Institución Universitaria	INFORME FINAL DE TRABAJO DE GRADO	Código	FDE 089
		Versión	03
		Fecha	2015-01-27

Desarrollo de librería para el método Spectral Clustering

Arnold Julián Morales Zapata

Ingeniería Electrónica

Andrés Eduardo Castro Ospina

INSTITUTO TECNOLÓGICO METROPOLITANO

2018

RESUMEN

La clasificación de datos con diferentes características, a lo largo del tiempo ha sido una necesidad en diferentes campos como la química, la medicina, la electrónica, entre otras. Hoy en día con el nacimiento de tecnologías basadas en Big data y teniendo en cuenta que los datos de cierta forma controlan el mundo, es imperativo desarrollar métodos para clasificar la información de una forma correcta y eficiente. Actualmente se han desarrollado varios algoritmos capaces de cumplir a cabalidad con esta tarea, entre los más comunes tenemos los K vecinos mas cercanos para conjuntos de datos con etiquetas conocidas o lo que se conocer como aprendizaje supervisado, K-means y Spectral Clustering para conjuntos de datos sin etiquetar o aprendizaje no supervisado. En el primer problema una de las falencias que se encuentra es clara, usualmente no se tienen los datos etiquetados, por el costo que conlleva algo así, situación que complica en gran manera el uso de este tipo de algoritmos. Los otros dos algoritmos anteriormente mencionados, tienen un funcionamiento similar. La diferencia de estos se encuentra en la forma de distribución de los datos, para el caso del K-means los datos deben ser linealmente separables y tener formas compactas y separadas entre grupos, mientras que para el spectral clustering, esta distribución no es importante, porque el método encuentra la distribución óptima, esto teniendo en cuenta ciertos parámetros de entrada que son la clave de éxito del algoritmo.

Este trabajo particularmente se enfoca en el desarrollo del método spectral clustering, y las diferentes estrategias que existen para encontrar una matriz de afinidad, que es el núcleo del método, como método de desarrollo se hizo uso del álgebra lineal para de esta manera expresar las diferentes funciones en forma matricial y así tener algoritmos más eficientes.

Para el proyecto se utilizaron bases de datos sintéticos, con distribuciones no separables linealmente y que no presentan necesariamente una forma compacta, que es el problema específico que busca resolver el método spectral clustering, además de ellos se grafican las diferentes matrices de afinidad y los resultados de agrupamiento de los algoritmos, para que se tenga una vista de qué tan eficaz puede llegar a ser el algoritmo.

Palabras clave: kernel, matriz de afinidad, spectral clustering, grafo, k-means, laplaciano, Python, Matlab.

RECONOCIMIENTOS

En el presente trabajo, participaron varias personas directa o indirectamente, leyendo, opinando, corrigiendo, animando o dando consejos para mejorar; la razón por la cual es necesario darles un reconocimiento y agradecerles por el acompañamiento en el desarrollo del proyecto.

Gracias a los profesores del ITM, que hicieron parte de mi formación y que, con su excelente labor, fortalecieron mi crecimiento como persona y profesional; gracias a Andrés Castro el asesor de trabajo de grados, quien siempre tuvo la mejor disposición para atender mis inquietudes y estuvo en pro del correcto desarrollo del documento final.

Agradecer de manera especial a mi familia que siempre estuvo al tanto de mi desarrollo académico y personal, y que hacen parte fundamental en mi vida, porque sin su apoyo, muchos de estos logros no se hubiesen llevado a cabo.

ACRÓNIMOS

CNN Common nearest neighbors

ALS Adjustable line segment

KNN K-nearest neighbors

CPU Central Processing Unit

RAM Random Acces Memory

SC Spectral Clustering

TABLA DE CONTENIDO

1. INTRODUCCIÓN

En diferentes áreas como: la biología, metrología, estadística, ciencias sociales, medicina, entre otras. Se cuenta con grandes cantidades de datos experimentales y es necesario en las primeras etapas de análisis identificar grupos con características similares, (von Luxburg, 2007) nos dice que en estos casos las mejores técnicas son las de clustering. Esto se debe a que en la mayor parte de los casos los datos no están etiquetados debido a la cantidad o complejidad de los mismos, por esta razón utilizar técnicas de aprendizaje supervisado no es algo a tener en cuenta, mientras que el clustering toma mucha fuerza, debido a que se acopla a la mayoría de los problemas.

La idea de realizar un trabajo basado en un método de aprendizaje no supervisado como lo es spectral clustering radica básicamente en la identificación de una necesidad de agrupar y separar la información de una forma eficaz, en diferentes áreas donde se tomen grandes o pequeñas cantidades de datos que requieran estos procedimientos; además de esto también se hace para apoyar a los estudiantes, profesores e investigadores del ITM, que necesiten hacer una clasificación de datos, y no disponga de los recursos para realizar el proceso.

Se debe tener en cuenta que la aplicación del presente trabajo se extiende a todas las áreas del conocimiento puesto que, las tomas de datos son una constante, y usualmente estas no son fácilmente clasificables, razón por la cual la aplicación se podría tomar como base para un proyecto más extenso y complejo e incluso como cimiento para un emprendimiento en el tratamiento de datos.

Objetivos

Objetivo general

Desarrollar una librería científica funcional, orientada a resolver tareas de agrupamiento, basándose en la aplicación de la teoría del método spectral clustering.

Objetivos específicos

- Investigar diferentes métodos del estado del arte para crear una matriz de afinidad o grafo ponderado.

- Implementar y validar algoritmos siempre teniendo en cuenta un trabajo matricial.
- Desarrollar un demo para Python y Matlab, para que el usuario final pueda hacer pruebas de funcionamiento del algoritmo.

El proyecto se realizó basándose en dos principios básicos: el beneficio del estudiante como futuro ingeniero y el beneficio de los demás estudiantes para la realización de investigaciones en el mismo tema o en diferentes campos que puedan beneficiarse de los algoritmos desarrollados. Inicialmente se contactó con el profesor Andrés Castro, quien fue el encargado de proponer la temática que se iba a trabajar, para ello se realizaron varias reuniones explicando la importancia del desarrollo para diferentes áreas, y que el desarrollo aún sigue en ejecución razón por la cual el campo investigativo es bastante amplio, lo cual lo hace un producto innovador.

Posteriormente se pactó un cronograma de trabajo, de entregas y de asesorías, el cual se hizo teniendo en cuenta los cruces de horarios de los participantes en el proyecto, para lograr un mayor aprovechamiento del tiempo y se definieron una serie de parámetros y objetivos que se debían de entregas en los tiempos pactados.

Teniendo un cronograma establecido se separarán las fuentes de información sobre las cuales se iba trabajar, se leyeron y se sacaron las partes fundamentales de cada documento que se eligió. Después de adquirir el conocimiento requerido y siempre apoyándose en el docente se procedió al desarrollo de los algoritmos en un orden secuencial, desde el más sencillo que es el kernel base hasta el más complejo que para este caso sería el kernel ALS.

Después del desarrollo de los algoritmos se realizaron los demos en las diferentes plataformas, en Matlab con interfaz gráfica y en Python de ejecución por consola.

2. MARCO TEÓRICO

Métodos de aprendizaje

Hoy por hoy, gracias al rápido avance de la tecnología en el campo de las ciencias de datos es posible almacenar y procesar cantidades de datos bastante amplias, teniendo esto es cuenta aparece el concepto de aprendizaje maquina o aprendizaje automático, (Gallegos, y otros, 2014), definen este concepto como programas computacionales que buscan optimizar los parámetros de un modelo usando datos previos o datos de entrenamiento. Los modelos pueden ser inductivos, cuando permiten hacer predicciones sobre el futuro o bien descriptivos cuando permiten generar conocimiento a partir de los datos.

(Gallegos, y otros, 2014) comentan que "El aprendizaje automático usa la teoría estadística para construir modelos matemáticos, pues de esta manera es posible hacer inferencias a partir de una muestra."

Existen dos modos diferenciados en el proceso del aprendizaje automático dependiendo de si se posee información acerca de la salida o no. Estos modos son denominados como supervisado y no supervisado respectivamente. En los siguientes puntos se hará una breve introducción a estos tipos de aprendizaje.

Aprendizaje Supervisado

El aprendizaje supervisado abarca todas aquellas aplicaciones o procesos en los que se dispone de información tanto de los valores de entrada del sistema como de los valores de salida. De manera global, dos de los problemas típicos en el aprendizaje supervisado son el de clasificación y el de regresión. En clasificación, los valores deseados se corresponden con las etiquetas de cada caso, mientras que, en regresión, la información de salida es el valor real a estimar.

Uno de los más grandes problemas que se encuentran en el aprendizaje supervisado es que en la mayoría de las aplicaciones es muy complejo etiquetar los datos debido a la densidad o complejidad de estos. Por esta razón no es muy común ver este concepto aplicado en sistemas funcionales, lo cual lo hace perder peso frente a su contraparte que es el Aprendizaje no supervisado.

Aprendizaje No Supervisado

Los sistemas de clasificación no supervisados son aquellos en los que no se dispone de una base de datos de ejemplos previamente clasificados, sino que únicamente a partir de las características de los ejemplos, se intenta agrupar diferentes conjuntos de datos según la similaridad de sus características, a este proceso se le llama clustering, (Yadav & Sharma , 2013) definen el clustering como "un proceso de agrupación de objetos de datos en clústeres inconexos para que los datos en el mismo clúster sean similares, pero los datos pertenecientes a diferentes clústeres difieren. Un clúster es una colección de objetos de datos que son similares entre sí y se encuentran en el mismo clúster y diferentes a los objetos que se encuentran en otros clústeres". Esta aplicación de clustering se hace mediante diferentes métodos, algunos de ellos se presentarán a continuación.

K-means

"K-means es el método de particionamiento más popular de clustering. Primero fue propuesto por MacQueen en 1967. K-means es un método de agrupamiento no iterativo, no determinista, numérico e iterativo. En k-mean, cada clúster está representado por el valor medio de los objetos en el clúster. Aquí dividimos un conjunto de n objetos en k clúster de manera que la similitud entre clústeres sea baja y la similitud intra-clúster sea alta. La similitud se mide en términos del valor medio de los objetos en un grupo" (Yadav & Sharma , 2013).

El algoritmo K-means consiste en dos fases.

- Seleccionar los k centroides al azar, donde el valor k se fija por adelantado.
- Cada objeto en el conjunto de datos está asociado al centroide más cercano. La distancia euclidiana se usa para medir la distancia entre cada objeto de datos y el centroide del grupo.

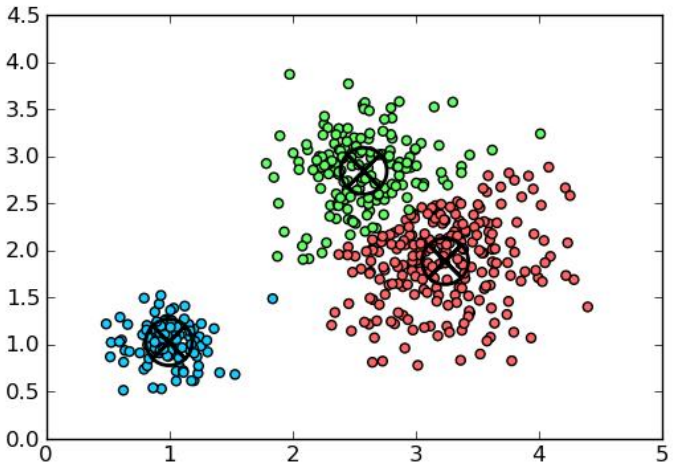


Imagen 1. *Resultado del algortimo K-means, aplicado a un conjunto de datos.*

Clustering Jerárquico

"Las técnicas jerárquicas producen una secuencia anidada de particiones, con un único clúster que incluye todo en la parte superior y clústeres únicos de puntos individuales en la parte inferior. Cada nivel intermedio se puede ver como la combinación de dos clústeres del siguiente nivel inferior (o la división de un clúster del siguiente nivel superior). El resultado de un algoritmo de agrupamiento jerárquico se puede mostrar gráficamente como árbol, llamado dendrograma. Este árbol muestra gráficamente el proceso de fusión y los clústeres intermedios" (Steinbach, Karypis, & Kumar, 2000).

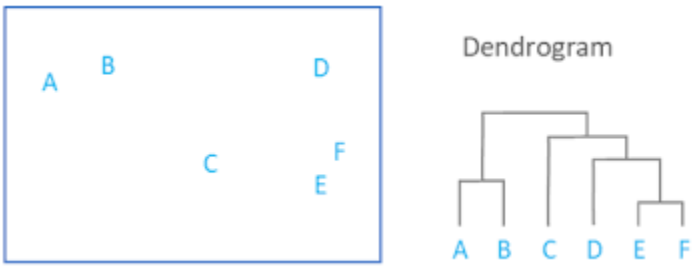


Imagen 2. *Grafico dendrograma, a la izquierda el conjunto de datos, a la derecha el resultado de la agrupación.*

K-means y clustering jerárquico son los algoritmos más conocidos del aprendizaje no supervisado, y en muchas ocasiones pueden solucionar el problemas que se plantea, pero estos tienen una gran debilidad y es que es necesario tener datos linealmente separables, para lograr una tasa de acierto

alta y útil para la resolución de problemas, en los casos donde los datos no son linealmente separables se generan errores en los grupos de datos que se clasifican, un ejemplo claro de ello es la **Imagen 3.** K-means aplicado a dos bases de datos que no son linealmente separables. En esta se puede apreciar como en ambos conjunto el algoritmo K-means intenta separarlos linealmente, pero la distribución de datos es diferente, teniendo como resultado un separación deficiente.

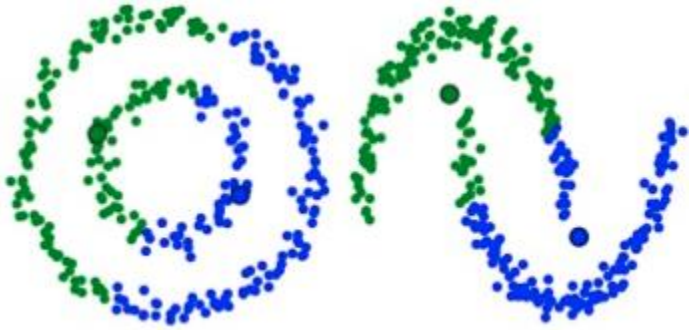


Imagen 3. *K-means aplicado a dos bases de datos que no son linealmente separables.*

Para tener resultados más apropiados sin tener una gran fijación por las características del conjunto de datos, nace spectral clustering, una técnica capaz de resolver estos problemas, porque en esta se busca a través de grafos encontrar las conexiones entre datos , para posteriormente realizar un clustering más acertado, esta técnica es la base del desarrollo de este trabajo, por ello en los próximos fragmentos se va a dar una introducción a spectral clustering y las diferentes formas de aplicar esta técnica, basado en la investigación de diferentes autores.

Spectral Clustering

Las técnicas de agrupación espectral utilizan el espectro (valores propios) de la matriz de similitud de los datos para realizar la reducción de dimensionalidad antes de la agrupación en menos dimensiones. La matriz de similitud se proporciona como una entrada y consiste en una evaluación cuantitativa de la similitud relativa de cada par de puntos en el conjunto de datos.

“La agrupación espectral puede arrojar resultados impresionantemente buenos donde la agrupación tradicional en busca de "manchas redondas" en los datos, como K-means, fracasaría miserablemente. Se basa en dos pasos principales: primero incrustando los puntos de datos en un espacio en el que los clústeres son más "obvios" (utilizando los vectores propios de una matriz de Gram), y luego aplicando un algoritmo de agrupamiento clásico como K-means. La matriz de afinidad M se forma usando un kernel tal como el kernel gaussiano. Se han propuesto varios pasos de normalización” (von Luxburg, 2007).

Si bien las ventajas del Spectral clustering que de ahora en más se denotará como SC, la convierten en una técnica bastante atractiva, tiene varios puntos en los cuales genera incertidumbre, ejemplo de ella es que no se una demostración teórica, de cuan alejado se puede estar de un resultado correcto al aplicar un algoritmo con SC, adicional a esto, en los diferentes algoritmo para encontrar la matriz de afinidad que es la base de éxito de la técnica, se tienen varios parámetros que no es posible sintonizar o afinar de forma general, y tampoco hay un trasfondo teórico para la selección de valor de los mismos.

A continuación se detallará el funcionamiento de SC y los fundamentos teóricos que sustentan esta técnica.

Grafos de similitud

Si se dispone de un conjunto datos y una función de similitud entre los mismos sij>=0 (para una pareja de puntos i, j), una buena representación de los datos es mediante el grafo de similitud G=(V,E). **V** es el conjunto de vértices y **E** es el conjunto de aristas. Es común que el grosor o color de las aristas denote el peso de la unión entre dos vértices.

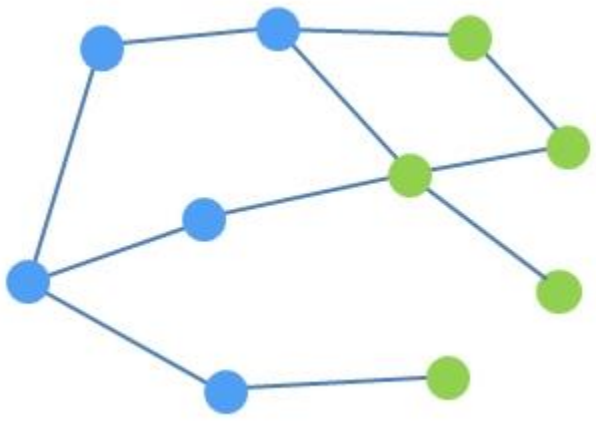


Imagen 4. *Estimación grafica de un grafo de similitud*

Teniendo en cuenta esta definición de gráfico, se podría replantear el problema de clustering, de tal manera que las aristas con pesos bajos, hagan referencia a aquellos vértices o registros que no tienen un alto grado de similitud y no se pueden incluir en un mismo clúster.

Dentro de los grafos de similitud, hay varios tipos, que se utilizan en algunos métodos de SC, (von Luxburg, 2007) dice que los tipos más comunes de grafos de similitud son los siguientes:

Grafo de vecindad ε

En este grafo se conectan todos los vértices cuyas distancias respecto a los demás sean menores a **ε**.

Grafo de K vecinos más cercanos

En este grafo cada vértice, se conecta con los ***K*** vértices cuyas distancias sean menores

Grafo totalmente conectado

Para este grafico cada vértice se conecta con cada uno de los demás vértices, por esta razón se llama totalmente conectado.

Grafos Laplacianos

La importancia de los grafos Laplacianos radica en "La idea de los algoritmos es usar las Laplacianas de forma de transformar los puntos originales a un nuevo espacio donde, como se verá más adelante, se realzan los clústeres y los puntos transformados se vuelven fácilmente separables. De esta forma, con la aplicación de algoritmos sencillos de clustering como por ejemplo K-means los puntos pueden clasificarse fácilmente" (von Luxburg, 2007).

Se debe tener en cuenta que no hay una convección única cuya matriz sea llamada grafo Laplaciano, para profundizar en detalles se recomienda (Chung, 1997), de este texto se van a exponer brevemente la formas de los tipos de grafos Laplacianos.

Grafo Laplaciano no normalizado

Está definido como se ve en la **Ecuación 1**.

$$L = D - W$$

***Ecuación 1**. Definición de Laplaciano no normalizado.*

En la **Ecuación 1**, D representa la diagonal ponderada de **W,** mientras **W** es un grafo de similitud o matriz de afinidad. (Chung, 1997) dice que las propiedades de este tipo de grafo son las siguientes.

- L simétrica y semi-definida positiva
- Menor valor propio de L igual a cero con vector propio asociado constante
- L tiene n valores propios reales positivos

Grafo laplaciano normalizado

Para el Laplaciano normalizado existen dos definiciones, que se verán a continuación en las ecuaciones **Ecuación 2**, **Ecuación 3**.

$$L_{sim} = D^{\frac{1}{2}} L D^{\frac{1}{2}} = I D^{\frac{1}{2}} W D^{\frac{1}{2}}$$

Ecuación 2. *Definición de grafo Laplaciano normalizado simétrico.*

$$L_{rw} = D^{-1} L = I - D^{-1} W$$

Ecuación 3. *Definición de grafo Laplaciano normalizado, por caminata aleatoria.*

Para **Ecuación 2** y **Ecuación 3**, **D** representa la diagonal ponderada de **W,** mientras **W** es un grafo de similitud o matriz de afinidad. (Chung, 1997) dice que las propiedades de este tipo de grafo son las siguientes.

- λ es valor propio de Lrw con vector propio μ. μ y λ cumplen L μ = λD μ.
- 0 es valor propio de Lrw con vector constante 1 como vector propio. 0 es valor propio de Lsim con vector propio $D^{\frac{1}{2}}1$.
- Lsim y Lrw son semi-definidas positivas y tienen n valores propios reales no negativos.

Algoritmos de Spectral Clustering

Para el desarrollo del método SC se utilizara el algoritmo presentado por, (Ng, Jordan, & Weiss, 2001) que se puede ver en el **Algoritmo 1** y cuya estructura esta descrita en la **Imagen 5**, de estos 5 bloques solo el de matriz de afinidad o kernel es variable para este documento, el resto del algoritmo es igual para los diferentes métodos que se van a tratar a continuación, el adecuado cálculo del bloque de matriz de afinidad o kernel, es la base de éxito del método SC, puesto que de este depende la realización del grafo de afinidad, de donde se van a obtener las características para hacer el proceso de clustering.

Algoritmo 1. *Algoritmo propuesto por (Ng, Jordan, & Weiss, 2001), para la realización del método spectral clustering.*

Entrada: Matriz de similitud $\boldsymbol{S} \in \boldsymbol{R^{nxk}}$ y numero $\boldsymbol{k}$ de clusters a construir.
Salida: Clústeres $\boldsymbol{A_1} \ldots \boldsymbol{A_k}$ con $\boldsymbol{A_i} = \{\boldsymbol{j} \mid \boldsymbol{\gamma_i} \in \boldsymbol{C_i}\}$.

- Construir un grafo de similitud por alguno de los métodos vistos en la **sección de grafos.**
- Calcular la Laplaciana normalizada $\boldsymbol{L_{sim}}$.
- Hallar los k primeros vectores propios de $\boldsymbol{L_{sim}}(\boldsymbol{u1}, \ldots . \boldsymbol{uk})$
- Sea $\boldsymbol{U} \in \boldsymbol{R^{nxk}}$ la matriz conteniendo los vectores propios $(\boldsymbol{u1}, \ldots . \boldsymbol{uk})$ como columnas.
- Formar la matriz $\boldsymbol{T} \in \boldsymbol{R^{nxk}}$ a partir de **U** normalizando sus filas, o sea $\boldsymbol{t_{ij}} = \frac{\boldsymbol{u_{ij}}}{\left(\sum_k \boldsymbol{u_{ij}^2}\right)^{\frac{1}{2}}}$
- Con i de 1 a n, sea $\boldsymbol{\gamma_i} \in \boldsymbol{R^k}$ el vector de la fila i de la matriz **T**.
- Clasificar los puntos $(\boldsymbol{\gamma_i})_{\boldsymbol{i=1\ldots.n}}$ en $\boldsymbol{R^k}$ con el algoritmo k-means en los clusters $\boldsymbol{C_1} \ldots \boldsymbol{C_k}$.

Imagen 5. *Diagrama de bloques, que muestra la metodología de trabajo seguida para los algoritmos SC.*

Kernel Gaussiano

Este algoritmo fue propuesto por (Ng, Jordan, & Weiss, 2001) y de ahora en adelante se nombrara a lo largo del documento como kernel base. Para el cálculo del kernel base se utiliza la **Ecuación 4** la cual representa un grafo totalmente conectado.

$$\boldsymbol{A_{ij}} = \boldsymbol{exp}\left(-\frac{\left\|\boldsymbol{x_i} - \boldsymbol{x_j}\right\|^2}{\boldsymbol{2\sigma^2}}\right)$$

Ecuación 4. *Ecuación para calcular el kernel base.*

Para la **Ecuación 4** $\|\boldsymbol{x_i} - \boldsymbol{x_j}\|^2$ representa la distancia euclidiana entre los valores x_i y x_j, que son elementos del conjunto inicial de datos. $\boldsymbol{\sigma}$ es un valor independiente, que se sintoniza dependiendo del conjunto de datos a trabajar.

Kernel, utilizando metodo Local Scaling

Este algoritmo fue propuesto por (Zelnik-Manor & Perona, 2004), y es básicamente una variación del kernel base propuesto por (Ng, Jordan, & Weiss, 2001), en el cual se busca afinar el parámetro $\boldsymbol{\sigma}$ dividiéndolo en dos parámetros $\boldsymbol{\sigma_i}$ **y** $\boldsymbol{\sigma_j}$ como se puede ver en la **Ecuación 5**. Los parámetros nuevos de esta ecuación se calculan utilizando un K-esimo vecino que es un valor de entrada del algoritmo y que para diferentes conjuntos de datos puede retornar diferentes resultados, según (Zelnik-Manor & Perona, 2004) el valor más óptimo para ***K*** es 7, cabe aclarar que esto no tiene ninguna sustentación matemática, y es un valor al cual el autor llegó a base de prueba y error, que retornará un vector con valores que darán los parámetros de afinación para ambos parámetros.

$$\boldsymbol{A_{ij} = exp\left(-\frac{\|x_i - x_j\|^2}{\sigma_i \sigma_j}\right)}$$

Ecuación 5. *Ecuación para calcular el kernel basado en el método local scaling.*

Al igual que en la **Ecuación 4**, para la **Ecuación 5** $\|\boldsymbol{x_i} - \boldsymbol{x_j}\|^2$ representa la distancia euclidiana entre los valores x_i y x_j, que son elementos del conjunto inicial de datos.$\boldsymbol{\sigma_i}$ **y** $\boldsymbol{\sigma_j}$ son valores que se calculan basándose en un parámetro ***K*** de entrada, con el cual se calcula el K-ésimo vecino, que resulta ser un vector de valores que en su estado original representan a $\boldsymbol{\sigma_i}$ y al estar traspuestos representan **a** $\boldsymbol{\sigma_j}$**.**

Kernel CNN (Common-Near-Neighbor)

Este algoritmo fue propuesto por (Xianchao , Jingwei , & Hong , 2011) y al igual que el método local scaling propuesto por (Zelnik-Manor & Perona, 2004), este método es una variación del kernel base propuesto por (Ng, Jordan, & Weiss, 2001), como se puede ver en la **Ecuación 6**.

$$A_{ij} = exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2(CNN(x_i, x_j) + 1)}\right)$$

Ecuación 6. *Ecuación para calcular el Kernel CNN.*

Para la ecuación **Ecuación 6** al igual que en la **Ecuación 4** $\|x_i - x_j\|^2$ representa la distancia euclidiana entre los valores x_i y x_j, que son elementos del conjunto inicial de datos. σ es un valor independiente, que se sintoniza dependiendo del conjunto de datos a trabajar. Para esta ecuación aparece un elemento nuevo **CNN**, que es una función que se encarga de sintonizar el valor de sigma de acuerdo al conjunto de datos, para obtener un mejor resultado.

CCN

Este método utiliza un **grafo de vecindad ε**, que posteriormente multiplica por el arreglo resultante traspuesto, este nuevo arreglo se usa para sintonizar los valores de sigma, de acuerdo al comportamiento del conjunto de datos.

Kernel Powered Gaussian

Este algoritmo fue propuesto por (Natalian & Yang, 2016) al igual que los métodos anteriores este método es una variación del kernel base propuesto por (Ng, Jordan, & Weiss, 2001), como se puede ver en la **Ecuación 7**.

$$A_{ij} = exp\left(-\frac{\|x_i - x_j\|^2}{\beta}\right)^{\gamma}$$

Ecuación 7. *Ecuación para calcular el Kernel Powered Gaussian.*

A diferencia del **kernel CNN y** el **kernel basado en el maetodo Local scaling,** este método además de modificar el valor de σ reemplazándolo por β también eleva el resultado a un valor equivalente a γ**.** El valor de β es igual al valor máximo, dentro de la fila cuyos valores sean los mínimos, con este se asegura que el valor de β varíe según el conjunto de datos, y el método se puede acoplar a diferentes sets de información. Por otra parte el valor de γ se calcula utilizando un algoritmo de umbralizacion en el cual el umbral es un valor estatico o de entrada (es sabido que el valor del umbral es inversamente proporcional a la precisión del algoritmo), en el algoritmo se calcula a cada iteración la **Ecuación 7**, y a su vez

se va sintonizando el valor de **γ** con un contador creciente, que se multiplica por un parámetro **c** de entrada, el umbral se mide como la media de los factores de correlación entre el valor actual y valor futuro, resultante de la **Ecuación 7.**

Kernel ALS (alignment segment)

Este algoritmo fue propuesto por (Alvarez-Meza, Castro-Ospina, & Castellanos-Dominguez, 2016), y a diferencia de las anteriores propuestas, no es una alteración al método propuesto por (Ng, Jordan, & Weiss, 2001), pero si usa las propiedades del kernel base, de tal manera se <<desarrolla un método de recorte gráfico que aprovecha si se estima correctamente la relación de las muestras de entrada, lo que permite extraer estructuras de datos complejas>> (Alvarez-Meza, Castro-Ospina, & Castellanos-Dominguez, 2016). La ecuación base del método se puede apreciar en la **Ecuación 8**.

$$\boldsymbol{K_{\varphi}(x_i, x_j; \sigma) = \varphi(x_i, x_j) k(x_i, x_j; \sigma)}$$

Ecuación 8. *Ecuación para calcular el Kernel ALS.*

Para la **Ecuación 8**, $\boldsymbol{K}$ es la matriz de similitud que codifica la estructura del gráfico $\boldsymbol{G(X, K)}$. Y $\boldsymbol{\varphi}$: $\boldsymbol{R^D x R^D \rightarrow R^+}$, es una función de base radial compactamente compatible. Para preservar la definición positiva de $\boldsymbol{K_{\varphi}}$ y para mejorar la consistencia de los datos locales y globales en $\boldsymbol{K}$, el operador $\boldsymbol{\varphi()}$ se elige como una función de dispersión **Ecuación 9**.

$$\boldsymbol{\varphi(x_i, x_j; b, v) = \left(\max\{1 - \frac{\|x_i - x_j\|^2}{b}, 0\} \right)^v}$$

Ecuación 9. *Función de dispersión, para la ejecución del kernel ALS.*

En la **Ecuación 9** $\boldsymbol{b \in R^+}$, la notacion $\boldsymbol{\max\{., 0\}}$ representa el valor máximo entre el argumento y cero. El término de potencia que gobierna el grado de suavidad de $\boldsymbol{\varphi()}$ se ajusta como $v \geq \frac{D+1}{2}$. Por lo tanto, la función de dispersión introduce un umbral difícil en **Ecuación 9**, haciendo que todas las entradas tengan una distancia $\boldsymbol{\|x_i - x_j\|^2 > b}$.

3. METODOLOGÍA

La idea principal del trabajo es realizar una librería que encapsule los diferentes **métodos** para hallar la matriz de afinidad con el fin de aplicarlos al método spectral clustering y ver el comportamiento de estos en el resultado final, para ello se utilizó como base el **Algoritmo 1**. Los algoritmos inicialmente se realizaron el lenguaje de programación Python y haciendo uso de las librerías:

- **Numpy:** Es una extensión de Python, que le agrega mayor soporte para vectores y matrices, constituyendo una biblioteca de funciones matemáticas de alto nivel para operar con esos vectores o matrices (NumPy).
- **Scipy:** Es un ecosistema basado en Python de software de código abierto para las matemáticas, la ciencia y la ingeniería (Scipy).
- **Networkx:** es un paquete de Python para la creación, manipulación y estudio de la estructura, dinámica y funciones de redes complejas (NetworkX).
- **Scikit-learn:** Es un conjunto de herramientas simples y eficientes para la extracción de datos y el análisis de datos, accesibles para todos y reutilizables en diversos contextos (Scikit-learn).
- **Matplotlib:** es una biblioteca de trazado 2D de Python que produce figuras de calidad de publicación en una variedad de formatos impresos y entornos interactivos en todas las plataformas (Matplotlib).

Es importante destacar que las librerías utilizadas en el desarrollo son de código abierto, lo que permite su uso libremente y evita cualquier tipo de conflicto legal, por el uso de las mismas. Este paquete de librerías se obtuvo utilizando anaconda, que es un framework el cual tiene varias librerías para diferentes usos, enfocado en las ciencias exactas y recolección de datos. Como IDE de desarrollo para Python se utilizó Spyder, que es un ambiente de desarrollo que permite la visualización de variables y graficas dentro del mismo entorno, además de facilitar la depuración y compilación del código desarrollado.

Después de tener todos los elementos necesarios para la realización del proyecto, se realizaron los algoritmos por separado, dando la posibilidad de que las personas que utilicen la librería desarrollado puedan realizar combinaciones entre algoritmos. Los algoritmos se realizaron siguiendo el esquema de la **Imagen 6**, con el fin de agilizar el desarrollo y enfocarse al final en la base del trabajo que es el diferente cálculo de las matrices de afinidad.

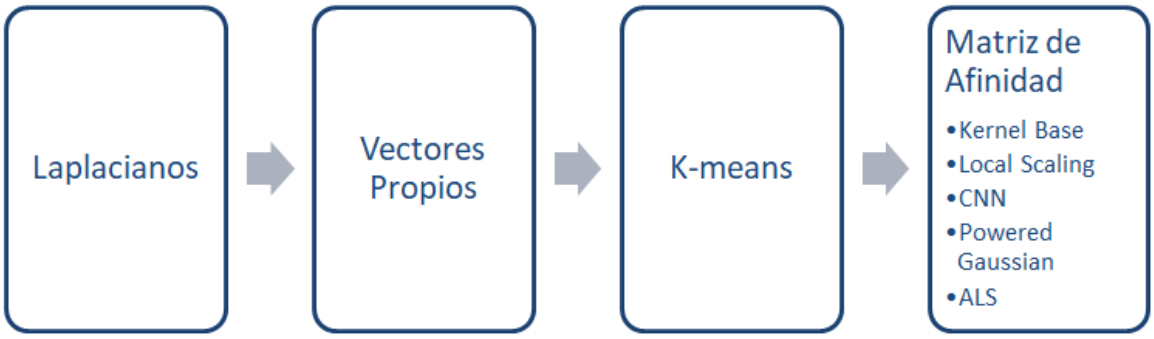


Imagen 6. *Diagrama de realización de la librería.*

Las primeras funciones que se realizaron fueron las de **Laplacianos** tanto el **normalizado** como el **no normalizado** en base a **Ecuación 1** y **Ecuación 2**, aunque para pruebas de los algoritmos se utilizó el **Laplaciano normalizado,** como paso a seguir se realizaron las funciones para encontrar los vectores propios y la realización de **K-means** respectivamente, dado que el cálculo de vectores y valores propios tiene cierta complejidad a nivel de algoritmos se utilizó la librería **Scipy** que brinda métodos para encontrar estos valores y vectores de una forma simple. En cuanto al método de clustering **K-means** se utilizó la librería **Scikit-learn**, con el fin de facilitar la realización del algoritmo y optimizar el tiempo de ejecución del mismo.

Teniendo las funciones base de la librería se procedió con el desarrollo de los algoritmos para la calcular la matriz de afinidad, para ello se realizó primero la función encargada de

calcular el **kernel base** puesto que los demás algoritmos de cálculo de la matriz de afinidad dependen en gran manera de este. Inicialmente se desarrolló utilizando ciclos para los cálculos de distancia dentro de la matriz, esto con el fin de tener un base con la que comparar los resultados que iba a arrojar el algoritmo en su forma de cálculo matricial, posteriormente se pasó la función a cálculos matriciales utilizando la librería **Scipy** que facilita el cálculo de estas distancias con la función **<<distance_matrix>>,** con ambos algoritmos realizados se compararon los resultados arrojados para comprobar el correcto desarrollo y de esta manera quedarse con el ultimo desarrollado puesto que genera un menor tiempo de procesamiento para el computador.

Teniendo el **kernel base** desarrollado, los algoritmos **local scaling**, **CNN** y **powered gaussian** se realizaron en base a este, realizando las variaciones mínimas que se presentaban entre los algoritmos, en el caso de **local scaling** se agregó un parámetro de entrada **K** con el fin calcular k-esimo vecino para afinar el valor de σ. En la caso de **CNN** se agregó un parámetro de entrada **ε** y se realizó una función adicional que recibía este parámetro y la matriz de datos inicial y retornaba un **grafo de vecindad ε,** necesario para afinar el valor de sigma de acuerdo al contenido del conjunto de datos. En el caso de la función encargada de retornar la matriz de afinidad con **powered gaussian,** se agregaron dos parámetros de entrada **c** y **m** para calcular el valor de γ, ademas se elimino el parametro σ, puesto que este se calculaba dentro de la función utilizando los parámetros de entrada nuevos. Para el desarrollo del algoritmo que calcula el **kernel ALS**, utilizaron tres parámetros de entrada: data → el conjunto de datos, f_d → Factor de densidad y un parámetro opcional K→ número de vecinos para construir, para las operaciones con grafos como rutas más cortas o creación de grafo a partir de una matriz de datos se utilizó la librería **Networkx,** con el fin de agilizar el desarrollo y centrarse en contenido principal que era el cálculo de la matriz de afinidad, en este algoritmo el costo computacional es muy alto en comparación con los algoritmos anteriores debido a la cantidad de iteraciones que realiza para el cálculo de las rutas más cortas, por lo cual se recomienda tener previamente el valor de k, con el fin de agilizar su ejecución.

Después de realizar los diferentes algoritmos en Python se realizó el paso de los mismos en Matlab para tener una librería que se pudiera usar en dos plataformas diferentes y aumentar la cantidad de beneficiados con el trabajo. Para el caso de Matlab se realizó el mismo esquema denotado en la **Imagen 6**, y básicamente se transformó el código Python a código en Matlab, con una excepción en el **kernel ALS**, en el cual se utilizó un toolbox **<<Toolbox_grafos>>** que facilitaba las operaciones con grafos, en este algoritmo se nota un mejora en el costo computacional respecto al realizado con Python.

Para realización de pruebas se utilizó una base de datos, encriptada en formato .mat que contenía diferentes conjuntos de datos, entre los cuales se utilizó happy **Imagen 7**.

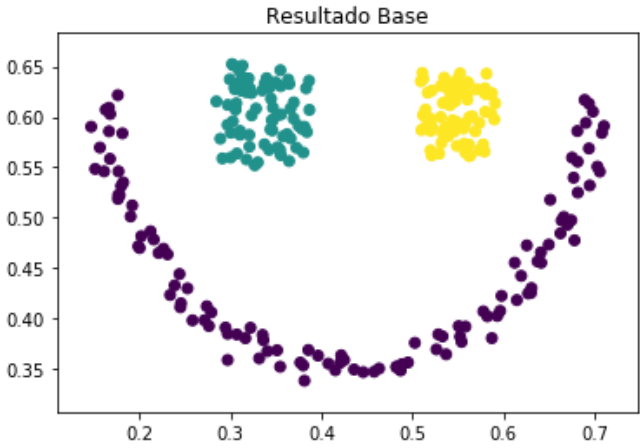


Imagen 7. *Vista graficada del conjunto de datos Happy.*

Para este conjunto de datos se calculó la matriz de afinidad siguiendo los diferentes algoritmos desarrollados y posteriormente se pasó por el algoritmo desarrollado en base a **Algoritmo 1**. Se graficaron los resultados de matriz de afinidad y resultado del spectral clustering y se compararon entre los sistemas desarrollados para Matlab y Python, además de esto se hizo una toma de tiempos de ejecución para comprobar que sistema brindaba un costo computacional menor y era más eficiente en la entrega de resultados, cabe destacar que para la toma de tiempos no se graficó ningún resultado puesto que esto costo computacional extra que no está asociado a la ejecución de los algoritmos, además estos tiempos se tomaron en repetidas ocasiones para sacar una media del

tiempo de ejecución de cada algoritmo, todos estos datos se consolidaron en tablas para hacer una comparación de rendimiento, eficiencia y eficacia de los algoritmos en ambos sistemas.

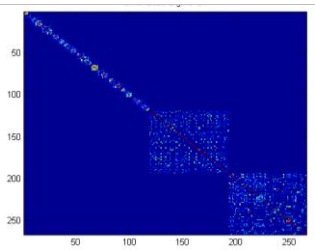
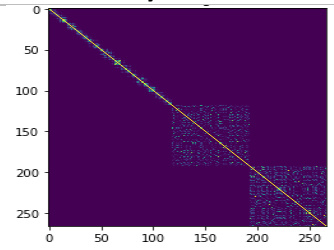
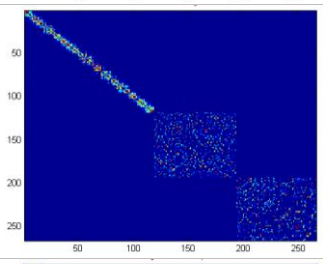
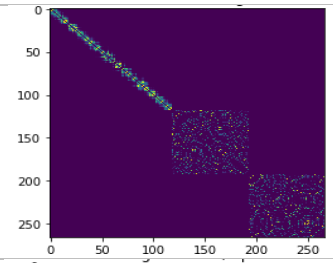
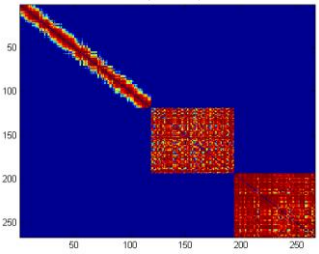
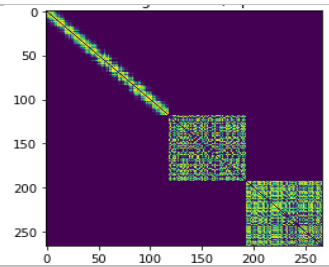
4. RESULTADOS Y DISCUSIÓN

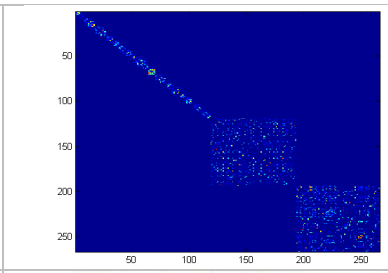
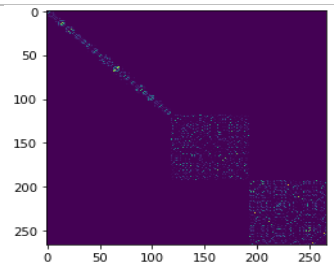
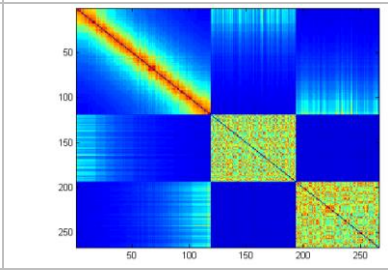
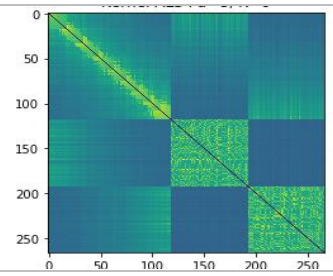
Matrices de afinidad

Se realizó la prueba utilizando la base de datos *happy*, para ellos se ejecutaron cada uno de los algoritmos por separado y se graficaron los resultados arrojados por Matlab y Python, para hacer una comparación visual del kernel o matriz de afinidad que se obtenía ejecutando la librería en las dos plataformas.

Tabla 1

Matrices de afinidad obtenidas en Python y Matlab para los métodos desarrollados

Algoritmo	Matlab	Python
Base Parametros: • Sigma = 0.7		
Local Scaling Parametros: • K=5		
CNN Parametros • Sigma=0.03 • Epsilon=0.05		

Powered Gaussian Parametros • C=0.7		
ALS Parametros • Fd = 8 • K=0		

Análisis

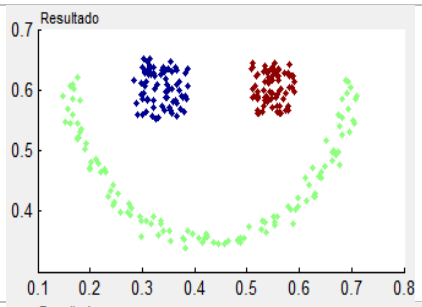
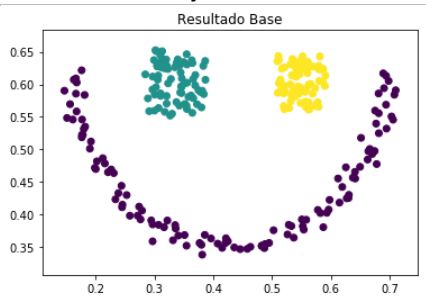
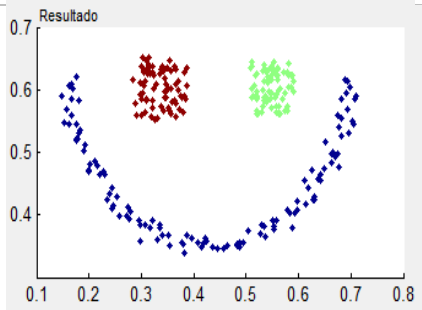
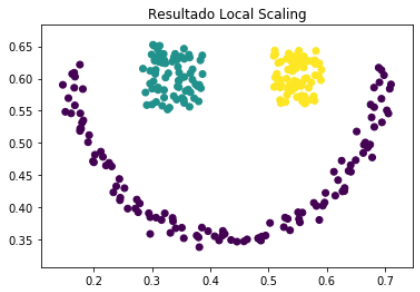
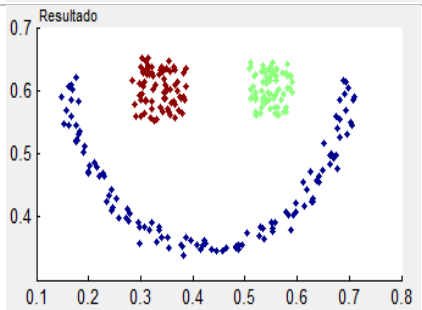
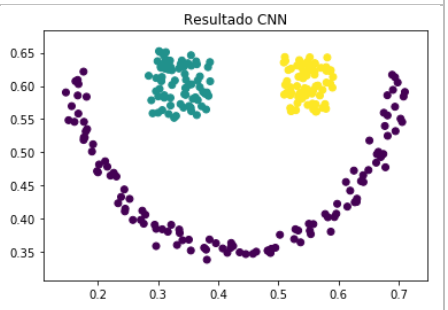
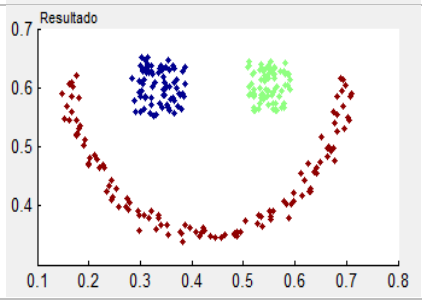
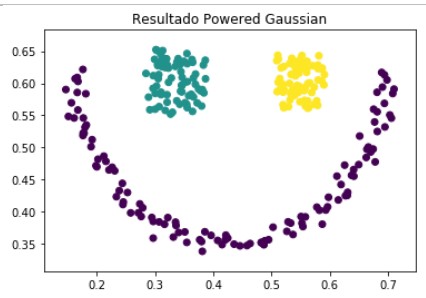
Se puede observar que los resultados obtenidos en las plataformas son prácticamente iguales, las variaciones que se presentan se ven dadas por la tabla de colores que usa el núcleo de cada lenguaje para la graficación de resultados. El Kernel ALS en Matlab es más marcado por cuestiones de desarrollo y uso de librerías, el grafo en Matlab estaba escalado por una variable incluida en la librería de un tercero, que se utilizó para la manipulación y operaciones con grafos. Teniendo en cuenta estas exposiciones se puede apreciar como a través de la diagonal que esta demarcado en las figuras hay simetría entre ambos lados, y en el análisis de grafos se pueden ver las agrupaciones de datos, que es el resultado y análisis que es realmente importante para las partes posteriores del desarrollo.

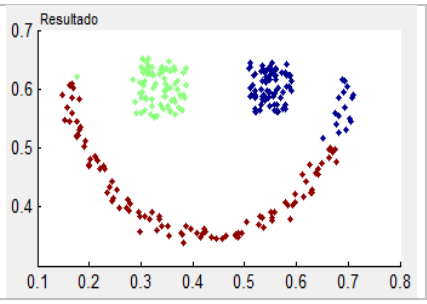
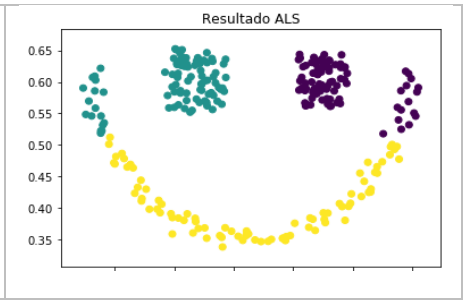
Resultados de los Algoritmos

Al igual que para las matrices de afinidad se realizó la prueba, agrupando mediante spectral clustering, para las matrices de afinidad previamente calculadas ver **Tabla 1**, y se graficaron resultados para Python y Matlab.

Tabla 2

Resultados obtenidos en Python y Matlab para los métodos desarrollados

Algoritmo	Matlab	Python
Base Parametros: • Sigma = 0.7		
Local Scaling Parametros: • K=5		
CNN Parametros • Sigma=0.03 • Epsilon=0.05		
Powered Gaussian Parametros • C=0.7		

ALS Parametros • Fd = 8 • K=0		

Análisis

Al igual que en el análisis de las matrices de afinidad, la principal diferencia entre los resultados se debe a la paleta de colores que usa el núcleo de las diferentes plataformas para graficar resultados. También es importante resaltar que el agrupamiento es dependiente de los vectores propios obtenidos del Laplaciano, estos vectores pueden ser positivos o negativos, realmente esto no afecta el funcionamiento del algoritmo, pero causa las etiquetas de clasificación cambien de orden y por esta razón se aprecia un cambio de colores en sectores, a través de los diferentes resultados. Por otra parte se puede observar cómo el error que se presenta en los resultados de Python para el algoritmo basado en un Kernel ALS, son más marcados, esto se debe al escalamiento que tiene el algoritmo para Matlab a la hora de buscar la ruta más corta de un grafo.

Tiempos de ejecución

Para la realización de esta prueba se ejecutaron los diferentes algoritmos por separado y para cada uno se hizo una toma de tiempo en milisegundos al inicio y final de la ejecución, utilizando librerías incorporadas en Matlab y Python. Para la toma de datos no se graficaron los datos, porque este tiempo de procesamiento gráfico no es realmente producido por la ejecución de los algoritmos.

Tabla 3

Toma de tiempo para los algoritmos realizados en Matlab

Algoritmo	Tiempo Medio	Desviación Estándar
Base	0,4375	0,205244022
Local Scalling	0,359375	0,14126014
CNN	0,0703125	0,022642776
Powered Gaussian	1,125	0,097994207
ALS	2,84375	0,373710166

Tabla 4

Toma de tiempo para los algoritmos realizados en Python

Algoritmo	Tiempo Medio	Desviación Estándar
Base	0,1328235	0,033839454
Local Scalling	0,1406365	0,027806952
CNN	0,0625043	0,014359327
Powered Gaussian	0,393579	0,112771797
ALS	42,98785	2,209967285

Análisis

Se realizaron 10 ejecuciones de cada algoritmo para cada librería realizado y se midieron los tiempos de ejecución, posteriormente se sacaron los valores medios de tiempos y la desviación estándar entre estos, y se tabularon los datos obtenidos.

Se puede observar una ligera mejoría en todos los valores que están en la Tabla 4**¡Error! No se encuentra el origen de la referencia.** con respecto a la Tabla 3Tabla 4**¡Error! No se encuentra el origen de la referencia.**; esto se lo atribuimos al uso, a la liberación de la unidad central de procesamiento y la memoria de acceso aleatorio, causada por el garbage collector del sistema. Es importante recalcar que, aunque hubo una leve mejoría entre tomas, los tiempos computacionales del algoritmo pueden considerarse estables, con una pequeña excepción en el ALS cuyos tiempos en Python son bastantes distantes.

En las tablas Tabla 3 y Tabla 4, también se puede observar una clara reducción del tiempo de procesamiento consumido por Python con respecto a Matlab, nuevamente con la excepción del ALS en el cual el tiempo de ejecución es mucho menor en Matlab.

En resumen, Python es superior en tiempos de ejecución en todos los algoritmos exceptuando el ALS, el cual tiene mayor rendimiento en Matlab dado que la librería que se usa para la manipulación de grafos, está desarrollada en C.

5. CONCLUSIONES, RECOMENDACIONES Y TRABAJO FUTURO

A medida que se realizó el trabajo se fueron encontrando diferentes conclusiones las cuales sustentaban el problema planteado inicialmente y al mismo tiempo reforzaban que el objetivo principal del proyecto si es viable.

Conclusiones

- El uso de diferentes métodos para generar un matriz de afinidad, es de suma utilidad a la hora de la realización de pruebas, puesto que estos ofrecen una mayor variedad de opciones que se pueden ajustar a diferentes conjuntos de datos.
- Aunque gran parte de los métodos para generar una matriz de afinidad, están basados en un Kernel gaussiano, el análisis y resultados que se obtienen con cada uno de ellos es muy variante y cada uno se ajusta mejor a ciertas distribuciones de información.
- El uso del algebra lineal, para el procesamiento matricial de los algoritmos, es una de las bases del éxito en el bajo consumo computacional y el performance de los mismos.
- Desarrollar los algoritmos en plataformas diferentes como Python y Matlab, aumenta el número de usuarios que pueden hacer uso de estos y así mismo se les puede dar un mejor aprovechamiento en los campos de investigación, teniendo una opción libre y otra que requiere pago.
- La realización de un demo para la ejecución de pruebas, brinda a los usuarios finales inexpertos una base para el desarrollo de nuevos módulos y el ensamble con las ya desarrollados.

Recomendaciones

- Para el desarrollo y utilización de los algoritmos se debe tener un conocimiento medio-alto de los lenguajes de programación Python o Matlab.
- Tener conocimientos de algebra lineal, puede ser de gran utilidad para el procesamiento matricial que se lleva a cabo dentro los diferentes algoritmos tratados en este trabajo.
- La documentación de las diferentes librerías utilizadas durante el desarrollo de los algoritmos de spectral clustering, es una herramienta vital para agilizar el desarrollo y codificar de manera más eficiente.

Trabajo Futuro

- Hacer pruebas y adaptar los algoritmos para segmentar imágenes en diferentes espacios de color como HSV y RGB.
- Desarrollar más métodos para la generación de la matriz de afinidad, en busca de una mejora en los tiempos de ejecución y precisión del algoritmo.
- Utilizar las librerías en ambientes con datos masivos para revisar el comportamiento que tienen

REFERENCIAS

Alvarez-Meza, A., Castro-Ospina, A., & Castellanos-Dominguez, G. (2016). Automatic graph pruning based on kernel alignment for spectral clusterin. *PatternRecognitionLetters*, 9.

Chung, F. (1997). *SPECTRAL GRAPH THEORY.* Philadelphia: University of Pennsylvania.

Gallegos, J. C., Torres Soto, A., Quezada Aguilera, F. S., Silva Sprock, A., Martínez Flor, E. U., Casali, A., y otros. (2014). *Inteligencia Artificial.* Chile: Iniciativa Latinoamericana de Libros de Texto Abiertos.

Matplotlib. (s.f.). *Matplotlib*. Obtenido de https://matplotlib.org/

Natalian, Y., & Yang, M.-S. (2016). Powered Gaussian kernel spectral clustering. *The Natural Computing Applications Forum 2017*, 16.

NetworkX. (s.f.). *NetworkX* . Obtenido de https://networkx.github.io/

Ng, A. Y., Jordan, M. I., & Weiss, Y. (2001). On Spectral Clustering:Analysis and an algorithm. *NIPS'01 Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, 8.

NumPy. (s.f.). *Numpy*. Obtenido de http://www.numpy.org/

Scikit-learn. (s.f.). *Scikit-learn*. Obtenido de http://scikit-learn.org/stable/index.html

Scipy. (s.f.). *Scipy*. Obtenido de https://www.scipy.org/

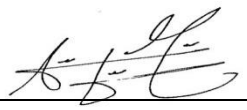
Steinbach, M., Karypis, G., & Kumar, V. (2000). A Comparison of Document Clustering Techniques. *semanticscholar*, 20.

von Luxburg, U. (2007). A Tutorial on Spectral Clustering. *springer*, 32.

Xianchao , Z., Jingwei , L., & Hong , Y. (2011). Local density adaptive similarity measurement for spectral clustering. *Science Direct*, 7.

Yadav, J., & Sharma , M. (2013). A Review of K-mean Algorithm. *International Journal of Engineering Trends and Technology*, 5.

Zelnik-Manor, L., & Perona, P. (2004). Self-tuning spectral clustering. *Advances in Neural Information Processing Systems 17*, 8.

FIRMA ESTUDIANTES 

FIRMA ASESOR Andrés Eduardo Castro O.

FECHA ENTREGA: 26/07/2018

FIRMA COMITÉ TRABAJO DE GRADO DE LA FACULTAD _____

RECHAZADO___ ACEPTADO____ ACEPTADO CON MODIFICACIONES_____

ACTA NO._____

FECHA ENTREGA: _____

FIRMA CONSEJO DE FACULTAD_____

ACTA NO._____

FECHA ENTREGA: _____