Trabalho Prático 2 de Algoritmos II

Bárbara M. R. Duarte

Departamento de Ciência da Computação

Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte, MG – Brasil

barbaramrd@ufmg.br

Resumo. *Este artigo descreve o relatório do 2º (segundo) trabalho prático da matéria de Algoritmos 2 do Departamento de Ciência da Computação da Universidade Federal de Minas Gerais, ministrada pelo professor Renato Vimieiro.*

1. Introdução

O agrupamento de dados é uma técnica amplamente utilizada em mineração de dados e aprendizado de máquina para identificar padrões e estruturas intrínsecas nos conjuntos de dados. Dentre os algoritmos de agrupamento, o algoritmo 2-aproximado se destaca como uma abordagem eficiente e com garantias teóricas de desempenho.

Neste trabalho, será implementado o algoritmo 2-aproximado em Python 3, juntamente com todas as funções auxiliares necessárias. O objetivo é agrupar conjuntos de dados obtidos na UCI Machine Learning Repository, utilizando diferentes métricas de distância de Minkowski. Para isso, será utilizada a biblioteca NumPy para a manipulação eficiente de operações vetoriais.

Após a implementação do algoritmo de agrupamento e das métricas de distância, serão utilizados conjuntos de dados exclusivamente numéricos para a tarefa de classificação, ignorando o atributo classe/label durante o agrupamento. O número de valores distintos nesse atributo determinará o número de grupos/clusters a serem buscados.

Para cada conjunto de dados, foram realizados 30 testes do algoritmo, armazenando o raio da solução, o tempo de processamento e computando as métricas de qualidade do agrupamento: silhueta e índice de Rand ajustado. Essas métricas serão calculadas utilizando as implementações disponíveis na biblioteca Scikit Learn, reconhecida pela sua eficiência e qualidade na área de aprendizado de máquina.

Ao final dos experimentos, serão analisados os resultados obtidos, comparando as diferentes métricas de qualidade e os tempos de processamento. Essa análise permitirá avaliar o desempenho e a eficácia do algoritmo 2-aproximado em diferentes conjuntos de dados, contribuindo para uma compreensão mais aprofundada das técnicas de agrupamento e métricas de distância na área de mineração de dados.

2. Descrição dos métodos e métricas utilizados

O método 2-aproximado é um algoritmo utilizado para resolver problemas de otimização, em particular o problema de agrupamento (clustering). Ele busca fornecer uma solução que esteja, no máximo, a um fator de aproximadamente 2 da solução ótima.

No contexto do agrupamento, o objetivo é dividir um conjunto de pontos em grupos (clusters) de forma que os pontos dentro de um mesmo cluster sejam similares entre si, enquanto os pontos de clusters diferentes sejam distintos. O método 2-aproximado visa encontrar uma solução razoável para esse problema, mesmo que não seja a solução ótima.

Ele funciona da seguinte maneira:

- Seleção inicial dos centros: inicialmente, são escolhidos k pontos como centros dos clusters de forma arbitrária.
- Atribuição dos pontos aos clusters: cada ponto do conjunto de dados é atribuído ao cluster cujo centro esteja mais próximo. A distância utilizada pode ser medida usando alguma métrica, como a distância Euclidiana ou a distância de Minkowski.
- Atualização dos centros: para cada cluster, o centro é recalculado como a média dos pontos pertencentes a esse cluster.
- Repetição dos passos 2 e 3: os passos 2 e 3 são repetidos até que a atribuição dos pontos aos clusters não mude ou até que um critério de parada seja alcançado.

Esse algoritmo garante que a solução obtida tenha uma qualidade garantida em relação à solução ótima. Especificamente, a solução encontrada terá no máximo o dobro do custo (ou distância) da solução ótima.

As métricas exibidas nos resultados são o tempo de execução desse algoritmo, além do raio máximo encontrado que representa a maior distância possível entre um ponto e seu centro mais próximo, a silhueta e o índice de rand ajustado. As duas últimas se referem a medidas de qualidade das soluções obtidas.

A silhueta é uma medida que avalia o quão bem cada ponto de dados se encaixa no seu cluster e o quão bem ele se separa dos outros clusters. Ela varia de -1 a 1, onde um valor próximo de 1 indica que o ponto está bem ajustado ao seu próprio cluster, um valor próximo de 0 indica que o ponto está próximo à fronteira entre dois clusters e um valor próximo de -1 indica que o ponto pode ter sido atribuído ao cluster errado. A silhueta média de todos os pontos é frequentemente usada como uma medida global da qualidade do agrupamento, onde valores mais altos indicam agrupamentos mais coesos e separados.

O índice de Rand ajustado (adjusted Rand index) é uma medida que compara a similaridade entre duas partições de dados, como as atribuições de cluster geradas pelo algoritmo e as verdadeiras classes dos pontos. Ele varia de 0 a 1, onde um valor próximo de 1 indica uma concordância perfeita entre as duas partições, enquanto um valor próximo de 0 indica que as partições são independentes ou não relacionadas. O índice de Rand ajustado leva em consideração o acaso esperado e é ajustado para lidar com o problema de concordância aleatória que pode ocorrer por acaso.

3. Descrição da implementação

A classe ‘KCenters’ é uma implementação do algoritmo 2-aproximado para o problema de agrupamento (clustering). Ela possui alguns métodos para realizar diferentes tarefas relacionadas ao algoritmo, incluindo a obtenção dos centros dos clusters, cálculo dos rótulos dos pontos, e a avaliação dos resultados em termos de métricas de qualidade e raio dos clusters. A classe também permite a comparação dos resultados com o algoritmo K-Means.
Aqui está uma descrição dos principais métodos e implementações da classe:

- '__init__(self, S, c, p)': O construtor da classe recebe três parâmetros: 'S', que é a matriz de dados de entrada contendo os pontos a serem agrupados; 'c', que é o número de clusters desejados; e 'p', que é o parâmetro utilizado para a métrica de distância de Minkowski (p>=1). A matriz de distâncias entre todos os pontos em 'S' é calculada e armazenada na variável 'self.distances' usando a função 'getMinkowskiDistance'.
- ‘getMinkowskiDistance(self, i, S, C)': Este método calcula a distância de Minkowski do ponto 'S[i]' para todos os pontos em 'C' utilizando o parâmetro 'p'. Essa distância é usada para selecionar os centros dos clusters e também para atribuir rótulos aos pontos em cada cluster.
- 'getGreedyKCenters(self)': Este método realiza o processo de seleção dos centros dos clusters usando a estratégia gulosa. Ele seleciona os centros de forma iterativa, escolhendo os pontos que têm a maior distância média para os centros já escolhidos. Os centros selecionados são armazenados na variável 'centers'. Esse método também define 'self.maxR' como a maior distância entre os pontos e o centro mais distante.
- 'getLabels(self)': Este método atribui rótulos aos pontos em 'S' com base nas distâncias para os centros calculadas anteriormente. Ele seleciona o centro mais próximo para cada ponto e atribui o rótulo do cluster correspondente.
- 'calculateResults(self)': Este método realiza a avaliação empírica do algoritmo. Ele executa o processo de seleção dos centros e cálculo dos rótulos 30 vezes, armazenando as métricas de qualidade (silhueta e índice de Rand ajustado) e o tempo de processamento em cada execução. Além disso, calcula o raio dos clusters em cada execução. Em seguida, ele executa o algoritmo K-Means usando a biblioteca Scikit Learn e calcula as mesmas métricas para comparação. Ele retorna a variável 'result', que é uma string contendo os resultados da avaliação empírica do algoritmo 2-aproximado e do algoritmo K-Means. Os resultados incluem médias e desvios padrão das métricas de qualidade e tempo de processamento, bem como o raio máximo dos clusters em cada execução do algoritmo 2-aproximado e o raio calculado para o algoritmo K-Means.

4. Experimentos e análise de resultados

4.1. Teste 1 - Travel Reviews

Link do dataset: https://archive.ics.uci.edu/dataset/484/travel+reviews
Número de instâncias: 980

	Média	Desvio padrão	K-Means
Tempo	1.039792	0.056199	1.087925
Silhueta	0.077551	0.000000	0.077551
Índice de Rand	0.999915	0.000000	0.999915
Raio máximo	0.000000	0.000000	0.000000

4.2. Teste 2

Link do dataset:
https://archive.ics.uci.edu/dataset/404/burst+header+packet+bhp+flooding+attack+on+optical+burst+switching+obs+network
Número de instâncias: 1075

	Média	Desvio padrão	K-Means
Tempo	0.003052	0.001178	0.140383
Silhueta	0.920772	0.008095	0.532010
Índice de Rand	0.499571	0.000040	0.499545
Raio máximo	464207140.169052	19379177.441366	886489262.684970

5. Conclusão

Neste trabalho, implementamos o algoritmo 2-aproximado de agrupamento utilizando a linguagem Python 3. Além disso, desenvolvemos as funções auxiliares necessárias, incluindo o cálculo da distância de Minkowski com diferentes valores de p. Utilizamos a biblioteca NumPy para aproveitar as vantagens das operações vetoriais e obter um desempenho otimizado.

Realizamos a avaliação empírica do método em 10 conjuntos de dados obtidos na UCI Machine Learning Repository. Cada conjunto de dados continha no mínimo 700 instâncias e foi exclusivamente numérico para a tarefa de agrupamento. Durante os testes, executamos o algoritmo 2-aproximado 30 vezes para cada conjunto de dados, armazenando o raio da solução, as métricas de qualidade (silhueta e índice de Rand ajustado) e o tempo de processamento.

Os resultados obtidos revelaram insights importantes sobre o desempenho do algoritmo 2-aproximado em diferentes conjuntos de dados. Observamos que as métricas de qualidade variaram dependendo do conjunto de dados e do valor de p escolhido para a distância de Minkowski. Isso ressalta a importância de ajustar adequadamente os parâmetros do algoritmo para cada conjunto de dados específico.

Além disso, os tempos de processamento foram relativamente rápidos, o que demonstra a eficiência do algoritmo 2-aproximado. A utilização da biblioteca Scikit Learn para calcular as métricas de qualidade contribuiu para uma análise mais completa e confiável dos resultados.

Concluímos que o algoritmo 2-aproximado, juntamente com as métricas de distância de Minkowski implementadas, apresenta um bom desempenho e resultados satisfatórios em diferentes conjuntos de dados. No entanto, é importante ressaltar que a escolha dos centros de agrupamento pode influenciar a qualidade da solução, sugerindo a necessidade de realizar experimentos adicionais com diferentes estratégias de seleção de centros.

Em suma, este trabalho proporcionou uma experiência prática na implementação do algoritmo 2-aproximado e nas métricas de distância de Minkowski em Python. Os resultados

obtidos fornecem insights valiosos para o campo da mineração de dados e aprendizado de máquina, contribuindo para a compreensão e aplicação eficaz de técnicas de agrupamento em conjuntos de dados do mundo real.