Trabalho Prático 2: Algoritmos Aproximativos

Bernardo R. de Almeida

Departamento de Ciência da Computação - Universidade Federal de Minas Gerais
Belo Horizonte - Minas Gerais - Brasil

bera01@ufmg.br (2021032234)

Abstract. *The present article aims at exploring a practical approach into approximative algorithms. For that objective, the clusterization problem will be tackled based on a k-centers algorithm implementation. Experimentation will also be conducted on several databases, for each one being evaluated various metrics and the results, compared with another approach, the k-means, for which an existing library's procedures will be used.*

Resumo. *O presente artigo tem por objetivo explorar de maneira prática conceitos relacionados a algoritmos aproximativos. Para tal, será realizada a implementação de um programa para o problema de clusterização pela abordagem dos k-centros. Também serão realizados testes com bases de dados, para as quais métricas serão avaliadas e os resultados, comparados com uma outra abordagem, k-means, para a qual uma pré-implementação será utilizada.*

1. Introdução

O problema da **clusterização** é notoriamente relevante dentro do contexto de ciência de dados. De maneira breve, clusterizar um conjunto de dados consiste em uma forma de agrupamento, em que grupos, vulgo "clusters", são organizados a partir de critérios coerentes com uma dada aplicação. Dentre as motivações para essa área de estudos, encontram-se o treinamento de modelos de previsão e a geração de insights a partir de informações empíricas. Sua relevância advém não somente de seus objetivos, mas da flexibilidade e da dinamicidade inerentes à sua implementação e às possibilidades quanto ao estabelecimento de critérios e de parâmetros para a geração dos clusters.

No presente trabalho prático, será explorada a abordagem dos **k-centros**: dado um conjunto de dados, representados por pontos em um espaço n-dimensional, uma função de distância, a qual mapeia duplas de pontos a uma métrica de valores, e um inteiro k, relativo ao número de clusters de interesse, deseja-se um conjunto de k pontos nesse mesmo espaço, vulgo "os centros", que minimizem a distância máxima de um ponto original qualquer ao seu respectivo centro, sendo este o mais próximo dentre os k existentes. Em outras palavras, deseja-se uma clusterização que minimize o raio máximo dos clusters, tomados pontos, seus centros, como referências. Para tal, dada a natureza difícil do problema em sua forma bruta, será elaborado um **algoritmo 2-aproximativo** na linguagem de programação **Python**. Também serão realizados testes com diversas bases de dados e os resultados serão agregados e comparados a uma outra abordagem para o problema de clusterização, seja ela nomeada **k-means**, para a qual uma biblioteca pré-definida será utilizada.

Em um primeiro momento, a implementação do algoritmo aproximativo será detalhada na seção **[Descrição da Implementação]**, em que decisões arbitrárias serão justificadas e pormenores, abordados, conforme sua relevância. Em seguida, na seção **[Descrição de Métodos e Métricas]**, os parâmetros relevados para fins de teste serão

percorridos e sua expressividade, debatida. A procedência dos testes e tomadas de decisões serão discorridas na seção **[Descrição dos Experimentos e das Bases de Dados]**. Os resultados em si, por sua vez, estarão organizados na seção **[Apresentação e Análise dos Resultados]**, em que uma breve análise também será apresentada. Por fim, a seção **[Conclusões]** sediará considerações finais sobre o trabalho como um todo.

2. Descrição da Implementação

O primeiro componente do algoritmo 2-aproximativo a ser implementado foi a função de distância entre dois pontos. Foi de interesse que esta se comportasse como uma **métrica**, isto é, que fosse não negativa, simétrica e que respeitasse a desigualdade triangular. Em termos teóricos, foi escolhida a **métrica Minkowski** para tal, a qual é formalmente definida pela fórmula $(\sum_{i=1}^{n} |x_i - y_i|^p)^{1/p}$, em que $x = (x_1, \cdots, x_n)$, $y = (y_1, \cdots, y_n)$ e p é um parâmetro para definição da norma ($p = 1$ equivale à distância de Manhattan e $p = 2$, à distância Euclidiana; essa função é uma métrica apenas para $p >= 1$). Este será passado para a função principal do algoritmo como um parâmetro. A implementação é um mapeamento direto da fórmula apresentada, utilizando métodos e estruturas de dados da biblioteca **Numpy**.

O mecanismo principal do algoritmo segue uma intuição bem simples. Recebidos um conjunto S de pontos no espaço, um parâmetro p para a métrica de distância e um parâmetro k para a quantidade de centros desejados, primeiro, compara-se a cardinalidade de S com o valor de k. Caso $|S| <= k$, a resposta é trivial: cada ponto é seu próprio centro e o raio máximo é 0. Caso contrário, escolhe-se um ponto de S aleatório, inclui-o no conjunto C de centros e, enquanto $|C| <= k$, inclui-se o ponto de S mais distante de seu centro em C, retornando este conjunto ao final.

O primeiro caso é implementado de maneira direta com uma verificação condicional, sendo retornados o próprio conjunto S, um raio igual a 0 e um conjunto L de identificadores ("labels") - o elemento l_i equivale ao índice do centro relativo ao ponto s_i em C - em que cada ponto é seu próprio centro. Para o segundo caso, são criadas estruturas de dados auxiliares que armazenam o conjunto C de centros, o conjunto D das distâncias dos pontos ao seu centro mais próximo e o conjunto L de identificadores dos centros de cada ponto. Inicialmente, um índice do conjunto S é aleatoriamente escolhido como o primeiro ponto a ser incluído em C. Em seguida, as distâncias máximas dos demais pontos ao seu centro mais próximo são atualizadas no conjunto D, comparando-se, para cada ponto, a distância nele existente (a menor até então) com a distância a esse novo centro, utilizando, para tal, o parâmetro p e a função de distância de Minkowski. O impacto dessas mudanças também é contabilizado no conjunto L de identificadores. Por fim, o conjunto D é novamente percorrido para se determinar o índice do elemento com a maior distância ao conjunto C, reiterando-se o processo para ele até que $|C| = k$. Ao final, são retornados o conjunto C, o valor da maior distância de um ponto ao seu centro e o conjunto L. O conjunto C consiste nos k centros, enquanto o conjunto L consiste na classificação de cada ponto de S (elementos

com um mesmo valor pertencem ao mesmo "grupo", ao mesmo "cluster", isto é, estão mais próximos de um mesmo centro). Novamente, a biblioteca Numpy foi utilizada como suporte para toda a implementação.

3. Descrição de Métodos e Métricas

Três principais métricas foram adotadas para a avaliação do algoritmo proposto. A primeira, ***Silhouette Score*** (silhueta), é uma medida que pondera a coesão dos elementos de um conjunto quanto ao seu cluster designado e sua separação quanto aos demais clusters. Em alto nível, ela avalia a consistência da classificação, isto é, o quão próximos são objetos intra-cluster e o quão distantes são objetos inter-clusters. Ela recebe o conjunto de pontos (ou instâncias) e sua classificação (o conjunto de identificadores, ou "labels") e retorna um valor no intervalo $[-1, +1]$ (quanto mais próximo de 1, mais coesa é a classificação). A segunda, ***Rand Score*** (índice de Rand), avalia a similaridade entre duas classificações, isto é, entre os resultados de duas operações de clustering. Uma maneira de interpretar essa métrica é que, dada a classificação real das instâncias, ela avalia a precisão de uma classificação alternativa, relevando pares de elementos designados e não designados ao mesmo cluster. Ela recebe o conjunto de identificadores real e o conjunto de interesse e retorna um valor no intervalo $[-0.5, +1]$ (quanto mais próximo de 1, mais próximas são as classificações; esse intervalo é referente à versão "ajustada" dessa métrica). Por fim, o **tempo de execução** do algoritmo também foi contabilizado, em que se avalia o momento desde a invocação da função de clusterização, até o seu retorno. As duas primeiras métricas foram computadas com o auxílio da biblioteca **SciKit Learn**, enquanto a terceira foi medida com métodos da biblioteca **Time**.

Além disso, para fins de comparação da qualidade do algoritmo 2-aproximativo implementado, uma abordagem alternativa ao problema de clusterização também foi explorada. Esta, denominada **k-means**, adota uma perspectiva alternativa em termos dos critérios elencados para a escolha dos centros e, consequentemente, na designação dos clusters. Ao invés de buscar minimizar o raio máximo, este algoritmo visa à minimização da **inércia** da classificação. Esta outra métrica é definida como a soma do quadrado do valor absoluto da distância de cada ponto ao seu centro, o que, em termos formais, é dado pela fórmula $(\sum_{i=1}^{n} |x_i - \mu_i|^2)$, em que μ_i é o centro relativo ao ponto x_i. Novamente, a implementação presente na biblioteca SciKit Learn foi utilizada. O raio e a inércia, portanto, também foram adotados como métricas secundárias de avaliação dos procedimentos de clusterização.

4. Descrição dos Experimentos e das Bases de Dados

Os algoritmos apresentados foram testados sobre 10 diferentes bases de dados. Essas coleções foram extraídas do repositório de domínio público *UCI Machine Learning*, mantido pelo *Center for Machine Learning and Intelligent Systems* da *University of California, Irvine*. Em ordem alfabética, são elas: *abalone.data*[1], *drybean.arff*[2], *electricalgrid.csv*[3], *iranianchurn.csv*[4], *optdigits.data*[5], *pendigits.data*[6], *redwine.csv*[7], *segmentation.data*[8], *whitewine.csv*[9], *yeast.data*[10]. Cada uma consiste em um conjunto com mais de 700 instâncias e que possui atributos predominantemente numéricos -

sendo estes aqueles utilizados para a tarefa de classificação ao serem interpretados como coordenadas de pontos em um espaço n-dimensional -, mas também um atributo "Class", o qual diz respeito ao identificador (ou à classe, "label") real de cada instância - sendo este utilizado para a avaliação da clusterização realizada pelos algoritmos propostos sob as métricas apresentadas previamente. As bases de dados não estão em formato bruto; todas foram tratadas conforme formalidades adotadas para este trabalho prático: todos os comentários foram removidos; o atributo referente à classe, possivelmente com um nome diferente, foi renomeado para "Class"; instâncias de teste e de treino separadas foram unidas em um único arquivo; nos casos em que não havia "header", este foi adicionado com os nomes dos atributos, os quais, quando não fornecidos de alguma forma pela base de dados, foram definidos como a sequência de números naturais, com exceção do atributo "Class". Novamente, reitera-se que apenas os atributos numéricos foram utilizados para fins de clusterização, além de que todas as instâncias com entradas faltantes foram removidas. A biblioteca **Pandas** foi utilizada como o suporte para essa implementação.

Em termos dos experimentos, para cada base de dados, cada um dos algoritmos, k-centers e k-means, foi executado 30 vezes, em cada qual foram computadas o raio, a inércia, a silhueta, o índice de Rand e o tempo de execução, como definidas previamente. O conjunto de pontos consistiu nas instâncias em si, filtradas apenas por seus atributos numéricos, e a quantidade de clusters desejada - o parâmetro k - foi definida como o número de valores únicos do atributo "Class". Os resultados foram agregados em uma tabela que registrou o valor médio e o desvio-padrão de cada uma dessas métricas com relação às 30 execuções para cada base de dados. Esse procedimento como um todo foi realizado duas vezes, uma para o valor do parâmetro p, relativo à função de distância, igual a 1 (distância de Manhattan) e outra, igual a 2 (distância Euclidiana). Dessa forma, no total, foram coletadas quatro tabelas, duas relativas ao algoritmo k-centers e duas relativas ao algoritmo k-means, sendo uma para cada valor de p. Cada linha diz respeito a uma base de dados e as colunas são a média e o desvio-padrão de cada uma das 5 métricas com relação às 30 execuções.

5. Apresentação e Análise dos Resultados

A seguir, tem-se os resultados obtidos durante os experimentos realizados.

Tabela 1. Resultados para o algoritmo k-centros ($p = 1$)

Mean	Radius	Inertia	Silhouette	Rand Score	Exec. Time
abalone.data	2,29	2.501,83	0,24	0,46	0,86
drybean.arff	52.450,68	3.117.056.481.230,38	0,46	0,25	0,72
electricalgrid.csv	23,97	489.343,07	0,14	0,05	0,15
iranianchurn.csv	8.417,62	38.458.297.967,00	0,68	-0,10	0,05
optdigits.data	266,07	8.408.251,47	0,09	0,31	0,50
pendigits.data	527,63	142.491.122,27	0,14	0,28	0,90
redwine.csv	59,61	608.163,63	0,45	-0,01	0,08
segmentation.data	736,03	54.217.499,19	0,38	0,11	0,13

whitewine.csv	119,89	9.104.589,80	0,19	0,01	0,26
yeast.data	1,10	165,37	0,13	0,07	0,12

STD	Radius	Inertia	Silhouette	Rand Score	Exec. Time
abalone.data	0,09	191,82	0,03	0,02	0,02
drybean.arff	3.502,57	1.526.294.592.303,04	0,10	0,06	0,02
electricalgrid.csv	1,19	65.500,15	0,00	0,08	0,00
iranianchurn.csv	1.223,19	12.708.154.560,79	0,01	0,01	0,00
optdigits.data	4,20	272.057,53	0,02	0,04	0,01
pendigits.data	14,72	10.549.799,14	0,04	0,04	0,02
redwine.csv	4,58	145.756,66	0,03	0,01	0,01
segmentation.data	38,72	15.254.142,71	0,11	0,03	0,01
whitewine.csv	9,67	3.668.603,71	0,07	0,01	0,01
yeast.data	0,06	46,66	0,05	0,05	0,00

Tabela 2. Resultados para o algoritmo k-centros ($p = 2$)

Mean	Radius	Inertia	Silhouette	Rand Score	Exec. Time
abalone.data	1,26	2.440,18	0,28	0,56	0,85
drybean.arff	36.974,03	2.541.032.238.483,62	0,52	0,26	0,72
electricalgrid.csv	11,76	497.629,98	0,14	0,05	0,15
iranianchurn.csv	7.162,27	32.988.226.798,12	0,68	-0,10	0,05
optdigits.data	51,68	8.470.361,30	0,09	0,29	0,50
pendigits.data	173,92	141.130.647,83	0,16	0,30	0,89
redwine.csv	40,14	636.500,06	0,45	-0,01	0,08
segmentation.data	353,03	71.789.613,90	0,48	0,02	0,13
whitewine.csv	81,38	7.823.969,82	0,29	0,01	0,26
yeast.data	0,58	154,25	0,18	0,07	0,12

STD	Radius	Inertia	Silhouette	Rand Score	Exec. Time
abalone.data	0,03	246,38	0,07	0,07	0,02
drybean.arff	2.951,13	1.279.677.107.145,33	0,08	0,04	0,02
electricalgrid.csv	0,56	65.076,99	0,01	0,06	0,01
iranianchurn.csv	677,46	9.318.318.604,25	0,02	0,01	0,01
optdigits.data	0,72	363.857,68	0,02	0,06	0,02
pendigits.data	3,52	14.449.219,60	0,05	0,05	0,02
redwine.csv	3,02	109.427,63	0,03	0,01	0,00
segmentation.data	27,62	23.149.549,71	0,04	0,04	0,01
whitewine.csv	4,33	2.358.891,21	0,09	0,01	0,01
yeast.data	0,02	43,04	0,05	0,04	0,00

Tabela 3. Resultados para o algoritmo k-means ($p = 1$)

Mean	Radius	Inertia	Silhouette	Rand Score	Exec. Time
abalone.data	3,34	401,34	0,55	0,65	0,09
drybean.arff	121.916,21	652.105.170.631,90	0,53	0,39	0,11
electricalgrid.csv	19,32	258.621,26	0,17	0,04	0,09
iranianchurn.csv	6.659,39	17.020.532.476,04	0,68	-0,11	0,10
optdigits.data	244,49	3.723.957,81	0,18	0,66	0,11
pendigits.data	561,13	50.673.603,68	0,29	0,57	0,10
redwine.csv	146,89	192.244,12	0,41	0,00	0,07
segmentation.data	970,08	14.018.118,21	0,32	0,33	0,08
whitewine.csv	435,21	1.394.019,88	0,31	0,01	0,11
yeast.data	1,74	46,21	0,17	0,14	0,06

STD	Radius	Inertia	Silhouette	Rand Score	Exec. Time
abalone.data	0,29	14,26	0,01	0,03	0,01
drybean.arff	16.822,47	68.610.403.226,82	0,00	0,02	0,02
electricalgrid.csv	0,09	28,33	0,00	0,00	0,01
iranianchurn.csv	2,29	89.955,64	0,00	0,00	0,02
optdigits.data	6,26	39.140,48	0,01	0,05	0,02
pendigits.data	8,64	720.384,69	0,01	0,03	0,01
redwine.csv	32,89	7.309,93	0,02	0,00	0,02
segmentation.data	135,78	316.520,13	0,01	0,05	0,02
whitewine.csv	71,55	26.634,01	0,01	0,00	0,02
yeast.data	0,06	0,50	0,01	0,01	0,02

Tabela 4. Resultados para o algoritmo k-means ($p = 2$)

Mean	Radius	Inertia	Silhouette	Rand Score	Exec. Time
abalone.data	2,69	407,20	0,55	0,65	0,09
drybean.arff	87.915,28	641.749.393.059,40	0,53	0,38	0,11
electricalgrid.csv	8,28	258.619,77	0,17	0,04	0,09
iranianchurn.csv	4.956,32	17.020.573.749,75	0,68	-0,11	0,10
optdigits.data	43,74	3.732.253,64	0,18	0,66	0,11
pendigits.data	161,22	50.739.053,43	0,30	0,55	0,10
redwine.csv	102,48	188.387,25	0,42	0,00	0,08
segmentation.data	617,62	13.970.887,47	0,32	0,32	0,07
whitewine.csv	301,13	1.392.624,93	0,31	0,01	0,11
yeast.data	0,76	46,42	0,17	0,14	0,08

STD	Radius	Inertia	Silhouette	Rand Score	Exec. Time

abalone.data	0,49	16,44	0,01	0,03	0,01
drybean.arff	10.209,86	65.363.470.626,17	0,00	0,01	0,02
electricalgrid .csv	0,09	27,09	0,00	0,00	0,01
iranianchurn .csv	15,30	79.559,76	0,00	0,00	0,01
optdigits.data	1,02	43.154,89	0,01	0,05	0,02
pendigits.data	2,96	816.191,33	0,02	0,04	0,01
redwine.csv	49,75	9.809,87	0,02	0,00	0,01
segmentation .data	101,41	278.218,86	0,02	0,06	0,02
whitewine.csv	45,34	23.491,50	0,01	0,00	0,01
yeast.data	0,03	1,53	0,01	0,01	0,01

Em termos do tempo de execução, pode-se observar uma diferença clara: o algoritmo k-means apresenta valores para essa métrica visivelmente menores. Um detalhe que pode ser observado, ainda, é que, conforme o tamanho das bases de dados cresce, o algoritmo k-centros parece apresentar um aumento maior nos tempos de execução, o que pode ser um indício de um pior comportamento assintótico deste. O desvio-padrão dessa métrica não foi relevante, apresentando uniformidade nos testes.

Os algoritmos em voga abordam o problema da clusterização de maneiras diferentes. O k-centros almeja minimizar o raio máximo dos clusters, enquanto o k-means, a inércia total do sistema. A partir da fórmula para esta métrica, entretanto, extrai-se que a inércia nada mais é que a soma dos raios de cada ponto com relação a seu centro elevados ao quadrado, utilizando a distância de Minkowski com o valor de $p = 2$. Em outras palavras, abstraindo detalhes formais, enquanto o primeiro algoritmo busca minimizar o máximo dentro de um conjunto de valores, o segundo almeja minimizar sua soma total. A grande questão, portanto, é se isso levaria ao mesmo lugar ou quais são os impactos na clusterização de maneira geral. Em termos do raio e da inércia, os resultados estão dentro do esperado: o algoritmo k-centros apresentou, em geral, valores menores para aquele, enquanto o k-means, valores menores para esta. Os desvios-padrão para essas métricas, nos dois casos, foram maiores em comparação às outras, o que pode estar associado à natureza aleatória de alguns dos passos do algoritmo k-centros e, possivelmente, do algoritmo k-means (não há, entretanto, confirmação para esta afirmação), além apenas da maior ordem de grandeza em que elas se encontram. Não houve nenhum resultado discrepante, entretanto. O interessante reside na análise das métricas de avaliação da qualidade da clusterização: a silhueta e o índice de Rand. Enquanto, em geral, houve uniformidade quanto aos casos para os quais os resultados foram bons ou ruins, o algoritmo k-means apresentou valores melhores para ambas em quase todas as situações de teste, o que indica, pelo menos sob a óptica dessas duas métricas, que a clusterização por ele construída é mais coesa e mais próxima da classificação real das instâncias.

Entre os valores de p não houve diferenças significativas. Pode ser observado que o algoritmo k-means se manteve um pouco mais consistente com a variação quando comparado ao algoritmo k-centros, além de a métrica do raio ter sido aquela mais afetada (é significativamente maior para $p = 1$). Entretanto, não foi extraído nenhum conhecimento mais profundo sobre o impacto desse parâmetro.

6. Conclusões

O presente trabalho explorou de maneira prática algoritmos aproximativos dentro do domínio da ciência de dados, abordando a questão da clusterização com um algoritmo 2-aproximativo para o problema dos k-centros. Dada uma implementação, testes foram realizados e comparados a uma outra abordagem, k-means, por meio de métricas que avaliam a clusterização de diferentes maneiras. Aqui, retoma-se os resultados obtidos: para as métricas escolhidas, a abordagem k-means apresentou melhores valores em comparação à aproximação por k-centros, o que levanta a questão acerca de se sua perspectiva para o problema - minimizar a inércia, e não o raio -, não seria melhor de maneira geral. No caso específico das métricas adotadas, das bases de dados escolhidas, dos parâmetros definidos e do ambiente executado, observou-se que sim, mas é extremamente difícil generalizar essa ideia para o universo massivo de instâncias que existiram, existem e poderão existir. Novamente, reitera-se a riqueza inerente a essa porção da ciência de dados em termos das possibilidades que podem ser exploradas, dos resultados que podem ser obtidos e da necessidade por ponderação quanto aos recursos investidos e à qualidade do produto recuperado.

Referências Bibliográficas

[1]Nash,Warwick, Sellers,Tracy, Talbot,Simon, Cawthorn,Andrew, and Ford,Wes. (1995). Abalone. UCI Machine Learning Repository. https://doi.org/10.24432/C55C7W.

[2]Dry Bean Dataset. (2020). UCI Machine Learning Repository. https://doi.org/10.24432/C50S4B.

[3]Arzamasov,Vadim. (2018). Electrical Grid Stability Simulated Data . UCI Machine Learning Repository. https://doi.org/10.24432/C5PG66.

[4]Iranian Churn Dataset. (2020). UCI Machine Learning Repository. https://doi.org/10.24432/C5JW3Z.

[5]Alpaydin,E. and Kaynak,C.. (1998). Optical Recognition of Handwritten Digits. UCI Machine Learning Repository. https://doi.org/10.24432/C50P49.

[6]Alpaydin,E. and Alimoglu,Fevzi.. (1998). Pen-Based Recognition of Handwritten Digits. UCI Machine Learning Repository. https://doi.org/10.24432/C5MG6K.

[7, 9]Cortez,Paulo, Cerdeira,A., Almeida,F., Matos,T., and Reis,J.. (2009). Wine Quality. UCI Machine Learning Repository. https://doi.org/10.24432/C56S3T.

[8]Image Segmentation. (1990). UCI Machine Learning Repository. https://doi.org/10.24432/C5GP4N.

[10]Nakai,Kenta. (1996). Yeast. UCI Machine Learning Repository. https://doi.org/10.24432/C5KG68.