Segundo Trabalho Prático de Algoritmos 2 - 2023/1

Artur Gaspar da Silva[1]

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

ags2020@ufmg.br

Resumo. *Esse é um relatório do segundo trabalho prático da disciplina de Algoritmos 2 na UFMG. Iremos explicar o problema, descrever os métodos e métricas usadas, descrever a implementação e os experimentos. Os bancos de daos utilizados para os experimentos também serão descritos, com uma análise dos resultados para cada base de dados e uma conclusão.*

1. Introdução ao Problema

O problema abordado nesse trabalho é o problema de clusterização: queremos agrupar os dados em Clusters, tentando otimizar esse agrupamento segundo alguma métrica. No caso desse trabalho, lidamos apenas com dados numéricos, ou seja, que em que cada instância de dado pode ser representada como um ponto num espaço d-dimensional.

A implementação que queremos analisar é um algoritmo 2-aproximado pro problema de k-centros: buscar k centros de clusters, de forma que um ponto a está no cluster de centro b se b é o centro mais próximo de a dentre todos os centros. Dada uma solução, definimos o raio de um cluster como a distância do centro desse cluster pro ponto mais distante dele que ainda pertence ao cluster. Queremos minimizar o maior raio, dentre todos os raios de cluster. Nesse trabalho nos contentamos com medidas de distâcia de Minkowiski: a distância com parâmetro p entre os vetores n-dimensionais v_1, v_2 é definida como $(\sum_{i=0}^{n} |(v_1 - v_2)_i|^p)^{\frac{1}{p}}$, em que $(v_1 - v_2)_i$ é a i-ésima entrada do vetor $v_1 - v_2$.

Para avaliar a implementação, usamos 10 datasets de pelo menos 700 instâncias, com pontos numéricos que gostaríamos de classificar. Compararemos com a implementação de K-means disponível na biblioteca Scikit-Learn [Pedregosa et al. 2011], além de observar as métricas Silhueta e Índice de Rand ajustado, também disponíveis na biblioteca Scikit Learn. Por fim, armazenamos também o tempo de processamento de cada execução, e realizamos 30 testes pra cada dataset, pois o resultado pode variar de acordo com o primeiro ponto escolhido como centro.

2. Descrição dos Métodos e Métricas Usadas

2.1. Métodos

O algoritmo que utilizamos é 2-aproximado: obtém uma resposta de raio não mais que duas vezes maior que o raio ótimo do conjunto de pontos. Seja S o conjunto de pontos do dataset, retornamos o conjunto C de K-Centros obtido a partir do seguinte procedimento:

1. Se $k \geq |S|$, então retornamos S.
2. Começamos com um conjunto vazio, $C \leftarrow \emptyset$.
3. Escolhemos um ponto x_1 qualquer, arbitrariamente (e essa é a parte arbitrária do algoritmo, o motivo pelo qual o repetimos várias vezes na parte de análise dos resultados). Fazemos $C \leftarrow \{x_1\}$.

4. Encontramos o ponto x_i que está mais distante de todos os pontos de C, e o adicionamos a C, fazendo $C \leftarrow C \cup \{x_i\}$.
5. Se $|C| = k$, retornamos C. Senão, voltamos ao passo 4.

Utilizamos também o K-Means, algoritmo de clusterização já implementado no `sklearn` [Pedregosa et al. 2011], que utilizamos para comparar com o nosso algoritmo. Esse algoritmo busca encontrar os centros de Cluster (que não necessariamente serão pontos do dataset), chamados centróides, que minimizam uma métrica chamada inércia, ou soma-de-quadrados-intracluster. Note que isso é diferente do que o nosso algoritmo se propõe, que é encontrar um raio total de cluster aproximado do menor possível.

2.2. Métricas

As métricas que utilizamos para avaliar a solução foram:

1. Raio máximo r de Cluster: dado um conjunto C de centros, um ponto p pertence ao cluster do centro $c \in C$ se $\min_{x \in C} dist(p, x) = dist(p, c)$, ou seja, se p está pelo menos tão perto de c que de qualquer outro centro em C. O raio de um Cluster é a maior distância pro centro dentre os pontos que pertencem a esse Cluster, e o raio de uma solução é o maior raio de Cluster dessa solução. Para o K-means, computamos isso com o conjunto de centroides como o conjunto de centros de Cluster: como especificado na documentação do K-Means (`https://scikit-learn.org/stable/modules/clustering.html#k-means`), os centroides são os pontos médios dos clusters. Esse valor varia de acordo com a magnitude dos valores do dataset e quanto menor, melhor (indica que os clusteres separam bem os pontos do dataset). Essa métrica depende da noção de distância que usamos, então calculamos uma vez para cada valor de p considerado.
2. Coeficiente Silhueta: pra cada instância p, é calculada a sua distância intra-cluster média a (média de distâncias de p pra outros pontos desse mesmo Cluster) e a distância média de cluster-mais-próximo b (distância entre p e o cluster mais próximo do qual p não pertence). Esse coeficiente é definido como $\frac{b-a}{\max(a,b)}$. Esse valor varia em $[-1, 1]$, e quanto maior, melhor. Essa métrica depende da noção de distância que usamos, então calculamos uma vez para cada valor de p considerado.
3. Índice de Rand Ajustado: o Índice de Rand "cru"é, segundo a documentação do Sklearn, basicamente $RI = \frac{a}{b}$, em que a é o número de pares de pontos que foram classificados como deveriam (ficam no mesmo Cluster se e somente se são da mesma classe, possuem a mesma label no dataset original), e b é o número total de pares de pontos. O índice ajustado, por outro lado, é $\frac{RI-ERI}{MRI-ERI}$, em que ERI é o RI esperado (no sentido probabilístico mesmo, se escolhéssemos clusteres ao acaso, qual seria o RI esperado) e MRI é o RI máximo (dentre todas as formas de escolher clusteres, alguma dará o maior valor possível de RI). Esse índice ajustado varia no intervalo $[-0.5, 1]$, quanto maior melhor, e se for próximo de 0 indica que o resultado final é quase uma escolha ao acaso de clusteres, se for negativo indica que é uma escolha pior que escolher ao acaso. Essa métrica não depende da noção de distância que usamos, mas como nosso algoritmo retorna clusteres diferentes pra diferentes métricas, nós a avaliamos com cada resultado que nosso algoritmo retorna.

4. Podemos considerar também o tempo de execução como uma métrica plausível, que foram medidos em segundos.

3. Descrição da Implementação

Para implementar o algoritmo descrito na subseção 2.1, utilizamos a linguagem Python, com os seguintes métodos de uso mais geral implementados em `utils.py`:

1. `mink_dist(v1,v2,p)` computa a distância de Minkowski entre $v1$ e $v2$ com parâmetro p. Em particular, se $p = 1$ temos a distância de Manhattan, e se $p = 2$ temos a distância euclidiana.
2. `dist_matrix(points,dist_f)` computa a matriz de distâncias entre todos os pares de pontos de `points`, calculada segundo a função `dist_f`. Na prática vamos usar `dist_f` como uma distância de Minkowski apenas, mas o método permite outros tipos de distância, para fins de generalidade.

Em `k_centers.py` foram implementadas as seguintes funcionalidades especificamente relacionadas à tarefa de clusterização:

1. O método `get_kmeans_centroids(points, cindexes)` computa os centroids de cada cluster dados os pontos e os índices do cluster de cada ponto (cindexes é retornado pela implementação de KMeans do sklearn).
2. O método `get_kmeans_cluster_radiuses(points, centroids, cindexes, dist)` computa os raios de cada cluster (`dist` é a função de distância, `cindexes` são os índices do Cluster de cada ponto, e `centroids` são os centroids de cada cluster).
3. O método `cluster_list(points, centers, dists)` computa o índice do cluster ao qual cada ponto pertence, dados os centros de cada cluster (centers é computado pela função `k_centers`, que descreveremos em breve).
4. O método `get_cluster_radiuses(points, cluster_of_point, dists)`, computa os raios de cada cluster, mas diferentemente do método `get_kmeans_cluster_radiuses(points, centroids, cindexes, dist)`, os centros são pontos de `points`, então não é necessário o argumento `centroids`, e a distância já vem pré-calculada na matriz `dists`.
5. O método `get_r(radiuses_list)` simplesmente retorna o maior raio da lista de raios que ele recebe como argumento.
6. O método `k_center(points, k, dists)`, em que `points` é um array numpy (biblioteca em Python para manipulação de matrizes [Harris et al. 2020]) de pontos, `k` é a quantidade de centros que queremos, e `dists` é a matriz de distâncias entre pontos. O retorno do método é um numpy array de `k` centros.
7. Temos também uma funcionalidade de depuração muito simples, em que se o arquivo for usado como script de python ao invés de como módulo (isso é testado com `__name__ == "__main__"`). É possível passar manualmente os parâmetros e os pontos desejados para testar a implementação do algoritmo de k centros.

No arquivo `prepare_datasets.py` temos os métodos relacionados ao download e à preparação em geral dos datasets, que são os seguintes (matriz de dados é a matriz $n \times m$, com uma linha por instância de dado e uma coluna por atributo, e vetor de classes é um vetor com n entradas que indica a classe de cada instância do dataset):

1. `download_and_unzip(url,name,unzipped_name)` baixa e descomprime um dataset presente no url. Note que esse método foi pensado para funcionar especialmente com os datasets usados nesse trabalho, é bem possível que não funcione em outros.
2. `get_Handwritten_Digit_data(unzipped_name, normalize = True)` é usado para obter a matriz de dados do primeiro dataset e seu vetor de classes, pois sua variável target (label) está em um formato diferente do resto, então foi melhor fazer um método separado só para lidar com esse dataset.
3. `get_csv_data(unzipped_name, ninstances, nfeatures, separator = ',', nheaders = 1, target = -1, normalize = True, startcol = 0)` é um método bem geral, usado para obter a matriz de dados e o vetor de classes para qualquer dataset, exceto o primeiro. Os parâmetros desse método são usados para ajustar de acordo cada dataset.
4. `get_all_datasets()` computa a matriz de dados e o vetor de classes de todos os 10 datasets que utilizamos para esse trabalho.

Finalmente, também temos o arquivo `test_all.py`, que é um script Python que obtém todos os datasets, executa todos os experimentos e imprime o resultado em formato legível para humanos. Os resultados também são salvos em um arquivo adicional `results.txt` exatamente como foram colocados neste documento.

4. Descrição dos Experimentos

Agora vamos descrever brevemente os experimentos que foram realizados neste trabalho.

4.1. Descrição das Bases de Dados

Utilizamos as seguintes bases de dados (chamarei os Clusteres de classes algumas vezes):

1. Semeion Handwritten Digit (`https://archive.ics.uci.edu/dataset/178/semeion+handwritten+digit`): um dataset com 1593 dígitos escritos à mão, colocados em uma caixa retangular 16×16, com valores em escala de cinza de 0 a 256. Consideramos cada pixel como uma feature da instância, e o dígito em si como a classe da instância.
2. Red Wine Quality (`https://archive.ics.uci.edu/dataset/186/wine+quality`): um dataset com dados de 1599 amostras de vinho vermelho, com 11 indicadores diversos como o PH e a quantidade de açúcar residual. A variável que indica a classe é a qualidade do vinho.
3. White Wine Quality (`https://archive.ics.uci.edu/dataset/186/wine+quality`): um dataset com dados de 4898 amostras de vinho vermelho, com 11 indicadores diversos como o PH e a quantidade de açúcar residual. A variável que indica a classe é a qualidade do vinho.
4. Absenteeism at work (`https://archive.ics.uci.edu/dataset/445/absenteeism+at+work`): um dataset com 740 indivíduos, com 20 variáveis relacionadas à ausência de indivíduos em seus trabalhos, como mês da ausência e idade do indivíduo. A variável que indica a classe é o tempo de ausência em horas (queremos prever quanto tempo no total o índíduo vai se ausentar no trabalho dadas as outras informações). Ignoramos a variável id.

5. Blood Transfusion Service Center (`https://archive.ics.uci.edu/dataset/176/blood+transfusion+service+center`): um dataset com 748 indivíduos, com 4 variáveis relacionadas ao histórico de doação de sangue do indivíduo como quantos meses se passaram desde a última vez que ele doou sangue e quantas vezes ele já doou sangue. A variável que indica a classe é um indicador binário de se o indivíduo doou ou não sangue em março de 2007.
6. Parkinson's Disease Classification (`https://archive.ics.uci.edu/dataset/470/parkinson+s+disease+classification`): um dataset com 756 indivíduos e 754 variáveis com iformações sobre a voz do indivíduo. A variável que indica a classe é um indicador de se a pessoa tem ou não Parkinson (variável "class"do dataset). Ignoramos a variável id.
7. Audit Data (`https://archive.ics.uci.edu/dataset/475/audit+data`): um dataset com 772 indivíduos e 17 variáveis com informações e pontuações diversas relacionadas à condição financeira do indivíduo. A variável que indica a classe é um indicador de se é ou não é arriscado realizar um empréstimo para essa pessoa.
8. QSAR Bioconcentration classes dataset (`https://archive.ics.uci.edu/dataset/510/qsar+bioconcentration+classes+dataset`): um dataset com 779 amostras biológicas e 10 variáveis com informações químicas diversas. A variável que indica a classe está relacionada ao tipo de bioconcentração da amostra, como informado na descrição original do dataset.
9. South German Credit (`https://archive.ics.uci.edu/dataset/522/south+german+credit`): um dataset com 1000 indivíduos e 20 variáveis relacionadas aos hábitos financeiros do indivíduo em questão. A variável que indica a classe é um indicador de se o indivíduo é ou não um bom pagador de empréstimos.
10. Banknote Authentication (`https://archive.ics.uci.edu/static/public/267/banknote+authentication.zip`): um dataset com 1372 notas de tesouro e 4 variáveis que indicam informações visuais sobre a autenticação dessas notas para bancos (são features coletadas a partir de métricas como entropia e informações relacionadas à Transformada de Wavelet da iimagem). A variável que indica a classe é um indicador binário de se a nota é ou não é forjada.

4.2. Sobre os experimentos

Cada uma das 10 bases de dados descritas é baixada e tratada adequadamente para processamento: extraímos uma matriz $n \times m$ em que cada linha é uma instância e cada coluna uma feature, e um vetor de tamanho n que indica a classe de cada instância. Em seguida, para cada valor de p em $\{1, 2, 10\}$ calculamos os K-Centros segundo o algoritmo visto em aula, considerando que temos n pontos m-dimensionais, segundo a distância de Minkowski com parâmetro p.

Como a escolha do primeiro ponto é arbitrária, a fazemos de forma pseudo-aleatória (a seed é configurada como 123456 para os resultados serem consistentes), e assim para cada p e base de dados executamos nosso algoritmo 30 vezes, como especificado no relatório. Apresentaremos os resultados médios e os desvios-padrão pra cada um desses pacotes de 30 execuções. Executamos também o algoritmo de K-Means já

implementado na biblioteca `sklearn` para fins de comparação, como foi pedido na especificação do trabalho. As métricas já descritas foram tomadas tanto para o K-Means da biblioteca quanto para o K-Centros implementado para este trabalho.

5. Apresentação e Análise dos Resultados

5.1. Apresentação dos resultados

Mostraremos uma tabela para cada valor de p da distância de Minkowski e algoritmo possível (K-Centros ou K-Means), em que as linhas representam os datasets e as colunas representam as médias e desvios-padrão das diferentes métricas ("R"para Raio Máximo,"Sihl."para Coeficiente Silhueta, "IRA"para Índice de Rand Ajustado, "T"para tempo de execução em segundos) para os dois métodos que estamos avaliando (o tempo total de execução para gerar esses dados, depois de baixar e preparar os datasets, foi de 691 segundos):

Tabela 1. Resultado para K-Centros e $p = 1$

	Médias				Desvios-Padrão			
	R	Silh.	IRA	T	R	Silh.	IRA	T
Dataset 1	241.279	0.047	0.173	0.034	3.386	0.010	0.028	0.003
Dataset 2	20.013	0.169	0.017	0.031	0.690	0.057	0.019	0.003
Dataset 3	20.597	0.120	0.016	0.174	0.948	0.047	0.015	0.015
Dataset 4	17.693	0.209	0.026	0.025	0.373	0.031	0.005	0.008
Dataset 5	11.310	0.680	0.024	0.009	0.921	0.142	0.026	0.000
Dataset 6	1500.662	0.607	0.006	0.012	66.267	0.030	0.004	0.003
Dataset 7	41.968	0.854	-0.001	0.010	2.243	0.000	0.000	0.002
Dataset 8	18.543	0.233	0.011	0.015	1.602	0.059	0.020	0.009
Dataset 9	32.185	0.162	0.005	0.017	1.737	0.053	0.027	0.008
Dataset 10	7.521	0.372	0.073	0.021	0.906	0.032	0.034	0.003

Tabela 2. Resultado para K-Means e $p = 1$

	Médias				Desvios-Padrão			
	R	Silh.	IRA	T	R	Silh.	IRA	T
Dataset 1	262.754	0.109	0.367	0.207	6.264	0.005	0.029	0.018
Dataset 2	23.227	0.177	0.070	0.188	3.724	0.016	0.007	0.013
Dataset 3	42.524	0.122	0.035	0.305	0.860	0.006	0.004	0.014
Dataset 4	20.401	0.319	0.048	0.137	1.304	0.031	0.006	0.028
Dataset 5	14.668	0.476	0.042	0.019	0.577	0.044	0.010	0.000
Dataset 6	1464.333	0.175	0.050	0.154	69.482	0.130	0.031	0.026
Dataset 7	63.393	0.612	-0.000	0.151	8.134	0.086	0.005	0.014
Dataset 8	21.423	0.280	0.037	0.030	0.739	0.006	0.001	0.012
Dataset 9	30.362	0.128	-0.001	0.183	0.404	0.023	0.005	0.022
Dataset 10	8.186	0.354	0.013	0.040	0.010	0.000	0.000	0.005

Tabela 3. Resultado para K-Centros e $p = 2$

	Médias				Desvios-Padrão			
	R	Silh.	IRA	T	R	Silh.	IRA	T
Dataset 1	23.191	0.027	0.170	0.034	0.113	0.005	0.026	0.003
Dataset 2	8.249	0.283	0.007	0.033	0.353	0.107	0.013	0.003
Dataset 3	9.322	0.230	0.005	0.171	0.478	0.066	0.008	0.010
Dataset 4	6.154	0.126	0.037	0.022	0.094	0.017	0.011	0.007
Dataset 5	8.675	0.711	0.028	0.009	0.640	0.050	0.013	0.001
Dataset 6	110.349	0.696	0.005	0.013	2.000	0.004	0.000	0.006
Dataset 7	18.879	0.871	-0.001	0.010	0.931	0.000	0.000	0.002
Dataset 8	8.218	0.437	0.008	0.013	0.326	0.075	0.007	0.007
Dataset 9	9.643	0.219	0.003	0.012	0.494	0.052	0.027	0.001
Dataset 10	4.559	0.346	0.066	0.020	0.568	0.043	0.039	0.003

Tabela 4. Resultado para K-Means e $p = 2$

	Médias				Desvios-Padrão			
	R	Silh.	IRA	T	R	Silh.	IRA	T
Dataset 1	19.434	0.060	0.375	0.208	0.439	0.004	0.040	0.013
Dataset 2	10.599	0.176	0.069	0.189	1.964	0.020	0.008	0.017
Dataset 3	19.012	0.127	0.035	0.295	0.636	0.007	0.005	0.016
Dataset 4	6.460	0.243	0.047	0.143	0.394	0.009	0.005	0.023
Dataset 5	9.083	0.427	0.041	0.021	0.442	0.048	0.013	0.008
Dataset 6	114.344	0.156	0.051	0.182	2.938	0.156	0.032	0.023
Dataset 7	34.577	0.476	-0.002	0.155	0.282	0.046	0.008	0.021
Dataset 8	13.210	0.239	0.037	0.034	0.003	0.001	0.000	0.031
Dataset 9	9.165	0.097	-0.002	0.162	0.121	0.042	0.008	0.006
Dataset 10	4.289	0.329	0.013	0.037	0.006	0.000	0.000	0.002

Tabela 5. Resultado para K-Centros e $p = 10$

	Médias				Desvios-Padrão			
	R	Silh.	IRA	T	R	Silh.	IRA	T
Dataset 1	3.830	0.024	0.025	0.037	0.012	0.007	0.021	0.006
Dataset 2	5.697	0.374	0.002	0.032	0.295	0.146	0.006	0.003
Dataset 3	7.102	0.405	0.003	0.177	0.396	0.047	0.002	0.010
Dataset 4	3.429	0.087	0.034	0.023	0.117	0.021	0.006	0.007
Dataset 5	7.333	0.724	0.002	0.008	1.104	0.044	0.016	0.001
Dataset 6	28.548	0.727	0.005	0.009	0.224	0.000	0.000	0.001
Dataset 7	16.796	0.884	-0.001	0.013	0.453	0.000	0.000	0.008
Dataset 8	6.142	0.465	0.010	0.011	0.074	0.093	0.008	0.003
Dataset 9	5.380	0.362	-0.001	0.014	0.063	0.063	0.015	0.003
Dataset 10	3.476	0.322	0.030	0.019	0.383	0.022	0.029	0.002

Tabela 6. Resultado para K-Means e $p = 10$

	Médias				Desvios-Padrão			
	R	Silh.	IRA	T	R	Silh.	IRA	T
Dataset 1	5.013	-0.003	0.367	0.214	0.045	0.004	0.034	0.024
Dataset 2	9.085	0.153	0.069	0.191	1.584	0.022	0.007	0.015
Dataset 3	13.984	0.113	0.034	0.314	0.166	0.007	0.006	0.020
Dataset 4	4.206	0.112	0.050	0.140	0.716	0.011	0.005	0.022
Dataset 5	7.893	0.359	0.041	0.020	0.014	0.029	0.013	0.002
Dataset 6	32.112	0.049	0.059	0.177	0.675	0.164	0.024	0.010
Dataset 7	25.125	0.389	0.000	0.166	2.250	0.137	0.005	0.026
Dataset 8	12.944	0.173	0.036	0.031	0.719	0.003	0.001	0.017
Dataset 9	5.464	0.052	-0.001	0.169	0.016	0.066	0.004	0.023
Dataset 10	3.200	0.288	0.013	0.039	0.007	0.000	0.000	0.010

5.2. Análise dos Resultados

A primeira observação é que os desvios-padrão foram todos bem pequenos em relação à média. Ou seja, as métricas obtidas foram consistentes, o que indica que não deve haver muito ganho em executar mais do que algumas dezenas de vezes os algoritmos em questão (ao menos para datasets com estrutura parecida com os que foram testados neste trabalho).

Além disso, notamos que os tempos de execução foram em geral bem pequenos, mas um pouco menores para o K-centros. Uma forma de explicar esse fenômeno é a razoável simplicidade desse algoritmo, em comparação com o K-means.

A partir das tabelas apresentadas podemos notar que, para $p = 1$, em apenas dois dos datasets o raio ficou menor com uso do K-means, para $p = 2$ isso ocorreu com três dos datasets e com $p = 10$ isso ocorreu com apenas um dataset. Temos então evidência de que o algoritmo K-centros, ao menos para os datasets testados, costuma possuir um desempenho melhor que o K-means para a métrica de raio máximo.

Já na métrica de Índice de Rand ajustado, ambos algoritmos possuíram desempenho muito fraco, sendo que o melhor foi para o dataset 0. Isso é um indicativo de que esses algoritmos chegam a clusteres bem diferentes dos "reais"(que seriam os que surgiriam se dividíssemos segundo a variável `label`). Ou seja, nos datasets testados esses métodos de clusterização não são boas formas de tentar classificar as instâncias nos seus grupos reais, pelo menos não da forma que foi feito nesse trabalho.

Obtivemos valores maiores (portanto melhores) para o índice Silhueta com o K-centros em quatro dos datasets para $p = 1$, em oito dos datasets para $p = 2$ e em nove para $p = 10$. Ou seja, em geral para valores de p maiores o K-centros teve um desempenho melhor que o K-means. Isso signfica que (ao menos para esses datasets) o K-centros costuma criar clusters mais separados, sem muita sobreposição. Isso talvez possa ser explicado pela forma como o algoritmo realiza a separação, tentando sempre tomar pontos o mais distante possível dos clusteres atuais a cada passo, levando então a menos sobreposição nos clusteres.

6. Conclusão

Podemos concluir que o algoritmo proposto, K-centros, possui resultados competitivos com o do K-means para as métricas de Silhueta e de Raio Máximo, ao menos nos datasets analisados. Portanto, se o raio máximo ou o índice Silhueta foem métricas de grande interesse, pode ser útil testar para os datasets em questão se temos alguma melhoria usando o K-centros.

Além disso, concluimos a métrica de distância tem uma grande influência no resultado de ambos algoritmos, então é interessante que a métrica que estamos usando tenha uma interpretação plausível para o problema de clusterização que temos em mãos. Alguns exemplos bem simples: se estamos tentando agrupar regiões de uma cidade com ruas bem organizadas (perfeitamente paralelas ou perpendiculares entre si), pode ser mais interessante a distância de Manhattan que a Euclidiana. Se estamos dividindo formigueiros em clusteres de tipos de formigueiro, sendo que formigas podem andar em qualquer direção no plano, faz mais sentido a distância euclidiana. Valores de p maiores se aproximam da métrica conhecida como Distância de Chebyshev (ou distância do xadrez), que pode ser interessante em cenários em que a coordenada que mais difere entre dois pontos é um fator relevante, por exemplo em logística de armazéns, já que ela efetivamente mede o tempo tomado por uma ponte rolante para se mover de um ponto para outro [Langevin and Riopel 2005].

7. Referências Bibliográficas

Referências

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., Fernández del Río, J., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585:357–362.

Langevin, A. and Riopel, D. (2005). *Logistics Systems: Design and Optimization*. Springer.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.