Trabalho Prático 2 - K-Centros

Jorge A. L. Silva[1]

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte, MG – Brazil

jorge.lima2407@gmail.com

Abstract. *This article describes the development of the second practical assignment of the Algorithms 2 course in the first semester of 2023, detailing the algorithm implementation and the conducted experimental analysis, which compares the implemented algorithm with the version provided by the Python library Scikit-Learn[Pedregosa et al. 2011], with an emphasis on comparing the found radius, silhouette, and adjusted Rand index.*

Resumo. *Este artigo descreve o desenvolvimento do segnudo trabalgo prático da disciplina Algoritmos 2 no primeiro semestre de 2023, detalhando a implementação do algoritmo, e a análise experimental realizada, a qual compara o algoritmo implementado com a versão fornecida pela biblioteca Scikit-Learn[Pedregosa et al. 2011] do Python, com ênfase na comparação dos raios encontrados, silhueta e índice de Rand ajustado.*

1. Introdução

A clusterização é um algoritmo de aprendizado de máquina não-supervisionado que visa agrupar um determinado conjunto de dados em grupos os mais homogêneos possíveis, de acordo com alguma métrica. Uma das formas de realizar este agrupamento é através da distância entre estes pontos e o centro do cluster, ou seja, uma instância pertenceria a cluster se a distância entre o centro de este cluster e o ponto em questão for menor do que a distãncia do mesmo ponto e qualquer um dos outros centros.

Tendo em vista o contexto do problema que queremos resolver, as duas questões a seguir são de extrema importância: Qual métrica de distância será utilizada? Como será feita a escolha dos k centros? Neste artigo respondemos essas perguntas da seguinte forma, utilizaremos tanto a distância de Manhattan quanto a Euclidiana como formar de medir a separação entre dois pontos em um espaço multidimensional, e utilizaremos o algoritmo 2-aproximativo K-Centros para realizar a escolha dos centros.

2. Algoritmo dos K-Centros

A escolha dos centros do problema se dá através do algoritmo dos K-Centros, um algoritmo clássico da área de clusterização. Este é uma abordagem simples e eficiente para o problema, e que trás soluções no máximo duas vezes pior do que a solução ótima, ou seja, a distância entre o ponto e o seu centro é no máximo duas vezes a distância entre o mesmo ponto e seu centro ótimo.

A implementação do algoritmo segue os seguintes passos, dado que se tem um raio pré-definido:

1. Computar a distância entre todos os pontos do problema.
2. Escolher um ponto arbitrário no conjunto de pontos.
3. Adicionar este ponto ao conjunto de centros.
4. Encontrar os pontos que estão a, no máximo, um raio de distância deste novo centro e removê-los do conjunto de pontos.
5. Verificar se a quantidade de centros é menor do que k, retornando uma flag que indique que o algoritmo não encontrou solução.
6. Se não existir mais pontos no conjunto de pontos, retorne os centros obtidos, caso contrário, repita os passos 2, 3, 4 e 5.

Devemos ressaltar que, nesta implementação, que o raio da solução deve ser passado como parâmetro para a função. Portanto, para encontrarmos o raio ótimo, realizamos uma busca binária no intervalo $(0, \max(\text{dist}(p_1, p_2))]$, onde p_1 e p_2 pertencem ao conjunto de pontos, até que o passo dado na busca binária seja menor do que 0.00001.

Portanto, ao fim do algoritmo implementado, recebemos como resposta do programa um conjunto de pontos tal que o raio da solução seja o menor possível.

3. Análise Experimental

Uma vez que temos o algoritmo implementado, a análise experimental foi realizada da seguinte forma: primeiro foram selecionados 10 datasets do site UC Irvine Machine Learning Repository, todos contendo apenas variáveis numéricas e feitos para a tarefa de classificação.

O tamanho e conteúdo dos datasets é bastante variado, com entre setecentas e vinte mil instâncias e entre quatro e mil e vinte e quatro features e tratando de assuntos desde feijões até raios gamma em um telescópio.

A partir disso, encontramos um problema, por a features estarem em unidades muito distintas, temos que o raio, em muitos casos, pode ficar muito grande por causa de uma única dimensão em específico, o que diminui a qualidade da clusterização, uma vez que o modelo passa a considerar uma dimensão muito mais do que as outras. Como solução para isso, foi aplicada a Z-Normalização nos dados, subtraindo as features por sua média e dividindo pelo desvio-padrão, de modo que agora todos os dados estão me unidades de desvio-padrão e centralizados em torno da origem.

3.1. Datasets e Nomenclatura

Os datasets utilizados para os testes são os seguintes e estão associados às seguintes linhas das tabelas geradas:

- androgen: QSAR androgen receptor [and 2019]
- banknote: banknote authentication [Lohweg 2013]
- bean: Dry Bean Dataset [bea 2020]
- churn: Iranian Churn Dataset [chu 2020]
- gamma: MAGIC Gamma Telescope [Bock 2007]
- grid: Electrical Grid Stability Simulated Data [Arzamasov 2018]
- letter: Letter Recognition [Slate 1991]
- spam: Spambase [Hopkins and Suermondt 1999]
- wine: Wine Quality [Cortez and Reis 2009]
- yeast: Yeast [Nakai 1996]

3.2. Os Experimentos

Uma vez que temos os dataset definindos, foram iniciados os experimentos. Quatro tipos de experimentos foram feitos, dois utilizando a implementação desenvolvida e outros dois utilizando a implementação da biblioteca Scikit-Learn. Os dois primeiros experimento tinham como diferença a métrica de distância utilizada, um utilizando a distância de Manhattan (Tabela 1) e o outro utilizando a distância Euclidiana (Tabela 2). Já nos dois outros experimentos, utilizamos os dados originais sem nenhum tratamento (Tabela 3) em um e dados Z-Normalizados no outro (Tabela 4). Como o algoritmo dos K-Centros possui alguns passos não-deterministas, cada um dos experimentos foi repetido 30 vezes, e foram inseridos nas tabelas as médias e desvios padrões das estatísticas computadas, ao invés dos dados absolutos, tendo em vista que estas são muito mais relevantes quando queremos ter uma visão geral de como o algoritmo vai se comportar.

Tabela 1. Experimento realizado com a distância de Manhattan

	radius_mean	radius_std	silhouette_mean	silhouette_std	rand_mean	rand_std
gamma	33.938487	1.309827	0.595344	5.192278e-02	0.034154	3.469125e-02
spam	129.162673	5.701737	0.819616	1.110223e-16	0.000234	5.421011e-20
androgen	1649.208846	4.349431	0.755800	1.068501e-02	0.000000	0.000000e+00
bean	22.506517	0.762691	0.275884	9.194612e-02	0.316455	5.197630e-02
letter	16.868088	0.207471	0.071976	1.244360e-02	0.069254	1.714621e-02
wine	16.774464	0.494673	0.146557	4.621738e-02	0.017207	1.593309e-02
banknote	7.722778	0.658137	0.355646	3.001223e-02	0.065604	3.869833e-02
yeast	8.549816	0.233453	0.189604	4.438240e-02	0.061674	3.793915e-02
churn	24.183086	1.256897	0.386853	4.636596e-02	-0.048839	2.034945e-02
grid	21.712243	0.599354	0.078850	6.954548e-03	0.072612	7.605904e-02

Tabela 2. Experimento realizado com a distância Euclidiana

	radius_mean	radius_std	silhouette_mean	silhouette_std	rand_mean	rand_std
gamma	14.937937	0.549490	0.668870	2.841294e-02	0.003355	3.233763e-03
spam	46.951417	0.553610	0.862246	3.330669e-16	0.000234	5.421011e-20
androgen	96.325082	0.000001	0.565924	1.023518e-02	0.000000	0.000000e+00
bean	7.886501	0.254864	0.291823	6.514900e-02	0.227213	7.568036e-02
letter	5.437258	0.072408	0.077679	1.321055e-02	0.065575	1.821653e-02
wine	7.020021	0.218007	0.153880	4.355562e-02	0.008040	1.479622e-02
banknote	4.501636	0.471070	0.351965	3.154882e-02	0.058454	3.168335e-02
yeast	4.880097	0.185093	0.215207	5.053509e-02	0.068886	5.697520e-02
churn	8.529150	0.295214	0.347873	6.374750e-02	-0.047442	3.500418e-02
grid	6.862238	0.201853	0.068466	9.119856e-03	0.055259	7.722139e-02

3.3. Comparações

Nesta seção iremos comparar as formas como os diferentes testes se diferenciam entre si, comparando, portanto, o uso da distância Euclidiana ou de Manhattan, o algoritmo da biblioteca Scikit-Learn e o implementado, e se existe alguma diferença nos resultados ao normalizarmos os dados.

Tabela 3. Experimento realizado com os dados não tratados

	radius_mean	radius_std	silhouette_mean	silhouette_std	rand_mean	rand_std
gamma	666.991693	9.377714e-03	0.431190	2.323665e-04	0.059552	2.812016e-04
spam	13698.120043	1.818989e-12	0.847532	2.220446e-16	0.039417	1.387779e-17
androgen	19.161137	2.650865e-01	0.088566	1.003738e-02	0.000000	0.000000e+00
bean	88274.304090	1.200769e+04	0.534556	2.838435e-03	0.382976	1.572121e-02
letter	15.853660	4.788054e-01	0.146030	3.848559e-03	0.131384	5.089585e-03
wine	244.611834	7.134603e+01	0.338906	3.807741e-03	0.002090	7.247567e-04
banknote	15.458294	9.909342e-03	0.432288	5.551115e-17	0.048538	1.387779e-17
yeast	0.734848	4.043507e-02	0.193728	1.326349e-02	0.142494	6.858538e-03
churn	4946.117524	1.685231e+01	0.677106	4.116353e-04	-0.107402	8.000467e-06
grid	8.249659	6.997597e-02	0.174172	9.799379e-05	0.036700	2.994133e-03

Tabela 4. Experimento realizado com os dados Z-Normalizados

	radius_mean	radius_std	silhouette_mean	silhouette_std	rand_mean	rand_std
gamma	33.938487	1.309827	0.595344	5.192278e-02	0.034154	3.469125e-02
spam	129.162673	5.701737	0.819616	1.110223e-16	0.000234	5.421011e-20
androgen	1649.208846	4.349431	0.755800	1.068501e-02	0.000000	0.000000e+00
bean	22.506517	0.762691	0.275884	9.194612e-02	0.316455	5.197630e-02
letter	16.868088	0.207471	0.071976	1.244360e-02	0.069254	1.714621e-02
wine	16.774464	0.494673	0.146557	4.621738e-02	0.017207	1.593309e-02
banknote	7.722778	0.658137	0.355646	3.001223e-02	0.065604	3.869833e-02
yeast	8.549816	0.233453	0.189604	4.438240e-02	0.061674	3.793915e-02
churn	24.183086	1.256897	0.386853	4.636596e-02	-0.048839	2.034945e-02
grid	21.712243	0.599354	0.078850	6.954548e-03	0.072612	7.605904e-02

3.3.1. Distância de Manhattan vs Euclidiana

Comparando assim as tabelas 1 e 2, podemos observar que os raios encontrados quando usamos a distância de Manhattan são maiores do que os raios encontrados ao utilizar a distância euclidiana, em especial o raio do dataset androgen, o qual originalmente composto por 1024 features binárias, portanto, é natural que a distância encontrada neste, ainda mais quando estamos considerando a distância de Manhattan e que os dados foram normalizados, seja maior do que nos outros casos.

Por outro lado, temos que tanto a silhueta quando o coeficiente de Rand ajustado não apresentaram mudanças significativas com diferentes métricas de distância, o que é um indicador de que o algoritmo funciona de uma maneira relativamente estável independente de qual medida de distância esteja sendo utilizada para o problema.

3.3.2. Scikit-Learn vs Implementação Própria

O primeiro fator essencial de se ressaltar é o tempo de execução. O tempo de execução do algoritmo do Scikit-Learn é significativamente mais rápido do que a implementação própria, além de ocupar um espaço menor de memória. Enquanto na implementação própria temos o custo do algoritmo dominado pela computação da matriz de distâncias, a biblioteca Scikit-Learn usa alguns truques mais inteligentes para que o algoritmo funcione de forma bem mais eficientes. O uso da implementação própria se torna inviável quando

temos uma grande quantidade de instâncias, uma vez que o custo de tempo e de memória é quadrático com base neste, portanto, podemos esperar que datasets com mais de trinta mil linhas levem algumas horas para serem computados e ocupem alguns gigabytes de memória do computador.

Comparando então a tabela 2 com a tabela 4, uma vez que ambos os teste utilizam dados normalizados e a distância euclidiana como métrica, chegamos nas seguintes conclusões: os raios das soluções da implementação própria são menores do que os raios encontrados no Scikit-Learn, ao mesmo tempo, temos os valores de silhueta bem semelhantes na maioria dos casos, exceto nos datasets gamma e spam, nos quais a implementação própria obteve melhores resultados. Por outro lado, temos que o coeficiente de Rand ajustado dos resultados da implementação própria são inferiores em quase todos os casos aos encontrados na função do Scikit-Learn.

3.3.3. Dados Originais vs Normalizados

Como podemos ver nas tabelas 3 e 4, temos que no geral os raios de solução encontrados quando temos os dados originais são muito maiores do que os encontrados quando normalizamos os dados, salvo algumas exceções como o dataset androgen e yeast, os quais tem um raio de solução pequeno com os dados não tratados pois os valores da features são, no geral, pequenos em módulo.

Diferentemente do raio, a silhueta não apresentou um padrão de mudança tão marcante, possuindo instâncias tanto que apresentaram melhores resultados quanto piores, com o caso de androgen, que apresentou uma melhora significativa, e bean, que apresentou uma solução bem inferior.

Quando falamos do coeficiente de Rand ajustado, porém, encontramos um resultado interessante, a normalização dos dados, no geral, trouxe soluções piores do que o uso dos dados originais. Ou seja, a diferença entre os clusteres encontrados e as classes originais das instâncias aumentou ao normalizarmos os dados.

4. Conclusão

Desta forma, este artigo apresentou uma visão geral sobre o clássico algoritmo de clusterização dos K-Centros, analisando seu comportamento em contextos reais e descrevendo a forma como diferanças na implementação podem levar a diferentes resultados.

Por mais que este algoritmo esteja defasado quando o comparamos com outros mais modernos, tendo em vista que ele foi proposto pela primeira vez em 1957 e sua simplicidade tanto de implementação quanto de entendimento e explicabilidade, este algoritmo ainda tem seu lugar na área, servindo como fundamento para o desenvolvimento e entendimento de modelos mais complexos.

Referências

(2019). QSAR androgen receptor. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C53317.

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

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

Arzamasov, V. (2018). Electrical Grid Stability Simulated Data . UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5PG66.

Bock, R. (2007). MAGIC Gamma Telescope. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C52C8B.

Cortez, Paulo, C. A. A. F. M. T. and Reis, J. (2009). Wine Quality. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C56S3T.

Hopkins, Mark, R. E. F. G. and Suermondt, J. (1999). Spambase. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C53G6X.

Lohweg, V. (2013). banknote authentication. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C55P57.

Nakai, K. (1996). Yeast. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5KG68.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Slate, D. (1991). Letter Recognition. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5ZP40.