MC886 - Relatório do Assignment 03
Técnicas de Aprendizado Não-Supervisionado

Pedro Stramantinoli P. Cagliume Gomes
175955
p175955@dac.unicamp.br

Ruy Castilho Barrichelo
177012
r177012@dac.unicamp.br

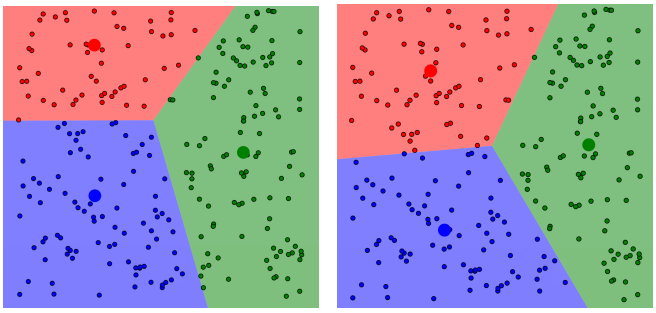
I. Introdução

Ainda que muitas das populares e notáveis aplicações do aprendizado de máquina estejam associadas a regressões e algoritmos de classificação, nem sempre é possível se ter em mãos um conjunto de dados categorizado ou rotulado.

Para casos como esse, tornou-se imprescíndivel a utilização de técnicas de aprendizado não-supervisionado, que possibilitam o particionamento dos dados de acordo com buscas de padrões ou métodos de agrupamento mais complexos, como alguns algoritmos de *clusterização*.

Dentre eles, estão o *K-means* e o *DBSCAN*, algoritmos que utilizam estratégias diferentes para tentar resolver um mesmo problema.

O primeiro basea-se em k (valor fornecido) pontos centrais - os centróides - para agrupar os elementos do conjunto de dados em k diferentes grupos - os *clusters* -, a partir do cálculo da distância de cada um desses elementos aos k centróides, de forma que sejam designados ao grupo mais próximo a si.

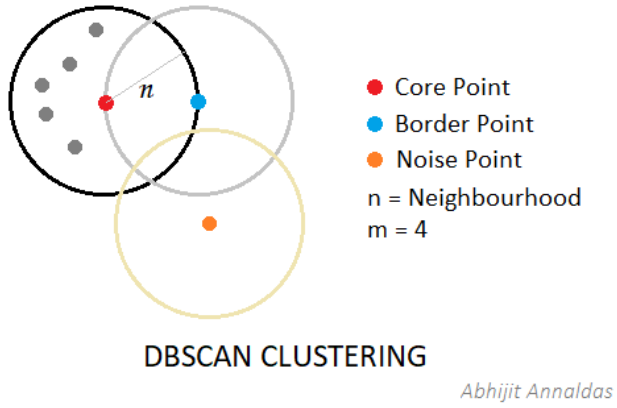


(a) Início (b) Após atualizações

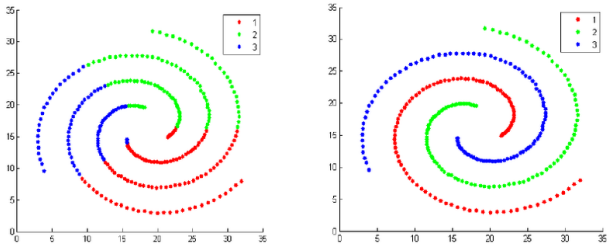
Figura 1. Algoritmo de *K-means*.[3]

Já o segundo, embora também se baseie na distribuição espacial dos elementos, não necessita receber um valor de *clusters* para sua execução. Isso ocorre uma vez que, a partir de um ponto aleatório do conjunto, são iniciadas classificações dos pontos restantes em *core points*, *border points* ou *noise points*, isto é, determina-se quais pontos devem ser agrupados ou não ao subconjunto atual analisado.

Com a realização sucessiva desse processo, são formados *clusters*, que consistem nos *core points* e *border points* agrupados em cada subconjunto. Dessa maneira, é possível realizar agrupamentos inatingíveis a partir de *K-means*, como visto na Figura 2-b.



(a) Classificação de elementos no *DBSCAN*. [4]



(b) Diferentes resultados para *K-means* e *DBSCAN*, respectivamente. [5]

Figura 2. Algoritmo de *DBSCAN*.[3]

No entanto, a aplicação de algoritmos como esses não é sempre viável. *Datasets* com um número muito grande de dimensões tornam qualquer análise mais complexa e mais custosa em relação a complexidades de tempo e espaço e, consequentemente, em recursos computacionais.

Esse problema pode ser contornado mediante a aplicação de métodos de redução de dimensionalidade, como o *PCA* e o *LDA*, que também tornam os dados mais visualizáveis e mais facilmente interpretáveis, ao auxiliar na redução de ruído.

O *PCA*, que é estudado neste trabalho, consiste em determinações de hiperplanos que mais se aproximam dos dados disponíveis, seguidas de projeções desses dados sobre eles, de forma que o erro de projeção seja o minimizado.

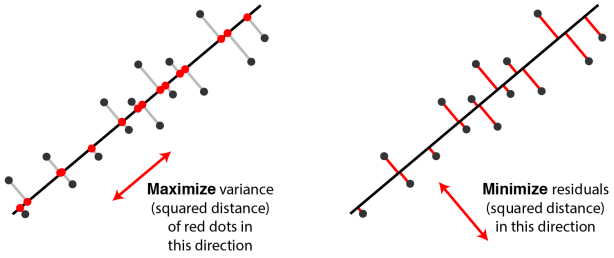


Figura 3. Projeção em componente do *PCA* e foco da otimização do algoritmo

II. Tratamento dos Dados

Para o tratamento dos dados, não foram utilizadas as técnicas de *Mean Normalization* e *Feature Scaling*, pois é possível se observar que os dados - uma *bag of words* extraída da coleção de *tweets* - já se encontravam em uma escala relativamente pequena e bem distribuídos.

III. Experimentação e Resultados

A. Métricas Utilizadas

Devido à ausência de rótulos para os dados, fez-se necessária a utilização de métricas que também os dispensassem. Por isso, foram escolhidos o Índice de Silhouette (*Silhouette Score*) e o índice de Calinski e Harabaz (*Calinski and Harabaz score*), ambos implementados no *Scikit-Learn*.

Ainda, as duas métricas se baseiam tanto na similaridade/dispersão entre elementos de um mesmo *cluster* e diferença/dispersão entre os elementos de *clusters* diferentes. [6] Por fim, vale notar que, embora o algoritmo *K-means* utilize da distância euclidiana - e por isso as métricas correspondentes usaram esse mesmo tipo de cálculo -, a distância utilizada para o *DBSCAN* foi a baseada nos cossenos (*cosine distance*), uma vez que não só tem uma boa aplicação no contexto de frequência de termos (*term frequency (tf)*, do *bag of words*, mas também exibiu resultados melhores quanto testados inicialmente, portanto as métricas associadas usaram, também, essa distância em seus cálculos.

B. K-means

Primeiramente, foi necessário um estudo para determinar a quantidade, ainda que aproximada, ideal de *clusters*, *k*, que seria utilizada neste algoritmo.

Para isso, utilizou-se o *Elbow Method*, que consiste em submeter os dados a repetidas execuções do *K-means*, com valores variados *k*, que resultam em diferentes valores de erro, calculados como a soma dos quadrados das distâncias entre as amostras e o centro do *cluster* mais próximo, ou seja, a distância da amostra em relação ao centróide do *cluster* a qual era foi designada.

Em seguida, controi-se um gráfico dos erros obtidos em função do valor de *k*, no qual é esperado se observar a existência de um ponto no gráfico, o *Elbow Point*, na qual o

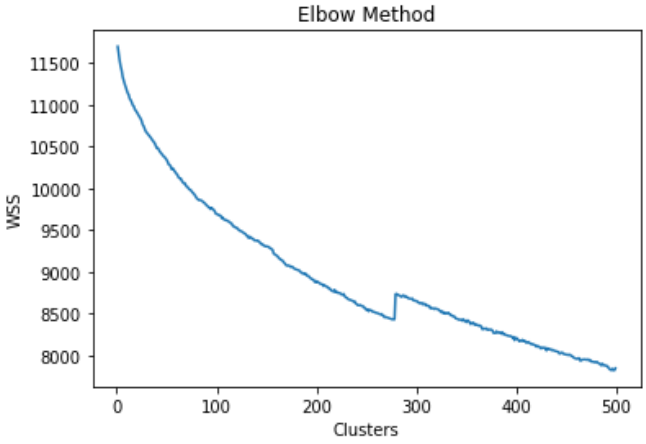


Figura 4. *Elbow Method*

incremento do valor de *k* não represente uma melhoria muito significativa no erro. Assim, escolhe-se esse valor de *k* como a quantidade de *clusters* a serem agrupados.

A figura 4 mostra o resultado obtido pelo *Elbow Method*, com variação *k* de 1 a 500. Como pode-se notar, o *Elbow Point* não é muito perceptível, portanto foi escolhido o valor considerado mais próximo a ele, com *k* igual a 278.

C. DBSCAN

Neste caso, não há necessidade de um passo inicial, como a execução do *Elbow Method*, na medida em que o algoritmo não necessita do fornecimento prévio do número de *clusters*. Por isso, foi possível focar no estudo e análise do impacto da variação dos parâmetros *minEps* e *min_samples* no resultado obtido, isto é, a distância mínima para que dois elementos sejam considerados da mesma vizinhança e o número mínimo de amostras na vizinhança de um elemento para que ele seja considerado um *core point*, respectivamente.

Assim, foram executadas diversas vezes o algoritmo e os resultados obtidos em função desses parâmetros podem ser observados nas figuras 5 e 6:.

O melhor resultado, de acordo com uma ordenação crescrente de índice de *Silhouette* e de Índice de *Calinski-Harabaz*, foi obtido a partir de *eps* de 0.7 e *min_samples* de 2. No entanto, como se pode observar, trata-se de valores liberais no que se refere à inclusão de elementos no *cluster*, logo, o resultado consiste em apenas um *cluster* e alguns *outliers*.

O primeiro resultado entre os mais bem avaliados que fornece mais de um *cluster* ocorre para *eps* de 0.6 e *min_samples* de 5, com índices de *Silhouette* de -2.82e-05 e índice de *Calinski-Harabaz* de 22.18, ou seja, já possuem métricas ruins.

D. Redução de Dimensionalidade

Como explicitado anteriormente, o meio pelo qual se reduziu a dimensionalidade do conjunto de dados, neste trabalho, foi o algoritmo *PCA* (*Principal component Analysis*).

Primeiramente, foi realizada uma análise do resultado, no que tange quantidade de dimensões restantes, após a execução

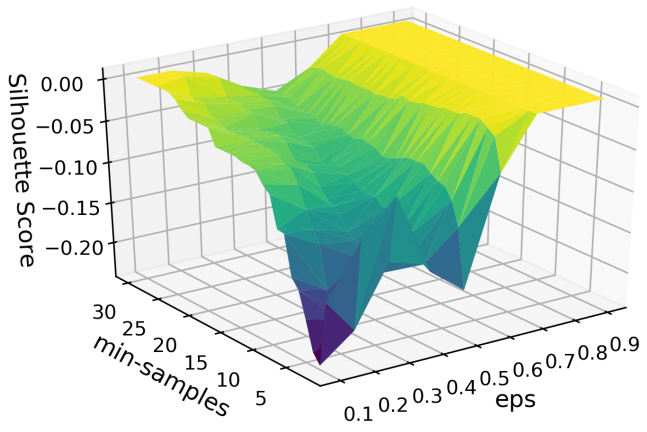


Figura 5. Índice Silhouette em função de *eps* e *min_samples*

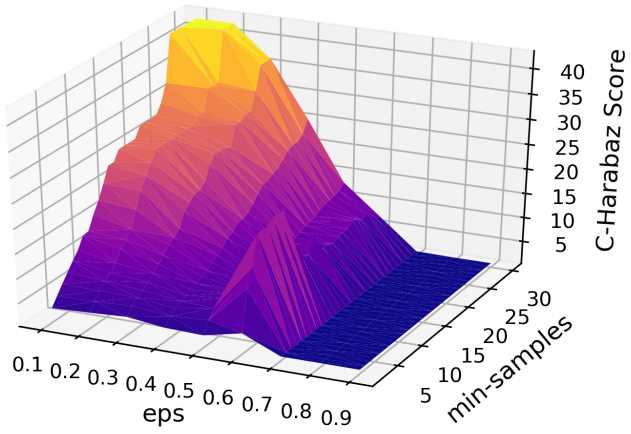


Figura 6. Índice Calinski-Harabaz em função de *eps* e *min_samples*

do algoritmo para diversos valores de variância/energias desejáveis de serem mantidas, que podem ser observados na tabela abaixo.

Quantidade de Dimensões - *PCA*							
Variância	1	.99	.98	.95	.90	.85	.50
Dimensões	1230	1165	1138	1070	973	887	427

Em seguida, foram selecionados os melhores resultados de cada algoritmo anterior para que os conjuntos de dados redimensionados pudessem ser aplicados e avaliados conforme as mesmas métricas, com o intuito de analisar a efetividade da redução ocorrida.

Para o *K-means*, utilizou-se o valor de k de 278, cujos resultados podem ser vistos a seguir:

Avaliação pós *PCA* - *K-means*							
Variância	1	.99	.98	.95	.90	.85	.50
Silhouette	.046	-.043	-.041	-.036	-.034	-.030	.080
Calinski-Harabaz	15.85	12.95	13.18	13.68	14.56	15.42	25.42

Já para o DBSCAN, a partir dos valores de *eps* de 0.6 e de *min_samples* de 5, os resultados foram:

Avaliação pós *PCA* - *DBSCAN*							
Variância	1	.99	.98	.95	.90	.85	.50
Silhouette	-2.8e-5	-.0069	.0092	.0092	.0090	.0087	.0039
Calinski-Harabaz	22.179	.665	.098	.095	.094	.095	.126

IV. Análise e Comparações

O algoritmo *K-means*, quando analisado sem a execução do *PCA*, apresentou o melhor resultado em relação ao *DBSCAN*, com um índice de *Silhouette* de 0.046, enquanto o de *Calinski-Harabaz* é de 15.85.

Com o intuito de melhor exemplificar o resultado da *clusterização*, abaixo estão duas amostras de elementos designados a um mesmo *cluster*.

"Maysoon Zayid, a touring standup comic with Cerebral Palsy, has a message to share"

"RT @drsanjaygupta: what are you having for dinner? a lot more sugar thank you think.."

Em relação ao algoritmo *DBSCAN*, os resultados obtidos também não foram adequados. As métricas resultaram em valores bem abaixo do esperado para uma clusterização razoável, por exemplo, bem próximas de 0 no índice de *Silhouette*. Ainda, a partir dos gráficos, nota-se que os melhores valores atingidos ocorrem para valores altos de *eps* e de *min_samples*, ou seja, pouco restritos na inclusão de elementos nos *clusters*, fato que resulta em um número muito baixo de *clusters* de gerados, chegando a apenas 1 *cluster* e *outliars*.

Outro ponto notável foi a observação que o uso de distâncias por cossenos de fato apresentou melhores resultados do que o de distâncias euclidianas para este algoritmo, conforme visto na literatura [11,12] . Por fim, como exemplo, a seguir podem ser vistos um par de *tweets* do mesmo *cluster* encontrado nesse algoritmo.

"Software architect reprograms his diet – and loses 140 lbs"

"Am I a bad parent if I give my child candy for Halloween?".

A análise inicial da redução de dimensionalidade parte da observação do número de dimensões restantes após a execução do algoritmo para diferentes valores de variância fornecidos.

Nota-se que para 99%, a redução foi muito baixa, de apenas dezenas entre as 1230 originais, mas torna-se mais clara logo a partir dos 98% e impactante a partir dos 90%.

O resultado pós redução de dimensionalidade, para o algoritmo *K-means*, foram piores do que os previamente observados, já que, por exemplo, mantiveram-se negativos no índice de *Silhouette*, com exceção do teste em que a variância fornecida foi a menor, contra-intuitivamente, pois, para este, ambos índices se apresentarem melhores.

Para o algoritmo *DBSCAN*, os resultados obtidos após a redução de dimensionalidade foram claramente melhores, porém, o número de clusters manteve-se baixo - entre 2 e 3 -, em que o maior deles novamente foi associado ao pior resultado dentre os 6 calculados. Embora seja uma melhora, não deixa de ser uma clusterização aquém do esperado, em que não foi possível agrupar bem os elementos do *dataset*.

V. Conclusão

Pode-se notar, facilmente, que em nenhum dos casos há semelhança entre as amostras de elementos pertencentes aos mesmos *clusters*, resultado condizente com as métricas obtidas nos experimentos.

Contudo, cabe a ressalva de que foram selecionados 2 pares de *tweets* dentre *13229* amostras, agrupadas, no caso do algoritmo *K-means*, em *278* grupos. Isto é, devido a vasta quantidade de amostras em um mesmo grupo, a comparação de apenas duas amostras não necessariamente fornece uma boa generalização da representatividade dos grupos.

Ainda que tenham sido usadas métricas conhecidas e já implementadas para validação de desempenho dos métodos de *clusterização*, deve-se lembrar de que nem toda métrica é um boa avaliação para todo tipo de *dataset*. O índice de *Calinski-Harabaz*, por exemplo, adequa-se mais a cenários com *clusters* mais densos e bem separados ou convexos.

A maneira mais apropriada de se validar as amostras de um mesmo grupo seria a checagem e comparação completa entre todos elementos, por indivíduos com conhecimento sobre o campo do problema, mas, para uma grande quantidade de amostras, esse tipo de comparação é claramente inviável.

No entanto, apesar dos resultados indesejados, pode-se experienciar como se dá o tratamento e solução de problemas reais, nos quais não se possui informação suficiente para direcionar a estratégia a ser tomada. Assim, nota-se a importância náo só do conhecimento sobre o assunto e sobre o *dataset* , mas também do impacto da quantidade de dados possuídos.

Referências

[1] UCI Machine Learning Repositor. Health News in Twitter. Disponível em: https://archive.ics.uci.edu/ml/datasets/Health+News+in+Twitter

[2] NumPy v1.15 Manual. https://docs.scipy.org/doc/numpy/

[3] Harris, Naftali. Visualizing K-Means Clustering. Disponível em: https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

[4] Abhijit Annaldas. Density Based Spatial Clustering of Applications with Noise (DBSCAN). Disponível em: https://www.kdnuggets.com/2017/10/density-based-spatial-clustering-applications-noise-dbscan.html

[5] K.P. Agrawal, Sanjay Garg, Pinkal Patel. Performance Measures for Densed and Arbitrary Shaped Clusters. Disponível em: https://www.researchgate.net/figure/Results-of-a-k-means-b-DBSCAN-on-Jain-dataset_fig2_291831757

[6] Clustering performance evaluation. Disponível em: http://scikit-learn.org/stable/modules/clustering.htmlclustering-evaluation

[7] Alex Williams. Everything you did and didn't know about PCA. Disponível em: http://alexhwilliams.info/itsneuronalblog/2016/03/27/pca/

[8] Scikit-Learn Decomposition. Disponível em: http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

[9] Scikit-Learn Clustering. Disponível em:http://scikit-learn.org/stable/modules/clustering.html

[10] Michael Galarnyk. PCA using Python (scikit-learn). Disponível em: https://towardsdatascience.com/pca-using-python-scikit-learn-e653f8989e60

[11] Evaluation measures of goodness or validity of clustering (without having truth labels). Disponível em: https://stats.stackexchange.com/questions/21807/evaluation-measures-of-goodness-or-validity-of-clustering-without-having-truth

[12] Sanket Gupta. Overview of Text Similarity Metrics in Python. Disponível em: https://towardsdatascience.com/overview-of-text-similarity-metrics-3397c4601f50