de dimensionalidade conhecido como PCA (*Principal Component Analysis*) [Jolliffe 2002] é abordado em associação ao aprendizado não supervisionado realizado pelo k-Means, no trabalho descrito em [Ding & He 2004]. Os resultados obtidos no trabalho indicam que os processos de redução de dimensionalidade não-supervisionada e de aprendizado não supervisionado estão fortemente relacionados.

Algumas estratégias para a aceleração do k-Means podem também ser abordadas como estratégias para a inicialização do k-Means. Considerando que o efeito de uma inicialização adequada dos centroides promove a indução mais rápida de um agrupamento, métodos de inicialização de centroides podem, de certa forma, ser abordados como algoritmos de aceleração do k-Means.

Inicialmente Forgy, em [Forgy 1965], propôs a atribuição de cada instância a um dos K grupos de forma uniforme e aleatória, e os centroides são dados pela média das instâncias atribuídas ao grupo. Esse método não possui base teórica, pois não apresenta homogeneidade interna [Anderberg 1973] [Celebi *et al.* 2013].

Jancey, em [Jancey 1966], propôs considerar cada centroide como uma instância artificialmente criada dentro do espaço das instâncias. Esse método pode gerar centroides em regiões do espaço desprovidas de instâncias, cujo efeito colateral pode ser a indução incorreta de agrupamentos em que grupos podem ser vazios [Anderberg 1973].

MacQueen, em [MacQueen 1967], propôs duas abordagens, sendo que a primeira delas consiste na escolha das K instâncias iniciais do conjunto de instâncias. Essa abordagem, entretanto, é desvantajosa em algumas situações, devido à sua sensibilidade à posição das instâncias no conjunto. Por exemplo, em situações em que as instâncias se apresentam ordenadas no conjunto. Como segunda abordagem, MacQueen propôs a escolha de K instâncias aleatórias para serem consideradas como os centroides iniciais. Essa abordagem tende a escolher centroides em regiões mais densas, porém o processo que a implementa não trata a possibilidade da inconveniência da escolha de *outliers*. Como comentado em [Celebi *et al.* 2013], a segunda abordagem, de maneira equivocada, é comumente atribuída a Forgy.

O algoritmo Furthest-First, proposto por Hochbaum e Shmoys, em [Hochbaum & Shmoys 1985], pode ser empregado na fase de inicialização do k-Means. O Furthest-First começa escolhendo randomicamente uma instância como o primeiro centroide e, então, repetidamente seleciona, como o próximo centroide, a instância que está mais longe de qualquer centroide já escolhido. O algoritmo, apesar de simples e de ser facilmente implementável, tem a tendência de escolher *outliers*, como parte do conjunto inicial de centroides, dada a tendência de *outliers* estarem situados mais nas fronteiras das regiões definidas pelas instâncias.

Bradley e Fayyad em [Bradley & Fayyad 1998] apresentam um procedimento para calcular um refinamento do processo de inicialização de centroides, que permite que o algoritmo iterativo convirja para um melhor mínimo local. O procedimento em questão pode ser usado agregado a um grande número de algoritmos de agrupamento, tanto para dados discretos quanto contínuos. Os resultados obtidos e discutidos no trabalho evidenciam que o conjunto inicial de centroides, quando refinado pelo procedimento proposto e usado com o k-Means, provoca um melhoramento nos resultados do k-Means.

A proposta conhecida como k-Means++ de Arthur e Vassilvitskii, em [Arthur & Vassilvitskii 2007], sugere a escolha dos centroides iniciais aleatoriamente, porém, utilizando de uma ponderação de acordo com o quadrado de sua distância ao centroide que lhe seja mais próximo. A proposta de Arthur e Vassilvitskii é uma das principais formas adotadas para escolha dos centroides iniciais, pois gera bons ganhos de eficiência [Celebi *et al.* 2013]. A Seção 4.4 dessa Dissertação apresenta, em linhas gerais, o funcionamento do k-Means++.

Celebi e coautores em [Celebi *et al.* 2013] apresentam, em ordem cronológica, um breve histórico dos métodos mais comuns de inicialização de centroides. Um estudo empírico sobre a contribuição de estratégias de inicialização do k-Means com vistas à diminuição do número de iterações do algoritmo e, consequentemente, acelerando-o, pode ser visto em [Oliveira & Nicoletti 2018].

Algumas propostas para acelerar o k-Means se utilizam de uma propriedade que tem origem na geometria euclidiana, conhecida como desigualdade triangular, para acelerar o

tempo de execução do algoritmo. A adequação do k-Means para o emprego dessa propriedade permite que muitos cálculos realizados pelo algoritmo possam ser evitados e, com isso, o seu tempo de execução é acelerado. O Capítulo 6 desta Dissertação apresenta e discute em detalhes a propriedade da desigualdade triangular e como é empregada.

Phillips, em [Phillips 2002], propôs duas modificações relativamente simples para acelerar o k-Means, que são conhecidas como algoritmo compare-Means e algoritmo sort-Means. Ambos os algoritmos foram objeto de estudo/pesquisa durante o desenvolvimento do trabalho realizado e são abordados em detalhes na Seção 7.3 e Seção 7.4 desta Dissertação, respectivamente.

Elkan, em [Elkan 2003], propôs uma estratégia a ser agregada ao algoritmo k-Means, que faz uso da desigualdade triangular para monitorar os limites superiores e inferiores da distância entre as instâncias e centroides, com vistas à aceleração do tempo de execução do algoritmo. O k-Means assim modificado foi nomeado nesta Dissertação como k-Means-Elkan e é abordado em detalhes na Seção 7.5.

O algoritmo proposto por Hamerly, em [Hamerly 2010], e nomeado nesta Dissertação como k-Means-Hamerly, é considerado por seu autor não apenas como uma modificação do k-Means-Elkan mas, também, como uma simplificação do k-Means-Elkan. Assim como o k-Means-Elkan, o k-Means-Hamerly usa limites para as distâncias, que são eficientemente atualizados, assim como usa a desigualdade triangular para evitar cálculos de distâncias entre instâncias e centroides.

O k-Means-Hamerly emprega dois limites de distância, por instância de dados, para seus dois centroides mais próximos. Um deles é um limite superior na distância da instância ao seu centroide mais próximo e o outro é um limite inferior na distância da instância ao seu segundo centroide mais próximo. Com base nos resultados obtidos dos experimentos descritos em [Hamerly 2010], o autor comenta que o k-Means-Hamerly teve melhor desempenho em conjuntos de dados com dimensionalidade entre pequena e moderada (até 50 atributos), enquanto o k-Means-Elkan teve melhor desempenho em dados com alta dimensionalidade. De uma certa maneira esses dois algoritmos podem ser abordados

como complementares um do outro, sendo a escolha de um deles sempre dependente da dimensionalidade das instâncias de dados a serem agrupadas.

Drake e Hamerly, em [Drake & Hamerly 2012], combinaram os pontos fortes das propostas de Elkan e Hamerly em uma nova proposta, que faz uso dos b centroides mais próximos para gerar os limites inferiores, sendo que o valor de b é encontrado por uma estratégia de ajuste adaptativo. Dessa forma a nova proposta possibilita amenizar o custo de espaço utilizado pela estratégia de Elkan e utilizar os limites inferiores de forma mais consistente.

A estratégia proposta por Ding e coautores em [Ding *et al.* 2015], nomeada nesta Dissertação como k-Means-Yinyang, adota os conceitos de limite inferior e limite superior propostos por Elkan e, com base neles, propõe um filtro com dois níveis, que busca detectar cálculos desnecessários, contribuindo assim para acelerar a fase de atribuição de instâncias aos grupos. Outra característica importante do k-Means-YinYang é a de viabilizar a criação de grupos de centroides, permitindo, dessa forma, que os limites inferiores não sejam baseados na distância aos centroides, mas sim, aos grupos de centroides. Tal artifício traz, quando comparado ao k-Means-Elkan, economia no espaço de memória utilizado durante a execução do algoritmo. Ding e coautores também propõem a aceleração do método de cálculo da atualização da posição dos centroides a cada iteração. O k-Means-Yinyang é abordado em detalhes na Seção 7.6.

Por fim, a estratégia proposta por Yu e Dai, em [Yu & Dai 2017], nomeada nessa Dissertação como k-Means-Fission-Fusion, adota limites superiores e inferiores de grupos e subgrupos de instâncias até os centroides e utiliza filtros de grupos e subgrupos de instâncias, dessa forma evita cálculos de distâncias entre grupo ou subgrupo de instâncias a um determinado centroide. O k-Means-Fission-Fusion é abordado em detalhes na Seção 7.7.

Capítulo 6

O Uso da Desigualdade Triangular para Acelerar o Algoritmo k-Means

6.1 Considerações Iniciais

Este capítulo discute o uso da chamada desigualdade triangular, como uma maneira de acelerar a execução do algoritmo k-Means. Tal uso permite que muitos dos cálculos realizados pelo algoritmo, durante a indução do agrupamento, possam ser evitados e, com isso, o tempo de processamento na indução do agrupamento final diminui. Na Seção 6.2 são apresentadas conceituações relacionadas com a desigualdade triangular e resultados matemáticos que a envolvem.

6.2 Sobre a Desigualdade Triangular e Resultados Teóricos Associados ao Seu Emprego

O objetivo do uso da propriedade conhecida como *desigualdade triangular*, como comentam os autores que a utilizam para acelerar o k-Means padrão, se deve ao fato que o emprego de tal propriedade permite que muitos dos cálculos de distância, realizados pelo k-Means padrão, possam ser evitados, o que promove uma aceleração do processo de indução do agrupamento desejado.

Seja X um conjunto. Uma métrica em X é uma função (chamada *função distância* ou apenas *distância*) definida como $d: X \times X \rightarrow [0, \infty)$ e para todo x, y e $z \in X$, as seguintes propriedades são satisfeitas:

- (1) $d(x, y) \geq 0$;
- (2) $d(x, y) = 0 \leftrightarrow x = y$;
- (3) $d(x, y) = d(y, x)$;
- (4) $d(x, z) \leq d(x, y) + d(y, z)$ (*desigualdade triangular*).

Uma maneira de promover a visualização da desigualdade triangular é recorrer à Geometria, como mostra a Figura 6.1. Geometricamente, o lado direito da desigualdade

estabelece que a soma dos comprimentos de quaisquer dois lados de um triângulo qualquer for maior ou igual ao comprimento do lado restante.

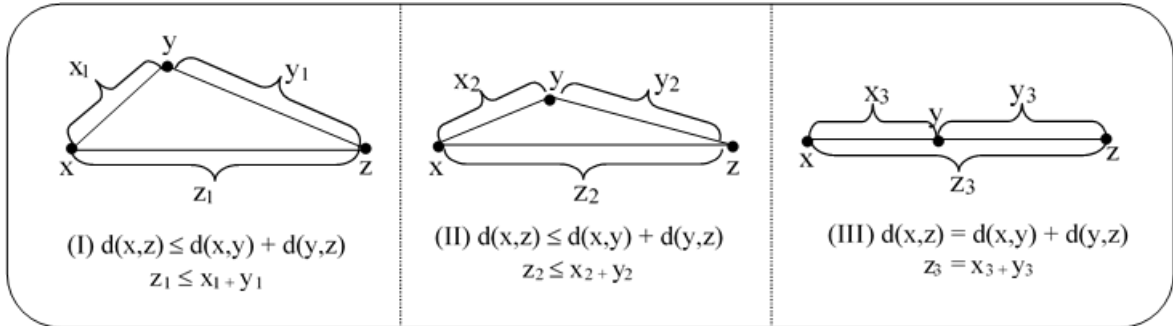


Figura 6.1 Três situações que evidenciam a desigualdade triangular. Particularmente, em (III) a desigualdade se torna uma igualdade, considerando que o triângulo é um triângulo degenerado (*i.e.*, seus três vértices são colineares).

Note que na Figura 6.1, a expressão do lado esquerdo da desigualdade triangular está sempre identificando a maior distância, dentre as três distâncias mostradas. Se a maior distância dentre as três é menor ou igual à soma das outras duas, as outras duas possíveis desigualdades triangulares são triviais *i.e.*, (I) $x_1 \leq y_1 + z_1$ e $y_1 \leq x_1 + z_1$ (II) $x_2 \leq y_2 + z_2$ e $y_2 \leq x_2 + z_2$ e (III) $x_3 \leq y_3 + z_3$ e $y_3 \leq x_3 + z_3$.

Como comenta Elkan, em [Elkan 2003], a dificuldade no uso da desigualdade triangular está no fato de tal desigualdade estabelecer um limite superior, mas não inferior. Abordando o problema de maneira simplista, considere que x seja uma instância e b e c sejam centroides.

Se de alguma forma for possível garantir que $d(x,c) \geq d(x,b)$, os resultados teóricos discutidos a seguir mostram que não é preciso, efetivamente, calcular o valor de $d(x,c)$. situação evidenciada no diagrama da Figura 6.2. Os dois lemas enunciados e provados a seguir tratam desse assunto.

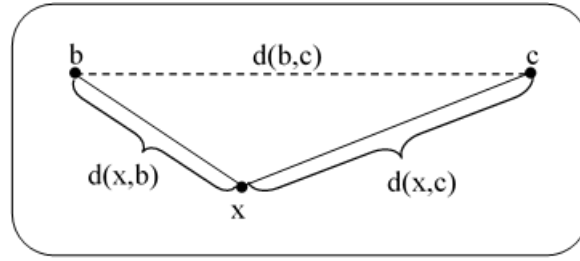


Figura 6.2 Diagrama exibindo dois centroides (b e c) e uma instância de dado x em que $d(x,b)$ e $d(x,c)$ são as distâncias da instância x a cada um dos centroides e a linha pontilhada marca a distância entre os dois centroides.

Lema 1. Seja x uma instância e sejam b e c dois centroides. Se $d(b,c) \geq 2d(x,b)$ então $d(x,c) \geq d(x,b)$.

Prova

Hipótese: x é uma instância e b, c são centroides e $d(b,c) \geq 2d(x,b)$.

Tese: $d(x,c) \geq d(x,b)$.

Sabe-se, pela desigualdade triangular, que

$$d(b,c) \leq d(b,x) + d(x,c) \text{ que pode ser reescrita como}$$

$$d(b,c) - d(b,x) \leq d(x,c), \text{ que pode ser reescrita como}$$

$$d(x,c) \geq d(b,c) - d(b,x) (*)$$

Considere agora o lado direito da expressão acima, que é:

$$d(b,c) - d(b,x)$$

e, considerando a hipótese, que assume $d(b,c) \geq 2d(x,b)$, a expressão (*) se reduz a

$$d(x,c) \geq d(b,c) - d(b,x) \geq 2d(x,b) - d(b,x) = d(x,b) \text{ e, portanto, tem-se}$$

$$d(x,c) \geq d(x,b)$$

Lema 2. Seja x uma instância e sejam b e c dois centroides. Então $d(x,c) \geq \max\{0, d(x,b) - d(b,c)\}$.

Prova Sabe-se que $d(x,b) \leq d(x,c) + d(b,c)$. Reescrevendo a desigualdade tem-se $d(x,c) + d(b,c) \geq d(x,b)$ e, portanto $d(x,c) \geq d(x,b) - d(b,c)$. Como $d(x,c)$ é uma distância, $d(x,c) \geq 0$.

Como apontado em [Elkan, 2003] o Lema 1 pode ser usado como descrito a seguir. Considere uma instância x e considere que c seja o centroide do grupo ao qual a instância x pertence, em um determinado momento. Considere também, um outro

centroide c^* . O Lema 1 estabelece que se $d(c, c^*) \geq 2d(x, c)$ ou seja, $1/2 d(c, c^*) \geq d(x, c)$, então, $d(x, c^*) \geq d(x, c)$. Neste caso, não é necessário calcular $d(x, c^*)$.

Considere o exemplo mostrado na Figura 6.3, em que os centroides considerados são $c = (5, 8)$ e $c^* = (13, 5)$ e $x = (6, 5)$ pertence ao grupo cujo centroide é c , delineado com linha pontilhada na figura.

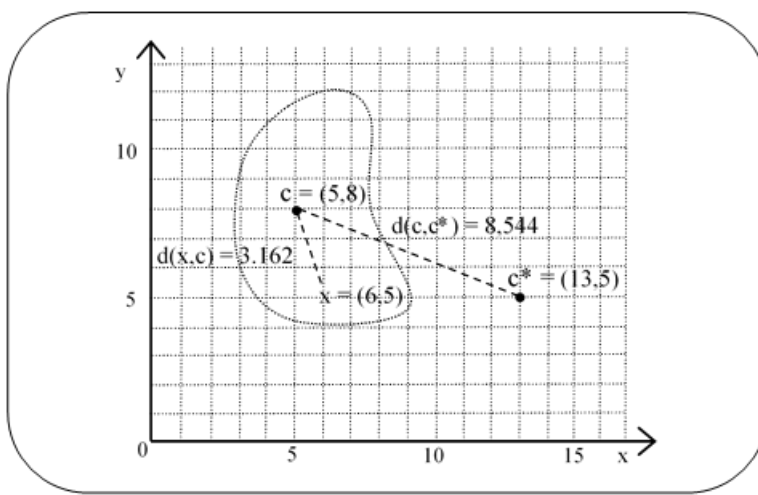


Figura 6.3 Diagrama exibindo dois centroides, $c = (5, 8)$ e $c^* = (13, 5)$ e uma instância de dado x que pertence ao grupo cujo centroide é $c = (5, 8)$. Como $1/2 \times d(c, c^*) \geq d(x, c)$ i.e., $1/2 \times 8,544 \geq 3,162$, é desnecessário calcular $d(x, c^*)$, uma vez que $d(x, c^*)$, garantidamente (com base no Lema 1), será maior que $d(x, c)$.

Tem-se pois que $d(c, c^*) = \text{SQRT}((5-13)^2 + (8-5)^2) = \text{SQRT}(64 + 9) = \text{SQRT}(73) = 8,544$ e que $d(x, c) = \text{SQRT}((6-5)^2 + (8-5)^2) = \text{SQRT}(1^2 + 3^2) = \text{SQRT}(10) = 3.162$. O Lema 1 estabelece que se $1/2 \times d(c, c^*) \geq d(x, c)$, então $d(x, c^*) \geq d(x, c)$, o que é o caso nesse exemplo uma vez que $1/2 \times 8,544 \geq 3.162$ i.e., $4,272 \geq 3,162$. No caso $d(x, c^*) = \text{SQRT}((6-13)^2 + (5-5)^2) = \text{SQRT}(7^2) = 7$.

Considerando que o número de instâncias de dados a serem agrupadas é N , que K seja o número de grupos a serem criados e que W representa o número de iterações necessárias para o algoritmo convergir, a complexidade em tempo do k-Means padrão é $O(NKW)$.

O número de cálculos de distância realizados pelo k-Means é dado pelo produto NKW . A principal contribuição no uso da propriedade da desigualdade triangular para acelerar o k-Means padrão, como apontado em [Elkan 2003], está na diminuição do número de cálculos de distâncias que, na prática, passa a estar mais perto de N do que de NKW .

Entretanto, é preciso que o algoritmo acelerado, k-Means_AC, satisfaça três propriedades:

- (1) deve ser capaz de começar o processamento a partir de um grupo arbitrário de K centroides (de maneira que todos os métodos de inicialização possam continuar a ser usados);
- (2) se um mesmo conjunto inicial de centroides for utilizado, o k-Means_AC deve sempre induzir os mesmos centroides finais, como acontece com o k-Means padrão e
- (3) deve ser capaz de usar qualquer métrica de distância (*i.e.*, não deve se restringir à otimização específica, por exemplo, da distância euclidiana). A condição (3), particularmente, é importante uma vez que muitas aplicações usam uma métrica de distância específica ao domínio de dados da aplicação. Considere a notação e definições que seguem.

O Capítulo 7 dessa Dissertação apresenta cinco estratégias que utilizam a desigualdade triangular para acelerar a execução do algoritmo k-Means, além de trazer detalhes de cada uma das estratégias utilizadas.

Capítulo 7

Estratégias de Aceleração do Algoritmo k-Means

7.1 Considerações Iniciais

Este capítulo discute maneiras de acelerar a execução do algoritmo k-Means. O uso da desigualdade triangular, como comentado no Capítulo 6, permite que muitos dos cálculos realizados pelo algoritmo, durante a indução do agrupamento, possam ser evitados e, com isso, o tempo de processamento na indução do agrupamento final diminui. Como pode ser confirmado na literatura, tal desigualdade foi empregada em vários algoritmos. Na pesquisa realizada que está descrita nesta Dissertação foram objeto de estudo e pesquisa os seguintes algoritmos: (1) *compare-Means* e (2) *sort-Means*, ambos propostos por Phillips em [Phillips 2002] e descritos na Seção 7.3 e Seção 7.4, respectivamente, (3) *k-Means-Elkan*, proposto em [Elkan 2003] e tratado na Seção 7.5, (4) *k-Means-Yinyang*, proposto por Ding e coautores, em [Ding *et al.* 2015], e discutido na Seção 7.6 e (5) *e o k-Means-Fission-Fusion* proposto por Yu e Dai, em [Yu & Dai 2017], abordado na Seção 7.7.

7.2 Estratégias Para Acelerar o k-Means

Esta Dissertação descreve cinco estratégias para acelerar o processamento do algoritmo k-Means. O uso de tais estratégias não modifica o resultado obtido pelo algoritmo, ou seja, o k-Means com ou sem o uso de qualquer uma das estratégias, usando, entretanto, os mesmos centroides iniciais, induz o mesmo agrupamento.

A primeira delas é implementada pelo algoritmo *compare-Means* e a segunda, pelo algoritmo *sort-Means*, ambos propostos por Phillips, em [Phillips 2002]. Como terceira estratégia é apresentada a proposta de Elkan [Elkan 2003], que se aproveita de limites superiores e inferiores para induzir o agrupamento e evitar cálculos desnecessários de distâncias. Como quarta estratégia é apresentada a proposta feita por Ding e coautores em [Ding *et al.* 2015], nomeada nessa Dissertação como *k-Means-Yinyang*. Essa

proposta cria grupos de centroides e se aproveita dos limites superiores e inferiores, descritos por Elkan, para propor um filtro de 2 níveis (Global/Grupos e Local), que permite evitar cálculos de distâncias. Por fim é apresentada a estratégia proposta por Yu e Dai, nomeada nessa Dissertação k-Means-Fission-Fusion que, inspirado no k-Means-Yinyang, utiliza de grupos e subgrupos de instâncias, de forma a utilizar limites superiores e inferiores para verificar a necessidade de realizar o cálculo de distância entre instância e centroide.

Para as subseções que seguem, considere novamente um conjunto C contendo N instâncias de dados M -dimensionais a serem agrupadas, e o uso do k-Means para induzir um agrupamento de C em K grupos, em que cada centroide de grupo é notado por c_j , $j = 1, \dots, K$ e o grupo ao qual uma instância $x \in C$ pertence, tem seu centroide notado por $c(x_i)$. O k-Means proposto por Lloyd descrito no Capítulo 4, neste Capítulo é identificado como k-Means padrão.

7.3 O Algoritmo *Compare-Means*

Quando da execução do k-Means, o cálculo da distância de cada instância M -dimensional de dado pertencente a C , a cada um dos centroides c_j , $j = 1, \dots, K$ resulta em $O(NK)$ comparações e um tempo de execução da ordem de $O(NMK)$. O algoritmo compare-Means [Phillips 2002] é uma abordagem simples para evitar muitas das comparações que, provavelmente, são desnecessárias. O algoritmo evita comparações desnecessárias entre distâncias entre instâncias de dados e centroides, mas usa comparações de distâncias entre centroides. Inicialmente são calculadas todas as distâncias entre os K centroides, e uma matriz quadrada de ordem $K \times K$, em que cada elemento $d(c_i, c_j) = distancia(c_i, c_j)$, $i, j = 1, \dots, K$, é construída. Tal matriz é notada por d . A Figura 7.1 mostra uma situação em que, por meio do uso da desigualdade triangular, é possível evitar cálculos desnecessários. Na figura, c_i e c_j são centroides e x é uma instância. O Algoritmo 7.1 apresenta o pseudocódigo do algoritmo compare-Means proposto por Phillips.

```

procedure compare-Means(C,K,G)
input: C = {x1, x2, ..., xN}      % conjunto com N instâncias de dados a serem agrupadas
      K                          % número de agrupamentos
output: G = {G1, G2, ..., GK}    % agrupamento final
begin
  % fase de inicialização
  {c1, c2, ..., cK} ← escolha_inicial_centroides(C) % um centróide ci é escolhido para cada
                                                    % grupo Gi (i=1, ..., K)

  % todas as instâncias são atribuídas ao grupo G1
  G1 ← { }
  for i ← 1 to N do G1 ← G1 ∪ {xi}
  % fase iterativa
  repeat
    % determine a distância entre cada par de centroides e construa a matriz dK×K
    for i ← 1 to K do
      for j ← i+1 to K do d(ci,cj) ← distancia(ci,cj) %distância euclidiana
    call torne_matriz_simetrica (d)
    for i ← 1 to N do
      begin
        mindist ← distancia(xi,c(xi))
        for j ← 1 to K do
          if d(c(xi),cj) >= 2×mindist or c(xi) = cj then
            else begin
              dist ← distancia(xi,cj)
              if dist < mindist then
                begin
                  mindist ← dist
                  atualiza_centroide(xi,cj)
                end
              end
            end
          end
        % Atribuindo instâncias ao grupo por meio da identificação de seus centroides associados
        for i ← 1 to K do Gi ← { }
        for i ← 1 to N do
          begin
            c ← indice_centroide(xi) % índice associado ao centróide de uma instância
            Gc ← Gc ∪ {xi}
          end
        % Atualização dos centroides de grupo
        for i ← 1 to K do ci ← media_das_instancias_atribuidas(Gi)
      until centroides não mudam
    end
  return G = {G1, G2, ..., GK}
end procedure

```

Algoritmo 7.1 Pseudocódigo do k-Means implementando a estratégia compare-Means, proposto em [Phillip 2002].

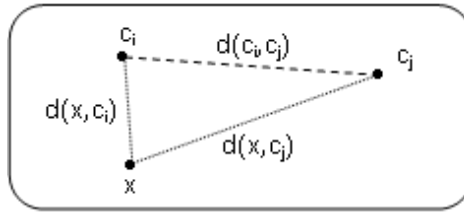


Figura 7.1 Situação em que o uso da desigualdade triangular ajuda a diminuir o número de cálculos (para realocação de instâncias aos novos centroides obtidos) a serem realizados.

Considerando a situação da Figura 7.1, e usando a desigualdade triangular, tem-se:

$$d(c_i, c_j) \leq d(x, c_i) + d(x, c_j) \text{ e, então}$$

$$d(x, c_j) \geq d(c_i, c_j) - d(x, c_i)$$

Portanto, se $d(c_i, c_j) \geq 2 \times d(x, c_i)$ (*), pode-se concluir que $d(x, c_j) \geq d(x, c_i)$, sem que $d(x, c_j)$ tenha sido calculada. A condição (*) sendo verdadeira para um determinado valor de j , faz com que o cálculo de $d(x, c_j)$ seja desnecessário, uma vez que a instância em questão está mais distante do centroide c_j , do que do centroide c_i .

Antes de cada iteração o algoritmo compare-Means calcula as distâncias entre pares de centroides, $d(c_i, c_j)$, $i, j = 1, \dots, K$ e $i \neq j$. Antes de calcular a distância de uma instância x a um centroide c_j , é realizado o teste (*), usando o centroide mais próximo de x , isto é, o centroide do grupo ao qual x foi atribuído.

7.3.1 Um Exemplo Completo do Uso do Compare-Means

Para o exemplo foi utilizado o conjunto de instâncias C na Tabela 7.1, com 10 instâncias, representadas como pontos no gráfico, em que cada instância é descrita por dois atributos numéricos, A_1 e A_2 , cujos valores foram artificialmente gerados. O conjunto C é ilustrado na Figura 7.2, de forma a permitir sua visualização gráfica, com o objetivo de promover o entendimento do processo.

Inicialmente o compare-Means escolhe aleatoriamente K ($K = 3$) instâncias do conjunto C como centroides iniciais, representados como estrelas no gráfico mostrado na Figura 7.3. Para o *trace* de alto nível que segue, foi usada a distância euclidiana para calcular a similaridade entre instância-centroide e centroide-centroide. Todas as instâncias são, inicialmente, atribuídas ao mesmo grupo que, no exemplo, é o G_1 .

Tabela 7.1 Conjunto C contendo 10 instâncias geradas artificialmente, descritas por dois atributos numéricos.

#ID	A ₁	A ₂	#ID	A ₁	A ₂
1	1	3	6	7	8
2	2	5	7	8	7
3	5	4	8	9	8
4	6	3	9	9	9
5	6	1	10	2	1

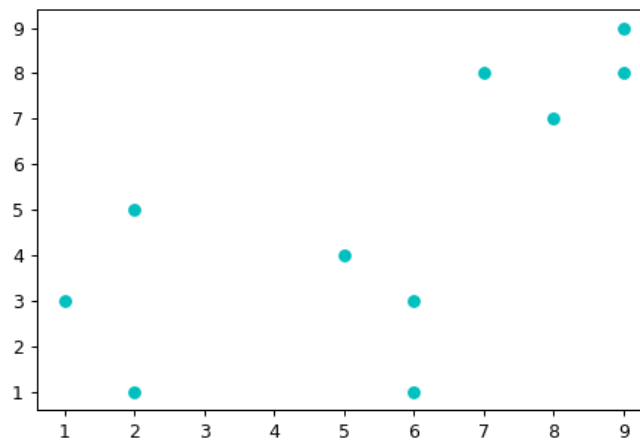


Figura 7.2 Representação das 10 instâncias do conjunto C apresentadas na Tabela 7.1.

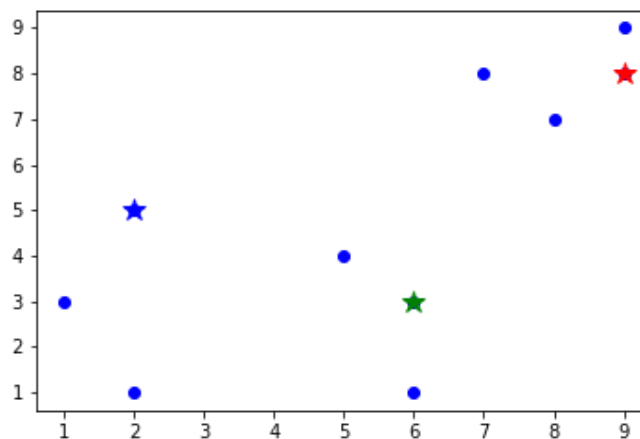


Figura 7.3 Representação dos 3 centroides iniciais $c_1 = (2,5)$, $c_2 = (6,3)$ e $c_3 = (9,8)$ representados como estrelas azul, verde e vermelho, respectivamente. As instâncias estão atribuídas ao grupo G_1 , ilustrado como pontos de cor azul.

Iteração 1: iniciando a parte iterativa do algoritmo compare-Means é calculada a matriz de distância $d_{K \times K}$ entre os centroides, com custo de $(K*(K-1))/2$. Para o exemplo foram realizados 3 cálculos de distância entre os centroides. A Tabela 7.2 apresenta os valores da matriz de distância d para o exemplo em desenvolvimento.

Tabela 7.2 Matriz de distância d da iteração 1 do algoritmo compare-Means

#ID	1	2	3
1	0	4,47	7,61
2		0	5,83
3			0

Após a construção da matriz de distância d , são identificadas as instâncias que satisfazem a condição $d(c_j, c(x_i)) > 2 \times d(x_i, c(x_i))$ ou $c_j = c(x_i)$, de forma a evitar o cálculo de distância entre a instância x_i e o centroide c_j . No total foram evitados 7 cálculos de distância entre instâncias e centroides, de um total de 30 que seria realizado pelo k-Means padrão na iteração 1. A Tabela 7.3 apresenta a lista dos cálculos de distâncias que foram evitados. Dessa forma foram realizados um total 26 cálculos de distância na iteração. A Figura 7.4 ilustra o novo agrupamento induzido na iteração 1. No fim da iteração são recalculadas as posições dos centroides, a partir das médias das instâncias atribuídas ao grupo G_j .

Tabela 7.3 Cálculos de distância entre as instâncias e os centroides que foram evitados na iteração 1.

#ID	1	2	3
1		×	×
2		×	×
3			×
4			×
5			×
6			
7			
8			
9			
10			

Iteração 2: No início da segunda iteração a matriz de distância d é recalculada para as novas posições dos centroides. A Tabela 7.4 apresenta a distância entre os centroides.

Tabela 7.4 Matriz de distância d da iteração 2 do algoritmo compare-Means.

#ID	1	2	3
1	0	4,01	8,26
2		0	5,92
3			0

Na sequência do processo são identificadas quais instâncias satisfazem a condição $d(c_j, c(x_i)) > 2 \times d(x_i, c(x_i))$ ou $c_j = c(x_i)$, de tal forma que os cálculos de distância entre as instâncias e os centroides sejam evitados. A Tabela 7.5 apresenta os cálculos que foram

evitados na iteração 2. No total foram evitados 18 cálculos de distância entre instâncias e centroides, de um total de 30 que seriam realizados no k-Means padrão, ou seja, foram realizados 12 cálculos de distância entre instâncias e centroides, com um total de 15 cálculos realizados na iteração. Como não houve mudança no agrupamento gerado na iteração atual, o algoritmo compare-Means finaliza. A Figura 7.5 apresenta o agrupamento final produzido.

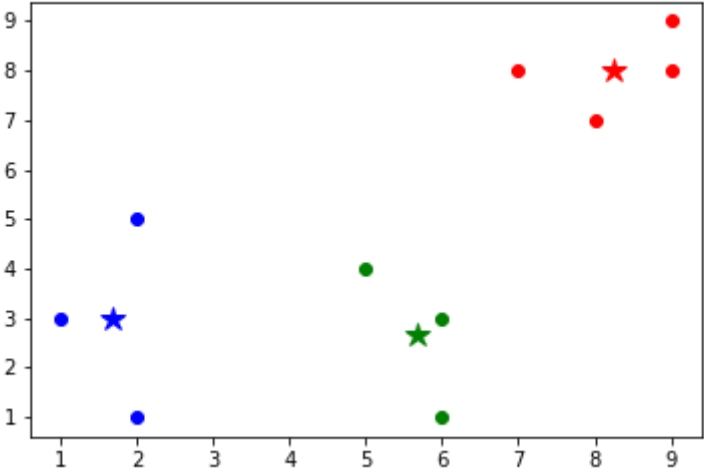


Figura 7.4 Representação do agrupamento formado após a iteração 1 do algoritmo compare-Means, $G_1=\{1,2,10\}$, $G_2=\{3,4,5\}$ e $G_3=\{6,7,8,9\}$.

Tabela 7.5 Cálculos de distância entre as instâncias e os centroides que foram evitados na iteração 2.

#ID	1	2	3
1		x	x
2			x
3	x		x
4	x		x
5	x		x
6	x	x	
7	x	x	
8	x	x	
9	x	x	
10			x

Para o exemplo desenvolvido foram realizados 41 cálculos de distância (6 de centroide–centroide e 35 de instância–centroide). Tal número seria 60, caso o k-Means padrão tivesse sido usado.

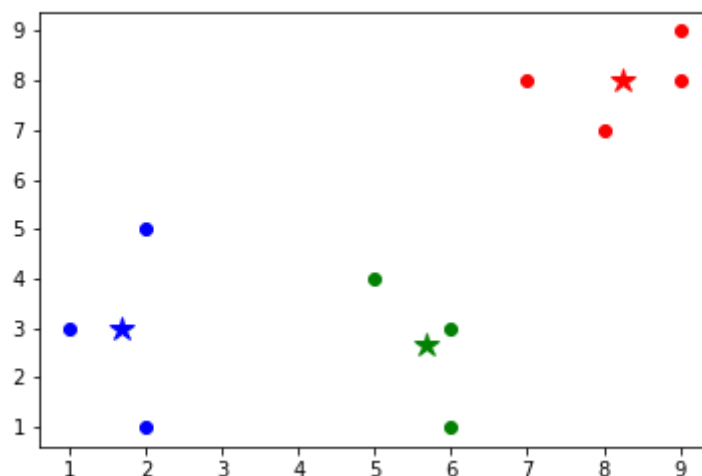


Figura 7.5 Representação o agrupamento final gerado pelo algoritmo compare-Means, $G_1=\{1,2,10\}$, $G_2=\{3,4,5\}$ e $G_3=\{6,7,8,9\}$.

Um fator importante a ser levado em consideração é que, em geral, quanto maior for o número de iterações realizadas pelo algoritmo, maior é o número de cálculos de distâncias que são evitados, por aumentar, a cada iteração, a probabilidade de as instâncias estarem corretamente atribuídas aos grupos, essa característica é comum as demais estratégias de aceleração, objetos de estudo na pesquisa realizada que está descrita nesta Dissertação.

7.4 O Algoritmo Sort-Means

Um outro melhoramento do tempo de execução do k-Means pode ser implementado por meio da estratégia utilizada pelo algoritmo *sort-Means* [Phillips 2002]. O pseudocódigo do algoritmo *sort-Means* está descrito em Algoritmo 7.2. Inicialmente são calculadas todas as distâncias entre os K centroides, e uma matriz quadrada de ordem $K \times K$, em que cada elemento $d(c_i, c_j) = distancia(c_i, c_j)$, $i, j = 1, \dots, K$ e $i \neq j$, é construída. Tal matriz é notada por d . Essa matriz é usada para a construção de outra matriz, a matriz $O_{K \times K}$, na qual a linha i é uma permutação de $1, \dots, K$, representando os grupos em ordem crescente de distâncias dos respectivos centroides, ao centroide i , para $i=1, \dots, K$.

O processo fica mais claro se explicado por meio de um exemplo. Considere uma situação envolvendo quatro centroides, cada um deles representado por um dos quatro

pontos mostrados na Figura 7.6, isto é, $c_1 = (4,2)$, $c_2 = (6,7)$, $c_3 = (12,9)$ e $c_4 = (13,3)$. Considere ainda as distâncias entre os centroides como mostra a Tabela 7.6.

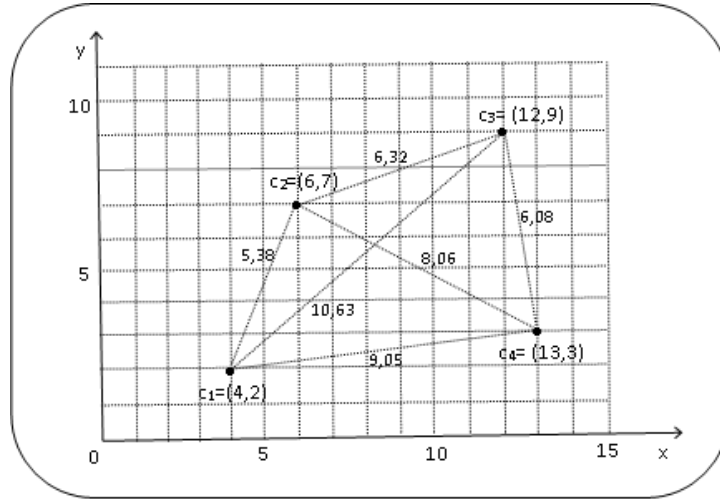


Figura 7.6 Diagrama exibindo quatro centroides, $c_1 = (4,2)$, $c_2 = (6,7)$, $c_3 = (12,9)$ e $c_4 = (13,3)$ e as respectivas distâncias entre os centroides.

Tabela 7.6 Distâncias inter-centroides, considerando os quatro centroides da Figura 7.6.

$d(1,2) = d(c_1, c_2) = d((4,2), (6,7)) = \text{SQRT}((4-6)^2 + (2-7)^2) = \text{SQRT}(4+25) = \text{SQRT}(29) = 5,38$
$d(1,3) = d(c_1, c_3) = d((4,2), (12,9)) = \text{SQRT}((4-12)^2 + (2-9)^2) = \text{SQRT}(64+49) = \text{SQRT}(113) = 10,63$
$d(1,4) = d(c_1, c_4) = d((4,2), (13,3)) = \text{SQRT}((4-13)^2 + (2-3)^2) = \text{SQRT}(81+1) = \text{SQRT}(82) = 9,05$
$d(2,3) = d(c_2, c_3) = d((6,7), (12,9)) = \text{SQRT}((6-12)^2 + (7-9)^2) = \text{SQRT}(36+4) = \text{SQRT}(40) = 6,32$
$d(2,4) = d(c_2, c_4) = d((6,7), (13,3)) = \text{SQRT}((6-13)^2 + (7-3)^2) = \text{SQRT}(49+16) = \text{SQRT}(65) = 8,06$
$d(3,4) = d(c_3, c_4) = d((12,9), (13,3)) = \text{SQRT}((12-13)^2 + (9-3)^2) = \text{SQRT}(1+36) = \text{SQRT}(37) = 6,08$

A matriz de distâncias entre os centroides (d) é pois:

$$d = \begin{pmatrix} 0 & 5,38 & 10,63 & 9,05 \\ 5,38 & 0 & 6,32 & 8,06 \\ 10,63 & 6,32 & 0 & 6,08 \\ 9,05 & 8,06 & 6,08 & 0 \end{pmatrix} \text{ e, portanto a matriz de permutações é } O = \begin{pmatrix} 1 & 2 & 4 & 3 \\ 2 & 1 & 3 & 4 \\ 3 & 4 & 2 & 1 \\ 4 & 3 & 2 & 1 \end{pmatrix}$$

Parafraseando o pseudocódigo do Algoritmo 7.2, uma instância x_i é comparada com os centroides, em ordem crescente de distância do centroide $c(x_i)$, que é o centroide do grupo ao qual x_i tinha sido previamente atribuída. Se for atingido um centroide que é suficientemente longe de $c(x_i)$, todos os centroides posteriores podem ser abandonados e uma próxima instância é considerada. Como o autor comenta em [Phillips 2002], o algoritmo sort-Means é significativamente mais rápido que o algoritmo compare-Means, exceto quando o número de centroides é muito alto. A diferença entre ambos é mais pronunciada quando a dimensionalidade (M) das instâncias é pequena.

```

procedure sort-Means(C,K,G)
input: C = {x1, x2, ..., xN}      % conjunto com N instâncias de dados a serem agrupadas
      K                          % número de agrupamentos
output: G = {G1, G2, ..., GK}    % agrupamento final
begin
  % Fase de Inicialização
  {c1, c2, ..., cK} ← escolha_inicial_centroides(C)      % um centróide ci é escolhido para cada
                                                          % grupo Gi (i=1, ..., K)

  % todas as instâncias são atribuídas ao grupo G1
  G1 ← { }
  for i ← 1 to N do G1 ← G1 ∪ {xi}
  % fase iterativa
  repeat
    % determine a distância entre cada par de centroides e construa a matriz dK×K
    for i ← 1 to K do
      for j ← i+1 to K do d(ci, cj) ← distancia(ci, cj)
    call torne_matriz_simetrica(d)
    % construa a matriz de permutação OK×K onde cada linha i (i=1, ..., K) é uma permutação de 1,...,K,
    % representando os grupos em ordem crescente de distância de seus centroides, para o centróide ci
    O ← contrucao_matriz_ordenada(d)
    for i ← 1 to N do
      begin
        c ← c(xi)
        mindist ← distancia(xi, c)
        for j ← 2 to K do
          begin
            h ← O[c,j]
            if d[c, ch] >= 2×mindist then
              break
            else begin
              dist ← distancia(xi, ch)
              if dist < mindist then
                begin
                  mindist ← dist
                  atualize_centroide(xi, ch)
                end
              end
            end
          end
        end
      end
    end
    % Atribuindo instâncias a clusters por meio da identificação de seus centroides associados
    for i ← 1 to K do Gi ← { }
    for i ← 1 to N do
      begin
        c ← indice_centroide(xi) % índice associado ao centróide de uma instância
        Gc ← Gc ∪ {xi}
      end
    % Atualizando os centroides de clusters
    for i ← 1 to K do ci ← media_das_instancias_atribuidas (Gi)
  until centroides não mudam
  end
  return G = {G1, G2, ..., GK}
end procedure

```

Algoritmo 7.2 Pseudocódigo do k-Means implementando a estratégia sort-Means, proposta em [Phillip 2002].

7.4.1 Um Exemplo Completo do Uso do Sort-Means

No desenvolvimento do exemplo foi utilizado o conjunto de instâncias C , mostradas na Tabela 7.1, com 10 instâncias, representadas como pontos no gráfico, em que cada instância é descrita por dois atributos numéricos, $A1$ e $A2$, cujos valores foram artificialmente gerados. O conjunto C é ilustrado na Figura 7.2, de forma a permitir maior entendimento sobre o conjunto de instâncias.

Inicialmente o sort-Means escolhe aleatoriamente K ($K = 3$) instâncias do conjunto C como centroides iniciais, representados como estrelas no gráfico. A Figura 7.7 ilustra as instâncias escolhidas como centroides e para o *trace* de alto nível que segue, foi usada a distância euclidiana para calcular a similaridade entre instância-centroide e centroide-centroide. Todas as instâncias são, inicialmente, atribuídas ao mesmo grupo que, para o exemplo sendo apresentado, é o grupo G_1 (mostrado em azul).

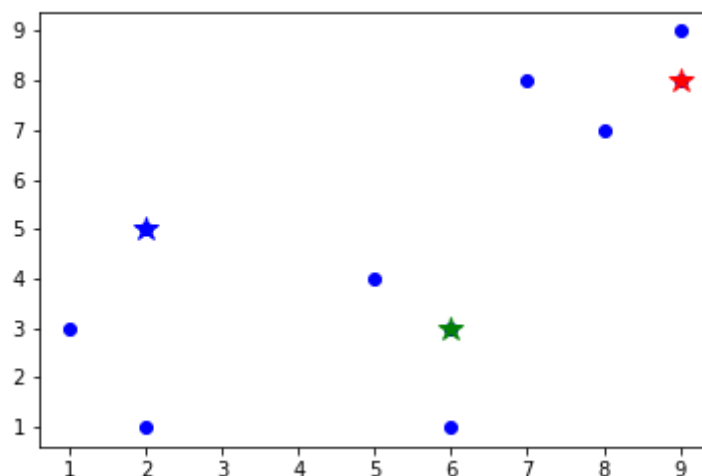


Figura 7.7 Representação dos 3 centroides iniciais $c_1 = (2,5)$, $c_2 = (6,3)$ e $c_3 = (9,8)$ representados como estrelas (azul, verde e vermelho, respectivamente). As instâncias foram atribuídas ao grupo G_1 , ilustrado como pontos de cor azul.

Iteração 1: Ao início da fase iterativa é realizado o cálculo de distância entre os centroides iniciais, criando, assim, a matriz de distância $d_{K \times K}$, com o custo de $(K \times (K-1))/2$. Para o exemplo foram realizados 3 cálculos de distância entre centroides na iteração atual. A Tabela 7.7 apresenta as distâncias para os centroides iniciais escolhidos. Em seguida é criada a matriz de permutação O , em que linhas são ordenadas

com base na distância das instâncias ao centroide c_j . A Tabela 7.8 apresenta a matriz de permutação para o exemplo em desenvolvimento.

Tabela 7.7 Matriz de distância d da iteração 1 do algoritmo sort-Means

#ID	1	2	3
1	0	4,47	7,61
2		0	5,83
3			0

Tabela 7.8 Matriz de permutação O da iteração 1 do algoritmo sort-Means

#ID	1	2	3
1	1	2	3
2	2	1	3
3	3	2	1

Em seguida são calculadas as distâncias da instância x_i até o centroide ao qual está atribuída; na primeira iteração todas as instâncias foram atribuídas ao grupo G_1 e então, são calculadas as distâncias entre x_i e c_1 . Para cada instância x_i são verificadas as seguintes condições $d(c(x_i), c_j) \geq 2 \times d(x_i, c(x_i)) (*)$, em que j varia de acordo com a ordem apresentada na linha da matriz de permutação. Se a condição (*) for verdadeira não é preciso mais calcular a distância entre x_i e os centroides uma vez que, devido à ordenação, é certo que nenhum dos centroides remanescentes pode estar mais próximo da instância que o $c(x_i)$, sendo assim, uma próxima instância é considerada.

Caso a condição (*) seja falsa, é necessário verificar a distância entre x_i e c_j . Caso a desigualdade $d(x_i, c_j) < d(x_i, c(x_i))$ seja verificada a instância x_i é atribuída a c_j e o processo segue para realizar a análise do próximo centroide. A Tabela 7.9 apresenta os cálculos que foram evitados na iteração 1 do exemplo.

Tabela 7.9 Cálculos de distância entre as instâncias e centroides que foram evitados na iteração 1.

#ID	1	2	3
1			×
2		×	×
3			×
4			
5			
6			
7			
8			
9			
10			

Por fim, as posições dos centroides são recalculadas, a partir das médias das instâncias atribuídas a cada um dos grupos G_i . Um total de 29 cálculos de distância foram realizados nessa iteração (26 instância-centroide e 3 centroide-centroide). A Figura 7.8 apresenta o agrupamento formado ao final da iteração 1.

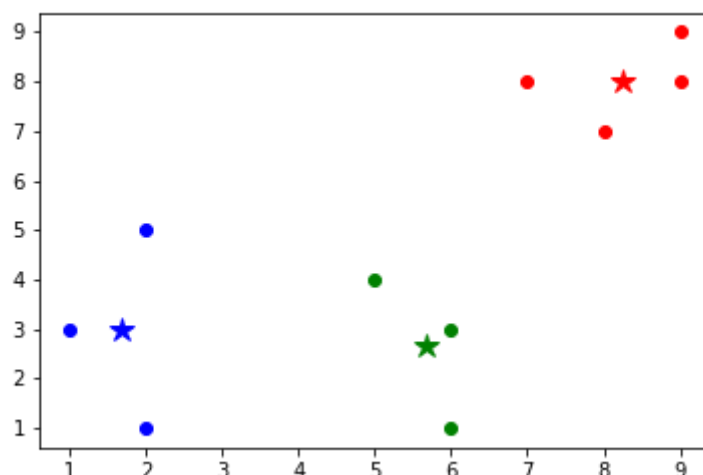


Figura 7.8 Representação o agrupamento formado após a iteração 1 do algoritmo sort-Means, $G_1=\{1,2,10\}$, $G_2=\{3,4,5\}$ e $G_3=\{6,7,8,9\}$.

Iteração 2: No início da iteração 2 a matriz de distância d é recalculada, com custo de 3 cálculos de distância entre os centroides e a matriz de permutação é atualizada. A Tabela 7.10 apresenta os novos valores da matriz de distância d e a Tabela 7.11 apresenta os valores da matriz de permutação O .

Tabela 7.10 Matriz de distância d da iteração 2 do algoritmo sort-Means.

#ID	1	2	3
1	0	4,01	8,26
2		0	5,92
3			0

Tabela 7.11 Matriz de permutação O da iteração 2 do algoritmo sort-Means.

#ID	1	2	3
1	1	2	3
2	2	1	3
3	3	2	1

Após realizar a verificação da condição $d(c(x_i), c_j) \geq 2 \times d(x_i, c(x_i))$ (*), em que j varia de acordo com a ordem da linha da matriz de permutação, como descrito na iteração 1. Foram evitados um total de 19 cálculos de distância entre instâncias e centroides de um total de 30 cálculos que seriam realizados pelo k-Means padrão. A Tabela 7.12 apresenta os cálculos de distância que foram evitados. Na iteração 2 foram realizados 14 cálculos de distâncias.

Tabela 7.12 Cálculos de distância entre as instâncias e centroides que foram evitados na iteração 2.

#ID	1	2	3
1		×	×
2			×
3	×		×
4	×		×
5	×		×
6	×	×	
7	×	×	
8	×	×	
9	×	×	
10		×	×

Como não houve mudança no agrupamento gerado na iteração atual, o algoritmo sort-Means finaliza. A Figura 7.9 apresenta o agrupamento final produzido.

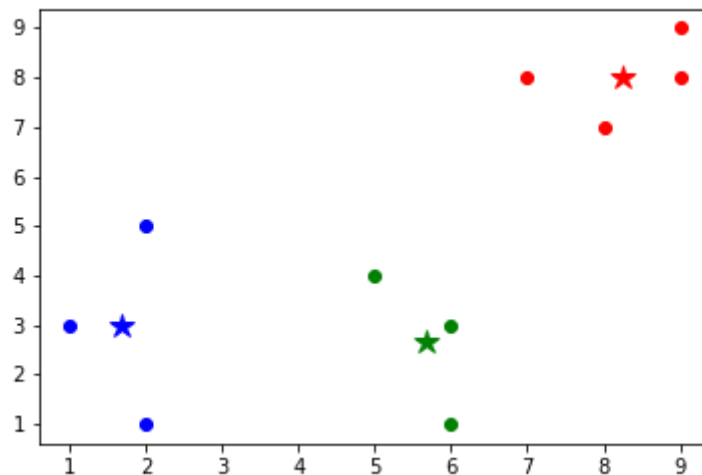


Figura 7.9 Representação o agrupamento final gerado pelo algoritmo compare-Means, $G_1=\{1,2,10\}$, $G_2=\{3,4,5\}$ e $G_3=\{6,7,8,9\}$.

Para o exemplo foram realizados 44 cálculos de distância (6 de centroide–centroide e 38 de instância–centroide), contra 60 que seriam realizados pelo k-Means padrão.

7.5 O Algoritmo de Elkan

A proposta de Elkan contempla o uso da desigualdade triangular de duas maneiras distintas, subsidiadas por dois resultados teóricos, bem como de um monitoramento dos limites superiores e inferiores das distâncias entre instâncias e centroides de grupos.

Como comenta Hamerly em [Hamerly 2010], o k-Means-Elkan elimina um grande número de cálculos de distâncias entre instâncias e centroides. Usa limites eficientemente atualizados de distâncias entre instâncias–centroides para determinar quando cálculos exatos de distâncias podem ser evitados. Especificamente, para N instâncias de dados e um número K de grupos (centroides), o algoritmo guarda as seguintes informações:

- (1) N limites superiores referentes à distância entre cada instância e seu centroide associado;
- (2) $K \times N$ limites inferiores associados às distâncias instâncias-centroides;
- (3) $O(K^2)$ distâncias entre os centroides
- (4) N atribuições de instâncias a grupos, da iteração anterior.

O algoritmo é bem similar ao k-Means padrão. Antes de calcular a distância entre uma instância x e qualquer centroide, entretanto, primeiro determina (usando as informações de limites superior e inferior e, também, as distâncias entre os centroides), se é possível que qualquer outro centroide esteja mais perto de x do que aquele já associado a x . Se não for possível, então os cálculos das distâncias não são realizados.

O algoritmo considera várias situações que devem ser verificadas para determinar se cada cálculo de distância pode ser evitado e, particularmente, aquela situação em que:

- (1) envolve uma instância x , o centroide c associado a x , c e um outro centroide c^* . Se o limite superior da distância entre x e c for menor do que o limite inferior da distância entre x e c^* , então c^* não pode ser um centro mais perto de x do que c ;
- (2) se o limite superior da distância entre x e c for menor do que metade da distância entre c e c^* então, c^* não pode ser um centroide mais perto de x .

O k-Means-Elkan atualiza os limites superiores e inferiores, ao final da iteração, cada vez que um centroide muda. O limite superior para uma instância x aumenta com o valor da distância com a qual o centroide a ela associado, foi movido. O limite inferior entre uma instância x e um centroide c diminui do valor da distância com a qual c foi movido. Usando esse método, embora os limites não sejam rígidos, eles permanecem corretos.

Suponha agora, uma situação em que não se saiba exatamente $d(x,c)$, mas é conhecido um limite superior u , tal que $u \geq d(x,c)$. Então, será necessário calcular $d(x,c^*)$ e $d(x,c)$ apenas se $u > 1/2 \times d(c,c^*)$.

Se $u \leq 1/2 \times \min\{d(c,c^*)\}$, em que o mínimo é calculado considerando todos os $c^* \neq c$, a instância x deve permanecer associada ao grupo cujo centro é c e todos os cálculos de distância associados a x podem ser evitados.

No que segue é mostrado como o Lema 2, apresentado no Capítulo 6 Seção 6.2, é usado. Considere uma instância de dado x qualquer. Considere também um centroide qualquer c e que c' seja a versão anterior desse mesmo centroide (*i.e.*, suponha que os centroides sejam numerados 1, 2, ..., K . Se c é o centroide identificado como de número j , na iteração corrente W , então c' é o centroide de número j da iteração anterior, $W-1$). Suponha também que na iteração $W-1$, conhecia-se um limite inferior l_{im}' , tal que $d(x,c') \geq l_{im}'$. Com base nessas informações, pode-se inferir um limitante inferior l_{im} para a iteração corrente W , como segue.

$$d(x,c) \geq \max\{0, d(x,c') - d(c,c')\} \geq \max\{0, l_{im}' - d(c,c')\} = l_{im}$$

Informalmente, se l_{im}' é uma boa aproximação para a distância anterior entre x e o j -ésimo centroide, e esse centroide se moveu apenas uma pequena distância, então l_{im} é uma boa aproximação para a distância atualizada.

A Figura 7.10, baseada em uma figura apresentada em [Hamerly & Drake 2015], ilustra o uso da desigualdade triangular para limitar a distância entre a instância x e o centroide c , que é a nova posição do centroide associado a instâncias em questão. Após a nova posição do centroide ser calculado, isto é, c' se torna c , a distância entre c' e c está assinalada pelo círculo pontilhado com centro em c' . Assim, o limite superior $u(x)$

permite saber a distância máxima que o centroide c poderá estar da instância x , permitindo assim, nos casos definidos, não realizar o cálculo de distância.

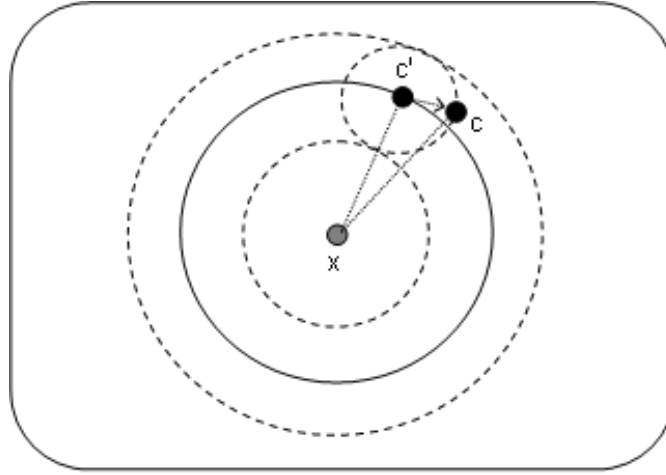


Figura 7.10 Ilustração do deslocamento do centroide c da iteração $W-1$ para a atual W .

Os limites superiores e inferiores da distância entre x e c são derivados dos dois usos da desigualdade triangular:

$$(1) d(x,c) \leq d(x,c') + d(c,c')$$

$$(2) d(x,c) \leq d(x,c') + d(c,c') \text{ e, então, } d(x,c) - d(c,c') \leq d(x,c')$$

$$\text{De (1) e (2) vem que } d(x,c) - d(c,c') \leq d(x,c') \leq d(x,c) + d(c,c')$$

Suponha que $u(x) \geq d(x,c)$ seja um limite superior da distância entre a instância x e o centroide c do grupo ao qual x pertence. Suponha também que $l_{im}'(x,c') \leq d(x,c')$ seja um limite inferior da distância entre x e algum outro centroide c' .

Se $u(x) \leq l_{im}'(x,c')$ então $d(x,c) \leq u(x) \leq l_{im}'(x,c') \leq d(x,c')$, de maneira que não é necessário calcular $d(x,c)$ ou $d(x,c')$. Note que nunca será necessário na corrente iteração do algoritmo acelerado, calcular $d(x,c')$, mas pode ser necessário ter que calcular $d(x,c)$ exatamente, devido a algum outro centroide c^* , para o qual $u(x) > l_{im}(x,c^*)$.

Resumidamente, para cada instância x_i a ser agrupada, o algoritmo de Elkan usa um limite superior e K limites inferiores. O limite superior é a distância entre a instância x_i e

o centroide do qual está mais próxima, ou seja, $u(x_i) \geq d(x_i, c(x_i))$, em que $c(x_i)$ representa o centroide do grupo ao qual x_i pertence. Cada limite inferior $l_{im}(x_i, c_j) \leq d(x_i, c_j)$ limita a distância entre a instância x_i e um centroide c_j . Tanto o limite superior quanto os limites inferiores podem ser eficientemente atualizados por meio da soma (subtração) da distância 'percorrida' por cada centroide, após cada iteração do k-Means.

Durante o processo iterativo, cada vez que os centroides se movem, o algoritmo calcula e armazena a distância entre cada par de centroides, assim como a metade da distância entre cada centroide c_j e seu centroide mais próximo, como s_j .

Quando a condição $u(x_i) \leq d(x_i, c(x_i))$ for verdade, o teste $u(x_i) \leq s(c(x_i))$ permite que o algoritmo de Elkan não execute o seu *loop* mais interno, para a instância x_i . Isso se deve ao fato que nenhum outro centroide poderia, possivelmente, estar mais perto de x_i , do que o centroide do grupo ao qual já pertence.

Quando o algoritmo de Elkan atinge o *loop* mais interno, pode ser preciso determinar se c_j é o centroide mais próximo de x_i do que o centroide correntemente atribuído. Entretanto, se $u(x_i) \leq l_{im}(x_i, c_j)$ ou $u(x_i) \leq 1/2 \times d(c(x_i), c_j)$, o cálculo se torna desnecessário uma vez que não é possível para c_j ser o centroide mais próximo. O último teste usa as distâncias entre centroides que foram armazenadas.

O algoritmo proposto por Elkan foi a primeira variante do k-Means a usar limites inferiores e, também, o primeiro algoritmo que passa informação de uma iteração à seguinte. O algoritmo proposto em [Judd *et al.* 1998] reusa informação de uma iteração para a próxima, mas, entretanto, não faz uso de limites inferiores ou superiores. O Algoritmo 7.3 apresenta o pseudocódigo do algoritmo k-Means-Elkan.

```

procedure k-Means-Elkan(C,K,G)
input: C = {x1, x2, ..., xN}      % conjunto com N instâncias de dados a serem agrupadas
      K                          % número de agrupamentos
output: G = {G1, G2, ..., GK}   % agrupamento final
begin
  % Fase de Inicialização
  {c1, c2, ..., cK} ← escolha_inicial_centroides(C)      % um centroide ci é escolhido para cada
                                                         % grupo Gi (i=1, ..., K)

  % todas as instâncias são atribuídas ao grupo G1
  G1 ← { }
  for i ← 1 to N do
    begin
      G1 ← G1 ∪ {xi}
      u(xi) ← ∞      % atribui o valor infinito a variável u(xi)
      for j ← 1 to K do lim(xi,cj) ← 0
    end
  % fase iterativa
  repeat
    % determine a distância entre cada par de centroides e construa a matriz dK,K
    for i ← 1 to K do
      for j ← i+1 to K do d(ci,cj) ← distancia(ci,cj)
    call torne_matriz_simetrica(d)
    for j ← 1 to K do
      s(j) ← metade_dist_centroide_mais_proximo_de_c(d)      % retorna a metade da menor
                                                         % distância de cj para ci, i ≠ j,

    for i ← 1 to N do
      begin
        if u(xi) > s(c(xi)) then
          begin
            r ← True
            for j ← 1 to K do
              begin
                if c(xi) ≠ cj and u(xi) > lim(xi,cj) and u(xi) > d(xi,cj)/2 then
                  begin
                    if r = True then
                      begin
                        u(xi) ← distancia(xi, c(xi))
                        r ← False
                      end
                    if u(xi) > lim(xi,cj) and u(xi) > d(xi,cj)/2 then
                      begin
                        lim(xi,cj) ← distancia(xi,cj)
                        if lim(xi,cj) < u(xi) then
                          begin
                            atualize_centroide(xi,cj)
                            u(xi) ← lim(xi,cj)
                          end
                        end end end
                      end end end
                  end end end
                end
              end
            end
            for j ← 1 to K do cj ← media_das_instancias_atribuidas(Gj)
            for j ← 1 to K do δ(j) ← distancia(cj,cj') % deslocamento do centroide cj da iteração anterior para a atual
            for i ← 1 to N do
              begin
                u(xi) ← u(xi) + δ(c(xi)) % atualiza o limite superior de xi para o centroide a qual está atribuído
                for j ← 1 to K do
                  lim(xi,cj) ← lim(xi,cj) - δ(j) % atualiza os valores de lim(xi,cj)
                end
              end
            until centroides não mudam
          end
        return G = {G1,G2,...,GK}
      end procedure

```

Algoritmo 7.3 Pseudocódigo do k-Means implementando a estratégia proposta por Elkan, adaptado de [Hamerly & Drake 2015]

7.5.1 Um Exemplo Completo do Uso do k-Means-Elkan

No desenvolvimento deste exemplo foi utilizado o conjunto de instâncias C , mostradas na Tabela 7.1 com 10 instâncias, representadas como pontos no gráfico da Figura 7.2, em que cada instância é descrita por dois atributos numéricos, $A1$ e $A2$, cujos valores foram artificialmente gerados.

Inicialmente, o k-Means-Elkan escolhe aleatoriamente K ($K = 3$) instâncias do conjunto C como centroides iniciais, representados como estrelas na Figura 7.11. Para o *trace* de alto nível que segue, foi usada a distância euclidiana para calcular a similaridade entre instância-centroide e centroide-centroide. Todas as instâncias são, inicialmente, atribuídas ao mesmo grupo que, no exemplo, é o grupo G_1 (ilustrado em azul), o limite superior de cada instância $u(x_i) = \infty$ e os limites inferiores $l_{im}(x_i, c_j) = 0$.

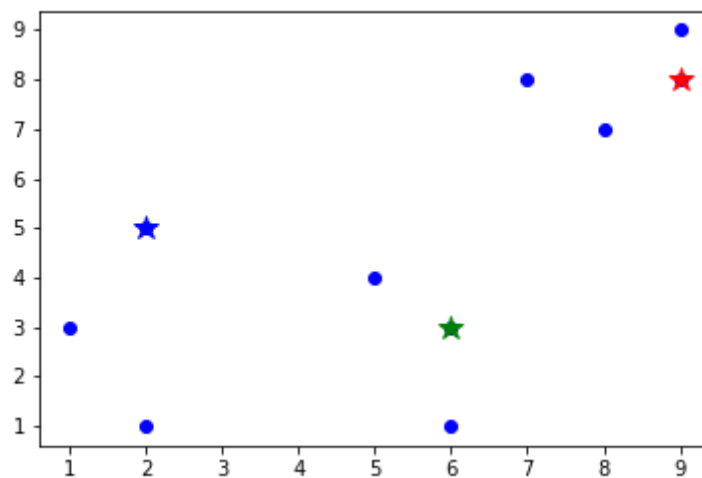


Figura 7.11 Representação dos 3 centroides iniciais $c_1=(2,5)$, $c_2=(6,3)$ e $c_3=(9,8)$ representados como estrelas (azul, verde e vermelho, respectivamente). As instâncias foram atribuídas ao grupo G_1 , ilustrado como pontos de cor azul.

Iteração 1: Ao iniciar a fase iterativa, os cálculos de distância entre os centroides iniciais são realizados, criando, assim, a matriz de distância $d_{K \times K}$, com o custo de $(K \times (K-1))/2$. No exemplo foram realizados 3 cálculos de distância entre centroides. A Tabela 7.13 apresenta as distâncias para os centroides iniciais escolhidos. Em seguida é definido o valor s_j , para $j=1, \dots, K$, de forma que s_j recebe a metade da menor distância entre c_j e c_k , em que $j \neq k$. A Tabela 7.14 apresenta os valores de s_j .

Tabela 7.13 Matriz de distância d da iteração 1 do algoritmo k-Means-Elkan

#ID	1	2	3
1	0	4,47	7,61
2		0	5,83
3			0

Tabela 7.14 Vetor contendo o valor de s_j , ou seja, a metade da menor distância entre o centroide c_j e qualquer outro centroide c_k , em que $j \neq k$.

#ID	1	2	3
s	2,23	2,23	2,91

Para cada instância $x_i \in C$ são verificadas 4 condições, antes de atribuir a instância x_i a um novo centroide c_j . As condições são:

(1) Considere a condição $u(x_i) < s(c(x_i))$. Se essa condição for verdadeira, a instância x_i não muda de grupo e o processo prossegue analisando a próxima instância. Porém, como todos os limites superiores foram definidos como ∞ antes da fase iterativa, para todas as instâncias x_i na iteração 1 essa condição é falsa.

(2) Para cada centroide c_j são verificados se $c_j \neq c(x_i)$ e $u(x_i) > l_{im}(x_i, c_j)$ e $u(x_i) > d(x_i, c_j)/2$. Se essas condições forem verdadeiras é necessário atualizar o limite superior $u(x_i) = d(x_i, c(x_i))$. O cálculo de distância entre a instância x_i e o centroide ao qual ela está atribuída somente ocorre uma única vez por iteração.

(3) Após a atualização do limite superior, é novamente verificado se $u(x_i) > l_{im}(x_i, c_j)$ e $u(x_i) > d(x_i, c_j)/2$. Caso sejam verdadeiras é necessário calcular o limite inferior $l_{im}(x_i, c_j) = d(x_i, c_j)$.

(4) Por fim, é verificada a condição $l_{im}(x_i, c_j) < u(x_i)$. Se verdadeira, $c(x_i) = c_j$, $u(x_i) = l_{im}(x_i, c_j)$.

A Tabela 7.15 apresenta os cálculos de distância que foram evitados na iteração 1 do algoritmo k-Means-Elkan.

Tabela 7.15 Cálculos de distância entre as instâncias e centroides que foram evitados na iteração

1.			
#ID	1	2	3
1		×	×
2		×	×
3			×
4			×
5			×
6			
7			
8			
9			
10			

Após a atribuição de instâncias ao centroide que lhes for mais próximo, centroides são atualizados como a média das instâncias que representam. Na Figura 7.12 é apresentado o agrupamento formado ao final da iteração 1.

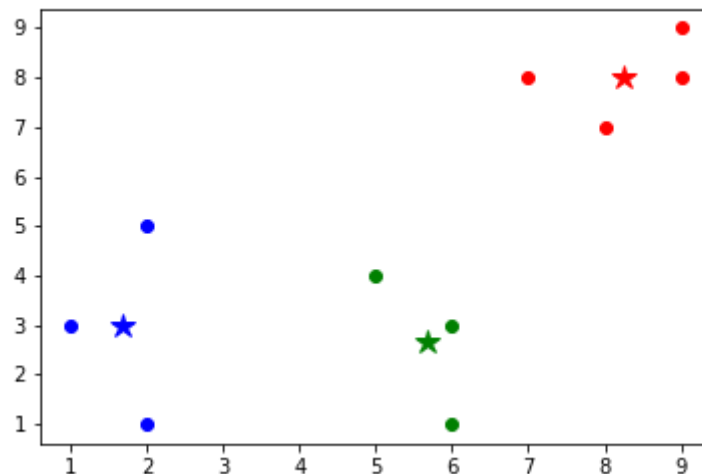


Figura 7.12 Representação o agrupamento formado após a iteração 1 do algoritmo k-Means-Elkan, $G_1=\{1,2,10\}$, $G_2=\{3,4,5\}$ e $G_3=\{6,7,8,9\}$.

Após a atualização dos centroides é calculado o deslocamento que cada centroide sofreu da sua posição inicial para a atual δ_j , com o custo de K cálculos de distância. O vetor de valores de deslocamento associados aos centroides é apresentado na Tabela 7.16.

Tabela 7.16 Vetor contendo o valor de δ_j , $j = 1, 2, 3$ ou seja, o deslocamento dos centroides da sua posição inicial para a atual.

#ID	1	2	3
δ	2,02	0,47	0,75

Por fim é realizada a atualização dos limites superiores e inferiores para cada instância x_i . Os limites superiores $u(x_i) = u(x_i) + \delta(c(x_i))$ estão apresentados na Tabela 7.17 e os limites inferiores $l_{im}(x_i, c_j) = l_{im}(x_i, c_j) - \delta(c(x_i))$ estão apresentados na Tabela 7.18.

Tabela 7.17 Valores do limite superior $u(x_i)$ ao final da primeira iteração.

#ID	1	2	3	4	5	6	7	8	9	10
u	4,26	2,02	1,88	0,47	2,47	2,75	2,16	0,75	1,75	6,02

Tabela 7.18 Valores dos limites inferiores $l_{im}(x_i, c_j)$ ao final da primeira iteração 1.

#ID	c_1	c_2	c_3
1	0	0	0
2	0	0	0
3	0	0,94	0
4	0	0	0
5	0	1,52	0
6	0	4,62	1,25
7	0	4	0,66
8	0	5,35	0
9	0	6,23	0,25
10	0	4	9,14

Foram realizados 29 cálculos de distância na iteração 1; caso o k-Means padrão tivesse sido utilizado, o total de cálculos seria de 30. As vantagens do uso do k-Means-Elkan ficam mais evidentes a partir da segunda iteração, como mostrado na sequência.

Iteração 2: Ao iniciar a segunda iteração são recalculadas as distâncias entre os centroides, com a atualização da matriz d com os valores apresentado na Tabela 7.19. Em seguida é recalculado o valor de s_j para cada um dos centroides c_j , cujos valores estão mostrados na Tabela 7.20.

Tabela 7.19 Matriz de distância d da iteração 2 do algoritmo k-Means-Elkan.

#ID	1	2	3
1	0	4,01	8,26
2		0	5,92
3			0

Tabela 6.20 Vetor contendo o valor de s_j , ou seja, a metade da menor distância entre o centroide c_j e qualquer outro centroide c_k , em que $j \neq k$.

#ID	1	2	3
s	2,00	2,00	2,96

Para cada instância x_i é verificado se as 4 condições descritas na iteração 1 são verdadeiras, para que uma instância seja atribuída a um novo centroide. Na iteração 2 foram evitados 25 cálculos de distância entre instâncias e centroides. A Tabela 7.21 identifica os cálculos de distância entre instâncias e centroides que foram evitados.

Tabela 7.21 Cálculos de distância entre as instâncias e centroides que foram evitados na iteração

2.			
#ID	1	2	3
1		×	×
2			×
3	×	×	×
4	×	×	×
5	×		×
6	×	×	×
7	×	×	×
8	×	×	×
9	×	×	×
10		×	×

Como não houve mudança no agrupamento gerado na iteração atual, o algoritmo k-Means-Elkan finaliza. A Figura 7.13 apresenta o agrupamento final induzido.

Na iteração 2 não foi necessário calcular o deslocamento δ_j dos centroides da iteração anterior para a atual e tampouco foi necessário atualizar os limites superiores e inferiores. No total foram realizados 8 cálculos. Supondo que o algoritmo não finalizasse ao término da segunda iteração, seriam realizados mais 3 cálculos de distância para estimar o deslocamento do centroide c'_j para c_j , o que totalizaria 11 cálculos de distância contra 30 do k-Means padrão.

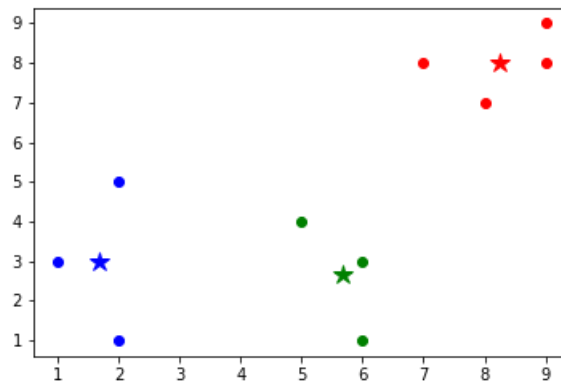


Figura 7.13 Representação o agrupamento final gerado pelo algoritmo k-Means-Elkan, $G_1=\{1,2,10\}$, $G_2=\{3,4,5\}$ e $G_3=\{6,7,8,9\}$.

7.6 O Algoritmo k-Means-Yinyang

O algoritmo proposto por Ding e coautores, em [Ding *et al.* 2015], identificado nessa dissertação como k-Means-Yinyang, se mostrou uma variante do algoritmo k-Means. A sua principal contribuição está em agrupar os centroides iniciais e, assim como o algoritmo proposto por Elkan, em [Elkan 2003], aplicar os conceitos de desigualdade triangular apresentados no Lema 1 e Lema 2 da Seção 6.2, aperfeiçoando o uso dos limites superiores e inferiores.

O algoritmo k-Means-Yinyang busca realizar uma manutenção cuidadosa do limite superior da distância de uma instância ao centroide ao qual foi atribuída, bem como dos limites inferiores da distância da instância até os grupos de centroides GC_f , diferentemente do k-Means-Elkan que mantém limites inferiores para cada centroide c_j . A iteração entre esses dois limites (superior e inferior) pode ser considerada com a funcionalidade de um filtro de dois níveis (filtro de grupo/global e filtro local), permitindo assim evitar cálculos desnecessários de forma mais eficaz.

Inicialmente os K centroides iniciais são agrupados em F grupos $GC = \{GC_1, GC_2, \dots, GC_F\}$, sendo cada $GC_f \in GC$, $f = 1, \dots, F$, um conjunto de centroides. O conjunto GC é formado antes de iniciar a primeira iteração do k-Means-Yinyang. Ding e coautores sugerem usar um método de agrupamento para criar os grupos de centroides iniciais e indicam o uso do próprio k-Means padrão para a formação dos grupos de centroides, executando o k-Means por cinco iterações, de forma a produzir grupos razoáveis. A Figura 7.14 ilustra um agrupamento GC de um conjunto C , com 9 centroides iniciais com dois atributos numéricos A_1 e A_2 , descritos na Tabela 7.22, divididos em 3 grupos GC_1 , GC_2 e GC_3 .

Tabela 7.22 Conjunto C contendo 9 centroides gerados artificialmente, descritos por dois atributos numéricos.

#ID	A_1	A_2	#ID	A_1	A_2
1	1	3	6	7	8
2	2	5	7	8	7
3	5	4	8	9	8
4	6	3	9	9	9
5	6	1			

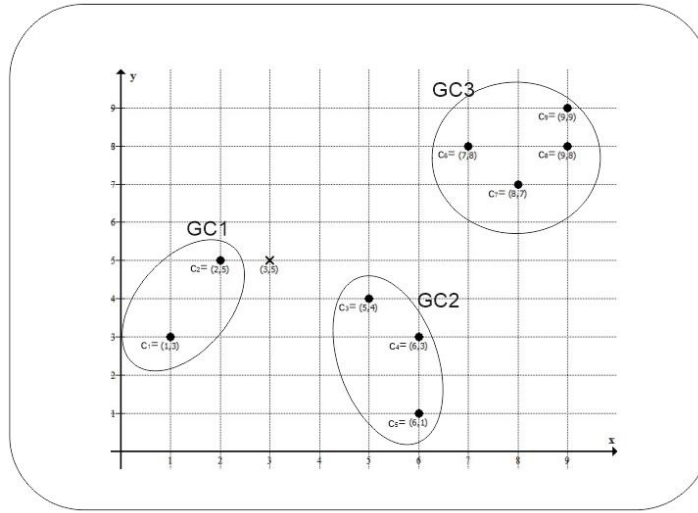


Figura 7.14 Diagrama exibindo um conjunto de centroides iniciais $C = \{c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9\}$ separados em três grupos $GC = \{GC_1, GC_2, GC_3\}$ em que $GC_1 = \{c_1, c_2\}$, $GC_2 = \{c_3, c_4, c_5\}$ e $GC_3 = \{c_6, c_7, c_8, c_9\}$.

Os valores dos limites superiores u são dados pela distância entre a instância e o centroide a qual ela foi associada, isto é, $u(x) = d(x, c(x))$, e os limites inferiores $l_{im}(x, GC_f)$ são dados pela menor distância entre a instância x e o centroide $c \in GC_f$ excluindo $c(x)$, isto é, $l_{im}(x, GC_f) = \min\{d(x, c)\}$, $\forall c \in GC_f - c(x)$. A Figura 7.15 ilustra de forma simples os valores de $l_{im}(x, GC_f)$ para o conjunto GC apresentado.

Resumidamente, para cada instância x a ser agrupada, o algoritmo de Ding e coautores usa um limite superior e F limites inferiores, onde $F < K$. Tanto o limite superior e os limites inferiores podem ser eficientemente atualizados por meio de soma (subtração) da distância ‘percorrida’ por cada centroide, após cada iteração do k-Means.

A atualização do limite superior é dado por $u(x) = u'(x) + d(c(x), c'(x))$, onde $u'(x)$ é o valor do limite superior $u(x)$ na iteração anterior. A atualização dos limites inferiores de cada instância x para cada grupo GC_f é dado por $l_{im}(x_i, GC_f) = l_{im}'(x_i, GC_f) - \max\{d(c, c')\}$, $\forall c \in GC_f$, onde $d(c, c')$ é o deslocamento do centroide c da iteração anterior para a atual e $l_{im}'(x, GC_f)$ é o valor de $l_{im}(x_i, GC_f)$ na iteração anterior a atual W , isto é, $W-1$.

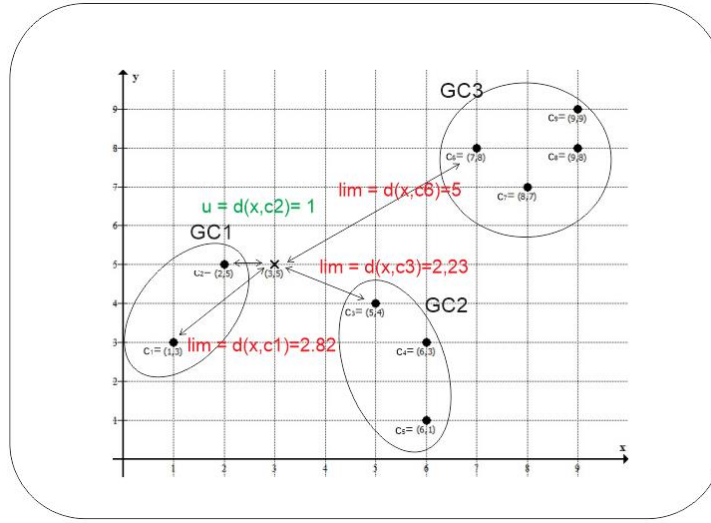


Figura 7.15 Diagrama exemplificando os valores do limite superior e limites inferiores de x, onde $u(x)=1$ e $l_{im}(x,GC_1) = 2,82$, $l_{im}(x,GC_2) = 2,23$ e $l_{im}(x,GC_3) = 5$.

Filtro Global/Grupo: Definido os valores de $l_{im}(x,GC_f)$, pode-se, assim, formar o filtro global. Esse filtro busca evitar comparações desnecessárias verificando se o menor valor entre os limites inferiores da instância x, isto é, $\min\{l_{im}(x,GC_f)\}$, é maior que o limite superior $u(x)$. Se a condição $\min\{l_{im}(x,GC_f)\} \geq u(x)$ (*) for satisfeita, pode-se afirmar que a instância x não muda de centroide ao qual estava associada na iteração anterior, e o processo pode ter continuidade com a análise da próxima instância. Porém, se a condição (*) for falsa, é necessário, em primeiro lugar, atualizar o valor de $u(x) = d(x,c(x))$ e realizar nova comparação (*), dessa forma, se a condição (*) for verdadeira, pode-se passar a análise da próxima instância, se falsa será necessário fazer a verificação no filtro de grupo.

O filtro de grupo analisa individualmente a distância de cada instância x para cada grupo GC_f isto é, $l_{im}(x,GC_f) \geq u(x)$, os grupos que satisfizerem essa condição são descartados, de forma que somente os grupos que não satisfizerem essa condição passam para análise do filtro local.

Filtro Local: Se um grupo GC_f para uma instância x passar pelo filtro global/grupo, um dos centroides $c \in GC_f$ pode ser o centroide mais próximo da instância x. Para evitar que se calcule a distância entre a instância x a todos os centroides $c \in GC_f$ foi projetado o filtro local.

Um centroide $c' \in GC_f$ (onde $c' \in GC_f$ refere-se ao centroide $c \in GC_f$ na iteração anterior a atual) não é mais próximo da instância x se há um centroide $c'_p \neq c'$ tal que $d(x, c'_p) \leq l_{im}'(x, GC_f) - d(c, c')$, dessa forma, é possível evitar os cálculos de distância até os centroides que satisfaçam essa condição. Os autores do algoritmo k-Means-Yinyang sugerem escolher como c'_p o segundo centroide mais próximo da instância x . A razão é que dessa forma o filtro local se mostrou mais efetivo para a próxima iteração.

Se o centroide passar pelo filtro local é necessário realizar o cálculo de distância entre a instância x e o centroide c , $d(x, c)$. Se $u(x) > d(x, c)$ (*), para $c \neq c(x)$, nesse caso, $u(x) = d(x, c)$, $c_p = c(x)$ e $c(x) = c$. Caso a condição (*) seja falsa, deve-se testar se $l_{im}(x, GC_f) > d(x, c)$, caso verdadeira $l_{im}(x, GC_f) = d(x, c)$.

O Algoritmo 7.4 apresenta o pseudocódigo do algoritmo Yinyang k-Means para maior clareza e entendimento.

Ding e coautores também propõem uma otimização do processo de atualização dos centroides, aproveitando o fato que apenas algumas instâncias mudam de grupo a cada iteração. Dessa forma, ao invés de calcular a média de todas as instâncias de um grupo, reutiliza o centroide como mostra a Eq. (7.1).

$$c = \frac{(c' * n'_j) - \left(\sum_{x' \in G'_j - G_j \cap G'_j} x' \right) + \left(\sum_{x \in G_j - G_j \cap G'_j} x \right)}{n_j} \quad \text{Eq. (7.1)}$$

onde, G_j e G'_j representa um grupo na iteração atual e na anterior, n_j e n'_j é o número de instâncias do grupo na iteração atual e na anterior, c e c' representam um centroide na iteração atual e na anterior e x e x' representam as instâncias pertencentes ao grupo G_j e G'_j , respectivamente.

O uso dessa proposta de aceleração da atualização dos centroides é vantajoso se e somente se menos da metade das instâncias tiver alterado de grupo. Ding e coautores afirmam que em seus experimentos em conjuntos de instâncias reais, essa condição nunca foi violada. No algoritmo k-Means-Yinyang implementado para uso nos experimentos desta Dissertação não foi utilizado aceleração de atualização da posição dos centroides, por não estar ligado diretamente ao escopo da investigação, ou seja, a desigualdade triangular.

```

procedure k-Means-Yinyang(C,K,G)
input: C = {x1, x2, ..., xN}    % conjunto com N instâncias de dados a serem agrupadas
      K                      % número de grupos do agrupamento
output: G = {G1,G2,...,GK}    % agrupamento final
begin
  % um centróide ci é escolhido para cada grupo Gi (i=1, ..., K)
  {c1, c2, ..., cK} ← escolha_inicial_centroides(C)
  % agrupar o K centroides em F=K/10 grupos (recomendado)
  GC ← agrupe_os_K_centroides_em_F_grupos({c1, c2, ..., cK})
  for i ← 1 to N do    % inicializa o limite superior e inferior para cada instância
    begin
      % determine a distância entre cada par de centroides e construa a matriz dK×K
      for j ← 1 to K do
        d(xi,cj) ← distancia(xi,cj)
      j ← argminj=1,...,K d(xi,cj)
      b ← cj
      atualize_centroide(xi,b)
      lim(xi,GCf) ← min{distancia(xi,cj)}, cj ∈ GCf – b
    end
    % Início da Fase iterativa
  repeat
    for j ← 1 to K do
      begin
        cj ← media_das_instancias_atribuidas(Gi)
        δ(cj) ← distancia(cj,c'j)    % δ(cj) é o deslocamento do centróide cj
      end    % da iteração anterior para atual
    δ(CGf) ← max{δ(cj)}, cj ∈ GCf    % maior valor de δ(cj) em que cj
    % pertença ao grupo GCf
    for i ← 1 to N do
      begin
        b ← centroide(xi)
        u(xi) ← u(xi) + δ (b)    % atualiza o limite superior de xi
        lim_old(xi,GCf) ← lim(xi,GCf)    % reserva os valores de lim(xi,GCf)
        for f ← 1 to F do
          lim(xi,GCf) ← lim(xi,GCf) – δ(GCf)    % atualiza os valores de lim(xi,GCf)
          b_old ← b    % reserva o valor de b
        end
        if min{lim(xi,GCf)} ≥ u(xi) then continue    % filtro global
        u(xi) ← distancia(xi,c(xi))    %atualiza o limite superior
        if min{lim(xi,GCf)} ≥ u(xi) then continue    % verifica novamente o filtro global
        % filtro de Grupo
        for f ← 1 to F do
          begin
            if lim(xi,GCf) < u(xi) then
              lim(xi,GCf) ← NULL
            end
          end
        end
      end
    end
  until

```

Algoritmo 7.4 Parte 1: pseudocódigo do algoritmo k-Means-Yinyang, adaptado de [Bottesch et al. 2016]

```

%Parte 2 : pseudocódigo do algoritmo k-Means-Yinyang
for f ← 1 to F do %filtro local
  begin
    if  $l_{im}(x_i, GC_f) = \text{NULL}$  then
      begin
        for  $c_j \in GC_f$  do
          begin
            if  $b\_old = c_j$  then continue
            if  $l_{im}(x_i, GC_f) < l_{im\_old}(x_i, GC_f) - \delta(c_j)$  then continue
             $dist\_centroide \leftarrow distancia(x_i, c_j)$ 
            if  $dist\_centroide < u(x_i)$  then
              begin
                 $l \leftarrow grupo\_ao\_qual\_pertence(b)$  % retorna a qual grupo b pertence
                 $l_{im}(x_i, GC_l) \leftarrow u(x_i)$ 
                 $u(x_i) \leftarrow dist\_centroide$ 
                 $b \leftarrow c_j$ 
              end
            else
              begin
                if  $dist\_centroide < l_{im}(x_i, GC_f)$  then
                   $l_{im}(x_i, GC_f) \leftarrow dist\_centroide$ 
                end
              end
            end
          end
        end
      end
    end
     $atualize\_centroide(x_i, b)$ 
  end
until centroides não mudam
end
return  $G = \{G_1, G_2, \dots, G_k\}$ 
end procedure

```

Algoritmo 7.4 Parte 2: pseudocódigo do algoritmo k-Means-Yinyang, adaptado de [Bottesch et al. 2016]

7.6.1 Um Exemplo Completo do Uso do k-Means-Yinyang

No desenvolvimento deste exemplo foi utilizado o conjunto de instâncias CI da Tabela 7.23 com 20 instâncias, representadas como pontos no gráfico mostrado na Figura 7.16, em que cada instância é descrita por dois atributos numéricos, A1 e A2, cujos valores foram artificialmente gerados.

Tabela 7.23 Conjunto CI contendo 20 instâncias geradas artificialmente, descritas por dois atributos numéricos.

#ID	A ₁	A ₂	#ID	A ₁	A ₂	#ID	A ₁	A ₂	#ID	A ₁	A ₂
1	1	3	6	7	8	11	3	9	16	4	1
2	2	5	7	8	7	12	4	8	17	4	3
3	5	4	8	9	8	13	9	1	18	6	9
4	6	3	9	9	9	14	9	5	19	2	9
5	6	1	10	2	1	15	7	3	20	1	7

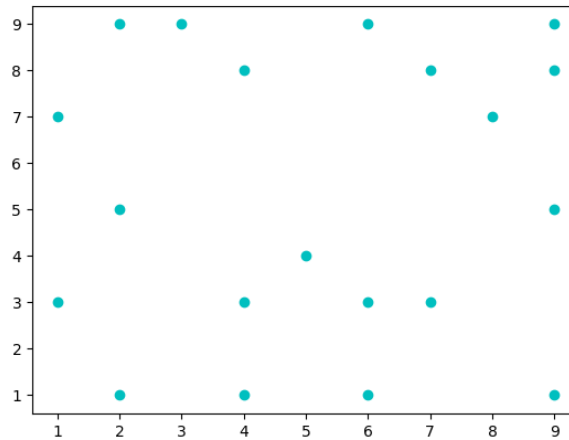


Figura 7.16 Representação das 20 instâncias do conjunto CI apresentadas na Tabela 7.23.

Inicialmente o k-Means-Yinyang escolhe aleatoriamente K ($K = 5$) instâncias do conjunto CI como centroides iniciais, representados como estrelas no gráfico mostrado na Figura 7.17.

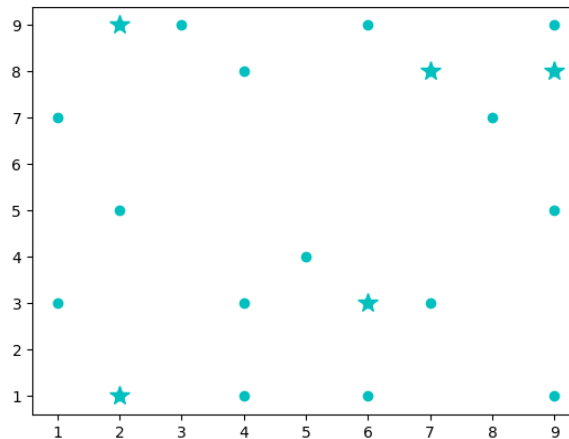


Figura 7.17 Conjunto CI com 20 instâncias e os 5 centroides iniciais identificados por estrelas *i.e.*, $c_1 = I_6$, $c_2 = I_4$, $c_3 = I_8$, $c_4 = I_{14}$, $c_5 = I_{10}$.

O próximo passo consiste na criação de um agrupamento dos centroides iniciais. Para este exemplo os centroides iniciais serão divididos em 2 grupos de centroides utilizando o k-Means padrão com no máximo 5 iterações. A Figura 7.18 apresenta os dois grupos gerados a partir dos 5 centroides iniciais. Foram realizados 20 cálculos de distância na indução dos grupos de centroides iniciais.

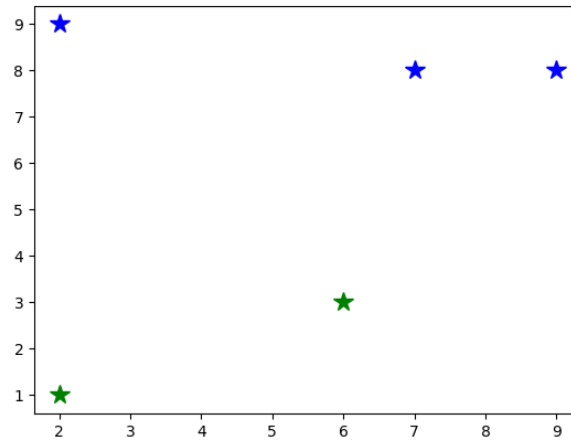


Figura 7.18 Representação dos 2 grupos de centroides, $GC_1=\{I_4, I_{10}\}$ em verde e $GC_2=\{I_6, I_8, I_{14}\}$ em azul, gerados a partir dos 5 centroides iniciais.

Iteração 1: Após a indução do agrupamento dos centroides iniciais em 2 grupos é realizada uma iteração completa do k-Means padrão, de forma a atribuir as instâncias aos centroides e definir os valores dos limites superiores $u(x_i)$ e inferiores $l_{im}(x_i, GC_f)$ de cada uma das instâncias pertencentes ao conjunto CI. Para isso foram realizados 100 cálculos de distância entre instância e centroide. Para finalizar a iteração é realizada a atualização da posição dos centroides.

A Tabela 7.24 apresenta os valores calculados para os limites superiores $u(x_i)$ da primeira iteração e a Tabela 7.25 apresenta os valores dos limites inferiores $l_{im}(x_i, GC_f)$. A Figura 7.19 apresenta o agrupamento formado após a primeira iteração.

Tabela 7.24 Valores do limite superior após a execução da primeira iteração.

#ID	1	2	3	4	5	6	7	8	9	10
u	2,2	4	1,4	0	2	0	1,4	0	1	0
#ID	11	12	13	14	15	16	17	18	19	20
u	1	2,2	3,6	3	1	2	2	1,4	0	2,2

Tabela 6.25 Valores dos limites inferiores após a execução da primeira iteração.

#ID	GC ₁	GC ₂	#ID	GC ₁	GC ₂	#ID	GC ₁	GC ₂	#ID	GC ₁	GC ₂
1	6	5	6	2	5	11	4,1	6,7	16	7,6	2,8
2	5,8	4	7	1,4	4,4	12	3	5,3	17	5,8	2,8
3	4,4	4,2	8	2	5,8	13	7	7	18	3,1	6
4	5	4,4	9	2,2	6,7	14	3,6	3,6	19	5	7,2
5	7	4	10	8	4,4	15	5	5,3	20	6	6

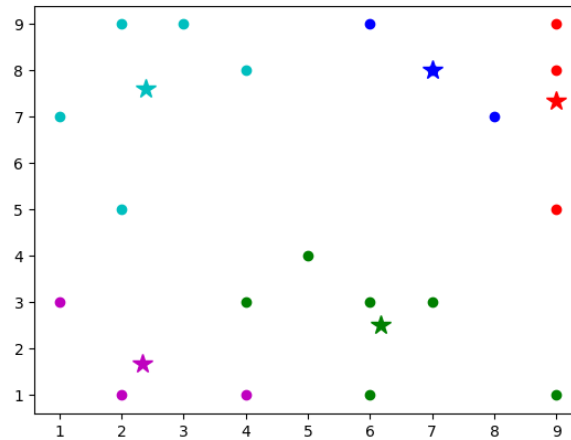


Figura 7.19 Agrupamento formado após a primeira iteração do algoritmo k-Means-Yinyang.

Iteração 2: No início da segunda iteração é calculado o $\delta(c_j)$, ou seja, o deslocamento do centroide de sua posição inicial para a posição atual, c'_j para c_j . Em seguida é calculado o valor do $\delta(GC_f)$, ou seja, o maior deslocamento realizado por um centroide pertencente ao grupo GC_f , $\delta(GC_f) = \max\{\delta_j\}$, $c_j \in GC_f$.

Para cada instância $x_i \in CI$, são atualizados os valores do limite superior $u(x_i) = u'(x_i) + \delta(c(x_i))$ e os limites inferiores $l_{im}(x_i, GC_f) = l'_{im}(x_i, GC_f) + \delta(GC_f)$. A Tabela 7.26 apresenta os valores atualizados do limite superior e a Tabela 7.27 os limites inferiores.

Tabela 7.26 Valores do limite superior após atualização no início da execução da segunda iteração.

#ID	1	2	3	4	5	6	7	8	9	10
u	3,5	2,6	2	0,9	1,5	0,7	0,7	0,9	1,7	1,2
#ID	11	12	13	14	15	16	17	18	19	20
u	1,5	1,6	4,5	2,2	1,9	1,6	1,6	0,7	1,4	3,6

Tabela 6.27 Valores dos limites inferiores após atualização no início da execução da segunda iteração.

#ID	GC ₁	GC ₂	#ID	GC ₁	GC ₂	#ID	GC ₁	GC ₂	#ID	GC ₁	GC ₂
1	3,9	3,7	6	1,9	3,8	11	1,9	5,4	16	5,4	2,9
2	3,6	2,7	7	2,1	3,1	12	2,2	4,1	17	3,6	1,6
3	2,3	2,9	8	1,2	4,5	13	4,8	5,7	18	0,9	4,7
4	2,9	3,1	9	2,5	5,4	14	2,8	2,3	19	2,9	5,9
5	4,9	2,7	10	5,8	3,1	15	2,8	4,1	20	3,9	4,8

Na sequência, o processo verifica cada instância x_i no chamado filtro global isto é, $u(x_i) < \min\{l_{im}(x_i, GC_f)\}$ (*). Se a condição for verdadeira a instância x_i esta corretamente

atribuída ao grupo e não são necessárias novas verificações, passando, assim, à análise da próxima instância. Caso a condição (*) seja falsa é realizada a atualização do $u(x_i) = d(x_i, c(x_i))$ e se faz necessário uma segunda verificação da condição (*). Se a condição se mostrar falsa nas duas verificações (*) a instância será analisada no filtro de grupo. No exemplo sendo considerado 9 instâncias não passaram pela primeira análise do filtro global e 2 instâncias não passaram pela segunda análise, ou seja, um total de 11 instâncias de 20 estavam corretamente atribuídas aos grupos, sem necessidade da análise dos demais níveis do filtro.

As 9 instâncias restantes precisam passar pelo filtro de grupo, onde para cada instância x_i são analisados quais grupos de centroides satisfazem a condição $l_{im}(x_i, GC_f) < u(x_i)$. Nos casos em que essa condição for falsa, todos os centroides que fazem parte desse grupo não precisam ser analisados, pois não podem estar mais próximos que o atual centroide ao qual a instância está atribuída. Considerando, por exemplo, a instância I_7 , que está atribuída ao centroide c_1 , somente será necessário verificar a possibilidade de a instância ser atribuída a um novo centroide para os centroides pertencentes ao grupo de centroides $GC_1 = \{c_2, c_5\}$, pois para o grupo $GC_2 = \{c_1, c_3, c_4\}$ é certo que nenhum de seus centroides pode estar mais perto que o centroide c_1 ao qual a instância já está atribuída.

Por fim, no filtro local, para cada instância x_i que passou pelo filtro global e para cada centroide pertencente aos grupos GC_f que passaram pelo filtro de grupo é verificado se a $l_{im}(x_i, GC_f) > l'_{im}(x_i, GC_f) - \delta c(x_i)$. Nos casos em que essa condição for falsa é necessário a $d(x_i, c_j)$ e verificar se $u(x_i) > d(x_i, c_j)$, nos casos essa condição for verdadeira a instância será atribuída ao novo centroide c_j e que seja atualizado o limite superior $u(x_i) = d(x_i, c_j)$. Caso a condição seja falsa é verificado se $d(x_i, c_j) < l_{im}(x_i, GC_f)$. Caso essa condição seja verdadeira é atualizado o valor do limite inferior $l_{im}(x_i, GC_f) = d(x_i, c_j)$.

Para o exemplo sendo considerado apenas 2 instâncias mudaram de grupo aos quais estavam atribuídas. A Figura 7.20 ilustra o agrupamento ao final da iteração 2. Ao final da iteração 2 são recalculadas as posições dos centroides com base nas instâncias atribuídas aos grupos G_j . Foram realizados 24 cálculos de distância na segunda iteração.

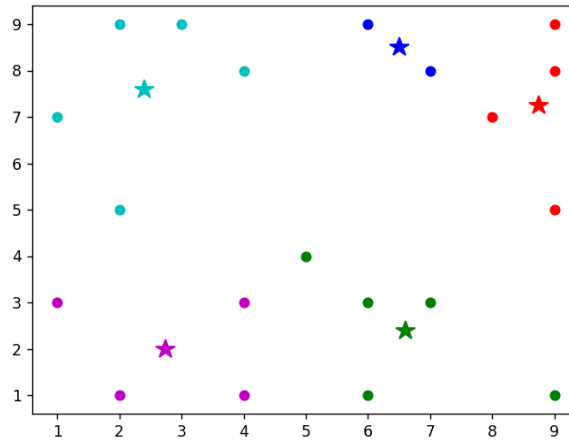


Figura 7.20 Agrupamento formado após a segunda iteração do algoritmo k-Means-Yinyang.

Iteração 3: Iniciando a iteração 3 é atualizado o $\delta(c_j)$, ou seja, são calculados os deslocamentos dos centroides da posição anterior para a atual, c'_j para c_j . Em seguida são atualizados os valores de $\delta(GC_f)$ com o maior deslocamento entre os centroides pertencentes ao grupo GC_f , $\delta(GC_f) = \max\{\delta_j\}$, $c_j \in GC_f$. Após os valores de $\delta(c_j)$ e $\delta(GC_f)$ terem sido obtidos, são atualizados os valores dos limites superiores $u(x_i) = u'(x_i) + \delta c(x_i)$ e inferiores $l_{im}(x_i, GC_f) = l'_{im}(x_i, GC_f) - \delta(GC_f)$ para cada instância. A Tabela 7.28 apresenta o valor do limite superior de cada instância no início da terceira iteração e a Tabela 7.29 apresenta os valores dos limites inferiores de cada instância.

Tabela 7.28 Valores do limite superior após a execução da segunda iteração.

#ID	1	2	3	4	5	6	7	8	9	10
u	3,5	2,6	2,3	0,9	2,9	0,7	1,3	0,9	1,9	1,2
#ID	11	12	13	14	15	16	17	18	19	20
u	2,4	1,6	4,5	2,5	1,9	2,3	2,6	2,1	1,4	3,6

Tabela 7.29 Valores dos limites inferiores após a execução da segunda iteração.

#ID	GC ₁	GC ₂	#ID	GC ₁	GC ₂	#ID	GC ₁	GC ₂	#ID	GC ₁	GC ₂
1	3,6	4	6	0,6	3,1	11	2,7	6	16	1,4	6,2
2	3	2,8	7	0,7	4,3	12	1,6	5,2	17	1,5	4,3
3	3	2,2	8	0,6	5	13	5,6	5	18	1,7	2
4	3,7	2,4	9	1,5	5	14	2,2	6	19	3,7	6
5	5,6	2	10	3	6	15	3,6	3,3	20	4,7	4

Após a atualização dos limites superiores e inferiores, cada instância passou pelo filtro global. Com isso 9 delas não passaram pela primeira verificação e 2 não passaram pela segunda verificação do filtro global e não precisaram ser analisadas no filtro de grupo.

As 9 instâncias restantes passaram pelo filtro de grupo para eliminar grupos de centroides que não satisfazem sua condição e são encaminhados para o filtro local. No exemplo sendo considerado nenhuma instância mudou de grupo ao final da iteração 3. Dessa forma, o algoritmo k-Means-Yinyang finaliza. Foram realizados 22 cálculos de distância na iteração 3 e a Figura 7.21 apresenta o grupamento final induzido.

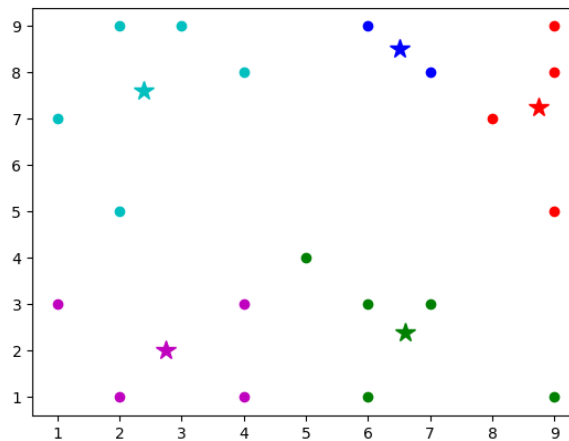


Figura 7.21 Agrupamento final induzido pelo algoritmo k-Means-Yinyang.

O Algoritmo k-Means-Yinyang realizou 166 cálculos de distância para o exemplo considerado, contra 300 que seriam realizados pelo k-Means padrão.

7.6.2 Propostas de Modificação do k-Means-Yinyang

O algoritmo k-Means-Yinyang é uma proposta que apresenta boa performance quando comparada com outras propostas de melhoria do algoritmo k-Means. Dessa forma, autores como Newling e Fleuret, em [Newling & Fleuret 2016] e Kwedlo, em [Kwedlo 2017] apresentaram propostas de modificação do k-Means-Yinyang na tentativa de melhorar sua performance.

A primeira proposta de modificação é chamada de “*Simplified Yinyang k-Means*” e está descrita em [Newling & Fleuret 2016]. Essa proposta sugere retirar o filtro local do algoritmo, alegando que até o momento não está claro se há benefícios no seu uso. Dessa forma, cada instância seria inicialmente analisada no filtro global. Para as instâncias que passarem pelo filtro global é verificado quais grupos poderiam estar mais próximos da

instância, ou seja, o filtro de grupo. Para os grupos que passarem pelo filtro de grupo todos os centroides pertencentes aos grupos GC_f a distância da instância ao centroide deve ser calculada.

Kwedlo, em Kwedlo, em [Kwedlo 2017] apresenta duas modificações para o algoritmo k-Means-Yinyang. A primeira modificação sugere que sejam reagrupados os grupos de centroides de forma a manter a coesão deles, pois a cada iteração a posição dos centroides é atualizada, causando a redução da eficiência do filtro de grupo. Dessa forma é possível manter a máxima eficiência do filtro de grupo, recalculando os grupos de centroides. Porém, essa atualização dos grupos de centroides obriga, também, a atualizar os limites superiores e inferiores das instâncias, criando, assim, um custo antes inexistente. Para minimizar o custo de reagrupamento o autor sugere que os reagrupamentos sejam realizados em determinadas iterações, seguindo algum parâmetro definido pelo usuário. Tal método, apesar de lógico, não apresentou ganho de performance nos testes realizados pelo autor.

A segunda proposta de modificação do algoritmo realizada por Kwedlo é a de utilizar uma variante do k-Means para agrupar e reagrupar os centroides. Essa variante o autor denomina de *same-size K-means*, e alega que essa modificação do algoritmo k-Means melhora a eficiência do filtro de grupo devido a usar grupos de centroides balanceados. O uso dessas duas propostas de modificação tende a reduzir o tempo de execução do algoritmo; porém, utilizando instâncias reais, em geral, com menor precisão nos valores, identificou-se que nem sempre o agrupamento final e o número de iterações são os mesmos do algoritmo k-Means padrão. Segundo o autor a causa mais provável desse comportamento é uma ordem diferente de iteração nos grupos de centroides, já que o centroide pode mudar de grupo à qual pertence. No caso de uma distância igual entre uma instância e dois centroides, essa ordem pode influenciar o resultado final do agrupamento.

7.7 k-Means Fission-Fusion

Essa seção aborda em detalhes a proposta descrita em [Yu & Dai 2017]. Na seção são apresentados e discutidos os principais conceitos envolvidos na proposta, são exemplificados alguns dos conceitos utilizados, é detalhado o pseudocódigo do algoritmo e um exemplo do uso do algoritmo é mostrado, identificando os cálculos feitos e o agrupamento obtido a cada iteração.

7.7.1 Considerações Iniciais

A estratégia proposta por Yu e Dai, em [Yu & Dai 2017], identificada nessa Dissertação como algoritmo k-Means-Fission-Fusion, faz uso de dois filtros para evitar realizar cálculos de distâncias entre instâncias e centroides e, assim como as estratégias propostas por Elkan e por Ding e coautores, o k-Means-Fission-Fusion se aproveita de limites superiores e limites inferiores. Inicialmente o k-Means-Fission-Fusion executa uma iteração completa criando um agrupamento inicial e, durante tal iteração, gera os limites superiores e inferiores dos grupos do agrupamento induzido.

A partir da segunda iteração o k-Means-Fission-Fusion utiliza filtros de grupos de instâncias. Nesse filtro os centroides que satisfazem a condição do filtro são descartados por não poderem ser o centroide mais próximo da instância,

A partir da terceira iteração os subgrupos pertencentes aos grupos de instâncias são verificados no filtro de subgrupos, pois os subgrupos possuem limites superiores e limites inferiores mais justos, permitindo, assim, uma filtragem mais eficaz. Somente é calculada a distância entre instâncias pertencentes aos subgrupos e os centroides que não foram descartados por esses dois filtros (grupos e subgrupos).

De forma sucinta, diferentemente do k-Means-Yinyang, que utiliza limites de uma instância para um grupo de centroides, o k-Means-Fission-Fusion utiliza limites de grupos e subgrupos de instâncias, para cada centroide. Portanto, cálculos de distâncias são evitados entre os grupos e subgrupos de instâncias e os centroides que foram filtrados (descartados).

7.7.2 Conceitos Básicos Utilizados no Algoritmo k-Means-Fission-Fusion

Limites superiores e inferiores dos grupos: Os K grupos de instâncias são induzidos na primeira iteração e concomitantemente são gerados os limites superiores e inferiores para cada centroide, isto é, para cada um dos grupos de instâncias é gerado o limite superior e K-1 limites inferiores. O limite superior é dado pela maior distância entre uma instância pertencente ao grupo e o centroide a qual foram atribuídas, ou seja, $u(G_i) = \max_{x \in G_i} d(x, c_i)$, $i = 1, \dots, K$. Os limites inferiores são dados pela menor distância entre uma instância pertencente ao grupo e o centroide c_j , ou seja, $l_{im}(G_i, c_j) = \min_{x \in G_i} d(x, c_j)$, $j=1, \dots, K$ e $i \neq j$.

A atualização dos limites superiores é feita por $u(G_i) = u'(G_i) + \delta(c_i)$, em que $u'(G_i)$ é o valor do limite superior do grupo G_i na iteração anterior à atual. A atualização dos limites inferiores é feita por $l_{im}(G_i, c_j) = l_{im}'(G_i, c'_j) - \delta(c_j)$, em que $l_{im}'(G_i, c'_j)$ é o limite inferior do grupo de centroides G_i até o centroide c_j na iteração anterior à atual. Dessa forma, para cada grupo de instâncias é possível determinar se é necessário calcular a distância entre as instâncias pertencentes ao grupo e um determinado centroide, ou se é possível evitar tais cálculos. A Figura 7.22 ilustra o limite superior e os limites inferiores de um grupo G_1 .

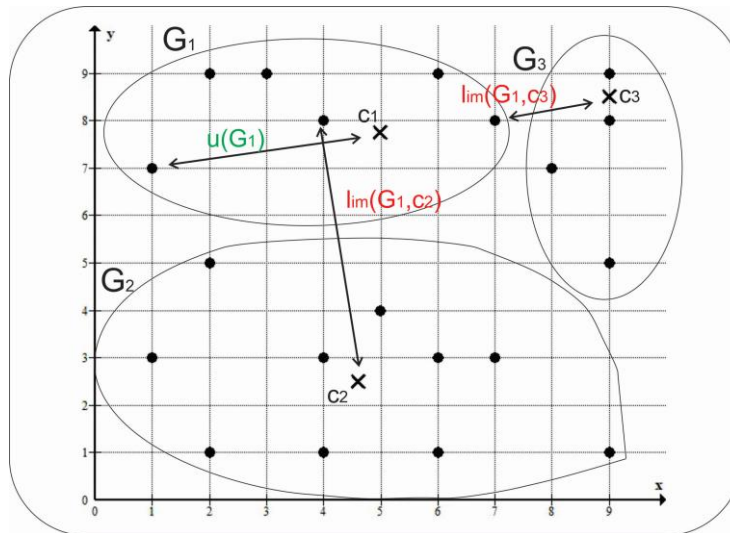


Figura 7.22 Diagrama ilustrando o limite superior e os limites inferiores do grupo G_1 . O limite superior $u(G_1)$ é dado pela maior dentre as distâncias entre o centroide c_1 e as instâncias pertencente ao grupo G_1 . Os limites inferiores $l_{im}(G_1, c_j)$ são dados pela menor dentre as distâncias entre instâncias pertencente a G_1 e o centroide c_j , em que $j \neq 1$.

Filtro de grupos de instâncias: O filtro de grupos de instâncias se caracteriza pela verificação se o limite superior de um grupo G_i é menor que o limite inferior do grupo G_i com relação ao centroide c_j , ou seja, se $u(G_i) \leq l_{im}(G_i, c_j)$. Caso essa condição seja satisfeita, os cálculos de distância entre as instâncias pertencentes ao grupo G_i até o centroide c_j podem ser evitadas.

Fissão dos grupos: A cada nova iteração é comum que algumas instâncias mudem de grupo à qual pertencem, Com base nessa tendência a estratégia proposta por Yu e Dai se propõe a criar subgrupos dentro de cada grupo, de forma que a cada iteração instâncias que mudam de um grupo X para um grupo Y , ficam confinadas em um subgrupo de instâncias em Y , ou seja, se 3 instâncias saírem do grupo G_i para o grupo G_j em uma determinada iteração, elas farão parte de um novo subgrupo dentro de G_j , e esse subgrupo não receberá novas instâncias em iterações subsequentes. Cada grupo pode conter múltiplos subgrupos. A Figura 7.23 ilustra o processo de fissão de grupos durante as iterações.

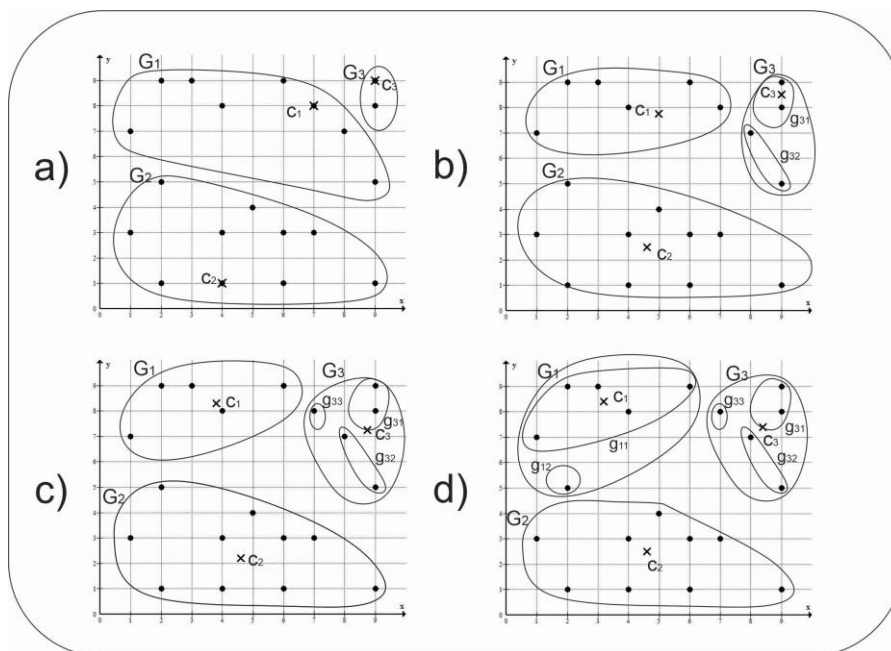


Figura 7.23 Diagrama ilustrando o processo de fissão dos grupos em subgrupos. (a) agrupamento gerado ao final da primeira iteração. (b) duas instâncias que pertenciam a G_1 migraram para G_3 , dando origem ao subgrupo g_{31} , contendo as instâncias originais do grupo G_3 e o subgrupo g_{32} com as instâncias que migraram do grupo G_1 . (c) uma instância que pertencia a G_1 migrou para G_3 , dando origem ao subgrupo g_{33} . (d) uma instância que pertencia a G_2 migrou para G_1 , dando origem ao subgrupo g_{11} com as instâncias originais do grupo G_1 e ao g_{12} com a instância que migrou do grupo G_2 .

Limite superior e inferiores dos subgrupos de instâncias: Assim como os grupos de instâncias, cada subgrupo g_{im} possui seu limite superior e seus limites inferiores para cada centroide. O limite superior é dado pela maior distância de uma instância pertencente ao subgrupo até o centroide à qual foi atribuída, ou seja, $u_{G_i}(g_{im}) = \max_{x \in g_{im}} d(x, c_i)$. Os limites inferiores são dados pela menor distância entre uma instância pertencente ao subgrupo g_{im} e um centroide c_j , em que $j \neq i$ ou seja, $l_{c_i}(g_{im}, c_j) = \min_{x \in g_{im}} d(x, c_j)$. O processo de atualização dos valores dos limites superiores e inferiores segue as mesmas características já descritas no filtro de grupo de instâncias. A Figura 7.24 ilustra o limite superior e os limites inferiores do subgrupo g_{32} .

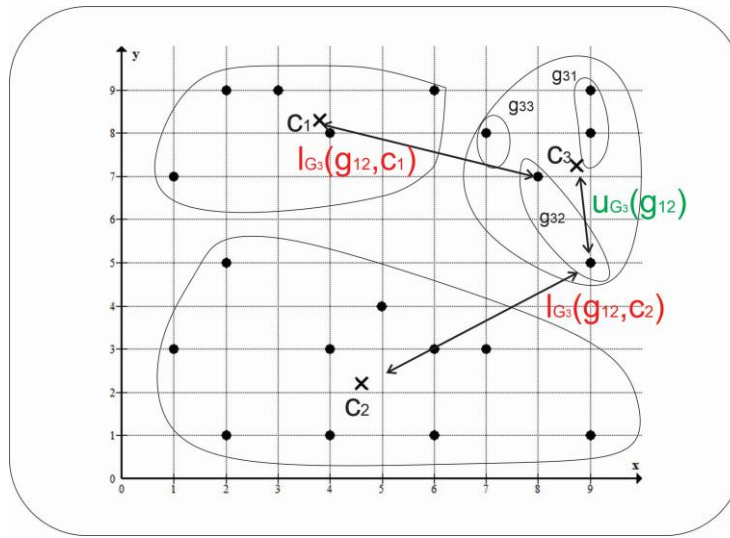


Figura 7.24 Diagrama ilustrando o limite superior e os limites inferiores do subgrupo g_{32} .

Filtro de subgrupos de instâncias: O filtro de grupos de instâncias se caracterizam pela verificação se o limite superior de um subgrupo g_{im} é menor que o limite inferior do grupo g_{im} até o centroide c_j , ou seja, $u_{G_i}(g_{im}) \leq l_{c_i}(g_{im}, c_j)$. Se tal condição for satisfeita os cálculos de distância entre as instâncias pertencentes ao subgrupo g_{im} até o centroide c_j podem ser evitados.

Fusão dos subgrupos: Embora a filtragem dos subgrupos seja bastante eficiente, o tempo de filtragem fica próximo à $O(NK)$ no pior caso, situação em que cada instância se torna um subgrupo. Visando minimizar esse custo os autores da proposta introduziram a fase de fusão dos subgrupos. Assim, se instâncias que pertençam a subgrupos distintos em um grupo G_i migrarem para um novo grupo G_j durante a

iteração, serão atribuídos ao mesmo subgrupo no grupo G_j . Dessa forma existe uma limitação de quantos novos subgrupos podem ser gerados a cada iteração em cada grupo, isto é, cada grupo pode crescer no máximo em k subgrupos por iteração. A complexidade de tempo do filtro passa a ser $O(g'K)$, onde g' é o número de subgrupos após a fase de fusão e $K \leq g' \leq K^2 \leq g$, em que g é o número de subgrupos que seriam gerados sem a fase de fusão.

7.7.3 Algoritmo k-Means-Fission-Fusion

O pseudocódigo do algoritmo k-Means-Fission-Fusion é apresentado no Algoritmo 7.6 com o objetivo de evidenciar em detalhes o seu funcionamento. Parafraseando o pseudocódigo do k-Means-Fission-Fusion, a primeira iteração é realizada em sua versão completa (padrão) do k-Means, em que são realizados todos os cálculos de distância entre as instâncias e os centroides, induzindo, assim, os grupos de instâncias e os limites superiores e inferiores para cada grupo de instâncias.

A partir da segunda iteração, para cada grupo de instâncias é utilizado o filtro de grupos, isto é, para cada grupos G_i de instâncias é verificado quais centroides estão dentro de seus limites, ou seja, $u(G_i) < l_{im}(G_i, c_j)$. Os centroides que satisfizerem essa condição são descartados e não é necessário realizar o cálculo de distâncias entre as instâncias pertencentes ao grupo G_i e o centroide c_j . Na segunda iteração não é necessário utilizar o filtro de subgrupos, uma vez que cada grupo possui apenas um subgrupo que tem as mesmas instâncias do grupo e consequentemente têm os mesmos valores de limites superiores e inferiores. Nessa iteração o uso desse filtro é inócuo.

A partir da terceira iteração são aplicados os filtros de grupos de instâncias e o filtro de subgrupos de instância evitando, com isso, o cálculo de distância entre os subgrupos e os centroides que satisfaçam a condição $u_{G_i}(g_{im}) < l_{c_i}(g_{im}, c_j)$. Para os centroides que passaram pelo filtro de subgrupos é calculada a distância entre as instâncias e o centroide c_j , verificando, dessa forma, se o centroide c_j é o mais próximo da instância que estiver sendo analisada.

Ao final da iteração todas as instâncias que mudaram de atribuição de centroide são atribuídas a um novo subgrupo g_{im} no grupo G_j ao qual foi atribuída. É importante reforçar que instâncias que migraram de grupos distintos são atribuídas a subgrupos

distintos no grupo de destino. Maiores detalhes podem ser encontrados na referência [Yu & Dai 2017].

```

procedure k-Means-Fission-Fusion(C,K,G)

input: C = {x1, x2, ..., xN}    % conjunto com N instâncias de dados a serem agrupadas
      K                        % número de grupos do agrupamento
output: G = {G1, G2, ..., Gk}    % agrupamento final
begin
  % um centróide ci é escolhido para cada grupo Gi (i=1, ..., K)
  {c1, c2, ..., ck} ← escolha_inicial_centroides(C)
  for i ← 1 to N do           % inicia o limite superior e inferior para cada instância
    begin
      % determine a distância entre cada par de centroides e construa a matriz dK×K
      for j ← 1 to K do
        d(xi, cj) ← distancia(xi, cj)
      j ← argminj=1,...,K d(xi, cj)
      b(xi) ← cj % armazena o centróide a qual a instância está atribuída
    end
  for j=1 to K do
    cj ← media_das_instancias_atribuidas(Gj) % atualiza a posição do centróide j
  for i ← 1 to K do % limites superiores e inferiores para cada grupo de instância.
    % considere que, na primeira iteração, cada grupo é também um subgrupo
    begin
      u(Gi) ← maxx ∈ Gi d(x, ci) % limite superior do grupo Gi
      uGi(gim) ← u(Gi) % limite superior do subgrupo gim
      for j ← 1 to K do
        begin
          lim(Gi, cj) ← minx ∈ Gi d(x, cj) % limites inferiores do grupo Gi
          lci(gim, cj) ← lim(Gi, cj) % limites inferiores do subgrupo gim
        end
      end
    end
  repeat
    for i ← 1 to K do δi ← d(ci, c'i) % deslocamento realizado pelo centróide
    for i ← 1 to K do % Atualiza limites de Grupo e Subgrupos
      begin
        u(Gi) ← u(Gi) + δi
        lim(Gi, cj) ← lim(Gi, cj) - δi %atualizar para todos os cj
        for gim ∈ Gi do
          begin
            uGi(gim) ← uGi(gim) + δi
            lci(gim, cj) ← lci(gim, cj) - δi %atualizar para todos os cj
          end
        end
      end
    for i ← 1 to K do
      begin
        grupos ← {}
        for j ← 1 to K do % filtro de Grupos de Instâncias
          begin
            if u(Gi) ≤ lim(Gi, cj) then
              continue
            grupos ← cj % armazenas centróide que não foram descartados
          end
        end
      end
  end

```

Algoritmo 7.5 Parte 1: Pseudocódigo do algoritmo k-Means-Fission-Fusion, adaptado de [Yu & Dai 2017].

```

% Parte 2 : pseudocódigo do algoritmo k-Means-Fission-Fusion
M = {{}}1...K % K conjuntos vazios
centroides ← {} % armazena centroides
for gim ∈ Gi do % para cada subgrupo dos grupos de instância que passaram
    % pelo filtro de grupos de instância
    begin
        for c ∈ grupos do
            if uGi(gim) ≥ lci(gim, c) then
                centroides ← c % Armazena centroides que satisfazem condição
            if centroides = empty do continue
            for x ∈ gim do
                begin
                    ub(x) ← d(x, ci)
                    b(x) ← i
                    b(x)old ← b(x)
                    for c ∈ centroides do
                        if ub(x) > distancia(x, c) do
                            begin
                                ub(x) ← distancia(x, c)
                                b(x) ← c
                            end
                        if b(x) ≠ b(x)old do
                            begin
                                % atribui a instância x no subconjunto Mb(x) removendo do gim
                                atribui(x, b(x))
                                update(ub(x)(Mb(x)), lb(x)(Mb(x)), c) % atualiza o limites do Mb(x)
                            end
                        else
                            update(uGi(gim), lci(gim, c)) % atualiza os limites do subgrupo
                        end
                    end
                end
            end
        for j=1 to K do
            if Mj ≠ empty do atribui(Mj, Gj) % atribui o subconjunto Mj para Gj
        end
    end
    for i=1 to K do
        begin
            for gim ∈ Gi do
                % Atualiza limites de subgrupos levando em consideração a fase de fusão dos grupos
                atualizaLimitesSubgrupos()
                u(Gi) ← max(uGi(gim))
                for j=1 to K do
                    lim(Gi, cj) ← min(lci(gim, cj)) % Para cada cj em que j ≠ i
                end
            end
        for j=1 to K do
            cj ← media_das_instancias_atribuidas(Gj) % atualiza a posição do centroide cj
        until centroides não mudam
    end
    return G = {G1, G2, ..., GK}

```

Algoritmo 7.5 Parte 2: pseudocódigo do algoritmo k-Means-Fission-Fusion, adaptado de [Yu & Dai 2017].

7.7.4 Um Exemplo Completo do Uso do k-Means-Fission-Fusion

Para o desenvolvimento deste exemplo foi utilizado o conjunto de instâncias CI da Tabela 7.23 com 20 instâncias, representadas como pontos no gráfico mostrado na Figura 7.16, em que cada instância é descrita por dois atributos numéricos, A1 e A2, cujos valores foram artificialmente gerados. Tal conjunto é mostrado a seguir, apenas com o intuito de evidenciar as identificações das 20 instâncias.

Inicialmente, o k-Means-Fission-Fusion escolhe aleatoriamente K ($K = 3$) instâncias do conjunto CI como centroides iniciais, representados como estrelas na Figura 7.25 que exibe o conjunto das 20 instâncias consideradas para o *trace* de alto nível que segue.

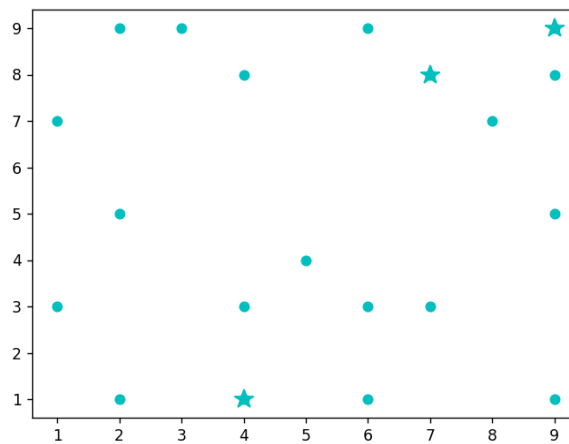


Figura 7.25 As 3 instâncias do conjunto CI que foram escolhidas como centroides iniciais $c_1 = I_5$, $c_2 = I_{15}$, $c_3 = I_8$ são identificadas por estrelas.

Iteração 1: Após as escolhas dos centroides iniciais é executada uma iteração completa (padrão) do algoritmo k-Means, de forma a calcular a distância entre as instâncias e os centroides, induzindo, assim, um agrupamento. Foi induzido um agrupamento com 3 grupos de instâncias, ilustrado na Figura 7.26, sendo G_1 na cor azul, G_2 na cor verde e G_3 na cor vermelha. Os limites superiores e inferiores dos três grupos foram calculados e estão apresentados na Tabela 7.29 e Tabela 7.30, respectivamente.

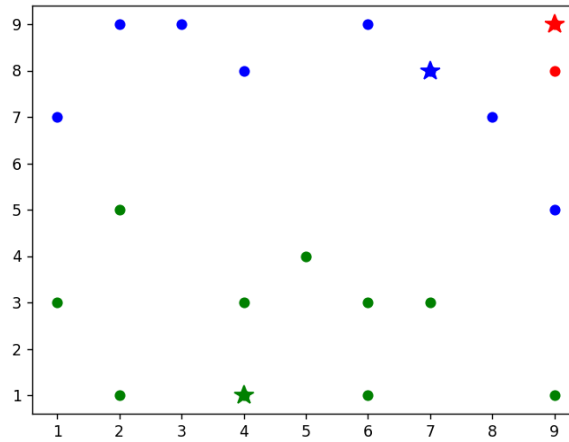


Figura 7.26 Diagrama ilustrando a primeira iteração do algoritmo k-Means-Fission-Fusion.

Tabela 7.30 Valores dos limites superiores dos grupos de instâncias após a execução da primeira iteração.

	G_1	G_2	G_3
$u(G_i)$	6,0	5,0	1,0

Tabela 7.31 Valores dos limites inferiores dos grupos de instâncias após a execução da primeira iteração.

G	c_1	c_2	c_3
1	—	6,4	2,23
2	4,47	—	6,32
3	2,0	8,6	—

Em cada grupo foi criado um subgrupo contendo todas as instâncias pertencentes ao grupo $g_{11} = G_1$, $g_{21} = G_2$ e $g_{31} = G_3$, e foram replicados os valores dos limites superiores e inferiores para os subgrupos. Sendo assim, $u_{G_1}(g_{11}) = u(G_1)$, $u_{G_2}(g_{21}) = u(G_2)$ e $u_{G_3}(g_{31}) = u(G_3)$. O mesmo acontece com os limites inferiores de cada um dos grupos. Os grupos ao final da iteração 1 ficaram da seguinte forma: $G_1 = \{6,7,11,12,14,18,19,20\}$, $G_2 = \{1,2,3,4,5,10,13,15,16,17\}$ e $G_3 = \{8,9\}$. Por fim, são atualizadas as posições dos centroides, formando um agrupamento, como ilustrado na Figura 7.27. Foram realizados 60 cálculos de distância nessa iteração.

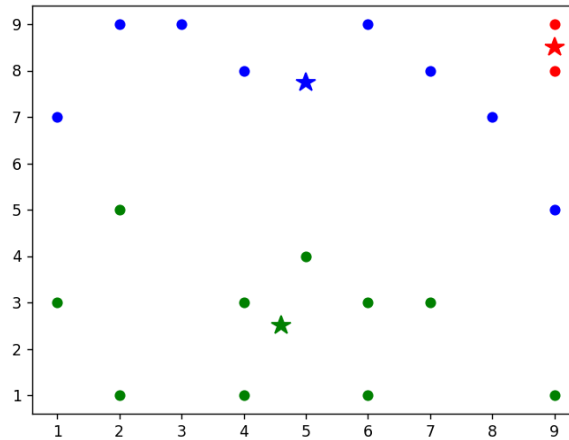


Figura 7.27 Diagrama ilustrando o agrupamento formado ao final da primeira iteração do algoritmo k-Means-Fission-Fusion.

Iteração 2: No início da iteração 2 é calculado o deslocamento que ocorreu nos centroides, de suas posições iniciais até suas posições atuais, isto é, δ_j , $j=1,...,K$. Com os valores dos deslocamentos de cada centroide é realizada a atualização dos limites superiores do grupos e subgrupos, somando o valor de $\delta_{c(x)}$ ao limite superior e subtraindo o valor δ_j dos limites inferiores dos grupos e subgrupos.

Após a atualização dos limites o filtro de grupo de instâncias é executado, de forma que para cada grupo de instância é verificado quais centroides podem ser descartados, ou seja, $u(G_i) < \lim(G_i, c_j)$ (*). Para o grupo G_1 (azul) de instâncias os centroides c_2 e c_3 não satisfizeram a condição (*), ou seja, nenhum centroide no filtro de grupo de instâncias na primeira iteração para o grupo G_1 foi descartado. O mesmo ocorreu com o grupo G_2 (verde). Já para o grupo G_3 (vermelho) o centroide c_2 satisfaz a condição (*) e foi descartado, isto é, não foi necessário realizar o cálculo de distância das instâncias do grupo G_3 para o centroide c_2 .

Nessa iteração a análise feita pelo filtro de subgrupos pode ser dispensada pois, nesse momento cada grupo possui apenas um subgrupo contendo todas as instâncias pertencentes ao grupo, logo o limite superior e os limites inferiores de cada grupo são iguais ao valor do limite superior e limites inferiores de seu único subgrupo.

Sendo assim, para cada instância x pertencente ao grupo G_1 e G_2 foi necessário calcular a distância até o centroide à qual pertence *i.e.*, $c(x)$, e aos centroides dos demais grupos c_j .

Para as instâncias pertencente ao grupo G_3 foi necessário calcular a distância para o centroide c_3 à qual a instância está atribuída e o centroide c_1 , sendo que o cálculo das distâncias das instâncias para o centroide c_2 é descartado.

Dessa forma, duas instâncias migraram do grupo G_1 para o grupo G_3 . Agora o grupo G_3 possui 2 subgrupos, nomeados g_{31} e g_{32} , em que g_{32} possui as instâncias que migraram do grupo G_1 .

Ao final da segunda iteração os grupos formados são: $G_1 = \{6,11,12,18,19,20\}$, $G_2 = \{1,2,3,4,5,10,13,15,16,17\}$ e $G_3 = \{\{8,9\}\{7,14\}\}$. O limite superior do grupo G_3 , $u(G_3)$, é atualizado com o maior limite superior dos subgrupos que lhe pertence e os limite inferiores com o menor limite inferior dos subgrupos que lhe pertence. A Figura 7.28 ilustra o agrupamento formado ao final da iteração 2 com os centroides atualizados. Tabela 7.31 e Tabela 7.32 apresentam os valores dos limites superiores e inferiores dos grupos de instâncias, respectivamente.

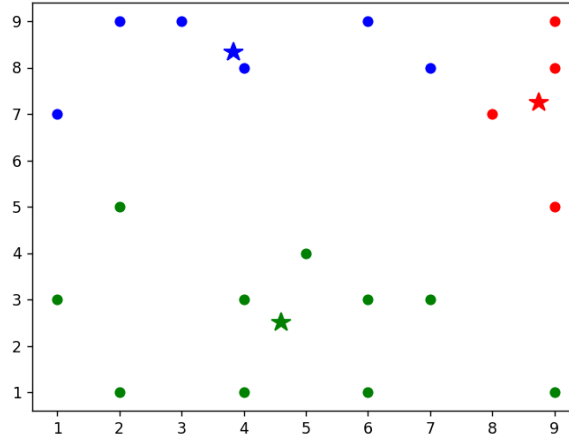


Figura 7.28 Diagrama ilustrando o agrupamento formado ao final da primeira iteração do algoritmo k-Means-Fission-Fusion.

Tabela 7.32 Valores dos limites superiores dos grupos de instâncias após a iteração 2.

	G_1	G_2	G_3
$u(G_i)$	4,0	4,6	3,5

Tabela 7.33 Valores dos limites inferiores dos grupos de instâncias após a iteração 2.

G	c ₁	c ₂	c ₃
1	–	4,5	2
2	3,75	–	3,7
3	3	4,5	–

Como o grupo G_3 possui mais de um subgrupo, os limites superiores e inferiores dos subgrupos estão apresentados na Tabela 7.33. Foram realizados 69 cálculos de distância nessa iteração.

Tabela 7.34 Valores dos limites inferiores dos subgrupos de instâncias pertencentes ao grupo G_3 após a iteração 2.

g_{im}	$l(g_{im}, c_1)$	$l(g_{im}, c_2)$	$u(g_{im})$
31	3	6,9	1,5
32	3	4,5	3,5

Iteração 3: No início da iteração 3 é realizado o cálculo de deslocamento de cada centroide, *i.e.*, o cálculo de δ_j e feita a atualização dos limites superiores e limites inferiores dos grupos e subgrupos de instâncias. Cada grupo de instâncias passa pelo filtro de grupos de instâncias e pelo filtro de subgrupos de instâncias. Ao final da iteração são formados os seguintes grupos de instâncias $G_1 = \{11,12,18,19,20\}$, $G_2 = \{1,3,4,5,10,13,15,16,17\}$ e $G_3 = \{\{8,9\}\{7,14\}\{6\}\}$. A Figura 7.28 ilustra o agrupamento formado ao final da iteração e a Tabela 7.34 e Tabela 7.35 apresentam os valores dos limites superiores e inferiores dos grupos, respectivamente. Foram realizados 47 cálculos de distância nessa iteração. Será suprimida a apresentação dos valores dos limites superiores e inferiores dos grupos e subgrupos, para evitar excesso de informações no exemplo.

Iteração 4: No início da iteração 4 é realizado o cálculo de deslocamento de cada centroide, *i.e.*, o cálculo de δ_j , e é feita a atualização dos limites superiores e limites inferiores dos grupos e subgrupos de instância.

Cada grupo de instâncias passa pelo filtro de grupos de instância e pelo filtro de subgrupos de instância. A instância 2 migrou do grupo G_2 para o grupo G_1 . Ao final da iteração são formados os seguintes grupos de instâncias: $G_1 = \{\{11,12,18,19,20\}\{2\}\}$, $G_2 = \{1,3,4,5,10,13,15,16,17\}$ e $G_3 = \{\{8,9\}\{7,14\}\{6\}\}$. A Figura 7.29 ilustra o agrupamento formado ao final da iteração. Foram realizados 36 cálculos de distância nessa iteração.

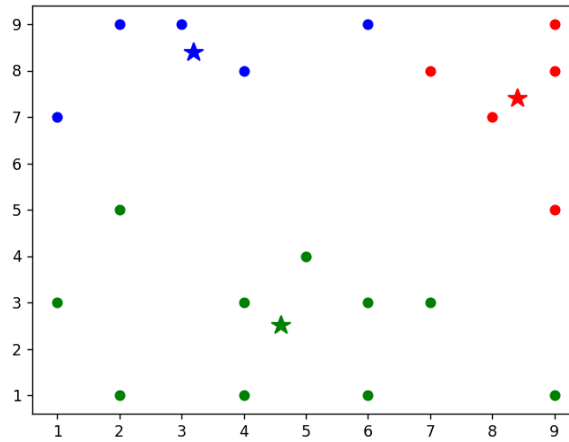


Figura 7.29 Diagrama ilustrando o agrupamento formado ao final da iteração 3 do algoritmo k-Means-Fission-Fusion.

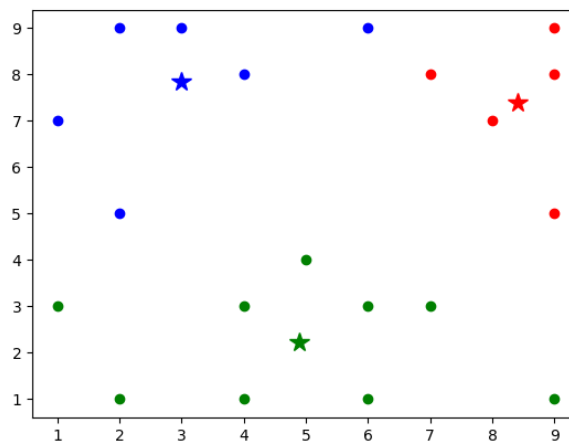


Figura 7.30 Diagrama ilustrando o agrupamento formado ao final da iteração 4 do algoritmo k-Means-Fission-Fusion.

Iteração 5: No início da iteração 5 é realizado o cálculo de deslocamento de cada centroide, i.e., o cálculo de δ_j , e é feita a atualização dos limites superiores e limites inferiores dos grupos e subgrupos de instâncias.

Cada grupo de instâncias passa pelo filtro de grupos de instância e pelo filtro de subgrupos de instância. A instância 18 migrou do grupo G_1 para o grupo G_3 . Ao final da iteração são formados os seguintes grupos de instâncias: $G_1 = \{\{11,12,19,20\}\{2\}\}$, $G_2 = \{1,3,4,5,10,13,15,16,17\}$ e $G_3 = \{\{8,9\}\{7,14\}\{6\}\{18\}\}$. A Figura 7.30 ilustra o

agrupamento formado ao final da iteração. Foram realizados 31 cálculos de distância nessa iteração.

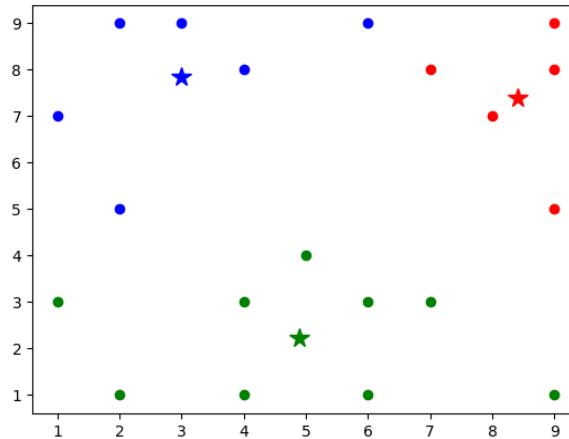


Figura 7.31 Diagrama ilustrando o agrupamento formado ao final da iteração 4 do algoritmo k-Means-Fission-Fusion.

Iteração 6: No início da iteração 6 é realizado o cálculo de deslocamento de cada centroide, *i.e.*, o cálculo de δ_j , e é feita a atualização dos limites superiores e limites inferiores dos grupos e subgrupos de instâncias.

Cada grupo de instâncias passa pelo filtro de grupos de instâncias e pelo filtro de subgrupos de instância. Como nenhuma instância migrou de grupo, o algoritmo k-Means-Fission-Fusion finaliza. O agrupamento final formado pelo algoritmo pode ser visualizado na Figura 7.30. Foram realizados 31 cálculos de distância nessa iteração.

Um total de 274 cálculos de distância foram realizados na execução do k-Means-Fission-Fusion contra um total de 360 pelo k-Means padrão. Em geral, o k-Means-Fission-Fusion apresenta performance ruim em situações em que o agrupamento a ser induzido tem um número pequeno de grupos, quando comparado com as demais estratégias de aceleração apresentadas. Particularmente, em alguns casos, pior até que o k-Means padrão, ganhando destaque, entretanto, à medida que o número de grupos cresce.

Capítulo 8

Sistema *5Accelerated k-Means*, Experimentos e Análise dos Resultados

8.1 Considerações Iniciais

Este capítulo está dividido em duas seções: a Seção 8.2, com foco em uma plataforma computacional desenvolvida durante o trabalho de pesquisa descrito nessa Dissertação e a Seção 8.3 apresenta um conjunto de experimentos relativos ao uso de métodos de aceleração do k-Means.

A Seção 8.2 apresenta e descreve a arquitetura e as principais funcionalidades de uma plataforma computacional, implementada em Python, nomeada de *5Accelerated k-Means* (*5ACCK-Means*), desenvolvida com o intuito de agregar a implementação de cinco algoritmos de aceleração do k-Means disponíveis na literatura, bem como um conjunto de funcionalidades computacionais básicas, relativas a dados, em um sistema computacional único. O *5ACCK-Means* disponibiliza, além das cinco implementações de algoritmos voltados à aceleração do k-Means, a leitura do arquivo contendo as instâncias, o pré-processamento, a visualização em um gráfico, e pôr fim ao uso de índices de validação interna e externa no agrupamento induzido.

A Seção 8.3 apresenta inicialmente os conjuntos de dados utilizados e, também, a metodologia adotada nos experimentos realizados sob a plataforma *5ACCK-Means*, com o objetivo de avaliar as cinco propostas de aceleração do k-Means e analisa os resultados obtidos.

8.2 Sistema *5Accelerated k-Means* (*5ACCK-Means*)

O *5ACCK-Means* foi desenvolvido utilizando a linguagem de programação Python 3.6 e a biblioteca PyQt (interface gráfica) por meio da plataforma de desenvolvimento PyCharm e sob o sistema operacional Microsoft Windows. Como pré-requisito para o

uso do *5ACCK-Means* é necessário a instalação do compilador do Python 3.6. O *5ACCK-Means* está dividido em quatro módulos:

- *Módulo Leitura do conjunto de instâncias*: permite a leitura de arquivos do tipo texto contendo um conjunto de instâncias. Obtendo a localização do arquivo de texto o usuário deve informar o separador de atributos e se o arquivo possui cabeçalho contendo o nome dos atributos. Após a importação das instâncias, as cinco primeiras instâncias são exibidas na tela, com vistas à uma identificação/conferência rápida, pelo usuário, do arquivo de dados a ser utilizado.
- *Módulo Pré-Processamento*: nesse módulo o usuário pode verificar se existem problemas com relação ao conjunto de instâncias carregado, entre eles, valores qualitativos, identificadores numéricos, etc. Nesses casos o módulo disponibiliza ao usuário duas ações: (1) excluir o atributo com problema, (2) transformar dados qualitativos em discretos, utilizando a função aqui chamada de *Encoder*. O módulo também disponibiliza a possibilidade de normalizar o conjunto de instâncias, ou seja, converter os valores para uma faixa comum entre eles, com o intuito de evitar distorções.

No módulo de pré-processamento também é possível informar se existe um atributo que apresenta valores de classe; a informação da classe é usada para o cálculo de índices de validação.

- *Módulo de Execução*: esse módulo é o núcleo do sistema, uma vez que nele ocorrem a escolha da estratégia a ser executada, e define-se o número k de grupos que serão induzidos. Esse módulo é também responsável pela apresentação dos resultados da execução do algoritmo escolhido, entre eles: (1) os rótulos de cada instância, (2) os centroides gerados ao final da execução, (3) o número de iterações, (4) o número de cálculos de distâncias e o (5) tempo de execução. O módulo também disponibiliza a opção ao usuário de apresentar em um gráfico do agrupamento induzido, utilizando duas dimensões.
- *Módulo Índices de Validação*: nesse módulo é possível escolher os índices de validação interna e externa que serão executados sobre o agrupamento induzido. Os índices de validação externa necessitam de classe previamente definida no módulo de pré-processamento. Os índices de validação interna implementados e

disponibilizados nesse módulo são: (1) índice Dunn, (2) índice Silhouette, (3) índice DB score. Os índices de validação externa implementados são: (1) índice Rand, (2) índice Jaccard.

8.2.1 Módulo de Leitura do Conjunto de Instâncias

O módulo de leitura do conjunto de instâncias, apresentado na Figura 8.1, permite a leitura de arquivos do tipo texto (dataset) que possuem um conjunto de instâncias, onde, em geral, os atributos estão separados por caracteres especiais, tais como: “;”, “-”, “.”, “:”, “\t”, entre outros, e as instâncias são separadas por “\n”, ou seja, quebra de linha.

Inicialmente, por meio do botão “Explorar” (4) é possível informar a localização do arquivo contendo o conjunto de instâncias. Em seguida, o separador de atributos é definido por meio do Combo Box “Separador:” (2) e o fato do arquivo conter ou não cabeçalho com os nomes dos atributos é informado por meio do CheckBox “Arquivo contém Cabeçalho” (3).

Por meio do botão “Importar Arquivo” (2) o carregamento do conjunto de instâncias é feito e os dados que contém são exibidos na Table View (5). A Figura 8.1 mostra um exemplo da tela do módulo de leitura.

8.2.2 Módulo de Pré-processamento

O módulo de pré-processamento, apresentado na Figura 8.2, disponibiliza a implementação das seguintes funcionalidades:

1. *Verificar pendências*: verifica se todos os atributos são do tipo numérico, as inconsistências serão apresentadas no local intitulado de “Pendências:”.
2. *Definir Classe*: permite definir um atributo classe; esse atributo é retirado do conjunto de atributos e reservado para uso quando do cálculo de índices de validação externa.
3. *Excluir Atributo*: permite excluir atributos não desejados, tais como: índices, identificadores, atributos com pouca representatividade, etc.
4. *Encoder Atributo*: permite transformar atributos qualitativos em atributos numéricos.

5. Normalizar: permite normalizar todos os atributos dentro de uma faixa de valor, permitindo assim evitar distorções devido à variância no valor dos atributos.

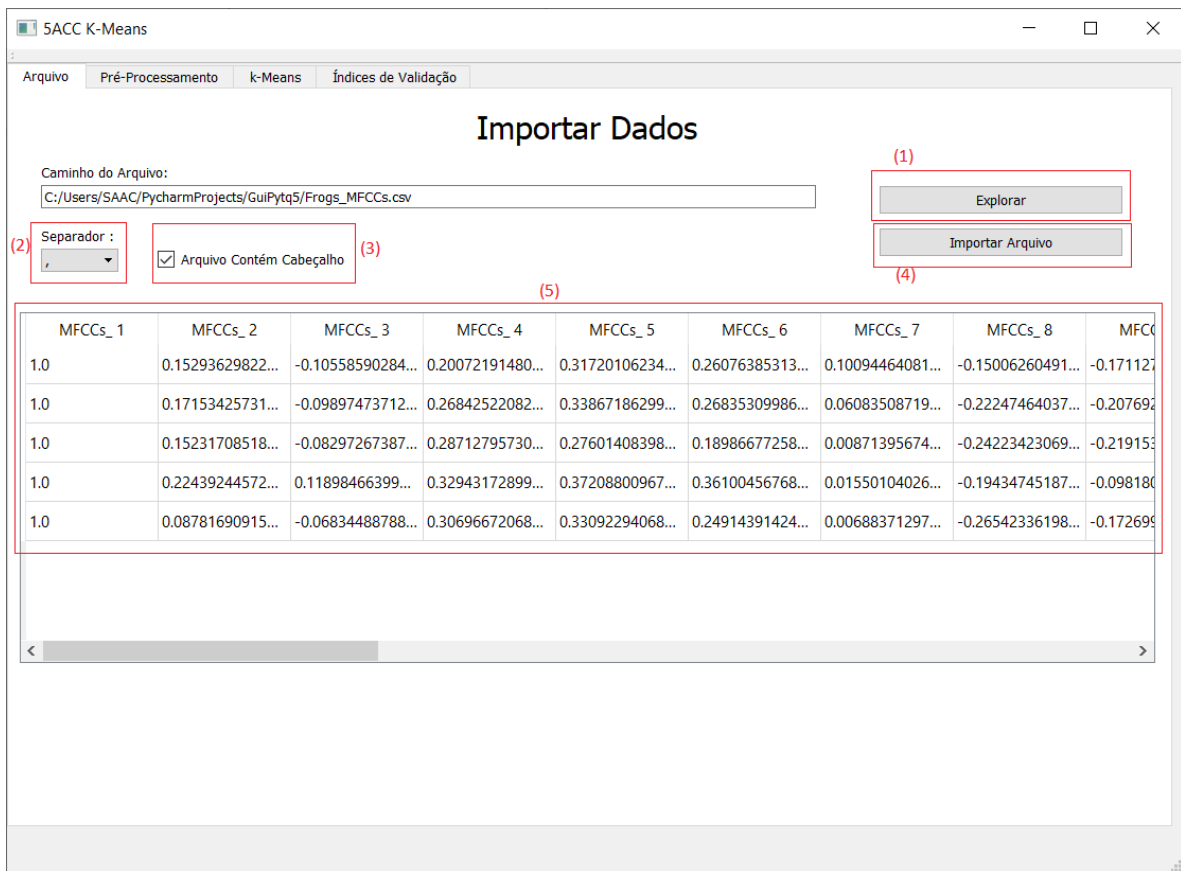


Figura 8.1 Imagem do módulo de leitura após carregamento de arquivo do conjunto de instâncias.

Após o carregamento do conjunto de instâncias é possível passar para o módulo de pré-processamento ou de execução.

As funções estão implementadas no módulo pré-processamento do sistema *5ACCk-Means* como mostrado na Figura 8.2, em que (1) verifica as pendências e mostra na (2) Table View nomeada de “Pendências”, (3) define a coluna que possui a classe das instâncias, (4) normaliza os dados dos atributos, (5) realiza ações de “Excluir” ou “Encoder” sobre um atributo, (6) permite a visualização das instâncias.

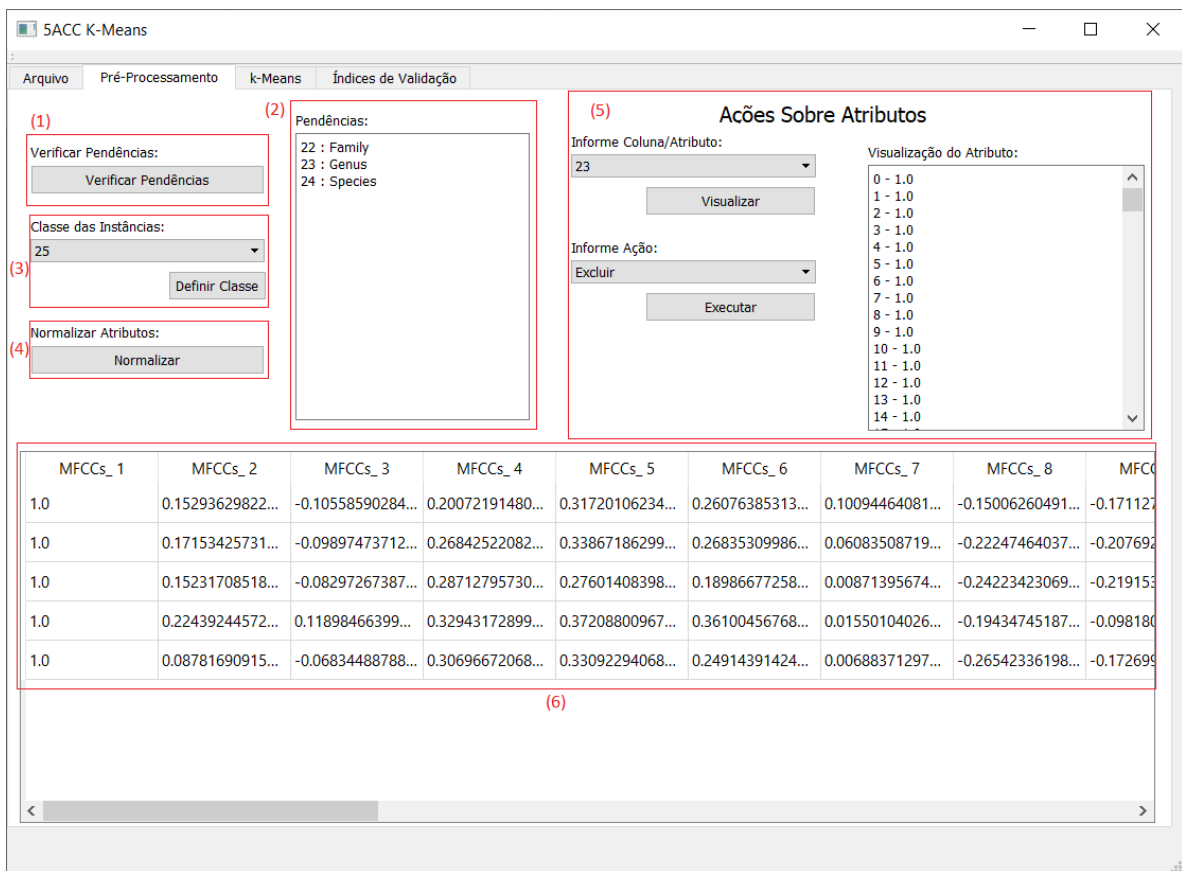


Figura 8.2 Imagem do módulo de pré-processamento após carregamento de arquivo, contendo as marcações sobre as funções disponíveis no módulo.

8.2.3 Módulo de Execução

O módulo de execução do k-Means é o módulo central do sistema *5ACCk-Means*. Nesse módulo é possível escolher a estratégia do k-Means que será executado, além de definir o número de grupos que serão induzidos, visualizar os rótulos e centroides e os dados de execução, tais como, número de iterações, número de cálculos de distância e tempo de execução, além de visualizar o agrupamento induzido em um gráfico de duas dimensões.

As funções do módulo de execução do *5ACCk-Means* estão dispostas da seguinte forma: (1) definir o número K de grupos a ser induzido, que tem por valor *default* K=3, (2) escolher a estratégia de aceleração do k-Means a ser executada, (3) visualizar os rótulos dos grupos induzidos, (4) visualizar os centroides finais induzidos, (5) visualizar os dados associados à execução do algoritmo (iteraões, número de cálculos de distância e tempo de processamento do algoritmo acelerado sendo utilizado), (6) gerar gráfico do agrupamento final induzido. A Figura 8.3 mostra a tela do módulo de execução e a

Figura 8.4 mostra o gráfico gerado no módulo de execução, considerando que o conjunto de dados é aquele apresentado em Tabela 7.1.

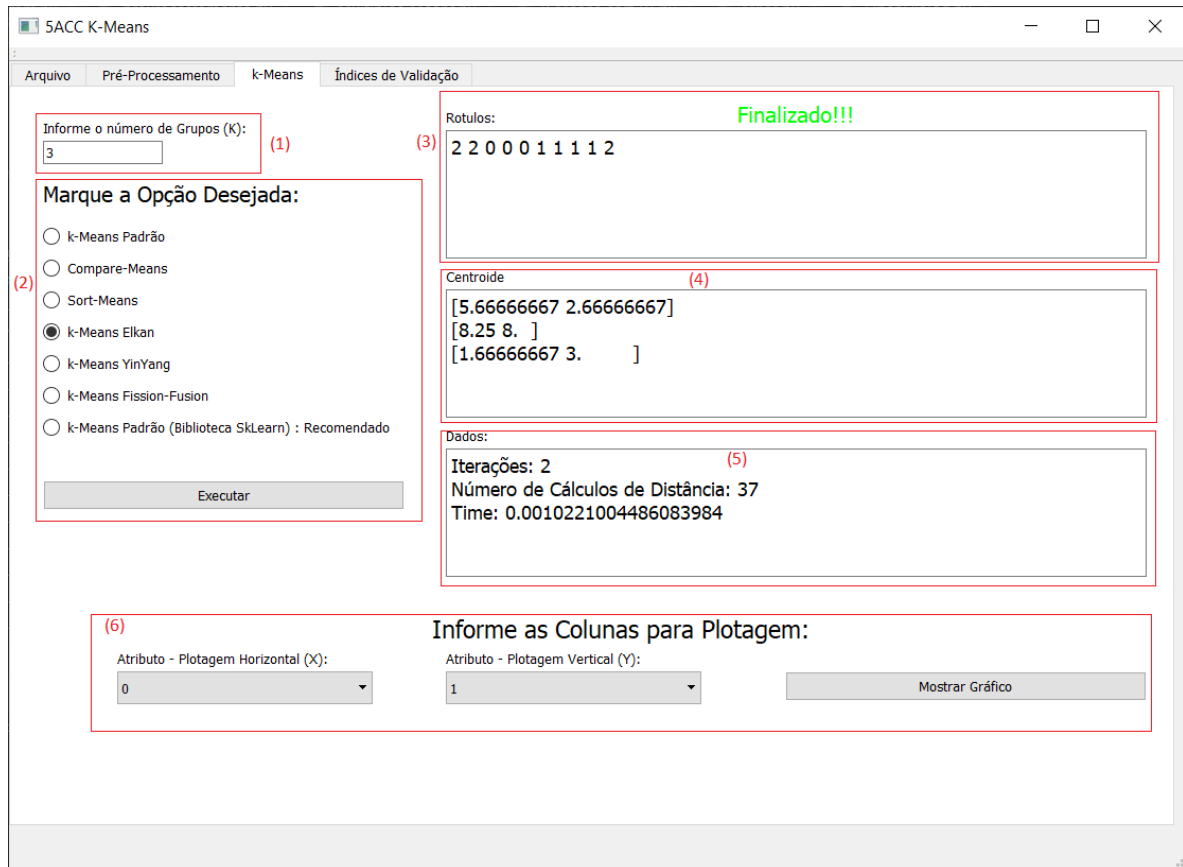


Figura 8.3 Imagem do módulo de execução, contendo as marcações sobre as funções disponíveis no módulo.

8.2.4 Módulo de Índices de Validação

O módulo que contém os índices de validação (interna e externa) permite escolher quais os índices de validação interna e externa serão aplicados sobre o agrupamento induzido. Os índices de validação externa necessitam que, previamente, seja definida a classe das instâncias no módulo de pré-processamento.

Como comentado anteriormente, os índices de validação interna disponíveis são: índice Dunn, índice Silhouette, índice DB score e os índices de validação externa são: índice Rand e índice Jaccard. Após definido o índice que será utilizado, sua execução é feita via botão “Executar”.

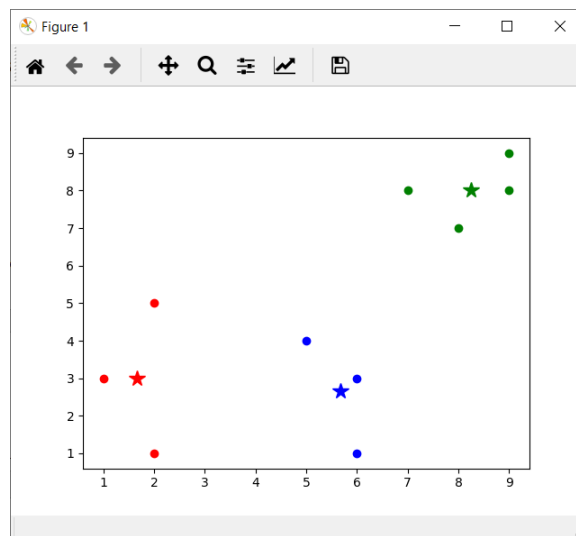


Figura 8.4 Gráfico gerado pelo módulo de execução do *5ACCK-Means*, considerando que o conjunto de dados carregado é o descrito em Tabela 7.1.

Índices de Validação

Validação Interna:

☒ Dunn ☒ Silhouette ☒ DB Score

Validação Externa:

☒ Rand ☒ Jaccard

*Precisa de Classe Definida

Executar

Índice Dunn:0.12126781251816648
Índice Silhouette:0.267728868347859
DB Score:0.954390787343506

Índice Rand:0.5210526315789473
Índice Jaccard:0.1574074074074074

Figura 8.5 Tela do módulo “Índices de Validação” do *5ACCK-Means*.

8.3 Dados, Metodologia, Experimentos e Análise de Resultados

Com o objetivo de apresentar uma descrição detalhada dos experimentos conduzidos, esta seção está organizada em três subseções. Na Seção 8.3.1 são apresentados e descritos os conjuntos de dados utilizados nos experimentos. Na Seção 8.3.2 a metodologia adotada para a condução dos experimentos é apresentada com foco em seu aspecto procedimental. A Seção 8.3.3 aborda os experimentos conduzidos sob o sistema computacional *5ACCK-Means*, desenvolvido durante a pesquisa realizada e descrita nessa Dissertação, bem como fornece uma análise dos resultados obtidos, seguida pelas principais conclusões derivadas do estudo, pesquisa e experimentos conduzidos. Possíveis linhas de pesquisa para fomentar a continuidade do trabalho realizado são sugeridas ao final da Seção 8.3.3.

8.3.1 Dados Utilizados nos Experimentos

Para a condução dos experimentos dois tipos de instâncias de dados foram considerados: (a) instâncias artificialmente criadas, identificadas nesta Dissertação como *instâncias sintéticas* e (b) instâncias de dados reais, identificadas como *instâncias reais*. A principal motivação para o uso de instâncias sintéticas foi a de facilitar o monitoramento da performance de cada uma das cinco estratégias implementadas e disponibilizadas no sistema *5ACCK-Means*.

Os conjuntos de instâncias sintéticas foram gerados utilizando a biblioteca *scikit-learn* do Python [Pedregosa *et al.* 2011]. Essa biblioteca permite a criação de conjuntos de instâncias artificiais com volumes e complexidades controlados. Para a criação optou-se para que as instâncias geradas tivessem os respectivos valores de atributos no intervalo $[-10,10]$ e, também, que as instâncias estivessem embaralhadas dentro do conjunto. A criação dos conjuntos de instâncias foi direcionada pelo objetivo do trabalho *i.e.*, o de avaliar as cinco estratégias de aceleração do k-Means. Para conduzir os experimentos e promover a avaliação dos resultados, duas características relevantes relativas às instâncias foram consideradas: o número de instâncias disponíveis e o número de atributos que descrevem as instâncias.

Foram criados 24 conjuntos de dados sintéticos, em que os respectivos números de instâncias (N) e números de atributos (M), variam nos conjuntos {1.000, 5.000, 10.000}

e {2, 8, 32, 64, 128, 256, 512, 1.024}, respectivamente. A motivação para tal escolha foi a de monitorar o desempenho de cada uma das cinco estratégias de aceleração do k-Means em face ao aumento do volume e da dimensionalidade horizontal das instâncias.

A Tabela 8.1 identifica cada um dos 24 conjuntos de dados sintéticos criados e informa, para cada um deles, seu associado número de instâncias bem como o número de atributos utilizados na descrição de suas instâncias. A Figura 8.9 apresenta, como exemplo, 40 instâncias artificialmente criadas, extraídas de um dos conjuntos de dados sintéticos com 8 atributos.

Tabela 8.1 Características dos 24 conjuntos de dados sintéticos criados, em que #Id é o identificador do conjunto, N é o número de instâncias de #Id e M é o número de atributos que descrevem as instâncias de #Id.

#Id	N	M	#Id	N	M	#Id	N	M
S1	1.000	2	S9	5.000	2	S17	10.000	2
S2	1.000	8	S10	5.000	8	S18	10.000	8
S3	1.000	32	S11	5.000	32	S19	10.000	32
S4	1.000	64	S12	5.000	64	S20	10.000	64
S5	1.000	128	S13	5.000	128	S21	10.000	128
S6	1.000	256	S14	5.000	256	S22	10.000	256
S7	1.000	512	S15	5.000	512	S23	10.000	512
S8	1.000	1.024	S16	5.000	1.024	S24	10.000	1.024

```

1 -3.1848846959755703,-2.5300017904759805,2.189151385828768,-0.12871890866046365,9.129317972763168,-8.18491487035975,-5.874355751097512,-7.264845522523203,
2 -8.962098148733085,6.043581793541629,5.337012549233053,9.326048702395909,11.524127856923094,7.2231118341618314,0.16817238130840517,4.591332376223979,
3 -7.430971782216723,0.7139882911889477,-7.139871693024994,0.803268902900689,0.028511003327325646,-0.6022067787174803,-4.775002983860664,5.062175071858898,
4 1.1182232418672746,3.984458910303295,2.7468062725030644,1.5924128035939429,-2.25250139168549,1.5345183059380674,-2.831194172081232,8.445839394748798,
5 1.342118866463532,5.601612594421976,2.5363826478217133,3.657018773959519,-1.601571991473055,3.176598701562869,-0.9726551007620938,9.270509402430749,
6 -3.4363295615277067,-4.467613598556802,0.5550925672282578,-0.4782200119485628,8.828002502746736,-8.573163012270776,-5.598249494580668,-9.711273386559466,
7 -3.7068099564524057,-3.0681889745948734,0.8014732337204377,-2.735580148823335,0.084302727708918,-6.660133287459887,-6.166250173187659,-6.413748871146226,
8 1.335009538399427,-4.490707931203989,-1.3785437959061548,-4.578966314549132,-6.3617668446426245,-7.185588315368254,-2.0097132166622815,-7.656869999644101,
9 1.9057751900260231,4.8860119188463145,-0.03933554977326681,1.0213855741714435,-1.6570109674156095,3.0118354907186915,-0.3052096874236324,5.095782848452037,
10 -1.423452487081366,0.6208173052311079,-9.35030240032574,2.1429393595416535,2.00239388449812,3.758908750209895,8.170213058510026,4.371770877392778,
11 -1.298334520941654,-1.688564838766388,3.6711337899308725,-9.09414107481766,4.306697193018061,4.578691537627167,-5.976854499755471,-7.414888862283757,
12 -10.63123778257798,6.794064075830281,5.55199991656342,8.74431733860981,0.072383540512584,4.551373506269433,-1.3993934625326314,6.680658037800086,
13 -3.528051700469787,-1.8927366531023421,4.538263269789425,-8.11740189954903,3.924365731052972,4.125116726357025,-7.077244359624842,-7.9820443531776105,
14 9.28700369735612,-4.175870787112491,5.473387626913426,1.186660736712613,-0.2305566535793857,8.515154982296377,-9.636752483843791,-8.813399037848152,
15 -8.94554753682117,8.473336185048435,4.782855150606945,8.85427871377555,9.315497147323365,6.276642681984294,0.2999472620318109,4.890582094438299,
16 -8.29653752783193,2.051504089487206,-7.13777591315412,10.641046571489376,1.494923996483319,-1.1015398418576714,-5.859493393661511,8.039123207435711,
17 -7.477442019763528,1.8179611809883212,-7.700701493339259,8.302218672069325,1.5195578054723273,-1.338757540908962,-4.3399990503217944,5.19835507874307,
18 -2.719137916322646,-2.558337044043618,0.25417326257179584,-1.7857346468397649,12.20003255643075,-5.929040293081034,-5.505246833087483,-5.7317546787192075,
19 1.719501925484526,1.7730018885326213,-9.015882058613544,2.4992711593371144,2.5914910588615854,2.105663480135177,9.169150214225933,2.8353991324912236,
20 -9.87607983266094,5.4721113047607569,6.575303314171797,8.784429689517202,10.824368827408312,4.838478653091873,-0.861679774667539,5.209012850216192,
21 -5.124928205421355,-2.9048657973488785,1.2808927727429347,-0.6897743512584592,10.832954854267733,-9.36042395323187,-7.150068608575422,-8.82012553014842,
22 10.54066684839244,-1.831671390015786,5.772447635854958,1.8370655246689025,2.065002244020228,7.016253459596204,-6.052910595686264,-6.4874926177724515,
23 -0.1980419642241129,-2.0764570397885196,4.182317372968403,-8.66020418327476,5.350643777193398,2.429162601013951,-5.832188658475283,-8.131459649579616,
24 -10.617545375620413,5.501959141140251,4.72702363949624,8.04253908265908,9.83116441199591,7.02341024858403,-0.957103677055973,4.466942129887259,
25 0.31980640325749636,1.4692328221736877,4.172058542916553,-0.7132147435119969,-1.5626720850021265,5.298627612552873,-0.9167990184717762,6.784706489199829,
26 1.9919353600885963,5.004826866613487,1.637790171547844,-0.19983299483203788,0.18540120812139116,2.125767240767973,-2.293780331815616,6.750603956180149,
27 -9.639676948569427,8.611958916164529,6.5054493272739755,5.395117538284234,10.32741652491117,4.586636046761313,-1.5299082453074271,3.559826848968959,
28 -0.4296928377214042,3.71372968157885,1.9447781163669502,-0.7630361519313265,-1.4117561400818133,2.5387346984532,-2.9906119725553806,6.532217261529278,
29 -3.4152597365404356,0.5686544175068455,5.594412227181337,-10.120783847345818,3.458875298334763,3.4796897208161184,-4.498938353374653,-6.252395080338664,
30 -9.015923133303618,2.1857818661819177,-7.514224121627367,7.644429407918803,0.10672854173688967,-2.5415683637240374,-3.4735053658979833,5.240869991545211,
31 9.28649497768812,-2.4531145510847225,6.173560017595722,-0.0117336468292941096,0.46507546146781864,9.0602608963512,-8.480611382226465,-8.060232950771818,
32 -3.7029216054887906,-3.0667526719668023,2.3259723594413284,-0.7731458064053541,10.8779871370104,-8.813149675511262,-6.356702963563428,-6.932533229772118,
33 1.7050548349674155,-4.7193927171890524,-0.06963422552648302,-5.594191027048199,-6.3942952609581525,-7.434443232058237,5.222192491465645,-7.5156831547456235,
34 -0.3396317319650261,3.842202722636812,1.9870259161082466,2.611063815878037,-2.2716588352703453,2.0914437226741085,-1.346708289175826,7.171981729279487,
35 -2.40523779646553,0.39089485687624137,4.201844752360051,-8.011332060788591,1.8529142813720392,2.958211718283343,-6.7646226140163535,-7.47240882735,
36 -5.421477883896283,-3.337821117614383,2.5698024956652885,-0.44309151042362904,9.234092413577358,-6.588911172488847,-6.4288640777382335,-7.629432329484701,
37 0.4801677445639327,4.228982344760049,2.067495050716953,2.466923744738414,-0.8364749888388564,3.7145543696977943,-1.9061818672829456,8.804342654204644,
38 -3.9950601358703604,0.14141672146701012,3.176860383014806,-9.923072181563247,3.168085896686473,2.6644588312851916,-5.499478049246227,-8.617805881774224,
39 -3.501564525048119,-1.4154413383690403,3.931420531610725,-10.89784142895148,2.47685315998085,4.52292425214455,-6.0469225527702175,-7.663099778877362,
40 -10.522994979266452,7.540735772955603,5.038393064034789,6.0149653811691834,9.674545171599252,6.48816006114482,0.5585479926118604,5.828486912850777,

```

Figura 8.9 Exemplo contendo 40 instâncias extraídas de um conjunto de instâncias descritas por 8 atributos, criadas artificialmente, utilizando a biblioteca scikit-learn do Python.

Os 10 conjuntos de instâncias com valores reais foram escolhidos a partir do Repositório de conjunto de instâncias *UCI Machine Learning Repository* [Dua & Graff 2019] e 2

conjuntos de instâncias utilizado por Elkan, em [Elkan 2003] (MNIST [LeCun 1998] e Birch [Zhang *et al.* 1997]). A escolha foi feita com o objetivo de promover a diversidade dos conjuntos escolhidos, em termos de número de instâncias e números de atributos que descrevem as instâncias. A Tabela 8.2 mostra o número de instâncias e o número de atributos que descrevem as instâncias, para cada um dos conjuntos escolhidos, como estão disponibilizados no UCI Repository.

Tabela 8.2 Conjuntos de instâncias de dados reais, extraídos do UCI Repository, em que #Id: identificação do conjunto, N: número de instâncias em #Id e M: número de atributos que descrevem as instâncias em #Id, incluso classes.

#Id	N	M
Poker	1025010	11
Optdigits	5620	64
Frogs	7195	26
Gene	801	20.531
3D	434.874	4
Skin	245.057	4
Absenteeism	740	21
Student	5820	33
Pendigitis	10992	17
Epileptic	11500	178
MNIST	70.000	785
Birch	100.000	2

Segue uma breve descrição de cada um dos 12 conjuntos de instância.

- **Poker Hand (Poker):** conjunto de instâncias de dados reais contendo 1.025.010 instâncias, cada uma descrita por 11 atributos, sendo o último a classe da instância. Cada instância é um exemplo de uma “mão de pôquer”, que é composta por cinco cartas de baralho retiradas de um baralho com 52 cartas. Cada uma das 5 cartas é descrita usando dois atributos (naipe e ranque), o que faz com que uma mão de pôquer” seja descrita por 10 atributos. É importante lembrar que a ordem das cartas na “mão de pôquer” é relevante e que no conjunto Poker não comparecem atributos com valor ausente. O atributo 11, que representa a classe, descreve a "mão de pôquer", cujos possíveis valores variam entre 0 e 9, em que cada valor é interpretado como segue:
0 (*nothing*): não é uma mão de pôquer reconhecida;
1 (*one pair*): um par de mesmo ranque dentre as cinco cartas;
2 (*two pairs*): dois pares de mesmo ranque, dentre as cinco cartas;
3 (*three of a kind*): três cartas de mesmo ranque, dentre as cinco cartas;
4 (*straight*): cinco cartas, ranqueadas na sequência, sem espaçamento entre elas;

- 5: (*flush*): cinco cartas de mesmo naipe;
- 6 (*full house*): um par + *three of a kind* de ranque diferente do par;
- 7 (*four of a kind*): quatro cartas com o mesmo ranque, dentre as cinco cartas;
- 8 (*straigh flush*): *straigh* + *flush*;
- 9 (*royal flush*): (Ás, Rei, Rainha, Valete, 10) + *flush*.

O conjunto de instâncias está dividida em 25.010 instâncias de treinamento e 1.000.000 no conjunto de teste. Para o experimento fora usadas as instâncias de treinamento.

- Optical Recognition of Handwritten Digits (Optdigits): conjunto de instâncias de dados reais contendo 5.620 instâncias, cada uma delas descrita por 64 atributos. Cada atributo tem valor inteiro entre 0 e 16. O último atributo é a classe da instância, cujos possíveis valores estão no conjunto {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}. Esse conjunto contém dados relativos a dígitos manuscritos. No conjunto Optical não comparecem atributos cujo correspondente valor esteja ausente.
- Anuran Calls (Frogs): conjunto de instâncias de dados reais contendo 7.195 instâncias, cada uma delas descrita por 22 atributos e 3 classes. Esse conjunto de instâncias foi criado analisando características acústicas extraídas dos sons de anuros (sapos), incluindo a família, o gênero e os rótulos das espécies. Esse conjunto de instâncias foi criado segmentando 60 registros de áudio pertencentes a 4 famílias (Bufonidae, Dendrobatidae, Hylidae, Leptodactylidae) , 8 gêneros (Adenomera, Ameerega, Dendropsophus, Hypsiboas, Leptodactylus, Osteocephalus, Rhinella, Scinax) e 10 espécies (AdenomeraAndre, AdenomeraHylaedact, Ameeregatrivittata, HylaMinuta, HypsiboasCinerascens, HypsiboasCordobae, LeptodactylusFuscus, OsteocephalusOopha, Rhinellagranulosa, ScinaxRuber). Cada áudio corresponde a uma amostra (um sapo individual), o ID do registro também é incluído como uma coluna extra. Utilizamos a entropia espectral e um método de cluster binário para detectar quadros de áudio pertencentes a cada sílaba. Após a segmentação, foi obtido 7195 sílabas. Esses registros foram coletados em condições reais de ruído (som ambiente). Algumas espécies são do campus da Universidade Federal do Amazonas, outros da Mata Atlântica, no Brasil, e um deles de Córdoba, na Argentina. As gravações foram armazenadas em formato wav com 44,1kHz de

frequência de amostragem e 32bit de resolução, o que permite analisar sinais de até 22kHz. De cada sílaba extraída, 22 MFCCs foram calculados usando 44 filtros triangulares. Esses coeficientes foram normalizados entre -1 e 1.

- Gene expression cancer RNA-Seq (Gene): conjunto de instâncias de dados reais contendo 801 instâncias e 20.531 atributos. Esse conjunto de instâncias é uma extração aleatória de expressões genéticas de pacientes com diferentes tipos de tumor. As amostras (instâncias) são armazenadas em linhas. Atributos de cada amostra são os níveis de expressão do gene RNA-Seq medidos pela illumina HiSeq platform. Um nome fictício (gene_XX) é dado a cada atributo.
- 3D spatial network (3D): conjunto de instâncias contendo 434.874 instâncias com 4 atributos. O conjunto de instâncias representam a rede rodoviária em 3D com informações de elevação altamente precisas da Dinamarca, usadas em algoritmos de roteamento ecológico e roteamento de estimativa de combustível / CO2. O conjunto de instância foi construído adicionando informações de elevação a uma rede de estradas 2D na North Jutland, Dinamarca (cobrindo uma região de 185 x 135 km²). Os valores de altitude foram extraídos de uma enorme nuvem de pontos de varredura a laser disponível na Dinamarca. Essa rede rodoviária 3D acabou sendo usada para comparar vários algoritmos de estimativa de combustível e CO2. Esse conjunto de instâncias pode ser usado por qualquer aplicativo que precise conhecer informações precisas de altitude de uma rede viária para executar roteamento mais preciso. Os atributos das instâncias são:
OSM_ID: OpenStreetMap ID para cada segmento de estrada ou aresta no gráfico.
LONGITUDE: longitude Web Mercator (formato Google)
LATITUDE: latitude Web Mercator (formato Google)
ALTITUDE: Altura em metros.
- Skin Segmentation (Skin): O conjunto de instâncias que representam informações sobre a cor da pele foram coletado por amostragem aleatória dos valores de R, G, B de imagens de rosto de pessoas de várias faixas etárias (jovens, adultos e idosos), grupos de raça (brancos, negros e asiáticos) e sexos obtidos no banco de dados FERET (Banco de dados de imagens FERET em cores) e PAL (banco de dados de face PAL do Productive Aging Laboratory,) ambos da Universidade do Texas em Dallas. O tamanho total da amostra de

aprendizado é 245057, das quais 50859 são as amostras de pele e 194198 são amostras que não são de pele.

- Absenteeism at work (Absenteeism): O conjunto de instâncias foi criado com registros de faltas no trabalho entre julho de 2007 a julho de 2010 em uma empresa de correios no Brasil, e apresenta os seguintes atributos:

1. Identificação individual (ID)
2. Motivo da ausência (CID). Ausências atestadas pelo Código Internacional de Doenças (CID) estratificadas em 21 categorias (I a XXI) e 7 categorias sem acompanhamento de pacientes (CID) (22), consulta médica (23), doação de sangue (24), exame laboratorial (25), ausência injustificada (26), fisioterapia (27), consulta odontológica (28).
3. Mês de ausência;
4. Dia da semana (segunda-feira (2), terça-feira (3), quarta-feira (4), quinta-feira (5), sexta-feira (6));
5. Estações do ano (verão (1), outono (2), inverno (3), primavera (4));
6. Despesas de transporte;
7. Distância entre residência e trabalho (quilômetros);
8. Tempo de serviço;
9. Idade;
10. Carga de trabalho Média/dia;
11. Objetivo atingido;
12. Falha disciplinar (sim = 1; não = 0);
13. Educação (ensino médio (1), graduação (2), pós-graduação (3), mestrado e doutorado (4));
14. Filho (número de filhos);
15. Bebedor social (sim = 1; não = 0);
16. Fumante social (sim = 1; não = 0);
17. Animal de estimação (número de animais);
18. Peso;
19. Altura;
20. Índice de massa corporal;
21. Tempo de ausência em horas (destino).

- Turkiye Student Evaluation (Student): Este conjunto de instâncias contém um total de 5820 notas de avaliação fornecidas por estudantes da Universidade Gazi em Ancara (Turquia). Há um total de 28 perguntas específicas do curso e 5 atributos adicionais. Os atributos são dados por:

1. instr: identificador do instrutor {1,2,3}
2. código do curso {1-13}
3. Reprovado: Número de vezes que o aluno está fazendo este curso {0,1,2,3, ...}
4. Presença: Código do nível de presença {0, 1, 2, 3, 4}
5. Nível de dificuldade do curso percebido pelo aluno {1,2,3,4,5}

Q1. O conteúdo do curso, o método de ensino e o sistema de avaliação do semestre foram fornecidos no início;

Q2. As metas e objetivos do curso foram claramente estabelecidos no início do período;

Q3. O curso valia a quantidade de crédito atribuída a ele;

Q4. O curso foi ministrado de acordo com o programa anunciado no primeiro dia de aula;

Q5. As discussões em classe, tarefas de casa, aplicações e estudos foram satisfatórios;

Q6. Os recursos do livro didático e de outros recursos foram suficientes e estavam atualizados;

Q7. O curso permitiu trabalhos de campo, aplicações, laboratório, discussão e outros estudos;

Q8. Os questionários, tarefas, projetos e exames contribuíram para ajudar o aprendizado;

Q9. Gostei muito da aula e estava ansioso para participar ativamente durante as palestras;

Q10. Minhas expectativas iniciais sobre o curso foram atendidas no final do período ou do ano;

Q11. O curso foi relevante e benéfico para o meu desenvolvimento profissional;

Q12. O curso me ajudou a olhar a vida e o mundo com uma nova perspectiva;

Q13. O conhecimento do instrutor era relevante e atualizado;

Q14. O instrutor veio preparado para as aulas;

Q15. O instrutor ensinou de acordo com o plano de aula anunciado;

Q16. O instrutor estava comprometido com o curso e era compreensível;

Q17. O instrutor chegou no horário para as aulas;

Q18. O instrutor tem uma fala suave e fácil de acompanhar;

Q19. O instrutor fez uso eficaz do horário de aula;

Q20. O instrutor explicou a matéria e estava disposto a ajudar os alunos;

Q21. O instrutor demonstrou uma abordagem positiva para os alunos;

Q22. O instrutor foi aberto e respeitou as opiniões dos alunos sobre o conteúdo;

Q23. O instrutor incentivou a participação no curso;

Q24. O instrutor atribuiu tarefas/projetos de casa relevantes e ajudou/guiou os alunos;

Q25. O instrutor respondeu a perguntas sobre a matéria dentro e fora da aula;

Q26. O sistema de avaliação do instrutor (provas parciais e finais, projetos, tarefas etc.) mediu efetivamente os objetivos do curso;

Q27. O instrutor forneceu soluções para os exames e discutiu-os com os alunos;

Q28. O instrutor tratou todos os alunos de maneira correta e objetiva;

Q1-Q28 são todos do tipo Likert, o que significa que os valores são obtidos na faixa de {1,2,3,4,5}.

- Pen-Based Recognition of Handwritten Digits (Pendigitis): Conjunto de instâncias de dígitos coletados de 250 dígitos manuscritos de 44 escritores. As instâncias manuscritas por 30 escritores são usadas para treinamento, validação cruzada e os dígitos manuscritos pelos outros 14 são usados para testes. Este

conjunto de instâncias também está disponível no formato UNIPEN. Foi utilizado um tablet sensível à pressão WACOM PL-100V com tela LCD integrada e caneta sem fio. As áreas de entrada e exibição estão localizadas no mesmo local. Conectado à porta serial de um PC baseado no Intel 486, ele permite coletar amostras de dígitos manuscrito. Foi solicitado que esses escritores escrevessem 250 dígitos em ordem aleatória dentro de caixas com uma resolução de 500 por 500 pixels do tablet. Cada tela contém cinco caixas com os dígitos a serem gravados exibidos acima. Os escritores são instruídos a escrever apenas dentro dessas caixas. Os dez primeiros dígitos são ignorados porque a maioria dos escritores não estão familiarizada com esse tipo de dispositivo de entrada.

No estudo, foi usado apenas informações de coordenadas (x,y). Os valores do nível de pressão da caneta são ignorados. Primeiro, foi aplicada a normalização para evitar distorções de escala. Os dados brutos que foram capturados no tablet consistem em valores inteiros entre 0 e 500 (resolução da caixa de entrada do tablet). As novas coordenadas variam no intervalo entre 0 e 100. O conjunto contém um total de 10992 instâncias com 16 atributos numéricos de $\{0,...,100\}$ e uma classe com valores $\{0,1,2,3,4,5,6,7,8,9\}$.

- Epileptic Seizure Recognition (Epileptic): O conjunto de instâncias original de referência consiste em 5 pastas diferentes, cada uma com 100 arquivos, cada arquivo representando um único sujeito/pessoa. Cada arquivo é uma gravação da atividade cerebral por 23,6 segundos. A série temporal correspondente é mostrada em 4097 atributos. Cada atributo é o valor da gravação do EEG em um momento diferente. Portanto, temos um total de 500 indivíduos, cada um com 4097 atributos por 23,6 segundos. Dividiu-se e misturou cada um dos 4097 atributos em 23 partes, cada parte contém 178 atributos por 1 segundo e atributo é o valor da gravação do EEG em um momento diferente. Portanto, agora existe $23 \times 500 = 11500$ instâncias, cada informação contém 178 atributos por 1 segundo (coluna), a última coluna representa a classe $y \in \{1,2,3,4,5\}$.
- MNIST of handwritten digits (MNIST): O conjunto de instâncias MNIST de dígitos manuscritos, possui um conjunto de treinamento de 60.000 instâncias e

um conjunto de testes de 10.000 instâncias. É um subconjunto de um conjunto maior disponível no NIST. Os dígitos foram normalizados em tamanho e centralizados em uma imagem de tamanho fixo. As imagens originais em preto e branco (em dois níveis) do NIST foram normalizadas em tamanho para caber em uma caixa de 20x20 pixels, preservando sua proporção. As imagens resultantes contêm níveis de cinza como resultado da técnica de suavização de serrilhado usada pelo algoritmo de normalização. As imagens foram centralizadas em uma imagem de 28x28, calculando o centro de massa dos pixels e traduzindo a imagem para posicionar esse ponto no centro do campo de 28x28. O conjunto de instância MNIST foi construído a partir do Banco de Dados Especial 3 (SD-3) e do Banco de Dados Especial 1 (SD-1) do NIST, que contém imagens binárias de dígitos manuscritos. O NIST designou originalmente SD-3 como seu conjunto de treinamento e SD-1 como seu conjunto de testes. No entanto, o SD-3 é muito mais limpo e fácil de reconhecer que o SD-1. A razão para isso pode ser encontrada no fato de que o SD-3 foi coletado entre funcionários do *Census Bureau*, enquanto o SD-1 foi coletado entre estudantes do ensino médio. Portanto, foi necessário construir um novo banco de dados misturando os conjuntos de dados do NIST. O conjunto de treinamento MNIST é composto por 30.000 padrões do SD-3 e 30.000 padrões do SD-1. O conjunto de testes foi composta por 5.000 instâncias do SD-3 e 5.000 instâncias do SD-1. O conjunto de treinamento de 60.000 instâncias contém manuscritos de aproximadamente 250 escritores. Garantindo que os conjuntos manuscritos do conjunto de treinamento e do conjunto de testes não fossem comuns. Para o experimento foram usadas as 60.000 instâncias de treinamento [LeCun 1998].

- Birch: Um conjunto de instâncias sintéticas com 100.000 instâncias com 2 atributos e estruturado em grade regular com $K=100$, apresentado em [Zhang *et al.* 1997].

Como informado na breve descrição de cada um dos conjuntos de instâncias, os conjuntos escolhidos não apresentam instâncias com valores ausentes de atributos e foram escolhidos de forma a terem valores distintos de número de instâncias e número de atributos, com vistas a ampliar o entendimento de cada umas das estratégias de

aceleração do k-Means em dados reais. Os conjuntos de instâncias com dados reais foram tratados como conjunto de instâncias sem classe associada.

8.3.2 Metodologia Adotada na Condução dos Experimentos

A metodologia empregada para a realização dos experimentos relativos à determinação dos tempos de execução e do número de cálculos realizados pelas cinco estratégias de aceleração, a saber, compare-Means, sort-Means, k-Means-Elkan, k-Means-Yinyang e k-Means-Fission-Fusion, apresentadas e discutidas no Capítulo 7 e implementadas no sistema computacional *5ACCK-Means*, está diagramada resumidamente no fluxograma mostrado na Figura 8.10.

Cada um dos conjuntos de instâncias de dados reais utilizados foi submetido a um processo, identificado na Figura 8.6 como Pré-processamento, antes de ser utilizado pelo sistema *5ACCK-Means*. O pré-processamento envolveu a exclusão do atributo classe em conjuntos de dados supervisionados bem como cuidou da conversão de atributos qualitativos em quantitativos e, também, da exclusão de identificadores de instâncias, quando presentes. Os conjuntos pré-processados têm suas características mostradas na Tabela 8.3.

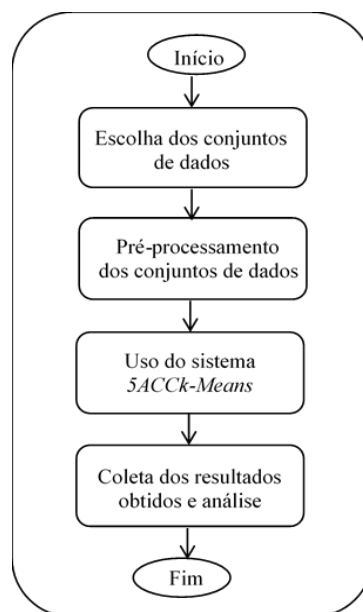


Figura 8.6 Fluxograma alto-nível da metodologia empregada nos experimentos.

Tabela 8.3 Conjuntos de instâncias de dados reais, extraídos do UCI Repository e submetidos ao processo de Pré-processamento, em que #Id: identificação do conjunto, N: número de instâncias em #Id e M: número de atributos que descrevem as instâncias em #Id.

#Id	N	M
Poker	25010	10
Optdigits	5567	64
Frogs	7195	22
Gene	801	20531
3D	434874	3
Skin	434874	3
Absenteeism	740	20
Student	5820	33
Pendigits	10992	16
Epileptic	11500	178
MNIST	60.000	784
Birch	100.000	2

Como comentado na Seção 8.1 deste capítulo, todas as implementações que compõem o sistema computacional *5ACk-Means*, foram desenvolvidas em Python 3.7 e foram utilizadas as funções *np.linalg.norm* para calcular a distância euclidiana entre instância e centroide e *np.mean* para atualizar a posição dos centroides, ambas da biblioteca *NumPy* [Van Der Walt *et al.* 2011]. O PyCharm foi utilizado como ambiente integrado de desenvolvimento (IDE). Durante a realização dos experimentos o sistema *5ACk-Means* foi executado em um mesmo notebook com processador Intel(R) Core(TM) i7-7700HQ CPU @2,80GHz com 32 Gb de memória RAM. Os conjuntos de instâncias e os associados conjuntos de centroides iniciais (escolhidos aleatoriamente) que foram fornecidos como input a cada uma das cinco estratégias de aceleração, foram os mesmos. Com isso, via de regra os agrupamentos induzidos também foram os mesmos, exceto em algumas situações comentadas no texto subsequente.

A estratégia compare-Means e sort-Means, em seu artigo original [Phillips 2002] utiliza a inicialização aleatória dos centroides iniciais; a estratégia de aceleração k-Means-Elkan, em seu artigo original [Elkan 2003] utiliza a inicialização de centroides conhecida como “furthest first” [Hochbaum & Shmoys 1985]; a estratégia k-Means-Yinyang, em seu artigo original [Ding *et al.* 2015], utiliza a inicialização aleatória dos centroides iniciais; a estratégia k-Means-Fission-Fusion, em seu artigo original [Yu & Dai 2017], utiliza a inicialização aleatória dos centroides iniciais. Com a intenção de padronizar a escolha dos centroides iniciais foi utilizada a inicialização aleatória, por ser a mais comumente utilizada.

A condição de parada associada a cada execução de cada um dos algoritmos implementados acontece quando nenhum centróide muda de posição, durante uma iteração. O resultado final relativo a cada conjunto de dados e a cada estratégia utilizada é dado pela média dos valores coletados em 10 execuções da estratégia em questão, em que cada execução tem como input o mesmo conjunto de dados e um determinado conjunto inicial de centróides, dentre 10 criados randomicamente.

Com vistas a tornar os experimentos mais amplos, viabilizando a possibilidade de um estudo da influência do número de grupos do agrupamento nas medidas sendo coletadas, o número (K) de grupos do agrupamento a ser induzido foi abordado como uma variável com valores no conjunto $\{10, 25, 50, 100, 500, 1000\}$ *i.e.*, $K \in \{10, 25, 50, 100, 500, 1000\}$. Como mostrado no pseudocódigo do procedimento que implementa a metodologia discutida, o valor de K varia de maneira crescente. É importante comentar que foi necessário levar em conta o custo computacional quando da escolha dos possíveis valores de K , uma vez que esse valor influencia o custo computacional na execução dos experimentos. Dessa forma, foram induzidos agrupamentos com número de $K \in \{500, 1000\}$ em conjuntos de instâncias reais em que K não ultrapassasse 15% do número de instâncias do conjunto e/ou 50.000 instâncias.

O tempo de execução final das estratégias é influenciado por vários quesitos, tais como: (1) método de escolha dos centróides iniciais, (2) método de cálculo de distância entre as instâncias e os centróides, (3) método de atualização da posição dos centróides, (4) linguagem de programação e funções otimizadas da linguagem (tais como: min, max, mean e etc), (5) hardware e sistema operacional utilizados, etc. Para o experimento realizado buscou-se utilizar os mesmos recursos no k-Means padrão e nas estratégias de aceleração.

As implementações do k-Means padrão e das estratégias compare-Means, sort-Means, k-Means-Elkan e k-Means-Yinyang foram desenvolvidas para este trabalho de pesquisa, baseadas nos artigos originais de cada estratégia. A implementação do k-Means-Fission-Fusion foi adaptada do código-fonte disponibilizada por um dos autores em [Yu 2018], por isso essa estratégia apresenta algumas diferenças de modelagem do algoritmo.

Como resultado dos experimentos tempos o *speed-up* e a porcentagem de cálculo realizados pelo k-Means Acelerado em relação ao k-Means padrão, dado por %Calc. Para esse experimento, speed-up é um número que mede o desempenho relativo de dois algoritmos que processam o mesmo problema, isto é, a melhoria na velocidade de execução de uma estratégia que acelera o k-Means e o k-Means padrão. O speed-up é apresentado pela Eq. 8.1 e o % Calc é apresentado na Eq. 8.2.

$$\text{speed_up} = \frac{\text{Tempo do k_Means padrão}}{\text{Tempo do k_Means acelerado}} \quad (8.1)$$

$$\% \text{Calc} = \frac{\text{Número de Cálculos de distância do k_Means Acelerado}}{\text{Número de Cálculos de distância do k_Means Padrão}} \times 100 \quad (8.2)$$

A Figura 8.11 apresenta o procedimento seguido durante os experimentos.

```

para cada conjunto de instâncias  $C_i \in \{C_1, C_2, \dots, C_H\}$ 
  begin
    para  $K \in \{10, 25, 50, 100\}$  repetir 10 vezes
      begin
        escolher aleatoriamente K centroides iniciais  $C_e = \{ce_1, ce_2, \dots, ce_k\}$ 
        k-Means-padrão( $C_i, K, C_e, AG_1$ )
        compare-Means( $C_i, K, C_e, AG_2$ )
        sort-Means( $C_i, K, C_e, AG_3$ )
        k-Means-Elkan( $C_i, K, C_e, AG_4$ )
        k-Means-Yinyang( $C_i, K, C_e, AG_5$ )
        k-Means-Fission-Fusion( $C_i, K, C_e, AG_6$ )
        checa_igualdade( $AG_1, AG_2, AG_3, AG_4, AG_5, AG_6$ )
      end
      calcular a média do tempo de execução e a média do número de
      cálculos de distância para cada um dos algoritmos usando  $C_i$ 
    end
  end

```

Figura 8.7 Descrição do procedimento adotado na condução dos experimentos realizados.

8.3.3 Experimentos e Análise dos Resultados Obtidos

Nesta seção são apresentados os resultados dos experimentos realizados utilizando as estratégias de aceleração apresentadas no Capítulo 7 dessa Dissertação. Na realização dos experimentos foi utilizada a metodologia apresentada na Seção 8.2. As estratégias executadas foram compare-Means, sort-Means, k-Means-Elkan, k-Means-Yinyang e k-Means-Fission-Fusion. Os experimentos contemplam também a execução do k-Means padrão (identificado como padrão nas três tabelas apresentadas nesta seção), com vista à comparação entre resultados obtidos por estratégias que aceleram o k-Means e resultados do k-Means padrão, que não emprega artifícios de aceleração.

Com o objetivo de organizar a apresentação dos resultados, os arquivos de dados utilizados nos experimentos foram caracterizados, em função do número de atributos (M) que descrevem as respectivas instâncias de dados, em: *pequeno* (quando $M \leq 8$), *médio* (quando $8 < M \leq 64$) e *grande* (quando $M > 64$). Com relação ao número de grupos do agrupamento induzido (k), a seguinte caracterização foi adotada: *pequeno* (quando $K \leq 10$), *médio* (quando $10 < K \leq 100$) e *grande* (quando $K > 100$).

Nas análises a seguir, a comparação da performance será feita em relação ao k-Means padrão. No que segue, a Tabela 8.4 apresenta a média dos resultados das execuções nos conjuntos de instâncias reais e a Tabela 8.5 e Tabela 8.6 apresentam a média dos resultados de execução nos conjuntos de instâncias sintéticas, respectivamente. Os campos destacados em cor cinza mostram as estratégias com melhor *speed-up* médio no experimento.

Cada uma das próximas subseções tem por foco os resultados obtidos por uma estratégia específica. Em cada uma delas os resultados obtidos são discutidos e analisados e algumas conclusões são apresentadas.

					Padrão		Compare		Sort		Elkan		Yinyang		Fission-Fusion	
#Id	N	M	K	W	Tempo	Calc. Dist	% Calc	Speed	% Calc	Speed	% Calc	Speed	% Calc	Speed	% Calc	Speed
Absenteeism	740	20	10	11	0,54	85.840	32,59	2,16	34,38	2,08	13,46	2,35	43,00	1,46	36,40	1,54
			25	14	1,54	247.234	19,00	3,08	21,86	2,91	8,50	2,91	37,60	1,75	20,85	2,70
			50	11	2,34	379.923	14,14	3,90	19,24	2,60	8,38	2,93	28,75	2,25	17,29	3,30
			100	10	4,33	703.992	12,81	4,25	19,71	1,90	9,95	2,89	24,19	2,45	14,46	3,55
			10	13	17,88	108.936	91,69	1,07	91,70	1,07	21,71	2,45	49,35	1,51	61,94	0,01
Gene	801	20.531	25	13	38,55	262.327	84,54	1,15	84,94	1,17	17,64	4,00	47,67	1,87	43,75	0,04
			50	11	63,22	452.565	77,62	1,29	78,55	1,35	17,23	4,69	31,57	2,61	35,04	0,02
			100	7	84,08	632.790	70,20	1,36	72,32	1,51	20,86	4,62	29,89	3,00	-	-
			10	35	12,90	1.981.852	90,20	0,95	87,46	0,91	12,23	2,47	27,34	2,10	82,03	0,13
			25	37	29,28	5.163.392	72,94	1,11	72,57	1,07	8,02	2,88	26,32	2,39	55,23	0,41
Optdigits	5.567	64	50	32	56,46	9.129.880	55,56	1,41	61,01	1,26	6,33	2,96	18,49	3,30	36,30	0,29
			100	32	104,90	18.037.080	40,30	1,74	41,21	1,74	4,95	2,87	13,31	3,92	22,09	0,42
			500	15	252,30	43.144.250	18,53	3,15	22,65	1,97	7,59	2,63	13,44	2,03	13,16	0,67
			10	24	9,10	1.449.180	45,71	1,58	44,72	1,68	9,62	2,52	29,79	1,93	52,00	0,71
			25	43	38,76	6.110.418	34,61	2,02	31,95	2,36	4,75	3,13	24,31	2,59	32,19	1,17
Student	5.820	33	50	33	57,30	8.991.841	28,55	2,06	30,22	2,41	4,35	3,28	19,67	3,12	23,73	1,60
			100	28	95,44	15.565.584	23,49	2,69	26,06	2,67	3,87	3,34	16,92	3,31	16,79	2,27
			500	15	290,46	44.232.000	17,61	3,51	22,22	2,08	6,37	3,40	15,84	2,06	11,50	3,31
			10	34	15,43	2.467.885	42,49	1,73	43,02	1,82	7,41	2,78	29,69	1,99	51,23	0,99
			25	46	53,20	8.346.200	25,47	2,55	26,09	2,84	3,73	3,22	25,54	2,51	26,86	1,85
Frogs	7.195	22	50	45	90,85	15.320.313	18,11	3,17	19,19	3,80	2,93	3,25	16,67	3,66	16,66	2,94
			100	45	171,83	30.743.731	14,10	3,73	15,52	4,35	2,47	3,18	12,55	4,37	11,13	4,21
			500	20	404,35	74.828.000	10,73	4,47	14,02	2,85	4,81	2,83	11,72	2,45	7,88	4,83
			1000	16	648,95	114.458.060	11,89	4,23	16,79	1,62	8,06	2,56	11,97	1,47	8,82	3,89
			10	36	25,67	4.023.072	42,86	1,69	43,58	1,74	6,90	2,78	22,43	2,33	45,81	1,31
Pendigits	10.992	17	25	44	72,98	11.724.067	22,34	2,77	23,24	3,22	3,51	3,29	16,69	3,48	20,51	2,78
			50	60	200,69	32.114.886	12,68	4,05	13,83	5,24	1,90	3,43	9,26	5,83	10,25	5,32
			100	57	365,91	59.490.528	8,11	5,08	9,39	6,97	1,58	3,42	7,41	6,85	6,70	8,06
			500	26	834,74	143.445.600	4,95	6,30	8,05	4,59	2,91	3,18	6,75	4,15	5,08	8,68
			1000	24	1.466,00	255.069.360	6,05	5,99	9,67	2,71	4,87	2,95	6,23	2,78	5,14	7,38
Epileptic	11.500	178	10	45	40,43	5.232.500	50,70	1,50	50,72	1,55	10,88	2,43	55,66	1,20	58,05	0,16
			25	109	239,30	30.739.500	39,18	1,90	39,97	1,96	4,75	3,38	40,18	1,81	34,20	0,28
			50	89	361,52	48.498.950	37,19	2,02	37,34	2,10	4,76	3,49	44,29	1,70	28,92	0,33
			100	101	764,09	111.339.895	34,56	2,13	36,90	3,69	3,62	3,69	37,82	1,92	22,48	0,43
			500	59	2.156,86	341.550.000	36,46	1,99	36,81	1,86	5,01	3,45	29,34	1,73	12,92	0,68
Poker	25.010	10	1000	50	2.977,13	507.610.000	40,05	1,87	40,69	1,41	7,20	3,08	28,87	1,19	12,35	0,67
			10	120	201,28	30.212.080	86,81	1,00	82,62	0,91	5,19	3,23	11,54	3,81	77,13	0,70
			25	80	289,18	50.582.725	51,87	1,47	55,36	1,40	3,92	3,22	14,10	3,99	44,94	1,91
			50	100	738,47	106.667.650	40,90	2,07	56,18	1,37	2,58	3,48	8,86	6,49	32,46	1,68
			100	98	1.464,65	246.848.700	22,81	2,65	28,18	2,57	1,67	3,15	6,43	7,43	16,80	2,78
MNIST	60.000	784	1000	31	4.621,03	790.941.250	8,15	4,95	11,44	3,82	2,82	3,03	6,15	2,48	6,09	4,94
			10	85	462,39	51.420.000	99,40	0,89	99,35	0,87	8,45	2,95	18,97	2,69	-	-
			25	126	1.525,64	182.592.000	96,02	0,91	96,02	0,88	4,90	3,92	14,97	3,89	-	-
			50	165	4.109,81	497.250.000	92,41	0,93	92,43	0,89	6,08	4,67	9,57	5,74	-	-
			100	119	5.586,40	715.500.000	89,08	0,92	89,14	0,88	3,02	4,41	8,96	5,81	-	-
Birch	100.000	2	10	128	822,68	128.200.000	16,96	3,37	17,15	4,30	1,48	3,93	6,16	4,89	25,04	6,00
			25	135	1.849,47	319.070.000	6,92	5,47	7,31	9,84	0,83	4,54	6,34	6,90	9,36	16,12
			50	201	6.744,36	1.174.375.000	3,05	7,18	3,43	20,78	0,35	4,53	2,48	14,58	4,08	34,83
			100	134	7.396,74	1.342.500.000	1,70	8,03	2,20	30,89	0,24	4,68	2,01	18,03	2,24	60,69
			10	35	507,23	87.485.349	17,33	3,22	18,04	3,69	3,28	3,42	20,18	2,44	20,62	3,97
Skin	245.057	3	25	43	1.681,00	263.436.275	8,09	5,05	9,17	8,13	1,67	4,22	21,20	2,98	10,50	11,29
			50	71	5.093,45	880.979.915	4,29	6,63	5,51	13,63	0,68	4,26	11,19	4,97	5,79	14,94
			100	111	15.765,10	2.738.511.975	2,37	7,76	3,35	20,24	0,32	4,18	7,61	6,82	3,03	28,34
			10	23	589,51	101.325.642	12,74	3,74	13,57	5,12	3,18	3,56	27,15	2,01	12,60	8,36
			25	36	2.347,27	397.909.710	5,05	5,92	6,27	11,50	1,17	4,40	16,21	3,64	5,46	22,99
3D	434.874	3	50	22	2.521,75	455.878.414	3,16	6,68	5,56	12,93	1,11	4,32	10,59	5,16	5,76	25,24
			100	22	5.527,26	982.815.240	1,48	8,29	4,15	17,90	0,54	4,97	6,83	7,73	4,82	37,58

Tabela 8.4 Resultado médio de 10 execuções de cada uma das estratégias de aceleração do k-Means nos conjuntos de instâncias reais, em que K é o número de grupos induzidos, W é o número médio de iterações, Calc. Dist. é a média do número de cálculos de distância realizados pelo k-Means padrão, Tempo é o tempo médio das execuções do k-Means padrão, % Calc é a porcentagem do número de cálculos realizada pelo k-Means acelerado em relação ao k-Means padrão e Speed (speed-up) é o tempo médio de execução do k-Means padrão dividido pelo tempo médio de execução do k-Means acelerado.

#Id	N	M	K	W	Padrão		Compare		Sort		Elkan		Yinyang		Fission-Fusion	
					Tempo	Calc. Dist	% Calc	Speed	% Calc	Speed	% Calc	Speed	% Calc	Speed	% Calc	Speed
S1	1.000	2	10	28	1,78	281.000	21,23	2,78	22,10	3,18	6,07	3,02	25,95	2,20	23,37	6,85
			25	26	3,95	628.100	10,70	4,34	12,52	4,59	4,12	3,43	18,31	3,26	11,81	12,34
			50	22	6,54	1.047.810	7,63	5,27	10,35	4,25	4,44	3,59	14,72	4,09	8,48	13,91
			100	18	10,89	1.744.781	7,62	5,50	11,49	2,93	6,17	3,41	12,54	4,41	7,59	10,37
S2	1.000	8	10	12	0,82	129.000	21,94	2,83	24,21	2,93	8,01	3,04	22,92	2,34	16,03	6,83
			25	18	2,75	432.900	15,32	3,57	18,26	3,48	6,12	2,99	24,22	2,64	12,62	8,59
			50	14	4,21	658.290	15,19	3,76	20,27	2,75	6,72	2,86	18,62	3,40	14,51	6,48
			100	11	6,85	1.100.000	15,40	3,74	21,89	2,05	7,89	2,80	14,28	3,98	13,60	6,06
S3	1.000	32	10	5	0,32	51.000	91,92	0,86	93,79	0,84	38,73	1,45	60,84	1,14	64,79	0,63
			25	4	0,61	97.600	53,10	1,45	60,58	1,20	28,88	1,79	65,37	1,13	47,87	0,88
			50	5	1,59	244.760	22,55	3,00	32,83	1,99	15,45	2,65	40,14	1,69	27,86	1,47
			100	5	3,33	514.476	14,47	4,01	27,91	1,83	12,29	2,66	29,22	2,13	22,83	1,71
S4	1.000	64	10	5	0,33	52.000	98,93	0,85	99,80	0,79	49,03	1,18	72,49	0,94	83,60	0,27
			25	4	0,68	110.200	86,72	0,93	90,33	0,82	40,56	1,42	69,53	1,03	60,53	0,36
			50	4	1,40	221.020	56,58	1,40	63,75	1,10	26,70	1,94	55,64	1,25	41,45	0,55
			100	3	2,34	362.102	28,07	2,52	42,59	1,38	21,65	2,32	49,78	1,24	36,35	0,60
S5	1.000	128	10	7	0,49	75.000	100,02	0,83	100,26	0,78	42,65	1,26	67,22	1,02	79,83	0,15
			25	6	0,92	140.000	97,67	0,85	99,10	0,77	41,91	1,39	65,90	1,11	65,49	0,19
			50	4	1,49	224.000	87,43	0,94	91,23	0,81	40,84	1,43	64,55	1,07	58,58	0,20
			100	3	3,04	442.400	60,53	1,32	67,99	0,95	29,61	1,85	55,43	1,09	42,15	0,28
S6	1.000	256	10	7	0,52	74.000	100,33	0,83	100,40	0,78	49,21	1,18	75,70	0,95	83,73	0,08
			25	6	1,05	147.400	100,13	0,84	100,80	0,76	46,77	1,27	69,50	1,08	71,40	0,09
			50	5	1,87	254.740	98,02	0,85	99,78	0,75	44,92	1,38	65,40	1,07	64,16	0,10
			100	4	3,37	465.474	88,79	0,94	92,75	0,75	40,14	1,53	60,45	1,02	54,35	0,12
S7	1.000	512	10	8	0,66	81.000	100,43	0,86	100,43	0,83	51,10	1,22	76,52	1,69	86,08	0,04
			25	6	1,10	138.100	100,74	0,85	101,08	0,80	55,94	1,18	78,26	0,98	80,09	0,05
			50	5	2,02	258.810	101,03	0,85	101,91	0,76	51,35	1,28	73,29	0,97	70,55	0,05
			100	4	3,56	455.881	99,76	0,87	101,83	0,70	49,80	1,33	69,48	0,89	65,82	0,05
S8	1.000	1.024	10	6	0,81	68.000	100,41	0,89	100,44	0,86	53,90	1,29	86,82	0,90	86,36	0,03
			25	4	1,20	104.300	100,85	0,87	101,12	0,84	55,38	1,28	73,57	1,09	73,75	0,04
			50	2	1,52	135.430	101,38	0,89	102,26	0,82	79,13	1,00	81,38	0,99	83,69	0,03
			100	2	2,45	213.543	102,00	0,91	104,33	0,78	96,76	0,85	95,91	0,77	99,12	0,03
S9	5.000	2	10	39	14,18	1.975.000	20,77	3,00	21,46	3,48	4,22	3,66	17,32	3,04	23,69	7,35
			25	66	50,87	7.985.000	8,93	4,84	9,66	7,15	1,77	4,01	10,36	5,04	9,38	14,58
			50	63	96,61	15.073.500	4,97	6,39	5,92	10,18	1,43	4,13	7,61	6,94	5,12	24,46
			100	51	154,30	24.207.350	3,30	7,30	4,68	9,90	1,58	4,17	6,48	7,94	3,71	28,84
S10	5.000	8	10	23	7,16	1.160.000	19,82	3,00	20,95	3,53	5,31	3,30	15,93	3,06	12,70	8,04
			25	41	30,61	5.016.000	13,51	3,88	14,79	4,75	3,13	3,26	12,98	4,16	9,59	10,85
			50	42	55,89	10.001.600	13,08	4,00	14,71	4,72	2,67	3,15	9,52	5,83	9,33	11,11
			100	34	102,79	17.350.000	12,77	4,02	14,75	4,24	2,57	3,13	7,47	6,99	7,94	11,32
S11	5.000	32	10	4	1,52	235.000	92,19	0,89	94,25	0,84	41,67	1,36	64,54	1,07	70,07	0,58
			25	11	8,74	1.398.500	46,75	1,59	49,39	1,54	10,26	2,97	24,20	2,70	17,29	2,34
			50	16	23,10	4.089.850	14,55	3,37	17,55	3,62	4,83	3,30	14,27	3,77	9,22	3,83
			100	16	51,60	8.150.000	6,04	6,06	9,96	5,84	3,87	3,40	11,97	4,67	7,91	4,85
S12	5.000	64	10	6	2,12	305.000	98,55	0,89	99,25	0,83	40,42	1,44	62,67	1,15	78,08	0,30
			25	5	4,15	630.500	86,41	0,96	89,46	0,89	34,25	1,61	58,13	1,25	54,27	0,42
			50	5	8,69	1.338.050	54,75	1,45	60,57	1,27	21,56	2,23	47,97	1,41	35,15	0,64
			100	8	27,40	4.183.805	17,97	3,43	24,23	2,82	8,50	3,29	23,16	2,60	16,13	1,38
S13	5.000	128	10	6	2,31	315.000	99,64	0,84	99,91	0,81	47,10	1,24	67,99	1,02	88,05	0,14
			25	6	5,17	731.500	96,66	0,86	97,99	0,79	38,81	1,44	63,91	1,11	65,82	0,19
			50	5	8,22	1.223.150	86,08	0,95	89,62	0,87	35,98	1,61	59,09	1,16	53,56	0,23
			100	4	14,82	2.222.315	58,03	1,37	65,50	1,15	25,83	2,04	54,12	1,12	41,08	0,30
S14	5.000	256	10	9	3,63	495.000	99,99	0,85	100,02	0,82	39,53	1,41	63,60	1,09	82,11	0,08
			25	7	6,25	912.000	99,15	0,85	99,65	0,81	38,44	1,47	59,10	1,22	66,75	0,10
			50	5	8,69	1.250.000	95,91	0,88	97,63	0,81	41,39	1,47	61,23	1,14	62,56	0,11
			100	5	16,35	2.375.000	85,25	0,98	88,99	0,88	35,84	1,64	58,39	1,06	52,36	0,13
S15	5.000	512	10	13	5,73	650.000	100,11	0,86	100,08	0,83	37,59	1,52	63,66	1,10	-	-
			25	8	7,89	965.000	99,90	0,85	100,12	0,82	41,92	1,43	65,90	1,14	-	-
			50	5	11,03	1.346.500	99,25	0,86	100,06	0,80	49,97	1,28	67,01	1,08	-	-
			100	5	19,77	2.434.650	96,38	0,88	98,25	0,81	46,15	1,38	65,56	0,98	-	-
S16	5.000	1.024	10	12	7,82	605.000	100,14	0,88	100,08	0,87	44,34	1,44	72,37	1,04	-	-
			25	8	11,70	960.500	100,11	0,88	100,20	0,86	46,44	1,48	70,70	1,13	-	-
			50	6	17,47	1.471.050	100,03	0,88	100,39	0,85	52,14	1,37	69,90	1,11	-	-
			100	5	28,58	2.447.105	99,54	0,89	100,49	0,84	55,16	1,33	73,99	0,97	-	-

Tabela 8.5 Resultado médio de 10 execuções de cada uma das estratégias de aceleração do k-Means nos conjuntos de instâncias sintéticas, em que K é o número de grupos induzidos, W é o número médio de iterações, Calc. Dist. é a média do número de cálculos de distância realizados pelo k-Means padrão, Tempo é o tempo médio das execuções do k-Means padrão, % Calc é a porcentagem do número de cálculos realizada pelo k-Means acelerado em relação ao k-Means padrão e Speed (speed-up) é o tempo médio de execução do k-Means padrão dividido pelo tempo médio de execução do k-Means acelerado.

#Id	N	M	K	W	Padrão		Compare		Sort		Elkan		Yinyang		Fission-Fusion	
					Tempo	Calc. Dist.	% Calc	Speed	% Calc	Speed	% Calc	Speed	% Calc	Speed	% Calc	Speed
S17	10.000	2	10	64	45,79	6.490.000	20,01	3,06	20,39	3,69	2,85	3,93	12,52	3,63	22,25	6,81
			25	87	138,77	20.874.000	8,79	5,02	9,30	7,68	1,50	4,11	9,51	5,36	9,85	13,95
			50	86	260,57	40.737.400	4,65	6,53	5,33	12,15	1,03	4,17	6,52	7,49	5,25	23,00
			100	83	463,64	79.073.740	2,68	7,54	3,53	14,87	0,90	4,33	4,31	10,95	2,79	40,25
S18	10.000	8	10	46	29,77	4.600.000	21,13	2,85	21,69	3,40	3,21	3,45	10,65	3,69	12,64	7,30
			25	56	88,17	14.075.000	13,05	3,96	13,93	5,23	2,31	3,43	11,26	4,85	8,85	11,54
			50	62	186,00	29.657.500	12,79	4,03	13,90	5,20	1,93	3,22	7,71	6,77	8,73	11,52
			100	58	360,26	58.400.000	12,36	4,11	13,56	5,04	1,63	3,22	5,10	9,45	7,26	12,83
S19	10.000	32	10	6	3,08	620.000	91,29	0,74	92,80	0,70	31,12	1,32	50,57	1,08	57,79	0,56
			25	19	26,89	4.662.000	46,76	1,61	48,27	1,60	6,44	3,40	15,40	3,96	11,31	3,52
			50	27	71,67	13.066.200	10,71	4,53	12,61	5,71	3,15	3,97	10,37	5,53	6,19	6,33
			100	29	156,46	27.906.620	5,19	6,33	7,60	8,38	2,48	3,64	9,07	6,08	5,41	7,08
S20	10.000	64	10	6	3,92	640.000	98,58	0,84	99,26	0,79	38,18	1,40	59,80	1,14	79,70	0,27
			25	6	8,80	1.514.000	83,92	0,97	86,50	0,92	27,60	1,84	48,39	1,50	45,35	0,49
			50	9	25,65	4.701.400	49,10	1,54	52,51	1,46	12,14	2,85	27,45	2,37	19,71	1,10
			100	15	80,69	14.870.140	14,28	3,84	17,77	3,97	4,92	3,72	14,31	4,02	9,41	2,28
S21	10.000	128	10	10	6,55	1.000.000	99,66	0,85	99,82	0,82	32,17	1,61	52,16	1,32	77,04	0,16
			25	5	7,72	1.300.000	97,36	0,86	98,81	0,81	44,06	1,36	65,55	1,13	72,18	0,17
			50	5	14,47	2.480.000	86,01	0,97	89,48	0,89	34,67	1,65	57,60	1,20	52,81	0,23
			100	5	29,86	5.148.000	55,94	1,41	62,28	1,25	22,14	2,24	47,38	1,30	35,28	0,35
S22	10.000	256	10	9	7,15	970.000	99,98	0,83	100,01	0,82	39,53	1,41	64,17	1,08	83,57	0,08
			25	7	12,61	1.747.000	99,14	0,85	99,69	0,82	39,25	1,48	60,01	1,21	70,32	0,09
			50	5	17,68	2.574.700	96,43	0,88	98,15	0,81	43,32	1,41	64,61	1,10	65,50	0,10
			100	5	34,39	4.757.470	85,78	0,96	89,53	0,88	35,96	1,62	58,88	1,02	53,04	0,12
S23	10.000	512	10	15	13,53	1.530.000	100,07	0,83	100,03	0,78	34,46	1,46	61,02	1,08	-	-
			25	9	17,94	2.128.000	99,87	0,84	100,06	0,80	39,00	1,50	61,35	1,19	-	-
			50	6	24,94	3.062.800	98,90	0,87	99,61	0,81	43,72	1,40	61,27	1,13	-	-
			100	5	42,62	5.106.280	95,68	0,89	97,45	0,83	43,60	1,44	64,21	0,96	-	-
S24	10.000	1.024	10	15	18,87	1.520.000	100,10	0,87	100,04	0,86	39,22	1,56	66,71	1,11	-	-
			25	10	27,82	2.377.000	99,98	0,88	100,04	0,85	39,19	1,64	62,43	1,25	-	-
			50	6	35,27	3.087.700	99,79	0,88	100,12	0,84	49,82	1,41	67,07	1,13	-	-
			100	5	62,36	5.308.770	99,01	0,89	99,88	0,83	50,35	1,39	67,35	1,04	-	-

Tabela 8.6 Resultado médio de 10 execuções de cada uma das estratégias de aceleração do k-Means nos conjuntos de instâncias sintéticas, em que K é o número de grupos induzidos, W é o número médio de iterações, Calc. Dist. é a média do número de cálculos de distância realizados pelo k-Means padrão, Tempo é o tempo médio das execuções do k-Means padrão, % Calc é a porcentagem do número de cálculos realizada pelo k-Means acelerado em relação ao k-Means padrão e *Speed* (speed-up) é o tempo médio de execução do k-Means padrão dividido pelo tempo médio de execução do k-Means acelerado.

8.3.3.1 Compare-Means

A estratégia de aceleração compare-Means apresenta suas melhores performances em conjunto de instâncias que são descritas por um número pequeno de atributos, como pode ser confirmado pelos resultados obtidos utilizando os conjuntos S1, S9, S17, Birch, Skin e 3D. Pode ser observado, também em relação aos resultados obtidos pelo compare-Means nesses mesmos conjuntos, que tal estratégia promove a diminuição dos cálculos de distância em torno de 75%.

À medida que o número de atributos que descrevem as instâncias cresce, a performance da estratégia compare-Means começa a degradar e o número de cálculos de distâncias evitados começa a diminuir, até o ponto em que a estratégia se torna desvantajosa,

inclusive quando tem seus resultados comparados com os resultados do k-Means padrão. Os conjuntos S3, S4, S11, S12, S19, S20 e Optical mostram o ponto onde o compare-Means começa a apresentar performance inferior àquela exibida pelo k-Means padrão e seu uso não é mais recomendado. Os conjuntos S5, S6, S7, S8, S13, S14, S15, S16, S21, S22, S23, S24 e MNIST exibem situações em que o compare-Means tem performance inferior àquela obtida pelo k-Means padrão.

Outra observação pertinente aos resultados obtidos é a de que, para um mesmo conjunto de dados, à medida que o número de grupos (K) do agrupamento a ser induzido aumenta, a performance do compare-Means também aumenta, quando comparada àquela do k-Means padrão.

O número de cálculos de distância entre instância e centroide evitados pode chegar até 98% em conjuntos de instâncias com número pequeno de atributos, como pode ser observado nos resultados associados aos conjuntos de dados S1, S9, S17, Birch, Skin e 3D. Porém, à medida que o número de atributos cresce a porcentagem do número de cálculos evitados diminui até se tornar insignificante ou, então, até levemente superior que o k-Means padrão, como mostram os resultados relativos aos conjuntos de instâncias sintéticas S8, S16, S24.

Com base nos resultados obtidos nos experimentos, não é possível evidenciar o número exato de atributos a partir do qual o compare-Means começa a apresentar performance inferior àquela do k-Means padrão; isso se deve, particularmente, à influência que outras variáveis (número de instâncias e número de grupos) exercem no processo de indução do agrupamento. É importante comentar, entretanto, que nos experimentos realizados utilizando conjuntos de instâncias sintéticas descritas com 32 ou mais atributos, foi possível identificar casos em que a estratégia compare-Means se mostrou desvantajosa, como pode ser confirmado nos resultados obtidos com os conjuntos de dados S3, S11, S19, Optdigits e MNIST.

Nos conjuntos de instâncias reais Optical (K=10) e MNIST o compare-Means teve performance inferior àquela obtida pelo k-Means padrão. Já no conjunto Gene, cujas instâncias são descritas por um número alto de atributos *i.e.*, 20.531, o compare-Means teve melhor performance que a exibida pelo k-Means padrão. Tais resultados

evidenciam que outras variáveis influenciam nos resultados de performance obtidas pelas estratégias envolvidas. Além das mencionadas anteriormente, também influenciam a performance das estratégias a dispersão das instâncias no espaço amostral definido pelos atributos que as descrevem, o número de iterações necessárias à convergência do k-Means, etc.

Como pode ser constatado por meio do exame dos valores apresentados nas três tabelas anteriores, o compare-Means foi a única estratégia que apresentou uma relação direta entre o número de cálculos de distâncias evitados com o *speed-up* de execução que pode se expresso por: quanto maior o número de cálculos de distância evitados, maior é o valor do *speed-up*.

Analisando os resultados apresentados, os experimentos sugerem que o compare-Means:

- Apresenta performance decrescente à medida que o número de atributos que descrevem as instâncias aumenta, podendo apresentar performance inferior àquela apresentada pelo k-Means padrão;
- Apresenta melhora na performance à medida que o número de grupos do agrupamento sendo induzido cresce;
- Apresenta melhora na performance à medida que o número de instâncias do conjunto aumenta.

8.3.3.2 Sort-Means

A estratégia de aceleração sort-Means, assim como a compare-Means, foi proposta por Phillips, em [Phillips 2002]. Tal estratégia tem como principal característica a ordenação da consulta aos centroides, usando como referência a distância que centroides estão do centroide do grupo ao qual a instância pertence.

Devido à ordenação da consulta aos centroides a cada iteração, pode acontecer que o agrupamento final induzido pelo sort-Means seja diferente do agrupamento final induzido pelo k-Means padrão. A diferença entre os agrupamentos induzidos acontece quando, durante o processo de indução, uma instância está equidistante de dois centroides. O k-Means padrão atribui a instância ao grupo associado ao primeiro centroide que lhe é apresentado. Apesar do mesmo processo ocorrer no sort-Means,

como a ordem de consulta aos centroides é ordenada pela distância, a instância pode ser atribuída ao grupo representado por outro centroide, igualmente válido.

Apesar de tal ocorrência não ser um erro, uma vez que é consequência do próprio funcionamento de cada uma das estratégias, ela influencia o número de iterações e o número de cálculos de distância evitados.

Empiricamente, foi verificado que essa ocorrência é mais comum em conjuntos de instâncias com atributos do tipo inteiro e número pequeno de atributos. Os conjuntos de instâncias sintéticas não apresentaram nenhum agrupamento final induzido que fosse diferente daquele induzido pelo k-Means padrão. É fato, entretanto, que resultados obtidos em mais da metade dos conjuntos de instâncias reais apresentaram mudanças no agrupamento final induzido em pelo menos 1 das 10 execuções realizadas. Optou-se por não apresentar esses valores em detalhes devido ao entendimento que o agrupamento final gerado pelo sort-Means é um agrupamento igualmente válido.

Com base nos resultados obtidos pode ser afirmado que a estratégia sort-Means apresenta boa performance em conjunto de instâncias com número pequeno de atributos e número pequeno ou médio de grupos induzidos. A partir de um determinado valor associado ao número de grupos, nos experimentos com $K = \{500, 1000\}$, a estratégia sort-Means começa a ter sua performance degradada, devido ao custo computacional associado ao processo de ordenação dos centroides, como pode ser observado nos resultados associados aos conjuntos Poker, Optical e Frogs, e confirmado pelo próprio autor em [Phillips 2002].

Analisando os resultados apresentados, os experimentos sugerem que o sort-Means:

- Apresenta performance decrescente à medida que o número de atributos aumenta, podendo apresentar performance inferior àquela exibida pelo k-Means padrão;
- Apresenta melhora na performance à medida que o número de grupos do agrupamento sendo induzido cresce; porém, em conjuntos com número grande de grupos induzidos, a partir de um determinado valor associado ao número de grupos, a performance da estratégia começa a degradar;

- Uma relação direta entre número de cálculos de distâncias evitados e speed-up só é observada em conjuntos com número pequeno e médio de grupos no agrupamento a ser induzido. Em conjuntos com número grande de grupos associado ao agrupamento a ser induzido, o custo computacional do processo de ordenação da consulta aos centroides interfere nessa relação;
- Apresenta performance crescente à medida que o número de instâncias do conjunto cresce;

8.3.3.3 k-Means-Elkan

A estratégia utilizada no algoritmo k-Means-Elkan é a que mostrou performance mais estável nos experimentos. Nos conjuntos com instâncias reais o k-Means-Elkan foi, em média, de 2 a 5 vezes mais rápido que o k-Means padrão e evitou maior número de cálculos, com uma média superior a 80% de cálculos evitados.

Os resultados obtidos nos experimentos realizados mostraram que a estratégia k-Means-Elkan tem sua performance afetada à medida que o número de atributos cresce, porém, de forma mais moderada do que acontece com as demais estratégias. Observou-se também que em agrupamentos com grande número de grupos induzidos, a partir de um determinado número de grupos, a estratégia k-Means-Elkan começa a degradar sua performance, como pode ser visto nos resultados relativos aos conjuntos Optdigits, Frogs, Pendigits e Poker. Isso ocorre devido ao custo de verificação da necessidade de realizar os cálculos de distância; ainda assim a estratégia se mostrou eficiente nos experimentos realizados.

Apenas em um caso a estratégia k-Means-Elkan se mostrou menos eficiente que aquela do k-Means padrão, como pode ser visto nos resultados associados ao conjunto instâncias sintéticas S8 com número de grupos igual a 100. Tal conjunto possui 1.000 instâncias e 1.024 atributos e o processo de indução do agrupamento convergiu com 2 iterações e teve um *speed-up* com a estratégia k-Means-Elkan de 0,85. Quando do uso de conjuntos de instâncias reais não é comum uma convergência com número tão pequeno de iterações.

Analisando os resultados apresentados, os experimentos sugerem que o k-Means-Elkan:

- Apresenta performance decrescente, porém com queda moderada, à medida que o número de atributos aumenta;
- Apresenta melhora na performance à medida que o número de grupos participantes do agrupamento induzido cresce; porém, a partir de um determinado número de grupos a performance começa a degradar de maneira moderada;
- Apresenta melhora na performance crescente à medida que o número de instâncias do conjunto cresce;
- Apresenta performance superior que o k-Means padrão nos casos experimentados.

8.3.3.4 k-Means-Yinyang

Os resultados dos experimentos realizados evidenciam que a estratégia utilizada no k-Means-Yinyang apresentou uma boa performance em geral, isto é, apresentou performance superior que a alcançada pelo k-Means padrão em todos os conjuntos de instâncias reais. Nos conjuntos de instâncias sintéticas com número grande de atributos e que convergiram em poucas iterações, como, por exemplo os conjuntos S7 e S8, a estratégia k-Means-Yinyang apresentou *speed-up* inferior ao k-Means padrão.

A estratégia k-Means-Yinyang apresentou seus melhores resultados em conjuntos com número médio de atributos e número médio de grupos induzidos, como pode ser visto nos resultados relativos aos conjuntos Optdigits, Frogs, Pendigits e Poker. Contudo, a estratégia k-Means-Yinyang não necessariamente foi responsável pelo melhor *speed-up* entre as estratégias consideradas. Nos experimentos realizados a performance de k-Means-Yinyang começa a degradar em agrupamentos que foram induzidos com número grande de grupos.

Analisando os resultados apresentados, os experimentos sugerem que o k-Means-Yinyang:

- Apresenta a melhor performance em conjunto de instâncias com número médio de atributos;
- Apresenta performance crescente à medida que o número de instâncias do conjunto aumenta;

- Apresenta melhora na performance à medida que o número de grupos participantes do agrupamento induzido cresce, porém, a partir de um determinado número de grupos a performance começa a degradar.

8.3.3.5 k-Means-Fission-Fusion

A estratégia utilizada pelo k-Means-Fission-Fusion apresenta excelente performance em conjuntos com número pequeno de atributos, como pode ser comprovado nos resultados relativos aos conjuntos de dados S1, S9, S17, Birch, Skin, 3D e degrada a performance de forma agressiva à medida que o número de atributos que descrevem o conjunto de instância cresce. A estratégia k-Means-Fission-Fusion apresenta uma performance ruim em conjuntos de instâncias com número grande de atributos.

A performance em conjunto de instâncias com número grande de atributos é ruim a ponto do *speed-up* cair a valores abaixo de 0,10, isto é, o k-Means padrão processa 10 vezes mais rápido que a estratégia de aceleração k-Means-Fission-Fusion, como pode ser observado nos resultados associados aos conjuntos de instâncias S4, S5, S6, S7, S8, S12, S13, S14, S21, S22, Gene e Epileptic. Devido a essa baixa performance em conjuntos com número grande de atributos, essa estratégia não foi executada nos conjuntos de instâncias sintéticas S15, S16, S23 e S24 e nos conjuntos de instâncias reais Gene (K=100) e MNIST, tal exclusão ocorreu depois que teste preliminares mostraram *speed-up* inferior a 0,10.

Em conjuntos de instâncias com número médio de atributos a estratégia k-Means-Fission-Fusion se mostrou vantajosa somente à medida que o número de grupos no agrupamento induzido cresceu, como pode ser visto nos resultados relativos aos conjuntos S3, S11, S12, S19, S20, Frogs, Poker e Student.

Analisando os resultados apresentados, os experimentos sugerem que o k-Means-Fission-Fusion:

- Apresenta excelente performance em conjuntos com número pequeno de atributos;
- Apresenta performance crescente à medida que o número de grupos associados ao agrupamento induzido cresce;

- Apresenta performance decrescente à medida que o número de atributos que descrevem as instâncias do conjunto de instâncias aumenta;
- Apresenta performance crescente à medida que o número de instâncias do conjunto aumenta.

8.4 Considerações Finais sobre os Resultados dos Experimentos Realizados

Este trabalho de pesquisa em nível de mestrado investigou as estratégias de aceleração que utilizam a desigualdade triangular como forma de evitar cálculos de distância entre as instâncias e os centroides, de forma a acelerar o algoritmo k-Means. Tais estratégias são nomeadas nessa Dissertação como: compare-Means, sort-Means, k-Means-Elkan, k-Means-Yinyang e k-Means-Fission-FusionGC.

As cinco estratégias de aceleração consideradas no trabalho realizado produzem resultados de tempo de execução e número de cálculos distintos entre si e não foi possível fazer uma correlação direta entre número de cálculos de distâncias evitados e tempo de execução. A relação não é possível porque cada estratégia tem custos distintos associados aos processos que verificam se o cálculo de distância entre instância e centroide será realizado.

Após análise dos resultados dos experimentos envolvendo as estratégias, foi possível concluir que o k-Means-Fission-Fusion apresenta as melhores performances em conjuntos com número pequeno de atributos.

Os experimentos também mostraram que, em geral, a estratégia k-Means-Elkan é a que evitou o maior número de cálculos; entretanto, tal estratégia não necessariamente apresentou o melhor resultado de tempo de execução. A estratégia k-Means-Elkan foi a que apresentou performance mais estável durante os experimentos e que apresentou mais frequentemente, maior *speed-up*.

A estratégia do k-Means-Yinyang apresentou em conjuntos de instâncias reais com número médio de atributos e número médio de grupos induzidos boa performance, porém, não necessariamente, em todos os casos, apresentou o melhor *speed-up*.

O compare-Mean e sort-Means apresentaram alguns casos de melhor *speed-up*, porém não apresentaram valores muito superiores àqueles obtidos pelo k-Means-Elkan ou pelo k-Means-Fission-Fusion.

Com isso, no uso aplicado das estratégias, as estratégias do k-Means-Elkan e do k-Means-Fission-Fusion se mostraram mais vantajosas de serem implementadas em um sistema pois, em geral, foram as duas estratégias que mostraram, além de performance, clareza das situações em que são vantajosas e estabilidade nos resultados.

As estratégias de aceleração investigadas não modificam, em geral, os resultados induzidos pelo k-Means padrão, dessa forma, entendeu-se que os índices de validação de agrupamentos não trariam informações relevantes para a interpretação dos experimentos.

Capítulo 9

Conclusões

Este capítulo apresenta inicialmente um breve resumo do trabalho desenvolvido desde o início do projeto de pesquisa que resultou nesta Dissertação, além de revisitar os objetivos do projeto, apresentados no Capítulo 1. A Seção 9.1 descreve os principais pontos levantados e investigados na pesquisa realizada, bem como as principais contribuições desta Dissertação e conclusões derivadas dos experimentos conduzidos. A Seção 9.2 apresenta possíveis atividades para dar continuidade ao trabalho realizado e descrito nesta Dissertação.

9.1 Principais Pontos Investigados e Conclusões dos Experimentos

O trabalho realizado contemplou uma fase inicial de levantamento e familiarização com a terminologia empregada na área de AM, bem como com os principais conceitos e técnicas envolvidos. De particular interesse foram os estudos sobre métodos de AM caracterizados como não-supervisionados, com ênfase em algoritmos de agrupamento, técnicas de pré-processamento, para tratamento inicial dos dados e estudo sobre técnicas para a validação de agrupamentos induzidos.

A proposta básica que norteou o trabalho de pesquisa descrito nesta Dissertação foi identificar, junto à literatura, algoritmos propostos com o objetivo de acelerar a execução do algoritmo de agrupamento k-Means e, particularmente, investigar a contribuição daqueles baseados em técnicas que empregam a chamada desigualdade triangular.

Dentre as propostas encontradas de estratégias de aceleração do k-Means envolvendo o uso de desigualdade triangular, cinco foram escolhidas para investigação, sendo elas: compare-Means, sort-Means, k-Means-Elkan, k-Means-Yinyang e k-Means-Fission-Fusion. Os resultados dos experimentos conduzidos, descritos nesta Dissertação, são evidência empírica que as cinco estratégias de aceleração do algoritmo k-Means

consideradas tendem a apresentar uma performance melhor que aquela apresentada pelo algoritmo k-Means e, em geral, sem qualquer alteração do agrupamento final induzido.

As estratégias compartilham o uso da desigualdade triangular para evitar o cálculo das distâncias entre as instâncias e os centroides. Cada uma das estratégias utiliza, de forma distinta das demais, a desigualdade triangular, como foi abordado no Capítulo 7. Contudo, evitar a realização de cálculos de distâncias não necessariamente leva a um ganho de performance, pois existe um custo computacional para prospectar se o cálculo de distância entre instância e centroide será realizado ou não. Dessa forma, cada estratégia mostrou vantagens e desvantagens, com relação ao uso da desigualdade triangular, na dependência das características do conjunto de instâncias, como é mostrado na Seção 8.3.

Para os experimentos realizados foram escolhidos conjuntos de instâncias com características distintas, de forma que fosse possível verificar a performance em conjuntos que variavam em número de instâncias e em número de atributos que descrevem as instâncias. Também foi objetivo analisar a performance de cada estratégia, na indução de agrupamentos com número de grupos diferentes.

Com base nos resultados obtidos, entretanto, não é possível afirmar que uma estratégia é melhor que uma outra em todos os conjuntos experimentados. Contudo, os resultados indicam que as estratégias do k-Means-Elkan e k-Means-Fission-Fusion apresentaram *speed-up* superior com maior frequência. Os resultados obtidos por essas duas estratégias conseguiram evidenciar, com maior clareza, as características dos conjuntos de instâncias de dados em que tal superioridade acontece.

Houve casos que as demais estratégias apresentam *speed-up* superior. Entretanto, as duas estratégias anteriormente citadas, mesmo apresentado um *speed-up* inferior, tal inferioridade não foi muito significativa. Em situações em que as estratégias compare-Means, sort-Means e k-Mean-Yinyang apresentaram *speed-up* superior, ou o k-Means-Elkan ou o k-Means-Fission-Fusion também apresentaram, pelo menos, uma performance aproximada às performances caracterizadas como superiores.

As estratégias de aceleração foram implementadas, com base em suas respectivas descrições encontradas nos artigos já referenciados, em linguagem Python 3.7 (<https://www.python.org/>) e, também, fazendo uso do *software* livre PyCharm

(<https://www.jetbrains.com/pycharm/>), que é um IDE (Ambiente Integral de Desenvolvimento) de desenvolvimento de software que pode ser usado em uma ampla variedade de plataformas, tais como Windows, Linux e MacOS.

9.2 Possíveis Atividades para Continuidade do Trabalho

O algoritmo k-Means é um dos principais algoritmos de aprendizado de máquina e melhorar o custo computacional de sua execução é mandatório, particularmente quando conjuntos de instâncias volumosos devem ser agrupados abrindo, dessa forma, ampla gama de possibilidades de investigação. Existem diversas pesquisas que utilizam a desigualdade triangular para acelerar o k-Means e podem inspirar novos trabalhos a serem iniciados em continuidade a esta pesquisa.

Além de novas propostas de aceleração com o uso da desigualdade triangular, também é possível investigar novas propostas de escolha dos centroides iniciais e métodos mais eficientes de atualizar a posição dos centroides durante as iterações, como a apresentada por Kwedlo, em [Kwedlo 2017].

Uma possível nova estratégia de aceleração está na unificação dos conceitos utilizados pelo k-Means-Yinyang e pelo k-Means-Fission-Fusion. Como foi visto no Capítulo 7 a estratégia do k-Means-Yinyang utiliza grupos de centroides e a estratégia do k-Means-Fission-Fusion utiliza grupos de instâncias para acelerar o k-Means. Um desdobramento do trabalho realizado nesta Dissertação é o da investigação das possibilidades de unificação dessas duas estratégias, em uma nova estratégia híbrida.

Referências

- [Abu-Mostafa *et al.* 2012] Abu-Mostafa, Y. S.; Magdon-Ismael, M.; Lin, H. T. (2012) *Learning from Data*, New York, NY, USA: AMLBook.
- [Anderberg 1973] Anderberg, M.R. (1973) *Cluster Analysis for Applications*, New York: Academic Press.
- [Angluin 1980] Angluin, D. (1980) Inductive inference of formal languages from positive data, *Information and Control*, v. 42, no. 2, pp. 117–135.
- [Arlot & Celisse 2010] Arlot, S.; Celisse, A. (2010) A survey of cross-validation procedures for model selection, *Statistics Surveys Online Journal*, no. 4, pp. 40–79.
- [Arthur & Vassilvitskii 2007] Arthur, D.; Vassilvitskii, S. (2007) k-Means++: the advantages of careful seeding, *Proc. of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics*, pp. 1027–1035.
- [Bentley 1975] Bentley, J. L. (1975) Multidimensional binary search trees used for associative searching, *Communications of the ACM*, v. 18, no. 9, pp. 509–517.
- [Berkhin 2006] Berkhin, P. (2006) A survey of clustering data mining techniques, *Grouping Multidimensional Data*, Berlin, Heidelberg: Springer, pp. 25–71.
- [Bottesch *et al.* 2016] Bottesch, T.; Bühler, T.; Kächele, M. (2016) Speeding up k-means by approximating Euclidean distances via block vectors, *International Conference on Machine Learning*, pp. 2578–2586.
- [Bradley & Fayyad 1998] Bradley, P. S.; Fayyad, U. (1998) Refining initial points for k-means clustering, *Proc. of the 15th International Conference on Machine Learning*, pp. 91–99.
- [Brown 2015] Brown, R. A. (2015) Building a balanced k-d tree in $O(KN \log N)$ time, *Journal of Computer Graphics Techniques*, v. 4, no. 1, pp. 50–68.

- [Camargos 2016] Camargos, R. C. (2016) Algoritmos aglomerativos de agrupamento baseados em teoria de matrizes (Dissertação de Mestrado, Faculdade Campo Limpo Paulista).
- [Celebi *et al.* 2013] Celebi, M. E.; Kingravi, H. A.; Vela, P. A. (2013) A comparative study of efficient initialization methods for the k-Means clustering algorithm, *Expert Systems with Applications*, v. 40, pp. 200–210
- [Chapelle *et al.* 2006] Chapelle, O.; Scholkopf, B.; Zien, A. (2006) *Semi-supervised Learning*, MIT Press.
- [Dash & Liu 1997] Dash, M.; Liu, H. (1997) Feature selection for classification, *Intelligent Data Analysis*, v. 1, pp. 131–156.
- [Desgraupes 2003] Desgraupes, B. (2013) Clustering índices, University of Paris Ouest-Lab Modal'X, v. 1.
- [Dietterich 1995] Dietterich, T. (1995) Overfitting and undercomputing in machine learning, *ACM computing surveys*, v. 27, no. 3, pp. 326–327.
- [Ding & He 2004] Ding, C.; He, X. (2004) K-means clustering via principal component analysis, *Proc. of the 21st Int. Conference on Machine Learning, Banff, Canada*, pp. 255–232.
- [Ding *et al.* 2015] Ding, Y.; Zhao, Y.; Shen, X.; Musuvathi, M.; Mytkowicz, T. (2015) Yinyang k-means: A drop-in replacement of the classic k-means with consistent speedup, *Proc. of the Int. Conference on Machine Learning*, pp. 579–587.
- [Drake & Hamerly 2012] Drake, J.; Hamerly, G. (2012) Accelerated k-means with adaptive distance bounds, *5th NIPS Work. Optim. Mach. Learn.*, pp. 579–587.
- [Drineas *et al.* 2004] Drineas, P.; Frieze, A.; Kannan, R.; Vempala, S.; Vinay, V. (2004) Clustering large graphs via the singular value decomposition, *Machine Learning*, v. 56, no. 1-3, pp. 9–33.

- [Dua & Graff 2019] Dua, D; Graff, C. (2019) UCI Machine Learning Repository [<http://archive.ics.uci.edu/ml>], Irvine, CA: University of California, School of Information and Computer Science.
- [Dunn 1974] Dunn, J. (1974) Well separated clusters and optimal fuzzy partitions, *Journal of Cybernetics*, v. 4, no. 1, pp. 95–104
- [Elkan 2003] Elkan, C. (2003) Using the triangle inequality to accelerate k-means, *Proceedings of the 20th International Conference on Machine Learning ICML-03*, pp. 147–153.
- [Elkan 2012] Elkan, C. (2012) Evaluating classifiers. San Diego: University of California.
- [Fayyad & Irani 1993] Fayyad, U.; Irani, K. (1993) Multi-interval discretization of continuous-valued attributes for classification learning, *Proc. 13th Int. Joint Conf. on Artificial Intelligence*, pp. 1022–1027.
- [Forgy 1965] Forgy, E. W. (1965) Cluster analysis of multivariate data: efficiency versus interpretability of classifications, *Biometrics*, v. 21, pp. 768–769.
- [Gowda & Diday 1992] Gowda, K. C.; Diday, E. (1992) Symbolic clustering using a new similarity measure, *IEEE Transactions on Systems, Man, and Cybernetics*, V. 22, no. 2, pp. 368–378.
- [Hamerly 2010] Hamerly, G. (2010) Making k-Means even faster, *Proc. of the SIAM International Conference on Data Mining*, pp. 130–140.
- [Hamerly & Drake 2015] Hamerly, G.; Drake, J. (2015) Accelerating Lloyd’s algorithm for k-Means clustering, *Partitional Clustering Algorithms*, Springer, pp. 41–78.
- [Han et al. 2012] Han, J.; Kamber, M.; Pei, J. (2012) *Data Mining Concepts and Techniques*, 3rd. Ed., Amsterdam: Morgan Kaufmann Publishers.
- [Hartigan & Wong 1979] Hartigan, J. A.; Wong, M. A. (1979) Algorithm AS 136: A k-Means clustering algorithm, *Journal of the Royal Statistical Society, Series C*, v. 28, no. 1, pp. 100–108.

- [Hochbaum & Shmoys 1985] Hochbaum, D. S.; Shmoys, D. B. (1985) A best possible heuristics for the k-center problem, *Mathematics of Operations Res.*, v. 10, no. 2, pp. 180–184.
- [Hodge & Austin 2004] Hodge, V.; Austin, J. (2004) A survey of outlier detection methodologies, *Artificial Intelligence Review*, v. 22, no. 2, pp. 85–126.
- [Jaccard 1901] Jaccard, P. (1901) Étude comparative de la distribution florale dans une portion des Alpes et des Jura, *Bull Soc Vaudoise Sci Nat*, v. 37, pp. 547–579.
- [Jain & Dubes 1988] Jain, A.K; Dubes, R.C. (1988) *Algorithms for Clustering Data*, Prentice Hall.
- [Jain et al. 1999] Jain, A. K.; Murty, M. N.; Flynn, P. J. (1999) Data clustering: a review, *ACM Computing Surveys (CSUR)*, v. 31, no. 3, pp. 264–323.
- [Jain 2010] Jain, A. K. (2010) Data clustering: 50 years beyond K-means, *Pattern Recognition Letters*, v. 31, no. 8, pp. 651–666.
- Jancey, R. C. (1966) Multidimensional group analysis, *Australian Journal of Botany*, v. 14, no. 1, pp. 127–130.
- [Jolliffe 2002] Jolliffe I. T. (2002) *Principal Component Analysis*, Series: Springer Series in Statistics, 2nd ed., Springer, NY, pp. 487.
- [Judd et al. 1998] Judd, D.; McKinley, P. K.; & Jain, A. K. (1998) Large-scale parallel data clustering, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 20, no. 8, pp. 871–876.
- [Kameshwaran & Malarvizhi 2014] Kameshwaran, K.; Malarvizhi, K. (2014) Survey on clustering techniques in data mining, *International Journal of Computer Science and Information Technologies*, v.5, no. 2, pp. 2272–2276.
- [Kanungo et al. 2002] Kanungo, T.; Mount, D. M.; Netanyahu, N. S.; Piatko, C. D.; Silveman, R.; Wu, A. Y. (2002) An efficient k-Means clustering algorithm: analysis and implementation, *IEEE Transactions on Pattern Analysis and Machine Intel*, v. 24, no. 7, pp. 881–892.

- [Khan & Ahmad 2004] Khan, S. S.; Ahmad, A. (2004) Cluster center initialization algorithm for k-Means clustering, *Pattern Recognition Letters*, v. 25, pp. 1293–1302.
- [Kodinariya & Makwana 2013] Kodinariya, T. M.; Makwana, P. R. (2013) Review on determining number of cluster in k-Means clustering, *International Journal*, v. 1, no. 6, pp. 90–95.
- [Kohavi 1995] Kohavi, R. (1995) A study of cross-validation and bootstrap for accuracy estimation and model selection, *International Joint Conference on Artificial Intelligence*, v. 14, no. 2, pp. 1137–1145.
- [Kotsiantis & Kanellopoulos 2006] Kotsiantis, S.; Kanellopoulos, D. (2006) Discretization techniques: a recent survey, *GESTS International Transactions on Computer Science and Engineering*, v. 32, no. 1, pp. 47–58.
- [Kwedlo 2017] Kwedlo, W. (2017) Two modifications of Yinyang K-means algorithm, *International Conference on Artificial Intelligence and Soft Computing*, Springer, Cham, pp. 94–103.
- [Liu *et al.* 2013] Liu, Y.; Li, Z.; Xiong, H.; Gao, X.; Wu, J. (2010) Understanding of internal clustering validation measures, *2010 IEEE International Conference on Data Mining*, pp. 911–916.
- [Lloyd 1957] Lloyd, S. (1957) Least square quantization in PCM, *Bell Telephone Laboratories Paper*.
- [Lloyd 1982] Lloyd, S. (1982) Least squares quantization in PCM, *IEEE Transactions on Information Theory*, v. 28, no. 2, pp. 129–137.
- [MacQueen 1967] MacQueen, J. (1967) Some methods for classification and analysis of multivariate observations, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, v. 1, no. 14, pp. 281–297.
- [Matte & Nicoletti 2019] Matte, M. K.; Nicoletti, M. C. (2019) Revisão de estratégias para a aceleração do algoritmo k-Means, *Anais do Workshop em Computação da FACCAMP (WCF 2019)*, pp. 1–6.

- [Michalski *et al.*1983] Michalski, R. S.; Carbonell, J. G.; Mitchell, T. M. (1983) *Machine Learning: An Artificial Intelligence Approach*, Springer-Verlag Berlin Heidelberg.
- [Mitchell 1997] Mitchell, T. M. (1997) *Machine Learning*. McGraw-Hill Science/Engineering/Math.
- [Moore 2000] Moore, A. W. (2000) The anchors hierarchy: using the triangle inequality to survive high dimensional data, *Proc. of The Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI'00)*, pp. 397–405.
- [Newling & Fleuret 2016] Newling, J.; Fleuret, F. (2016) Fast k-means with accurate bounds, *International Conference on Machine Learning*, pp. 936–944.
- [Nicoletti, 1994] Nicoletti, M. C. (1994) Ampliando os limites do aprendizado indutivo de máquina através das abordagens construtiva e relacional (Tese de Doutorado, Universidade de São Paulo).
- [Nietto 2016] Nietto, P. R. (2016) Algoritmos de agrupamento divisivos com pré-determinação automática do número de grupos (Dissertação de Mestrado, Faculdade Campo Limpo Paulista).
- [Oliveira 2018] Oliveira, A. F. (2018) Favorecendo o desempenho do k-Means via métodos de inicialização de centroides de grupos (Dissertação de Mestrado, Centro Universitário Campo Limpo Paulista).
- [Oliveira & Nicoletti 2018] Oliveira, A. F.; Nicoletti, M. C. (2018) Favoring the k-Means algo-rithm with initialization methods, *Proc. of The International Conference on Intelli-gent Systems Design and Applications (ISDA 2018)*, pp. 21–31.
- [Pedregosa *et al.* 2011] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; ... ; Vanderplas, J. (2011) Scikit-learn: Machine learning in Python, *Journal of machine learning research*, n. 12, v. Oct, pp. 2825-2830.

- [Pelleg & Moore 1999] Pelleg, D.; Moore, A. (1999) Accelerating exact k-Means algorithms with geometric reasoning, *Proc. of The Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 277–281.
- [Pelleg & Moore 2001] Pelleg, D.; Moore, A. (2001) Mixtures of rectangles: Interpretable soft clustering. *ICML*, pp. 401–408.
- [Phillips 2002] Phillips, S. J. (2002) Acceleration of k-Means and related clustering algorithms, *Proceedings of the Workshop on Algorithm Engineering and Experimentation*, Springer, pp. 166–177.
- [Prati 2006] Prati, R. C. (2006) Novas abordagens em aprendizado de máquina para a geração de regras, classes desbalanceadas e ordenação de casos (Tese de Doutorado, Universidade de São Paulo).
- [Pyle 1999] Pyle, D. (1999) *Data Preparation for Data Mining*, Morgan Kaufmann.
- [Rand 1971] Rand, W. M. (1971) Objective criteria for the evaluation of clustering methods, *Journal of the American Statistical Association*, v. 66, no. 336, pp. 846–850.
- [Real 2014] Real, E. M. (2013) Investigação de algoritmos sequenciais de agrupamento com pré-processamento de dados em aprendizado de máquina (Dissertação de Mestrado, Faculdade Campo Limpo Paulista)
- [Reddy & Jana 2012] Reddy, D.; Jana, P. K. (2012) Initialization for k-Means clustering using Voronoi diagram, *Procedia Technology*, v. 4, pp. 395–400.
- [Rousseeuw 1987] Rousseeuw, P. (1987) Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, *J. Computational Applied Mathematics*, v. 20, no. 1, pp. 53–65.
- [Steinhaus 1957] Steinhaus, H. (1957) The problem of estimation, *The Annals of Mathematical Statistics*, v. 28, no. 3, pp. 633–648.
- [Theodoridis & Koutroubas 2009] Theodoridis, S.; Koutroubas, K. (1999) *Pattern Recognition*, New York: Academic Press.

- [Vendramin *et al.* 2010] Vendramin, L.; Campello, R. J.; Hruschka, E. R. (2010) Relative clustering validity criteria: a comparative overview, *Statistical Analysis and Data Mining: the ASA Data Science Journal*, v. 3, no. 4, pp. 209–235.
- [Van Der Walt *et al.* 2011] Van Der Walt, S.; Colbert, S. C.; Varoquaux, G. (2011). The NumPy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2), 22.
- [Xiong & Li 2013] Xiong, H.; Li, Z. (2013) Clustering Validation Measures, *Data Clustering: Algorithms and Applications*, Chapman and Hall/CRC, pp. 571–606.
- [Yu 2018] Yu, Q. (2018) Fission-Fusion K-means [código-fonte], disponível em: https://github.com/yuqiao76558/Fission_Fusion-k-means/blob/master/Fission-Fusion%20K-means.ipynb, acessado em: 08/09/2019.
- [Yu & Dai 2017] Yu, Q.; Dai, B. R. (2017) Accelerating K-Means by Grouping Points Automatically, *International Conference on Big Data Analytics and Knowledge Discovery*, Springer, Cham, pp. 199-213.
- [Zhu & Goldberg 2009] Zhu, X.; Goldberg, A. B. (2009) *Introduction to Semi-Supervised Learning*, Morgan & Claypool.