Diego Luchi

Novas técnicas de amostragem tendenciosa para os algoritmos de análise de agrupamento k-médias e DBSCAN

Vitória

Fevereiro de 2019

Diego Luchi

Novas técnicas de amostragem tendenciosa para os algoritmos de análise de agrupamento k-médias e DBSCAN

Tese apresentada ao Programa de Pós-Graduação do Departamento de Informática da Universidade Federal do Espírito Santo, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação.

Universidade Federal do Espírito Santo

Departamento de Informática

Programa de Pós-Graduação

Orientador: Prof. Dr. Flávio Miguel Varejão

Vitória

Fevereiro de 2019

Diego Luchi

Novas técnicas de amostragem tendenciosa para os algoritmos de análise de agrupamento k-médias e DBSCAN

Tese apresentada ao Programa de Pós-Graduação do Departamento de Informática da Universidade Federal do Espírito Santo, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação.

Trabalho aprovado. Vitória, 28 de Março de 2019:

Prof. Dr. Flávio Miguel Varejão
Orientador

Prof. Dr. Thomas W. Rauber
Membro Interno

Prof. Dr. Thiago Oliveira dos Santos
Membro Interno

Prof. Dr. Alexandre Loureiros Rodrigues
Membro Externo

Prof. Dr. Alexandre Plastino de Carvalho
Membro Externo

Vitória
Fevereiro de 2019

Agradecimentos

Agradeço a minha família. Em especial a minha mãe, por ter sempre me apoiado e incentivado para que esse objetivo fosse atingido.

A meu orientador pela disponibilidade e paciência.

A FAPES pelo apoio financeiro que foi essencial à dedicação exclusiva ao Programa de Pós-graduação em Informática.

Resumo

A análise de agrupamento é um conjunto de técnicas destinadas a identificação de grupos de elementos similares em um conjunto de dados. Tais técnicas são utilizadas nas mais variadas aplicações, como segmentação de imagens, processamento de sinais, compressão de dados, aprendizado não supervisionado, seleção de características, amostragem, dentre outras. Embora sejam importantes nas mais diversas aplicações, a utilização dessas técnicas em conjunto de dados de grande cardinalidade é um problema em virtude da escalabilidade ruim de vários algoritmos tradicionais. Uma das formas de se contornar esse problema é a amostragem, afinal, reduzir a cardinalidade do conjuntos de dados reduz bastante o esforço computacional exigido pelos métodos. Nesse trabalho são apresentados três métodos amostrais novos especificamente projetados para serem utilizados em conjunto com os algoritmos de análise de agrupamento k-médias e DBSCAN. Os resultados experimentais mostram que os métodos propostos para o algoritmo DBSCAN obtiveram melhores resultados que os competidores. Contudo, a abordagem amostral proposta para o k-médias ficou em segundo lugar, retornando resultados de qualidade inferior a outro método recentemente proposto denominado DENDIS.

Palavras-chaves: Amostragem, análise de agrupamento, aprendizado não supervisionado.

Abstract

The cluster analysis is a set of techniques designed to identify groups of similar elements in a dataset. Such techniques are used in many different applications, such as image segmentation, signal processing, data compression, unsupervised learning, selection of characteristics, sampling, among others. Although they are important in a wide range of applications, the use of these techniques in large cardinality data is a problem due to the poor scalability of several traditional algorithms. One way to circumvent this problem is to sample, after all, reducing the cardinality of data sets greatly reduces the computational effort required by the methods. This thesis presents three new sampling methods specifically designed to be used in conjunction with the cluster analysis algorithms k-means and DBSCAN. The experimental results show that those designed for the DBSCAN algorithm obtained better results than the competitors. However, the proposed sampling approach for k-means returned lower quality results than DENDIS, a recently proposed method.

Key-words: Sampling, cluster analysis, unsupervised learning.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

A análise de agrupamento consiste em um conjunto de técnicas destinadas à identificação de grupos compostos por elementos similares, tendo por base uma métrica para se aferir a similaridade entre os elementos. A vantagem e – ao mesmo tempo – o desafio das técnicas está em particionar o conjunto de dados de modo adequado sem nenhum conhecimento prévio a respeito do mesmo.

As técnicas de análise de agrupamento são importantes ferramentas que foram, e ainda são, utilizadas com sucesso em diversas áreas como segmentação de imagens (HARALICK; SHAPIRO, 1985), processamento de sinais (CHEN; MULGREW; GRANT, 1993), compressão (YANG et al., 2008), aprendizado não supervisionado (BERKHIN et al., 2006), seleção de características (SONG; NI; WANG, 2013), amostragem (VISWANATH; BABU, 2009), dentre outras.

1.1 Motivação

Apesar da importância das técnicas e de seu grande sucesso em diversas aplicações, a aplicação dos métodos sobre conjuntos de dados volumosos, cada vez mais comum hoje em dia, é um desafio para muitas abordagens clássicas. Esse problema ocorre devido a escalabilidade ruim de muitos métodos que não foram projetados para processar com eficiência esse volume de informação.

Tais problemas de escalabilidade normalmente são associados a uma complexidade computacional elevada dos métodos em relação a cardinalidade do conjunto de dados. No entanto, uma alta dimensionalidade também impacta nos requisitos computacionais demandados pelos métodos, seja em tempo de execução, seja em termos do espaço requerido na memória principal. Em ambos os casos, a utilização de técnicas com o objetivo de reduzir os problemas de escalabilidade dos métodos de análise de agrupamento é uma necessidade.

Diversas variantes dos métodos tradicionais foram propostas, com o objetivo de reduzir o esforço computacional. Por exemplo, CURE (GUHA; RASTOGI; SHIM, 1998) e o CLARA (BERKHIN et al., 2006), são dois algoritmos de análise de agrupamento que

incluem uma etapa amostral para diminuir a cardinalidade do conjunto de dados, conseguindo assim manter um desempenho satisfatório frente um grande volume de informação. Todavia, diversos métodos tradicionais não possuem nenhum mecanismo para contornar esse problema e ainda são muito utilizados na prática. Podemos citar o k-médias (STEINHAUS, 1956; LLOYD, 1982; MACQUEEN et al., 1967), PAM (KAUFMAN; ROUSSEEUW, 1990), DBSCAN (ESTER et al., 1996), agrupamento espectral (SHI; MALIK, 2000), o algoritmo de estimação de maximização da esperança (DEMPSTER; LAIRD; RUBIN, 1977), ISODATA (BREIMAN et al., 1965), CHAMELEON (KARYPIS; HAN; KUMAR, 1999), dentre outros. Métodos hierárquicos também sofrem do mesmo problema, o que inviabiliza a sua aplicação direta em grandes conjuntos de dados (KOLATCH et al., 2001).

Diversas técnicas foram propostas com o objetivo de amenizar o problema de escalabilidade dos algoritmos de análise de agrupamento, dentre as quais podemos destacar: indexadores espaciais, técnicas de busca aproximada, sumarização dos dados, amostragem, paralelismo, agrupamento incremental (ANDONI; INDYK, 2006; JAIN, 2010). Embora muitas dessas abordagens reduzam significativamente o esforço computacional, muitas delas são vinculadas a certos algoritmos de análise de agrupamento.

As duas primeiras abordagens citadas aceleram o algoritmo DBSCAN, mas não o k-médias, pois o último executa a varredura completa no conjunto de dados a cada iteração e não realiza buscas em nenhuma região particular do espaço. Portanto, essa estratégia não fornece nenhum benefício em termos de desempenho computacional. Além disso, indexadores espaciais, apresentam uma rápida degradação do desempenho quando o conjunto de dados possui um elevado número de dimensões (ARLIA; COPPOLA, 2001; CHÁVEZ et al., 2001).

Por outro lado, o k-médias (STOFFEL; BELKONIENE, 1999; NICKOLLS et al., 2008; ZHAO; MA; HE, 2009) é de fácil paralelização, enquanto um algoritmo hierárquico (OLSON, 1995), devido sua natureza fortemente sequencial, é praticamente impossível. Explorar as vantagens do paralelismo não é algo simples e não há uma forma geral de paralelização que se adapte a qualquer algoritmo.

Enquanto algumas técnicas só fazem sentido em conjunto com certos algoritmos – ou sequer proporcionam um ganho de desempenho significativo perante a alta dimensio-

nalidade do conjunto de dados – outras, como a amostragem e a sumarização dos dados, se mostram muito mais versáteis. Essas técnicas podem ser aplicadas conjuntamente com qualquer algoritmo de análise de agrupamento, em uma etapa anterior à sua execução como um pré-processamento, ou estando presente como parte do método.

Em ambos os casos, o montante de informação a ser processada pelo método e o esforço computacional diminuem. Alinhado com outros trabalhos da literatura (ZHOU et al., 2000; BREUNIG et al., 2001; NANOPOULOS; THEODORIDIS; MANOLOPOULOS, 2001; WANG et al., 2011), o objeto de estudo dessa tese é o uso de técnicas amostrais com o propósito de viabilizar a execução dos métodos em grandes conjuntos de dados.

Uma das dificuldades inerentes ao processo de amostragem em análise de agrupamento é a possibilidade de um grupo não ser representado na amostra. Esse risco é acentuado em conjuntos de dados onde as cardinalidades das partições sejam discrepantes. Fazendo um paralelo com aprendizado supervisionado, haja visto o conhecimento prévio dos rótulos, o processo de amostragem de modo a garantir a representatividade de cada rótulo na amostra final é muito mais simples. De fato, a abordagem escolhida para esses casos, em diversos trabalhos de classificação, é o uso da amostragem estratificada uniforme. A amostragem estratificada uniforme consiste em amostrar aleatoriamente a mesma quantidade de elementos de cada estrato e, nesse caso, cada rótulo é considerado um estrato. Esse procedimento simples garante que uma amostra pequena terá a mesma quantidade de representantes de cada rótulo, e que todos os rótulos estejam representados.

No contexto de análise de agrupamento, é impossível o uso da amostragem estratificada devido a ausência dos rótulos no conjunto de dados. Por esse motivo, inúmeros trabalhos encontrados na literatura limitam-se ao uso da amostragem aleatória uniforme como alternativa para redução da quantidade de elementos. Em Bejarano et al. (2011), os autores ajustam o tamanho da amostra para tentar garantir que todos os grupos estejam representados. Contudo, em muitos casos, o tamanho amostral deve se tornar excessivamente grande para que partições de baixa cardinalidade tenham uma alta probabilidade de estar presente na amostra.

Se considerarmos que os conjuntos de dados podem ser arbitrariamente desbalanceados – somado ao fato que a amostragem aleatória uniforme dá a mesma chance de sorteio para cada elemento – podemos concluir que não existe um limite inferior para o

tamanho da amostra, para que a mesma contenha representantes de todos os grupos com uma margem de confiança alta (KOLLIOS et al., 2001). Além disso, como a cardinalidade de cada grupo do conjuntos de dados é desconhecida, determinar o tamanho necessário da amostra é uma tarefa complexa. Em Bejarano et al. (2011), os autores propuseram uma variante do algoritmo k-médias que ajustava o tamanho da amostra a cada iteração do algoritmo, e em Luchi et al. (2016) (Capítulo 4) os autores estimam o tamanho da amostra para o k-médias através de um estudo piloto do conjunto de dados.

Os problemas citados – com a amostragem aleatória uniforme – motivaram diversos trabalhos sobre amostragem tendenciosa no contexto de análise de agrupamento. O principal objetivo dessas abordagens é aumentar a representatividade de elementos em partições de menores cardinalidades para tentar garantir que, com uma amostra pequena, todas as partições estejam representadas na amostra.

1.2 Histórico

O primeiro trabalho publicado sobre amostragem tendenciosa para análise de agrupamento foi o DBS (*density biased sampling*), proposto por Palmer e Faloutsos (2000). No DBS, a chance de sorteio de cada elemento é inversamente proporcional à densidade da região em que se encontra. Tal método de amostragem tendenciosa ficou conhecido na área e, desde então, diversos trabalhos de amostragem para análise de agrupamento foram publicados (KOLLIOS et al., 2003; NANOPOULOS; THEODORIDIS; MANOLOPOULOS, 2006; APPEL et al., 2007a), alinhados com o mesmo objetivo.

Ao analisar as propostas existentes na literatura, fica evidente que as abordagens se enquadram em três vertentes. A primeira vertente, como em Palmer e Faloutsos (2000), é composta de métodos que calculam a chance do elemento ser inserido na amostra baseado na densidade do elemento no conjunto de dados original. Estimar a densidade de modo ingênuo exige um custo computacional proibitivo. O grande desafio enfrentado pelos métodos que se enquadram nessa vertente é de estimar a densidade, ainda que de modo aproximado, a um custo computacional reduzido.

Em Franco-Lopez, Ek e Bauer (2001), Mitra, Murthy e Pal (2002) os autores utilizaram uma abordagem utilizando o algoritmo k vizinhos mais próximos para estimar a densidade de cada elemento. Apesar de obter uma estimativa precisa da densidade, o custo

computacional elevado torna, muitas vezes, proibitivo o uso desses métodos. Em Mitra, Murthy e Pal (2002) os autores apontam que é quase uma obrigatoriedade contornar a alta complexidade computacional apresentada pelo algoritmo.

Em Kollios et al. (2001), Kollios et al. (2003), os autores utilizam uma abordagem com *kernel* para se estimar a densidade de cada elemento. A técnica apresentada realiza uma estimativa de densidade de boa qualidade e estima a densidade a um custo computacional reduzido em relação a uma estimativa ingênua; contudo, o custo computacional ainda é elevado. Além disso, o método apresentado requer um ajuste de parâmetro que controla a região de influência de cada elemento na densidade.

Outra forma de estimativa de densidade a um custo computacional reduzido, utilizada em muitos trabalhos, é através do uso de indexadores espaciais. No DBS (PALMER; FALOUTSOS, 2000) os autores utilizam de índices espaciais utilizando um reticulado sobre os dados. Todavia, o reticulado não é armazenado e não há a necessidade de realizar uma varredura em todas as células do reticulado. O reticulado é usado apenas para mapear os elementos do conjunto de dados para uma tabela *hash*.

Outra abordagem que utiliza indexadores espaciais é proposta em Kerdprasop, Kerdprasop e Sattayatham (2005), os autores dividem o conjunto de dados utilizando um reticulado e percorrem todas as células calculando a densidade de cada célula; ou como em Appel et al. (2007a), Appel et al. (2007b), Appel (2010), que possui uma alta complexidade de espaço para armazenar todas as células em uma árvore na primeira etapa do algoritmo – que é simplificada em uma etapa posterior. Essas abordagens apresentam bons resultados e executam rapidamente nos cenários de baixa dimensionalidade. Contudo, para conjuntos de dados de alta dimensionalidade a quantidade de células do reticulado cresce exponencialmente, inviabilizando a utilização desses algoritmos nesses cenários.

Em Nanopoulos, Manolopoulos e Theodoridis (2002), Nanopoulos, Theodoridis e Manolopoulos (2006) os autores adotam uma nova estratégia utilizando árvores R (GUTTMAN, 1984) para diminuir a quantidade de células em relação à abordagem com reticulado. Assim como outras abordagens que utilizam indexadores espaciais, a vantagem da estrutura de dados utilizada – e consequentemente o algoritmo – perde desempenho com o aumento da dimensionalidade (CHÁVEZ et al., 2001; ARLIA; COPPOLA, 2001).

Em Qian et al. (2014) os autores utilizaram um reticulado variado para diminuir o número de células existentes - ao preço da perda de precisão na estimativa de densidade. A proposta dos autores analisa individualmente cada dimensão na construção do reticulado final e, portanto, possui uma complexidade de tempo linear em relação à dimensionalidade do conjunto de dados. Devido ao DBS (PALMER; FALOUTSOS, 2000) e o algoritmo amostral utilizando reticulado variado (QIAN et al., 2014) possuírem uma boa escalabilidade tanto em relação à cardinalidade do conjunto de dados quanto em relação à dimensionalidade, ambos foram utilizados na etapa experimental deste trabalho e são apresentados em detalhes nas seções 3.2.1 e 3.2.2.

Além das abordagens de amostragem tendenciosa que utilizam critérios de densidade, também há propostas que usam critérios de distância para determinar se um elemento será inserido na amostra.

Em Sarma e Viswanath (2009), Sarma, Viswanath e Reddy (2013), os autores utilizaram uma variação do algoritmo de análise de agrupamento Líder para reduzir o volume de dados em uma etapa de pré-processamento. Após essa etapa inicial, o algoritmo k-médias é executado sobre a amostra. O algoritmo Líder garante que os centros de cada partição estejam espaçados por uma distância mínima (determinada por um parâmetro do algoritmo) e são esses centros que irão compor a amostra. Os autores reportam um ganho de velocidade considerável ao utilizar somente os centros de cada partição como amostra, porém, a versão modificada produziu uma diferença insignificante em relação à utilização do algoritmo Líder original. Essa proposta é descrita em mais detalhes na seção 3.1.1.

Em Ros e Guillaume (2016a), Ros e Guillaume (2016b) os autores utilizaram dois algoritmos amostrais que utilizam critérios de distância para comparar com a proposta apresentada. Um deles é o método de inicialização do k-médias++ (Seção 3.1.2); e, o outro algoritmo, é uma heurística proposta para o problema do caixeiro viajante, chamada FFT (*Furthest-first-traversal*) (Seção 3.1.3). Ambos os métodos amostram os elementos de forma iterativa e, a cada passo, os elementos mais distantes dos previamente selecionados possuem uma maior probabilidade de serem inseridos na amostra. No caso do FFT, o elemento mais distante é inserido na amostra – sem sorteio. Assim como o algoritmo amostral utilizando o Líder, esses dois métodos tendem a produzir uma amostra com espaçamento aproximadamente uniforme entre os elementos.

Outra abordagem utilizando critérios de distância é proposta em Dolnicar e Leisch (2004) (Seção 3.1.4). Os autores propuseram um método que cria várias amostras pequenas de modo aleatório uniforme. Então, um algoritmo de agrupamento baseado em distância – como PAM, k-médias ou hierárquico – é executado sobre cada amostra. Os protótipos (centroides ou medoides) de todas as partições encontradas, em todas as amostras, são combinados em um conjunto de dados final. Nesse conjunto de dados final que o algoritmo de agrupamento de interesse é executado. Como padrões ruidosos possuem pouca influência sobre o protótipo, esse algoritmo amostral é pouco afetado devido à existência de padrões ruidosos no conjunto de dados.

Em Xiao et al. (2014), os autores propõe uma forma de amostragem que executa o algoritmo de agrupamento k-médias no conjunto de dados e seleciona os elementos situados longe da média de cada partição, isto é, nas bordas das partições. O método amostral foi utilizado para reduzir a quantidade de elementos necessários para treinar uma máquina de vetor de suporte. Apesar de conceitualmente interessante, a proposta não pode ser utilizada para acelerar os algoritmos de análise de agrupamento, em especial, o k-médias - devido aos próprios problemas de escalabilidade do algoritmo de análise de agrupamento. Em Luchi et al. (2015), Luchi et al. (2016) os autores propuseram um algoritmo genético amostral que retorna um resultado similar ao algoritmo proposto em Xiao et al. (2014), contudo sem a necessidade da execução do algoritmo de agrupamento k-médias no conjunto de dados completo. O algoritmo genético amostral é descrito em detalhes no capítulo 4.

Por fim, a última vertente de algoritmos amostrais são métodos híbridos. Ou seja, adotam tanto critérios de distância quanto critérios de densidade para a construção da amostra. O princípio base desses métodos é computar a densidade da menor quantidade de elementos possível, afinal, essa etapa possui uma complexidade computacional alta. Para evitar esse problema, os métodos dessa categoria criam uma amostra utilizando critérios de distância e, em seguida, descartam os elementos em regiões pouco densas.

Na literatura, alguns trabalhos propõe algoritmos de amostragem com critérios híbridos que se encaixam bem com a descrição dada. Em Feldman e Langberg (2011), os autores propuseram um método híbrido que serviu de base para o DIDES, proposto em Ros e Guillaume (2016b). No trabalho DIDES, os resultados foram excelentes em comparação com os demais métodos testados, obteve o segundo melhor tempo de execução

– perdendo apenas para a amostragem aleatória uniforme – quanto um alto nível de concordância entre os resultados obtidos ao executar o k-médias na amostra e no conjunto de dados original. No mesmo ano os autores criaram uma variação do DIDES, chamado DENDIS (ROS; GUILLAUME, 2016a) ea qual obteve melhores resultados em relação ao seu antecessor nos experimentos realizado pelos autores. Portanto, esse último foi o algoritmo escolhido para participar dos experimentos deste trabalho e é descrito em detalhes na seção 3.3.1.

Uma outra abordagem híbrida foi proposta por Viswanath e Babu (2009) e, ao contrário do DENDIS (ROS; GUILLAUME, 2016a), o método foi desenvolvido especificamente para atuar em conjunto com o algoritmo de agrupamento DBSCAN obtendo excelentes resultados. A proposta, chamada *Rough*-DBSCAN (Seção 3.3.2), executa o algoritmo de agrupamento Líder – que realiza uma única varredura no conjunto de dados – e em seguida estima a densidade dos líderes (elementos no centro de cada partição) para estimar a densidade. Apenas os líderes com densidade acima do requerido pelo DBSCAN são inseridos na amostra.

Em Luchi, Rodrigues e Varejão (2019) os autores propuseram dois algoritmos para o DBSCAN. O primeiro, chamado *Rough**-DBSCAN, é uma variante da proposta original de Viswanath e Babu (2009). Porém, com uma estimativa de densidade mais precisa. O segundo algoritmo amostral para o DBSCAN apresentado no trabalho é o I-DBSCAN. Esse método também tenta criar uma amostra cuja densidade seja satisfatória para o DBSCAN ao mesmo tempo que tenta manter uma estrutura de vizinhança entre os líderes próximos. Além disso, o I-DBSCAN não necessita de sintonia pois, não necessita do parâmetro τ, existente tanto no *Rough*-DBSCAN quanto no *Rough**-DBSCAN. Os dois algoritmos apresentados obtiveram melhores resultados que o *Rough*-DBSCAN e são descritos em detalhes no capítulo 5.

1.3 Objetivo

Embora não exista uma limitação entre o uso em conjunto de métodos de amostragem e algoritmos de agrupamento, é possível que certos algoritmos de agrupamento se beneficiem pela distribuição dos elementos advinda de um método amostral em particular. Afinal, cada método possui um mecanismo amostral diferente e pode produzir

distribuição dos elementos bastante diferente do resultado obtido por outros métodos.

Dependendo da distribuição resultante certos métodos de agrupamento terão mais facilidade em identificar corretamente as partições. Essa hipótese é de difícil validação pois requer uma etapa experimental complexa envolvendo uma série de conjunto de dados, algoritmos de análise de agrupamento, métodos de amostragem, e uma série de parâmetros a serem sintonizados. Até a presente data, não houve realização desse estudo; porém, os resultados experimentais existentes na literatura indicam essa tendência (KOLLIOS et al., 2003; NANOPOULOS; THEODORIDIS; MANOLOPOULOS, 2006; ROS; GUILLAUME, 2016a).

Como a hipótese é muito mais razoável do que a existência de um método amostral cujo desempenho seja superior para qualquer algoritmo de agrupamento independente dos parâmetros utilizados, a elaboração de métodos amostrais para explorar as particularidades de determinados algoritmos de análise de agrupamento é uma abordagem promissora. De fato, há métodos propostos seguindo essa premissa. Por exemplo, em Viswanath e Babu (2009) os autores fizeram isso ao propor um método de amostragem tendenciosa específico para o DBSCAN e que utiliza os parâmetros do algoritmo de agrupamento para a construção da amostra.

Seguindo a ideia apresentada em Viswanath e Babu (2009), esse trabalho possui o objetivo de apresentar algoritmos amostrais para serem utilizados em conjunto com algoritmos de agrupamentos específicos; e testar o desempenho deles contra outras abordagens já estabelecidas. O primeiro método a ser apresentado é um algoritmo genético de amostragem tendenciosa projetado para o algoritmo de análise de agrupamento k-médias, publicados previamente em Luchi et al. (2015), Luchi et al. (2016), e outros dois métodos amostrais, baseados no trabalho de Viswanath e Babu (2009), para o DBSCAN (LUCHI; RODRIGUES; VAREJÃO, 2019).

1.4 Metodologia

O trabalho se iniciou com uma revisão bibliográfica de todos os métodos de amostragem tendenciosa aplicados a análise de agrupamento disponíveis na literatura. O primeiro passo foi a identificação dos algoritmos mais promissores, isto é, selecionar os que possuíam uma complexidade computacional capaz de fornecer algum ganho de desempe-

nho em relação ao algoritmo de agrupamento executado no conjunto de dados completo; além disso, que tivessem resultados promissores publicados.

Durante a revisão foram identificados alguns pontos fracos em algoritmos que inspiraram os três métodos desenvolvidos nessa tese. O algoritmo genético amostral pode ser visto como uma melhoria do método proposto em Qian et al. (2014), retornando resultado similar sem a necessidade de executar no conjunto de dados completo. Já o *Rough**-DBSCAN é uma versão de seu antecessor com uma estimativa de densidade precisa. Por fim, o I-DBSCAN é uma proposta de método com estimativa precisa de densidade que não necessita de parâmetros extrasalém dos necessários para a execução, do DBSCAN.

Os algoritmos mais promissores, juntamente com os métodos propostos, foram implementados na linguagem JAVA e foram testados em conjuntos de dados de domínio público. Dois experimentos foram realizados para comparar os métodos propostos com outros disponíveis na literatura e, medir o nível de acordo obtido entre as soluções. Isto é, o nível de aproximação obtida entre o algoritmo de agrupamento executado na amostra em relação ao resultado obtido no conjunto de dados completo. A métrica *Adjusted Rand-Index* (ARI) (HUBERT; ARABIE, 1985) foi adotada para medir a diferença entre as soluções obtidas.

O primeiro experimento compara os algoritmos de amostragem para o algoritmo de análise de agrupamento k-médias e foi realizado utilizando 15 conjuntos de dados – de tamanhos e dimensionalidade variados – disponíveis no repositório da LIBSVM.

O segundo experimento, para comparar os algoritmos de amostragem para o algoritmo DBSCAN utilizou 12 conjuntos de dados – também da LIBSVM – além de 8 conjuntos de dados gerados sinteticamente.

Para comparar os resultados obtidos através da execução na amostra com o resultado obtido no conjunto de dados completo deve-se conhecer o resultado do algoritmo no conjunto de dados completo. Devido a alta complexidade computacional do DBSCAN e, a inviabilidade de execução do algoritmo em conjuntos de dados de cardinalidade alta, os conjuntos de dados sintéticos foram utilizados. Esses conjuntos de dados sintéticos foram gerados de modo que o resultado obtido pelo algoritmo DBSCAN fosse conhecido, eliminando assim a necessidade de execução do algoritmo de análise de agrupamento no conjunto de dados completo. Os experimentos realizados nesses conjuntos tiveram seus

resultados comparados contra o resultado esperado pelo DBSCAN. A etapa experimental é discutida em detalhes no capítulo 6.

Os métodos propostos apresentaram bom desempenho, tanto em tempo de execução quanto no nível de aproximação obtida. Os resultados dos dois métodos propostos, para serem utilizados em conjunto com o DBSCAN, foram os melhores métodos no experimento envolvendo o DBSCAN; contudo, em relação ao k-médias, o algoritmo genético ficou em segundo lugar, empatado com o *Bagging* (DOLNICAR; LEISCH, 2004); porém, apresentou piores resultados que o DENDIS (ROS; GUILLAUME, 2016a). O resultado superior do DENDIS, muito provavelmente, se deve ao fato de que a abordagem – diferente de outras da literatura e do algoritmo genético proposto – possui um mecanismo que impede a seleção de padrões ruidosos para compor a amostra.

1.5 Trabalhos publicados

MOREIRA, J. C. H.; BARONI, M. D. V.; LUCHI, D.; FABRIS, F.; VAREJAO, F. M. A multidimensional multiple knapsack approach for the problem of loss reduction in electricity distribution. In: XXXV Ibero-Latin American Congress on Computational Methods in Engineering, 2014, Fortaleza. Proc. of XXXV Ibero-Latin American Congress on Computational Methods in Engineering, 2014.

RAUBER, T. W.; VAREJAO, F. M.; MELLO, L. H. S.; LUCHI, D.; ROCHA, V. F. Recursive dependent binary relevance model for multi-label classification. In: Ibero-American Conference on Artificial Intelligence. Springer, Cham, 2014. p. 206-217.

BATISTA, L. M.; LUCHI, D.; VAREJAO, F. M.; RODRIGUES, A.; LIMA, F.; SANTOS, T. O. Electricity Readers Routing Based on Clustering and Communities Detection. In: 2018 IEEE 16th International Conference on Industrial Informatics (INDIN). IEEE, 2018. p. 583-588.

LUCHI, D.; RODRIGUES, A.; OLIVEIRA, W.; VAREJAO, F. M. Genetic sampling k-means for clustering large data sets. In: Iberoamerican Congress on Pattern Recognition. Springer, Cham, 2015. p. 691-698.

LUCHI, D.; RODRIGUES, A.; VAREJAO, F. M.; OLIVEIRA, W. A genetic algorithm approach for clustering large data sets. In: 2016 IEEE 28th International Conference

on Tools with Artificial Intelligence (ICTAI). IEEE, 2016. p. 570-576.

LUCHI, D.; RODRIGUES, A.; VAREJAO, F. M. Sampling approaches for applying DBSCAN to large datasets. Pattern Recognition Letters, v. 117, p. 90-96, 2019.

1.6 Estrutura da tese

Esse trabalho é organizado da seguinte forma: o capítulo 2 apresenta os métodos de análise de agrupamento k-médias e DBSCAN, para qual os algoritmos de amostragem tendenciosa propostos foram desenvolvidos. Em seguida, serão apresentados no capítulo 3 alguns métodos de amostragem tendenciosa existentes na literatura que foram selecionados para a etapa experimental. Os capítulos 4 e 5 contêm os métodos amostrais para análise de agrupamento desenvolvidos nesse trabalho. Por fim, os experimentos computacionais e os resultados encontrados são descritos no capítulo 6 e as conclusões e trabalhos futuros são apresentados no capítulo 7.

Na descrição dos métodos no decorrer da tese, especialmente nos pseudocódigos, foram utilizados dois padrões distintos para os nomes de variáveis, entradas e saídas. Os métodos baseados no DBSCAN (Capítulo 5), além do próprio algoritmo DBSCAN (Seção 2.2), utilizam a mesma notação de Viswanath e Babu (2009) e Luchi, Rodrigues e Varejão (2019). Já o restante dos métodos, por advirem de diversos trabalhos com diferentes notações, foram padronizados para manter uma uniformidade ao longo do texto.

2 Análise de agrupamento

As contribuições desta tese correspondem a três métodos de amostragem tendenciosa para análise de agrupamento. Embora os métodos propostos possam ser utilizados em conjunto com qualquer algoritmo de análise de agrupamento, cada um foi elaborado se beneficiando do funcionamento de um método em particular. Portanto, é importante que os algoritmos de análise agrupamento k-médias e DBSCAN sejam brevemente apresentados juntamente com alguns problemas tipicamente abordados na literatura.

2.1 k-médias

O algoritmo de análise de agrupamento k-médias foi proposto de forma independente por vários autores. Em 1957, o algoritmo foi apresentado como uma técnica de modulação de pulso com a finalidade de representar sinais analógicos em formato digital. O trabalho foi publicado fora do Bell Labs somente décadas mais tarde (LLOYD, 1982).

Em MacQueen et al. (1967), desconhecendo o trabalho de Lloyd (1982), os autores apresentam um método para análise de agrupamento com o nome de k-médias; entretanto, a ideia remonta ao trabalho apresentado pelo matemático polonês Steinhaus (1956). Finalmente, Forgy (1965) apresenta o mesmo algoritmo descrito por Lloyd (1982) antes do mesmo ter sido publicado abertamente. As diferenças entre as propostas são pequenas e todas costumam ser referenciadas na literatura como k-médias.

O nome "k-médias" foi cunhado por MacQueen et al. (1967) para se referir ao método (heurística) de análise de agrupamento proposto em seu trabalho. Contudo, tal termo gera ambiguidade pois também se refere ao problema de otimização que a heurística visa resolver. Tal problema pode ser formulado do seguinte modo: dado um conjunto de elementos $(x_1, x_2, \ldots, x_n)$ onde cada elemento é um vetor d-dimensional, o objetivo é particionar os n elementos em k $(\leq n)$ conjuntos $S = \{S_1, S_2, \ldots, S_k\}$ minimizando a soma das distâncias quadráticas entre os elementos da i-ésima partição S_i e a média de

seus elementos μ_i, ou seja,

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||^2, \tag{2.1}$$

onde $\arg\min_{S}$ representa a partição ótima dos elementos; isto é, que resulte em um valor mínimo para o somatório. A função objetivo representada na equação 2.1 é equivalente a minimizar a soma das variâncias das partições (Equação 2.2).

$$\arg\min_{S} \sum_{i=1}^{k} |S_i| \ \sigma_i^2 \tag{2.2}$$

A equivalência entre as duas funções objetivo é de fácil visualização, afinal, a variância da partição S_i é definida em termos da diferença quadrática dos elementos até a média (μ_i) da partição (Equação 2.3). Substituindo o termo σ_i^2 na equação 2.2 resulta na equação 2.1.

$$\sigma_i^2 = \frac{1}{|S_i|} \sum_{x \in S_i} ||x - \mu_i||^2 \tag{2.3}$$

O problema de otimização descrito acima é reconhecidamente NP-difícil (GAREY; JOHNSON; WITSENHAUSEN, 1982; KLEINBERG; PAPADIMITRIOU; RAGHAVAN, 1998; ALOISE et al., 2009; DASGUPTA; FREUND, 2009; MAHAJAN; NIMBHORKAR; VARADARAJAN, 2009), e mesmo a versão simplificada do problema com os valores de k e d (número de dimensões) fixados, como normalmente as heurísticas propostas executam, o problema ainda possui uma complexidade computacional exponencial, $O(n^{dk+1})$ (INABA; KATOH; IMAI, 1994).

É importante ressaltar que os algoritmos propostos por Lloyd (1982), MacQueen et al. (1967) e outros, são heurísticas para a solução do problema de otimização conhecido como k-médias e não fornecem nenhum fator aproximativo em relação a função objetivo do problema. Assim, como a maioria dos textos encontrados na literatura, o nome "k-médias" será utilizado no decorrer do texto para referenciar o algoritmo de agrupamento; em particular, para remeter à versão proposta em Lloyd (1982).

O k-médias é uma heurística gulosa que refina as estimativas das médias a cada iteração, até convergir para um mínimo local. Em síntese, o algoritmo consiste em três etapas: inicialização, atribuição dos elementos a cada partição e ajuste das médias. Cada uma dessas etapas será descrita sucintamente nos próximos parágrafos.

Na primeira etapa do algoritmo, são escolhidas as médias iniciais para cada partição s_i. A implementação original sorteia k elementos do conjunto de dados e considera seus valores como as médias iniciais. Existem diversos trabalhos na literatura que propõem diferentes mecanismos de inicialização para o algoritmo (ARTHUR; VASSILVITSKII, 2007; BAHMANI et al., 2012; PAVAN et al., 2012; SU; DY, 2004). Uma seleção cuidadosa do ponto de partida do algoritmo leva a uma convergência mais rápida e diminui a chance do algoritmo ficar preso em um mínimo local sub-ótimo. Tal estratégia – de procurar soluções iniciais melhores para diminuir a quantidade de iterações necessárias na etapa de refinamento – é bastante conhecida na literatura de otimização. Para citar um exemplo, a meta-heurística GRASP (*Greedy Randomized Adaptive Search Procedure*) (FEO; RESENDE, 1995) é inteiramente baseada nesse princípio.

A segunda etapa do algoritmo associa cada elemento a uma partição s_i tal que a distância quadrática do elemento até a média da partição seja a menor possível. Seja s_i^t a partição s_i na t-ésima iteração do algoritmo e μ_i^t a média dos elementos de s_i na t-ésima iteração, temos a seguinte regra de associação:

$$s_i^t = \{x_p : ||x_p - \mu_i^t||^2 \leq ||x_p - \mu_j^t||^2 \ \forall j, 1 \leq j \leq k\}, \tag{2.4}$$

onde cada elemento x_p é atribuído a somente uma partição s_i^t. Caso o elemento x_p esteja à mesma distância de duas ou mais médias, o desempate é resolvido de modo arbitrário. Vale notar que, como a raiz quadrada é uma função monotônica crescente a menor distância quadrática também é a menor distância euclidiana. Consequentemente, escolher o grupo com a média mais próxima para atribuir o elemento x_p é equivalente a achar a partição cuja média possui a menor distância quadrática.

A terceira, e última, etapa do algoritmo k-médias é a atualização das médias de cada partição. Como as relações de pertinência entre os elementos e as partições foram alteradas na etapa anterior do algoritmo, as médias das partições devem ser reajustadas. Considerando μ_i^t a média da t-ésima iteração, temos:

$$\mu_i^{t+1} = \frac{1}{|s_i^t|} \sum_{x_j \in s_i^t} x_j. \tag{2.5}$$

A versão proposta por MacQueen et al. (1967) mantém as médias atualizadas a

cada atribuição e pode ser interpretada como uma versão em "tempo real" do algoritmo proposto por Lloyd (1982), que executa um processamento em lote e atualiza as médias somente após processar todas as atribuições no conjunto de dados.

As duas últimas etapas ocorrem em um laço que é executado até que uma das condições de parada do algoritmo sejam satisfeitas. As implementações mais comuns do algoritmo adotam dois critérios de parada. O primeiro critério corresponde à convergência do método; caso duas iterações sucessivas não modifiquem as médias das partições o algoritmo está em um mínimo local e, a partir desse ponto, não haverá nenhuma mudança na solução retornada. Portanto, o algoritmo pode ser interrompido. O segundo critério corresponde a um limite superior do número de iterações do algoritmo.

A razão para a existência do segundo critério decorre da análise de complexidade de pior caso da heurística. Apesar de possuir uma complexidade de tempo polinomial no caso médio, o pior caso do algoritmo é super-polinomial sendo necessárias $2^{\Omega(\sqrt{n})}$ iterações, como apontado por Arthur e Vassilvitskii (2006). Portanto, fixar um número máximo de iterações do algoritmo garante que o pior caso possua complexidade de tempo polinomial. De outro lado, os resultados obtidos para determinadas instâncias podem ser prejudicados por uma parada prematura do algoritmo.

Do ponto de vista da escalabilidade, o tempo de execução esperado para o algoritmo proposto por Lloyd (1982), e a maioria das variações, é de $O(nkdt)$, onde n representa a cardinalidade do conjunto de dados, k a quantidade de partições, d o número de dimensões do problema, e t o número de iterações do algoritmo (HARTIGAN; WONG, 1979; SCHÜTZE; MANNING; RAGHAVAN, 2008). Já foi citado que t possui o pior caso de $2^{\Omega(\sqrt{n})}$. Contudo, como muitas implementações limitam o número de iterações do algoritmo, t torna-se constante, portanto, pode ser desconsiderado no comportamento assintótico do algoritmo. Normalmente as propostas apresentadas para o k-médias para se contornar os problemas de escalabilidade do algoritmo envolvem os fatores n, d, e k.

As técnicas de amostragem impactam a escalabilidade do algoritmo pois diminuem diretamente o termo n. Enquanto que a diminuição da dimensionalidade do problema através de técnicas como análise de componentes principais (PEARSON, 1901) e seleção de características (DASH; LIU, 2000; GUYON; ELISSEEFF, 2003), por impactar diretamente o termo d, também diminui o custo computacional necessário para a execução

do algoritmo. Além dos dois exemplos citados, existem outras abordagens que minimizam o número de cálculos de distâncias que devem ser realizados entre cada elemento para a média de cada partição, isto é, o produto nk. Tais abordagens também diminuem consideravelmente o tempo de execução do algoritmo.

Diversas variações do k-médias foram propostas ao longo dos anos, muitas delas visando melhorar algum aspecto do algoritmo, normalmente algum aspecto de escalabilidade. Em Pelleg e Moore (1999), Kanungo et al. (2002), os autores utilizam árvores k-d como indexadores espaciais a fim de acelerar a segunda etapa do algoritmo para que não seja necessário calcular a distância de todos elementos para todas as médias a cada iteração.

Outra forma elegante de acelerar a execução do algoritmo é eliminando a necessidade de diversos cálculos de distância através do uso da desigualdade triangular (PHILLIPS, 2002; ELKAN, 2003; HAMERLY, 2010; HAMERLY; DRAKE, 2015).

Várias implementações exploram o paralelismo como forma de agilizar o tempo computacional demandado no cálculo das distâncias de cada elemento até a média de uma partição; afinal, são operações completamente independentes (ZHANG et al., 2006).

Também podemos citar diversos trabalhos utilizando técnicas amostrais para grandes conjuntos de dados que não cabem na memória principal dos computadores (LAZAROV; AVERBUCH, 2009; SCULLEY, 2010; BEJARANO et al., 2011).

Em Fahim et al. (2006), os autores fazem o uso de cache para evitar o cálculo desnecessário das distâncias. Pode-se armazenar a distância de cada elemento até a média da partição em que o mesmo está associado e guardar esse resultado em memória. Se a distância para a média da partição em que o elemento está associado diminuir em relação a iteração anterior, o elemento permanece na mesma partição. Desse modo, não é necessário verificar a distância desse elemento para as demais médias.

Diversas modificações na etapa de inicialização do algoritmo já citados anteriormente também foram propostas para agilizar a convergência do algoritmo. A mais famosa delas, o k-médias++ (ARTHUR; VASSILVITSKII, 2007), seleciona as médias iniciais de modo espaçado, minimizando assim a chance de duas médias pertencerem à mesma partição.

Conforme os exemplos citados, fica evidente que existe um grande esforço no sen-

tido de contornar os problemas de escalabilidade do k-médias em diversos trabalhos da literatura. Apesar desta tese também ter como objetivo contornar os problemas de escalabilidade do k-médias, existem variações do método que melhoram outros aspectos do algoritmo e não poderiam deixar de ser citadas.

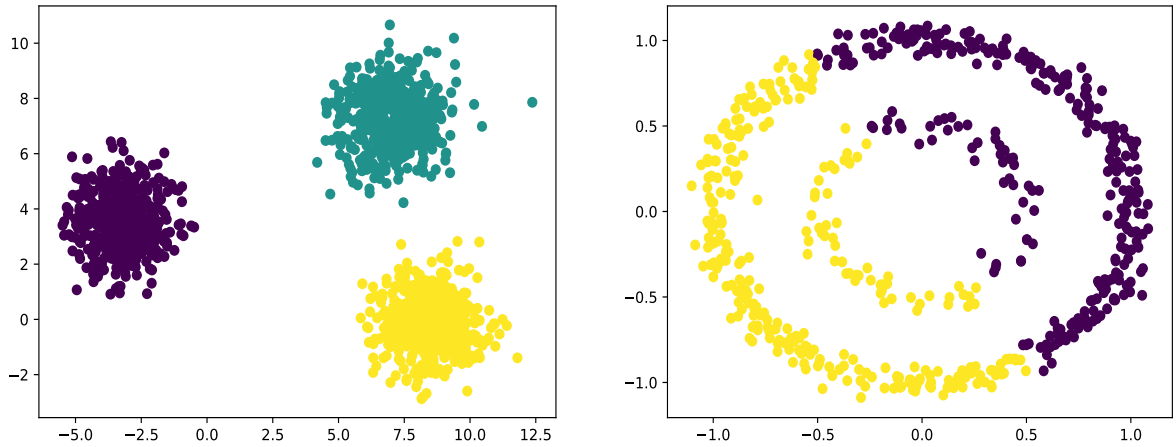
Hartigan e Wong (1979) propuseram uma variação do algoritmo que trocava elementos de uma partição para outra, mesmo que impactasse em um resultado pior na função objetivo naquela iteração. Ao final da execução, a melhor solução encontrada durante a execução do algoritmo é retornada. Assim como muitas meta-heurísticas, essa estratégia tem a finalidade de escapar de mínimos locais indesejados.

Além do número enorme de modificações do algoritmo clássico, o k-médias também inspirou o surgimento de novos métodos. Por exemplo, o trabalho de Rdusseeun e Kaufman (1987), que apresentaram uma heurística para o problema k-medoide. Dunn (1973) propôs uma versão nebulosa do algoritmo conhecida como *Fuzzy C-Means*, em que cada elemento possui um grau de pertinência para cada partição.

Existe também uma variação do problema original utilizado em diversos trabalhos. Tal variação utiliza a mediana, ao invés da média de cada partição, para definir a associação dos demais elementos. Contrastando com a versão original que minimiza a variância, essa abordagem equivale a minimizar os desvios absolutos. Existem outros métodos que podem ser considerados variações, ou generalizações, do k-médias, como o modelo de misturas gaussianas baseado no algoritmo de estimação de maximização da esperança (DEMPSTER; LAIRD; RUBIN, 1977), contudo, a discussão de tais métodos foge ao escopo dessa tese.

Se forem ignorados os problemas de convergência para mínimos locais indesejados, complexidade computacional super-polinomial do algoritmo, incapacidade de lidar com conjunto de dados que não cabem na memória principal; e, se tivesse uma caixa preta que retornasse a solução ótima para o problema de otimização k-médias, alguns problemas ainda surgiriam. Todavia, não por algum problema do método em si. Contudo, dependendo do contexto, a solução do problema de otimização muitas vezes não está alinhada com a expectativa de quem executa o algoritmo. Alguns desses casos serão discutidos a seguir.

Do ponto de vista de análise de agrupamento, o objetivo é a identificação de



(a) Conectividade alta e coesão alta. (b) Conectividade alta e baixa coesão.

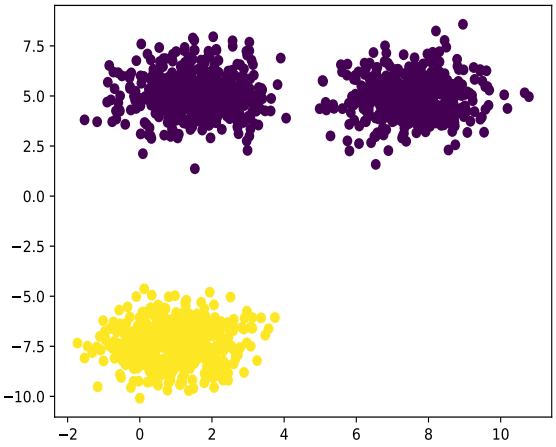
Figura 1 – Exemplos de resultados do k-médias em diferentes conjuntos de dados.

elementos no conjunto de dados que são similares entre si, e não a solução do problema de otimização. Porém, muitas vezes a partição encontrada (com variância mínima) e nossa noção subjetiva de "grupo" estão em desacordo. Na figura 1 pode-se ver dois exemplos de soluções ótimas para problema de otimização k-médias em dois conjuntos de dados distintos.

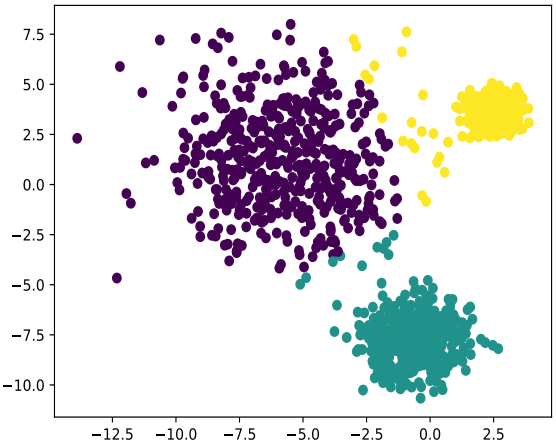
O primeiro exemplo (Figura 1a), exibe o resultado de três partições em acordo com a nossa noção subjetiva de "grupo", enquanto o segundo (Figura 1b), não está alinhado à noção básica de agrupamento. Contudo, vale ressaltar que não é uma falha do método; afinal, a solução ótima foi encontrada em ambos os casos. Porém, é evidente que o critério geométrico utilizado pelo algoritmo não se adapta ao nosso conceito de "grupo" em todos os casos.

Além do caso exemplificado na figura 1b, há outros casos bastante conhecidos que também mostram problemas similares. O primeiro (Figura 2a) acontece quando se tem uma suposição errada em relação a quantidade de partições existentes. Tipicamente, em conjunto de dados desconhecido não se conhece esse parâmetro.

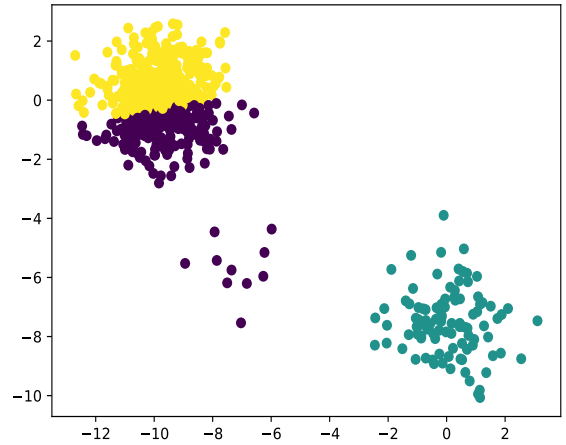
Há várias maneiras de contornar esse problema. Uma delas é através do uso do coeficiente de silhueta (ROUSSEEUW, 1987), comparando o resultado do coeficiente para vários valores de k tem-se uma ideia do número correto de partições existente no conjunto de dados. Contudo, essa métrica só obtém resultados válidos nos conjuntos de dados em que o algoritmo k-médias produz resultados alinhados com nossa noção de agrupamento.



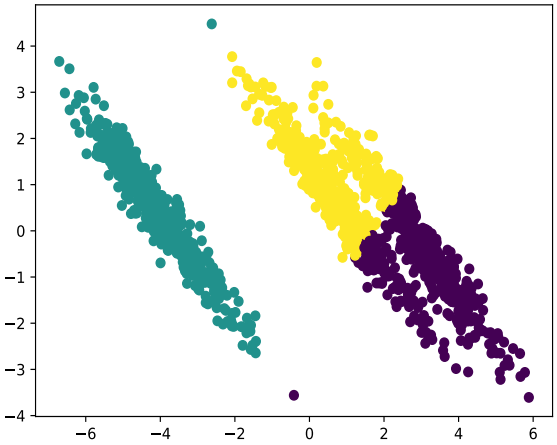
(a) Diferente número de partições.



(b) Partições com diferentes variâncias.



(c) Partições com diferentes cardinalidades.



(d) Covariância das variáveis.

Figura 2 – Problemas encontrados com o k-médias quando o conjunto de dados não atende as premissas assumidas pelo método.

Para o exemplo da figura 1b essa técnica também não seria válida.

Na figura 2b verifica-se que quando existe alguma partição cuja variância é muito maior que das demais partições e localizadas próximas umas das outras, alguns elementos são alocados para as partições erradas. Ocorre assim porque os elementos na borda da partição com maior variância estão mais próximos da média de outra partição do que da média da partição que deveriam estar inseridos; e, consequentemente, tais elementos são atribuídos ao grupo de menor variância.

Noutro caso, exibido na figura 2c, também indesejado, o conjunto de dados consiste em três partições com variância similar, afastadas umas das outras e, ainda assim o resultado, apesar de ótimo, parece contradizer nossa intuição. Tal problema ocorre devido as cardinalidades das partições serem desbalanceadas – e como a partição de menor cardi-

nalidade possui pouco impacto na função objetivo – a função objetivo é mínima se as duas médias forem alocadas para o grupo de maior cardinalidade. Uma forma de minimizar a ocorrência desse tipo de problema é através da aplicação de métodos de amostragem tendenciosa no conjunto de dados, a fim de diminuir a representatividade das partições com maiores cardinalidades.

Por fim, conjunto de dados que possuem uma elevada covariância ou que as partições simplesmente não possuam formatos esferoidais, tendem a ser particionados incorretamente pelo algoritmo (Figura 2d). Caso as partições estejam suficientemente próximas umas das outras, as associações dos elementos às partições poderão ser feitas de modo contraintuitivo, como representado na figura. Felizmente, havendo forte covariância entre as características, técnicas como análise de componentes principais (PEARSON, 1901) ajudam a mitigar o problema. Contudo, tais técnicas não solucionam completamente o problema para outros casos. Afinal, a transformação acontece para todos os elementos do conjunto de dados e, se uma partição for positivamente correlacionada e uma outra for inversamente correlacionada, a eficácia de tal técnica é reduzida.

É importante ressaltar que o desalinhamento entre o conceito subjetivo de grupo e o resultado retornado pelos algoritmos não é uma particularidade do k-médias, vários outros métodos de análise de agrupamento, senão todos, apresentam problemas nesse sentido. O conceito que temos de agrupamento é algo complexo de ser modelado matematicamente ou expressado na forma de um algoritmo e a maioria, senão todas, as formas ilustradas anteriormente acabam, em certas situações, apresentando resultados que contrariam nossa intuição. Todavia, em cenários para qual foram projetados, a maioria dos métodos apresentam excelentes resultados e, por esse motivo, o k-médias é provavelmente o método de análise de agrupamentos mais conhecido e utilizado até hoje.

2.2 DBSCAN

O DBSCAN (*Density-based spatial clustering of applications with noise*) é um método proposto por Ester et al. (1996) de grande importância para a área de análise de agrupamentos. Corresponde ainda a um dos métodos mais estudados e bem sucedidos de análise de agrupamentos desde a publicação do trabalho original (SIGKDD, 2014; XU; WUNSCH, 2005; CHANDOLA; BANERJEE; KUMAR, 2009).

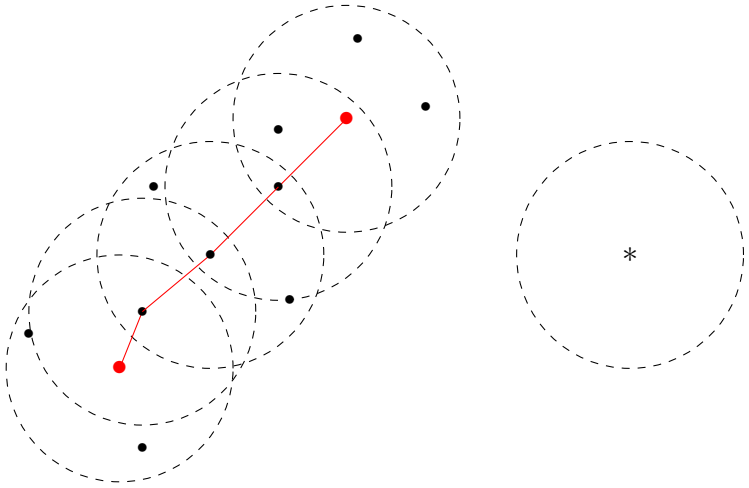


Figura 3 – Exemplo de dois elementos alcançáveis, em vermelho, e um elemento que não é alcançável a partir de nenhum outro, simbolizado por um asterisco na cor preta. Os círculos foram desenhados apenas para os elementos atravessados pela linha em vermelho e para o simbolizado com o asterisco.

Diferente do k-médias, apresentado na seção anterior, o DBSCAN se baseia no conceito de densidade para determinar a partição dos dados e, portanto, não há nenhuma premissa atrelada ao algoritmo a respeito do formato das partições. Adicionalmente, o algoritmo também é capaz de tratar dados ruidosos (KASSAMBARA, 2017). É importante destacar que é difícil expressar matematicamente a função que o algoritmo otimiza pois é um procedimento muito mais complexo que o k-médias.

O DBSCAN verifica se os elementos são densos de acordo com a seguinte regra:

$$\text{denso}(x_i) = \begin{cases} 1, & \text{se } |\text{VIZINHANÇA}(D, x_i, \epsilon)| \geq \text{minPts} \\ 0, & \text{caso contrário,} \end{cases} \tag{2.6}$$

onde x_i é um elemento do conjunto de dados D, minPts o número mínimo de elementos que deve existir na vizinhança de x_i para que seja considerado denso, e ϵ representa a distância máxima que o elemento deve estar de x_i para ser considerado vizinho. Em outras palavras, ϵ representa o raio da vizinhança, e $\text{VIZINHANÇA}(D, x_i, \epsilon)$ simboliza a função que retorna todos os elementos vizinhos do elemento x_i. Caso seja retornado um número maior que minPts vizinhos o elemento é considerado denso.

Uma outra caraterística muito importante do DBSCAN, que inexiste na maioria dos outros métodos de análise de agrupamentos, é a capacidade de identificação de elementos ruidosos. Mas, antes de uma definição mais formal do que caracteriza um elemento ruidoso, é necessário definir o conceito de alcançabilidade. Dizemos que um elemento x_i

Entrada: Conjunto de dados $\mathcal{D}$, raio ϵ, número mínimo de elementos $minPts$
Saida: Conjunto de rótulos $\{\text{rotulo}_i, i = 1, \ldots, |\mathcal{D}|\}$

```
1  C ← 0;
2  para cada d ∈ D faça
3      se rótulo_d é indefinido então
4          V_d ←VIZINHANCA(D, d, ε);
5          se |V_d| < minPts então
6              rótulo_d ← ruído;
7          senão
8              C ← C + 1;
9              rótulo_d ← C;
10             S ← V_d \ {d};
11             para cada q ∈ S faça
12                 se rótulo_q é ruído então
13                     rótulo_q ← C;
14                 se rótulo_q é indefinido então
15                     rótulo_q ← C;
16                     V_q ←VIZINHANÇA(D, q, ε);
17                     se |V_q| ≥ minPts então
18                         S ← S ∪ V_q;
19 retorne {rótulo_i (i = 1, ..., |D|)}
```

Algoritmo 1: DBSCAN (SCHUBERT et al., 2017).

é alcançável a partir de x_j se existir uma sucessão de vizinhos densos que interliguem os dois elementos. A figura 3 exemplifica o conceito de alcançabilidade.

A alcançabilidade é o mecanismo pelo qual o DBSCAN particiona o conjunto de dados. Todos os elementos alcançáveis a partir de um elemento denso x_i pertencerão a mesma partição que x_i, enquanto todos os elementos não alcançáveis por nenhum elemento denso são rotulados como ruído. Normalmente padrões ruidosos se situam em regiões de baixa densidade, portanto, esse mecanismo é efetivo na identificação do ruído. Contudo, se um elemento ruidoso estiver inserido em uma região densa, por exemplo, inserido em um outro "grupo", o mesmo será rotulado como um membro daquele grupo. O algoritmo 1 mostra o pseudocódigo do método tal como apresentado em Schubert et al. (2017).

Com uma rápida explicação do pseudocódigo pode-se ver como o DBSCAN define as partições. Nas linhas 2 e 3 o algoritmo percorre todos os elementos do conjunto de dados verificando se existe um rótulo já associado ao elemento. Inicialmente nenhum elemento possui rótulo. Caso o elemento d não possua rótulo, será feita uma busca de

todos os elementos vizinhos de d, chamado V_d (linha 4) e em seguida, é verificado se a cardinalidade da vizinhança é superior a minPts (linha 5). Caso não seja, o elemento é temporariamente classificado como ruído (linha 6).

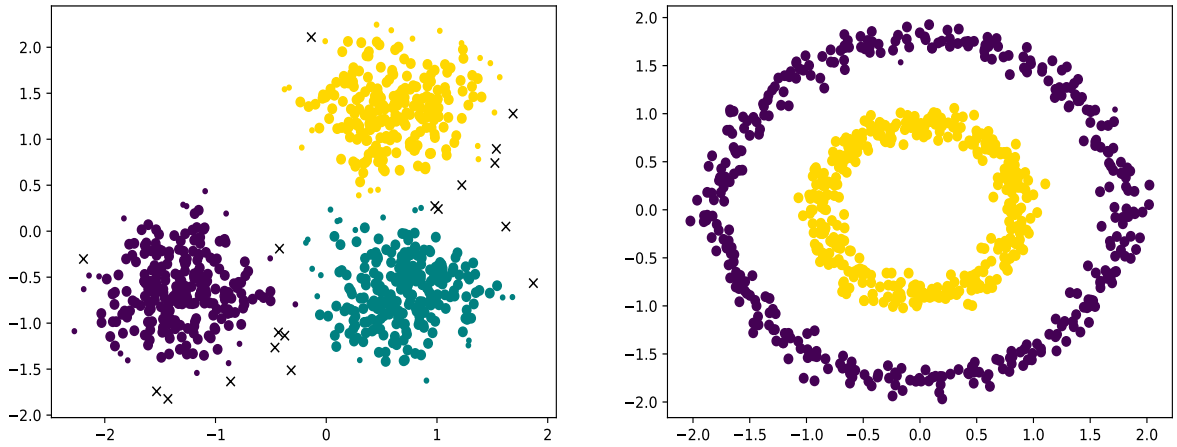
Se d for denso (linha 7) ele é automaticamente atribuído a uma nova partição (linha 9) e um laço é executado para todos os elementos contidos em S (linha 11). Para cada elemento q, dentro do laço, é verificado se o elemento foi temporariamente rotulado como ruído ou se ainda não possui um rótulo. Em ambos os casos, como o elemento é alcançável, será atribuído a mesma partição que d (linhas 13 e 15).

Na primeira iteração do laço, S é composto apenas pelos vizinhos de d; contudo, toda vez que um elemento q que não possuir rótulo for atribuído a partição de d, todos os vizinhos de q, caso q seja denso também, serão adicionados em S. Essa última etapa é equivalente a realizar uma busca em largura; inicialmente, verificam-se os elementos vizinhos, depois vizinhos de vizinhos, e assim por diante. Interrompendo quando todos os elementos alcançáveis tiverem sido atribuídos.

Em seguida o algoritmo volta ao laço principal a procura de um novo elemento denso ainda sem rótulo. O processo se repete até que o laço principal tenha iterado sobre todos os elementos do conjunto de dados. Os elementos que forem classificados temporariamente como ruído, e não foram atribuídos a nenhuma partição, permanecerão como ruído na solução retornada.

É importante lembrar que, apesar de determinístico, o DBSCAN é um pouco sensível à ordem em que os dados são apresentados ao algoritmo. Caso exista um elemento não-denso situado na borda de duas partições, esse pode ser atribuído a uma, ou outra, dependendo do ordenamento do conjunto de dados.

Pelo exposto, duas propriedades das partições formadas pelo DBSCAN podem ser inferidas: a primeira garante que todas as partições são formadas por elementos mutuamente alcançáveis; e a segunda, que todos os elementos densos alcançáveis a partir de qualquer elemento denso de uma partição arbitrária também pertencerá a partição. A primeira propriedade garante uma coesão dos grupos, enquanto que a segunda garante a completude. Na figura 4 podemos ver a versatilidade do DBSCAN nos dois conjuntos de dados apresentados na seão anterior em que o k-medias não obtém os resultados apropriados. Em ambos os casos o DBSCAN alcança os resultados desejados.



(a) Conectividade alta e coesão alta. (b) Conectividade alta e baixa coesão.

Figura 4 – Exemplos de resultados do DBSCAN em diferentes conjuntos de dados. Os elementos marcados como "x" foram rotulados como ruído pelo algoritmo e os círculos de menor diâmetro representam os elementos não-densos.

O DBSCAN possui diversas pontos positivos. Um deles é que não há a necessidade de conhecer a quantidade de partições existentes no conjunto de dados, ou tampouco assumir que as partições sejam de um formato em particular. Contudo, o algoritmo também possui pontos negativos; os dois parâmetros necessários para a execução do algoritmo são de difícil estimativa, principalmente com um número elevado de dimensões.

Para exemplificar, se for atribuído um valor pequeno para ϵ, ou grande para minPts, pode ocorrer de todos os elementos do conjunto de dados serem rotulados como ruído; portanto, nenhuma partição seria identificada. O contrário também ocorre, com um valor alto para ϵ todos os elementos do conjunto de dados podem ser considerados como uma única partição, pois o conjunto de dados inteiro será alcançável a partir de um elemento denso arbitrário.

Um valor de minPts $= 1$ e ϵ pequeno irá fazer com que um número muito grande de partições sejam detectadas, talvez tantas partições quanto a cardinalidade do conjunto de dados. Apesar dos exemplos citados serem casos extremos, a estimativa incorreta pode facilmente agrupar dois grupos diferentes em um único, ou dividir uma partição coesa em mais de um grupo. Portanto, um ajuste cuidadoso dos parâmetros é necessário.

Vários trabalhos foram publicados com o objetivo de melhorar esse problema, ora propondo a seleção automática de parâmetros (ZHOU; WANG; LI, 2012; SAWANT, 2014; KARAMI; JOHANSSON, 2014), ora o uso de versões livres de parâmetros (CHEN et al.,

2011; HOU; GAO; LI, 2016).

Outra característica positiva do método é que os formatos das partições não são afetados pelos elementos rotulados como ruído. Fazendo um paralelo com k-médias, um elemento muito longe da média de uma partição irá deslocar a média da partição e, consequentemente, terá impacto na associação dos elementos restantes. Afinal, a decisão é tomada inteiramente com base na distância até a média. Em contrapartida, no DBSCAN, o elemento longe da partição é simplesmente ignorado pelo algoritmo pois o elemento não altera nenhuma relação de alcançabilidade entre os elementos da partição e os demais elementos do conjunto de dados.

Em Kriegel et al. (2011), os autores mostram que, devido aos parâmetros do algoritmo serem aplicados para todo o conjunto de dados, partições com densidades muito discrepantes tornam-se um problema. Liu, Zhou e Wu (2007) propuseram uma versão capaz de lidar com partições de densidade variáveis.

No entanto, as modificações propostas para o DBSCAN são muito mais numerosas no sentido de tentar acelerar a execução do método. A maioria delas é focada em diminuir o custo computacional da busca de vizinhos. O DBSCAN possui uma complexidade de tempo de $O(n^2)$. A operação de retornar todos os vizinhos de um elemento, que é realizada várias vezes ao longo da execução do algoritmo é responsável por esse custo computacional. De modo ingênuo, deve-se computar a distância de todos os elementos até o elemento de interesse e retornar àqueles que possuírem uma distância menor que ϵ. Muitos trabalhos propõem alternativas para mitigar o custo computacional dessa busca.

As alternativas mais comuns para acelerar a busca em vizinhança são o uso de estruturas de dados na criação de indexadores espaciais, como árvores k-d (BENTLEY, 1975) em (RUIZ; SPILIOPOULOU; MENASALVAS, 2007; PATWARY et al., 2012), e árvores R (GUTTMAN, 1984) em (KUMAR; REDDY, 2016). Outras alternativas envolvem usar um reticulado sobre o conjunto de dados buscando os elementos apenas nas células vizinhas para agilizar a busca de elementos (UNCU et al., 2006), ou usar a vantagem da desigualdade triangular evitando cálculos de distância desnecessários (CHEN et al., 2018).

Existem centenas de trabalhos na literatura com diferentes estruturas de dados para indexar espacialmente os elementos do conjunto de dados a fim de melhorar o de-

sempenho na consulta dos elementos vizinhos. Embora a maioria obtenha um bom desempenho em conjuntos de dados com um baixo número de dimensões, o ganho fornecido pela maioria rapidamente desaparece quando o conjunto de dados possui uma alta dimensionalidade (CHÁVEZ et al., 2001; ARLIA; COPPOLA, 2001).

Há também variações aproximadas do DBSCAN que utilizam busca em vizinhança aproximada (DATAR et al., 2004). Tais técnicas retornam, na maioria dos casos, os elementos na região, mas não há a garantia que o resultado encontrado será exatamente o mesmo (WU; GUO; ZHANG, 2007). Em contra partida, tais abordagens possuem uma complexidade computacional baixa, em muitos casos $O(n)$, e o desempenho não se degrada com o número elevado de dimensões.

Existem trabalhos que propõem variações do DBSCAN usando técnicas de amostragem tendenciosa – foco dessa tese. Em Al-Mamory et al. (2016), os autores propõem um método amostral para o DBSCAN muito similar ao método de amostragem tendenciosa (*Density Biased Sampling*) de Palmer e Faloutsos (2000). O método proposto possui um custo computacional elevado e é incapaz de identificar dados ruidosos.

Em Borah e Bhattacharyya (2004), os autores utilizam um critério de distância na busca em largura realizada pelo DBSCAN para evitar selecionar elementos próximos uns dos outros, limitando assim a largura da árvore e melhorando a escalabilidade do método.

Há também variantes do DBSCAN que utilizam abordagens híbridas, isto é, um algoritmo de análise de agrupamento de menor complexidade executado em uma etapa anterior. Em El-Sonbaty, Ismail e Farouk (2004), os autores utilizaram o algoritmo de análise de agrupamento CLARANS (NG; HAN, 2002) para fazer uma pré-partição do conjunto de dados formando regiões menores, e executar o DBSCAN em cada uma para, posteriormente, unir os resultados. Devido a complexidade quadrática do método, a execução em várias instâncias menores leva muito menos tempo do que a execução na instância completa.

Em Viswanath e Babu (2009), os autores também propõem uma abordagem híbrida utilizando o algoritmo de análise de agrupamento Líder. O algoritmo Líder realiza uma única varredura sobre o conjunto de dados selecionando um conjunto de elementos que serão os líderes (amostras) utilizados pelo algoritmo. O DBSCAN é executado nessa amostra e, ao final, os elementos que foram excluídos do conjunto de dados são inseridos

novamente associados no mesmo grupo de seu líder.

Essa abordagem possui uma complexidade computacional de $O(nk^2)$ e, como verificado em Khan et al. (2014), apresenta excelentes resultados. Uma das variantes propostas neste trabalho para o DBSCAN é uma melhoria direta dessa proposta obtida através de uma estimativa melhor das densidades dos líderes (amostras).

É importante ressaltar que executar o DBSCAN em uma amostra do conjunto de dados é uma tarefa complexa em razão das mudanças requeridas nos parâmetros do algoritmo. O k-médias usa o mesmo valor do parâmetro k seja executado sobre uma amostra ou sobre o conjunto de dados completo. De outro lado, no DBSCAN os parâmetros utilizados no conjunto de dados completo provavelmente não serão válidos para a amostra; afinal, a amostra terá uma densidade de elementos menor que o conjunto de dados completo. Portanto, é muito provável que muitos elementos sejam rotulados como ruído.

Além dessas variações específicas, existem diversos métodos de amostragem tendenciosa propostos especificamente para a análise de agrupamento. Alguns selecionados serão apresentados no capítulo 3. Essa triagem foi realizada considerando o conhecimento necessário para se executar o DBSCAN na amostra (parâmetros) e a complexidade computacional do método amostral.

3 Métodos de amostragem tendenciosa

Pelo exposto anteriormente é evidente que tanto o k-médias quanto o DBSCAN possuem uma alta complexidade computacional e, portanto, apresentam uma escalabilidade inadequada para conjuntos de dados de alta cardinalidade. A amostragem continua sendo uma técnica utilizada para contornar esse problema.

A amostragem tendenciosa, ao contrário da amostragem aleatória uniforme, não produz uma amostra com uma distribuição semelhante à existente no conjunto de dados original. Ao invés disso, formula-se uma tendência que gera uma probabilidade maior de certos elementos serem sorteados. É difícil generalizar quais elementos são privilegiados, afinal, cada método amostral altera as probabilidades de sorteio de um modo diferente. Certos métodos priorizam regiões menos densas, outros evitam selecionar elementos próximos dos previamente selecionados, e ainda existem abordagens que tendem a escolher elementos nas periferias das partições para compor a amostra.

Como existem muitos métodos de amostragem tendenciosa na literatura, apenas alguns foram escolhidos. Essa triagem foi realizada utilizando três critérios. O primeiro, e talvez mais importante, é a complexidade computacional do método de amostragem. Se o método de amostragem for lento, isto é, possuir a mesma complexidade computacional que o algoritmo de análise de agrupamento, não há sentido em utilizar tal método de amostragem.

O segundo critério adotado corresponde à informação necessária para executar o algoritmo amostral, tendo por base apenas o conhecimento dos parâmetros do algoritmo de agrupamento. Por exemplo, caso o usuário deseje executar o algoritmo DBSCAN em um conjunto de dados, ele deve conhecer os parâmetros minPts e ϵ. Nesse caso, não faz sentido a utilização de um método amostral que requer que o usuário especifique a quantidade de partições existentes (k) no conjunto de dados, como o algoritmo genético proposto no capítulo 4. Por fim, o último critério leva em consideração a qualidade dos resultados reportados em outras publicações.

O experimento realizado em Ros e Guillaume (2016a) compara o algoritmo proposto com outras doze técnicas de amostragem e é o experimento mais completo dis-

ponível na literatura. Os métodos amostrais comparados foram: amostragem aleatória uniforme, duas versões do método amostral utilizando o Líder (LING, 1981; SARMA; VISWANATH; REDDY, 2013), amostragem baseada no k-médias (XIAO et al., 2014), o método de inicialização do *k-means++* (ARTHUR; VASSILVITSKII, 2007), o método de amostragem tendenciosa usando *kernel* apresentado em (KOLLIOS et al., 2001; KOLLIOS et al., 2003), o DBS (PALMER; FALOUTSOS, 2000), a amostragem utilizando o *k-nearest-neighbors* para estimar a densidade (FRANCO-LOPEZ; EK; BAUER, 2001), *Tree sampling* (ROS et al., 2003), amostragem usando agregação do bootstrap (DOLNICAR; LEISCH, 2004), o FFT (ROSENKRANTZ; STEARNS; LEWIS II, 1977), e um método híbrido apresentado em (FELDMAN; FAULKNER; KRAUSE, 2011).

Muitos métodos amostrais utilizados nos experimentos apresentados em Ros e Guillaume (2016a) não foram incluídos nesse capítulo por apresentarem um desempenho computacional ruim em termos da métrica (*Rand-Index*) ou devido ao elevado tempo de execução. Dentre eles, pode-se destacar o algoritmo amostral que calcula a densidade dos elementos usando *kernel*, apresentado em Kollios et al. (2003), que apesar de conseguir resultados de qualidade apresentou um tempo computacional excessivamente alto. Esse tempo computacional elevado é corroborado por outros trabalhos da literatura como em Nanopoulos, Theodoridis e Manolopoulos (2006), Appel (2010).

Além do método proposto em Kollios et al. (2003), o algoritmo amostral baseado no k-médias (XIAO et al., 2014) também foi excluído dos experimentos (Capítulo 6). O motivo da exclusão é o fato que esse método amostral precisa executar o k-médias no conjunto de dados completo para retirar uma amostra do mesmo. Portanto, não existe sentido na utilização dessa proposta.

Outro método deixado de fora devido ao elevado tempo de execução foi um método de amostragem que calcula a densidade dos elementos de modo ingênuo, proposto em Franco-Lopez, Ek e Bauer (2001). Isto é, através da execução do k-vizinhos mais próximos para cada elemento do conjunto de dados para estimar a densidade da região que compreende esses k vizinhos. Esse método foi um dos primeiros métodos amostrais a serem propostos e serve mais como prova de conceito do que para qualquer uso prático. De fato, o método apresenta tempos tão elevados que, na maioria das vezes, é mais rápido executar o algoritmo de análise de agrupamento no conjunto de dados completo.

A variação do algoritmo amostral usando o Líder, proposto em Sarma, Viswanath e Reddy (2013), também foi utilizado nos experimentos em Ros e Guillaume (2016a) e, como explicado (Seção 3.1.1), apresenta pouca diferença para a proposta amostral utilizando somente os líderes. Os experimentos realizados em Ros e Guillaume (2016a) confirmam essa similaridade. Além disso, a variação do líder apresentou um tempo de execução bastante elevado e, por esses motivos foi deixado de fora da comparação experimental.

Por fim, o algoritmo amostral proposto em Feldman e Langberg (2011) apresenta uma proposta similar ao DENDIS (ROS; GUILLAUME, 2016a). Na realidade os autores do DENDIS se inspiraram nesse método. Todavia, o DENDIS superou seu precursor tanto em tempo de execução quanto em qualidade dos resultados. Em razão da qualidade dos resultados optou-se pela escolha do DENDIS.

Além desses algoritmos, utilizados nos experimentos em Ros e Guillaume (2016a), Ros e Guillaume (2016b), outros métodos também foram descartados. Em Appel et al. (2007a), os autores utilizam um reticulado sobre o conjunto de dados para estimar a densidade de cada região. O método apresentado obteve um elevado tempo de execução, além de possuir uma complexidade de espaço elevada, por demandar um número exponencial de células em relação a dimensionalidade do conjunto de dados, tornando inviável sua aplicação em conjuntos de dados com um número elevado de dimensões. O método amostral proposto em Mitra, Murthy e Pal (2002) também foi excluído da etapa experimental pois, de modo similar ao proposto em Franco-Lopez, Ek e Bauer (2001), também estima a densidade de modo ingênuo.

De modo similar a abordagem apresentada em Appel et al. (2007a), outros algoritmos também apresentam essa perda de desempenho em relação ao número de dimensões do conjunto de dados. Em Kerdprasop, Kerdprasop e Sattayatham (2005), os autores utilizam um reticulado sobre os dados e, de forma similar a Appel et al. (2007a), enfrentam os mesmos problemas. Já em Nanopoulos, Theodoridis e Manolopoulos (2006), os autores utilizam uma árvore para criar índices espaciais para auxiliar no cálculo de densidade. No entanto, o desempenho da estrutura de dados (de modo similar às árvores k-d) se degrada rapidamente com o aumento dimensionalidade. Por esse motivo, esses métodos amostrais também ficaram de fora da etapa experimental no capítulo 6.

Os métodos amostrais foram organizados em três grupos, de acordo com o critério

de seleção dos elementos para compor a amostra: distância, densidade e híbrido. Nas seções seguintes os algoritmos amostrais selecionados serão apresentados.

3.1 Métodos de amostragem tendenciosa utilizando critérios de distância

Os métodos de amostragem tendenciosa apresentados nessa seção utilizam critérios de distância para realizar a amostragem. Em outras palavras, diminuem (ou tornam nula) a chance de seleção de elementos próximos aos selecionados previamente.

Como característica principal, pode-se destacar a velocidade de execução e uma forte tendencia a incluir os elementos situados em regiões pouco densas (ruídos) do conjunto de dados – devido a distância que se situam dos demais elementos. Nas seções seguintes alguns representantes dessa categoria serão discutidos em detalhes.

3.1.1 Amostragem com o algoritmo de análise de agrupamento Líder

Em Sarma e Viswanath (2009), Sarma, Viswanath e Reddy (2013), os autores utilizaram uma variação do algoritmo de agrupamento Líder (HARTIGAN, 1975) com o propósito de reduzir o tempo de execução do k-médias através da redução da cardinalidade do conjunto de dados. O procedimento de amostragem foi utilizado em uma etapa anterior à execução do algoritmo, isto é, como pré-processamento. Após a construção da amostra, o algoritmo k-médias é executado e, ao final da execução, os demais elementos do conjunto de dados são associados ao grupo cuja média esteja a uma distância menor.

Para entender o funcionamento do algoritmo Líder é necessário entender a noção de líderes e seguidores utilizada no algoritmo. No jargão do algoritmo, os líderes são os elementos que representam seus respectivos grupos. Todos os demais elementos são seguidores, isto é, membros de algum grupo. Nos trabalhos citados, apenas os elementos rotulados como líderes compunham a amostra.

O algoritmo de análise de agrupamento Líder executa uma única varredura, de forma sequencial, sobre o conjunto de dados. Na inicialização do algoritmo, o primeiro elemento do conjunto de dados é rotulado como líder. A partir disso, o algoritmo percorre todo conjunto de dados verificando se o elemento corrente se encontra a uma distância

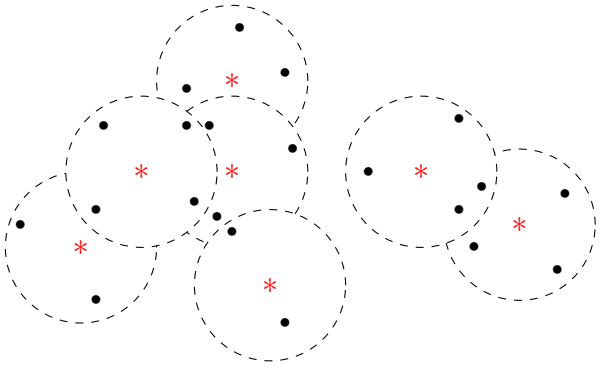


Figura 5 – Exemplo de resultado obtido pelo algoritmo Líder. Os asteriscos, marcados em vermelho, são os líderes e os demais pontos, em preto, representam os seguidores dentro da região do líder (raio τ).

inferior a τ de algum líder definido previamente.

Caso não haja líderes próximos ao elemento, ou seja, uma distância maior que τ, o elemento é rotulado como líder e passa a representar aquela região. Por outro lado, se o elemento estiver nas proximidades de um ou mais líderes ele será um seguidor do primeiro líder encontrado cuja distância seja inferior a τ. Na seção 5.1 o algoritmo Líder é discutido em mais detalhes. Por hora, a descrição simplificada apresentada é suficiente.

O único parâmetro necessário para a execução do algoritmo é o raio τ. Esse parâmetro representa a distância mínima em que os líderes devem estar um dos outros. A figura 5 ilustra o resultado obtido pelo Líder. Ao analisar a figura, pode-se notar a relação inversa entre o valor de τ e a quantidade de elementos rotulados como líderes. Isto é, quanto maior o valor de τ menor a quantidade de líderes.

No contexto de amostragem, devido ao único parâmetro do Líder ser um limiar de distância, esse método amostral não possui um controle direto sobre a cardinalidade da amostra, a qual vai variar em função de τ e da distribuição do conjunto de dados.

Devido ao espaçamento mínimo obrigatório entre os elementos da amostra, fica evidente que regiões mais densas terão baixa representatividade na amostra. Em contrapartida, qualquer elemento distante dos demais, como um padrão ruidoso, certamente estará presente na amostra devido a ausência de algum líder nas proximidades. Na prática, é como se os elementos da amostra estivessem distribuídos aproximadamente de forma homogênea no espaço independente da distribuição real do conjunto de dados original.

Do ponto de vista de escalabilidade, esse procedimento amostral é excelente e possui uma complexidade computacional de $O(n + k^2)$, onde n representa a cardinalidade

do conjunto de dados e k a quantidade de líderes encontrados pelo algoritmo. Como mencionado, k possui uma relação inversamente proporcional ao valor de τ e pode ser ajustado para aproximar o tamanho amostral desejado.

Em Sarma, Viswanath e Reddy (2013), os autores propõem uma variação do algoritmo original que considera que cada líder possui um próprio limite de distância τ. O valor é dinamicamente ajustado ao longo da execução do algoritmo; contudo, o ganho oferecido por essa abordagem foi marginal e, em alguns casos, inexistente em relação ao Líder original.

Portanto, o acréscimo em complexidade do método não é justificado e a versão canônica do método foi utilizada nesse trabalho. Tanto a abordagem com o algoritmo Líder original quanto a versão modificada (SARMA; VISWANATH; REDDY, 2013) apresentam um custo computacional reduzido em relação ao k-médias, devido ao número de líderes ser muito inferior à quantidade de elementos do conjunto de dados completo, e podem ser utilizados para acelerar a execução do algoritmo.

Apesar dessa técnica amostral ter sido proposta como um método amostral para o k-médias, em Ros e Guillaume (2016a), os autores relatam uma grande dificuldade de sintonia do parâmetro τ – devido a falta de conhecimento do conjunto de dados. Os resultados exibidos pelos autores apontam esse algoritmo como o de mais difícil parametrização, dentre os analisados, para ser utilizado em conjunto com o k-médias.

Os resultados e conclusões dos autores fazem sentido em relação a aplicação ao k-médias. No entanto, este método amostral é similar ao *Rough*-DBSCAN (Seção 3.3.2), o qual é de fácil parametrização em relação ao DBSCAN. Por este motivo, na etapa experimental, a amostragem com o algoritmo de análise de agrupamento Líder será utilizado somente com o DBSCAN.

3.1.2 Amostragem com a inicialização do k-médias++

No k-médias, o procedimento de inicialização das médias é feito através do sorteio de elementos de modo aleatório uniforme no conjunto de dados. Esse procedimento, descrito na seção 2.1, acarreta em uma convergência mais lenta e possui uma chance maior de o algoritmo convergir para um mínimo local indesejado. Para contornar esse problema, em Arthur e Vassilvitskii (2007), os autores descrevem um método alternativo de inicialização

para atribuir as médias iniciais.

O modo proposto também utiliza o sorteio de elementos, contudo, o procedimento dá uma chance maior para os elementos distantes dos selecionados previamente. A vantagem dessa proposta é minimizar a chance de duas médias iniciais pertencerem à mesma partição, evitando assim mínimos locais indesejados. Além disso, outra consequência é a diminuição da quantidade de iterações necessárias para o algoritmo atingir a convergência; afinal, o processo de inicialização das médias provavelmente as alocará em partições diferentes. O algoritmo k-médias utilizando esse procedimento de inicialização é chamado de k-médias++.

No procedimento amostral utilizando o k-médias++ a amostra é formada pelos elementos sorteados durante o procedimento de inicialização proposto em Arthur e Vassilvitskii (2007). Apesar de ser um algoritmo bem conhecido e difundido na literatura, o algoritmo 2 mostra as etapas executadas até que k elementos do conjunto de dados tenham sido sorteados.

O algoritmo inicia sorteando de modo aleatório uniforme um elemento do conjunto de dados (linha 1) e em seguida executa o laço principal $k-1$ vezes (linha 2), em cada iteração do laço um novo elemento é sorteado para compor a amostra. O primeiro laço interno (linha 4) soma a distância quadrática entre cada elemento do conjunto de dados para o elemento previamente selecionado ($x_j \in Y$) mais próximo (linha 6). Todos os elementos do conjunto de dados têm a distância mínima calculada nessa etapa do algoritmo – incluindo os elementos já selecionados. Contudo, os elementos selecionados não possuem nenhuma chance de serem escolhidos devido a possuírem uma distância zero em relação ao selecionado mais próximo.

Em seguida, um valor de 0 até a distância quadrática total é sorteado (linha 7). Por fim, no segundo laço interno (linha 8), esse valor é decrementado da distância quadrática enquanto for positivo. Quando se tornar negativo, o elemento é selecionado. Note que elementos distantes dos selecionados irão decrementar a variável aux de um valor muito maior, e portanto, possuem uma maior chance de serem selecionados.

Apesar de fundamentalmente diferente do algoritmo amostral utilizando o Líder, descrito na seção anterior, o resultado do processo amostral é, de certa forma, similar. Caso o tamanho amostral seja suficientemente grande, o método produz uma distribuição

Entrada: $X = \{x_i\}, (i = 1, \ldots, n), k, (1 \leq k \leq n)$
Saida: $Y = \{y_i \in X\}, (i = 1, \ldots, k)$

```
1  Y ← {x_rand(1,n)};
2  para i ← 1 até k − 1 faça
3      soma ← 0;
4      para j ← 1 até n faça
5          d ← min(dist(x_j, Y));
6          soma ← soma + d²;
7      aux ← rand(0, soma);
8      para j ← 1 até n faça
9          aux ← aux − min(dist(x_j, Y))²;
10         se aux ≤ 0 então
11             Y ← Y ∪ {x_j};
12             interrompa ;
13 retorne Y
```

Algoritmo 2: Algoritmo amostral k-médias++ (ARTHUR; VASSILVITSKII, 2007).

semelhante à distribuição uniforme; afinal, a chance de dois elementos próximos estarem presentes na amostra é muito pequena. Elementos mais isolados dos demais, típico de padrões ruidosos, quase certamente pertencerão ao conjunto de elementos sorteados.

A diferença do método no contexto de amostragem, em relação à sua utilização na inicialização do algoritmo de análise de agrupamento é, basicamente, o valor atribuído ao parâmetro k. Quando utilizado para inicialização do k-médias++ o algoritmo itera poucas vezes, selecionando um elemento por partição do conjunto de dados. Se tratando de amostragem, o valor de k é tipicamente muito mais elevado, pois k, nesse contexto, representa o tamanho amostral fazendo com que o número de iterações seja, na prática, muito maior.

Um ponto positivo desse método é precisar muito pouco conhecimento de algum parâmetro do conjunto de dados, apenas o valor k, que define o tamanho da amostra. Por conta disso, o usuário tem o controle exato do tamanho da amostra. Não há a necessidade de conhecer quantos grupos existem, tampouco escolher algum parâmetro que influencie fortemente o tamanho da amostra (como a distância τ no Líder).

O procedimento amostral baseado na inicialização do k-médias++ possui uma complexidade de tempo relativamente alta em relação ao Líder, pois executa várias varreduras sobre o conjunto de dados. No caso do Líder, como o parâmetro de distância

é conhecido de antemão, é possível realizar uma única varredura no conjunto de dados. Já no k-médias++ deve-se realizar uma varredura para cada elemento a ser selecionado na amostra. Além disso, esse procedimento é de difícil paralelização (BAHMANI et al., 2012). Em Bahmani et al. (2012), os autores propõem uma alternativa paralelizável do k-médias++, contudo, essa alternativa nunca foi utilizada no contexto de amostragem.

Esse método foi incluído em uma bateria de experimentos computacionais de métodos de amostragem em Ros e Guillaume (2015), Ros e Guillaume (2016a), Ros e Guillaume (2016b) e apresentou bons resultados. Por esse motivo, foi incluído nessa seção. Todavia, não foi encontrado nenhum trabalho que utilize, na prática, o processo de inicialização do k-médias++ como método amostral.

3.1.3 Amostragem baseada no *Furthest-first-traversal*

Proposto como um método heurístico para gerar soluções para o *Traveller Salesman Problem* em Rosenkrantz, Stearns e Lewis II (1977), a heurística *Furthest-first-traversal* (FFT) consiste em selecionar o próximo elemento que está mais longe dos previamente selecionados para formar o caminho a ser percorrido.

No contexto de algoritmo de amostragem o FFT é muito similar ao método de amostragem, apresentado previamente, baseado na inicialização do k-médias++. Na realidade, a única diferença entre os dois é que o primeiro seleciona o elemento mais distante dos previamente escolhidos de modo determinístico, enquanto o segundo realiza um sorteio com a chance de seleção ponderada pela distância, como pode ser visto no algoritmo 3.

O FFT possui a mesma complexidade computacional que o procedimento amostral utilizando a inicialização do k-médias++; afinal, ambos métodos executam uma quantidade de varreduras no conjunto de dados proporcional ao número de elementos amostrados (representado pelo parâmetro k). Além disso, geram as amostras com distribuições semelhantes.

De mesmo modo que o algoritmo amostral baseado no k-médias++, não foi encontrado nenhum trabalho na literatura que utilize o FFT como algoritmo amostral. No entanto, devido a similaridade com o algoritmo apresentado na seção anterior e, como também apresentou bons resultados nos experimentos computacionais – em Ros e Guil-

Entrada: $X = \{x_i\}, (i = 1, \ldots, n), k, (1 \leq k \leq n)$
Saida: $Y = \{y_i \in X\}, (i = 1, \ldots, k)$

$Y \leftarrow \{x_{rand(1,n)}\}$;
para $i \leftarrow$ *1 até* $k - 1$ **faça**
  $max \leftarrow 0$;
  para $j \leftarrow$ *1 até* n **faça**
    $d \leftarrow \min(dist(x_j, Y))$;
    se $d \geq max$ **então**
      $max \leftarrow d$;
      $x_{max} \leftarrow x_j$;
  $Y \leftarrow Y \cup \{x_{max}\}$;
retorne Y

Algoritmo 3: Algoritmo amostral FFT.

laume (2015), Ros e Guillaume (2016a), Ros e Guillaume (2016b) – o FFT foi incluído no rol de métodos de amostragem tendenciosa.

3.1.4 Amostragem por Agregação do *Bootstrap*

Na área de aprendizado de máquina a agregação do *boostrap* (*Bootstrap AGGregatING*), ou *Bagging*, é um meta-algoritmo com objetivo para melhorar a estabilidade e acurácia de métodos de aprendizado de máquina (BREIMAN, 1996). Além disso, a abordagem diminui a variância e ajuda reduzir super ajustes do modelo ao conjunto de dados. O processo é feito através de várias execuções de um método variando a entrada de dados. O resultado final é obtido através da ponderação dos resultados de cada execução individual.

Em Dolnicar e Leisch (2004), os autores propõem um método de análise de agrupamento que utiliza a amostragem como uma etapa intermediária, chamado *Bagged Sampling*. O método de agrupamento apresentado consiste em retirar n amostras de modo aleatório uniforme com reposição. Isto é, cada elemento tem chance de aparecer mais de uma vez na mesma amostra. Após a etapa de construção de n amostras, um algoritmo de agrupamento base é executado em cada uma das amostras.

Os autores indicam que esse algoritmo base pode ser o k-médias, hierárquico, ou qualquer outro algoritmo que tenha como parâmetro a quantidade de grupos que serão identificados. É importante ressaltar que em virtude do tamanho reduzido das amostras em relação à cardinalidade do conjunto de dados completo, não há problema na adoção de

um algoritmo de agrupamento base que possua uma escalabilidade ruim. Após a execução do algoritmo de agrupamento em todas as n amostras e identificando k grupos em cada uma, obtêm-se nk médias (centroides). Por fim, o terceiro passo do algoritmo é considerar as nk médias como um novo conjunto de dados. Esse novo conjunto será a amostra.

No trabalho original, os autores executam um algoritmo de análise de agrupamento hierárquico na amostra para identificar as m partições desejadas. O último passo do algoritmo de agrupamento consiste em associar os elementos do conjunto de dados original às $1 \leq m \leq nk$ partições obtidas pelo algoritmo hierárquico. Essa associação é feita atribuindo cada elemento à média mais próxima, de uma das m partições.

Como o intuito da etapa experimental (Capítulo 6) é avaliar se a amostra produzida por esse método amostral é boa para o k-médias, o algoritmo executado sobre a amostra final foi o k-médias – não o hierárquico como no trabalho original.

É importante ressaltar que se utilizado o k-médias como algoritmo de agrupamento base, os nk centroides não são elementos pertencentes ao conjunto de dados. Portanto, esse método não é tecnicamente um algoritmo amostral. Em uma classificação mais técnica, esse método se enquadraria como um método de sumarização dos dados. Contudo, há uma forma natural de gerar uma amostra através da substituição do algoritmo base. Substituindo o k-médias por um outro algoritmo – como o PAM (KAUFMAN; ROUSSEEUW, 1990) – que retorne um elemento existente como o centro do grupo (medoide), o resultado final será um subconjunto do conjunto de dados original.

Se cada uma das amostras iniciais for dividida em k partições por um método de análise de agrupamento como o PAM (KAUFMAN; ROUSSEEUW, 1990), a comparação com outros métodos seria mais justa, pois todos retornam uma amostra do conjunto de dados original. Por esse motivo, nos experimentos computacionais, descritos no capítulo 6, o algoritmo de análise de agrupamento base utilizado foi o PAM.

Nos resultados experimentais apresentados em Ros e Guillaume (2015), Ros e Guillaume (2016a), Ros e Guillaume (2016b) o *Bagged Sampling* apresentou ótimos resultados. Todavia, nesse experimento não ficou documentado de que forma o algoritmo foi utilizado para gerar as amostras. Portanto, reproduzir o mesmo procedimento para comparar os resultados é difícil.

Os parâmetros utilizados pelo método são: a quantidade de amostras n, a quanti-

dade de elementos em cada amostra b, e a quantidade de partições que serão identificadas em cada amostra k. Além disso deve-se definir o algoritmo de agrupamento base, além de todos os parâmetros necessário para o mesmo. Caso o algoritmo de análise de agrupamento base possua o parâmetro k, como o k-médias, somente os outros dois parâmetros precisam ser especificados.

Essa abordagem possui algumas vantagens em relação às outras abordagens apresentadas. A primeira delas é que o método é pouco influenciado por ruído, pois, a amostra final é composta por várias médias; e a influência de padrões ruidosos é, normalmente, pequena nesses elementos. Além disso, a execução também é rápida, afinal, não há a necessidade de executar diversas varreduras no conjunto de dados completo.

O terceiro aspecto positivo do método é que não há a necessidade de conhecer precisamente quantas partições existem no conjunto de dados para a construção da amostra. Qualquer valor de k, desde que seja maior ou igual a quantidade de grupos existentes no conjunto de dados, deve ser suficiente. Para ilustrar, em Dolnicar e Leisch (2004) os autores utilizaram $n = 50$ e $k = 20$, gerando amostras com 1000 (nk) elementos para identificação de poucas partições na solução final. Os parâmetros k e n são responsáveis por regular o tamanho da amostra e, valores altos acarretará em uma amostra com uma quantidade grande de elementos e, consequentemente demandará um maior esforço computacional dos algoritmos de agrupamento.

3.2 Métodos de amostragem tendenciosa utilizando critérios de densidade

A segunda categoria engloba os métodos que consideram a densidade da região em que o elemento está situado para determinar a chance do elemento ser incluído na amostra.

Um grande desafio enfrentado por essa abordagem é estimar a densidade a um custo computacional reduzido, dado que o cálculo da densidade de modo ingênuo é custoso. Os métodos selecionados – DBS e VGDBS (Seções 3.2.1 e 3.2.2) – como representantes dessa categoria tentam contornar esse problema, porém, sofrem com uma perda de precisão na estimativa da densidade.

3.2.1 Amostragem tendenciosa pela densidade

Em Palmer e Faloutsos (2000), os autores propõem uma abordagem amostral inovadora no contexto de análise de agrupamento. De fato, esse é o primeiro trabalho da literatura que aborda o assunto de amostragem tendenciosa nesse contexto. Os autores mostram as desvantagens da amostragem aleatória uniforme, já citadas na introdução, e propõem um novo método amostral para contornar os problemas identificados. Tal método é denominado de amostragem tendenciosa pela densidade, do inglês, *Density Biased Sampling* (DBS).

O DBS é similar a amostragem proporcional ao tamanho. Na amostragem proporcional os elementos de cada estrato são escolhidos de acordo com a cardinalidade do mesmo em relação à quantidade de elementos do conjunto de dados. Por exemplo, se desejamos retirar uma amostra de um conjunto de dados que possui três partições, cada uma representando 10%, 20% e 70% do total de elementos, a amostra será composta por um elemento do primeiro estrato, dois elementos do segundo e sete do último; ou seja, proporcional ao tamanho de cada estrato.

Contudo, se tratando de um problema de análise de agrupamento os elementos não são rotulados, portanto, a amostragem proporcional ao tamanho deve ser executada em dois estágios. O primeiro responsável pela partição do conjunto de dados e o segundo para amostrar proporcionalmente de acordo com a cardinalidade de cada partição.

A amostragem proporcional ao tamanho de cada estrato é uma estratégia contraproducente quando o objetivo é aumentar a representatividade na amostra das partições com poucos elementos. Por esse motivo, o DBS inverte a proporcionalidade, ou seja, é inversamente proporcional à cardinalidade. Isto é, elementos em regiões mais densas terão uma probabilidade menor de serem selecionados para compor a amostra. Outra característica que diferencia o DBS da amostragem proporcional é o fato do DBS ser uma técnica de um estágio apenas; assim, não há a necessidade de particionar o conjunto de dados em um estágio anterior. O DBS utiliza o conceito de densidade de elementos em uma dada região e considera cada uma dessas como uma partição em razão da ausência de prévio conhecimento dos rótulos.

A ideia do DBS, descrito em Palmer e Faloutsos (2000), basicamente consiste em colocar um reticulado sobre o conjunto de dados e contar quantos elementos há em cada

Entrada: $x = \{v_1, \ldots, v_d\}, c$
Saida: índice da hash

1 $h \leftarrow 0$;
2 **para** $i \leftarrow$ *1 até* d **faça**
3 $\quad h \leftarrow 65599h + \lfloor v_i/c \rfloor$;
4 **retorne** $h \bmod H$; `/* H é o tamanho da hash */`

Algoritmo 4: Função de *Hash* do DBS (PALMER; FALOUTSOS, 2000).

Entrada: $X = \{x_i\}, (i = 1, \ldots, n), m, c$
Saida: $Y = \{y_j \in X\}, (j = 1, \ldots, \approx m)$

1 $n \leftarrow \{0, 0, \ldots, 0\}$; `/* Tamanho H */`
2 **para todo** $x_i \in X$ **faça**
3 $\quad n_{\text{hash}(x_i,c)} \leftarrow n_{\text{hash}(x_i,c)} + 1$;
4 Calcule α; `/* Ver definição no texto */`
5 **para todo** $x_i \in X$ **faça**
6 $\quad p \leftarrow \alpha / n^e_{hash(x_i,c)}$;
7 $\quad$ **se** *rand(0,1)* $< p$ **então**
8 $\quad\quad Y \leftarrow Y \cup \{n^e_{hash(x_i,c)}/\alpha, x_i\}$;
9 **retorne** Y

Algoritmo 5: Algoritmo DBS (PALMER; FALOUTSOS, 2000).

célula do reticulado. A partir disso, a chance de cada elemento ser selecionado é ajustada de acordo com a densidade da célula em que está inserido. É importante notar que o número de células cresce exponencialmente em relação ao número de dimensões. Isto é, em um reticulado com n divisões em cada um dos k eixos produz n^k células.

Armazenar e iterar sobre um número exponencial de células é problemático, o DBS mapeia cada célula para uma tabela *hash* (Algoritmo 4), e esta é usada para calcular a densidade e fazer a amostragem. O uso da tabela *hash* elimina o problema causado pelo número exponencial de células por completo. O primeiro passo para posicionar os elementos no reticulado é determinar qual é o tamanho de cada célula. Por exemplo, se o tamanho da célula for definido como 10 a coordenada de cada elemento x será: $(\lfloor v_1/10 \rfloor, \lfloor v_2/10 \rfloor, \ldots, \lfloor v_d/10 \rfloor)$. O processo de quantização dos valores ocorre no algoritmo 4 (linha 3).

A função do algoritmo 4 é receber um elemento X (d-dimensional) e o tamanho da célula c como entrada. O próximo passo é iterar sobre cada dimensão d de x (linhas 2-3) com o objetivo de mapear o elemento para uma tabela *hash* de tamanho H (linha 4).

Note que elementos que pertencerem a uma mesma célula serão mapeados para a mesma posição na tabela *hash*. Desse modo, é possível mapear um número exponencial de células – em relação à dimensionalidade – para uma tabela *hash* de H posições. Contudo, células diferentes podem ser mapeadas a mesma posição (colisão).

O problema de colisão na tabela *hash* acarreta em imprecisões no cálculo de densidade; ou seja, um elemento de uma região pouco densa pode ser mapeado para a mesma posição na tabela *hash* ocupada por uma grande quantidade de elementos. Nesse cenário, a densidade do elemento seria estimada incorretamente e, sua chance de ser selecionado para compor a amostra seria pequena.

O algoritmo 5 recebe o conjunto de dados X, o tamanho amostral m, e o tamanho da célula c. A primeira etapa do algoritmo (linhas 1-3) percorre todo o conjunto de dados calculando a posição na *hash* para cada elemento – utilizando o algoritmo 4 – e incrementando um contador em uma lista (n) na posição correspondente. Ao final do laço obtêm-se a contagem dos elementos em cada posição da *hash*. O procedimento seguinte é calcular α para que que o tamanho amostral esperado seja aproximadamente m.

O cálculo de α é determinado pela equação 3.1, onde H representa o número de posições na *hash* e o parâmetro e regula a força da amostragem tendenciosa. Caso $e = 0$ a amostragem se torna idêntica à amostragem aleatória uniforme, no entanto, caso $e = 1$ o número de elementos amostrados em cada posição da *hash* será (aproximadamente) constante, ou seja, criando uma tendência no processo de amostragem para regiões com poucos elementos. Para efeitos práticos, considera-se $e = 1$ sempre que o valor do mesmo for omitido.

$$\alpha = \frac{m}{\sum_{i=1}^{H} n_i^{1-e}} \tag{3.1}$$

O passo seguinte é realizar uma nova varredura em todo os elementos do conjuntos de dados sorteando, para cada elemento, se ele será incluído na amostra (linhas 5-8). Ao final do segundo laço, uma amostra (Y) de tamanho aproximadamente m é retornada.

A probabilidade p do elemento ser inserido na amostra é dado pela expressão na linha 7 do algoritmo 5, onde $n_{hash(x_i,c)}$ é o número de elementos contidos na mesma posição que o elemento x_i na tabela *hash*. Pela fórmula, fica evidente que quanto maior a quantidade de elementos menor a probabilidade de seleção – considerando $e = 1$.

Por exemplo, suponha $\alpha = 1, e = 1$, e duas posições na tabela *hash*, uma com 10 elementos e outra com 100 elementos. A chance de sorteio de cada elemento na primeira célula será de 1/10 e esse sorteio será realizado dez vezes (uma para cada elemento). Na média apenas um elemento será sorteado. Na posição com 100 elementos, a chance de sorteio de cada elemento será de 1/100 e, do mesmo modo, após cem sorteios – na média – apenas um elemento será escolhido para compor a amostra. Se ignorarmos os problemas de colisões, o DBS garante que regiões esparsas possuam, proporcionalmente, uma quantidade maior de elementos que as regiões mais densas.

A grande vantagem do DBS é seu baixo custo computacional e possuir uma excelente escalabilidade independente do número de dimensões. Em relação aos parâmetros, o algoritmo necessita conhecer apenas o tamanho da amostra e o tamanho da tabela *hash*. Contudo, posteriormente diversos trabalhos (NANOPOULOS; MANOLOPOULOS; THEODORIDIS, 2002; NANOPOULOS; THEODORIDIS; MANOLOPOULOS, 2006; KERDPRASOP; KERDPRASOP; SATTAYATHAM, 2005; APPEL, 2010) relataram que o DBS se deteriora rapidamente na presença de dados ruidosos.

Além disso, o aumento de dimensionalidade do conjunto de dados acarreta em um número maior de colisões na tabela *hash*. Esse último resultado é esperado, afinal o aumento do número de regiões deveria aumentar a probabilidade de colisões quando mapeados para uma lista de tamanho fixo. Portanto, uma alternativa seria aumentar o tamanho da tabela *hash* para reduzir a ocorrência de colisões.

O problema de selecionar elementos isolados (ruído) também ocorre com esse método, pois esses tendem a se situar em regiões com baixa densidade de elementos. O método DBS não possui nenhum mecanismo para impedir a seleção de células de cardinalidade muito baixa; contudo, essa modificação poderia ser implementada facilmente.

O DBS possui uma importância histórica pra área de amostragem tendenciosa no contexto de análise de agrupamento, afinal, foi o primeiro trabalho publicado a respeito do assunto. Além disso, possui excelente escalabilidade, e mesmo com os problemas apontados por outros trabalhos, na prática, produz bons resultados. Pelo exposto, a inclusão do DBS no conjunto de métodos a serem comparados é de grande valia.

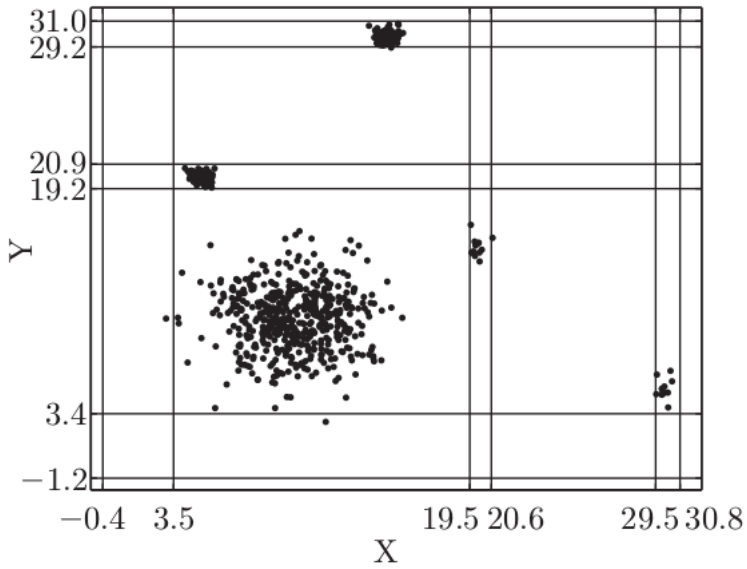


Figura 6 – Exemplo de reticulado variado. Fonte: Qian et al. (2014).

3.2.2 Amostragem tendenciosa pela densidade com reticulado variado

Em Qian et al. (2014), um novo algoritmo amostral chamado de amostragem tendenciosa pela densidade com reticulado variado, do inglês *Variable Grid Density-Biased Sampling* (VGDBS), foi proposto usando o conceito de reticulado variado para se medir a densidade dos elementos. Por reticulado variado entende-se que o tamanho das células não é fixo, o reticulado é composto tanto por células grandes, cobrindo uma ampla região do espaço, quanto por células pequenas.

A motivação dessa técnica é diminuir a quantidade de células existentes no reticulado, evitando assim, um número exponencial de células que alguns métodos que utilizam um reticulado uniforme enfrentam, como no RDBS (KERDPRASOP; KERDPRASOP; SATTAYATHAM, 2005) e BBS (APPEL et al., 2007a; APPEL et al., 2007b; APPEL, 2010). Por esse motivo, esses dois métodos amostrais foram excluídos desse trabalho.

Como pode ser visto na figura 6, o reticulado de tamanho variado é capaz de cobrir grandes regiões vazias com uma única célula. Essa técnica consegue diminuir bastante o número de células necessárias para cobrir o conjunto de dados, especialmente quando a dimensionalidade é alta. Nesse caso, a existência de grandes regiões vazias são comuns.

Para construir o reticulado o algoritmo analisa cada dimensão separadamente. Para cada dimensão o seguinte procedimento é realizado: inicialmente, o algoritmo divide o eixo correspondente a dimensão em intervalos contendo a mesma quantidade de ele-

mentos; após essa divisão, os intervalos adjacentes que possuírem densidade similares são unidos. Quando o procedimento descrito for realizado para todas as dimensões o reticulado variado está construído. Portanto, a complexidade computacional cresce linearmente com a dimensionalidade. Se prestarmos atenção novamente à figura 6, se um dado elemento está no terceiro intervalo em ambas dimensões, ele pertencerá à célula de índice $(3, 3)$, que na figura corresponde a partição de maior área.

O algoritmo utiliza uma variável, chamada de t, para controlar esse procedimento de união das k divisões de cada eixos. O parâmetro t varia no intervalo $[0, 1]$, caso seja próximo de 0 a maioria dos intervalos adjacentes serão combinados, e caso possua o valor próximo de 1, praticamente nenhum dos intervalos serão unidos, acarretando em um grande número de células no reticulado final. Os autores fornecerem uma fórmula para que esse valor seja ajustado dinamicamente ao longo da execução do algoritmo.

Uma vez construído o reticulado, o processo de amostragem é simples. Cada elemento possui a chance de ser selecionado inversamente proporcional à densidade da célula. Do mesmo modo que outros métodos que utilizam reticulados para estimar a densidade do conjunto de dados, os elementos de uma mesma célula possuem a mesma densidade.

A abordagem do VGDBS é inovadora pois elimina por completo o problema de colisões da tabela *hash* enfrentado no DBS (PALMER; FALOUTSOS, 2000) ao mesmo tempo que suprime uma grande quantidade de células vazias devido ao uso do reticulado variado. Além disso, a complexidade computacional e qualidade dos resultados não se degradam rapidamente com uma elevada dimensionalidade do conjunto de dados.

O ponto negativo dessa abordagem é uma imprecisão maior na estimativa da densidade dos elementos. Como ilustrado na figura 6, a estimativa de densidade pode ser muito ruim para alguns grupos. Em especial, para aqueles que possuem formatos mais alongados em uma dimensão. Nos experimentos realizados no trabalho original, os autores não identificaram esse problema. Todavia, as células alongadas apresentadas na figura parecerão muito menos densas do que são na realidade e, portanto, terão uma chance maior de serem selecionadas do que deveriam.

Além do tamanho amostral, o algoritmo possui dois parâmetros e os autores dão sugestões de como atribuir valores para os mesmos. O primeiro parâmetro é o k e determina em quantos segmentos cada eixo será dividido na etapa inicial do algoritmo. Os

autores utilizam o valor $k = \lceil \sqrt[d]{n} \rceil$ onde n representa o número de elementos do conjunto de dados e d a dimensionalidade. O segundo parâmetro do algoritmo (e), controla quão forte é a tendência no sorteio dos elementos menos densos. Apesar do valor ótimo de e variar de um conjunto de dados para outro, o valor de 0.5, como regra geral, apresentou bons resultados.

Os experimentos computacionais no trabalho original compararam o VGDBS com um método que utiliza um reticulado uniforme, chamado *Grid Density-Biased Sampling* (GDBS) (HUANG et al., 2013), e com a amostragem aleatória uniforme. O VGDBS apresentou os melhores resultados identificando o número correto de grupos mesmo na presença de dados ruidosos.

A resistência a padrões ruidosos do VGDBS também é um ponto positivo do algoritmo. Mesmo não resolvendo completamente o problema, é bastante atenuado em relação a outros algoritmos amostrais. O processo de dividir os eixos usando intervalos iguais de elementos impossibilita que um padrão ruidoso fique isolado em uma célula e, consequentemente, com uma alta probabilidade de seleção.

A densidade das células contendo padrões ruidosos tendem a ser subestimadas e dificilmente serão unidas às células vizinhas, devido a diferença de densidade, o que resulta em uma tendência de seleção dos elementos que estejam na mesma célula que os dados ruidosos. Afinal, quanto maior for a célula menor será a densidade. Contudo, como todos os elementos de uma dada célula possuem a mesma densidade, a seleção do elemento ruidoso é improvável.

3.3 Métodos de amostragem tendenciosa utilizando critérios híbridos

A última categoria representa os métodos híbridos; isto é, que consideram tanto critérios de distância quanto densidade para determinar qual elemento será incluído na amostra.

Os dois métodos escolhidos como representantes dessa categoria executam duas etapas sequenciais. A primeira etapa, executa rapidamente utilizando critérios de distância e seleciona uma amostra cobrindo uma grande região do espaço (com distribuição

aproximadamente homogênea); a segunda etapa tem a função de eliminar os dados ruidosos utilizando critérios de densidade.

A vantagem dessa abordagem é que estimar a densidade de um conjunto reduzido de elementos, filtrados pela primeira etapa, acelera bastante o algoritmo em relação a calcular a densidade de todos elementos.

3.3.1 DENDIS

Em Ros e Guillaume (2015), Ros e Guillaume (2016a), os autores apresentam um método de amostragem chamado DENDIS, cujo nome é uma contração de Densidade e Distancia (*Density and Distance*). A inovação apresentada no trabalho corresponde ao fato de o algoritmo utilizar um processo de dois estágios na construção da amostra. O método é uma variante do DIDES (ROS; GUILLAUME, 2016b), dos mesmo autores, e se diferencia apenas pelo critério de parada do algoritmo. Esse último interrompe o laço principal utilizando um critério de distância, enquanto o DENDIS utiliza um critério de densidade.

O DENDIS é um algoritmo iterativo que adiciona elementos na amostra a cada iteração considerando dois objetivos. O primeiro critério, assegura que áreas densas do conjunto de dados X estejam representadas na amostra Y preocupando-se em manter o tamanho da amostra pequena para evitar um grande número de representantes nas regiões densas. Já o segundo critério, procura cobrir homogeneamente o espaço e garantir, de certo modo, uma cobertura dos formatos das partições existentes no conjunto de dados. Para atender o objetivo de densidade, o novo representante (elemento da amostra) é escolhido no conjunto mais populoso de padrões associados. Já para propósitos de distância, cada novo representante é escolhido como o mais distante dos existentes.

A sobre representação é evitada pelo controle dinâmico dos dois parâmetros que definem a densidade: volume e cardinalidade. O segundo é definido pelo único parâmetro do algoritmo, a granularidade (gr) e a cardinalidade do conjunto de dados (n). O produto $n \cdot gr$ define um limite (wt) que representa o número mínimo de padrões associados a um representante. O volume é estimado pela distância máxima entre o representante e algum padrão associado a ele.

Assim como em Ros e Guillaume (2016a), a descrição do DENDIS foi dividida em

duas etapas. A primeira etapa é responsável por escolher representantes considerando a densidade e também noções de distância. Já a segunda etapa pode ser vista como uma etapa de pós-processamento cujo objetivo é evitar a seleção de padrões ruidosos como representantes.

A primeira amostra é escolhida de modo aleatório (linhas 1 e 2). Então, o algoritmo itera para selecionar os representantes (linhas 4-23). No primeiro laço interno (linhas 5-7), cada elemento não selecionado – que está no conjunto de dados mas não na amostra $(X \setminus Y)$ – é associado ao representante mais próximo, formando conjuntos $X_{y_{\min}}$ de todos os elementos representados pelo representante $y_{\min}$ (linha 7).

No segundo laço interno (linhas 8-10), para cada um dos conjuntos X_{y_k} – processados no laço anterior – o elemento mais distante associado ao representante y_k é armazenado ($x_{\max}(y_k)$), localizado a uma distância $d_{\max}(y_k)$. Os dois primeiros laços são descritos pelos autores como uma etapa de preparação, afinal, o novo representante será escolhido na segunda parte do laço principal.

Os representantes já selecionados são ordenados em ordem decrescente de número de elementos associados (linha 12). Esse ordenamento serve para determinar quando interromper o laço que se inicia a seguir (linha 13). Este laço executa para determinar qual elemento deve ser adicionado na amostra e, como se deseja elementos em regiões densas, caso o número de elementos associados ao elemento y_k for inferior ao limiar wt (linha 14) o laço pode ser interrompido – pois devido a ordenação realizada, todos os elementos subsequentes também terão menos padrões associados.

Pode-se ver como a granularidade impacta diretamente no tamanho da amostra Y. Quanto menor o valor para a granularidade, maior será o tamanho da amostra. Afinal, o valor associado a wt (linha 3) será pequeno e, consequentemente, o critério que determina se o elemento é apto para ser inserido na amostra torna-se mais relaxado (linhas 14 e 17).

Se não houvesse nenhuma restrição adicional, os representantes seriam selecionados apenas nas regiões mais densas do conjunto de dados, sobre representando essas regiões. Portanto, a segunda condição é ligada à densidade (linha 17) que assegura um espaçamento mínimo entre os elementos. O elemento escolhido será o elemento (denso) mais distante dos padrões já selecionados. O elemento que passar nas duas condições é marcado para inserção (linhas 18-19); o laço é interrompido (linha 20); e o elemento é adicionado como

Entrada: $X = \{x_i\}, (i = 1, \ldots, n), g_r$
Saida: $\{Y, Y_{y_j}, (j = 1, \ldots, |Y|), y_j \in X, Y_{y_j} \subset X\}$

$x_{\text{init}} \leftarrow \{x_{rand(1,n)}\}$;
$Y \leftarrow \{y_1 = x_{\text{init}}\}$;
$Adicionar \leftarrow verdadeiro, k \leftarrow 0.2, wt \leftarrow n \cdot gr$;
enquanto *Adicionar* **faça**
  para todo $x_l \in X \setminus Y$ **faça**
    $y_{\min} \leftarrow \arg\min_{y_k \in Y} d(x_l, y_k)$;
    $Y_{y_{\min}} \leftarrow Y_{y_{\min}} \cup \{x_l\}$; `/* Elementos representados por` $y_{\min}$ `*/`
  para todo $y_k \in Y$ **faça**
    $d_{\max}(y_k) \leftarrow \max_{x_m \in X_{y_k}} d(x_m, y_k)$;
    $x_{\max}(y_k) \leftarrow \arg\max_{x_m \in X_{y_k}} d(x_m, y_k)$;
  $Adicionar \leftarrow falso$;
  ORDENE$(y_1, \ldots, y_{|Y|})$ com $|Y_{y_1}| \geq \cdots \geq |Y_{y_{|Y|}}|$;
  para todo $y_k \in Y$ **faça**
    se $|Y_{y_k}| < wt$ **então**
      interrompa ;
    $\alpha_k \leftarrow \max(\frac{wt}{|Y_{y_k}|}, k)$;
    se $d_{\max}(y_k) \geq \alpha_k \overline{d_{\max}(y_k)}$ **então**
      $x\star \leftarrow x_{\max}(y_k)$;
      $Adicionar \leftarrow verdadeiro$;
      interrompa ;
  se *Adicionar* **então**
    $Y \leftarrow Y \cup \{x\star\}$;
`/* Pós-processamento */`
para todo $y_i \in Y$ **faça**
  se $|Y_{y_i}| \leq wt$ **então**
    $Y \leftarrow Y \setminus \{y_i\}$;
  se $d_{\max}(y_i) \geq \overline{d_{\max_{y_i \in Y}}(y_i)}$ **então**
    $y_i \leftarrow \arg\min_{y_j \in Y_{y_i}} d(y_j, c)$; `/*` c `é o centroide de` Y_{y_i} `*/`
  $Y_{y_i} \leftarrow \{y_i\}$;
para todo $x_i \in X \setminus Y$ **faça**
  $d_{\min}(x_l) \leftarrow \arg\min_{y_k \in Y} d(x_l, y_k)$;
  $Y_{y_k} \leftarrow Y_{y_k} \cup \{x_l\}$;
retorne $\{Y, Y_{y_j}, (j = 1, \ldots, |Y|), y_j \in X, Y_{y_j} \subset X\}$

Algoritmo 6: DENDIS (ROS; GUILLAUME, 2016a).

representante (linhas 21-22).

Após a inserção de todos os representantes a etapa de pós-processamento ocorre para remover os elementos em regiões de baixa densidade (ruído). O primeiro critério ocorre quando um representante y_i possui a densidade abaixo do limiar wt. É importante ressaltar que a quantidade de elementos associados a cada representante tende a decrescer a cada iteração do laço principal no algoritmo 6. Isso ocorre devido ao aumento no número de representantes a cada iteração devido à distribuição das associações para os recém adicionados na amostra. Tal elemento com poucos elementos associados – localizado em uma região de baixa densidade – é removido da amostra (linhas 24-25).

O segundo critério para detecção de ruído é pela distância (linha 26). Na fase de pós-processamento os representantes estão espalhados de forma aproximadamente homogênea e a distância média $\overline{d_{\max_{y_k \in Y}(y_k)}}$ pode ser utilizada como um limiar de descarte. Nesse caso, o representante é substituído pelo elemento y_j mais próximo do centroide (linha 27). Por fim, as associações dos elementos aos representantes são recalculadas (linhas 29-31). Ao final da execução do algoritmo é retornado tanto o conjunto de representantes Y quanto $|Y|$ conjuntos representando os elementos associados a cada representante.

Os experimentos computacionais mostrando a eficiência do DENDIS (ROS; GUILLAUME, 2015; ROS; GUILLAUME, 2016a) foram realizados usando os algoritmos de agrupamento k-médias e hierárquico, contudo, a comparação entre os demais algoritmos amostrais foi realizada apenas com o k-médias.

Nos experimentos, o DENDIS obteve o segundo menor tempo de execução, ficando atrás somente da amostragem aleatória uniforme. Além disso, obteve a melhor média de aproximação dos resultados obtidos pelo k-médias no conjunto de dados completo quando comparado aos demais métodos. Também conseguiu o melhor resultado em metade dos conjuntos de dados testados. A métrica de qualidade utilizada foi o *Rand Index* (RI) (RAND, 1971), bastante difundida na área de análise de agrupamento, para medir a diferença do resultado obtido pelo algoritmo de agrupamento no conjunto de dados completo em relação ao mesmo executado sobre a amostra.

Apesar de ser uma métrica bastante difundida, o RI é equivalente a métrica de acurácia no contexto de classificação e apresenta os mesmos problemas em conjuntos de dados desbalanceados. Nesses cenários, é esperado um valor alto para a métrica mesmo

se os métodos de análise de agrupamento atribuírem todos os elementos a apenas uma partição. Portanto um valor alto para a métrica não é tão informativo, afinal, não se sabe se o método teve um bom desempenho ou se existe um desbalanceamento na distribuição dos elementos no conjunto de dados.

A métrica utilizada nos experimentos computacionais realizados neste trabalho (Capítulo 6) foi o *Adjusted Rand-Index* (ARI) (HUBERT; ARABIE, 1985). O ARI retorna o quão melhor o resultado obtido é em relação a uma rotulação aleatória e elimina o problema existente no RI. Devido a essa diferença nas métricas os resultados apresentados em Ros e Guillaume (2016a) e no capítulo 6 não são diretamente comparáveis.

O único problema do DENDIS é a escolha do valor para o parâmetro de granularidade g_r, que apesar de ser de mais fácil estimativa do que, por exemplo, o número de grupos de um dado conjunto de dados, pode fazer com que o algoritmo apresente resultados de baixa qualidade em caso de uma estimativa equivocada desse parâmetro.

3.3.2 *Rough-DBSCAN*

Em Viswanath e Babu (2009), os autores propõem uma variação do DBSCAN, chamada *Rough-DBSCAN*, que utiliza um processo de amostragem tendenciosa para minimizar o tempo de execução. Conceitualmente a proposta é muito parecida com o DENDIS e ocorre em duas etapas.

A primeira etapa do algoritmo utiliza um algoritmo de amostragem de menor complexidade computacional, baseado no algoritmo Líder (Seção 3.1.1), com a finalidade de gerar uma amostra pequena e de distribuição homogênea. Por reduzir a quantidade de elementos a serem analisados na etapa subsequente, esse procedimento também reduz o tempo de execução do algoritmo.

A segunda etapa, da mesma forma que o DENDIS, descarta os elementos localizados em regiões de baixa densidade; afinal, existe uma alta probabilidade da primeira etapa do algoritmo selecionar tais elementos. Como o método foi desenvolvido especificamente para o DBSCAN, a segunda etapa do algoritmo utiliza os mesmo parâmetros do algoritmo de análise de agrupamento para determinar se cada elemento possui a densidade miníma exigida, descartando os que estiverem abaixo desse limite. Por esse motivo, o algoritmo também não é afetado pela existência de padrões ruidosos no conjunto de dados.

Estimar a densidade de um elemento de modo ingênuo é uma tarefa que exige um esforço computacional grande pois, requer a varredura completa do conjunto de dados verificando quantos elementos se encontram a uma distância menor que o ϵ (parâmetro do DBSCAN). Assim, os autores propõem uma alternativa de menor custo computacional para aproximar a densidade real dos elementos.

A proposta para estimar a densidade de maneira aproximada apresentada pelos autores é chamada de cardinalidade grosseira, *rough-cardinality*, e aproveita o resultado do algoritmo Líder, executado na primeira etapa. Para estimar a densidade de cada líder, o algoritmo considera que a quantidade de elementos a uma distância menor que ϵ é igual ao somatório de seguidores de todos os líderes cuja distância seja menor que ϵ. Dessa forma, não é necessário percorrer o conjunto de dados completo, apenas os elementos filtrados pela primeira etapa. Os autores mostram que existem dois tipos de erros nessa estimativa de densidade. Ainda, afirmam que, em regiões onde a distribuição dos elementos é uniforme, os erros tendem a se cancelar.

Estimar a densidade baseando-se apenas na saída do algoritmo Líder é uma tarefa difícil. A proposta apresentada em Viswanath e Babu (2009) é uma boa solução, porém, é possível modificar o algoritmo Líder para que a estimativa da densidade fique trivial, eliminando os erros do algoritmo decorrente de uma estimativa de densidade equivocada. Esse fato, motivou a criação dos métodos propostos no capítulo 5.

Após a construção da amostra, o DBSCAN é executado sobre a mesma e o resultado obtido é extrapolado para todo conjunto de dados usando o resultado da primeira etapa do algoritmo. Isto é, a partição que foi atribuída a cada líder é replicada para todos os seus seguidores. Os elementos que permaneceram sem grupo são rotulados como ruído.

Por se tratar de um método amostral específico para o DBSCAN, os autores assumem o conhecimento dos parâmetros do algoritmo de agrupamento no método amostral, caso contrário a execução do algoritmo de agrupamento não seria possível. Além desses parâmetros do algoritmo, o método precisa de um terceiro para usar como distância limite para o algoritmo Líder, isto é, o parâmetro τ. Apesar do conhecimento de τ não ser fornecido para o DBSCAN, existe limites para o valor desse parâmetro, $0 \leq \tau \leq \epsilon$, que auxiliam na escolha do valor para o mesmo. O parâmetro τ regula o tamanho da amostra e, consequentemente, o nível de aproximação em relação ao DBSCAN.

Nos experimentos realizados pelos autores, o método foi comparado contra o DBSCAN no conjunto de dados completo, em relação a tempo de execução e proximidade dos resultados. Os resultados foram medidos utilizando a métrica *Rand-Index*, variando o parâmetro τ e, no geral, apresentaram um alto nível de acordo com os resultados obtidos pelo DBSCAN no conjunto de dados completo a um custo computacional bastante reduzido.

4 Algoritmo genético amostral

Neste capítulo será apresentado o método de amostragem tendenciosa proposto para o k-médias baseado no algoritmo genético. Essa proposta foi previamente publicada em Luchi et al. (2015), Luchi et al. (2016) e foi motivada pelo trabalho de Xiao et al. (2014). A meta-heurística proposta faz uso do algoritmo genético para extrair uma amostra contendo os elementos nas periferias das partições.

O algoritmo k-médias talvez seja o algoritmo de agrupamento mais estudado e conhecido da literatura, porém, poucas vezes foi utilizado como um método amostral. Em Xiao et al. (2014), os autores amostraram os conjuntos de dados com o seguinte procedimento: executaram o k-médias para particionar o conjunto de dados em k grupos e, em seguida, selecionaram os elementos nas bordas das partições. Essa amostragem foi realizada para reduzir a quantidade de elementos necessários para treinar uma máquina de vetor de suporte (CORTES; VAPNIK, 1995). Devido ao classificador buscar a separação máxima entre os grupos, os elementos que se situam nas bordas dos grupos são os mais relevantes e a técnica foi bastante efetiva.

Apesar de ser possível utilizar o k-médias para criar uma amostra para acelerar um algoritmo mais lento, como em Xiao et al. (2014), não se pode usar a mesma abordagem para acelerar o próprio k-médias. Afinal, a abordagem apresentada exige a execução do algoritmo no conjunto de dados completo. O algoritmo genético apresentado nesse capítulo produz um resultado similar ao obtido pelo método de Xiao et al. (2014); contudo, a um custo computacional reduzido, viabilizando sua execução em conjunto com o k-médias.

A principal diferença entre as abordagens é o método proposto não executar o algoritmo k-médias no conjunto de dados completo para particioná-lo. Ao invés disso, trabalha em pequenas amostras do conjunto de dados e executa o k-médias apenas nesses elementos. Com o passar das iterações a amostra sofre alterações para minimizar a função objetivo do problema.

A dificuldade da abordagem está em selecionar os elementos nas periferias das partições sem a necessidade de defini-las através da execução de um algoritmo de análise de agrupamento no conjunto de dados completo. Para isso, o método proposto busca

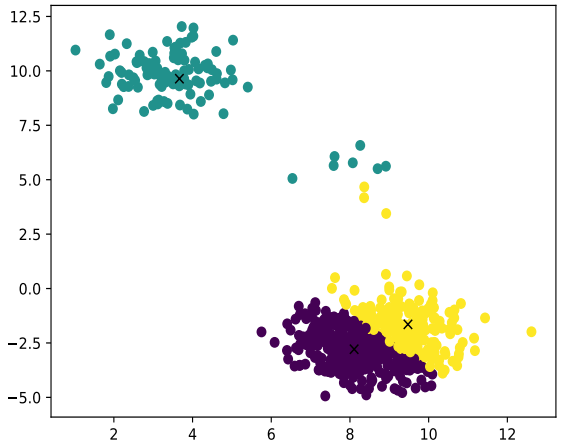
uma amostra do conjunto de dados que, ao minimizar a variância, resulte em um valor elevado para a mesma. Em outras palavras, busca-se uma amostra que a melhor solução possível na amostra ainda seja a ruim em termos da função otimizada pelo k-médias. Isso quer dizer que mesmo com a divisão correta dos grupos a soma dos elementos até suas respectivas médias totaliza um valor elevado e, portanto, tais elementos estão distantes das médias.

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||^2 \tag{4.1}$$

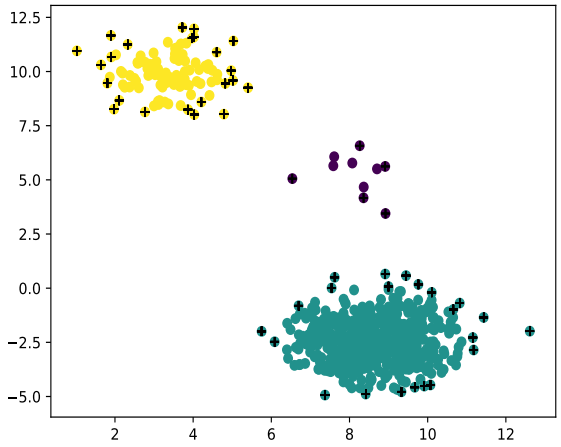
Analisando a função objetivo do problema, pode-se perceber que os elementos que mais contribuem para a função objetivo do k-médias (Equação 4.1) são os elementos x de cada partição S_i que estão distantes das médias μ_i. Caso um elemento esteja na mesma localização da média de sua respectiva partição, ele não contribuirá para a função objetivo, nesse caso $x - \mu_i = 0$. De outro lado, um elemento distante da média, $x \neq \mu_i$, terá uma contribuição positiva no valor obtido do somatório, e quanto maior a distância entre eles, maior será a contribuição. Portanto, uma amostra que possuir uma variância mínima elevada é majoritariamente composta de elementos com grande contribuição para função objetivo, e portanto, distante das médias das partições.

Assumindo-se que estar distante do centro de massa da partição (média) é equivalente a estar próximo da borda – o que é verdade para conjunto de dados com partições esferoidais – a heurística sempre retorna os elementos nas bordas das partições. Em conjuntos de dados que o k-médias particiona corretamente essa premissa é verdadeira. Para os demais casos discutidos na seção 2.1, com apenas uma exceção que será discutida a seguir, o algoritmo genético, assim como o k-médias, não produz o resultado desejado. Em síntese, se o uso do k-médias for indicado o algoritmo genético produz uma boa amostra, caso contrário, não deveria ser utilizado nenhum dos dois métodos para aquele conjunto de dados pois as premissas assumidas por ambos não são verdadeiras.

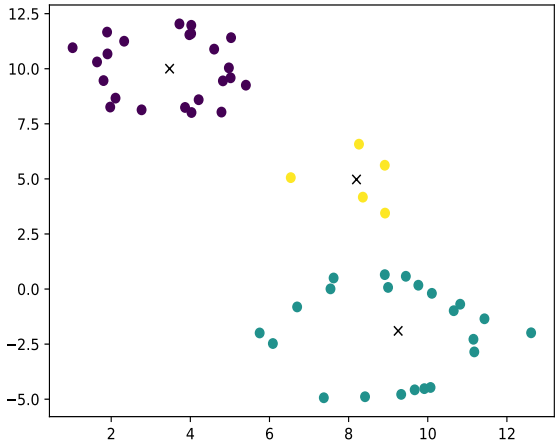
Além de mitigar o problema de escalabilidade do próprio k-médias, evitando sua execução no conjunto de dados completo, o algoritmo genético também tende a aumentar a representatividade de partições de baixa cardinalidade. Portanto, a chance de ocorrer um particionamento incorreto do conjunto de dados devido as partições possuírem cardinalidades muito discrepantes, tal como o conjunto ilustrado pela figura 2c, é reduzida.



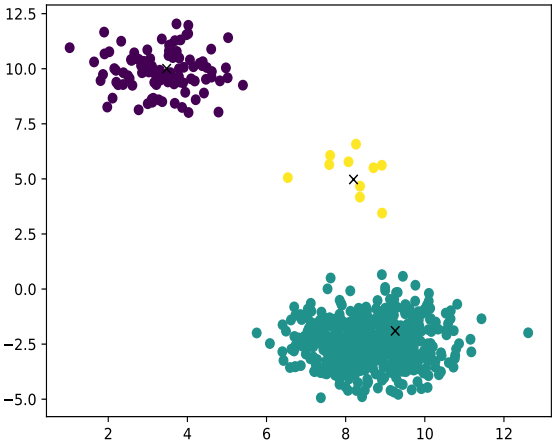
(a) Resultado do k-médias no conjunto de dados completo.



(b) Amostra final retornada pelo algoritmo genético (elementos destacados com cruz).



(c) Resultado da execução do k-médias apenas com os elementos contidos na amostra.



(d) Conjunto de dados reclassificado pela proximidade da média.

Figura 7 – Comparação entre o algoritmo genético e o k-médias em um conjunto de dados com cardinalidades desbalanceadas das partições.

Se a premissa de que elementos distantes da bordas pertencem às periferias das partições for mantida, pode-se observar que o método identifica corretamente os elementos nas bordas. Apesar da partição resultante do método amostral e do k-médias serem diferentes – pois o k-médias tende a ignorar partições com poucos elementos – o resultado obtido através da amostra possui uma qualidade superior.

A figura 7 ilustra o cenário descrito acima. No exemplo, as partições possuem cardinalidades desbalanceadas e o k-médias, apesar de minimizar a variância de modo efetivo, não retorna o agrupamento desejado (Figura 7a). Na figura 7b podemos ver a amostra final retornada pelo algoritmo genético (elementos destacados com uma cruz) contendo vários elementos nas bordas das partições. Em seguida se pode ver o resultado

do k-médias executado na amostra final (Figura 7c). É possível visualizar que a soma das distâncias até a média é elevada. Finalmente, todos os elementos do conjunto de dados são reclassificados (Figura 7d) de acordo com as distâncias das médias das partições obtidas na amostra.

4.1 Implementação

Nesta seção serão discutidos alguns detalhes da implementação realizada no algoritmo genético e nos seus operadores de cruzamento, mutação e seleção natural.

O algoritmo genético é uma técnica bem conhecida de otimização que imita o processo de seleção natural (GOLDBERG et al., 1989; HOLLAND, 1975). Através das operações de cruzamento, mutação, e seleção natural, pode-se simular o processo de evolução natural que ocorre na natureza. O algoritmo genético trabalha com uma população de indivíduos e cada indivíduo representa uma solução do problema.

Dessa maneira, os indivíduos mais adaptados são propensos a sobreviverem ao processo de seleção natural, produzindo descendentes ou se perpetuando por várias gerações. Em contrapartida, os menos adaptados tendem a sumir da população rapidamente devido à seleção natural. O critério que mede a adaptabilidade de cada indivíduo varia de aplicação para aplicação.

A estrutura principal de todas as implementações do algoritmo genético é composta por uma população de indivíduos, e cada indivíduo representa uma solução para o problema. No caso específico do problema de amostragem, o indivíduo é uma amostra do conjunto de dados e é codificada como uma lista de índices correspondentes aos elementos do conjunto de dados. O algoritmo 7 descreve a estrutura básica do genético, na qual $X = \{x_i\}, (i = 1, \ldots, n)$ representa o conjunto de dados contendo n elementos, k é a quantidade de partições na qual o conjunto de dados será dividido, s a quantidade de elementos de cada indivíduo da população, e p representa a quantidade de indivíduos da população, o parâmetro m determina a probabilidade de mutação e, por fim, g impõe um limite máximo de iterações (gerações) para o laço principal do algoritmo. A saída do algoritmo ($Y = \{y_i\}, (i = 1, \ldots, s)$) representa uma amostra de s elementos retirada do conjunto de dados completo (X).

O primeiro passo do algoritmo é iniciar a população de p indivíduos com uma

amostra aleatória uniforme, sem repetições, e calcular a adaptabilidade de cada indivíduo (linhas 1-6) – resolvendo o problema de otimização descrito na equação 4.1 e retornando o valor obtido no somatório das distâncias quadráticas. Todavia, devido a dificuldade de achar a solução ótima do problema de otimização, a função ADAPTABILIDADE, utilizada nas linhas 5 e 18, é realizada pela heurística k-médias. Essa função diferencia-se do k-médias apenas por retornar o valor da função objetivo e não as associações de cada elemento. Os indivíduos que possuírem uma soma elevada das distâncias quadráticas dos elementos até os respectivos centroides são considerados bem adaptados.

O laço principal se inicia (linha 8) até atingir o número máximo de iterações para o algoritmo, isto é, o número de gerações. A cada iteração do laço principal os seguintes passos são executados até que o número de indivíduos na população dobre: os pais são selecionados através de uma roleta dentre os p indivíduos mais antigos da população (linhas 10-11). A probabilidade de seleção na roleta é proporcional a adaptabilidade de cada indivíduo (BACK, 1996), isto é, a probabilidade de seleção do indivíduo x_i em cada sorteio será de $p(x_i) = a_i / \sum_{j=1}^{p} a_j$, onde a_i representa a adaptabilidade do i-ésimo indivíduo e p o tamanho da população. Após a seleção, os filhos são gerados através da função de cruzamento (linha 12) e a chance de cada filho sofrer mutação é sorteada (linhas 13-16). Os operadores de cruzamento e mutação serão descritos em mais detalhes adiante.

Quando a população atinge o dobro do tamanho inicial, todos os indivíduos são avaliados pela função de avaliação (linha 18-19) e a metade da população menos adaptada é descartada pela seleção natural (linha 20). Após essas etapas, uma nova iteração do laço principal (geração) é realizada até que o critério de parada do algoritmo seja atendido. Após sair do laço principal, a amostra que possuir maior adaptabilidade é retornada.

O operador de cruzamento é responsável por misturar dois indivíduos da população e gerar novas soluções a partir dessas. Os detalhes da implementação variam de uma implementação para outra. Mas, normalmente, tal processo exige a divisão das variáveis atribuídas aos pais (cromossomo) em duas partes, e cada filho recebe uma parte de cada. Na implementação do operador (Algoritmo 8) acontece basicamente o processo descrito, com uma diferença para prevenir que algum elemento seja duplicado. Caso ambos os pais possuam um elemento em comum na amostra o operador obriga que cada um dos filhos receba uma cópia daquele elemento, prevenindo assim que algum elemento seja duplicado nos filhos. Os elementos em comum, isto é, a interseção dos pais $p1$ e $p2$ são atribuídos a

Entrada: $X = \{x_i\}, (i = 1, \ldots, n), k, s, p, m, g$
Saida: $Y = \{y_i\}, (i = 1, \ldots, s)$

1 *população* $\leftarrow \emptyset$;
2 **para** *i* $\leftarrow$ *0 até P* **faça**
3 EMBARALHA(X);
4 *individuo* $\leftarrow \{x_1, \cdots, x_s\}$;
5 *individuo.adaptabilidade* $\leftarrow$ ADAPTABILIDADE(*individuo*,*k*);
6 *população* $\leftarrow$ *população* $\cup$ *individuo*;
7 $g' \leftarrow 0$;
8 **enquanto** *g'* $< g$ **faça**
9 **enquanto** $|$*população*$| < 2p$ **faça**
10 *p1* $\leftarrow$ ROLETA(*população*,*p*);
11 *p2* $\leftarrow$ ROLETA(*população*,*p*);
12 {*f1*,*f2*} $\leftarrow$ CRUZAMENTO(*p1, p2*);
13 **se** *rand(0,1)* $\leq m$ **então**
14 MUTAÇÃO(*f1*,*X*);
15 **se** *rand(0,1)* $\leq m$ **então**
16 MUTAÇÃO(*f2*,*X*);
17 *população* $\leftarrow$ *população* $\cup \{f1, f2\}$;
18 **para** *i* $\leftarrow$ *0 até 2p* **faça**
19 *individuo.adaptabilidade* $\leftarrow$ ADAPTABILIDADE(*individuo*,*k*);
20 SELEÇÃO_NATURAL(população);
21 $g \leftarrow g + 1$;
22 $Y = \text{argmax}_{individuo}\{individuo_1.adaptabilidade, \cdots, individuo_p.adaptabilidade\}$;
23 **retorne** *Y*

Algoritmo 7: Algoritmo genético amostral (LUCHI et al., 2015; LUCHI et al., 2016).

cada filho ($f1$ e $f2$) (linhas 1-2). Os elementos restantes (linha 3), que aparecem somente em um dos pais, são sorteados para um dos filhos (linhas 5-6).

O operador de mutação é responsável por introduzir nova informação na população, mantendo uma diversidade genética de uma geração para a seguinte. É fácil perceber que sem o operador de mutação, os elementos sorteados inicialmente para compor a população seriam apenas recombinados através do cruzamento. Todavia, nenhum outro elemento diferente seria adicionado até o término do algoritmo.

De modo simples, a mutação remove um elemento contido no indivíduo e o substitui por um novo elemento escolhido aleatoriamente do conjunto de dados. A probabilidade de um elemento ser escolhido para ser removido do indivíduo é inversamente proporcional à distância da média de sua partição. Assim, elementos que contribuem pouco para

Entrada: $p1, p2$
Saída: $\{f1, f2\}$

1 $f1 \leftarrow p1 \cap p2$;
2 $f2 \leftarrow p1 \cap p2$;
3 $aux \leftarrow (p1 \cup p2) \setminus (p1 \cap p2)$;
4 EMBARALHA(aux);
5 $f1 \leftarrow f1 \cup \{aux_1, \cdots, aux_{|aux|/2}\}$;
6 $f2 \leftarrow f2 \cup \{aux_{|aux|/2+1}, \cdots, aux_{|aux|}\}$;
7 **retorne** *{f1, f2}*

Algoritmo 8: Operador de cruzamento (LUCHI et al., 2015; LUCHI et al., 2016).

Entrada: $individuo, X = \{x_i\}, (i = 1, \ldots, n)$
Saída: *individuo*

1 $e1 \leftarrow$ ROLETA_INDIVIDUO(*individuo*);
`/* Probabilidade de seleção do elemento é inversamente proporcional à distância até o centroide de sua partição. */`
2 $e2 \leftarrow x_{\text{rand}(1,|\text{X}\setminus individuo|)} \in X \setminus individuo$;
3 $individuo \leftarrow individuo \setminus \{\text{e1}\}$;
4 $individuo \leftarrow individuo \cup \{\text{e2}\}$;
5 **retorne** *individuo*

Algoritmo 9: Operador de mutação (LUCHI et al., 2015; LUCHI et al., 2016).

o somatório dos erros quadráticos são mais propensos a serem removidos do indivíduo (Algoritmo 9, linha 1). Essa função é muito similar à roleta para a seleção dos pais na iteração do genético e, devido a semelhança, foi chamada de ROLETA_INDIVIDUO. Apesar das semelhanças é importante não confundir a função de cada uma, a primeira seleciona um indivíduo da população dando uma chance maior de seleção para os indivíduos mais adaptados, já a última seleciona um elemento ruim que compõe um indivíduo para ser removido dele.

Para efetuar esse procedimento é necessário calcular a distância de todos os elementos para o centroide mais próximo e então calcular a chance de cada elemento ser sorteado. Como existem poucos elementos na amostra essa análise é rápida. Porém, para escolher qual elemento deve ser inserido no indivíduo é inviável analisar o conjunto de dados inteiros, sem comprometer o desempenho do algoritmo. Portanto, o elemento que deve entrar na amostra é simplesmente sorteado aleatoriamente (linha 2). Depois de escolhidos os elementos, a substituição é realizada (linhas 3 e 4).

Vale ressaltar que tanto o algoritmo genético apresentado nessa seção quanto o algoritmo amostral baseado no k-médias, de Xiao et al. (2014), possuem problemas em

relação à seleção de ruídos para compor a amostra; estes elementos normalmente contribuem significativamente para a soma dos erros quadráticos, devido a estarem distantes dos centróides.

A condição de parada do algoritmo foi fixada em 50 gerações, isto é, 50 iterações do laço principal (linha 6). O tamanho da população foi fixado em 10 e, a probabilidade de mutação também se manteve fixa em 5%. Os valores para esses parâmetros foram fixados pois produziram bons resultados sem a necessidade de um ajuste fino de um conjunto de dados para outro.

O parâmetro k também é conhecido, pois se assume que o usuário possui o conhecimento desse parâmetro para executar o k-médias na base de dados completa e deseja executar o método amostral apenas para aliviar o custo computacional. O único parâmetro que realmente precisa de ajustes é o tamanho da amostra (s). Na seção seguinte será abordada uma forma de ajustar esse parâmetro.

4.2 Ajuste do tamanho amostral

Nesta sessão se discutirá um método para determinar o tamanho da amostra necessário para se obter bons resultados para o algoritmo genético. Essa análise foi realizada com base no tamanho amostral necessário para o k-médias retornar bons resultados utilizando uma amostragem aleatória uniforme. Portanto, primeiramente será descrito este método.

O problema de determinar o tamanho amostral necessário para o k-médias foi encarado como um problema de estimativa no qual os parâmetros a serem determinados são as médias das partições. A prática usual para se determinar o tamanho amostral necessário em um estudo multivariado no qual se deseja estimar a média de várias variáveis consiste em especificar, para cada variável i, o nível de precisão requerido – em termos do erro aceitável d_i e nível de precisão α. Para cada variável i, o tamanho da amostra n_i é dado por $n_i = (z_{\alpha/2}\sigma_i)^2/d_i^2$, onde σ_i é o desvio padrão da respectiva variável e $z_{\alpha/2}$ é o $100(1-\alpha/2)$ percentil da distribuição normal. Esse processo se repete para todas as M variáveis, e o maior n_i é o tamanho amostral necessário.

Considerando $\alpha = 95\%$ e $d_i = 0.1\sigma_i$, isto é, o erro para a i-ésima variável é 10% do σ_i, é necessária uma amostra de tamanho de $n = 385$ elementos (NAING; WINN; RUSLI,

2006) de cada partição para estimar sua média. Todavia, em teoria, o total de $N = k \times n$ elementos, sendo n de cada partição, é necessário para se estimar as k médias, onde k representa o número de partições. Contudo, como em análise de agrupamento os elementos não são rotulados de antemão, o número de elementos para cada partição oriundo de uma amostra de tamanho N é uma variável aleatória. De fato, seja N_i o número de elementos advindos da i-ésima partição dentre os N elementos amostrados, a lista $(N_1, N_2, \ldots, N_k)$ segue uma distribuição multinomial com parâmetros $p_1, \ldots, p_k$ e N, onde p_i representa a porção dos elementos em cada partição.

Definimos o tamanho necessário para a amostra como o menor N que satisfaça $P(N_1 \geq n, N_2 \geq n, \ldots, N_k \geq n) \geq \alpha_N$, ou seja, o valor mínimo de N que satisfaça a equação 4.2

$$\sum_{n_1} \ldots \sum_{n_k} P(N_1 = n_1, \ldots, N_k = n_k) \geq \alpha_N, \tag{4.2}$$

para $N = n_1 + \ldots + n_k$ e $n_i \geq n$ $(i = 1, \ldots, k)$. Na equação 4.2 $P(N_1 = n_1, \ldots, N_k = n_k)$ é a função de densidade da distribuição multinomial dado pela equação 4.3.

$$P(N_1 = n_1, \ldots, N_k = n_k) = \frac{N!}{n_1! \ldots n_k!} p_1^{n_1} \ldots p_k^{n_k}. \tag{4.3}$$

O parâmetro p_i é desconhecido e deve ser obtido com estudos prévios ou através de um estudo piloto. Além disso, o problema de encontrar o valor de N que satisfaça a equação 4.2 se torna intratável para valores altos de k. Para superar essa dificuldade se determina o tamanho amostral pelo menor valor de N que torna o número de elementos da partição de menor cardinalidade (N^*) maior que n com a probabilidade de α.

A variável aleatória N^* segue uma distribuição binomial com parâmetros N e $p^* = min(p_1, \ldots, p_k)$ e, portanto, o tamanho amostral N mínimo tal que $P(N^* \geq n) \geq \alpha_N$. Se considerarmos que a distribuição normal serve como uma boa aproximação da distribuição binomial o cálculo do tamanho amostral torna-se trivial.

Em resumo, obtém-se uma amostra do conjunto de dados pequena para realizar um estudo piloto. O primeiro passo é executar o k-médias em uma amostra do conjunto de dados com uma amostra de 512 elementos, como sugerido em Chakravarty (1999), e verificar quantos elementos estão contidos na partição de menor cardinalidade. A partir

Tabela 1 – Tamanho amostral necessário para o k-médias para cada conjunto de dados e valor de k.

k	NE	FLA	COL	BAY	LKS	NY	CAL	NW	E	W	CTR	USA
2	1158	1352	1433	1367	1439	884	822	1249	865	899	829	1577
3	1472	2943	1601	1889	1773	1412	4800	1568	1625	1605	1579	1895
4	3859	2740	2957	3315	1963	1945	3500	2390	2462	1881	2318	2377
5	3665	2805	4036	4698	2707	2684	3928	3186	2841	2426	2911	5641
6	4505	5238	7851	6665	7018	3074	5156	3836	4258	2917	3528	3556
7	5077	5259	4125	4800	4490	3987	6082	4907	4731	3975	4714	5617
8	7056	5789	4648	10057	5177	5739	6908	5432	9906	5197	5432	5789
9	5366	13552	8674	10794	6255	6438	16400	6167	7538	7759	5815	6225
10	9347	8042	9549	16005	8191	8731	12411	28277	8242	8848	6255	7805
11	8399	7945	11252	16005	7581	7289	9619	9347	7625	7669	6438	9619
12	11252	14130	11546	15446	9833	13415	11157	9549	9549	9481	7249	8789
13	10134	18200	14130	14924	7805	13552	20442	13835	9689	10973	7495	8968
14	11446	15268	13149	18200	9347	13281	12072	8907	11446	12528	10134	9833
15	22911	17480	17953	14924	11856	14924	17032	13149	14758	12769	9091	15446
16	26058	17032	13552	60431	12183	13415	14758	14758	18980	15814	17953	15094
17	18980	23313	28277	17032	12411	17253	20442	17714	17714	17714	12528	12647
18	13835	14924	15268	18980	14758	16005	24609	22911	28892	12769	18713	12528
19	13415	15628	23730	30910	21783	16005	18453	30910	19831	15814	17714	14130
20	16200	17714	21783	78210	13835	47478	21091	14437	24162	18713	13835	15094

disso, pode-se estimar o tamanho da amostra necessário para que o grupo menor possua pelo menos $n = 385$ exemplos com a probabilidade de $\alpha = 0.95$ usando a distribuição normal como uma binomial aproximada.

Para determinar o tamanho amostral necessário para o algoritmo genético, foi realizado um estudo em seis conjuntos de dados (BAY, CAL, COL, CTR, LKS e NY) disponíveis no site do *9th DIMACS Implementation Challenge*[1] e validado nos outros seis. O tamanho da amostra foi variado para que o algoritmo genético obtivesse um desempenho similar ao do k-médias com o tamanho amostral calculado pelo procedimento descrito acima. O k-médias executou para os conjuntos de dados utilizando os tamanhos amostrais exibidos na tabela 1.

O objetivo do experimento foi descobrir a relação entre o tamanho amostral necessário para o k-médias, obtido por cálculos analíticos baseados no resultado do estudo piloto, e o do algoritmo genético. Os resultados (Figura 8) apontam que o algoritmo genético necessitou de $1/k$ do tamanho amostral utilizado pelo k-médias, que possui o valor de 1 na escala, para atingir resultados de qualidade similares. As linhas em preto representam a fração da amostra necessária para o algoritmo genético apresentar a mesma qualidade, em termos do somatório dos erros quadráticos, que a solução obtida pelo k-médias em cada um dos seis conjuntos de dados (utilizando como tamanho amostral o valor da tabela 1). A linha roxa exibe a curva $1/k$, a qual se ajusta bem às frações de amostras verificadas

[1] Disponível em: http://www.diag.uniroma1.it/challenge9/download.shtml

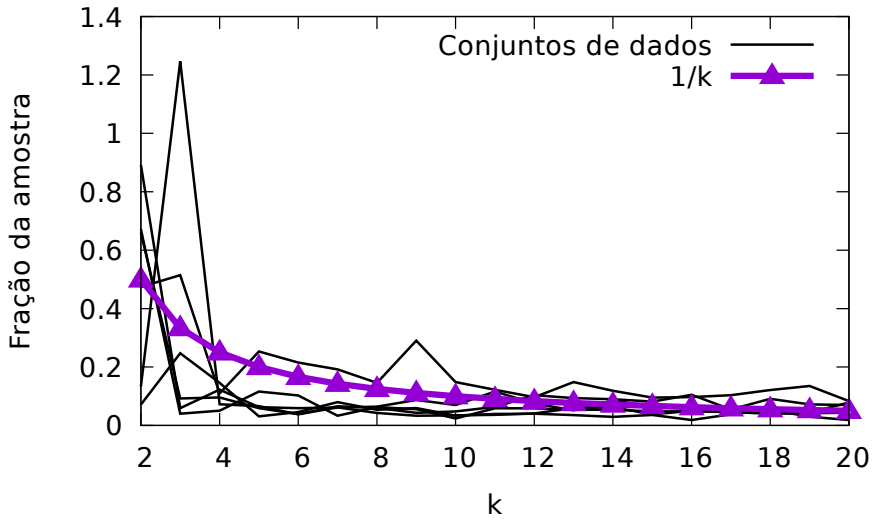
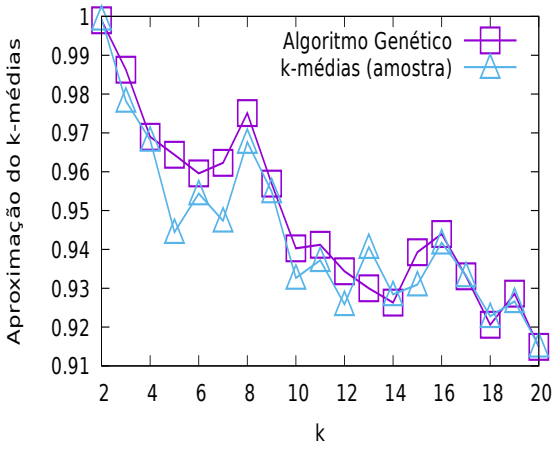


Figura 8 – Relação entre o tamanho amostral requerido pelo k-médias e o algoritmo genético.

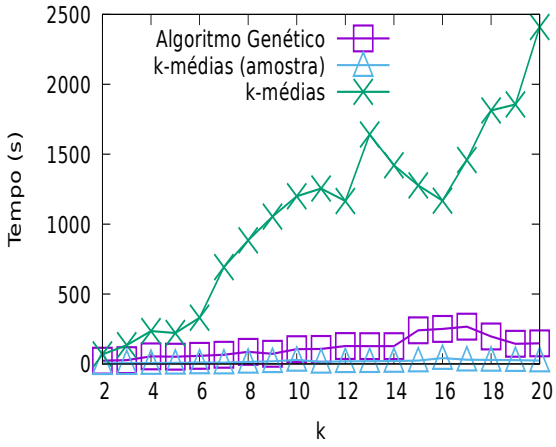
experimentalmente. A figura mostra alguns pontos distantes da previsão, especialmente para valores pequenos de k. A causa desses valores é desconhecida; contudo, muito provavelmente se deva a dificuldade de representar regiões menos densas em conjuntos de dados de alta cardinalidade – que é justamente o problema que a amostragem tendenciosa tenta solucionar.

Na validação desses experimentos, nos outros seis conjuntos de dados, o tamanho amostral de $1/k$ do requerido pelo k-médias se mostrou suficiente. Abaixo (Figuras 9a e 9c), pode-se ver os resultados de dois conjuntos de dados comparando o algoritmo genético contra o k-médias amostral e o k-médias executando sobre o conjunto de dados completo. O algoritmo genético teve uma aproximação similar utilizando apenas uma fração da amostra, confirmando que o fator $1/k$ do tamanho amostral foi suficiente.

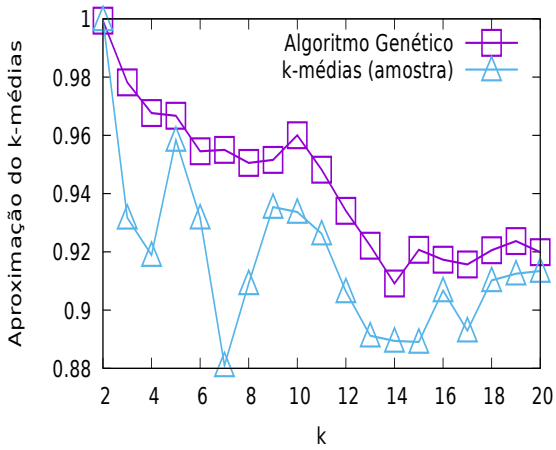
Em relação ao tempo de execução (Figuras 9b e 9d), embora o algoritmo genético tenha apresentado piores resultados – quando comparado com o k-médias, utilizando uma amostragem aleatória uniforme – ambos foram muito mais rápidos que o k-médias executando sobre o conjunto de dados completo. Além disso, pode-se visualizar que o tempo de execução das duas abordagens não cresce linearmente com k, como acontece com o k-médias.



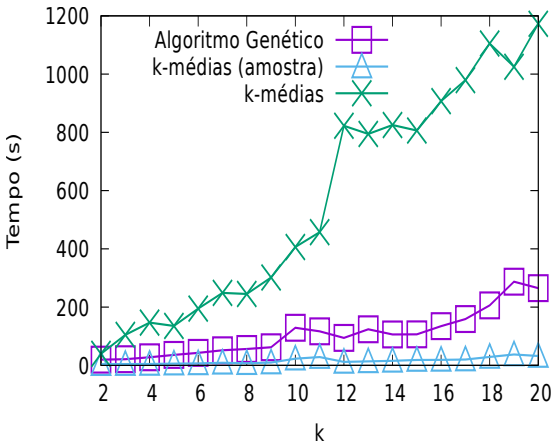
(a) Resultados para o conjunto de dados NE.



(b) Tempo de execução (NE).



(c) Resultados para o conjunto de dados NW.



(d) Tempo de execução (NW).

Figura 9 – Comparação da qualidade dos resultados e tempo de execução entre o k-médias e o algoritmo genético para os conjuntos de dados NE e NW.

5 Métodos amostrais para o DBSCAN

Neste capítulo serão apresentadas duas técnicas amostrais desenvolvidas com a finalidade de usufruir do comportamento do DBSCAN (LUCHI; RODRIGUES; VAREJÃO, 2019). A primeira, chamada *Rough**-DBSCAN é uma variante do *Rough*-DBSCAN, proposto em Viswanath e Babu (2009) conforme seção 3.3.2, que melhora a estimativa de densidade proposta no método original através da utilização de uma versão modificada do algoritmo Líder (HARTIGAN, 1975) na primeira etapa do algoritmo.

A segunda técnica de amostragem tendenciosa, chamada I-DBSCAN, é livre de parâmetros adicionais além daqueles requeridos pelo algoritmo de análise de agrupamento. Essa técnica explora o modo como a densidade é calculada e o conceito de alcançabilidade do DBSCAN para compor a amostra.

Como as duas versões propostas executam uma versão modificada desse algoritmo Líder, chamado Líder*, na etapa inicial do algoritmo, é importante detalhar esse processo e explicar a motivação dessa proposta.

5.1 Líder*

O Líder (HARTIGAN, 1975) é um algoritmo de análise de agrupamento incremental que executa apenas uma varredura no conjunto de dados. Ao contrário do k-médias, o algoritmo Líder não implementa nenhuma etapa de otimização. Portanto, possui um tempo de execução muito menor.

A versão canônica do algoritmo Líder (Algoritmo 10) inicia tornando o primeiro elemento do conjunto de dados $\mathcal{D}_0$ um líder (linha 1) e percorre o restante dos elementos (linha 3), verificando se o elemento corrente se situa a uma distância inferior a τ de algum líder existente. Se o elemento não estiver próximo de nenhum líder, ele se torna um (linhas 10-12). Caso contrário, ele será um seguidor do primeiro líder encontrado a uma distância menor que τ e o laço principal é interrompido (linha 9). É importante notar que quando $\tau \to 0$ todos os elementos do conjunto de dados serão líderes, $\mathcal{L} \to \mathcal{D}$.

É fácil perceber que o Líder é muito sensível a ordem em que os elementos são

Entrada: Conjunto de dados $\mathcal{D}$, distância limite τ, distância ϵ
Saida: Conjunto de líderes $\mathcal{L}$, e conjunto de seguidores $\mathcal{F}$

```
1  L ← D_0;
2  F_0 ← D_0;
3  para cada d ∈ D \ {D_0} faça
4      líder ← verdadeiro;
5      para cada l ∈ L faça
6          se ||l − d|| ≤ τ então
7              F_l ← F_l ∪ d;
8              líder ← falso;
9              break ;
10     se líder então
11         L ← L ∪ d;
12         F_d ← d;
13 retorne {L, F}
```

Algoritmo 10: Algoritmo de análise de agrupamento Líder.

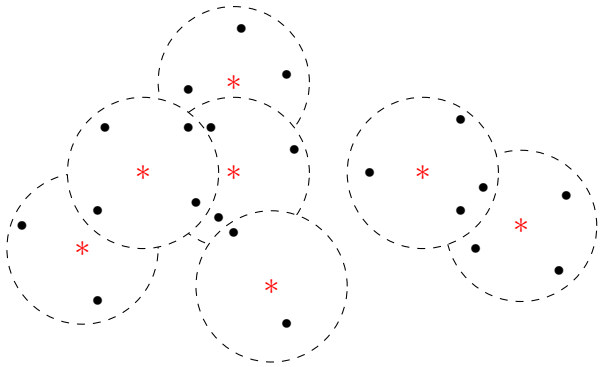


Figura 10 – Exemplo de resultado obtido pelo algoritmo Líder. Os asteriscos, marcados em vermelho, são os líderes e os demais pontos, em preto, representam os seguidores dentro da região do líder (raio τ).

apresentados ao algoritmo. Os primeiros elementos apresentados ao algoritmo possuem uma grande chance de se tornarem líderes. Os últimos elementos, muito provavelmente, seguirão algum líder estabelecido em etapas anteriores. Prestando-se atenção nas linhas 6-9 do algoritmo 10, pode-se notar que o primeiro líder a ser descoberto sempre terá preferência sobre os outros na decisão sobre qual líder o elemento d seguirá. O segundo líder descoberto terá preferência sobre todos os outros, com exceção do primeiro, e assim por diante. Essa preferência na associação entre os elementos e os líderes é ilustrada na figura 10.

Ao se considerar apenas os elementos que seguem cada líder para estimar a densidade dos mesmos, a densidade será subestimada em muitos casos. Provavelmente existirão

Entrada: Conjunto de dados $\mathcal{D}$, distância limite τ, distância ϵ
Saida: Conjunto de líderes $\mathcal{L}$, e conjunto de seguidores $\mathcal{F}$

```
1  L ← D_0;
2  para cada d ∈ D \ {D_0} faça
3      líder ← verdadeiro;
4      para cada l ∈ L faça
5          se ||l − d|| ≤ τ então
6              líder ← falso;
7              break ;
8      se líder então
9          L ← L ∪ d;
10 para cada d ∈ D faça
11     para cada l ∈ L faça
12         se ||l − d|| ≤ ε então
13             F_l ← F_l ∪ d;
14 retorne {L, F}
```

Algoritmo 11: Líder*.

elementos a uma distância menor que τ de um líder; todavia, associado a outro. Isso ocorre devido a prioridade relatada na associação entre os líderes e os demais elementos do conjunto de dados. Esse problema pode ser visualizado na figura 10. Na figura, o líder mais a direita possui três seguidores associados a ele; porém, existem cinco elementos a uma distância menor que τ. Portanto, calcular a densidade dos líderes com base apenas nos seus seguidores é uma tarefa complexa.

As técnicas amostrais propostas neste capítulo e o *Rough*-DBSCAN, proposto em Viswanath e Babu (2009), usam o algoritmo Líder para gerar a amostra. Contudo, a amostra não é construída utilizando todos os líderes, somente os considerados densos em termos do ϵ e minPts utilizados pelo DBSCAN. Por isso, a estimativa correta de densidade de cada líder é de grande importância e, como vimos, realizar essa tarefa a partir da saída do líder original é uma tarefa difícil. Em Viswanath e Babu (2009), os autores contornam esse problema através de uma estimativa da densidade, ao invés de tentar encontrar seu valor exato.

Com uma pequena modificação no algoritmo Líder (Algoritmo 11) é possível contornar o problema de estimativa de densidade existente. A versão modificada basicamente consiste em executar uma segunda varredura sobre o conjunto de dados associando os seguidores com todos os líderes que estão a uma distância inferior a τ (linhas 10-13). Essa

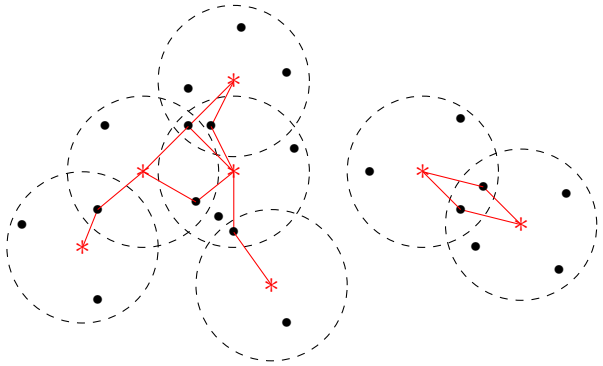


Figura 11 – Exemplo de saída do Líder*. Os líderes são representados em vermelho e os seguidores como os pontos pretos dentro da região do líder. As linhas que interligam alguns elementos aos líderes sinalizam os elementos que seguem simultaneamente mais de um líder.

segunda passagem pelo conjunto de dados elimina o efeito indesejado causado pela preferência das associações e torna o processo de estimativa da densidade muito mais simples. A figura 11 ilustra o resultado da versão modificada. Apesar desse pequeno acréscimo no custo computacional, os resultados são superiores em termos de qualidade devido a correta estimativa de densidade proporcionada pela versão modificada do algoritmo Líder.

É importante notar que a distância utilizada para manter o espaçamento dos líderes na versão modificada é definida pelo parâmetro τ (linha 5); todavia, as associações dos elementos aos líderes, realizadas na segunda varredura do algoritmo, utilizam o parâmetro ϵ do DBSCAN. Isso garante que as associações dos elementos a um dado líder serão idênticas à vizinhança obtida pelo DBSCAN no conjunto de dados completo.

5.2 *Rough**-DBSCAN

O *Rough**-DBSCAN é muito similar ao *Rough*-DBSCAN (Seção 3.3.2) proposto por Viswanath e Babu (2009). Na realidade, a única diferença entre as abordagens está no algoritmo Líder. Ao invés de executar a versão canônica do algoritmo Líder e estimar a densidade de cada líder – através do método de estimativa apresentado em Viswanath e Babu (2009) – o *Rough**-DBSCAN utiliza o Líder* para obter, de modo preciso, as densidades dos elementos. A partir dessa etapa inicial os dois métodos apresentam basicamente o mesmo funcionamento.

Dado os parâmetros do DBSCAN, ϵ e minPts, e o parâmetro τ, o *Rough**-DBSCAN executa a versão modificada do algoritmo de análise de agrupamento Líder e seleciona

Entrada: Conjunto de dados $\mathcal{D}$, τ, ϵ, minPts
Saida: $\{\text{rotulo}_i, (i = 1, \ldots, |D|)\}$

1 $\{\mathcal{L},\mathcal{F}\} \leftarrow$ Líder*$(\mathcal{D}, \tau, \epsilon)$; /* Versão modificada */
2 $\mathcal{S} \leftarrow \{\}$;
3 **para** *cada* $l \in \mathcal{L}$ **faça**
4 **se** $|F_l| \geq minPts$ **então**
5 $\mathcal{S} \leftarrow \mathcal{S} \cup l$;
6 rotulo$' \leftarrow$ DBSCAN$(S, \epsilon, 1)$; /* Grupo de cada líder de S */
7 rotulo $\leftarrow \{R_1 = ruído, \ldots, R_{|D|} = ruído\}$;
8 **para** *cada* $x \in X$ **faça**
9 **se** $\mathcal{L}(x) \in S$ **então**
10 rotulo$_x \leftarrow$ rotulo$'_{\mathcal{L}(x)}$; /* x terá o rótulo do líder de x */
11 **retorne** $\{\text{rótulo}_i\ (i = 1, \ldots, |\mathcal{D}|)\}$

Algoritmo 12: *Rough**-DBSCAN.

apenas os líderes com mais de minPts seguidores em uma esfera de raio ϵ para compor a amostra. Em outras palavras, apenas os líderes que tiverem mais que $minPts$ seguidores são selecionados. Todos os demais elementos do conjunto de dados são descartados. Pode-se ver no algoritmo 12 as etapas que o Rough*-DBSCAN executa para a construção da partição final dos dados baseado na amostra obtida pelo Líder*.

Como todos os elementos da amostra possuem mais que minPts elementos a uma distância menor que ϵ, eles possuem uma densidade acima da requerida pelo DBSCAN para não serem rotulados como ruído. Portanto, o parâmetro $minPts$ torna-se irrelevante e a atribuição minPts= 1 pode ser feita sem nenhum prejuízo. O passo seguinte é a execução do algoritmo DBSCAN nessa amostra com parâmetros ϵ e minPts= 1.

Após a execução do DBSCAN na amostra, obtendo-se o conjunto de rótulos para cada elemento de S, o restante dos elementos do conjunto de dados serão associados a uma das partições identificadas. Essa associação é realizada rotulando todos os elementos seguidores dos líderes, resultante da primeira etapa do algoritmo, com o mesmo rótulo que o DBSCAN atribuiu para o líder em questão.

Devido às intersecções de elementos associados a mais de um líder, é possível que o algoritmo tente rotular um elemento mais de uma vez. Nessa situação, o elemento será associado à partição de um dos líderes escolhido arbitrariamente. Apenas o último rótulo fornecido para cada elemento é considerado, na linhas 10 do algoritmo 12 podemos ver que os rótulos atribuídos aos seguidores podem ser sobrescritos nas iterações seguintes.

Os passos descritos podem ser visualizados no algoritmo 12. Na linha 1 a versão modificada do líder é executada e retorna o conjunto de líderes e seguidores, indicado respectivamente pelas letras $\mathcal{L}$ e $\mathcal{F}$. O passo seguinte é percorrer a lista de líderes (linha 3) e verificar se cada líder l possui o número de seguidores acima do requerido pelo DBSCAN (linha 4), isto é, acima de minPts. Somente os elementos que tiverem a densidade igual ou maior que esse limite serão inseridos na amostra $\mathcal{S}$ (linha 5). O passo seguinte é a execução do DBSCAN em $\mathcal{S}$, com os parâmetros ϵ e minPts $= 1$ (linha 6).

A última etapa do *Rough**-DBSCAN é a atribuição dos rótulos para todos os elementos. Inicialmente todos os elementos são rotulados como ruído (linha 7). A seguir, um laço se inicia para atribuir um rótulo para cada elemento (x) do conjunto de dados (linha 8). Se o líder de x, representado por $\mathcal{L}(x)$, for denso (linha 9) – e, consequentemente pertencer a S – o rótulo de x ($rotulo_x$) será o mesmo atributo ao seu líder ($rotulo'_{\mathcal{L}(x)}$) pelo DBSCAN (linha 10).

É importante ressaltar que quando $\tau \to 0$ o resultado do *Rough**-DBSCAN aproxima do resultado obtido pelo DBSCAN no conjunto de dados completo. Nesse cenário, a saída do algoritmo Líder* será composta por todos os elementos do conjunto de dados $\mathcal{D}$, onde todos serão líderes, formando $|\mathcal{D}|$ partições sem seguidores. A densidade de cada líder é aferida e a entrada do algoritmo DBSCAN será composta apenas dos elementos densos. O DBSCAN irá agrupar os líderes em partições e cada elemento, ao final, será substituído por ele mesmo, no mesmo grupo. O resultado do algoritmo Líder* será composto de várias partições de cardinalidade unitária. Todos os demais elementos serão categorizados como ruído.

Na prática, o parâmetro τ age como um fator de aproximação do DBSCAN no conjunto de dados completo. Quando $\tau = 0$, o algoritmo retorna o mesmo resultado do DBSCAN. A medida que o valor de τ aumenta, os resultados se diferenciam cada vez mais do obtido pelo DBSCAN no conjunto de dados completo, mas, com considerável redução no tempo de execução.

5.3 I-DBSCAN

Além do *Rough**-DBSCAN, uma segunda técnica amostral chamada I-DBSCAN (I é uma abreviação para Intersecção) foi desenvolvida para gerar uma amostra que explora

as características de funcionamento do DBSCAN. Ao contrário do *Rough**-DBSCAN, que é uma melhoria em cima do algoritmo original, o I-DBSCAN é um método inteiramente novo.

Assim como o *Rough**-DBSCAN, o I-DBSCAN também utiliza a versão modificada do algoritmo Líder apresentado nesse capítulo (Seção 5.1). A ideia principal do método é construir uma amostra contendo todos os líderes, não somente os densos, advindos da primeira etapa do algoritmo juntamente com elementos extras para garantir a densidade e alcançabilidade requerida para o DBSCAN. Esses elementos extras são aqueles que estão associados a mais de um líder nas intersecções.

Ao contrário do *Rough*-DBSCAN e do *Rough**-DBSCAN, o I-DBSCAN não possui o parâmetro τ. A técnica precisa somente dos parâmetros ϵ e minPts, que são conhecidos; afinal, são requeridos pelo algoritmo DBSCAN. Portanto, não existe a necessidade de sintonia.

Devido a ausência do parâmetro τ no I-DBSCAN, o algoritmo executa o Líder* com $\tau = \epsilon$, e por essa razão não pode executar utilizando somente os líderes como amostra. O parâmetro τ atua como um separador no Líder* e todos os líderes estarão no mínimo a essa distância um dos outros. No caso de $\tau = \epsilon$, todos os líderes estarão a pelo menos ϵ um dos outros. Se o DBSCAN fosse executado sobre essa amostra com minPts > 1 todos os elementos seriam rotulados como ruído, afinal, a vizinhança de cada elemento conteria apenas ele próprio. No caso de minPts $= 1$, o oposto ocorreria e cada elemento da amostra seria associado a uma partição diferente. Portanto, fica evidente que compor a amostra apenas utilizando os líderes encontrados pelo Líder* quando $\tau = \epsilon$ não é o suficiente.

Para assegurar que um líder denso no conjunto de dados completo, isto é, com mais de minPts seguidores, permaneça denso na amostra, essa deve conter no mínimo minPts-1 seguidores. Contudo, devido a existência de elementos que seguem mais de um líder não há a necessidade de adicionar minPts-1 elementos para cada um. Adicionar elementos nas interseções é uma boa ideia por duas razoes; a primeira é contribuir para a densidade de mais de um líder, o que ajuda a manter o tamanho da amostra reduzido. A segunda razão é ajudar no processo de alcançabilidade do DBSCAN, facilitando que um líder alcance o outro.

Os elementos que seguem mais de um líder, por definição, estão a uma distância

Entrada: Conjunto de dados $\mathcal{D}$, ϵ, $minPts$
Saida: $\{\text{rotulo}_i, (i = 1, \ldots, |D|)\}$

1 $\{\mathcal{L},\mathcal{F}\} \leftarrow$ Líder*$(\mathcal{D}, \epsilon, \epsilon)$; /* Versão modificada */
2 $\mathcal{S} \leftarrow \mathcal{L}$;
3 **para** *cada* $l \in \mathcal{L}$ **faça**
4 $S' \leftarrow \bigcup_{i \in \mathcal{L} \setminus \{l\}} l \cap i$;
5 **se** $|\mathcal{F}_l| > minPts$ **então**
6 **se** $|S'| > minPts$ **então**
7 $S' \leftarrow$ AMOSTRAGEM_FFT$(S', minPts)$;
8 **senão**
9 $S' \leftarrow S' \cup$ AMOSTRAGEM_UNIFORME$(F_l, minPts - |S'|)$;
10 $\mathcal{S} \leftarrow \mathcal{S} \cup S'$;
11 rotulo$' \leftarrow$ DBSCAN$(S, \epsilon, minPts)$; /* Grupo de cada líder de S */
12 rotulo $\leftarrow \{y_1 = ruido, \ldots, y_{|D|} = ruido\}$;
13 **para** *cada* $x \in X$ **faça**
14 **se** $\mathcal{L}(x) \in S$ **então**
15 rotulo$_x \leftarrow$ rotulo$'_{\mathcal{L}(x)}$; /* x terá o rótulo do líder de x */
16 **retorne** $\{\text{rótulo}_i\ (i = 1, \ldots, |\mathcal{D}|)\}$

Algoritmo 13: I-DBSCAN.

menor que ϵ de todos eles. Portanto, tais elementos atuam como uma ponte interligando diversos líderes, ajudando a todos os líderes serem interligados através desse elemento. A única exceção ocorre quando um dos líderes é classificado como ruído pelo DBSCAN. Nesse caso o elemento da interseção será atribuído à partição do líder denso.

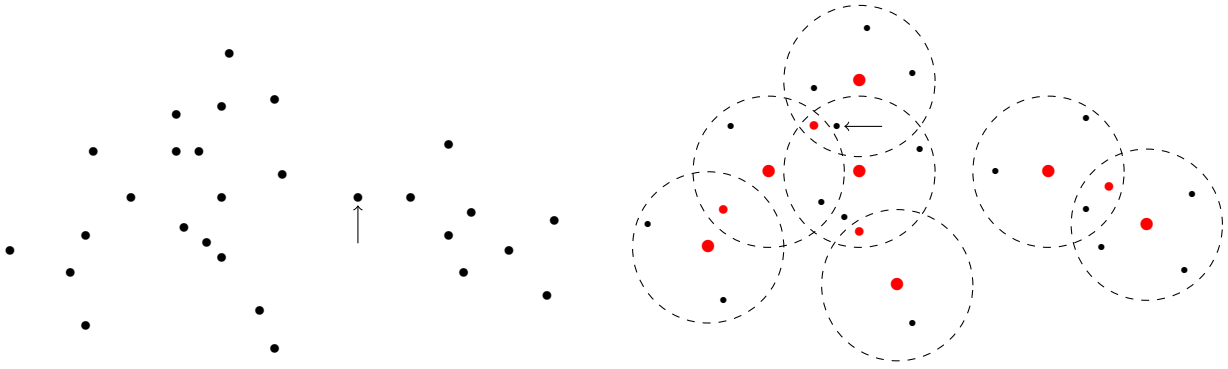
O I-DBSCAN inicia executando o Líder* com o parâmetro $\tau = \epsilon$, como pode ser visto na primeira linha do algoritmo 13. Depois da execução do Líder*, todos os líderes são inseridos na amostra (linha 2). O restante do algoritmo amostral basicamente examina cada líder (linha 3) para adicionar novos elementos na amostra. Se o líder contiver até minPts seguidores, ou seja, todos os seguidores daquele líder são adicionados na amostra (linha 10). Porém, caso o líder possua mais seguidores do que o requisito de densidade do DBSCAN, alguns elementos podem ser descartados sem prejuízo. É importante notar que os líderes pouco densos serão rotulados como ruído na execução do DBSCAN em um primeiro momento, contudo, os seguidores desses líderes podem estar contidos em interseções com uma densidade elevada. Quando isso ocorre, os elementos nas interseções servirão de ponte – durante etapa da busca em largura do DBSCAN – para atribuir um grupo ao líder que inicialmente foi rotulado como ruído.

Caso o líder seja denso (linha 5), dois cenários são considerados pelo algoritmo. O primeiro acontece quando o líder possui menos que minPts elementos nas interseções. Nesse caso, todos os elementos nas interseções são adicionados na amostra além de alguns elementos escolhidos de forma aleatória uniforme pela função Amostragem Uniforme (linha 9). Esses elementos extras são adicionados simplesmente para garantir que o líder permaneça denso na amostra para os mesmo parâmetros ϵ e minPts utilizado pelo DBSCAN, evitando assim que um líder originalmente denso no conjunto de dados seja rotulado como ruído. Caso o líder possua exatamente minPts elementos na interseção, essa etapa não altera a amostra, afinal, $minPts - |S'| = 0$.

O segundo cenário ocorre quando um líder possui mais que minPts elementos nas interseções com outros líderes (linha 6). Nesse caso, não há a necessidade de inserir todos eles na amostra para garantir a densidade do líder, com o objetivo de manter a amostra a menor possível, somente a quantidade necessária é escolhida. Essa escolha é realizada utilizando o algoritmo FFT (Seção 3.1.3) como mecanismo amostral (linha 7). A amostragem FFT inicialmente sorteia um elemento e então sucessivamente seleciona o elemento mais distante dos selecionados previamente. Esse procedimento assegura que os elementos sorteados estejam bem espaçados e, provavelmente, abrangendo múltiplas interseções. Nas figuras 12a e 12b pode-se ver respectivamente o conjunto de dados completo e a amostra gerada pelo I-DBSCAN.

Depois de construída a amostra, os passos seguintes são similares ao *Rough**-DBSCAN. O DBSCAN é executado na amostra com os parâmetros minPts e ϵ originais (linha 11), e após a execução, a partição final é retornada rotulando cada elemento do conjunto de dados com o rótulo de seu líder (linhas 12-15). Os elementos nas interseções são desconsiderados nessa rotulação final pois foram apenas adicionados para garantir a densidade dos líderes e ajudar na alcançabilidade do DBSCAN para encontrar a partição correta do conjunto de dados.

Um problema raro, mas que pode ocorrer devido ao sorteio de quais elementos nas intersecções irão compor a amostra, é a segmentação de uma partição na amostra, especialmente nos casos que minPts possui um valor pequeno. Na figura 12b, se o seguidor escolhido para compor a amostra do líder mais acima na figura fosse o elemento destacado por uma flecha aconteceria o problema descrito.



(a) (b)

Figura 12 – Exemplo de um conjunto de dados completo (a) e a amostra gerada pelo I-DBSCAN com minPts = 2 (b). Os elementos escolhidos para compor a amostra estão destacados em vermelho. Os elementos na cor preta, e com tamanho reduzido, representam os demais elementos do conjunto de dados. Os elementos apontados por flechas serão utilizados como exemplo no texto.

Nesse exemplo, os dois líderes a esquerda na figura 12b não conseguiriam alcançar os três líderes situados no meio da figura através dos elementos contidos na amostra pois estariam a uma distância superior a ϵ. Portanto, a solução do DBSCAN executado na amostra identificaria erroneamente três partições. Esse problema é, experimentalmente, raro; afinal, o I-DBSCAN consegue um resultado com um nível de acordo alto em relação ao DBSCAN executado no conjunto de dados completo. Contudo, um possível ponto de melhoria para o algoritmo seria a solução deste problema.

No caso descrito acima, o algoritmo dividiu uma partição coesa em mais de uma componente; mas, um problema inverso também pode ocorrer. Isto é, unir partições distintas devido à ordem de escolha dos líderes. Na figura 12a existe um elemento situado entre as duas partições (destacado com uma flecha). Se esse elemento fosse um líder, devido à ordenação dos elementos no conjunto de dados, os dois grupos seriam unidos formando uma única partição. Esse erro também é raro; além disso, os únicos casos de possíveis ocorrências são aqueles em que diferentes partições estejam separadas por uma distância menor que ϵ, isto é, somente quando as duas partições estiverem muito próximas uma da outra a ponto de um elemento na borda unir as duas.

6 Experimentos computacionais

Neste capítulo serão descritos os procedimentos experimentais e os resultados obtidos na comparação dos diferentes métodos de amostragem. Os experimentos foram realizados com o objetivo de medir o nível de aproximação entre os resultados obtidos pelos algoritmos de análise de agrupamento (k-médias e DBSCAN) – executados sobre os conjuntos de dados completos – e os resultados obtidos pelos mesmos quando executados sobre as amostras resultantes de diferentes métodos de amostragem.

Os passos executados para cada teste foram: a execução do algoritmo amostral para se retirar uma amostra do conjunto de dados, seguido da execução do algoritmo de análise de agrupamento sobre a amostra. Após a execução, os demais elementos do conjunto de dados – que não estavam presentes na amostra – são associados às partições encontradas.

O procedimento de associação adotado variou de acordo com o algoritmo de análise de agrupamento. Para o k-médias, os elementos foram associados de acordo com a proximidade das médias das partições identificadas na amostra. Por outro lado, nos testes utilizando o algoritmo DBSCAN, caso o elemento seja alcançável por algum outro elemento denso, os dois pertencerão à mesma partição. Contudo, caso o elemento não seja alcançável, o mesmo foi rotulado como ruído.

Uma vez que todos os elementos estejam devidamente rotulados, o nível de acordo entre a solução com aplicação do algoritmo amostral e com aplicação direta do algoritmo de agrupamento sobre o conjunto completo de dados é aferido pela métrica *Adjusted Rand Index* (ARI) (HUBERT; ARABIE, 1985). O ARI é uma variação do *Rand Index* (RI) (RAND, 1971) e mede o nível de acordo entre dois conjuntos de rótulos. Entretanto, ao contrário do RI, o ARI considera a acurácia esperada por uma rotulação aleatória. Portanto, é uma métrica mais significativa.

Por exemplo, em um cenário no qual existam duas partições de igual cardinalidade é esperado 50% de acerto por uma rotulação aleatória dos elementos. Nesse caso, se algum método obtiver o nível de acordo – entre a solução obtida em relação à retornada no conjunto de dados completo – de 50%, o valor do ARI será 0. Outro exemplo extremo,

mas bastante ilustrativo, é o caso do conjunto de dados possuir duas partições altamente desbalanceadas – uma partição contendo poucos elementos e a outra partição contendo todos os elementos restantes do conjunto de dados. Nesse exemplo, um RI próximo de 1, indicando 100% de concordância entre duas soluções não significa muito; afinal, se fosse atribuído o rótulo do grupo de maior cardinalidade para todos os elementos o nível de acordo seria próximo de 100%. No ARI esse desbalanceamento seria levado em consideração e, no exemplo de rotulação citado acima o ARI seria próximo de 0.

O ARI não retorna sempre o resultados no intervalo $[0, 1]$ como no RI. A fórmula geral pode ser vista na equação 6.1, onde o índice esperado é o nível de acordo obtido através de uma rotulação aleatória. Ao analisar a equação 6.1, nota-se que, nos casos em que a concordância entre as soluções for abaixo de um resultado aleatório, a métrica resultará em um valor negativo. O limite inferior para a métrica varia dependendo do número de grupos e do desbalanceamento encontrado no conjunto de dados. Já nos casos de concordância completa entre dois conjuntos de rótulos o ARI retorna o valor máximo (1). Esse procedimento é descrito em detalhes em Hubert e Arabie (1985).

$$ARI = \frac{\text{índice} - \text{índice esperado}}{\text{índice máximo} - \text{índice esperado}} \tag{6.1}$$

O primeiro passo para o cálculo do ARI é a construção de uma tabela de contingência. Dado um conjunto de dados S com n elementos e dois particionamentos desse conjunto ($X = \{X_1, X_2, \ldots, X_r\}$ e $Y = \{Y_1, Y_2, \ldots, Y_s\}$), sendo que r e s são, respectivamente, a quantidade de grupos identificados nos particionamentos X e Y. Cada elemento da tabela (n_{ij}) representa a quantidade de elementos em comum entre as partições X_i e Y_j, ou seja, $n_{ij} = |X_i \cap Y_j|$. Além disso, a_i representa a soma da i-ésima linha da tabela e b_j representa a soma da j-ésima coluna. A equação 6.2 exibe a formula completa para o cálculo do ARI.

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}} \tag{6.2}$$

Idealmente, busca-se uma amostra que seja equivalente ao conjunto de dados completo ou, pelo menos, que seja uma boa aproximação deste conjunto para o algoritmo de agrupamento adotado. Portanto, nos experimentos, os resultados obtidos pelos algoritmos de análise de agrupamento – k-médias e DBSCAN – no conjunto de dados completo são

considerados corretos, não importando a qualidade da solução encontrada. E os resultados obtidos pelos métodos de agrupamento executados sobre a amostra devem aproximar do resultado obtido no conjunto de dados original.

A motivação principal das amostragens apresentadas nesse trabalho é atenuar os problemas de escalabilidade apresentados pelos algoritmos de análise de agrupamento. Portanto, é desejado um método amostral que seja capaz de gerar uma amostra que se aproxime – para o algoritmo adotado – do conjunto de dados completo, e que possua, ao mesmo tempo, um custo computacional reduzido.

Os experimentos computacionais foram divididos em duas partes, uma para cada método de análise de agrupamento. Os experimentos realizados com k-médias utilizaram sete algoritmos de amostragem tendenciosa, descritos no capítulo 3, além da amostragem aleatória uniforme. Os algoritmos excluídos nesta primeira etapa experimental foram: Líder, *Rough*-DBSCAN, *Rough**-DBSCAN e I-DBSCAN, descritos respectivamente nas seções 3.1.1, 3.3.2, 5.2, 5.3.

A exclusão desses algoritmos se deve aos parâmetros de entrada exigidos pelos mesmos. Com exceção do I-DBSCAN, os outros três métodos precisam conhecer um parâmetro τ que representa uma distância que é bastante relacionada ao parâmetro ϵ do DBSCAN. Estimar esse valor, baseado apenas no parâmetro do k-médias, é uma tarefa árdua. Em Ros e Guillaume (2016a), os autores relatam a dificuldade do uso do método amostral baseado no algoritmo Líder em conjunto com o k-médias. Além disso, os métodos *Rough*-DBSCAN, *Rough**-DBSCAN e o I-DBSCAN, apresentam dificuldades ainda maiores devido à necessidade de se conhecer os parâmetros minPts e ϵ do DBSCAN.

Nos experimentos envolvendo o algoritmo de análise de agrupamento DBSCAN, somente os quatro algoritmos apropriados para o DBSCAN foram utilizados. Isto é, Líder, *Rough*-DBSCAN, *Rough**-DBSCAN e I-DBSCAN. A exclusão dos outros algoritmos de amostragem da segunda etapa experimental é motivada por dois fatores.

O primeiro fator é a necessidade de alguns métodos amostrais de conhecer a quantidade de partições existentes no conjunto de dados. Como essa informação tipicamente não está disponível para quem executa o algoritmo de agrupamento DBSCAN, esses métodos não podem ser utilizados com sucesso. Tanto o método amostral baseado na agregação do *bootstrap* quanto o algoritmo genético amostral se enquadram nesse cenário.

O segundo fator considerado é muito mais relevante, e corresponde ao desconhecimento em relação aos parâmetros para se executar o DBSCAN na amostra obtida. Por exemplo, se possuirmos o conhecimento dos parâmetros ϵ e minPts para executar o DBSCAN no conjunto de dados completo e, retirarmos uma amostra de 1% da cardinalidade do conjunto de dados através de um processo aleatório uniforme, não se pode executar o algoritmo utilizando os mesmos parâmetros na amostra. Afinal, a amostra é muito menos densa que o conjunto de dados completo e a utilização dos mesmos parâmetros faria com que todos os elementos fossem rotulados como ruído pelo algoritmo.

No exemplo citado acima, o ajuste do parâmetro minPts também não é viável; afinal, o mesmo é, tipicamente, pequeno e só permite valores inteiros. A tentativa de remediar o problema devido a perda de densidade da amostra foi considerar minPts = 1; porém, essa tentativa também se mostrou infrutífera e produziu resultados com baixo nível de acordo com o resultado obtido pelo DBSCAN no conjunto de dados completo.

Portanto, mesmo sendo possível a adoção dos outros métodos amostrais em conjunto com o DBSCAN, exige-se o conhecimento dos parâmetros a serem utilizados na amostra. Como não foi encontrado nenhuma forma de realizar essa sintonia na literatura, os demais métodos amostrais foram excluídos da segunda etapa experimental.

Esse problema não ocorre na amostra produzida pelos os quatro métodos selecionados, permitindo a execução do DBSCAN na amostra com parâmetros de fáceis estimativa. O I-DBSCAN e o Líder, executam o DBSCAN na amostra com os mesmos parâmetros, ϵ e minPts, usados pelo DBSCAN no conjunto de dados completo. Para os algoritmos amostrais *Rough*-DBSCAN e *Rough**-DBSCAN, o DBSCAN é executado na amostra com o mesmo valor de ϵ e minPts = 1. A relação completa dos métodos de amostragem executados para cada algoritmo de análise de agrupamento é exibido na tabela 2.

Todos os métodos foram implementados na linguagem JAVA e foram executados em uma máquina Intel(R) Core(TM) i7-4500U CPU @ 1.80GHz com 8 GB de memória física, utilizando um sistema operacional GNU/LINUX com *kernel* versão 4.10.0-28. Nas seções seguintes serão descritas cada uma das etapas experimentais.

Tabela 2 – Métodos de amostragem tendenciosa por algoritmo de análise de agrupamento.

Algoritmo de amostragem	Seção	k-médias	DBSCAN
Aleatório Uniforme		✓	
Líder	3.1.1		✓
k-médias++	3.1.2	✓	
FFT	3.1.3	✓	
Bagging	3.1.4	✓	
DBS	3.2.1	✓	
VGDBS	3.2.2	✓	
DENDIS	3.3.1	✓	
Rough-DBSCAN	3.3.2		✓
Algoritmo Genético	4	✓	
Rough*-DBSCAN	5.2		✓
I-DBSCAN	5.3		✓

6.1 Resultados dos métodos de amostragem tendenciosa para o k-médias

Nos experimentos para medir o nível de concordância dos resultados do k-médias, executado nas amostras, com o resultado obtido no conjunto de dados completo, foram selecionados 15 conjuntos de dados de domínio público[1]. A cardinalidade dos conjuntos de dados selecionados variam de 150 a 1.025.010; além disso, possuem um número variado de características e classes.

Os conjuntos de dados selecionados nesse experimento, juntamente com os valores do parâmetro k, são listados na tabela 3. Com exceção dos conjuntos de dados Abalone e Cadata, os valores escolhidos para k correspondem ao número de classes existentes no conjunto de dados. Para esses dois (Abalone e Cadata), devido serem destinados para problemas de regressão, o valor $k = 5$ foi escolhido arbitrariamente. Além disso, o k-médias executou com a inicialização proposta em Arthur e Vassilvitskii (2007) e com um limite de iterações 100 no laço principal do algoritmo.

Devido a aleatoriedade do algoritmo k-médias e, de todos os métodos amostrais, cada método foi executado dez vezes. Além disso, para eliminar uma possível inicialização inadequada do k-médias, quando executado sobre a amostra, as execuções utilizaram as mesmas médias e valores para o parâmetro k que no conjunto de dados completo.

[1] Disponíveis no repositório da LIBSVM: https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

Tabela 3 – Lista dos conjuntos de dados utilizados nos experimentos computacionais com o método de agrupamento k-médias.

Conjunto de dados	#elementos	#dimensões	k
Iris	150	4	3
German Numer	1.000	24	2
Abalone	4.177	8	5
Mushrooms	8.124	112	2
Pendigits	10.992	10	10
Letter	20.000	26	26
Cadata	20.640	8	5
Shuttle	58.000	7	7
Sensorless	58.509	48	11
Mnist	70.000	780	10
SensIT (acoustic)	98.528	50	3
SensIT (seismic)	98.528	50	3
Skin nonskin	245.057	3	2
Covtype	581.012	54	7
Poker	1.025.010	10	10

Para ilustrar, na primeira execução do k-médias no conjunto de dados completo a inicialização do algoritmo utilizou as médias M_1; na segunda execução, utilizou as médias M_2 e assim por diante. Em seguida, os métodos de amostragem foram executados dez vezes sobre o conjunto de dados completo, produzindo dez amostras cada. O passo seguinte foi a execução do k-médias utilizando a mesma inicialização do k-médias, ou seja, as médias M_1 na primeira execução, M_2 para a segunda; até M_{10} na última amostra. Após a execução do k-médias na amostra, o restante dos elementos – existentes no conjunto de dados – são rotulados na partição cuja média seja mais próxima. O resultado obtido pela execução do algoritmo amostral de análise de agrupamento na primeira amostra foi comparado com a primeira execução do k-médias no conjunto de dados completo. O mesmo procedimento foi realizado para as outras nove amostras.

Esse procedimento de utilizar a mesma inicialização – embora seja impossível de ser aplicado em um cenário real, devido ao desconhecimento das médias utilizadas pelo algoritmo de agrupamento no conjunto de dados completo – foi adotado para diminuir a aleatoriedade presente nos experimentos. O k-médias, os métodos de amostragem tendenciosa e o k-médias executado na amostra, apresentam uma componente estocástica significativa. Na prática, o que os experimentos medem é se o k-médias, utilizando as mesmas condições iniciais, produz resultados em acordo com a execução do algoritmo no conjunto de dados completo.

Sem o uso desse artifício, alguns resultados preliminares apontaram um baixo nível de concordância entre as soluções, em razão do pareamento incorreto dos resultados. Por exemplo, quando comparados os dez resultados obtidos para o k-médias com os dez resultados obtidos através de um método amostral qualquer, aconteceu do primeiro resultado de um se assemelhar com o segundo do outro. Isso ocorre devido a variação na etapa de inicialização do algoritmo k-médias. Isto é, quando se aferiu o ARI das duas primeiras soluções, o nível de acordo foi baixo. Para evitar esse problema, foi utilizada a mesma inicialização no k-médias – tanto na execução sobre a amostra quanto sobre o conjunto de dados completo.

Em relação aos parâmetros dos algoritmos de amostragem, todos os algoritmos tiveram seus parâmetros escolhidos para que a cardinalidade da amostra fosse aproximadamente do mesmo tamanho dos outros. O algoritmo aleatório uniforme, o k-médias++, o FFT, o DBS, o VGDBS e o algoritmo genético, recebem o tamanho amostral como entrada do algoritmo. Todavia, no DENDIS e no *Bagging*, o tamanho amostral é resultado dos outros parâmetros dos algoritmos.

Os experimentos executaram com os seguintes tamanhos amostrais: 100, 250, 500 e 750 elementos. Contudo, em razão da quantidade reduzida de elementos no conjunto de dados Iris, somente o valor de 100 foi utilizado. Os demais parâmetros que não são relacionados com k, nem com o tamanho amostral, seguiram as orientações fornecidas pelos autores – quando disponível.

O algoritmo *Bagging* retirou n/k amostras de $40 + 2k$ elementos e executou o algoritmo de agrupamento PAM em cada uma, onde n corresponde à cardinalidade da amostra e k o número de partições do conjunto de dados. A execução do PAM na amostra identifica os k medoides e, a união dos medoides obtidos em todas as amostras gera uma amostra final composta de nk elementos; sendo que $nk \in \{100, 250, 500, 750\}$. O termo $40 + 2k$ vem do tamanho amostral sugerido pelo CLARA (KAUFMAN; ROUSSEEUW, 1990), que é uma versão amostral do algoritmo de análise de agrupamento PAM.

O DBS, além do tamanho amostral, exige que seja especificado o tamanho da tabela *hash*. Em virtude da falta de sugestão dos autores, o valor utilizado para esse parâmetro foi $10d$, onde d representa a dimensionalidade do conjuntos de dados. A utilização desse valor ocorreu devido ao número de células existentes em um reticulado d-dimensional; e,

devido este número de células ser relacionado diretamente com d, considerar o termo no cálculo é razoável. Além disso, o tamanho da célula utilizada foi de 10% do menor eixo do conjunto de dados.

Para o VGDBS os autores sugerem como estimar cada parâmetro do algoritmo. Essa sugestão foi adotada e o parâmetros que regulam o número de divisões para cada eixo foi de $\lceil \sqrt[d]{N} \rceil$, onde N representa a cardinalidade do conjunto de dados. Os autores sugerem o valor 0.5 para o segundo parâmetro do algoritmo, responsável por criar a tendência de amostrar regiões menos densas, para qualquer conjunto de dados.

Apesar do algoritmo de amostragem DENDIS conter um único parâmetro, chamado granularidade, este parâmetro é responsável indireto pela cardinalidade da amostra. Isto é, quanto maior a granularidade, mais espaçados serão os elementos amostrados; portanto, esse parâmetro foi ajustado para que o DENDIS reproduzisse o mesmo tamanho amostral que os outros algoritmos, ou seja, 100, 250, 500 e 750 da cardinalidade dos conjuntos de dados.

No algoritmo genético amostral, os parâmetros utilizados foram o mesmos descritos no capítulo 4. Os demais métodos de amostragem utilizam apenas o tamanho amostral como parâmetro, portanto, não houve necessidade de sintonia.

Os resultados podem ser vistos na tabelas 4 e 5, os valores exibidos tanto para os tempos de execução quanto para o ARI são as médias das dez execuções realizadas. A tabela 4 exibe os resultados obtidos pelos algoritmos que apresentaram os piores níveis de concordância em comparação com o k-médias executado no conjunto de dados completo e a tabela 5 mostra o resultado para os quatro melhores métodos amostrais utilizados.

Os valores em vermelho nas tabelas indicam um tempo de execução superior ao do k-médias executado no conjunto de dados completo. Para os melhores algoritmos (Tabela 5), em todos os casos a variação dos resultados foi pequena, com o ARI variando sempre menos que 0.015 (no pior caso) entre as diferentes execuções. Contudo, para algoritmos amostrais que apresentaram os piores desempenho, a amostragem aleatória uniforme e o DBS apresentaram uma variação alta. Para a amostragem aleatória uniforme a variação entre a melhor e a pior solução foi, em média, de 0.04 e para o DBS foi de 0.05.

Analisando os resultados da tabela 4, fica evidente a discrepância entre os tempos de execuções obtido tanto pelo k-médias++ quanto para o FFT em relação aos outros

Tabela 4 – Resultados experimentais para os métodos amostrais: aleatório uniforme, *k*-médias++, FFT e DBS.

		k-médias	Uniforme		*k*-médias++		FFT		DBS	
Conjunto	Amostra	Tempo	ARI	Tempo	ARI	Tempo	ARI	Tempo	ARI	Tempo
Iris	100	0.011	0.92	0.067	0.95	0.12	0.95	0.11	0.90	0.011
German Numer	100	0.032	0.90	0.004	0.95	1.20	0.95	1.22	0.91	0.033
	250	0.032	0.95	0.006	0.97	7.69	0.96	7.52	0.95	0.040
	500	0.032	0.96	0.010	0.97	33.02	0.96	31.90	0.95	0.041
	750	0.032	0.97	0.036	0.97	75.20	0.97	74.47	0.95	0.044
Abalone	100	0.213	0.86	0.001	0.87	1.90	0.87	1.86	0.86	0.017
	250	0.213	0.87	0.003	0.87	12.01	0.87	11.93	0.88	0.019
	500	0.213	0.89	0.005	0.89	50.10	0.90	51.87	0.89	0.035
	750	0.213	0.90	0.006	0.90	140.30	0.91	139.27	0.90	0.035
Mushrooms	100	0.343	0.92	0.002	0.94	40.28	0.94	38.33	0.93	0.062
	250	0.343	0.92	0.006	0.95	251.81	0.95	249.76	0.93	0.078
	500	0.343	0.94	0.022	0.95	1204.49	0.95	1102.10	0.95	0.095
	750	0.343	0.95	0.036	0.96	3484.28	0.95	3350.96	0.95	0.121
Pendigits	100	0.592	0.87	0.001	0.85	1.82	0.89	1.79	0.85	0.172
	250	0.592	0.87	0.006	0.85	11.20	0.90	11.23	0.87	0.204
	500	0.592	0.88	0.015	0.87	54.37	0.90	56.18	0.87	0.256
	750	0.592	0.88	0.029	0.87	142.22	0.91	157.32	0.88	0.307
Letter	100	1.62	0.80	0.001	0.81	4.06	0.84	3.89	0.81	0.205
	250	1.62	0.82	0.013	0.84	24.45	0.85	23.23	0.81	0.421
	500	1.62	0.83	0.024	0.85	115.04	0.85	120.98	0.82	0.693
	750	1.62	0.83	0.041	0.85	320.01	0.86	332.32	0.83	0.903
Cadata	100	0.819	0.81	0.001	0.83	11.97	0.83	10.65	0.80	0.029
	250	0.819	0.83	0.002	0.83	65.27	0.83	62.56	0.82	0.027
	500	0.819	0.85	0.010	0.84	329.70	0.84	322.812	0.87	0.033
	750	0.819	0.86	0.007	0.84	922.44	0.84	931.875	0.87	0.055
Shuttle	100	1.60	0.90	0.001	0.87	7.01	0.83	6.79	0.91	0.113
	250	1.60	0.92	0.005	0.87	41.82	0.85	42.48	0.91	0.253
	500	1.60	0.92	0.027	0.87	220.30	0.86	212.43	0.92	0.779
	750	1.60	0.93	0.043	0.88	594.09	0.86	594.82	0.92	1.040
Sensorless	100	19.903	0.79	0.158	0.79	112.28	0.80	108.47	0.75	0.128
	250	19.903	0.80	1.065	0.82	606.84	0.83	590.48	0.76	0.192
	500	19.903	0.80	0.894	0.82	3271.05	0.83	3151.40	0.76	0.205
	750	19.903	0.81	1.846	–	7200+	–	7200+	0.76	0.230
Mnist	100	520.127	0.72	3.307	–	7200+	–	7200+	0.73	46.59
	250	520.127	0.73	14.711	–	7200+	–	7200+	0.75	55.48
	500	520.127	0.74	20.066	–	7200+	–	7200+	0.77	90.61
	750	520.127	0.74	33.476	–	7200+	–	7200+	0.77	97.42
Acoustic	100	3.94	0.85	0.003	0.82	196.2	0.82	193.0	0.83	0.147
	250	3.94	0.86	0.009	0.85	1248.4	0.85	1206.53	0.86	0.152
	500	3.94	0.87	0.010	0.85	6184.2	0.85	6031.68	0.88	0.174
	750	3.94	0.87	0.024	–	7200+	–	7200+	0.88	0.191
Seismic	100	6.798	0.84	0.004	0.80	180.6	0.80	172.7	0.81	0.157
	250	6.798	0.85	0.005	0.82	1105.0	0.82	1056.55	0.85	0.159
	500	6.798	0.85	0.014	0.82	5180.5	0.82	5197.78	0.85	0.182
	750	6.798	0.86	0.109	–	7200+	–	7200+	0.86	0.751
Skin nonskin	100	0.622	0.95	0.007	0.97	55.3	0.97	51.9	0.95	0.116
	250	0.622	0.96	0.007	0.99	358.37	0.99	324.79	0.96	0.116
	500	0.622	0.96	0.008	0.99	1750.3	0.99	1423.96	0.96	0.123
	750	0.622	0.96	0.008	0.99	4799.23	0.99	4247.11	0.96	0.131
Covtype	100	183.823	0.84	0.021	0.82	1539.4	0.85	1485.41	0.82	0.976
	250	183.823	0.84	0.030	–	7200+	–	7200+	0.83	0.996
	500	183.823	0.85	0.055	–	7200+	–	7200+	0.83	1.048
	750	183.823	0.85	0.193	–	7200+	–	7200+	0.84	1.062
Poker	100	2.846	0.86	0.002	0.86	13.46	0.87	12.79	0.86	0.018
	250	2.846	0.87	0.005	0.87	62.74	0.87	65.98	0.87	0.023
	500	2.846	0.87	0.018	0.87	352.90	0.88	349.93	0.88	0.037
	750	2.846	0.88	0.032	0.88	909.85	0.88	919.85	0.88	0.042

Tabela 5 – Resultados experimentais para os métodos amostrais: *Bagging*, VGDBS, DENDIS e o algoritmo genético.

		k-médias	*Bagging*		VGDBS		DENDIS		Genético	
Conjunto	Amostra	Tempo	ARI	Tempo	ARI	Tempo	ARI	Tempo	ARI	Tempo
Iris	100	0.011	**0.97**	0.02	0.92	0.017	0.95	0.013	0.95	0.109
German Numer	100	0.032	**0.96**	0.05	0.90	0.036	0.94	0.047	0.95	0.043
	250	0.032	0.96	0.08	0.93	0.063	**0.97**	0.056	0.96	0.067
	500	0.032	**0.98**	0.10	0.96	0.044	**0.98**	0.056	**0.98**	0.113
	750	0.032	**1.0**	0.11	0.97	0.050	0.99	0.057	0.98	0.115
Abalone	100	0.213	**0.90**	0.008	0.86	0.043	**0.90**	0.043	**0.90**	0.095
	250	0.213	0.90	0.03	0.88	0.030	**0.91**	0.051	0.90	0.125
	500	0.213	0.91	0.06	0.89	0.038	**0.92**	0.075	0.90	0.256
	750	0.213	0.92	0.08	0.90	0.044	**0.93**	0.126	0.91	0.416
Mushrooms	100	0.343	0.92	0.06	0.93	0.243	**0.95**	0.126	0.92	0.262
	250	0.343	0.93	0.15	0.93	0.267	**0.95**	0.159	0.92	0.326
	500	0.343	0.93	0.24	0.94	0.282	**0.96**	0.195	0.93	0.341
	750	0.343	0.93	0.49	0.95	0.289	**0.96**	0.204	0.93	0.402
Pendigits	100	0.592	0.92	0.01	0.88	0.180	0.92	0.275	**0.94**	0.308
	250	0.592	0.92	0.12	0.88	0.246	**0.94**	0.352	**0.94**	0.402
	500	0.592	0.93	0.27	0.88	0.295	**0.95**	0.471	**0.95**	0.485
	750	0.592	0.93	0.55	0.89	0.405	**0.95**	0.529	**0.95**	0.557
Letter	100	1.62	0.88	0.05	0.82	0.268	**0.90**	0.387	0.87	0.519
	250	1.62	0.90	0.23	0.82	0.461	**0.91**	0.529	0.90	0.703
	500	1.62	**0.91**	0.51	0.83	0.822	**0.91**	0.837	**0.91**	0.890
	750	1.62	0.91	1.15	0.83	1.050	**0.92**	1.209	0.91	1.103
Cadata	100	0.819	0.82	0.01	0.81	0.082	**0.90**	0.203	0.82	0.314
	250	0.819	0.82	0.04	0.83	0.084	**0.91**	0.259	0.83	0.427
	500	0.819	0.82	0.09	0.85	0.106	**0.91**	0.304	0.83	0.541
	750	0.819	0.83	0.12	0.85	0.118	**0.92**	0.322	0.83	0.661
Shuttle	100	1.60	**0.92**	0.07	**0.92**	0.080	0.89	0.167	**0.92**	0.268
	250	1.60	**0.94**	0.32	0.92	0.193	0.89	0.282	0.93	0.346
	500	1.60	**0.94**	0.63	0.93	0.482	0.90	0.690	0.93	0.591
	750	1.60	**0.95**	1.31	0.93	0.627	0.91	0.917	0.94	0.839
Sensorless	100	19.903	0.85	0.05	0.80	1.266	**0.86**	1.172	0.84	1.405
	250	19.903	0.85	0.18	0.80	1.269	**0.86**	1.384	0.85	1.738
	500	19.903	0.85	0.34	0.81	1.323	**0.87**	1.693	0.85	1.959
	750	19.903	0.85	0.50	0.81	1.430	**0.87**	1.997	0.85	2.553
Mnist	100	520.127	0.80	18.91	0.75	61.30	**0.83**	47.799	0.80	53.51
	250	520.127	0.81	58.69	0.75	73.22	**0.83**	49.535	0.80	101.73
	500	520.127	0.81	120.14	0.78	101.39	**0.84**	69.409	0.81	146.74
	750	520.127	0.81	202.67	0.78	120.38	**0.84**	80.093	0.81	191.41
Acoustic	100	3.94	**0.90**	0.17	0.87	2.173	0.89	1.142	0.88	1.203
	250	3.94	**0.91**	0.3	0.87	2.182	0.90	1.345	0.90	1.495
	500	3.94	**0.92**	0.59	0.88	2.229	0.90	2.084	0.90	2.299
	750	3.94	**0.93**	0.89	0.89	2.271	0.92	2.251	0.91	2.501
Seismic	100	6.798	0.88	0.09	0.83	2.231	**0.90**	0.160	0.86	0.394
	250	6.798	0.88	0.32	0.85	2.310	**0.90**	0.290	0.86	0.485
	500	6.798	0.88	0.60	0.88	2.316	**0.92**	0.440	0.87	0.603
	750	6.798	0.88	0.82	0.89	2.364	**0.92**	0.710	0.87	0.912
Skin nonskin	100	0.622	**0.99**	0.46	0.95	0.150	**0.99**	0.210	**0.99**	0.396
	250	0.622	**0.99**	1.13	0.95	0.154	**0.99**	0.410	**0.99**	0.454
	500	0.622	0.99	2.04	0.96	0.165	**1.0**	0.600	0.99	0.730
	750	0.622	0.99	3.06	0.96	0.207	**1.0**	0.740	0.99	0.968
Covtype	100	183.823	0.87	1.07	0.85	1.093	**0.90**	1.030	0.88	1.359
	250	183.823	0.88	1.05	0.85	1.080	**0.90**	1.307	0.88	1.604
	500	183.823	0.88	1.76	0.86	1.502	**0.90**	1.859	0.88	2.195
	750	183.823	0.88	2.67	0.87	1.650	**0.90**	2.047	0.88	2.443
Poker	100	2.846	0.86	0.01	0.86	0.055	0.86	0.105	**0.90**	0.158
	250	2.846	0.86	0.04	0.87	0.058	0.86	0.160	**0.90**	0.181
	500	2.846	0.87	0.09	0.89	0.069	0.86	0.236	**0.91**	0.259
	750	2.846	0.87	0.13	0.89	0.095	0.87	0.301	**0.91**	0.340

algoritmos. Ambos os métodos, apresentaram um tempo de execução muito elevado e demandaram inúmeras varreduras no conjunto de dados para construir a amostra. Ao se relembrar do funcionamento do FFT e do k-médias++, pode-se ver que, para cada elemento amostrado é necessário realizar uma nova varredura no conjunto de dados; e, à medida que o conjunto de dados aumenta tanto em cardinalidade quanto em dimensionalidade, o custo computacional para a execução dos métodos aumenta consideravelmente.

Para alguns casos, como o conjunto de dados Sensorless, Mnist, SensIT (acoustic), SensIT (Seismic) e Covtype, a execução do FFT e k-médias++ foi interrompida quando o tempo de execução excedeu duas horas. Ou seja, muito além do necessário para que o k-médias execute no conjunto de dados completo. Esse resultado diverge do resultado apresentado em Ros e Guillaume (2016a). A única explicação plausível para essa discrepância é que, nos experimentos realizados em Ros e Guillaume (2016a), os autores utilizaram um tamanho amostral reduzido. Em razão da complexidade computacional quadrática dos métodos, reduzir o tamanho amostral pela metade implica em uma execução quatro vezes mais rápida.

Em Bahmani et al. (2012), os autores propõem uma etapa de inicialização para o k-médias com melhor escalabilidade. Nesse trabalho, os autores mostram os problemas de escalabilidade da inicialização do k-médias++ (ARTHUR; VASSILVITSKII, 2007). Além disso, a inicialização proposta é de possível paralelização. Talvez, essa abordagem seja uma alternativa viável para sanar o problema em relação ao tempo apresentado pelo k-médias++ e FFT, quando houver a necessidade de selecionar uma maior quantidade de elementos.

Em virtude dos tempos de execução apresentados pelo k-médias++ e FFT, ambos os métodos foram excluídos das análises subsequentes pois a finalidade dos métodos amostrais estudados é reduzir o problema de escalabilidade dos métodos de análise de agrupamento. Portanto, não há motivo para se utilizar um algoritmo que demande um maior esforço computacional do que o próprio algoritmo executado no conjunto de dados completo.

Ainda em relação ao tempo de execução, os métodos (com exceção do k-médias++ e FFT) apresentaram um tempo de execução pequeno em relação ao do k-médias. Contudo, nos conjuntos de dados de menor cardinalidade, o ganho em desempenho obtido

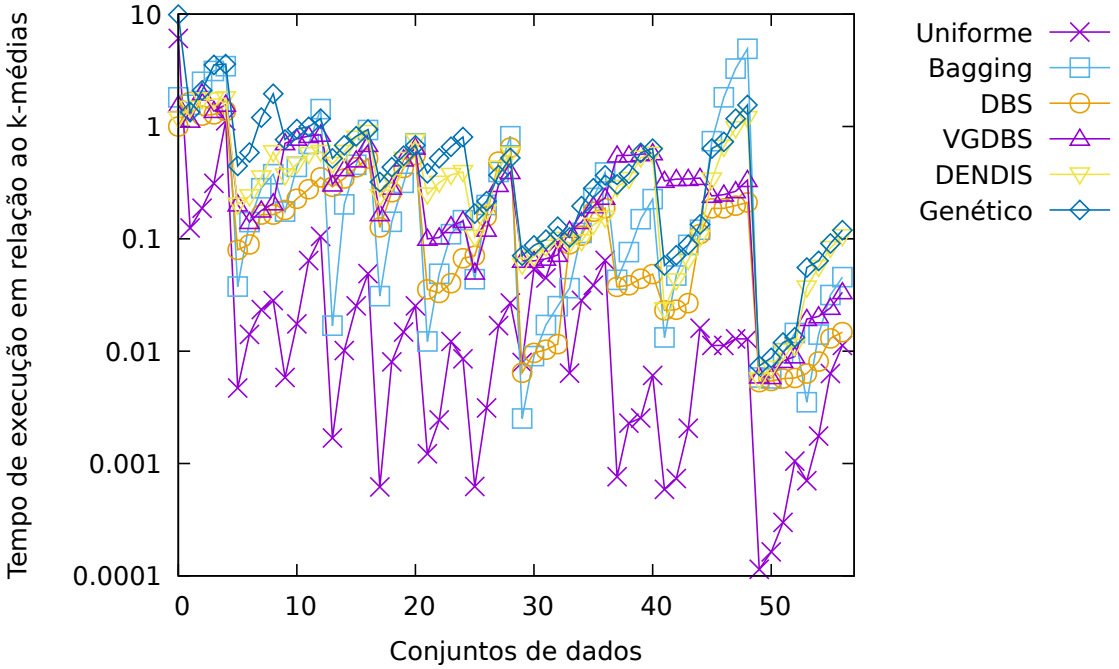


Figura 13 – Razão entre o tempo de execução dos métodos de amostragem pelo tempo de execução do k-médias.

pelo processo amostral não compensa o custo da execução do algoritmos de amostragem. Para o conjunto de dados Iris, todos os métodos – inclusive o método aleatório uniforme – apresentaram tempos de execução maiores do que o algoritmo de análise de agrupamento k-médias.

A figura 13 exibe a razão dos tempos obtidos pelos algoritmos amostrais, nos conjuntos de dados, pelo tempo de execução do k-médias no conjunto de dados completo. O eixo-x corresponde a um sequenciamento dos conjuntos de dados, obedecendo a mesma ordem em que os resultados são apresentados nas tabelas 4 e 5. O valor 0 corresponde à execução no conjunto de dados Iris, e o valor 56 corresponde à execução no conjunto de dados Poker com o tamanho amostral de 750.

Analisando a figura 13, fica evidente o ganho em tempo de execução proporcionado pelos algoritmos amostrais nos maiores conjuntos de dados. Inicialmente – para os menores conjuntos de dados – os métodos apresentaram um tempo de execução aproximadamente 10 vezes maior que a execução do k-médias no conjunto de dados completo. Todavia, para os maiores conjuntos de dados – representado na figura a partir do rótulo 50 – a execução foi de 10 a 100 vezes mais rápida. Além disso, a tendência decrescente apresentada na figura indica que tais técnicas são ainda mais vantajosas em conjuntos de dados ainda

maiores.

Nos casos em que os algoritmos apresentaram um tempo computacional acima do demandado pelo k-médias, os resultados indicam que essas técnicas possuem um custo fixo elevado. Entretanto, essas técnicas possuem uma melhor escalabilidade que o k-médias. Em outras palavras, o ganho em desempenho obtido pelo uso do método amostral apenas supera o tempo, para a execução do mesmo, em conjuntos de dados relativamente grandes.

O único resultado inesperado, em relação ao tempo de execução, foi o obtido no conjunto de dados Skin nonskin. Neste caso, alguns métodos tiveram uma execução mais lenta que o k-médias. Isso ocorreu devido ao k-médias ter convergido após pouquíssimas iterações no conjunto de dados completo.

Em relação ao nível de concordância entre os resultados obtidos nas amostras e o resultado obtido pelo k-médias no conjunto de dados completo, vemos que os dois piores algoritmos foram o DBS e a amostragem aleatória uniforme. O DBS obteve uma concordância média de 0.864, enquanto que o método de amostragem aleatória uniforme obteve 0.867 – melhor que o DBS. Esse resultado do DBS, talvez se origine das colisões na tabela *hash*, ou devido ao número de células no reticulado, ou até mesmo em razão da seleção de elementos em regiões de baixa densidade (ruído). Apesar disso, esse resultado está em concordância com comportamento visto em Ros e Guillaume (2016a).

Ignorando-se o k-médias++ e o FFT (por apresentarem um tempo de execução muito alto), e o aleatório uniforme e o DBS, por terem apresentado os piores resultados, pode-se observar que os outros métodos, no geral, apresentaram resultados de boa qualidade.

É importante frisar que o nível de concordância obtido foi elevado – por todos os métodos amostrais – e se deve, em grande parte, à utilização dos mesmo parâmetros de inicialização no algoritmo de análise de agrupamento; isto é, inicializando o k-médias de maneira aleatória na amostra, o nível de concordância diminui. Porém – como explicado anteriormente – o experimento foi realizado desse modo para evitar a busca de qual resultado da amostra se parece com qual execução do k-médias no conjunto de dados completo.

A figura 14 mostra as posições médias de cada método em relação ao nível de concordância com o resultado obtido pelo k-médias no conjunto de dados completo. Na

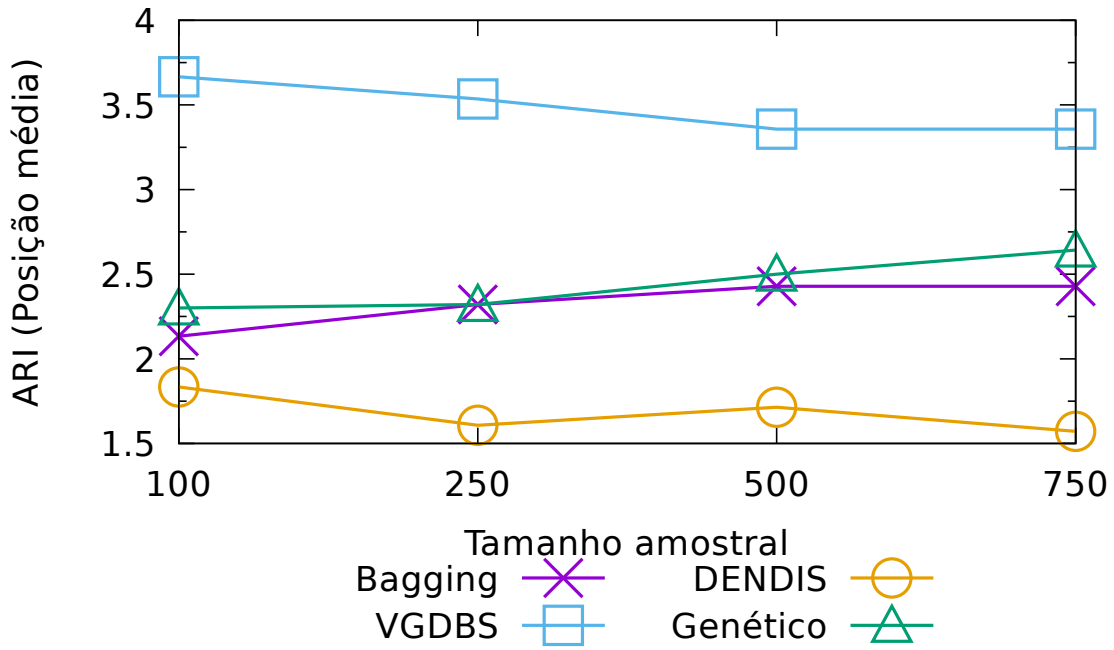


Figura 14 – Posição média dos algoritmos na métrica ARI, comparando o resultado obtido com o k-médias no conjunto de dados completo.

figura, pode-se visualizar que o VGDBS obteve as piores posições médias em relação a métrica adotada; e essa posição, se manteve praticamente constante, independente do tamanho amostral. O algoritmo genético amostral proposto e o *Bagging* apresentaram praticamente o mesmo desempenho. Entretanto, ambos foram superados pelo DENDIS.

O teste de Wilcoxon pareado (WILCOXON, 1945) foi executado sobre os resultados obtidos para verificar se a diferença observada é suficiente para alegar, de fato, a diferença entre os métodos (considerando o nível de confiança de 95%). O teste identificou uma diferença entre todos os métodos em relação ao VDGBS, para todos os tamanhos amostrais. O VGDBS apresentou os piores resultados (Figura 14) em relação aos outros três métodos analisados. Além desse caso, apenas houve evidência suficiente para diferenciar o DENDIS do *Bagging* (p-valor de 0.0488) para o tamanho amostral de 750.

Pelos resultados mostrados nas tabelas 4 e 5, além da figura 14, é evidente que o DENDIS apresentou os melhores resultados. O teste estatístico não apontou uma diferença significativa entre ele e os demais métodos por pouco. Muito provavelmente, essa diferença seria identificada caso os experimentos computacionais fossem realizados utilizando um número maior de conjuntos de dados. Portanto, é razoável assumir que o DENDIS foi o melhor método dessa etapa experimental e, empatados em segundo lugar, o algoritmo

genético amostral e o *Bagging*.

6.2 Resultados dos métodos de amostragem tendenciosa para o DBSCAN

Para avaliar a qualidade do resultado obtido através da métrica adotada (ARI) é necessário conhecer tanto a rotulação a ser avaliada quanto o resultado do DBSCAN no conjunto de dados completo. Esse último parâmetro é particularmente de difícil obtenção devido aos problemas de escalabilidade do próprio DBSCAN; portanto, a capacidade de execução do DBSCAN nos conjuntos de dados completo foi o fator limitante em relação ao tamanho dos conjuntos de dados selecionados.

Na tentativa de contornar esse problema, a etapa experimental do DBSCAN foi dividida em duas partes. A primeira, contendo 12 conjuntos de dados de domínio público[2] de pequeno a médio porte, onde a execução do DBSCAN no conjunto de dados completo foi viável. Neste caso, o nível de acordo obtido pelos métodos foi aferido utilizando a solução obtida pelo DBSCAN nos referidos conjuntos de dados.

A segunda parte experimental, foi realizada com oito conjuntos de dados de maior cardinalidade, gerados de maneira sintética. Nesses conjuntos de dados a execução do DBSCAN não se fez necessária, pois foram gerados de modo que o resultado obtido pelo DBSCAN fosse previsível. Para esses conjuntos de dados, a qualidade dos resultados obtidos foi aferida utilizando a previsão do resultado do DBSCAN.

É importante relatar que o único método utilizado nesta etapa experimental que possui alguma aleatoriedade é o I-DBSCAN, sendo que este apresentou consistentemente os mesmo resultados entre diferentes execuções. Portanto, os métodos foram executados uma única vez para cada conjunto de dados e configuração de parâmetros. Os conjuntos de dados, suas características e parâmetros, utilizados pelo DBSCAN no conjunto de dados completo, podem ser visualizados nas tabelas 6 e 7.

É importante lembrar que todos métodos de amostragem utilizados constroem a amostra para que seja possível a utilização do algoritmo DBSCAN utilizando os mesmos valores para os parâmetros ϵ e minPts. Os parâmetros do algoritmo de agrupamento

[2] Disponíveis no repositório da LIBSVM: https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

Tabela 6 – Conjunto de dados reais (LIBSVM).

Conjunto de dados	#elementos	#dimensões	ϵ	minPts
Iris	150	4	0.3	4
German Numer	1000	24	0.8	3
Abalone	4177	8	0.2	3
Mushrooms	8124	112	2.5	4
Pendigits	10992	10	40	4
Letter	20000	26	0.5	8
Cadata	20640	8	200	8
Shuttle	58000	7	0.03	20
Sensorless	58509	48	0.3	20
SensIT (acoustic)	98528	50	0.5	5
SensIT (seismic)	98528	50	0.4	5
Skin nonskin	245057	3	60	10

Tabela 7 – Conjunto de dados sintéticos.

Conjunto de dados	#elementos	#dimensões	ϵ	minPts
S1	1000000	5	1	2
S2	1000000	10	1	2
S3	2500000	5	1	3
S4	2500000	10	1	3
S5	5000000	5	1	4
S6	5000000	10	1	4
S7	10000000	5	1	5
S8	10000000	10	1	5

também são passados para todos os métodos de amostragem, com exceção do Líder que utiliza apenas um parâmetro relacionado com o valor de ϵ, para construir a amostra.

Portanto, não existe uma relevância muito grande nos valores escolhidos para ϵ e minPts exibidos nas tabelas. Tais valores foram arbitrariamente escolhidos para que o número de partições detectadas pelo DBSCAN fosse aproximadamente 10, variando de 3 partições para conjunto de dados Iris até 18 para o SensIT (acoustic).

Para os conjuntos de dados sintéticos (Tabela 7), o DBSCAN nunca foi executado; porém, para que os resultados fossem previsíveis, os mesmos foram construídos utilizando os parâmetros ϵ e minPts exibidos na tabela. Todos os conjuntos sintéticos foram criados exatamente com 10 partições; além disso, possuem a cardinalidade desbalanceada – variando de 1% a 19%, com incremento de 2%, dos elementos totais. Da menor partição, que contem 1% dos elementos, 5% dos elementos foram reservados para a geração de dados ruidosos.

A construção dos conjuntos de dados sintéticos é relativamente simples. Para criar cada partição, o seguinte processo é realizado: primeiro, cria-se um elemento considerado denso pelo DBSCAN, isto é feito utilizando o parâmetro minPts (Tabela 7). Depois de criado minPts elementos em uma região aleatória do espaço, o algoritmo executa um laço para adicionar novos elementos densos alcançáveis a partir dos existentes. Esse processo se repete até que a partição atinja a cardinalidade especificada.

A adição de novos elementos densos na partição foi implementada em três etapas. A primeira etapa, sorteia um elemento denso da partição que servirá como referência; isto é, certamente estará a uma distância ϵ do novo elemento, portanto, passará no critério de alcançabilidade do DBSCAN.

O passo seguinte é sortear uma coordenada para o novo elemento (denso) que será adicionado. Essa coordenada está a uma distância ϵ do elemento de referência. Em outras palavras, a coordenada do novo elemento é determinada pela soma de um vetor unitário com direção aleatória à coordenada do elemento de referência.

A terceira etapa garante que o elemento adicionado também seja denso. Para isso, minPts-2 elementos são criados nas proximidades dele. Esses elementos extras estão a uma distância $0.1 \cdot \epsilon$ do novo elemento. Após a execução das três etapas, o laço repete para a inclusão de outros elementos no conjunto de dados.

O processo descrito se repete para cada uma das 10 partições do conjunto de dados. Entretanto, antes de construir cada nova partição uma nova região do espaço é sorteada, garantindo assim que este nova partição esteja distante das partições existentes. Além disso, o ruído é inserido no conjunto de dados de modo aleatório uniforme após a criação de todas as partições.

Existe uma pequena chance de duas partições serem unidas através de um padrão ruidoso. Com exceção desse caso, o processo descrito garante que as partições criadas sejam identificadas corretamente pelo DBSCAN utilizando os mesmos parâmetros, ϵ e minPts, usados na criação dos conjuntos de dados.

Além disso, fica claro que $\epsilon = 1$ implica em uma perda de generalidade; afinal, o parâmetro atua apenas como um fator de escala nos conjuntos de dados sintéticos. Por exemplo, se fosse utilizado $\epsilon = 10$ os elementos estariam mais espaçados uns dos outros, contudo, o DBSCAN iria compensar esse espaçamento utilizando uma distância maior na

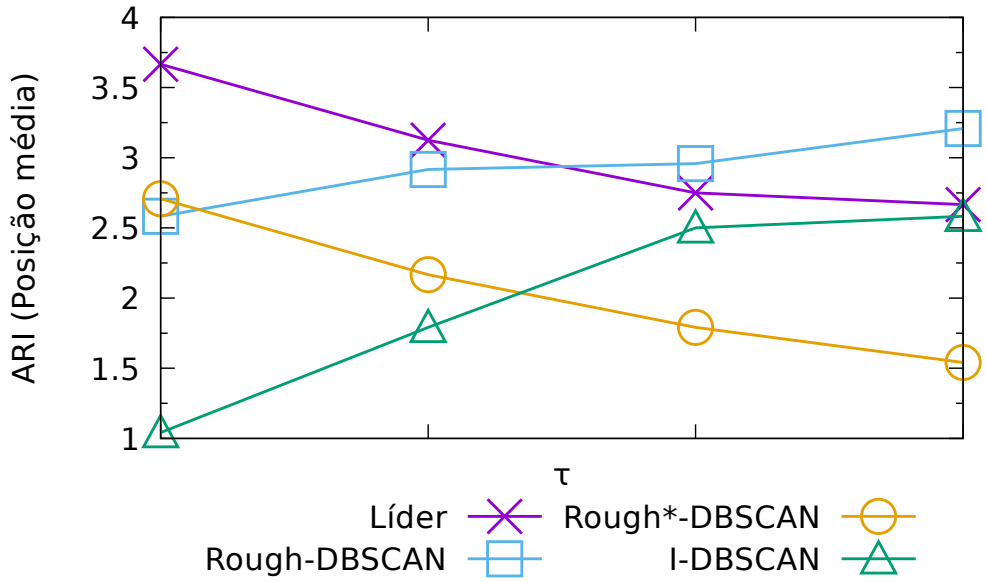


Figura 15 – Posição média dos algoritmos na métrica ARI, comparando o resultado com o resultado obtido pelo DBSCAN no conjunto de dados completo.

etapa de verificação da densidade – cancelando completamente o efeito produzido devido ao maior espaçamento.

Em relação aos parâmetros necessários para os métodos de amostragem tendenciosa, com exceção do I-DBSCAN, os outros três métodos necessitam do parâmetro τ. Esse parâmetro é responsável pelo tamanho amostral gerado, e consequentemente, impacta no nível de concordância entre os resultados. Como não existe nenhum trabalho da literatura sobre como realizar a sintonia desse parâmetro, cada algoritmo foi executado com quatro configurações desse parâmetro.

A primeira configuração usa o valor próximo de ϵ usado no DBSCAN e gera uma amostra pequena. Cada uma das outras configurações subsequentes aumenta progressivamente o tamanho da amostra, através da diminuição do parâmetro τ, para a obtenção de aproximações melhores.

Os resultados obtidos na etapa experimental foram bem comportados, com poucos casos inesperados. Na primeira etapa experimental – utilizando os conjuntos de dados de domínio público – ficou evidente o comportamento geral de cada método. As figuras 15 e 16 sumarizam os resultados obtidos nos conjuntos de dados reais. A figura 15 exibe as posições médias dos algoritmos de amostragem em relação à concordância com o DBSCAN executado no conjunto de dados completo. A figura 16 exibe as posições médias dos

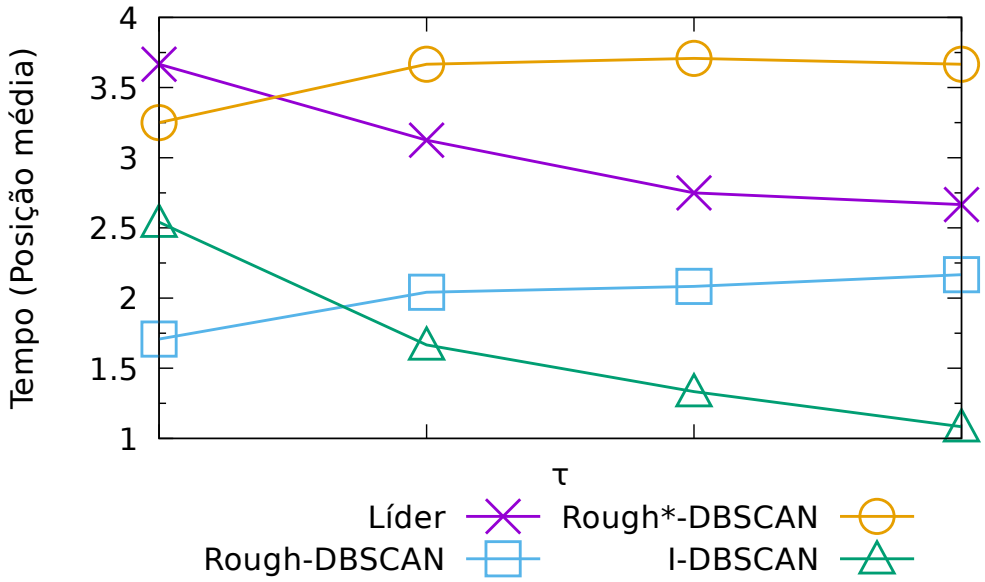


Figura 16 – Posição média em relação ao tempo de execução dos métodos de amostragem.

algoritmos em relação ao tempo de execução.

Em ambos os casos (Figuras 15 e 16), o eixo-x representa as quatro variações dos parâmetros τ utilizadas pelos métodos. O primeiro valor τ (a esquerda do eixo) corresponde a aplicação dos métodos com o menor tamanho amostral, o segundo valor corresponde a uma amostra um pouco maior; e assim por diante. É importante ressaltar que o I-DBSCAN não depende de τ e sempre retornou os mesmos resultados; contudo, a posição média obtida pelo algoritmo variou devido aos outros métodos apresentarem melhores resultados à medida que o tamanho da amostra aumenta.

A figura 15 mostra o comportamento geral dos métodos em relação ao nível de concordância com os resultados obtidos no conjunto de dados completo. Pode-se ver que inicialmente o I-DBSCAN apresenta os melhores resultados. Nos 12 conjuntos de dados utilizados na primeira etapa o I-DBSCAN apresentou 11 melhores resultados e 1 empate. Contudo, à medida que o parâmetro τ diminui – acarretando em um tamanho amostral maior para os demais métodos – o resultado do I-DBSCAN é superado pelo *Rough**-DBSCAN em quase todos os casos e, ocasionalmente, pelos outros também. Todavia, o deslocamento da posição média obtida pelo I-DBSCAN é causado, principalmente, pela rápida aproximação do *Rough**-DBSCAN devido ao aumento no tamanho amostral.

Outra característica evidente que pode ser observada é que, tanto o Líder quanto o *Rough*-DBSCAN apresentaram – em média – os piores resultados; além disso, inverteram

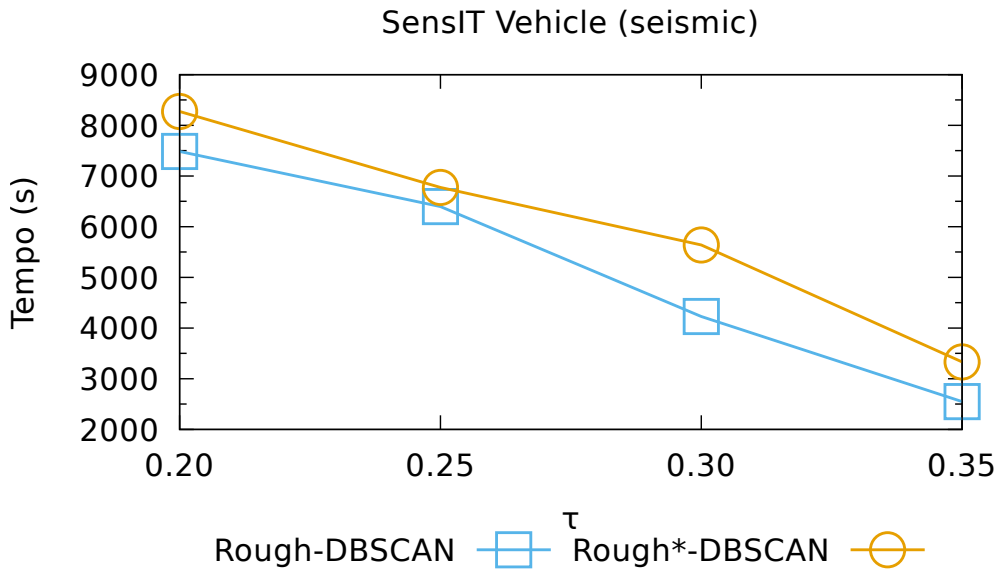


Figura 17 – Tempo de execução de dois dos algoritmos de amostragem tendenciosa para o conjunto de dados SensIT Vehicle (seismic).

de posição a medida que o parâmetro τ diminuiu. Com exceção de quando o *Rough**-DBSCAN utilizou o menor tamanho amostral, em nenhum momento apresentaram resultados (em média) melhores que os algoritmos propostos. O mesmo comportamento, isto é, a inversão da posição média no gráfico, também ocorreu entre os algoritmos propostos. Os resultados completos encontram-se na tabela 8. Os resultados marcados em vermelho apresentam uma aproximação pior, ou igual, à aleatória ou um tempo de execução acima do requerido pelo algoritmo de análise de agrupamento executar no conjunto de dados completo.

O gráfico da posição média em relação ao tempo de execução (Figura 16) mostra o *Rough**-DBSCAN como o algoritmo mais lento, apresentando uma curva similar ao *Rough*-DBSCAN, porém deslocada para cima. Os dois algoritmos são muito similares, sendo que o *Rough**-DBSCAN executa uma segunda varredura no conjunto de dados devido ao Líder*.

Para ilustrar, a figura 17 exibe tempo de execução para os dois algoritmos no conjunto de dados que demandou o maior esforço computacional dos métodos. O exemplo ilustrado na figura – mostrando a similaridade das curvas do tempo de execução – se repetiu para todos os conjuntos de dados.

Analisando tempo de execução apresentado pelos métodos amostrais na tabela

Tabela 8 – Resultados comparativos entre os métodos amostrais para os conjuntos de dados reais.

		DBSCAN	Líder		Rough-DBSCAN		Rough*-DBSCAN		I-DBSCAN	
Conjunto	τ	Tempo	ARI	Tempo	ARI	Tempo	ARI	Tempo	ARI	Tempo
Iris	0.16	0.028	0.86178	**0.009**	0.86306	0.014	**0.91424**	0.015	**0.91424**	0.011
	0.14	0.028	0.96684	0.012	0.94138	0.018	**0.99200**	0.020	0.91424	**0.011**
	0.12	0.028	**0.99200**	0.014	0.94138	0.019	**0.99200**	0.020	0.91424	**0.011**
	0.10	0.028	**1.0**	0.015	0.94138	0.020	**1.0**	0.022	0.91424	**0.011**
German Numer	0.7	0.4	*0.0*	*0.56*	0.44951	0.32	0.85841	**0.30**	**1.0**	0.32
	0.6	0.4	0.77354	*0.56*	0.46564	0.34	0.93377	**0.32**	**1.0**	**0.32**
	0.5	0.4	0.85889	*0.57*	0.48128	0.37	**1.0**	0.35	**1.0**	**0.32**
	0.4	0.4	0.85889	*0.59*	0.48128	0.4	**1.0**	0.36	**1.0**	**0.32**
Abalone	0.15	2.0	0.89168	0.4	0.89498	0.4	0.82760	0.4	**0.98212**	**0.3**
	0.1	2.0	**0.99578**	0.7	0.99542	0.8	0.97705	0.9	0.98212	**0.3**
	0.075	2.0	**0.99878**	1.3	0.99834	1.5	**0.99878**	1.7	0.98212	**0.3**
	0.05	2.0	**1.0**	*2.5*	0.99953	*2.7*	**1.0**	*2.8*	0.98212	**0.3**
Mushrooms	2.25	97.8	0.90707	2.2	0.90744	1.9	0.90723	4.3	**0.91018**	**1.8**
	2	97.8	0.90707	2.1	0.90744	2.1	0.90967	29.5	**0.91018**	**1.8**
	1.75	97.8	0.91016	17.0	**0.91018**	16.8	0.90967	29.5	**0.91018**	**1.8**
	1.5	97.8	0.91016	19.0	**0.91018**	16.6	0.90967	29.4	**0.91018**	**1.8**
Pendigits	30	25.6	0.57317	6.4	0.90735	5.9	0.87531	8.2	**0.97226**	**5.5**
	25	25.6	0.97226	14.6	0.97434	12.6	**0.98108**	16.4	0.97226	**5.5**
	20	25.6	0.99369	*30.0*	0.97997	25.5	**0.99382**	*30.0*	0.97226	**5.5**
	15	25.6	0.99731	*45.5*	0.98396	*43.8*	**0.99873**	*45.0*	0.97226	**5.5**
Letter	0.4	83.2	*-0.07850*	31.0	0.48805	**25.9**	0.51722	35.9	**0.78703**	36.5
	0.35	83.2	0.42781	55.9	0.60939	47.6	0.74672	62.1	**0.78703**	**36.5**
	0.3	83.2	0.61758	75.5	0.65552	63.2	**0.82271**	77.5	0.78703	**36.5**
	0.25	83.2	0.88270	*118.5*	0.71589	*104.2*	**0.92820**	*114.6*	0.78703	**36.5**
Cadata	190	105.5	*0.0*	2.8	0.00176	**2.6**	0.00086	4.0	**0.95008**	5.3
	170	105.5	*-0.02694*	3.4	0.32304	**3.0**	0.23303	4.8	**0.95008**	5.3
	150	105.5	*-0.02829*	4.3	0.80000	**3.8**	0.75335	5.8	**0.95008**	5.3
	120	105.5	0.61885	6.3	0.88185	5.4	0.93695	8.0	**0.95008**	**5.3**
Shuttle	0.025	791.6	*0.0*	4.5	0.20430	**4.2**	0.17485	5.9	**0.82221**	8.5
	0.02	791.6	*0.0*	7.0	0.75313	**6.8**	0.74034	9.2	**0.82221**	8.5
	0.015	791.6	*0.0*	10.0	**0.85545**	9.3	0.85470	13.0	0.82221	**8.5**
	0.01	791.6	0.05655	18.9	0.99861	15.3	**0.99992**	21.4	0.82221	**8.5**
Sensorless	0.2	7379	0.58890	53.9	0.79940	**32.6**	0.81089	104.7	**0.99891**	34.0
	0.175	7379	0.73631	68.8	0.76622	55.4	0.77719	146.7	**0.99891**	**34.0**
	0.15	7379	0.81341	122.1	0.80119	107.2	0.81217	264.8	**0.99891**	**34.0**
	0.125	7379	0.95633	218.0	0.93949	208.5	0.95797	484.5	**0.99891**	**34.0**
Acoustic	0.4	31640	0.87897	1400.8	0.77247	1265.0	0.86352	1775.5	**0.93737**	**1033.4**
	0.35	31640	0.95006	2104.5	0.79622	1879.4	**0.95174**	2530.8	0.93737	**1033.4**
	0.3	31640	0.98351	2990.6	0.79910	2734.2	**0.98590**	3461.8	0.93737	**1033.4**
	0.25	31640	0.99215	4278.0	0.79992	3953.5	**0.99487**	5155.7	0.93737	**1033.4**
Seismic	0.35	10626	0.68412	3012.8	0.70192	**2405.2**	0.56338	3296.8	**0.96184**	3507.2
	0.3	10626	0.95775	4749.0	0.77549	4232.6	0.93424	4970.6	**0.96184**	**3507.2**
	0.25	10626	**0.99210**	7126.5	0.79154	5788.6	0.98769	6986.2	0.96184	**3507.2**
	0.2	10626	0.99210	7643.2	0.79323	7440.9	**0.99821**	7766.6	0.96184	**3507.2**
Skin nonskin	28	92073	0.53831	3.8	0.71763	**2.5**	0.77121	5.8	**0.84658**	6.3
	26	92073	0.69381	4.1	0.78998	**2.6**	0.82629	6.3	**0.84658**	6.3
	24	92073	0.75912	5.3	0.83719	**3.0**	**0.84728**	7.3	0.84658	6.3
	22	92073	0.83193	6.4	0.83719	**3.7**	**0.84728**	8.6	0.84658	6.3

8, com exceção do I-DBSCAN, todos os métodos apresentaram resultados com o tempo computacional acima do requerido pelo algoritmo de agrupamento no conjunto de dados completo. Esses resultados aconteceram para os casos em que o conjunto de dados era pequeno – Abalone, Pendigits e Letter – e o tamanho amostral era grande. A única exceção a essa regra aconteceu com o algoritmo amostral baseado no Líder para o conjunto de dados German Numer; exibidindo um tempo sempre superior ao do DBSCAN.

Todavia, para os conjuntos de dados de maior porte, e também para os sintéticos (Tabela 9), todos os métodos sempre apresentaram ganho em tempo de execução em relação ao DBSCAN executado no conjunto de dados completo. O I-DBSCAN se destacou dos demais algoritmos em relação ao tempo de execução e apresentou, em todos os casos, um tempo de execução menor que o DBSCAN.

Nos experimentos constatou-se que o tempo de execução é proporcional ao tamanho amostral. Apesar da obviedade da afirmação, o tamanho amostral não é um parâmetro que os métodos recebem. E observamos que esse valor não é determinado em função da cardinalidade do conjunto de dados, e sim da distribuição dos elementos. Isso porque todos os três métodos usam alguma versão do algoritmo Líder para extrair a amostra, e o mesmo não se importa se existem 5 ou 1000 seguidores para cada líder.

Para ilustrar um caso extremo: se houver dois conjuntos de elementos bem compactos de modo que cada grupo teria somente um líder, e suficientemente separados, a amostra seria composta por dois elementos apenas; mesmo que o conjunto de dados original tenha milhões de elementos em cada partição. Por outro lado, conjuntos de dados de maior esparsidade tendem a gerar amostras maiores devido a existência de muitos líderes. Esse fato ocorreu mesmo para conjunto de dados relativamente pequenos.

Isso explica a variação no tempo de execução obtido pelos métodos. O tempo obtido na Letter e Shuttle são comparáveis. O mesmo é verdade para os conjuntos de dados Skin nonskin e Pendigits, sendo que o primeiro conjunto de dados contém aproximadamente 25 vezes mais elementos que a último. Contudo, tanto no primeiro como no segundo caso, a execução nas duas bases tiveram amostras com aproximadamente a mesma quantidade de elementos.

Outro comportamento indesejado, observado em todos os métodos amostrais, com exceção do Líder, foi a criação de regiões onde o DBSCAN não consegue alcançar a

partir dos elementos da partição, acarretando em um número maior de partições. Em outras palavras, caso exista uma partição coesa – na qual todos os elementos deveriam ser alcançáveis – devido ao processo de amostragem, algum elemento que serviria de "ponte" pode ter ficado de fora da amostra. Quando isso ocorre, a execução do DBSCAN na amostra identificará um número maior partições em relação ao algoritmo executado no conjunto de dados completo.

Esse fenômeno é raro no centro das partições mas se constatou que ocorre frequentemente nas periferias das partições. Na prática, é comum a grupos de grande cardinalidade apresentarem vários grupos menores em torno deles. Esses pequenas partições são compostas por uma quantidade pequena de elementos e quase não afetam a métrica.

A figura 18 mostra a quantidade de partições detectadas a medida que o parâmetro τ varia. Em todos os conjuntos dessa etapa experimental o *Rough**-DBSCAN e o *Rough*-DBSCAN identificaram um número de elevado de partições na amostra devido a existência de partições com pouquíssimos elementos ao redor das partições "verdadeiras". Quando o valor de τ é pequeno (amostra grande), o *Rough**-DBSCAN apresenta uma maior robustez a esse problema em relação ao *Rough*-DBSCAN, que continuou a identificar um número grande de partições mesmo quando τ é pequeno. Em alguns exemplos esse problema também afetou o I-DBSCAN. Contudo, de forma menos acentuada.

Para o algoritmo amostral baseado no Líder, o problema inverso ocorre. Inicialmente a amostra gerada é muito pouco densa e uma grande quantidade de elementos é classificada como ruído. À medida que o tamanho amostral aumenta – devido ao decréscimo de τ – os elementos passam a se situar em regiões de maior densidade, e consequentemente, passam a ser rotulados corretamente, acarretando um número maior de partições detectadas. É importante ressaltar que o número de partições identificadas pelo Líder em todos os casos foi menor (ou igual) ao resultado obtido pelo DBSCAN. O Líder identificou poucos grupos (entre 40% a 80% do valor esperado) quando a amostra era pequena e se aproximando do resultado esperado a medida que o tamanho amostral aumentava.

Os experimentos nos conjuntos de dados sintéticos reproduziu o comportamento observado nos reais (Tabela 9). A maior diferença observada entre os dois experimentos são os tempos de execução dos métodos, que não apresentaram uma variação muito grande de um conjunto de dados para outro. Na realidade, os tempos computacionais apresentam

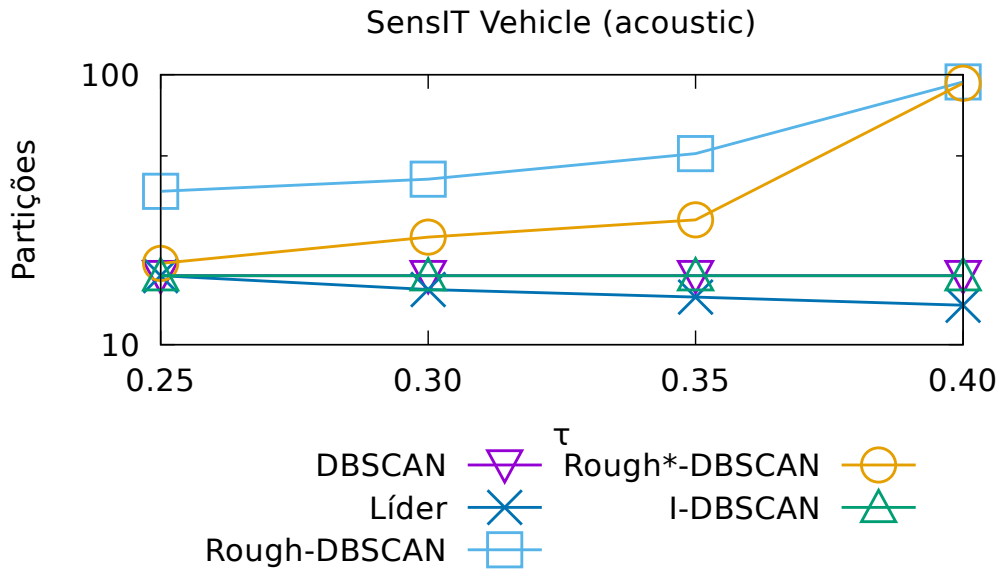


Figura 18 – Número de partições detectadas em função do parametro τ no conjunto de dados SensIT Vehicle (acoustic).

uma tendência crescente, proporcional à cardinalidade do conjunto de dados.

Nos experimentos com os conjunto de dados reais não foi encontrado uma relação direta entre o tempo de execução do algoritmo amostral e a cardinalidade do conjunto de dados. Contudo, essa relação parece verdadeira nos resultados obtidos nos conjuntos de dados sintéticos. Na tabela 9 pode-se ver uma forte tendência entre a cardinalidade do conjunto de dados ($|S1| \leq |S2| \leq \cdots \leq |S8|$) e o tempo de execução. Provavelmente esse comportamento ocorreu em razão de todos os conjuntos de dados sintéticos terem sido gerados através do mesmo procedimento e, provavelmente apresentarem uma distribuição similar dos elementos. O tamanho amostral também seguiu a mesma tendência, porém, com um crescimento mais lento. O maior conjunto de dados possui dez vezes mais elementos que o menor; entretanto, o tamanho da amostra foi, aproximadamente, 2.5 vezes maior que o do menor conjunto de dados.

Além do comportamento no tempo de execução dos algoritmos descritos acima, o restante dos resultados apresentaram os mesmos padrões e problemas vistos nos experimentos com os conjuntos de dados reais.

Com exceção do Líder, os outros três métodos apresentaram, em diversos casos, um número de partições acima do que deveria – acima de 10. Enquanto que o Líder também detectou um número pequeno de partições, principalmente para os valores altos

Tabela 9 – Resultados comparativos entre os métodos amostrais para os conjuntos de dados sintéticos.

		Líder		Rough-DBSCAN		Rough*-DBSCAN		I-DBSCAN	
Conjunto	τ	ARI	Tempo	ARI	Tempo	ARI	Tempo	ARI	Tempo
S1	0.8	0.74331	210.4	0.82739	**197.2**	0.91283	232.8	**0.97371**	204.6
	0.7	0.84119	243.7	0.89281	237.2	0.94289	294.3	**0.97371**	**204.6**
	0.6	0.96027	299.1	0.95812	284.6	**0.99282**	353.4	0.97371	**204.6**
	0.5	0.99100	324.8	0.99213	301.4	**1.0**	389.3	0.97371	**204.6**
S2	0.8	0.72050	**261.0**	0.87240	280.9	0.90291	302.5	**0.94372**	264.2
	0.7	0.81623	321.5	0.89937	292.9	0.93395	322.7	**0.94372**	**264.2**
	0.6	**0.95928**	372.0	0.93821	325.3	0.94010	349.3	0.94372	**264.2**
	0.5	0.95928	381.1	0.94022	395.0	**0.97866**	440.2	0.94372	**264.2**
S3	0.8	0.77681	364.4	0.90274	351.0	0.86215	413.1	**0.96321**	**328.5**
	0.7	0.84386	402.2	0.93028	385.1	0.91387	442.5	**0.96321**	**328.5**
	0.6	0.93491	461.7	**0.97125**	429.4	0.97025	493.3	0.96321	**328.5**
	0.5	0.95013	495.7	0.99288	452.7	**0.99923**	531.1	0.96321	**328.5**
S4	0.8	0.85392	320.6	0.85392	**318.8**	0.84482	352.2	**0.92281**	322.4
	0.7	0.87004	364.2	0.87105	347.4	0.87291	399.7	**0.92281**	**322.4**
	0.6	0.91828	415.6	0.91728	393.2	**0.93954**	463.9	0.92281	**322.4**
	0.5	0.92090	471.5	0.92090	453.5	**0.94185**	500.4	0.92281	**322.4**
S5	0.8	0.88103	703.0	0.89327	688.2	0.90283	761.0	**1.0**	**596.3**
	0.7	0.95912	792.0	0.94924	753.0	0.96445	876.6	**1.0**	**596.3**
	0.6	0.99133	834.4	0.96421	830.4	**1.0**	942.4	**1.0**	**596.3**
	0.5	**1.0**	900.3	0.97382	893.9	**1.0**	951.9	**1.0**	**596.3**
S6	0.8	0.83914	**532.2**	0.82374	580.7	0.81388	630.4	**0.89274**	550.2
	0.7	0.85372	640.3	0.84728	626.3	0.83471	703.6	**0.89274**	**550.2**
	0.6	0.90015	711.8	0.88371	682.6	**0.90283**	743.4	0.89274	**550.2**
	0.5	0.95510	783.5	0.88371	737.1	**0.94927**	798.2	0.89274	**550.2**
S7	0.8	0.74951	799.3	0.82392	753.6	0.84285	938.4	**0.95497**	**625.3**
	0.7	0.83125	921.3	0.84929	894.1	0.87030	1028.7	**0.95497**	**625.3**
	0.6	0.90140	1104.4	0.87472	1029.3	0.89370	1389.4	**0.95497**	**625.3**
	0.5	0.91447	1372.8	0.88480	1285.4	0.91836	1608.5	**0.95497**	**625.3**
S8	0.8	0.71942	**548.0**	0.82937	552.3	0.87284	703.0	**0.97486**	883.9
	0.7	0.79822	760.3	0.86371	**754.5**	0.94829	916.8	**0.97486**	883.9
	0.6	0.96230	904.8	0.92755	894.4	**0.97582**	1125.8	0.97486	**883.9**
	0.5	0.97588	980.1	0.95728	950.7	**0.98392**	1340.3	0.97486	**883.9**

de τ. O I-DBSCAN apresentou os melhores resultados quando os outros métodos estavam com o tamanho amostral relativamente baixo e o *Rough**-DBSCAN se aproximou mais rapidamente do resultado do DBSCAN no conjunto de dados completo que os outros algoritmos. Em resumo, todos os comentários realizados a respeito do experimento nos conjuntos de dados reais foram reproduzidos para os conjuntos de dados sintéticos.

Para verificar se existe significância estatística nos resultados obtidos foi realizado um teste de Wilcoxon pareado (WILCOXON, 1945) nos resultados obtidos pelos métodos amostrais tanto nos conjuntos de dados reais quanto em todos os conjuntos de dados (reais e sintéticos). A tabela 10 mostra o resultado do teste e os resultados replicam o comportamento observado na figura 15. As células apresentam os p-valor comparando os algoritmos amostrais em pares em cada um dos cenários analisados. Os p-valores apre-

Tabela 10 – Resultado do teste estatístico de Wilcoxon pareado aplicado sobre os resultados obtidos nos conjuntos de dados reais. O valor de cada célula representa o p-valor obtido. O valor entre parênteses em algumas células indica os casos em que não existia diferença e passou a ter caso fosse considerado também os resultados obtidos nos conjuntos de dados sintéticos. No cabeçalho, os rótulos L, R, R*, e I correspondem, respectivamente, aos métodos Líder, *Rough*-DBSCAN, *Rough**-DBSCAN e I-DBSCAN.

Teste	L vs R	L vs R*	L vs I	R vs R*	R vs I	R* vs I
1^o teste	**0.006836**	0.05225 (0.00486)	**0.0004883**	0.7334	**0.0004883**	**0.0009766**
2^o teste	0.5693	**0.01611**	0.06396 (0.0003948)	0.09229	**0.01221**	0.07715
3^o teste	0.8501	0.08296 (0.01333)	0.1514	**0.03418**	0.08296 (0.01491)	0.3505
4^o teste	0.3394	**0.008045**	0.8501	**0.001465**	0.23	0.05593 (0.01661)

sentados entre parenteses na tabela representam os casos em que a conclusão do teste em todos os conjuntos de dados (reais e sintéticos) divergiu do resultado obtido utilizando apenas os conjuntos de dados reais.

Para eliminar a dependência entre os resultados obtidos no mesmo conjunto de dados, a análise foi dividida em quatro testes. O primeiro foi realizado utilizando o resultado obtido no conjunto de dados com o maior valor de τ, o segundo utilizando o segundo maior valor, e assim por diante. O valor de 0.05 para o p-valor foi utilizado para determinar se há diferença entre dois métodos. Um valor abaixo desse limiar aponta diferença significativa entre eles.

No primeiro teste (utilizando um valor alto para τ), o I-DBSCAN apresentou os melhores resultados (Figura 15) e o resultado foi suficiente para que fosse apontado diferença em relação aos outros três outros métodos. Na mesma figura, podemos ver que o Líder obteve os piores resultados. Esse resultado foi confirmado no teste estatístico pois pode-se afirmar uma diferença significativa em relação aos algoritmos *Rough*-DBSCAN e I-DBSCAN (com p-valor de 0.0068 e 0.00048 respectivamente); no caso do *Rough**-DBSCAN a diferença só foi significativa com a inclusão dos resultados obtidos nos conjuntos de dados sintéticos ($p-\text{valor} = 0.0048$).

Do segundo teste em diante, não houve evidência suficiente nos resultados obtidos entre o Líder e o *Rough*-DBSCAN para apontar qualquer diferença entre os dois métodos. Porém, quando comparado o Líder com o *Rough**-DBSCAN – e, considerando os valores nos parênteses – os testes apontaram diferenças entre os métodos; reafirmando o desempenho superior apresentado pelo *Rough**-DBSCAN a medida que o tamanho amostral aumenta. Tal comportamento também pode ser visualizado na figura 15.

Nos dois últimos testes – com as maiores amostras – o *Rough**-DBSCAN apresen-

tou os melhores resultados e o teste estatístico apontou diferenças significativas entre os três métodos, com exceção do I-DBSCAN no terceiro teste. Também vale ressaltar que o melhor resultado obtido para o *Rough*-DBSCAN, em relação ao I-DBSCAN, foi igualar a qualidade dos resultados. Porém, o I-DBSCAN executou muito mais rapidamente e é um método livre de parâmetros.

Em síntese, o *Rough*-DBSCAN e o Líder, quando comparados ao I-DBSCAN, iniciam piores e, no máximo, se igualam em termos de qualidade dos resultados. Quando comparado com o Rough*-DBSCAN, o *Rough*-DBSCAN inicia com qualidade similar e é ultrapassado em termos de qualidade do resultado. Na comparação do Rough*-DBSCAN com o Líder, o primeiro possui vantagem em todos os casos.

A adição dos resultados obtidos nos conjuntos de dados sintéticos aos testes estatísticos não altera significativamente a conclusão, apenas evidencia mais rapidamente as diferenças percebidas nos experimentos. Por fim, todas as diferenças destacadas na tabela 10 entre os métodos propostos nesta tese, *Rough**-DBSCAN e I-DBSCAN, e os outros dois, Líder e *Rough*-DBSCAN, sempre apontaram os métodos propostos como mais vantajosos.

7 Conclusão

A análise de agrupamento é um conjunto de técnicas destinadas à identificação de grupos de elementos similares em um conjunto de dados, com aplicações nas mais variadas áreas da computação. Nesta tese, foi apresentado o funcionamento, pontos fortes e fracos, dos algoritmos k-médias e DBSCAN.

Dentre os pontos fracos dos algoritmos de análise de agrupamento estudados, o problema de escalabilidade foi o foco do estudo. Para lidar com esse problema, existem diversas alternativas, uma delas – objeto dessa tese – é a diminuição da cardinalidade do conjunto de dados através de métodos de amostragem tendenciosa. Nesta tese foram propostos três métodos de amostragem tendenciosa com essa finalidade: um algoritmo genético amostral, desenvolvido para o k-médias, e outros dois métodos de amostragem para o DBSCAN.

7.1 Contribuições

Os métodos propostos são inovadores em certos aspectos. O algoritmo genético amostral, descrito no capítulo 4, obtém um resultado parecido com o método proposto por Xiao et al. (2014), porém, a um custo computacional reduzido. Esse último, executa o k-médias no conjunto de dados completo para depois amostrar os elementos nas periferias das partições. Já o algoritmo genético amostral, consegue selecionar uma amostra de característica similar sem executar o k-médias no conjunto de dados completo; possuindo o custo computacional reduzido em relação ao algoritmo apresentado em Xiao et al. (2014).

Em termos da qualidade do resultado, a etapa experimental que compara qual algoritmo produz a melhor amostra para o algoritmo de análise de agrupamento k-médias – isto é, que mais se aproxima dos resultados do algoritmo quando executado no conjunto de dados completo – comparou o algoritmo genético amostral contra 7 outros algoritmos em 15 conjunto de dados de domínio público (disponíveis no repositório da LIBSVM). O DENDIS (ROS; GUILLAUME, 2016a) apresentou os melhores resultados, seguido pelo algoritmo genético amostral e *Bagging* (DOLNICAR; LEISCH, 2004) empatados em segundo lugar.

É importante ressaltar que tanto o DENDIS quanto o *Bagging* possuem uma maior tolerância à existência de padrões ruidosos no conjunto de dados e, possivelmente, esse foi o fator determinante para que os dois métodos obtivessem bons resultados. O *Bagging* forma sua amostra baseado em centroides/medoides de diversas execuções em pequenas amostras, esse comportamento diminui muito a chance de qualquer elemento ruidoso ser incluído na amostra. Já o DENDIS, realiza um pós-processamento para remover os elementos inseridos em regiões de baixa densidade na primeira etapa do algoritmo.

Para o DBSCAN, existem pouquíssimos métodos amostrais na literatura desenvolvidos para funcionar especificamente em conjunto com o DBSCAN. Um dos motivos é a dificuldade de construir uma amostra no qual os parâmetros ϵ e minPts do DBSCAN sejam conhecidos. O método amostral *Rough*-DBSCAN, proposto por Viswanath e Babu (2009), é um dos melhores – se não o melhor – algoritmo amostral construído para atuar em conjunto com o DBSCAN. A grande inovação desse método é a capacidade de gerar uma amostra que seja boa para o DBSCAN utilizando os parâmetros ϵ e minPts passados ao algoritmo amostral. No entanto, a proposta original apresenta um ponto negativo no modo em como estima a densidade de alguns elementos (os líderes). Essa estimativa, embora funcione muito bem, apresenta discrepâncias em relação ao cálculo da densidade realizado pelo DBSCAN no conjunto de dados completo.

Para contornar esse problema, uma das propostas apresentadas na tese foi a modificação do algoritmo Líder, denominado, Líder*, para tornar o processo de estimativa de densidade trivial. O método *Rough**-DBSCAN – também proposto nessa tese (Seção 5.2) – é essencialmente o algoritmo original de Viswanath e Babu (2009) utilizando o algoritmo Líder*, no lugar do Líder, em sua primeira etapa.

Na comparação experimental, com pouquíssimas exceções, o *Rough**-DBSCAN obteve resultados melhores que o *Rough*-DBSCAN. Porém, é importante ressaltar que que a nova proposta possui um tempo de execução um pouco maior que a proposta original. Esse custo computacional é devido a necessidade do Líder* executar uma segunda varredura no conjunto de dados enquanto a proposta original executa somente uma varredura.

Um ponto fraco tanto do método original (*Rough*-DBSCAN) quanto na modificação proposta (*Rough**-DBSCAN) é a necessidade de sintonia do parâmetro τ. Até a presente data não existe um método, ou estudo, para se estimar um valor correto para

o parâmetro. Apenas é conhecido um limite superior para o mesmo, a saber $\tau \leq \epsilon$, caso contrário os grupos (na amostra) seriam compostos apenas por um elemento.

Além do *Rough**-DBSCAN, outra contribuição da tese foi o método amostral I-DBSCAN (Seção 5.3). Esse método foi projetado para o DBSCAN utilizando apenas os parâmetros ϵ e minPts. Portanto, sem a necessidade de sintonia (de τ) uma vez que ϵ e minPts devem ser conhecidos de antemão; caso contrário, nem o DBSCAN poderia ser executado no conjunto de dados completo. Em outras palavras, se houver conhecimento necessário (dos parâmetros) para a execução do DBSCAN no conjunto de dados completo, o I-DBSCAN também pode ser utilizado. É importante ressaltar que o I-DBSCAN também utiliza o Líder* em sua primeira etapa para evitar os problemas na estimativa de densidade.

Na etapa experimental, o I-DBSCAN obteve, no geral, os melhores resultados, tanto em tempo de execução quanto no nível de concordância com o resultados obtidos pelo DBSCAN no conjunto de dados completo. Todavia, quando o tamanho amostral era suficientemente grande, o I-DBSCAN foi superado pelo *Rough**-DBSCAN em aproximadamente metade dos casos.

7.2 Trabalhos futuros

Os experimentos foram realizados para medir o nível de concordância entre as soluções obtidas através da amostra em relação ao obtido pelo k-médias e pelo DBSCAN no conjunto de dados completo. Os melhores métodos – *Bagging*, DENDIS, I-DBSCAN, *Rough**-DBSCAN, e *Rough*-DBSCAN – possuem mecanismos para evitar a seleção de elementos em regiões de baixa densidade. Entretanto, no algoritmo genético amostral não existe nenhuma etapa do algoritmo que desempenha essa função. Na realidade, o algoritmo tipicamente tenta inserir esses elementos na amostra, considerando que eles provavelmente estarão situados a uma distância grande dos centroides.

Dentre os métodos que não possuem tal mecanismo para descarte de ruído, o algoritmo genético apresentou os melhores resultados. Portanto, é provável que se uma etapa de pós-processamento fosse adicionada para descartar os elementos de baixa densidade – assim como no DENDIS – o nível de acordo entre o algoritmo genético amostral e o resultado obtido pelo k-médias no conjunto de dados completo seria ainda maior. Em trabalhos

futuros, essa modificação pode ser incorporada no algoritmo genético para melhorar seus resultados.

Embora o *Rough**-DBSCAN e o I-DBSCAN tenham apresentado os melhores resultados, eles ainda podem ser aprimorados. O *Rough**-DBSCAN utiliza o parâmetro τ e não há uma forma estabelecida para se escolher um valor para este parâmetro. Todavia, sabe-se que τ está diretamente relacionado com ϵ e com a esparsidade do conjunto de dados. Talvez seja possível implementar – em trabalhos futuros – uma rápida análise do conjunto de dados e determinar um valor apropriado para τ. Nos experimentos realizados neste trabalho, com a identificação do melhor valor ajustado deste parâmetro, o *Rough**-DBSCAN apresentou os melhores resultados. Portanto, se existir uma forma simples de determinar o valor ideal para esse parâmetro, a maior fraqueza do *Rough**-DBSCAN estaria resolvida.

Outro ponto de melhoria, que será investigado em trabalhos futuros, corresponde à diminuição da quantidade de computação necessária para a execução do Líder*. Essa modificação diminuirá o tempo de execução dos dois algoritmos propostos; afinal, ambos dependem da saída do Líder*. Isso pode ser feito através da inserção dos elementos que não se tornaram líderes, na primeira varredura do conjunto de dados, em uma estrutura de dados para indexá-los espacialmente. Estes índices podem ser utilizados para agilizar a segunda varredura – que é a etapa que demanda um maior esforço computacional – eliminando assim, uma quantidade grande de operações desnecessárias (cálculos de distância).

Apesar dos excelentes resultados obtidos pelo I-DBSCAN, foi identificado dois cenários nos quais o algoritmo retorna um número diferente de partições em relação ao resultado obtido pelo DBSCAN. O primeiro ocorre quando um grupo coeso é particionado em uma quantidade maior de grupos na amostra. Isso ocorre devido a escolha incorreta de elementos nas interseções para fazer parte da amostra. Apesar da amostragem com o FFT utilizada no I-DBSCAN minimizar a ocorrência desse problema, ela não o elimina. O segundo efeito indesejado é o inverso do primeiro, isto é, quando duas partições distintas – próximas umas das outras – são unidas devido a uma amostragem de elementos na borda de ambas; ou devido a ordem no qual os elementos se tornam líderes no Líder*. Esses dois problemas com I-DBSCAN serão estudados em trabalhos futuros. É importante ressaltar que o *Rough**-DBSCAN também sofre do mesmo problema. Porém, com o ajuste correto

do parâmetro τ o problema é suprimido quase completamente.

Por fim, estender a etapa experimental com novos conjuntos de dados e métodos dará uma confiança maior em relação aos resultados apresentados no capítulo 6. Um método bastante promissor e, recentemente publicado, é o *ProTraS* (*PRObabilistic TRAversing Sampling algorithm*) Ros e Guillaume (2018). Os resultados relatados pelos autores são bastante promissores, superando o resultado do DENDIS nos testes realizados. Além disso, os autores aplicaram o algoritmo tanto ao k-médias quanto para o DBSCAN. Em trabalhos futuros planeja-se comparar os métodos propostos nesta tese (Capítulos 4 e 5) em relação ao *ProTraS*.

Referências

AL-MAMORY, S. O. et al. Enhancing of dbscan based on sampling and density-based separation. *Iraqi Journal for Computers and Informatics ijci*, University Of Informatics Technology And Communications, v. 42, n. 1, p. 38–47, 2016. Citado na página 41.

ALOISE, D. et al. Np-hardness of euclidean sum-of-squares clustering. *Machine learning*, Springer, v. 75, n. 2, p. 245–248, 2009. Citado na página 28.

ANDONI, A.; INDYK, P. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In: IEEE. *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*. [S.l.], 2006. p. 459–468. Citado na página 16.

APPEL, A. P. *Métodos para o pré-processamento e mineração de grandes volumes de dados multidimensionais e redes complexas*. Tese (Doutorado) — Instituto de Ciências Matemáticas e de Computação, 2010. Citado 4 vezes nas páginas 19, 44, 58 e 59.

APPEL, A. P. et al. Biased box sampling-a density-biased sampling for clustering. In: ACM. *Proceedings of the 2007 ACM symposium on Applied computing*. [S.l.], 2007. p. 445–446. Citado 4 vezes nas páginas 18, 19, 45 e 59.

APPEL, A. P. et al. A density-biased sampling technique to improve cluster representativeness. In: SPRINGER. *European Conference on Principles of Data Mining and Knowledge Discovery*. [S.l.], 2007. p. 366–373. Citado 2 vezes nas páginas 19 e 59.

ARLIA, D.; COPPOLA, M. Experiments in parallel clustering with dbscan. In: SPRINGER. *European Conference on Parallel Processing*. [S.l.], 2001. p. 326–331. Citado 3 vezes nas páginas 16, 19 e 41.

ARTHUR, D.; VASSILVITSKII, S. How slow is the k-means method? In: ACM. *Proceedings of the twenty-second annual symposium on Computational geometry*. [S.l.], 2006. p. 144–153. Citado na página 30.

ARTHUR, D.; VASSILVITSKII, S. k-means++: The advantages of careful seeding. In: SOCIETY FOR INDUSTRIAL AND APPLIED MATHEMATICS. *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. [S.l.], 2007. p. 1027–1035. Citado 8 vezes nas páginas 29, 31, 44, 48, 49, 50, 95 e 101.

BACK, T. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. [S.l.]: Oxford university press, 1996. Citado na página 73.

BAHMANI, B. et al. Scalable k-means++. *Proceedings of the VLDB Endowment*, VLDB Endowment, v. 5, n. 7, p. 622–633, 2012. Citado 3 vezes nas páginas 29, 51 e 101.

BEJARANO, J. et al. *Sampling Within k-Means Algorithm to Cluster Large Datasets*. [S.l.], 2011. Citado 3 vezes nas páginas 17, 18 e 31.

BENTLEY, J. L. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, ACM, v. 18, n. 9, p. 509–517, 1975. Citado na página 40.

BERKHIN, P. et al. A survey of clustering data mining techniques. *Grouping multidimensional data*, Springer, v. 25, p. 71, 2006. Citado na página 15.

BORAH, B.; BHATTACHARYYA, D. An improved sampling-based dbscan for large spatial databases. In: IEEE. *Intelligent Sensing and Information Processing, 2004. Proceedings of International Conference on*. [S.l.], 2004. p. 92–96. Citado na página 41.

BREIMAN, L. Bagging predictors. *Machine learning*, Springer, v. 24, n. 2, p. 123–140, 1996. Citado na página 52.

BREIMAN, L. et al. *Isodata, a novel method of data analysis and classification*. [S.l.], 1965. Citado na página 16.

BREUNIG, M. M. et al. Data bubbles: Quality preserving performance boosting for hierarchical clustering. In: ACM. *ACM SIGMOD Record*. [S.l.], 2001. v. 30, n. 2, p. 79–90. Citado na página 17.

CHAKRAVARTY, S. Sample size determination for multinomial population. In: *National association for welfare research and statistics 39th annual workshop*. [S.l.: s.n.], 1999. Citado na página 77.

CHANDOLA, V.; BANERJEE, A.; KUMAR, V. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, ACM, v. 41, n. 3, p. 15, 2009. Citado na página 35.

CHÁVEZ, E. et al. Searching in metric spaces. *ACM computing surveys (CSUR)*, ACM, v. 33, n. 3, p. 273–321, 2001. Citado 3 vezes nas páginas 16, 19 e 41.

CHEN, S.; MULGREW, B.; GRANT, P. M. A clustering technique for digital communications channel equalization using radial basis function networks. *IEEE Transactions on neural networks*, IEEE, v. 4, n. 4, p. 570–590, 1993. Citado na página 15.

CHEN, X. et al. Apscan: A parameter free algorithm for clustering. *Pattern Recognition Letters*, Elsevier, v. 32, n. 7, p. 973–986, 2011. Citado na página 40.

CHEN, Y. et al. A fast clustering algorithm based on pruning unnecessary distance computations in dbscan for high-dimensional data. *Pattern Recognition*, Elsevier, 2018. Citado na página 40.

CORTES, C.; VAPNIK, V. Support-vector networks. *Machine learning*, Springer, v. 20, n. 3, p. 273–297, 1995. Citado na página 69.

DASGUPTA, S.; FREUND, Y. Random projection trees for vector quantization. *IEEE Transactions on Information Theory*, IEEE, v. 55, n. 7, p. 3229–3242, 2009. Citado na página 28.

DASH, M.; LIU, H. Feature selection for clustering. In: *Knowledge Discovery and Data Mining. Current Issues and New Applications*. [S.l.]: Springer, 2000. p. 110–121. Citado na página 30.

DATAR, M. et al. Locality-sensitive hashing scheme based on p-stable distributions. In: ACM. *Proceedings of the twentieth annual symposium on Computational geometry*. [S.l.], 2004. p. 253–262. Citado na página 41.

DEMPSTER, A. P.; LAIRD, N. M.; RUBIN, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, JSTOR, p. 1–38, 1977. Citado 2 vezes nas páginas 16 e 32.

DOLNICAR, S.; LEISCH, F. Segmenting markets by bagged clustering. *Australasian Marketing Journal (AMJ)*, Elsevier, v. 12, n. 1, p. 51–65, 2004. Citado 6 vezes nas páginas 21, 25, 44, 52, 54 e 119.

DUNN, J. C. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. Taylor & Francis, 1973. Citado na página 32.

EL-SONBATY, Y.; ISMAIL, M. A.; FAROUK, M. An efficient density based clustering algorithm for large databases. In: IEEE. *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*. [S.l.], 2004. p. 673–677. Citado na página 41.

ELKAN, C. Using the triangle inequality to accelerate k-means. In: *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*. [S.l.: s.n.], 2003. p. 147–153. Citado na página 31.

ESTER, M. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: *Kdd*. [S.l.: s.n.], 1996. v. 96, n. 34, p. 226–231. Citado 2 vezes nas páginas 16 e 35.

FAHIM, A. et al. An efficient enhanced k-means clustering algorithm. *Journal of Zhejiang University SCIENCE A*, Springer, v. 7, n. 10, p. 1626–1633, 2006. Citado na página 31.

FELDMAN, D.; FAULKNER, M.; KRAUSE, A. Scalable training of mixture models via coresets. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2011. p. 2142–2150. Citado na página 44.

FELDMAN, D.; LANGBERG, M. A unified framework for approximating and clustering data. In: ACM. *Proceedings of the forty-third annual ACM symposium on Theory of computing*. [S.l.], 2011. p. 569–578. Citado 2 vezes nas páginas 21 e 45.

FEO, T. A.; RESENDE, M. G. Greedy randomized adaptive search procedures. *Journal of global optimization*, Springer, v. 6, n. 2, p. 109–133, 1995. Citado na página 29.

FORGY, E. W. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *biometrics*, v. 21, p. 768–769, 1965. Citado na página 27.

FRANCO-LOPEZ, H.; EK, A. R.; BAUER, M. E. Estimation and mapping of forest stand density, volume, and cover type using the k-nearest neighbors method. *Remote sensing of environment*, Elsevier, v. 77, n. 3, p. 251–274, 2001. Citado 3 vezes nas páginas 18, 44 e 45.

GAREY, M.; JOHNSON, D.; WITSENHAUSEN, H. The complexity of the generalized lloyd-max problem (corresp.). *IEEE Transactions on Information Theory*, IEEE, v. 28, n. 2, p. 255–256, 1982. Citado na página 28.

GOLDBERG, D. E. et al. *Genetic algorithms in search optimization and machine learning*. [S.l.]: Addison-wesley Reading Menlo Park, 1989. Citado na página 72.

GUHA, S.; RASTOGI, R.; SHIM, K. Cure: an efficient clustering algorithm for large databases. In: ACM. *ACM Sigmod Record*. [S.l.], 1998. v. 27, n. 2, p. 73–84. Citado na página 15.

GUTTMAN, A. *R-trees: A dynamic index structure for spatial searching*. [S.l.]: ACM, 1984. Citado 2 vezes nas páginas 19 e 40.

GUYON, I.; ELISSEEFF, A. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, JMLR. org, v. 3, p. 1157–1182, 2003. Citado na página 30.

HAMERLY, G. Making k-means even faster. In: SIAM. *Proceedings of the 2010 SIAM international conference on data mining*. [S.l.], 2010. p. 130–140. Citado na página 31.

HAMERLY, G.; DRAKE, J. Accelerating lloyd's algorithm for k-means clustering. In: *Partitional clustering algorithms*. [S.l.]: Springer, 2015. p. 41–78. Citado na página 31.

HARALICK, R. M.; SHAPIRO, L. G. Image segmentation techniques. *Computer vision, graphics, and image processing*, Elsevier, v. 29, n. 1, p. 100–132, 1985. Citado na página 15.

HARTIGAN, J. A. *Clustering Algorithms*. 99th. ed. New York, NY, USA: John Wiley & Sons, Inc., 1975. ISBN 047135645X. Citado 2 vezes nas páginas 46 e 81.

HARTIGAN, J. A.; WONG, M. A. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, JSTOR, v. 28, n. 1, p. 100–108, 1979. Citado 2 vezes nas páginas 30 e 32.

HOLLAND, J. H. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. [S.l.]: U Michigan Press, 1975. Citado na página 72.

HOU, J.; GAO, H.; LI, X. Dsets-dbscan: a parameter-free clustering algorithm. *IEEE Transactions on Image Processing*, IEEE, v. 25, n. 7, p. 3182–3193, 2016. Citado na página 40.

HUANG, J. et al. Esc: An efficient synchronization-based clustering algorithm. *Knowledge-Based Systems*, Elsevier, v. 40, p. 111–122, 2013. Citado na página 61.

HUBERT, L.; ARABIE, P. Comparing partitions. *Journal of classification*, Springer, v. 2, n. 1, p. 193–218, 1985. Citado 4 vezes nas páginas 24, 66, 91 e 92.

INABA, M.; KATOH, N.; IMAI, H. Applications of weighted voronoi diagrams and randomization to variance-based k-clustering. In: ACM. *Proceedings of the tenth annual symposium on Computational geometry*. [S.l.], 1994. p. 332–339. Citado na página 28.

JAIN, A. K. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, Elsevier, v. 31, n. 8, p. 651–666, 2010. Citado na página 16.

KANUNGO, T. et al. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, IEEE, n. 7, p. 881–892, 2002. Citado na página 31.

KARAMI, A.; JOHANSSON, R. Choosing dbscan parameters automatically using differential evolution. *International Journal of Computer Applications*, Citeseer, v. 91, n. 7, 2014. Citado na página 39.

KARYPIS, G.; HAN, E.-H.; KUMAR, V. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, IEEE, v. 32, n. 8, p. 68–75, 1999. Citado na página 16.

KASSAMBARA, A. *Practical Guide to Cluster Analysis in R: Unsupervised Machine Learning*. [S.l.]: STHDA, 2017. Citado na página 36.

KAUFMAN, L.; ROUSSEEUW, P. J. Partitioning around medoids (program pam). *Finding groups in data: an introduction to cluster analysis*, Wiley Online Library, p. 68–125, 1990. Citado 3 vezes nas páginas 16, 53 e 97.

KERDPRASOP, K.; KERDPRASOP, N.; SATTAYATHAM, P. Density-biased clustering based on reservoir sampling. In: IEEE. *Database and Expert Systems Applications, 2005. Proceedings. Sixteenth International Workshop on*. [S.l.], 2005. p. 1122–1126. Citado 4 vezes nas páginas 19, 45, 58 e 59.

KHAN, K. et al. Dbscan: Past, present and future. In: IEEE. *Applications of Digital Information and Web Technologies (ICADIWT), 2014 Fifth International Conference on the*. [S.l.], 2014. p. 232–238. Citado na página 42.

KLEINBERG, J.; PAPADIMITRIOU, C.; RAGHAVAN, P. A microeconomic view of data mining. *Data mining and knowledge discovery*, Springer, v. 2, n. 4, p. 311–324, 1998. Citado na página 28.

KOLATCH, E. et al. Clustering algorithms for spatial databases: A survey. *PDF is available on the Web*, p. 1–22, 2001. Citado na página 16.

KOLLIOS, G. et al. Efficient biased sampling for approximate clustering and outlier detection in large data sets. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, v. 15, n. 5, p. 1170–1187, 2003. Citado 4 vezes nas páginas 18, 19, 23 e 44.

KOLLIOS, G. et al. An efficient approximation scheme for data mining tasks. In: IEEE. *Data Engineering, 2001. Proceedings. 17th International Conference on*. [S.l.], 2001. p. 453–462. Citado 3 vezes nas páginas 18, 19 e 44.

KRIEGEL, H.-P. et al. Density-based clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, Wiley Online Library, v. 1, n. 3, p. 231–240, 2011. Citado na página 40.

KUMAR, K. M.; REDDY, A. R. M. A fast dbscan clustering algorithm by accelerating neighbor searching using groups method. *Pattern Recognition*, Elsevier, v. 58, p. 39–48, 2016. Citado na página 40.

LAZAROV, G. D. D.; AVERBUCH, A. Smart-sample: An efficient algorithm for clustering large high-dimensional datasets. *Tel-Aviv University, Tel-Aviv*, v. 69978, 2009. Citado na página 31.

LING, R. F. *Cluster analysis algorithms for data reduction and classification of objects*. [S.l.]: Taylor & Francis, 1981. Citado na página 44.

LIU, P.; ZHOU, D.; WU, N. Vdbscan: varied density based spatial clustering of applications with noise. In: IEEE. *Service Systems and Service Management, 2007 International Conference on*. [S.l.], 2007. p. 1–4. Citado na página 40.

LLOYD, S. Least squares quantization in pcm. *IEEE transactions on information theory*, IEEE, v. 28, n. 2, p. 129–137, 1982. Citado 4 vezes nas páginas 16, 27, 28 e 30.

LUCHI, D. et al. A genetic algorithm approach for clustering large data sets. In: IEEE. *Tools with Artificial Intelligence (ICTAI), 2016 IEEE 28th International Conference on*. [S.l.], 2016. p. 570–576. Citado 6 vezes nas páginas 18, 21, 23, 69, 74 e 75.

LUCHI, D.; RODRIGUES, A. L.; VAREJÃO, F. M. Sampling approaches for applying dbscan to large datasets. *Pattern Recognition Letters*, Elsevier, v. 117, p. 90–96, 2019. Citado 4 vezes nas páginas 22, 23, 26 e 81.

LUCHI, D. et al. Genetic sampling k-means for clustering large data sets. In: *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*. [S.l.]: Springer, 2015. p. 691–698. Citado 5 vezes nas páginas 21, 23, 69, 74 e 75.

MACQUEEN, J. et al. Some methods for classification and analysis of multivariate observations. In: OAKLAND, CA, USA. *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. [S.l.], 1967. v. 1, n. 14, p. 281–297. Citado 4 vezes nas páginas 16, 27, 28 e 29.

MAHAJAN, M.; NIMBHORKAR, P.; VARADARAJAN, K. The planar k-means problem is np-hard. In: SPRINGER. *International Workshop on Algorithms and Computation*. [S.l.], 2009. p. 274–285. Citado na página 28.

MITRA, P.; MURTHY, C.; PAL, S. K. Density-based multiscale data condensation. *IEEE Transactions on pattern analysis and machine intelligence*, IEEE, v. 24, n. 6, p. 734–747, 2002. Citado 3 vezes nas páginas 18, 19 e 45.

NAING, L.; WINN, T.; RUSLI, B. Practical issues in calculating the sample size for prevalence studies. *Archives of orofacial Sciences*, School of Dental Sciences, Universiti Sains Malaysia, v. 1, p. 9–14, 2006. Citado na página 77.

NANOPOULOS, A.; MANOLOPOULOS, Y.; THEODORIDIS, Y. An efficient and effective algorithm for density biased sampling. In: ACM. *Proceedings of the eleventh international conference on Information and knowledge management*. [S.l.], 2002. p. 398–404. Citado 2 vezes nas páginas 19 e 58.

NANOPOULOS, A.; THEODORIDIS, Y.; MANOLOPOULOS, Y. C2p: clustering based on closest pairs. In: *VLDB*. [S.l.: s.n.], 2001. p. 331–340. Citado na página 17.

NANOPOULOS, A.; THEODORIDIS, Y.; MANOLOPOULOS, Y. Indexed-based density biased sampling for clustering applications. *Data & Knowledge Engineering*, Elsevier, v. 57, n. 1, p. 37–63, 2006. Citado 6 vezes nas páginas 18, 19, 23, 44, 45 e 58.

NG, R. T.; HAN, J. Clarans: A method for clustering objects for spatial data mining. *Knowledge and Data Engineering, IEEE Transactions on*, IEEE, v. 14, n. 5, p. 1003–1016, 2002. Citado na página 41.

NICKOLLS, J. et al. Scalable parallel programming with cuda. *Queue*, ACM, v. 6, n. 2, p. 40–53, 2008. Citado na página 16.

OLSON, C. F. Parallel algorithms for hierarchical clustering. *Parallel computing*, Elsevier, v. 21, n. 8, p. 1313–1325, 1995. Citado na página 16.

PALMER, C. R.; FALOUTSOS, C. *Density biased sampling: an improved method for data mining and clustering*. [S.l.]: ACM, 2000. Citado 8 vezes nas páginas 18, 19, 20, 41, 44, 55, 56 e 60.

PATWARY, M. A. et al. A new scalable parallel dbscan algorithm using the disjoint-set data structure. In: IEEE COMPUTER SOCIETY PRESS. *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. [S.l.], 2012. p. 62. Citado na página 40.

PAVAN, K. K. et al. Robust seed selection algorithm for k-means type algorithms. *arXiv preprint arXiv:1202.1585*, 2012. Citado na página 29.

PEARSON, K. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, Taylor & Francis, v. 2, n. 11, p. 559–572, 1901. Citado 2 vezes nas páginas 30 e 35.

PELLEG, D.; MOORE, A. Accelerating exact k-means algorithms with geometric reasoning. In: ACM. *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.], 1999. p. 277–281. Citado na página 31.

PHILLIPS, S. J. Acceleration of k-means and related clustering algorithms. In: SPRINGER. *Workshop on Algorithm Engineering and Experimentation*. [S.l.], 2002. p. 166–177. Citado na página 31.

QIAN, X. et al. An improved density biased sampling algorithm for clustering large-scale datasets. *JOURNAL OF INFORMATION &COMPUTATIONAL SCIENCE*, v. 11, n. 7, p. 2355–2364, 2014. Citado 4 vezes nas páginas 9, 20, 24 e 59.

RAND, W. M. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, Taylor & Francis Group, v. 66, n. 336, p. 846–850, 1971. Citado 2 vezes nas páginas 65 e 91.

RDUSSEEUN, L.; KAUFMAN, P. J. Clustering by means of medoids. *Statistical Data Analysis Based*, 1987. Citado na página 32.

ROS, F.; GUILLAUME, S. A new density-based sampling algorithm. *16th World Congress of the International Fuzzy Systems Association*, Atlantis Press, 2015. Citado 5 vezes nas páginas 51, 52, 53, 62 e 65.

ROS, F.; GUILLAUME, S. Dendis: A new density-based sampling for clustering algorithm. *Expert Systems with Applications*, Elsevier, v. 56, p. 349–359, 2016. Citado 19 vezes nas páginas 20, 22, 23, 25, 43, 44, 45, 48, 51, 52, 53, 62, 64, 65, 66, 93, 101, 103 e 119.

ROS, F.; GUILLAUME, S. Dides: a fast and effective sampling for clustering algorithm. *Knowledge and Information Systems*, Springer, p. 1–26, 2016. Citado 7 vezes nas páginas 20, 21, 45, 51, 52, 53 e 62.

ROS, F.; GUILLAUME, S. Protras: A probabilistic traversing sampling algorithm. *Expert Systems with Applications*, Elsevier, v. 105, p. 65–76, 2018. Citado na página 123.

ROS, F. et al. Development of predictive models by adaptive fuzzy partitioning. application to compounds active on the central nervous system. *Chemometrics and intelligent laboratory systems*, Elsevier, v. 67, n. 1, p. 29–50, 2003. Citado na página 44.

ROSENKRANTZ, D. J.; STEARNS, R. E.; LEWIS II, P. M. An analysis of several heuristics for the traveling salesman problem. *SIAM journal on computing*, SIAM, v. 6, n. 3, p. 563–581, 1977. Citado 2 vezes nas páginas 44 e 51.

ROUSSEEUW, P. J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, Elsevier, v. 20, p. 53–65, 1987. Citado na página 33.

RUIZ, C.; SPILIOPOULOU, M.; MENASALVAS, E. C-dbscan: Density-based clustering with constraints. In: SPRINGER. *International Workshop on Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing*. [S.l.], 2007. p. 216–223. Citado na página 40.

SARMA, T. H.; VISWANATH, P. Speeding-up the k-means clustering method: A prototype based approach. In: SPRINGER. *International Conference on Pattern Recognition and Machine Intelligence*. [S.l.], 2009. p. 56–61. Citado 2 vezes nas páginas 20 e 46.

SARMA, T. H.; VISWANATH, P.; REDDY, B. E. A hybrid approach to speed-up the k-means clustering method. *International Journal of Machine Learning and Cybernetics*, Springer, v. 4, n. 2, p. 107–117, 2013. Citado 5 vezes nas páginas 20, 44, 45, 46 e 48.

SAWANT, K. Adaptive methods for determining dbscan parameters. *International Journal of Innovative Science, Engineering & Technology*, v. 1, n. 4, p. 329–334, 2014. Citado na página 39.

SCHUBERT, E. et al. Dbscan revisited, revisited: why and how you should (still) use dbscan. *ACM Transactions on Database Systems (TODS)*, ACM, v. 42, n. 3, p. 19, 2017. Citado na página 37.

SCHÜTZE, H.; MANNING, C. D.; RAGHAVAN, P. *Introduction to information retrieval*. [S.l.]: Cambridge University Press, 2008. Citado na página 30.

SCULLEY, D. Web-scale k-means clustering. In: ACM. *Proceedings of the 19th international conference on World wide web*. [S.l.], 2010. p. 1177–1178. Citado na página 31.

SHI, J.; MALIK, J. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, IEEE, v. 22, n. 8, p. 888–905, 2000. Citado na página 16.

SIGKDD. *SIGKDD Test of Time Award*. 2014. Citado na página 35.

SONG, Q.; NI, J.; WANG, G. A fast clustering-based feature subset selection algorithm for high-dimensional data. *IEEE transactions on knowledge and data engineering*, IEEE, v. 25, n. 1, p. 1–14, 2013. Citado na página 15.

STEINHAUS, H. Sur la division des corp materiels en parties. *Bull. Acad. Polon. Sci*, v. 1, n. 804, p. 801, 1956. Citado 2 vezes nas páginas 16 e 27.

STOFFEL, K.; BELKONIENE, A. Parallel k/h-means clustering for large data sets. In: SPRINGER. *European Conference on Parallel Processing*. [S.l.], 1999. p. 1451–1454. Citado na página 16.

SU, T.; DY, J. A deterministic method for initializing k-means clustering. In: IEEE. *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*. [S.l.], 2004. p. 784–786. Citado na página 29.

UNCU, O. et al. Gridbscan: Grid density-based spatial clustering of applications with noise. In: IEEE. *Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on*. [S.l.], 2006. v. 4, p. 2976–2981. Citado na página 40.

VISWANATH, P.; BABU, V. S. Rough-dbscan: A fast hybrid density based clustering method for large data sets. *Pattern Recognition Letters*, Elsevier, v. 30, n. 16, p. 1477–1488, 2009. Citado 11 vezes nas páginas 15, 22, 23, 26, 41, 66, 67, 81, 83, 84 e 120.

WANG, L. et al. Approximate pairwise clustering for large data sets via sampling plus extension. *Pattern Recognition*, Elsevier, v. 44, n. 2, p. 222–235, 2011. Citado na página 17.

WILCOXON, F. Individual comparisons by ranking methods. *Biometrics bulletin*, JSTOR, v. 1, n. 6, p. 80–83, 1945. Citado 2 vezes nas páginas 104 e 115.

WU, Y.-P.; GUO, J.-J.; ZHANG, X.-J. A linear dbscan algorithm based on lsh. In: IEEE. *Machine Learning and Cybernetics, 2007 International Conference on*. [S.l.], 2007. v. 5, p. 2608–2614. Citado na página 41.

XIAO, Y. et al. A k-farthest-neighbor-based approach for support vector data description. *Applied intelligence*, Springer, v. 41, n. 1, p. 196–211, 2014. Citado 5 vezes nas páginas 21, 44, 69, 75 e 119.

XU, R.; WUNSCH, D. Survey of clustering algorithms. *IEEE Transactions on neural networks*, Ieee, v. 16, n. 3, p. 645–678, 2005. Citado na página 35.

YANG, A. Y. et al. Unsupervised segmentation of natural images via lossy data compression. *Computer Vision and Image Understanding*, Elsevier, v. 110, n. 2, p. 212–225, 2008. Citado na página 15.

ZHANG, Y. et al. The study of parallel k-means algorithm. In: IEEE. *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on*. [S.l.], 2006. v. 2, p. 5868–5871. Citado na página 31.

ZHAO, W.; MA, H.; HE, Q. Parallel k-means clustering based on mapreduce. In: SPRINGER. *IEEE International Conference on Cloud Computing*. [S.l.], 2009. p. 674–679. Citado na página 16.

ZHOU, H.; WANG, P.; LI, H. Research on adaptive parameters determination in dbscan algorithm. *JOURNAL OF INFORMATION &COMPUTATIONAL SCIENCE*, v. 9, n. 7, p. 1967–1973, 2012. Citado na página 39.

ZHOU, S. et al. Combining sampling technique with dbscan algorithm for clustering large spatial databases. In: SPRINGER. *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. [S.l.], 2000. p. 169–172. Citado na página 17.