

Descoberta de Estruturas Organizacionais em um conjunto de dados, utilizando técnicas de aprendizado não supervisionadas

Guilherme Fava Brunhole
140647
gui.brunhole@gmail.com

Guilherme Sbrolini Mazzariol
138466
gsmazzariol@gmail.com

I. INTRODUÇÃO

A utilização do Aprendizado de Máquina em diversas áreas (medicinas, biologia, geografia, entre outras) trouxe um grande avanço na identificação e predição de padrões. Um exemplo desta aplicação de classificação é a identificação do conteúdo de imagens que aprende a diferenciar animais em várias imagens de uma base de dados, e depois de um processo de treinamento, consegue distinguir para uma nova imagem a qual classe (qual animal) ela pertence.

Uma técnica para resolver esse problema de classificação é o da descoberta de estruturas organizacionais em um conjunto de dados utilizando técnicas de aprendizado de dados não supervisionadas, ou seja, a rede tem que descobrir sozinha as relações, padrões, regularidades ou categorias em um conjunto de dados apresentados e codificar a saída.

Neste experimento iremos demonstrar uma técnica para a resolução deste problema.

II. ATIVIDADES

Dado um conjunto de 19.924 documentos, realizamos alguns experimentos para identificar suas características e conseguir desenvolver um modelo para classificação utilizando o algoritmo de classificação baseado no vizinho mais próximo (Nearest Neighbor – NN). Esta técnica descobre o vizinho mais próximo de uma instância e a classifica de acordo com este vizinho. O algoritmo k-NN (*k-Nearest Neighbor*)[1] - utilizado neste experimento - leva em consideração a distância euclidiana [2] (*Figure. 1*) de uma instância em relação a

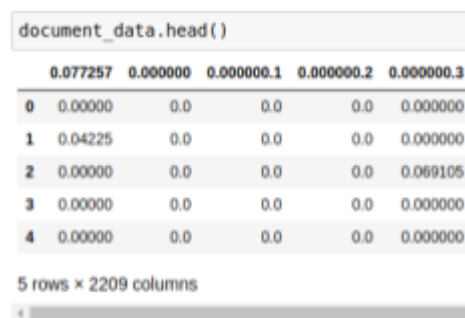
outras ‘k’ instâncias para fazer a classificação da mesma.

$$\sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^n (p_i - q_i)^2}.$$

Figure. 1

Existem duas formas de se medir a distância entre uma instância e seus vizinhos: distância Euclidiana e a distância Manhattan. A distância Euclidiana é utilizada porque conseguimos definir uma distância melhor entre as instâncias já que ela pode ser calculada para n-dimensões.

Realizando a leitura dos dados (*Figure. 2*), encontramos uma matriz de 5 linhas por 2209 colunas. A princípio não notamos nenhum padrão nestes dados. A seguir, iremos realizar outros experimentos para tentar identificá-los.



	0.077257	0.000000	0.000000	0.000000	0.000000
0	0.000000	0.0	0.0	0.0	0.000000
1	0.04225	0.0	0.0	0.0	0.000000
2	0.000000	0.0	0.0	0.0	0.069105
3	0.000000	0.0	0.0	0.0	0.000000
4	0.000000	0.0	0.0	0.0	0.000000

5 rows x 2209 columns

Figure. 2

Fazendo uma análise sobre os *medoids* dos documentos utilizando 10 Clusters (*Figure. 3*) identificamos que o segundo, terceiro e quinto arquivos, são de assuntos parecidos, ou seja, conseguimos captar algo que faz sentido.

With 10 clusters

```
X_train, X_test = train_test_split(document_data,
                                    test_size=0.33, random_state=42)

kmeans = KMeans(n_clusters=10, random_state=0)
kmeans.fit(X_train)
y_hat = kmeans.predict(X_test)

y_hat[:10]
array([4, 1, 1, 4, 1, 4, 1, 1, 7, 5], dtype=int32)
```

Figure. 3

Afim de melhorar este resultado, utilizamos agora 20 Clusters (Figure. 4). Olhando os arquivos que foram agrupados, podemos perceber que não fez muito sentido. Não são arquivos com assuntos similares.

With 20 clusters

```
X_train, X_test = train_test_split(document_data, test_size=0.33,
                                    random_state=42)

kmeans = KMeans(n_clusters=20, random_state=0).fit(X_train)
y_hat = kmeans.predict(X_test)

y_hat[:20]
array([ 1,  5,  3,  3, 18,  5,  0,  0, 14,  8,  3,  9,  0,  1,  5,  7,  5,
        18,  3, 16], dtype=int32)
```

Figure. 4

Tentamos então outra abordagem utilizando-se de 30 Clusters (Figure. 5). Olhando os arquivos que foram agrupados, podemos perceber que novamente, assim como no experimento anterior com menos Clusters, não fez muito sentido. Ou seja, não parecem ser arquivos com assuntos similares.

With 30 clusters

```
X_train, X_test = train_test_split(document_data, test_size=0.33,
                                    random_state=42)

kmeans = KMeans(n_clusters=30, random_state=0).fit(X_train)
y_hat = kmeans.predict(X_test)

y_hat[:30]
array([21, 15, 18, 25,  2,  4, 25,  9, 26, 16, 25, 12,  9, 25, 29, 13, 15,
        2,  4,  9, 21,  8, 15,  8,  2,  2,  4, 13,  4, 19], dtype=int32)
```

Figure. 5

IV. EXPERIMENTOS E DISCUSSÕES

Refazendo o experimento anterior considerando a redução da dimensionalidade PCA (*Principal Component Analysis*) [3] a fim de verificar a validade do Cluster, desta forma, a maioria dos dados se encaixam em uma distribuição estereotipada, e os pontos que desviarem drasticamente dessa distribuição se tornam pontos suspeitos, possíveis variações.

Utilizando o PCA, conseguimos ver a diferença entre as abordagens k-means++, random e PCA-based (Figure. 6).

k-means++	0.65s	36939	0.866	0.553	0.675	0.360	0.537	0.121
random	0.42s	39597	0.868	0.533	0.661	0.316	0.516	0.096
PCA-based	0.05s	48617	0.858	0.575	0.686	0.456	0.560	0.107

Figure. 6

Utilizamos três variâncias diferentes para cortar e reduzir a dimensionalidade: 50 Clusters (Figure. 7), 10 Clusters (Figure. 8) e 20 Clusters (Figure. 9).

K-means clustering on the digits dataset (PCA-reduced data)
Centroids are marked with white cross

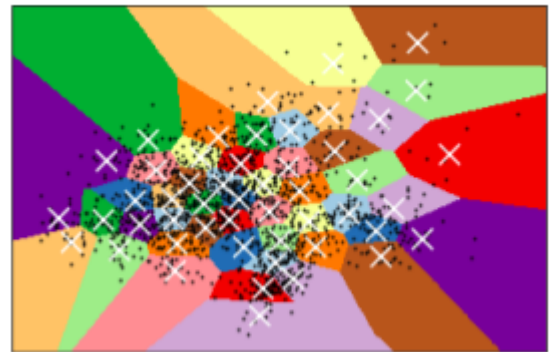


Figure. 7

K-means clustering on the digits dataset (PCA-reduced data)
Centroids are marked with white cross

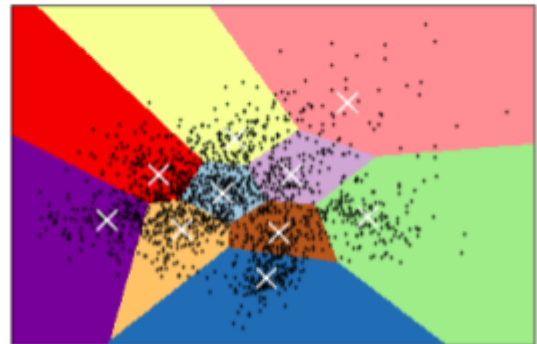


Figure. 8

K-means clustering on the digits dataset (PCA-reduced data)
Centroids are marked with white cross

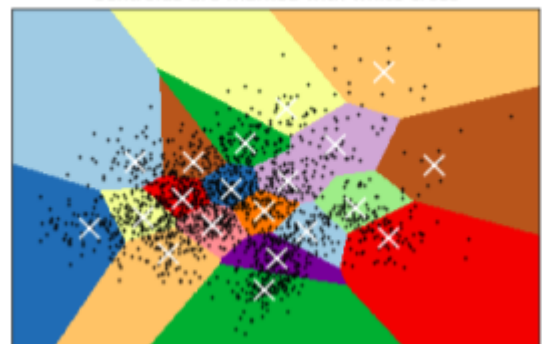


Figure. 9

Comparando estas três abordagens, sem dúvida a predição ficou muito mais rápida e o que faz bastante sentido, porque antes tínhamos mais de duas mil *features*, e conseguimos eliminar features que não são necessárias para o aprendizado do algoritmo. Conseguimos também reduzir para 2 dimensões e, assim, visualizar a distribuição em um scatter plot, facilitando em muito a compreensão dos dados

V. CONCLUSÕES

O problema de classificação de dados de forma não supervisionada utilizando o algoritmo de K-NN é muito útil para predição dos dados, mas não de forma tão eficiente. Conseguimos uma classificação melhor a partir da utilização do PCA, reduzindo os dados para duas dimensões e, realizando três vezes a variância dos dados. Sendo possível também visualizar em um gráfico para melhor compreensão dos dados.

REFERÊNCIAS

[1] Larose, Daniel T. "K-nearest neighbor algorithm." *Discovering Knowledge in Data: An Introduction to Data Mining* (2005): 90-106.

[2] Distância euclidiana, https://pt.wikipedia.org/wiki/Distância_euclidiana

[3] PCA (Principal Component Analysis) Machine Learning Tutorial, <https://www.dezyre.com/data-science-in-python-tutorial/principal-component-analysis-tutorial>