Relatório Trabalho Prático 2 - Algoritmos 2

Salvador Cândido da Silva Júnior, Victor Augusto Hon Fonseca

Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)

Abstract. This report discusses experiments conducted on different implementations of the K-centers algorithm and the KMeans algorithm, comparing the performance of both for different datasets, demonstrated through indicators.

Resumo. Este relatório aborda sobre experimentos realizados em diferentes implementações do algoritmo K-centros e no algoritmo K-Means, comparando o desempenho de ambos para diferentes dados, demonstrado através de indicadores.

1. Introdução

Uma das áreas que mais vem crescendo no setor de tecnologia é a área de Inteligência Artificial, e, por consequência, a área de aprendizado de máquinas.

No presente trabalho, foram trabalhados aspectos da implementação de algoritmos aproximativos do setor mencionado. Mais especificamente, buscou-se comparar implementações do algoritmo k-centros, útil em tarefas de agrupamento do setor, com algoritmos clássicos de agrupamento, no caso, o K-Means.

2. Métodos e Métricas usadas

2.1 Métodos usados

Neste trabalho utilizou-se de duas implementações diferentes do algoritmo K-centros e o algoritmo K-means.

Na primeira abordagem do K-centros, busca-se descobrir um valor para ser considerado o raio máximo da solução ótima com base em uma busca binária. Esse valor é gerado com base em diferentes valores de porcentagem que alteram o número de refinamentos a serem realizados, Inicialmente descobre-se a distância máxima entre dois "pontos" do conjunto e a considera como limite superior do raio, sendo o limite inferior igual a 0. Após isso, realizam-se execuções do algoritmo principal reduzindo-se o limite superior para a média dos limites, ou aumentando o limite inferior para ela até que a diferença entre ambos seja menor ou igual à porcentagem passada como parâmetro do limite superior inicial, ponto em que se assume que os limites convergiram em um único valor, o qual se estabeleceu como o limite superior.

No algoritmo principal da primeira abordagem do K-Centros, tendo-se acesso ao que se assume como raio da solução ótima, seleciona-se arbitrariamente um "ponto" e passa-se a considerá-lo como centro, e remove-se do conjunto todos os "pontos" que estiverem a uma distância máxima do dobro do raio em questão, processo repetido até que não haja mais pontos no conjunto. Se o número de centros for menor ou igual ao número de grupos passado como parâmetro ("k"), tem-se solução para o problema, caso contrário, não. A partir disso, percebe-se, claramente, que trata-se de um algoritmo 2-aproximado, sendo a distância usada o dobro da encontrada na solução ótima, que, com sucesso, é capaz de agrupar dados, estabelecendo centros, com base em uma distância pré-determinada, até que não hajam mais "pontos" (conjuntos de dados disponíveis).

Já na segunda abordagem do algoritmo K-centros, caso o parâmetro "k" passado seja maior ou igual ao tamanho do grupo de pontos passado, retorna-se como solução o

próprio grupo de “pontos”. Caso contrário, seleciona-se um “s” arbitrário e o adiciona no grupo de centros. Analisa-se para cada ponto qual a menor distância entre ele e um dos pontos do grupo de centros e adiciona-se ao grupo de centros o ponto cuja menor distância entre ele e um dos do grupo de centros for maior, repetindo-se o processo enquanto o tamanho desse grupo for menor que “k”, retornando-se ao final, o grupo de centros. Esse algoritmo é 2-aproximado, retornando um agrupamento com raio, no pior caso, duas vezes o ótimo, pois supondo-se que a solução ótima seja “r*”, a maior distância entre um ponto e o centro ao qual foi associado é, evidentemente, “r*”. Caso a solução (r) encontrada pelo algoritmo seja maior que o dobro da solução ótima, haverá contradições, pois, existem “k” + 1 pontos cuja distância entre si é, no mínimo, “r”, já que, se não, a solução encontrada seria ótima. Entretanto, na solução ótima, pelo menos dois desses pontos devem pertencer ao mesmo cluster. Assim, vai existir um cluster com diâmetro maior que o dobro do raio da solução ótima, indicando contradição, o que implica que a solução encontrada é, no máximo, o dobro de r*.

Por fim, o último algoritmo utilizado é o K-Means, clássico para o problema de agrupamento, ele define uma característica principal para cada “cluster”. A partir disso, calcula-se a distância de cada dado para a base central de cada “cluster” e o novo dado será associado ao “cluster” que estiver mais próximo dele, os centróides devem estar posicionados na média dos pontos pertencentes ao seu cluster.

2.2 Métricas usadas

Como métricas usadas, utilizou-se a distância de Minkowski, para realizar o cálculo da distância entre dois conjuntos de valores, que se trata de uma fórmula que dependendo do parâmetro “p” utilizado, faz com que calcule-se diferentes distâncias, sendo a distância de Manhattan entre os conjuntos mencionados (para p = 1), ou a distância Euclidiana (para p = 2), por exemplo.

Além disso, utilizou-se as métricas silhueta e índice de Rand ajustado. A primeira corresponde a uma métrica que analisa o quão bem um “ponto” (conjunto de valores, como mencionado anteriormente) se encaixa no grupo (“cluster”) ao qual foi atribuído, essa métrica vai de -1 a 1, um valor próximo de -1 indica uma atribuição errada de “cluster”, um valor próximo a 0 indica que clusters podem estar tendo muitas regiões comuns uns com os outros e um valor próximo a 1 indica uma atribuição apropriada. O índice de Rand ajustado, por sua vez, corresponde a uma medida a qual analisa a semelhança entre dois agrupamentos de dados, sendo que quanto mais próximo de 1 o valor retornado por ele maior a semelhança entre os conjuntos de dados.

Por fim, foram armazenados o tempo de execução de cada algoritmo e, no caso específico do algoritmo baseado no refinamento de intervalos, uma sequência linear de 5 valores entre 1-25%, que representam variações no número de refinamentos.

3. Descrição da Implementação

Primeiramente, armazena-se o conjunto de “pontos” (conjuntos de valores a serem usados), os quais contém apenas valores numéricos, armazena-se quais são as classificações já fornecidas dos “pontos” e, por fim, um Numpy array de dtype=object “pontos”, sendo cada item dele formado por um inteiro que representa a posição ocupada pelo “ponto” dentro do conjunto inicial e pelo conjunto de dados dele. Após isso, criou-se uma função para ser possível obter a distância de “Minkowski” entre dois “pontos”, usando funções vetoriais da biblioteca Numpy e que recebesse um parâmetro “p”, para que, diferentes tipos de distância pudessem ser calculadas, como a distância de “Manhattan” ou a distância Euclidiana. Também foi criada uma função para se gerar uma matriz que armazena a distância entre cada par de pontos baseando-se em seus índices no conjunto de “pontos” original, com cada subvetor “s” estando associado ao ponto de mesma posição no vetor original e, dentro de cada subvetor, cada elemento “e” estando associado, novamente, ao ponto de mesma posição no vetor original. Assim, o elemento matriz[s][e], corresponde a distância entre os pontos armazenados nos índices “s” e “e” do vetor original de pontos.

Após essa primeira parte, buscou-se implementar a primeira abordagem do algoritmo “K-centros”. Isso foi realizado, inicialmente, implementando-se, primeiramente, a função “removerPtos2r”, a qual recebe como parâmetro um elemento “s” de “pontos”, um Numpy array que contém elementos que associam os “pontos” às suas posições, um parâmetro “r” e a matriz de distâncias e armazena como variável “Slista” uma versão desse array como “list” de “lists”. Por fim essa função percorre os elementos do array original e, caso encontre algum que esteja a uma distância maior do que o dobro de “r” de “s”(consulta realizada através da matriz de distâncias, passando como índices, os índices presentes em cada “ponto”), este elemento é removido de “Slista”, retornando-se uma versão Numpy array de “Slista”. Após isso, implementou-se a função “kCentrosComplicado”, que corresponde à implementação da primeira versão do “K-Centros”, recebendo como parâmetros o “pontos”, um inteiro “k”, a matriz de distâncias e um número “r”. Por fim, implementou-se as funções “descobrirraiomax” que retorna o maior número da matriz de distâncias e a função “descobrirRaio” que segue o raciocínio descrito no segundo parágrafo da seção 2.1.

Depois da versão 1 do algoritmo “K-centros”, buscou-se implementar a segunda versão do algoritmo “K-centros”. Isso foi realizado, inicialmente, implementando-se “maxDist”, uma função que armazena a menor distância de cada ponto a um dos pontos classificados como “centros” e retorna a maior delas, juntamente com o ponto associado a ela. Após isso, implementou-se a função “kCentrosSimples” que segue o raciocínio lógico descrito no quarto parágrafo da seção 2.1.

Após isso, utilizou-se métodos da biblioteca Scikit Learn, para poder implementar o algoritmo K-Means, dentre eles, o “StandardScaler” e o “fit_transform” para normalizar os dados, o método “KMeans”, passando para ele como parâmetros o número de clusters desejados, e 0 e 10, como os parâmetros “random_state” e “n_init”. Além disso, usou-se o método “predict” para que o “KMeans” gera-se rótulos para os dados.

Por fim, utilizou-se dos métodos, também da biblioteca “Scikit Learn”, “silhouette_score” e “adjusted_rand_score” para gerar as métricas silhueta e índice de Rand ajustado, correspondente a cada algoritmo, passando como parâmetros para a primeira, um conjunto de dados que contenha os conjuntos de dados de cada um dos “pontos” e o conjunto de classificações previstas pelos algoritmos para o conjunto de pontos (para o caso do “KMeans” um método pronto “predict”, já retorna esse conjunto, para os demais algoritmos usou-se a função “rotular_pontos”, que classifica cada ponto de acordo com o índice do centro ao qual o ponto está mais próximo) e para a segunda as classificações verdadeiras e as classificações previstas. Dessa forma, a primeira avalia o quão bem foi feita a clusterização do conjunto dos “pontos” e a segunda o quão próximo a clusterização realizada pelo algoritmo ficou da real.

4. Experimentos

4.1 Descrição dos Experimentos

Para cada base de dados foram realizados testes considerando os valores p = 1 e p = 2 para a função de Minkowski. Em relação ao algoritmo K-Centros (1ª abordagem), foram considerados os seguintes valores de porcentagem do parâmetro de largura:

1%, 7%,13%,19%,25%

Além disso, foram realizados testes considerando todas as 10 combinações de valores de p com valores do parâmetro de largura. Para cada uma dessas combinações, foram realizadas 30 execuções do algoritmo.

Sobre o algoritmo K-Centros (2ª abordagem), foram realizadas 30 execuções do algoritmo para cada valor de p.

Também foram realizadas 30 execuções do algoritmo KMeans para cada base de dados.

Foram utilizadas as seguintes bases de dados reais:

Base de dados 1
Link: https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data

Base de dados 2
Link:https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv

Base de dados 3
Link:https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv

Base de dados 4
Link:
https://archive.ics.uci.edu/ml/machine-learning-databases/letter-recognition/letter-recognition.data

Base de dados 5
Link:
https://archive.ics.uci.edu/ml/machine-learning-databases/poker/poker-hand-training-true.data

Base de dados 6
Link: https://archive.ics.uci.edu/ml/machine-learning-databases/cmc/cmc.data

Base de dados 7
Link:https://archive.ics.uci.edu/ml/machine-learning-databases/covtype/covtype.data.gz

Base de dados 8
Link:
https://archive.ics.uci.edu/ml/machine-learning-databases/spambase/spambase.data

Base de dados 9
Link:
https://archive.ics.uci.edu/ml/machine-learning-databases/mammographic-masses/mammographic_masses.data

Base de dados 10
Link:
https://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/german.data

Essas bases de dados reais foram pré-processadas de forma a permitir a sua utilização nos testes.
O primeiro conjunto de 10 datasets sintéticos foi gerado a partir do código disponível no seguinte link:
https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html#sphx-glr-auto-examples-cluster-plot-cluster-comparison-py.
O segundo conjunto de 10 datasets sintéticos foi gerado utilizando-se a função de distribuição normal multivariada disponível na biblioteca NumPy (np.random.multivariate_normal).

4.2 Apresentação dos Resultados

Os resultados podem ser visualizados neste link:
https://docs.google.com/document/d/1VVtdqbd7Nju3Bve0acUwEsbUY7qo3nDPSHSExk7EgQw/edit

Neste link encontram-se todas as tabelas com os resultados gerados, correspondentes a cada algoritmo, a cada base de dados e a cada tipo de base de dados

4.3 Análise dos Resultados

Ao realizar a análise dos resultados e realizar mais cálculos para obter uma visão mais geral dos indicadores, o K-Centros Primeira Versão apresentou como média das médias de tempo um valor de 0.023, o K-Centros Segunda Versão apresentou como valor para a mesma categoria o valor 0.0205 e o K-Means média de 0.1888. Já para a categoria Silhueta, o K-Centros Primeira Versão apresentou como média das médias o valor 0.22, o K-Centros Segunda Versão apresentou o valor 0.36465 e o K-Means, 0.4804, como média. Já para a categoria Índice de Rand Ajustado, o K-Centros Primeira Versão apresentou como média das médias o valor 0.13, o K-Centros Segunda Versão 0.25285 e o K-Means, 0.466.

Ao se analisar os dados, percebemos que, embora o K-Centros Primeira Versão tenha apresentado o menor tempo, isso pode não ser um bom indicador isoladamente para medir o desempenho, já que os valores em que não retornou-se solução não foram incluídos na geração das médias das médias de tempos, mas foram incluídos na hora de realizar a divisão pelo número de valores, gerando um menor tempo.

5 Conclusão

Como o KMeans apresentou a maior silhueta e o melhor índice de Rand ajustado, então ele realizou a melhor clusterização de modo geral, embora tenha gasto maior tempo de processamento. Isso era esperado pois se tratava de uma implementação pronta.

O KCentros segunda versão apresentou a melhor silhueta e o melhor índice de rand em relação ao KCentros primeira versão. Isso era esperado pois trata-se de um método mais robusto que sempre retorna uma resposta.