A biased sampling method for applying DBSCAN

Igor de Moura Ventorim[1], **Diego Luchi**[2], **Flávio Miguel Varejão**[3]

[1]Departamento de Informática – Universidade Federal do Espírito Santo (UFES)

igor.ventorim@ufes.br, diego.lucchi@gmail.com, fvarejao@ninfa.inf.ufes.br

Abstract. *The DBSCAN algorithm is a traditional density-based clustering method. This algorithm allows to identify groups of different topological formats and also to discard isolated noises. The DBSCAN usually presents good results, however, it performs several calculations of distances in the clustering process, causing low efficiency, and its application in large datasets is not recommended. This work presents a new sampling method that makes it possible to apply DBSCAN to a reduced set of examples, in order to approximate the original result of the algorithm applied to the entire dataset. Therefore, the method makes it possible to execute large datasets with results similar to DBSCAN on entire dataset, however, with greater computational efficiency.*

Resumo. *O algoritmo DBSCAN é um método de agrupamento baseado em densidade. Permite identificar grupos de diferentes formatos topológicos e também descartar ruídos isolados. O DBSCAN normalmente apresenta bons resultados, no entanto, ele realiza diversos cálculos de distâncias no processo de agrupamento, possuindo baixa eficiência em grandes conjuntos de dados. Neste trabalho é apresentado um novo método amostral que possibilita aplicar o DBSCAN em um conjunto reduzido de exemplos, aproximando-se do resultado original do algoritmo aplicado sobre todo o conjunto de dados. Portanto, o método torna possível a execução de grandes conjuntos de dados com resultados semelhantes ao do DBSCAN na base de dados completa, porém, com maior eficiência.*

1. Introdução

O problema de Agrupamento é um dos mais frequentes no aprendizado de máquina, que tem por finalidade agrupar objetos com determinada similaridade. O algoritmo DBSCAN (Density-Based Spatial Clustering of Applications with Noise), proposto por [ESTER 1996] é um algoritmo clássico e um dos mais importantes algoritmos de agrupamento baseado em densidade espacial. O algoritmo permite realizar bons agrupamentos com formatos arbitrários mesmo com a presença de ruídos e sem a necessidade de um conhecimento prévio do número de grupos, como pode ser visto em [Xu R 2005] e [KDD]. Portanto, de acordo com suas características, o DBSCAN aparece como uma solução bastante atrativa para diversos problemas de agrupamento, se comparado a algoritmos clássicos da literatura como Single-Linkage[SIBSON 1973] e K-Means[MacQueen 1967]. No entanto, apesar de ser uma boa solução, ele possui o mesmo problema que outros algoritmos de agrupamento em relação a escalabilidade. Em seu processo de agrupamento é necessário calcular a densidade, ou seja, a distância de um ponto para todos os outros pontos, isto implica diretamente em um alto custo computacional ao se trabalhar com conjuntos de dados de alta dimensão.

Existem diversos métodos com o objetivo de reduzir o alto custo computacional do DBSCAN em grandes conjuntos de dados. Em um contexto geral as soluções propostas convergem para duas abordagens. A primeira abordagem tem como foco realizar a consulta do vizinho mais próximo de modo mais eficiente ou aproximado, como feito em [Chen 2018]. A segunda abordagem faz uso de técnicas de amostragem para encontrar uma solução aproximada, utilizando apenas um conjunto das instâncias para representar a base de dados inteira. A utilização da amostra no lugar da base de dados inteira reduz o tempo de execução do DBSCAN. Algumas abordagens de amostragem geram estatisticamente ou aleatoriamente a amostra, como em [Wang and Hamilton 2003], por outro lado, outras geram uma amostra como saída de um algoritmo de agrupamento, que possui uma complexidade computacional mais baixa, como em [El-Sonbaty and Farouk 2004] e [VISWANATH 2009].

O trabalho proposto foi motivado seguindo a abordagem realizada em [LUCHI 2019], que propõe uma nova técnica de redução do conjunto de dados aplicando um único passo do algoritmo de agrupamento Leader* no conjunto de dados. A partir disso, constrói-se a amostragem que é utilizada no DBSCAN com a finalidade de reduzir o tempo de execução do algoritmo em grandes bases de dados. Nesta nova técnica, substituímos o algortimo Leader* pelo algoritmo BIRCH [ZHANG 1996]. O objetivo é utilizar o algoritmo como intermediário para a construção da amostra que obterá um resultado aproximado do DBSCAN aplicado ao conjunto de dados inteiro. Apesar do algoritmo BIRCH ser mais complexo, ele constrói seus subgrupos de forma mais precisa utilizando uma CF Tree e diferentes cálculos de distância, diferenciando-se do Leader em não ser sensível a forma que os dados são apresentados para o algoritmo.

A técnica proposta consiste em quatro etapas, as quais são responsáveis por: gerar a amostra a partir do BIRCH, obter os centróides, aplicá-los no DBSCAN e rotular o conjunto de dados inteiro. Mesmo que a execução do algoritmo BIRCH possa ser mais lenta que o algoritmo Leader* em alguns cenários, ainda assim consideramos uma opção viável, devido o BIRCH ser um algoritmo rápido ao se trabalhar com grandes conjuntos de dados e fornecer uma amostragem que apresenta melhores resultados na maioria das vezes.

Neste artigo, diferentes algoritmos baseados em técnicas de amostragem foram selecionados para serem comparados. Os algoritmos DENDIS, DIDES, e ProTraS, são algoritmos baseados no príncipio de Farthest First Traversal que compõe sua amostra de modo iterativo até alcançar um critério de parada. Estes algoritmos se diferem no método de seleção do item da amostra em cada iteração, dado que no DIDES o critério principal na seleção do item a compor a amostra é a distância, no DENDIS a denside e no ProTraS o custo que é calculado baseado em probabilidades utilizando distância e densidade. Outro algoritmo comparado foi o Rough*-DBSCAN [LUCHI 2019], cujo tem como objetivo encontrar uma amostra da base de dados utilizando uma modificação do algoritmo de agrupamento Leader [Hartigan 1975], denominado Leader*.

Os experimentos realizados mediram o desempenho do método proposto comparando seus resultados com os resultados do DBSCAN aplicado em todo o conjunto de dados. A comparação foi feita com os métodos [LUCHI 2019],[ROS 2016],[ROS 2017] e [ROS 2018] que também utilizam amostragem. Para o método proposto foram encontrados resultados consistentes sobre todos os conjuntos de dados, diferentemente dos outros

métodos que apresentaram algumas oscilações. Além disso, o novo método apresentou o melhor resultado na maioria dos conjuntos de dados.

O restante do artigo está organizado do seguinte modo: Algoritmos de agrupamento baseados em amostragem são revistos na seção 2, introduzindo os algoritmos ProTraS, Dides, Dendis e Rough*-DBSCAN. A seção 3 apresenta o método BIRDBSCAN. Na seção 4, os experimentos realizados e seus resultados são descritos. Por fim, as conclusões e trabalhos futuros podem ser vistos na seção 5.

2. Algoritmos de Clustering baseados em Amostragem

Esta seção faz uma revisão rápida introduzindo o conceito de Farthest First Traversal apresentando algoritmos baseados em técnicas de amostragem e que são comparados com o método proposto.

2.1. Farthest First Traversal

Entre métodos de amostragem baseados em distância, o príncipio de Farthest First Traversal (FFT) é bastante utilizado, devido não requerer nenhum parâmetro de distância, diferentemente de outros algoritmos como, por exemplo, o Leader. O algoritmo de FFT trabalha interativamente adicionando um novo item a amostra em cada iteração até alcançar um critério de parada. Em sua versão canônica, o novo elemento é o mais distante dos elementos já selecionados e o critério de parada é o tamanho da amostra.

2.2. Dendis e Dides

Dois algoritmos FFT melhorados que combinam distância e densidade foram propostos, DENDIS [ROS 2016] e DIDES [ROS 2017], estes algoritmos se diferem com a amostra que é selecionada e também no critério de parada. Os dois algoritmos são direcionados por um único parâmetro, chamado granularidade, este parâmetro impacta diretamente no tamanho da amostra, quanto menor a granularidade maior o número de elementos na amostra, independente do conjunto de dados.

No algoritmo DENDIS, densidade é o conceito principal enquanto a distância é utilizada como controle, sendo que o elemento escolhido em cada iteração é o mais distante do grupo mais denso. Neste caso, na segunda etapa do algoritmo FFT, é realizado uma ordenação pela quantidade de elementos associados a cada elemento da amostra, a partir disso, se obtém o item mais distante do grupo associado ao primeiro elemento da amostra ordenada. O critério de parada também é definido baseado na densidade, dado que o algoritmo finaliza quando não há mais grupos com um número de elementos acima de um threshold e com uma distância máxima no grupo grande o bastante em relação a toda a distribuição. Este é um algoritmo focado na representação de densidade.

No algoritmo DIDES, a distância é o critério principal e o novo elemento selecionado para a amostra é o item mais distante de todos os outros presentes na amostra, como também é realizado na versão canônica do algoritmo FFT. Neste caso, no passo dois do algoritmo FFT, após encontrar o elemento mais distantes de cada grupo associado a cada elemento da amostra, verifica-se qual é o elemento mais distante entre todos os encontrados, e este é o elemento selecionado para compor a amostra. O threshold corresponde a quantidade mínima de elementos associados ao conjunto inicial, este é formado no primeiro passo do algoritmo para um grupo que se deseja ter representantes na amostra.

Um elemento da amostra com menos que o threshold é chamado de representante ruim. Quando a proporção dos elementos não selecionados, cujo representante é um representante ruim, é alta o suficiente, o espaço de entrada é coberto de maneira homogênea. A partir disso, a curva de evolução do elemento mais distante encontrado é modelado para definir um critério de parada como uma distância threshold.

A densidade é controlada em uma etapa de pós processamento, que descarta outliers e considera a representação de áreas conectadas, sendo um algoritmo com objetivo principal de garantir a cobertura espacial.

2.3. ProTraS

O algoritmo ProTraS [ROS 2018] é um algoritmo de amostragem dos mesmos autores do DIDES e DENDIS. O nível de aproximação, que é o custo da amostragem utilizado pelo algoritmo ProTraS, é o parâmetro ϵ, único parâmetro do algoritmo. Este serve também como critério de parada. O algoritmo finaliza quando o custo é inferior ao threshold. O parâmetro é utilizado para o processo de amostragem, dado que em cada iteração um novo elemento é adicionado a amostra, selecionado baseado na maior probabilidade de redução de custo, o que tende a limitar o tamanho da amostra.

Coresets são representações aproximadas do conjunto de dados. Um modelo utilizando um coreset é competitivo com o modelo utilizando o conjunto de dados inteiro, e o tamanho do coreset depende unicamente do parâmetro ϵ. Na apresentação do algoritmo ProTraS os autores provam a relação entre coresets e algoritmos FFT, afirmando que algoritmos baseados em FFT produzem coresets mesmo se eles não forem desenvolvidos com este objetivo, dado que a amostra final encontrada é um (k,ϵ)-coreset da base de dados completa com ϵ sendo igual a razão entre o custo da amostragem e o custo total.

O algoritmo ProTraS consiste basicamente em quatro passos, o primeiro seleciona um novo elemento no grupo de maior probabilidade de redução de custo, sendo este o grupo da amostra. O segundo passo consiste em associar cada elemento não selecionado ao elemento mais próximo presente na amostra. No terceiro passo a computação do custo é realizada. No quarto e último passo é verificado se o custo computado é maior que o parâmetro de custo, caso seja, retorna-se para o passo 1. Como o custo da amostragem em um determinado grupo é proporcional ao número de padrões e ao elemento com máxima distância dentro de um grupo, o novo elemento é escolhido baseado no que mais contribui para a redução do custo global. A probabilidade de redução de custo é uma combinação de duas probabilidades: uma baseada na distância e a outra na quantidade de elementos.

ProTraS, em contraste aos algoritmos FFT anteriores, como DENDIS ou DIDES, tem como objetivo produzir um coreset, devido a isso, o custo da amostragem tem papel principal, dado que é utilizado como parâmetro único e também critério de parada. A abordagem probabilística alcança um trade-off entre cobertura espacial e representação de densidade.

2.4. Rough*-DBSCAN

O algoritmo Rough*-DBSCAN [LUCHI 2019] foi baseado no algoritmo Rough-DBSCAN [VISWANATH 2009], cujo este encontra uma redução da base de dados composta pelos líderes encontrados aplicando o algoritmo de agrupamento Leader. Após a etapa de agrupamento com o algoritmo Leader, os autores propõem uma maneira para

decidir se os líderes de cada grupo irão ou não compor a amostra. Esta decisão é feita baseada na densidade de cada líder. Utilizando apenas o algoritmo Leader é difícil calcular a densidade de cada líder, por isto os autores propõem uma maneira de estimar a densidade, nomeada Rough-Cardinality.

Para entender o Rough*-DBSCAN é recomendado entender o algoritmo Leader, que é um algoritmo de agrupamento incremental que faz uma varredura única na base de dados. O algoritmo inicia com o primeiro elemento como líder e itera pelos outros elementos do conjunto de dados, verificando em cada iteração se o elemento está a uma distância τ de qualquer líder existente, caso esteja, este elemento se torna um seguidor, caso não, se torna um líder. É possível observar que o algoritmo é muito sensível a forma que os dados são apresentados para o mesmo, dado que os elementos que foram apresentados no começo têm uma chance maior de se tornarem líderes, e os primeiros líderes tem uma chance maior de terem mais seguidores. O Rough*-DBSCAN utiliza o Leader*, que é uma modificação do algoritmo Leader como uma estimativa de densidade mais direta. Esta variação do algoritmo precisa de uma segunda varredura na base de dados que, diferentemente do Leader original, associa um elemento não líder a vários líderes dentro de uma distância τ, tornando possível estimar a densidade de cada um dos líderes baseando-se no número de elementos associados ao mesmo.

O algoritmo ao obter os parâmetros do DBSCAN ϵ e $minPTS$ e o parâmetro τ executa o algoritmo Leader* e seleciona o líderes com mais de $minPTS$ seguidores em uma esfera de tamanho ϵ para compor a amostra e descarta o restante que não satisfaz a premissa. Dado que todos os líderes estão associados a mais que $minPTS$ seguidores o algoritmo DBSCAN é executado com $minPTS = 1$ na amostra de líderes. O último passo do algoritmo é substituir os grupos obtidos pelos líderes para todos os seus seguidores, não permitindo duplicatas escolhendo arbitrariamente apenas um líder nos casos de associação de mais de um líder, e os outros elementos são marcados como ruídos. O algoritmo Rough*-DBSCAN, obtém resultados superiores ao Rough-DBSCAN como pode ser visto em [LUCHI 2019].

3. A abordagem BIRDBSCAN

Esta seção introduz o algoritmo BIRCH que serve como base para o método proposto. Após a introdução do BIRCH, apresentamos o BIRDBSCAN, um método proposto de amostragem tendenciosa para aplicação do DBSCAN em grandes conjuntos de dados.

3.1. BIRCH

O método de agrupamento BIRCH (Balanced Interative Reducing and Clustering using Hierarchies) é recomendado para se trabalhar com conjuntos de dados muito grandes, como pode ser visto em [ZHANG 1996]. O algoritmo trabalha de forma incremental e dinâmica agrupando as entradas multi-dimensionais para tentar produzir o grupo de melhor qualidade preocupando-se com os recursos de hardware disponíveis.

Os conceitos de Clustering Feature (CF) e CF Tree são a base do agrupamento incremental do BIRCH. Uma Clustering Feature é um resumo do grupo contendo uma tripla de informações que é mantida sobre o mesmo, a tripla é formada pelo número de pontos, uma soma linear e uma soma quadrática dos pontos do grupo. A CF Tree é uma árvore de altura balanceada com dois parâmetros B e $Threshold$ que contém uma

representação compacta do conjunto de dados com cada nó folha sendo um subgrupo. Cada nó não folha contém no máximo B entradas e possui o formato [CF_i, $filho_i$] onde i = 1,2,...,B, "$filho_i$" é um ponteiro para o i-ésimo nó filho e CF_i é a CF do subgrupo representado por este filho. Cada nó folha contém no máximo L entradas no formato [CF_i] onde i = 1,2,...,L.

O algoritmo BIRCH é composto por 4 fases, sendo que algumas são opcionais:

Na fase 1 o algoritmo carrega os dados em memória para construir a CF Tree, sua principal tarefa é escanear todo o conjunto de dados e construir uma CF tree inicial em memória, tentando representar a informação do agrupamento do conjunto de dados da melhor forma possível de acordo com o limite de memória disponível. Inicia-se com um valor de $Threshold$, caso acabe a memória antes de finalizar a varredura dos dados, o valor do $Threshold$ é incrementado e reconstroi-se uma nova e menor CF Tree, reinserindo as folhas da antiga CF-Tree e continuando a varredura de onde tinha parado. Esta fase cria um resumo do conjunto de dados em memória. Ao finalizar a fase 1, obtém-se os seguintes benefícios nas computações subsequentes das próximas fases: rapidez e acurácia. O primeiro devido não haver mais necessidade de operações de I/O e o problema de agrupamento de dados original ser reduzido ao pequeno problema de agrupamento de subgrupos em folhas de entrada. O segundo devido aos outliers que são descartados.

A fase 2 é uma fase opcional e é similar a fase 1, porém, escaneando os nós folhas da CF tree gerada na fase 1 e reconstruindo uma pequena CF tree, enquanto remove novamente outliers e agrupa subconjuntos aglomerados em grandes conjuntos.

A Fase 3 é a fase de agrupamento global, nesta fase é utilizado um algoritmo para todos os nós folhas, permitindo que o algoritmo utilizado possa facilmente ser adaptado para trabalhar com um conjunto de subgrupos descrito pela CF. Inocentemente, é possível calcular o centroide do subgrupo e a partir da identificação do centroide, o subgrupo ser tratado como um único ponto, utilizando um algoritmo sem modificação. Outra opção é repetir n vezes o centroide e fazer uma leve alteração no algoritmo para considerar suas informações de contagem ou para se tornar geral e preciso, aplicar um algoritmo existente diretamente na tripla CF do subgrupo calculando maior distância e qualidade métrica.

Depois da fase 3, obtemos um conjunto de grupos que capturam o padrão principal da distribuição nos dados. No entanto, é possível encontrar imprecisões e falhas localizadas, como por exemplo, casos de ambiguidade quando um elemento duplicado pertence a nós folhas diferentes, ou o fato da fase 3 está aplicada em um resumo dos dados muito grosseiro. A Fase 4 é uma fase opcional, e implica no custo de passadas adicionais sobre os dados para corrigir imprecisões e refinar ainda mais o grupo. A fase 4 utiliza os centroides dos grupos produzidos pela fase 3 como sementes, e redistribui os elementos para as sementes mais próximas para obter um conjunto de novos grupos, e também prover a opção de descarte de outliers dos elementos muito distantes de sua semente mais próxima, fazendo com que não sejam incluídos no resultado.

3.2. BIRCH + DBSCAN: BIRDBSCAN

Com o objetivo de encontrar maneiras eficientes para se utilizar o algoritmo DBSCAN em grandes conjuntos de dados e observando o conceito de algoritmos de agrupamento utilizando técnicas de amostragem, podemos utilizar o algoritmo BIRCH em conjunto com o algoritmo DBSCAN. Esta combinação foi denominada de BIRDBSCAN e possi-

bilita encontrar soluções aproximadas do algoritmo DBSCAN aplicado no conjunto de dados inteiro. O BIRDBSCAN é uma alternativa ao DBSCAN em grandes conjuntos de dados, pois permite encontrar uma solução aproximada e em tempo de execução inferior ao do DBSCAN aplicado no conjunto de dados inteiro, esta aproximação do resultado é bastante competitiva se comparada a outros métodos com o mesmo propósito.

O método BIRDBSCAN possui três parâmetros em seu total, possui dois parâmetros $minPTS$ e ϵ de modo análogo ao DBSCAN, porém, possui também o parâmetro δ que é o parâmetro utilizado para o ajuste do parâmetro $Threshold$ utilizado pelo BIRCH. O método é composto por quatro etapas descritas a seguir:

A primeira etapa consiste na aplicação do algoritmo BIRCH em todo o conjunto de dados, nesta aplicação o algoritmo BIRCH é utilizado com o parâmetro $minPTS$ como o número limitante de subgrupos em cada nó da CF TREE e o parâmetro δ é um fator multiplicador que é aplicado ao parâmetro ϵ e é utilizado como $Threshold$ do algoritmo BIRCH ao compor a CF Tree. No final desta etapa se tem o resultado do algoritmo BIRCH com os subgrupos que o algoritmo retorna.

Na segunda etapa temos os subgrupos encontrados pelo BIRCH, em conjunto com os subgrupos obtemos os centroides, a quantidade de elementos e quais são os elementos que pertencem a cada subgrupo. Os centroides dos subgrupos passam a compor a amostra, semelhante a fase 3 do algoritmo BIRCH, dado que os elementos do conjunto de dados inicial são resumidos somente pelo centroide de seu subgrupo, deste modo passamos a ter um novo conjunto de dados que é inferior ao conjunto de dados inteiro.

Na terceira etapa, aplica-se o DBSCAN na amostra, utilizando os parâmetros $minPTS$ e ϵ utilizando os pesos de cada elemento da amostra, que correspondem a quantidade de elementos no subgrupo do elemento da amostra, isto permite que uma amostra com o peso de pelo menos $minPTS$ por si só seja um elemento principal e não seja considerado como outlier. No final desta etapa obtemos os grupos de cada elemento da amostra.

Na quarta e última etapa fazemos a rotulação dos elementos do conjunto de dados inteiro baseado na saída do DBSCAN, nesta etapa a rotulação do conjunto de dados inteiro é realizada a partir dos subgrupos retornados pelo algoritmo BIRCH na primeira etapa e que foram identificados e organizados na segunda etapa. Este processo, é um processo rápido e sem muito custo, devido a forma que o BIRDBSCAN trabalha. A rotulação é feita do seguinte modo, se um determinado centroide da amostra foi rotulado pelo DBSCAN com um rótulo, todo o seu subgrupo também será rotulado com este mesmo rótulo, fazendo com que o conjunto de dados inteiro seja rotulado.

Dado que a variação do parâmetro $Thresold$ influencia no tamanho da CF Tree do algoritmo BIRCH, os parâmetros δ e ϵ estão correlacionados ao parâmetro $Threshold$ do BIRCH no método BIRDBSCAN, por isto, estes também influenciam no tamanho da CF Tree e consequentemente no tamanho e qualidade da amostra que será encontrada e aplicada no DBSCAN. De modo empírico neste proposta foi definido $\delta = 0.5$.

4. Experimentos

A comparação dos métodos foi realizada utilizando um conjunto de bases de dados de teste escolhidas de [LUCHI 2019] contendo 7 bases de dados. As bases de dados foram

escolhidas para verificar o comportamento dos métodos em diferentes cenários, variando dimensões (de 8 a 112), tamanho (de pequenos a muito grandes), forma e densidade. As principais características das bases de dados podem ser vistas na tabela 1.

Para podermos certificar a qualidade dos métodos, foram escolhidos conjuntos de dados de tamanhos limitados, deste modo é possível a execução do algoritmo DBSCAN nestes conjuntos em tempo admissível, tornando possível a comparação entre o resultado do DBSCAN nos conjuntos de dados inteiros e o resultado dos métodos avaliados. É válido destacar que, todos os métodos amostrais em questão são recomendados em situações que a utilização do DBSCAN poderia ser útil caso não fosse tão custoso, ou seja, é recomendado para conjuntos de dados muito grandes que possuem um alto custo computacional e que inviabilizam a utilização do DBSCAN no conjunto de dados inteiro, caso contrário sempre será melhor a utilização somente do DBSCAN.

Tabela 1. Bases de dados selecionadas de [LUCHI 2019]

Dataset	elements	features
Abalone	4177	8
Cadata	20640	8
Letter	20000	16
Mushrooms	8124	112
Pendigits	10992	10
Sensorless	58509	48
Shuttle	58000	7

Os experimentos realizados tentam identificar qual dos métodos se aproxima mais dos resultados do DBSCAN no conjunto de dados inteiro.

Existem duas métricas normalmente utilizadas para se avaliar a similaridade entre do grupos, a Rand Index (RI) [Rand 1971] e a Adjusted Rand Index (ARI) [HUBERT 1985]. O RI é uma medida de similaridade entre dois grupos, no entanto, é possível que dois conjuntos aleatórios tenham um RI próximo de 1 em casos de muitos grupos, isto ocorre devido haver uma grande chance de um par de itens nos dois conjuntos estar em grupos diferentes, e esse fato é contado como um acerto no cálculo desta métrica. O ARI é uma correção do RI, essa correção estabelece uma linha de base usando a similaridade esperada de todas as comparações entre pares e os agrupamentos especificados por um modelo aleatório, enquanto o RI gera um valor entre 0 e +1, o ARI pode gerar valores negativos se o índice for menor que o esperado. Resultados negativos no ARI significam que o resultado é menor do que o esperado de um resultado aleatório.

Nas avaliações de [ROS 2018],[ROS 2017] e [ROS 2016] foi utilizado a métrica RI e em [LUCHI 2019] foi utilizado o ARI. A métrica selecionada para realizar a avaliação da perfomance foi o ARI [HUBERT 1985]. Como O ARI possui um valor máximo de 1 indicando que os grupos são exatamente os mesmos, em todos os experimentos foi utilizado seu valor percentual. Cada método retorna a sua amostragem que é executada no DBSCAN, e feito a expansão dos rótulos para todo o conjunto de dados da base. O ARI é medido comparando o agrupamento obtido por cada método com o agrupamento obtido pelo DBSCAN executado na base de dados completa.

O algoritmo DBSCAN é executado com dois parâmetros: A distância ϵ que de-

fine uma vizinhança esférica e o número mínimo de pontos, $minPTS$, que é necessário para considerar um ponto como núcleo e para rotular outliers. Todos os métodos estão utilizando o DBSCAN e consequentemente precisam dos parâmetros do algoritmo. Para a realização dos experimentos, um processo específico foi definido para testar diferentes parâmetros, este processo foi inspirado no procedimento dos experimentos descritos em [ROS 2018].

O procedimento utilizado em [ROS 2018] é um procedimento visando validar a qualidade do resultado dos métodos em avaliação, e consiste em executar o DBSCAN na amostra do conjunto de dados que é gerada pelos algoritmos DENDIS, DIDES ou PROTRAS com $minPTS = max(1, 0.01 * tamanhodobasededados)$ e identificar alguma distância D_{init} para ε que nos retorne no máximo 2 grupos quando aplicado o DBSCAN no conjunto de dados inteiro. Após encontrar o valor D_{init} então uma faixa de $[\frac{D_{init}}{1000}$ $D_{init}]$ com um valor incremental de $\frac{D_{init}}{1000}$ é gerada, esta faixa possui 1000 valores, e cada valor da faixa é considerado como um parâmetro ε a ser executado no DBSCAN. Deste modo, cada conjunto de dados é executado 1000 vezes, uma para cada valor da faixa $[\frac{D_{init}}{1000}, D_{init}]$. Este procedimento de teste é relevante, devido o valor de ε influenciar diretamente no número de grupos resultantes pelo DBSCAN, dado que quanto maior o valor de ε menor o número de grupos, portanto, o procedimento testa os métodos para diferentes números de grupos.

Para os experimentos realizados foi feita uma pequena alteração no procedimento de [ROS 2018], esta alteração descarta alguns valores que são assumidos por ε ao estarem presentes na faixa $[\frac{D_{init}}{1000}, D_{init}]$. Esta decisão foi tomada baseado que para estes valores o DBSCAN retorna somente elementos rotulados como outliers, isto ocorre nos casos em que o valor de ε é tão pequeno de modo a não ser possível encontrar nenhum grupo denso, ou seja com no mínimo $minPTS$. Por exemplo, ao considerarmos $D_{init} = 1$ os experimentos começam com $\varepsilon = \frac{D_{init}}{1000} = 0.001$, e no segundo teste $\varepsilon = 0.002$, dependendo do conjunto de dados esses valores podem ser valores inviáveis, retornando todos os pontos como outliers por não conseguir encontrar grupos densos com essa distância, ou seja, não fazendo sentido para uma aplicação real. Deste modo, somente foram considerados para análise os valores de ε dos experimentos que retornaram algum grupo e não apenas outliers.

Para que seja feito uma comparação correta e conseguir identificar se realmente o método de amostragem apresentou resultados úteis, é necessário realizar a comparação com o DBSCAN executado na base de dados inteira, pois quando se executa o DBSCAN somente na amostragem não é possível realizar essa comparação de modo concreto sem a realização de uma expansão dos rótulos para todo o conjunto de dados. Os métodos BIRDBSCAN e Rough*-DBSCAN utilizam o próprio método de rotulação do conjunto de dados inteiro, porém em [ROS 2016], [ROS 2017] e [ROS 2018] não é explicito qual a técnica de rotulação utilizada, baseado nessa premissa e assumindo a alta probabilidade de pontos próximos serem do mesmo grupo, a técnica de expansão que foi utilizada para estes métodos foi a estratégia do vizinho mais próximo.

O processo de execução realizado nestes métodos é da seguinte forma: inicialmente aplica-se a amostra gerada pelo método no DBSCAN e tem como saída os pontos rotulados com seus respectivos grupos, após obter-se a amostra rotulada inicia-se o processo de rotulação dos pontos do conjunto de dados que não estão presentes na amostra.

Utilizando o vizinho mais próximo, o ponto que não pertence a amostra recebe o rótulo do ponto mais próximo a ele que se encontra presente na amostra. Este processo de rotulação utilizando a técnica do vizinho mais próximo é um processo que pode ser custoso para conjuntos de dados muito grandes.

O procedimento realizado para cada conjunto de dados foi: executar o algoritmo DBSCAN no conjunto de dados inteiro utilizando os parâmetros que foram encontrados; executar o método que se deseja avaliar utilizando os parâmetros definidos; realizar a expansão para todo o conjunto de dados nos métodos que não possuem sua própria forma de rotular os elementos não presentes na amostra; executar a métrica ARI comparando as saída do método a ser avaliado com a saída do DBSCAN no conjunto de dados inteiro para os parâmetros informados. Após a execução deste processo para todos os parâmetros definidos para o conjunto de dados, se calcula a média do ARI para todos os experimentos do conjunto de dados que foram avaliados. Este processo é realizado para cada um de todos os conjuntos de dados selecionados e para todos os métodos a serem avaliados.

Após encontrar a média do ARI para cada conjunto de dados, visando facilitar a análise, foram organizadas três tabelas para exibição dos resultados dos experimentos. A primeira tabela contém os métodos, conjuntos de dados e as médias do ARI, o resultado pode ser visto na tabela 2. Para os casos do DIDES e DENDIS que devido a quantidade de dimensões o algoritmo não rodou, a tabela foi preenchida com o elemento $-$. Os valores em negrito são correspondentes aos algoritmos que apresentaram melhores resultados comparados aos outros.

Tabela 2. Tabela de ARI

DATASETS	PROTRAS	BIRDBSCAN	DENDIS	DIDES	R*
Abalone	94,25	88,95	**96,42**	88,55	33,21
Cadata	65,51	**80,42**	75,72	7,46	15,93
Letter	5,63	**59,67**	12,66	0,00	4,54
Mushrooms	-0,29	**60,47**	-	-	6,02
Pendigits	-1,21	**67,02**	4,51	-1,19	18,85
Sensorless	24,07	**78,85**	-	-	1,10
Shuttle	35,75	78,64	**89,98**	-0,45	0,05

A tabela 3 contém um ranking dos algoritmos baseado na tabela 2. Neste tipo de avaliação o algoritmo que possui um menor valor no somatório total é considerado o melhor colocado, como pode ser visto na tabela 3 o BIRDBSCAN obteve a melhor pontuação se comparado aos outros algoritmos.

A tabela 4 contém a média dos experimentos calculados a partir da tabela 2 para todos os conjuntos de dados de algoritmos avaliados confrontados com o BIRDBSCAN. Para a comparação ser justa, somente foram adicionados os resultados dos datasets que rodaram em ambos os algoritmos. Essa escolha foi realizada devido a um dataset que não rodou determinado algoritmo ter que ficar com resultado igual a 0% no cálculo da média caso resolvêssemos considerá-lo. Por outro lado, caso resolvêssemos desconsiderá-lo, seria equivalente ao concedermos pesos diferentes para cada dataset na mesma

Tabela 3. Tabela de ARI

DATASETS	PROTRAS	BIRDBSCAN	DENDIS	DIDES	R*
Abalone	2	3	**1**	4	5
Cadata	3	**1**	2	5	4
Letter	3	**1**	2	5	4
Mushrooms	3	1	4,5	4,5	2
Pendigits	5	**1**	3	4	2
Sensorless	2	1	4,5	4,5	3
Shuttle	3	2	**1**	5	4
Total	21	**10**	18	32	24

Tabela 4. Média dos experimentos: Considerando somentes os datasets que rodaram em ambos.

Algoritmo	Média Algoritmo	Média BIRDBSCAN
PROTRAS	31,96	**73,43**
DENDIS	55,85	**74,94**
DIDES	18,87	**74,94**
Rough*-DBSCAN	29,58	**73,43**

comparação, devido sua contagem ser ignorada na divisão com o número de datasets no cálculo da média. Ambas as considerações influenciariam bastante ao compararmos a média, fazendo com que o algoritmo apresentasse uma média inferior ou superior ao seu desempenho geral se comparado a outro algoritmo que rodou o dataset.

O BIRDBSCAN é o método que possui a maior consistência em seus resultados, influenciando com que ele possua a melhor média sempre que comparado aos outros algoritmos do experimento, como pode ser visto na tabela 4.

5. Conclusões

Este trabalho propôs um método que realiza a amostragem de grandes bases de dados com a finalidade de obter uma aproximação do resultado da execução do algoritmo DBSCAN na base de dados completa. Recomenda-se sua utilização nos casos em que o DBSCAN seria uma boa opção caso não fosse tão custoso nessas bases de dados. Ao utilizarmos o método proposto obtemos uma aproximação do método DBSCAN aplicado na base de dados inteira.

Através dos experimentos foi possível perceber que nenhum dos métodos é o melhor em todas as situações. No entanto, foi possível identificar que o método proposto é o mais consistente, devido apresentar menor variação. O método proposto pode ser melhor aproveitado se for possível identificar com precisão quais são as melhores bases de dados para a sua aplicação.

Neste trabalho foi empiricamente escolhido um valor para o parâmetro δ que apresentou resultados bons, porém, acredita-se ser possível encontrar uma transformação que a partir do valor de ε do DBSCAN definir um valor para δ resultando em uma amos-

tram de melhor qualidade. Este problema em conjunto com um estudo de bases de dados recomendados para o método proposto será feito em trabalhos futuros.

Referências

Chen, Yewang, e. a. (2018). A fast clustering algorithm based on pruning unnecessary distance computations in dbscan for high-dimensional data. *Pattern Recognition 83*, pages 375–387.

El-Sonbaty, Yasser, M. A. I. and Farouk, M. (2004). An efficient density based clustering algorithm for large databases. *16th IEEE international conference on tools with artificial intelligence. IEEE*.

ESTER, M. e. a. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. *Kdd*, pages 226–231.

Hartigan, J. A. (1975). Clustering algorithms. *John Wiley & Sons, Inc.*

HUBERT, Lawrence; ARABIE, P. (1985). Comparing partitions. *Journal of classification*, 2(1):193–218.

KDD. 2014 sigkdd test of time award. Disponível em `http://www.kdd.org/News/view/2014-sigkdd-test-of-time-award` (07/07/2020).

LUCHI, Diego; RODRIGUES, A. L. V. F. M. (2019). Sampling approaches for applying dbscan to large datasets. *Pattern Recognition Letters*, 117:90–96.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. *In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, 1(14):281–297.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association 66.336*, pages 846–850.

ROS, Frédéric; GUILLAUME, S. (2016). Dendis: A new density-based sampling for clustering algorithm. *Expert Systems with Applications*, 56:349–359.

ROS, Frédéric; GUILLAUME, S. (2017). Dides: a fast and effective sampling for clustering algorithm. *Knowledge and information systems*, 50(2):543–568.

ROS, Frédéric; GUILLAUME, S. (2018). Protras: a probabilistic traversing sampling algorithm. *Expert Systems with Applications*, 105:65–76.

SIBSON, R. (1973). Slink: an optimally efficient algorithm for the single-link cluster method. *The computer journal*, 16(1):30–34.

VISWANATH, P.; BABU, V. S. (2009). Rough-dbscan: A fast hybrid density based clustering method for large data sets. *Pattern Recognition Letters*, 30(16):1477–1488.

Wang, X. and Hamilton, H. J. (2003). Dbrs: a density-based spatial clustering method with random sampling. *Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, Berlin, Heidelberg*.

Xu R, W. D. (2005). Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–78.

ZHANG, Tian; RAMAKRISHNAN, R. L. M. (1996). Birch: an efficient data clustering method for very large databases. *ACM Sigmod Record*, 25(2):103–114.