Algoritmo aproximativo para o problema de K-centros

Igor Rahzel Colares Galdino

[1] Departamento de Ciência da Computação
Universidade Federal de Minas Gerais – Belo Horizonte, MG – Brasil

igorrahzel@ufmg.br

Resumo. *Este relatório descreve a implementação de um algoritmo 2-aproximado para o problema de K-centros, bem como a sua avaliação através de métricas como: raio da solução, coeficiente de silhueta, índice de Rand ajustado(ARI) e tempo de execução. Essas métricas serão avaliadas para 10 conjuntos de dados diferentes, para os quais o algoritmo realiza 30 testes para cada um deles. Por fim o algoritmo aproximativo implementado também será comparado ao algoritmo de K-centros disponível no Scikit Learn.*

1. Introdução

O problema em questão consiste em encontrar *K* centros em um conjunto de dados de *n* pontos, de modo que os pontos estejam o mais próximo possível de seus centros, em outras palavras, busca-se minimizar a distâncias entre os pontos e seus respectivos centros.

Esse problema pertence à classe NP-difícil,ou seja, não se conhece nenhum algoritmo determinístico com tempo polinomial para o mesmo. Uma forma de abordar problemas pertencentes à essa classe é através de algoritmos aproximativos, os quais executam em tempo polinomial, porém retornam soluções com um fator de qualidade, o qual é possível de se estimar.

Neste relatório vamos avaliar o desempenho de um algoritmo 2-aproximado para o problema de K-centros, isto é, as soluções desse algoritmo são no máximo duas vezes pior que a resposta ótima. Para fazer tal avaliação foram utilizadas métricas como o raio da solução, coeficiente de silhueta, índice de Rand ajustado(ARI) e o tempo de execução, cada um desses parâmetros será explicado nas próximas seções do relatório.

Foi também implementado o algoritmo de K-centros disponível no Scikit Learn, para que esse fosse comparado diretamente ao algoritmo aproximativo desenvolvido, ambos os algoritmos foram implementados na linguagem python3.

2. Métodos e Métricas

Nesta seção será apresentada um descrição das métricas e métodos utilizados para a avaliação do desempenho do algoritmos.

2.1. Métricas

Inicialmente é importante mencionar que a métrica de distância utilizada no algoritmo foi a distância de Minkowski, a qual é definida da seguinte forma:

$$d(x, y) = (\sum_{i=1}^{n} |x_i - y_i|^p)^{\frac{1}{p}}$$

onde *x* e *y* são pontos, os quais deseja-se calcular a distância entre eles e *p* é dito ser a ordem da distância de Minkowski.

A primeira métrica/parâmetro a ser explorada é o raio da solução, ou raio máximo dos clusters. O raio de um cluster é definido como sendo a maior distância entre um ponto qualquer daquele cluster e seu centro, como mencionado anteriormente o objetivo do problema é minimizar tal distância. Consequentemente o raio da solução é o maior valor de raio obtido ao se avaliar todos os clusters.

O coeficiente de silhueta avalia o quão coesa é a solução dada pelo algoritmo, ou seja, essa é uma medida que avalia a similaridade dos objetos designados a um mesmo cluster, os valores dessa métrica variam dentro do intervalo [-1,1], quanto mais próximo de 1 é o valor dessa métrica temos que os clusters estão mais distantes entre si e estão melhor definidos. Essa métrica é definida da seguinte forma:

$$\frac{b-a}{max(a,b)}$$

onde *a* é a distância média entre os pontos de um mesmo cluster e *b* é a distância entre um ponto e o cluster mais próximo que não inclui esse ponto. No algoritmo aproximativo foi utilizada a implementação do coeficiente de silhueta disponível na biblioteca sklearn, essa função retorna a média do coeficiente de silhueta dentre todos os pontos do conjunto de dados.

O Índice de Rand Ajustado, essa métrica calcula a similaridade entre dois clusters de um mesmo conjunto de dados, o ARI é um aprimoramento do Índice de Rand(RI) que basicamente verifica o número de pares de pontos que foi classificado de forma igual e diferente nos dois clusters avaliados. O ARI diferentemente do RI, leva em conta a aleatoriedade esperada nos agrupamentos quando os agrupamentos têm tamanhos variados. Essa métrica assim como o coeficiente de silhueta, assume valores no intervalo [-1,1], onde quanto mais próximo de 1 for o valor obtido, melhor é a concordância entre os agrupamentos.

Por fim, o último parâmetro utilizado na avaliação do desempenho do algoritmo foi o tempo de processamento de cada execução. Nesse trabalho prático foi avaliado o funcionamento do algoritmo em 10 conjuntos de dados diferentes e foram realizadas 30 execuções do algoritmo para cada um deles.

2.2. Métodos

O algoritmo tem um funcionamento bem simples, inicialmente é calculada a matriz de distância entre os pontos seguindo a métrica de Minkowski com o parâmetro *p* desejado.

O próximo passo é a execução do algoritmo 2-aproximativo, o qual inicialmente escolhe aleatoriamente um dos pontos do conjunto de dados como sendo um centro, então enquanto o número de centros escolhidos pelo algoritmo for menor que *k*, o algoritmo irá selecionar como um novo centro o ponto que possui a maior distância para seu centro.

O algoritmo é executado trinta vezes e a cada execução são salvos os valores das métricas desejadas em uma matriz. Ao final são utilizados os métodos *mean* e *std* da biblopteca *numpy* para obter a média e o desvio padrão de cada uma das métricas.

3. Implementação

O algoritmo aproximativo foi desenvolvido na linguagem python na versão 3.10.9. Para melhorar a modularização do código foram implementadas três classes sendo elas *minkowski*, *Kmeans_aprox*, *metrics*.

A classe *minkowski* implementa os métodos relacionados a distância de Minkowski, sendo estes a própria função de distância e a matriz de distância dos pontos do conjunto de dados.

Na classe *Kmeans_aprox* é onde está implementado o algoritmo aproximativo, essa implementação é realizada no método *kmeans*. Além desse método, essa classe também possui funções auxiliares para guardar os conjuntos de dados em uma matriz, bem como obter a matriz de distância dos pontos que compõem esse conjunto com o auxílio dos métodos implementados na classe *minkowski*.

Já a classe *metrics* é responsável pela implementação das métricas de avaliação das soluções do algoritmo através dos métodos *solution_radius*, *compute_silhouette*, *compute_ARI* que retornam o raio da solução, a silhoueta e o ARI respectivamente, vale mencionar que os métodos *compute_silhouette* e *compute_ARI*, utilizam funções disponíveis na biblioteca *sklearn* para realizar o cálculo dessas métricas.

As classes descritas acima estão implementadas nos arquivos *minkowski.py*, *Kmeans_aprox.py*, *metrics.py*, mas é no arquivo *main.py* onde são realizadas as leituras dos inputs e realizadas as chamadas dos métodos de classe para a execução devida do algoritmo.

É nesse arquivo onde também é feito o cálculo do tempo de execução do algoritmo através do uso da biblioteca *time*. É importante mencionar que o no primeiro dos trinta testes realizados para cada um dos dez conjuntos de dados selecionados é computada a matriz de distância, tal operação é contabilizada no tempo de execução do algoritmo.Para a execução do algoritmo no Linux basta executar o seguinte comando:

python3 main.py -f file -k num_clusters -p p_val -c col -d delimitador -sh num_linhas

onde os valores os parâmetros *file* corresponde ao arquivo contendo o conjunto de dados, *num_clusters* é o número de clusters desejado, *p_val* é valor da ordem da distância de Minkowski, *col* é a coluna do conjunto de dados do atributo classe, *num_linhas* é o número de linhas que se deseja saltar caso o conjunto de dados possua um cabeçalho, caso contrário esse número deve ser zero.

Já o arquivo *K_means_sklearn.py* é responsável por implementar o algoitmo de k-centros disponível na biblioteca *sklearn*, bem como computar as métricas da solução retornada por esse algoritmo. Esse arquivo é executado da mesma forma que o anterior, basta substituir *main.py* por *K_means_sklearn.py* no comando de execução.

4. Descrição dos Experimentos

Os conjuntos de dados utilizados no teste do algoritmo foram retirados do *UCI Machine Learning Repository*,foram selecionados conjuntos de dados contendo no mínimo 700 instâncias, e sendo eles exclusivamente numéricos.

Os conjuntos de dados selecionados foram:Spambase[1],Glioma Grading Clinical and Mutation Features Dataset[2], Blood Transfusion Service Center[3], Wireless Indoor Localization[4], banknote authentication[5], Diabetic Retinopathy Debrecen Data Set[6], South German Credit (UPDATE)[7], Wine Quality(do qual foram utilizados os dois arquivos)[8][9] e Pen-Based Recognition of Handwritten Digits[10].

Para cada um dos experimentos foi utilizado um valor diferente para a ordem da distância de Minkowski, o valor para esse parâmetro foi inicialmente 1 para o experimento realizado com o primeiro conjunto de dados e a cada novo conjunto ele foi acrescido em uma unidade.

O número de clusters escolhidos para cada um dos conjuntos de dados foi determinado com base nas informações disponíveis em suas descrições. Tais informações também permitiram a extração do valor verdade para os rótulos(ground truth label) dos clusters, esses valores são empregados no cálculo do *ARI*. É importante mencionar que nos conjuntos de dados disponibilizados em Wine Quality foi considerado um número menor de clusters do que aquele "sugerido" pela descrição do mesmo, apesar dos vinhos serem agrupados de acordo com as suas notas que variam de 0 até 10, ao analisar cada um dos conjuntos de dados verificou-se que algumas das notas não haviam sido atribuídas a nenhum dos vinhos, o que justificou um uma escolha menor em relação ao número de agrupametos.

5. Apresentação e Análise dos Resultados

Abaixo são apresentadas as tabelas contendo os resultados das métricas descritas previamente no relatório para o algoritmo 2-aproximado, bem como para o algoritmo o qual utilizou a implementação disponível no Scikit Learn.

Os valores obtidos para a média dos raios de cada um dos datasets é muito próximo nas duas tabelas principalmente ao levar em conta o desvio padrão obtido. Em alguns conjuntos de dados os valores obtidos foram melhores no algoritmo aproximativo do que na implementação do *sklearn*.

Os valores para o coeficiente de silhueta também se mostraram bem próximos em ambos os algoritmos, em alguns casos o algoritmo aproximativo foi que produziu o melhor resultado com respeito a essa métrica. A diferença média entre os coeficientes de silhueta foi de 0.092.

Já o ARI, foi melhor avaliado ao se utilizar a biblioteca do Scikit Learn, sendo a média do valor obtido superior em todos os conjuntos de dados testados. A média da diferença entre os valores de ARI obtidos foi de 0.096.

Ao se analisar os resultados obtidos para o coeficiente de silhueta e o ARI em conjunto para as duas tabelas, percebe-se que nem dos dois algoritmos realizou agrupamentos significativamente melhores que o outro.

O tempo de execução do algoritmo aproximativo foi maior em todos os testes, isso muito provavelmente é uma consequência do cálculo da matriz de distância, pois o número de computaçõs necessárias para construir essa matriz é de ordem quadrática. Tal fator pode favorecer a escolha da implementação do scikit learn, uma vez que ambos algoritmos performaram de maneira equivalente quando consideradas as outra métricas.

Table 1. Média e Desvio Padrão Obtidos no Algoritmo Aproximativo

DataSet	Raio		silhueta		ARI		Tempo			
	std	média	std	média	std	média	std	média	k	p
1	258.8	15659.0	0.003	0.96	0.00	0.00	13.73	2.77	2	1
2	3.76	31.39	0.059	0.48	0.073	0.10	1.70	0.36	2	2
3	561.74	5033.3	0.01	0.83	0.00	0.02	0.92	0.20	2	3
4	1.70	24.6	0.07	0.28	0.13	0.38	6.49	1.53	4	4
5	1.64	12.79	0.02	0.47	0.02	0.06	3.20	0.66	2	5
6	15.65	174.6	0.02	0.64	0.00	0.00	2.4	0.5	2	6
7	653.4	8088.9	0.00	0.73	0.00	0.04	1.7	0.37	2	7
8	1.9	33.4	0.02	0.44	0.00	0.00	4.47	1.05	7	8
9	3.95	62.2	0.04	0.32	0.00	0.01	39.2	8.2	8	9
10	1.97	95.1	0.03	0.16	0.04	0.31	94.9	19.4	10	10

Table 2. Média e Desvio Padrão Obtidos na Implementação do Sklearn

DataSet	Raio		silhueta		ARI		Tempo			
	std	média	std	média	std	média	std	média	k	p
1	2835.4	16620.6	0.03	0.86	0.01	0.03	0.15	0.73	2	1
2	0.09	25.67	0.00	0.58	0.00	0.20	0.02	0.05	2	2
3	131.1	7812.6	0.00	0.69	0.00	0.07	0.01	0.05	2	3
4	0.34	29.2	0.02	0.4	0.05	0.87	0.02	0.18	4	4
5	0.01	12.49	0.00	0.43	0.00	0.04	0.02	0.10	2	5
6	2.85	260.3	0.00	0.43	0.00	0.00	0.02	0.08	2	6
7	17.86	10017.6	0.00	0.72	0.00	0.05	0.02	0.07	2	7
8	53.1	96.1	0.01	0.39	0.00	0.00	0.02	0.14	7	8
9	28.03	229.8	0.00	0.30	0.00	0.01	0.062	0.61	8	9
10	2.09	86.0	0.01	0.29	0.03	0.57	0.17	1.7	10	10

6. Conclusão

Este trabalho mostrou que mesmo que um determinado problema seja da classe NP, isso não significa que ele não deve ser abordado pelo fato de não se conhecer um algoritmo polinomial determinístico para o mesmo, pois através de algoritmos aproximativos podemos muitas vezes obter soluções boas o suficientes para muitos casos.

Embora o algoritmo implementado seja 2-aproximado, temos os resultados obtidos para ele nos casos de teste realizados foi no geral tão boa quanto a aquela obtida utilizando-se a biblioteca do *Scikit Learn*, em alguns casos o raio obtido foi até melhor. A diferença nos resultados das métricas de coefiente de silhueta e ARI, como mostrado na

seção anterior foi inferior a um décimo. O que mostra uma similaridade entre as soluções apresentadas, quando feita uma análise global.

A grande desvantagem do algoritmo aproximativo não foi em relação ao resultados retornados, mas sim o seu tempo de execução que foi consideravelmente pior em alguns casos quando comparado ao outro algoritmo utilizado.

7. References

[Keiblerg and Tardos 2006]

References

[Keiblerg and Tardos 2006] Keiblerg, J. and Tardos, E. (2006). Algorithm desing. In *Ciências da Computação*. Pearson/Addison-Wesley.