Aprendizaje Automático - Trabajo 3

Maestría en ciencia de los datos y analítica
Universidad EAFIT

Liceth Mosquera - Código: 201910046228
Juan Diego Estrada - Código: 201910042228
Javier Rozo - Código: 201910017228
Steward Rios - Código: 201910050228
Mauricio Cuscagua - Código: 201910017228
Programa: Maestría en Ciencia de los Datos y Analítica
Docentes:
Olga Lucía Quintero

5 de noviembre de 2019

Algoritmos utilizados

1. Introducción

Luego de correr los algoritmos propuestos sobre los datos experimentales, utilizando diferentes herramientas: Matlab, Orange, Python y R, hemos escogido la mejor suite y metodología para seleccionar el mejor algoritmo de clusterización del problema a resolver en Proyecto Integrador.

Este trabajo describe la metodología seleccionada, así como el análisis de los resultados por cada algoritmo. Finalmente, la conclusión del trabajo, la cual incluye la justificación del algoritmo de aprendizaje no supervisado seleccionado.

2. Metodología

La metodología utilizada para abordar el problema fue la siguiente.

1. Particionar los datos en 70 % de entrenamiento y 30 % de prueba del conjunto de datos.

2. Estandarizar la partición de entrenamiento. Luego con la media y desviación generada de estos datos, se estandarizan los de validación.

3. Seleccionar el número óptimo de clusters.

4. Crear el modelo con el número de clusters óptimo.

5. Correr el modelo sobre los datos de prueba.

6. Calcular: ***silhoutte score*** y ***Calinski Harabasz*** sobre el modelo entrenado con los datos de entrenamiento.

7. Calcular el ***silhoutte score*** y ***Calinski Harabasz*** sobre el resultado de los datos de validación clusterizados.

8. Bajar de dimensionalidad los datos con PCA y repetir el proceso desde el paso 1 hasta el 7.

9. Bajar de dimensionalidad los datos con TSNE y repetir el proceso desde el paso 1 hasta el 7.

10. Comparar resultados en altas y bajas dimensiones y seleccionar modelo bajo los criterios calculados en 7.

Los resultados de cada algoritmo se pueden observar directamente en los notebooks, que se encuentran en la siguiente url:
`https://github.com/jdestradap/trabajo-3-aprendizaje-automatico`

K-means

k-means es uno de los algoritmos de aprendizaje no supervisado, es un algoritmo estocástico y heurístico, se le conoce como de aprendizaje duro, ya que al etiquetar cada uno de los objetos de la muestra le da una sola posibilidad de etiqueta, mientras que otros algoritmos dan la probabilidad de que pertenezca a uno de los clusters.

El número de clusters debe ser dado por el programador, la función objetivo de K-means es minimizar las distancias al cuadrado de cada objeto, también llamado como inercia.

K-means es un proceso iterativo dependiendo de los cluster de entrada tomando el mismo número de centroides, se le asigna a cada objeto al centroide más cercano, aplicando la medida de distancia seleccionada, generalmente la euclidiana, se recalcula los centroides de acuerdo a los objetos asignados a él y vuelve y asigna los objetos más cercanos a los nuevos centroides, esto se hace hasta que los centroides convergen o cuando no se hay una diferencia mayor al umbral de cada centroide entre una iteración y otra.

El seudo código sería el siguiente:

1. - Inicializar K Clusters con sus centroides μ_1,...μ_k de forma aleatoria.

2.- while not converge:

for i in range (dataset):

$c_k := argmin\|x_i - \mu_k\|^2$

for j in range (K):

$\mu_j := \frac{1}{N}\sum_{i=1}^{N} x_i$

El objetivo es minimizar la siguiente función de costo:

$$\mathcal{J} = \sum_{i=1}^{K}\sum_{j=1}^{n} u_{ij}||x_j - c_k||^2$$

Para este algoritmo se usaron varias suites con los datos jugetes como: Python, R y Orange. Escogiendo Python como la suite para hacer los cluster de los datos del proyecto integrador el cual tiene 65 variables y 4958 objetos.

La librería que acordamos utilizar es la de scikit-learn que se explica en la url:`https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html`.//

A la función de scikit-learn hay que pasarle la cantidad de clusters, el método de inicialización de los centroides, el número de veces que el algoritmo se ejecuta con diferentes semillas centroides, el máximo número de iteraciones, la tolerancia relativa, entre otros.

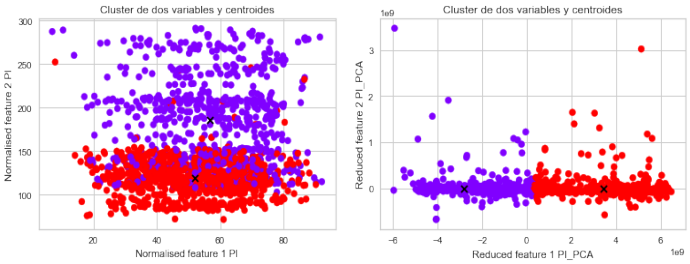
La salida de esta función son las coordenadas de los centroides de los clusters, las etiquetas, la inercia que es la suma de las distancias al cuadrado de cada objeto del Cluster a un punto μ conocido como Centroide y el número de iteraciones de la corrida.

Para escoger el número de clusters se utilizó el método codo 'Elbow Method' la métrica que utiliza es la distorsión media, definida por la suma de las distancias al cuadrado entre cada observación y su centroide más cercano. Si hay un punto de inflexión fuerte, es una buena indicación de que el modelo subyacente se ajusta mejor en ese punto. También se utilizó la medida del coeficiente de silhoutte que es la relación media de la distancia dentro del grupo y al grupo más cercano y finalmente se evaluó con la puntuación de Calinski Harabasz que calcula la relación de dispersión entre y dentro de los clusters.

Se hizo K-means para los datos de PI y aunque los resultados tanto en altas como en bajas dimensiones tienen coeficiente de silhoutte bueno alrededor de 0.65, están un poco alejados de la etiqueta real.

Fuzzy-c Means

Fuzzy c-means utiliza principios de lógica difusa, para crear clusters en datos multidimensionales, en donde asigna a cada punto un grado de pertenencia al centro de cada cluster dada por una probabilidad.



(a) Altas Dimensiones (b) Bajas Dimensiones

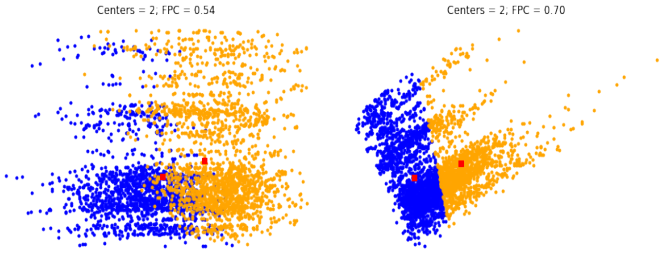
Figura 1: K-means en altas y bajas dimensiones

Zadeh es quien introduce esta idea en 1965: Representar la similaridad que un punto comparte con cada cluster, con una función (denominada membership) cuyos valores (llamados memberships) se encuentran entre 0 y 1.

La historia, filosofía y derivación como sistema es documentando por Bezdek, en 1981. El efecto fue crear un agrupamiento con particiones difusas sobre el conjunto de datos dado.

Se utilizaron dos herramientas para correr este algoritmo: Matlab y Python. La primera tiene por defecto este algoritmo implementado, bajo la función fcm. Matlab no cuenta con suficiente docuemtación, ejemplos y otras herramientas hizo más fácil realizar el ejercicio solicitado si lo comparamos con Python. Seleccionamos Python sobre todo, por las librerías de Scikit Learn y Jupyter notebook

Los resultados obtenidos luego de correr en altas y bajas dimensiones el resultado del ***silhoutte score*** sobre los datos de prueba fue de 0.38987059917830785



(a) Altas Dimensiones (b) Bajas Dimensiones

Figura 2: Fuzzy-C-means en altas y bajas dimensiones

Sustractive

Este método se plantea como una alternativa al método Mountain para la identificación de los centroides y el número de clusters, presentando varias mejoras en las que se incluyen:

- Menor costo computacional ya que recorre un espacio menor (no explora todos los vértices de la malla si no que explora solo los elementos).
- No requiere definir una resolución de la malla.

El algoritmo calcula un valor potencial, que es una calificación que denota que tantos datos cercanos tiene cada punto dado un radio, se elige el elemento con la mayor calificación como primer centroide y se recalculan los valores potenciales y se elige como siguiente centroide el nuevo mayor valor, se continua el proceso hasta que el mayor valor seleccionado sea menor a el valor potencial del primer centroide multiplicado por un ϵ menor que 1.

Evaluación de resultados.

De acuerdo con las instrucciones, se tomaron los datos del proyecto integrador con los que se realizó el primer trabajo, se evaluó en altas y bajas dimensiones el algoritmo sustractive, y finalmente con la información recolectada (número de centroides y posición), se realizó un K-means, de 1 sola iteración y ,de acuerdo con la metodología definida con el grupo, se evaluó la agrupación bajo la métrica silhouette obteniendo los siguientes resultados:

Ejericio	# Variables-componentes	# de Centroides	Centroides Seleccionados	Silhouette
Dimension original	65	3	3623; 1887; 3593	0.2721
Bajas dimensiones	19	4	1887; 3394; 3623; 3378	0.2547

Figura 3: Resultados Sustractive

Experiencia de aprendizaje

Se observo que dentro de las diferentes suites el algoritmo solo se encontraba implementado en Matlab, el equipo programó el algoritmo en Python basados en el paper original.

Clustering temporal data

Dado que el trabajo del proyecto integrador es sobre datos financieros, que generalmente tienen dimensión temporal, el equipo decidió investigar por algoritmos y aplicaciones no supervisadas sobre series de tiempo.

Se implementó K-shape Algorithm expuesto por Paparrizos y Gravano en el paper k-Shape: Efficient and Accurate Clustering of Time Series, usando la libreria tslearn.

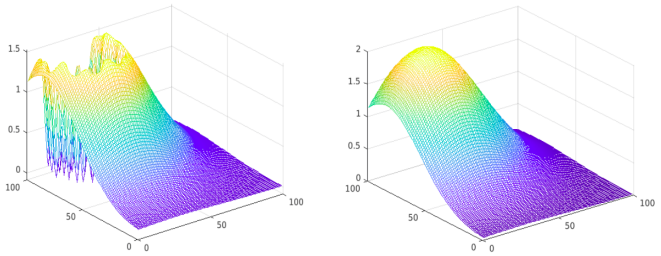
El algoritmo es una variación de k- means, que es iterativo en el calculo, y su principal diferencia es que calcula la distancia sobre la covarianza de las series.

un requisito de este tipo de algoritmo es que las series deben tener la misma dimensión.(tamaño de la serie).

Como ejercicio de aprendizaje, se replicó un ejercicio sobre unos datos de electrocardiogramas, y el ejercicio con los datos reales de series de tiempo del p.i.

Mountain Clustering

La idea detrás de éste algoritmo es aproximar la estimación de centros de cluster basados en el concepto de la función de montaña. Es útil para obtener los valores iniciales de clusters que son requeridos para algoritmos mas complejos. El método se basa en una cuadrícula en el espacio, para la construcción de una función de montaña a partir de los datos y luego una destrucción de las montañas para obtener los centros de los conglomerados.



(a) Destruyendo montaña (b) Montaña

Figura 4: Resultados Mountain Method

Conclusiones

Existen dos formas de realizar clustering sobre un conjunto de datos: Hard y Soft clustering. El primero agrupa los elementos de forma que, cada uno esta solamente asignado a un cluster. K-Means es un algoritmo de este estilo. La segunda forma no tiene una respuesta binaria, si no que la data o cada ítem tiene una probabilidad asociada de pertenecer en múltiples clusters, por ejemplo fuzzy c-means.

Trabajando sobre los datos de proyecto integrador, en donde el problema es tomar una decisión de cambiar la posición sobre los activos financieros - decisión binaria - y luego de analizar los resultados de cada algoritmo, K-Means logra obtener los clusters mas

definidos y separados que los demás algoritmos analizados. Esto basado en el resultado del coeficiente silhouette en bajas dimensiones el cual fue el mas alto de los algoritmos utilizados para el análisis, con un valor de 0.65.

Para el caso de clusters con algoritmos soft no aplicaría para nuestro problema.

Todos los códigos fuentes los encuentras en el siguiente repositorio:

`https://github.com/jdestradap/trabajo-3-aprendizaje-automatico`