Estruturas Organizacionais com Treinamento não Supervisionado

MC886 - Machine Learning and Pattern Recognition 2S2017

Giovani Nascimento Pereira
168609
giovani.x.pereira@gmail.com

Carlos Figueiredo Freire de Carvalho
165684
figueiredo.carlos02@gmail.com

I. Introdução

O projeto 3 - Estruturas Organizacionais com treinamento não supervisionado, foi feito com o intuito de compreender melhor algoritmos de clusterização e o funcionamento de modelos de aprendizado não supervisionado para encontrar o número ótimo de grupos (clusters) numa amostra dada.

Alguns algoritmos de clusterização foram utilizados, a base foi feita com o *K-means*, mas também foram analisados o KmeansMiniBatch e KMedoids.

Para tentar definir qual o melhor número de grupos, usamos o método do *gráfico do elbow* para analisar o formato da curva encontrada, mas isso não foi suficiente para determinar o número de clusters da mostra. Depois passamos para a análise do *Cluster Silhouette*, que funciona como uma medida do quão coeso estão os clusters. Isso permitiu estimar melhor o número de grupos da amostra de dados.

Para melhor estudar os resultados, fizemos uma redução de dimensionalidade dos dados usando o PCA. Testamos vários valores para a variância mantida pelo algoritmo e estudamos o comportamento dos resultados baseados nisso. Com o uso do PCA os scores Silhouette tiveram uma visível melhora com a clusterização, apesar de estarmos perdendo parte da informação para reduzir a dimensão.

O melhor modelo que encontramos foi usando o kmeans, aplicando o PCA com variância de 0.6, reduzindo o número de features para apenas 187, separando os dados em 132 clusters, onde obtivemos um Silhouette score de 0.0702776418654.

II. Número de grupos da amostra

Para compreender melhor o problema, o primeiro passo e entender mais sobre nossos dados. E um ponto importante e tentar descobrir o número de grupos (clusters) que estamos trabalhando.

E para isso existem vários métodos que permitem estimar o numero de grupos de uma amostra. Começamos usando o algoritmo de clusterização do *Kmeans* do Scikit.

A. Gráfico do cotovelo

O método do gráfico do cotovelo consiste em gerar vários modelos, preparados com um número k de clusters diferentes, e então plotar o erro quadrado para cada k (o quadrado da distância de cada ponto a seu centroide). O comportamento esperado é que o erro diminua conforme k aumenta, mas inicialmente, o erro deve cair numa taxa mais rápida, ou seja, quando a velocidade de queda do erro diminuir, significa que você já passou seu número ótimo de grupos.

O método tem esse nome, pois a curva formada lembra um cotovelo.

O teste foi feito variando o número de clusters usados para poder visualizar o comportamento da curva em várias seções, de 1 a 150 grupos. Mas o formato da curva não apresentou a semelhança clara com o cotovelo, como pode ser visto na fig. 1.

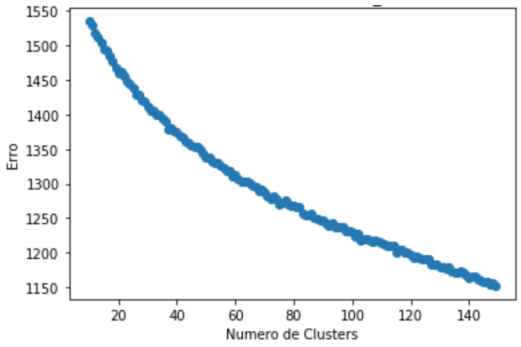

Figura 1. Gráfico obtido o erro quadrado pelo número de clusters usando k-means. Para esse gráfico foram usados de 10 a 150 grupos.

Isso impediu de fazer uma análise precisa dos valores da curva através desse método e tentar encontrar um número ideal de grupos. O que nos fez ir atrás de outros métodos de verificação da qualidade da clusterização para tentar melhorar esse resultado.

B. Análise da Silhueta

O valor de *Silhueta* é uma medida do quão semelhante um dado é ao seu próprio grupos (coesão), comparado com os outros grupos (separação). Esse valor varia de -1 a 1, onde altos valores marcam que o objeto está bem posicionado no seu grupos e mal colocado nos grupos vizinhos.

Esse score é obtido a partir da equação eq. (1).

$$\frac{b-a}{max(a,b)} \tag{1}$$

Sendo que "b"é dado pela distância média entre um ponto e todos os pontos no cluster mais próximo que não é o dele.

Enquanto "a"se refere à distância média entre um ponto e todos os outros pontos em seu cluster.

O cenário ideal, é que um grande número de objetos (dados) tenham valores altos (próximos de 1). Objetos com valores negativos, ou muitos objetos com valores baixos pode significar que o número de grupos é muito pequeno ou muito grande.

Foram feitos os gráficos de silhueta para:
$k \in [10; 150]$ e depois analisados conjuntamente.

Observando os gráficos de números mais baixos de grupos, como pode ser visto na primeira figura do fig. 2, pode-se inferir que o número de grupos colocado estava muito baixo, pela grande quantidade de objetos com valor de silhueta negativos.

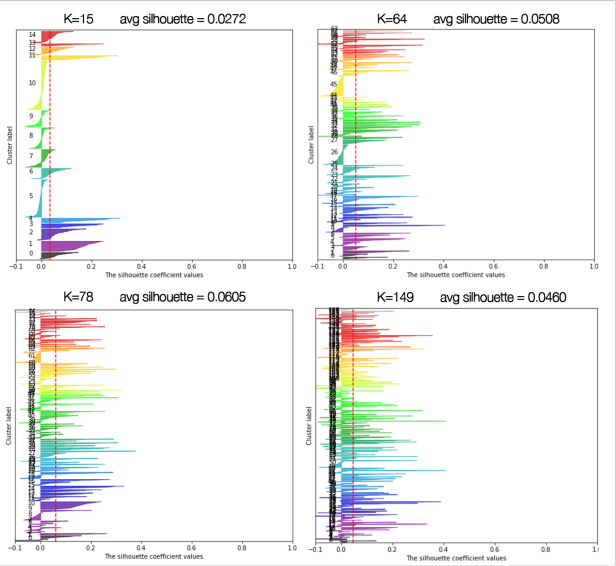

Figura 2. Seleção de 3 gráficos gerados para a análise de silhuetas gerados pelo método do K-means. K é o número de *clusters*e *avg silhouete* é o valor médio da silhueta para aquela instância. Da esquerda para direita, k=15, k=64, k=107.

Apesar de dar uma noção melhor que o gráfico do elbow (cotovelo), por mostrar a densidade de dados de cada cluster, o método também não foi muito claro para que pudéssemos ter certeza do número de grupos ótimo da amostra. Então observamos o valor da silhueta média para cada k fig. 3. O melhor valor encontrado foi quando $k = 78$ e $avg\_silhouete = 0.0605$

Colocando num gráfico os valores das silhuetas médias, fig. 3 é possível notar que $k = 78$ teve o melhor resultado que os outros modelos.

C. Análise da Clusterização

Encontramos o valor de $k = 78$ grupos para nossa amostra, mas precisávamos tentar reforçar que esse modelo era realmente um bom modelo.

A medida da silhueta já nos mostrava muita coisa, mas principalmente que o número de objetos agrupados em cada cluster era bem diferente, como podemos observar na fig. 4 que mostra o número de objetos agrupados em cada cluster.

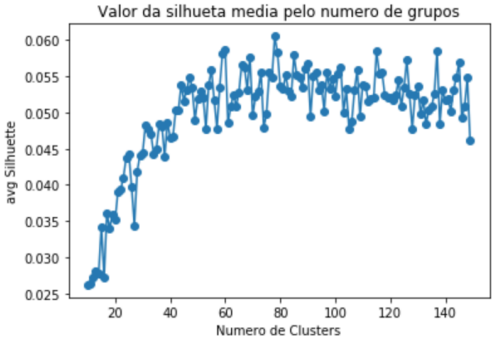

Figura 3. Gráfico do valor da silhueta média pelo número de clusters k.

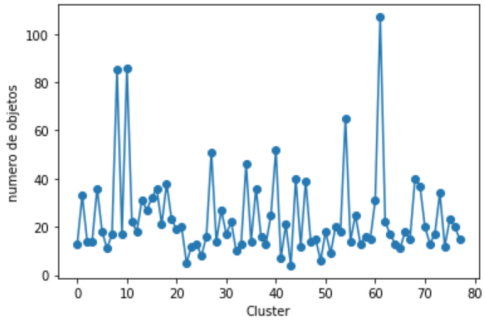

Figura 4. Gráfico do número de itens classificados em cada cluster

A média de objetos agrupados foi de $avg = 24.064 \pm 0.001$, mas o desvio padrão foi de $std = 18.030 \pm 0.001$, ou seja, os clusters não estavam nem um pouco uniformes, e se a base de dados tivesse um número semelhante de dados para cada cluster, isso pode mostrar que nosso agrupamento, apesar de ser nosso melhor resultado, ainda não foi suficiente para classificar corretamente nossos dados.

Para tentar melhorar o resultado da classificação, tentamos outros modelos, como pode ser visto na seção section III.

Para termos uma noção melhor se a clusterizações estava de fato funcionando, selecionamos um grupo de 2 arquivos que foram dispostos no mesmo clusters. Isso para 2 clusters diferentes.

Para o primeiro Cluster (o de número 5), os dois arquivos eram um email onde o *News group* era o mesmo: rec.sport.baseball. O conteúdo dos emails era bem próximo, e a palavra **baseball** tinha bastante repetições, esse provavelmente é o motivo pleo qual esses dois documentos foram agrupados conjuntamente.

Para o segundo cluster (o de número 8), os dois arquivos também eram emails, mas o conteúdo era bastante diferente. O primeiro email era sobre um programa para criação de gráficos e o segundo era sobre baseball novamente, aparentemente os dois divergiam bastante quanto ao conteúdo, eram semelhantes no mesmo ano em que foram enviados, e na mesma formatação (email).

O que podemos notar, é que o algoritmo de certa forma organizou os documentos, principalmente no primeiro caso onde

ficava claro que o conteúdo estava relacionado, mas analisando o segundo grupo, mostra que esse agrupamento provavelmente não foi tão bom quando analisando mais amplamente.

III. Outros modelos

Após utilizarmos o kmeans para estimar o número de clusters ideal para o nosso problema, tomamos a decisão de testar uma mudança na inicialização do kmeans e outros algoritmos de clusterização, mantendo constante o valor de clusters em 78.

Primeiro, rodamos novamente o kmeans utilizando o número ideal de clusters previamente obtido, alterando apenas a forma de inicialização dos centroides, da forma kmeans++ para randômica. Isso piorou ligeiramente o score silhueta do modelo, passando de 0.0605 para 0.0541.

Em seguida, testamos o MiniBatchKmeans. Desse modelo, era esperado que o treinamento fosse mais rápido, e que no final ele desse um resultado inferior ao do kmeans padrão. Isso se mostrou correto, visto que o treinamento foi rápido e o score obtido foi de 0.0051, o que representa um declínio muito considerável.

Por fim, foi a vez de usarmos o kmedoids. Neste caso, não tínhamos certeza se os resultados iam melhorar ou piorar. Obtivemos, então, a silhueta de 0.0390, ou seja, os resultados pioraram.

Concluímos, portanto, que nossa tentativa inicial, utilizando kmeans como inicialização kmeans++, era nossa melhor esperança de obter um bom resultado.

IV. PCA dimensionality reduction

Aplicamos o método PCA de redução de dimensões dos dados de entrada, com a esperança de que essa redução nos ajudasse no treinamento. Para isso man

Regulamos essa redução alterando o "n_components"da classe PCA do scikit, que é um valor que regula qual a variância dos dados após serem reduzidos, ou seja, qual a quantidade de dados após a redução. Alteramos esse valor de 0,1 a 0,9 e realizamos treinamentos com os dados gerados, mantendo o número de clusters ideal já obtido, 78.

O melhor resultado obtido se deu com uma redução para 117 features, onde obtivemos um Silhouette score de 0.0667.

Essa melhora no score em comparação a quando o PCA não foi utilizado pode ser justificado pois a redução deixa de lado informações que são pouco relevantes ao treinamento, permitindo, portanto, que o algoritmo tenha um foco maior no aprendizado do que é mais relevante.

Vale ressaltar, também, que, após a redução ter sido realizada, foi notável uma diminuição do tempo gasto no treinamento do modelo, quando em comparação com o kmeans "puro"(sem pré-processamento dos dados). Isso se dá pois o treinamento ocorre em um número menor de features.

Os resultados do *Silhouette Score* para o teste usando PCA pode ser observado na fig. 5.

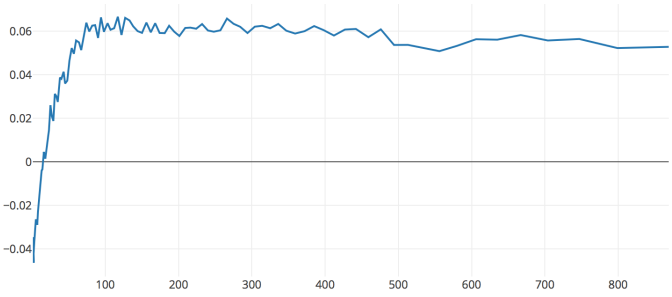

Figura 5. Gráfico do valor do Silhouette Score pelo número de features restantes quando aplicado o PCA.

V. Best Model

O melhor modelo que conseguimos obter foi, portanto, utilizando um pre-processamento de dados chamado PCA dimensionality reduction, e aplicando nos dados reduzidos o kmeans padrão do scikit learn, com inicialização kmeans++. O Silhouet Score desse modelo foi de 0.0667

VI. Conclusão e trabalho futuro

Considerando que o score máximo de silhueta é 1 (e um bom modelo de agrupamento tem o valor médio da silhueta o mais próximo de 1 possível), é possível notar que o nosso melhor resultado não obteve bom desempenho.

Concluímos, portanto, que agrupar dados não é uma tarefa simples, principalmente porque é muito difícil entender quais exatamente são as fraquezas do seu modelo. Entender se o resultado ruim ocorreu devido a existência de poucos clusters ou de muitos clusters, ou porque os cluster estão mal organizados no espaco, dentre outros motivos.

Outra coisa que podemos notar é que usar o PCA, ou outras técnicas de redução de dimensionalidade, pode ser muito vantajoso para o seu modelo. No nosso caso, nossos resultados melhoraram, e o tempo de processamento foi muito mais rápido. Esse ganho em velocidade pode ser muito importante para o seu modelo, se ele for aplicado em um sistema real.

Pensando em futuros esforços para melhor resolver esse problema, o que nós parece mais promissor seria testar outras formas de pre-processamento de dados, ou também conseguir outras métricas para analisar se o seu número de clusters está mais próximo do ideal.

Isso tupo, pode favorecer em muito o resultado do seu modelo.

Referências

[1] Christopher M. Bishop. “Pattern Recognition and Machine Learning”. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
[2] Learning Multiple Layers of Features from Tiny Images, Alex Krizhevsky, 2009
[3] Neural Net in 4 lines! using Scikit-Learn MLPClassifier. [http://rianadam.web.ugm.ac.id/2016/10/21/neural-net-in-4-lines-using-scikit-learn/]
[4] Scikit documentation and reference [http://scikit-learn.org/stable/]
[5] Sauspaugh Kmedoids code reference on github. [https://github.com/salspaugh/machine_learning/blob/master/clustering/kmedoids.py]
[6] [http://www.awesomestats.in/python-cluster-validation/]