Universidad de Cantabria

Facultad de Ciencias, Grado en Físicas

Estudio de algoritmos de reconstrucción de vértices usando información del detector "MIPs Timing Detector" del Solenoide Compacto de Muones

Study of vertex reconstruction algorithms using information on the detector "MIPs Timing Detector" of the Compact Muon Solenoid

Trabajo de Fin de Grado

Autor: Antonio Gómez Carrera
Tutor: Pablo Martínez Ruíz del Árbol
Fecha: 1 de julio de 2023

A mis padres por todo su apoyo a lo largo de mis estudios.
A mi hermano, Alejo, por su admiración incondicional y por ser capaz de aguantarme a diario.
A mis amigos, por ser mi fuente de inspiración y superación.
A Pablo, por ser un gran profesor que me ha acompañado durante la realización de este trabajo.

Resumen

Se han estudiado 6 algoritmos de clustering distintos aplicados a la reconstrucción de vértices en el experimento CMS para el futuro HL-LHC, teniendo en cuenta la información temporal que proporcionará el MTD. Los algoritmos seleccionados son K-Means, MeanShift, DBSCAN, Agglomerative Hierarchical Clustering, Expectation Maximization-Gaussian Mixture Models y BIRCH. Los resultados obtenidos en estos algoritmos fueron comparados con el utilizado actualmente en el experimento CMS, "Deterministic Annealing", observándose que ninguno de ellos es capaz de mejorarlo en cuanto al acierto en la reconstrucción de vértices, siendo el algoritmo K-Means ($\textit{Vértices OK/Totales}_{\text{K-Means}} = 0.022 \pm 0.009$) el más cercano al nominal ($\textit{Vértices OK/Totales}_{\text{CMS}} = 0.034 \pm 0.010$). En cuanto a la asignación correcta de trazas a vértices, se observa que los algoritmos proporcionan valores similares e incluso superiores al "Deterministic Annealing" ($\textit{Trazas OK/ Trazas totales}_{\text{CMS}} = 0.49 \pm 0.03$), siendo el algoritmo Expectation Maximization-Gaussian Mixture Models el más destacado ($\textit{Trazas OK/Trazas totales}_{\text{EM-GMM}} = 0.26 \pm 0.02$). Con esta información se concluye que los algoritmos clásicos de clustering estudiados no mejoran de manera general la reconstrucción de vértices y por tanto no se recomienda su uso en el futuro HL-LHC.

Palabras Clave: Física de Partículas, Experimento CMS, HL-LHC, Algoritmos de Clustering.

Abstract

Six different clustering algorithms have been studied for vertex reconstruction in the CMS experiment for the future HL-LHC, taking into account the temporal information provided by the MTD. The selected algorithms are K-Means, MeanShift, DBSCAN, Agglomerative Hierarchical Clustering, Expectation Maximization-Gaussian Mixture Models, and BIRCH. The results obtained from these algorithms were compared with the currently used algorithm in the CMS experiment, "Deterministic Annealing". It was found that none of the studied algorithms improve upon "Deterministic Annealing" in correctly reconstructed vertices. Among them, the K-Means algorithm ($\textit{Vertices OK/Total}_{\text{K-Means}} = 0.022 \pm 0.009$) is the closest to the current algorithm ($\textit{Vertices OK/Total}_{\text{CMS}} = 0.034 \pm 0.010$). Neither, any of the studied algorithms improved "Deterministic Annealing" in well-assigned tracks ($\textit{Tracks OK/Total tracks}_{\text{CMS}} = 0.49 \pm 0.03$), with the Expectation Maximization-Gaussian Mixture Models algorithm being the most prominent ($\textit{Tracks OK/Total tracks}_{\text{EM-GMM}} = 0.26 \pm 0.02$). Therefore, based on this information, it is concluded that the studied classical clustering algorithms are not suitable for track-to-vertex assignment in the CMS experiment for the future HL-LHC.

Keywords: Particle Physics, CMS experiment, HL-LHC, Clustering Algorithms.

Índice

1 Introducción

1.1 Modelo Estándar

El contexto teórico de este trabajo se encuentra enmarcado en el modelo estándar de partículas. El modelo estándar es una teoría cuántica de campos que explica el contenido material del universo y sus interacciones, a través de campos fermiónicos y bosónicos, describiendo con esto la estructura fundamental del Universo, exceptuando la interacción gravitatoria, de la manera más precisa conocida hasta la fecha. Aún así, la teoría todavía no está completada ya que hay muchos fenómenos a los que no da explicación: la naturaleza de la materia oscura, cómo se relaciona la gravedad con el resto de interacciones, el problema de la jerarquía, entre otros.

El modelo estándar se fundamenta en la existencia de fermiones y bosones. Los fermiones son partículas de spin semientero, que se subdividen a su vez en leptones y quarks. Estas partículas y sus correspondientes antipartículas son las constituyentes de la materia conocida. Los leptones a su vez pueden ser neutros o cargados, los neutros son los neutrinos electrónicos, muónicos o taúnicos y las partículas cargadas son los electrones, muones y tau (en orden de masa creciente). Los seis quarks se agrupan en tres familias up-down, charm-strange y top-bottom (ordenado de menor a mayor masa).A su vez, los bosones tienen spin entero y son los mediadores de cada una de las interacciones, la fuerza fuerte es mediada por los gluones, la fuerza débil por los bosones $W^{\pm}$ y el Z^0 y la fuerza electromagnética es mediada por el bosón γ o fotón. La última partícula descubierta que entra dentro del marco teórico del modelo estándar ha sido el bosón de Higgs, descubierto en 2012 [1], siendo este el encargado de dotar de masa al resto de partículas. Todas estas partículas se organizan en función de sus propiedades en la tabla del modelo estándar (ver figura 1).

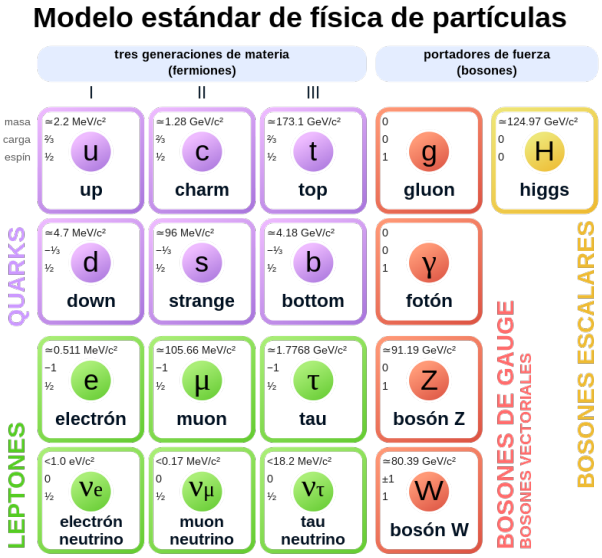


Figura 1: Tabla resumen de las partículas que componen el Modelo Estándar. Fuente [2].

1.2 The Large Hadron Collider

El LHC o Large Hadron Collider es un acelerador de partículas situado en el CERN o Conseil Européen de Recherche Nucléaire (Consejo Europeo de la Investigación Nuclear). Este dispositivo es el acelerador más grande jamás construido por el ser humano. Está situado cerca de Ginebra en la frontera entre Suiza y Francia y para la instalación de este acelerador se aprovecharon las instalaciones ya existentes del LEP (Large Electron-Positron Collider). Consta de un anillo de 27 km de circunferencia formado por materiales superconductores en el cuál se aceleran los protones hasta conseguir una energía en el sistema centro de masa de $\sqrt{s} = 13.6\text{TeV}$ (en la actualidad). Llegando a conseguir una luminosidad de $10^{34}\text{cm}^{-2}\text{s}^{-1}$. Con estos valores se han conseguido gracias a este acelerador grandes hitos de la física reciente como el descubrimiento del bosón de Higgs.

La aceleración de los protones en el LHC se consigue por medio de un complejo sistema de aceleradores sucesivos siendo el LHC el último de ellos (ver figura 2). En todos los aceleradores se usan campos eléctricos, diferencias de potencial, configuradas convenientemente para acelerar los protones; también se usan campos magnéticos para curvar y confinar los haces de protones, es decir, mantener los protones en una región del espacio lo más pequeña posible ([3]).

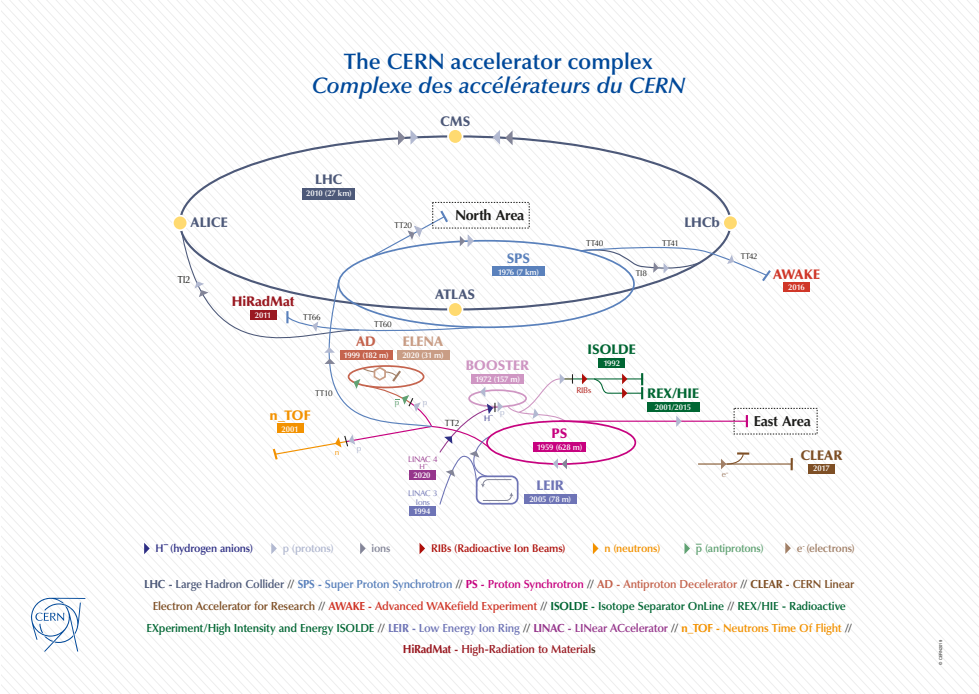


Figura 2: Mapa de los experimentos y túneles del LHC. Fuente [4].

Una vez los protones han sido acelerados hasta las energías deseadas, éstos colisionan entre sí (dos haces de protones viajando en direcciones opuestas colisionan) en 4 puntos del LHC dónde se encuentran los 4 experimentos principales LHCb, ATLAS, ALICE y CMS.

- **LHCb** o LHC beauty, se especializa en la búsqueda de discrepancias entre materia y antimateria que pueda explicar el predominio existente de la primera en el universo. Para ello estudia el decaimiento del llamado beauty-quark o b-quark.

- **ALICE** o A Large Ion Collider Experiment estudia la formación de estados de una muy alta densidad de energía formados principalmente por un plasma de quarks y gluones. Un estado que se formó en el Universo justo después del Big Bang, con esto se aspira, entre otras cosas, a entender los orígenes del Universo.

- **ATLAS** o A Toroidal LHC ApparatuS es uno de los experimentos con propósito general junto con CMS. ATLAS constituye el detector más grande del LHC con una longitud de 44 metros de largo y un peso de 7000 toneladas.

- **CMS** o Compact Muon Solenoid ([5]) es el otro experimento con propósito general del LHC. Pese a su diseño compacto, con una longitud de 22 metros, CMS es el detector más pesado del LHC, con un peso total de 14000 toneladas. Además, cabe destacar que el solenoide con el que cuenta este experimento, es el solenoide más potente jamás construido.

1.3 The Compact Muon Solenoid

El experimento CMS se caracteriza principalmente por su gran electroimán que produce un campo magnético de 3.8 Tesla. Con este gran campo magnético se consigue curvar las partículas cargadas, según la fuerza de Lorentz, de manera que se puede inferir el momento y carga de las partículas una vez detectadas en función de la curvatura.

Debido a la simetría cilíndrica existente en el acelerador todos los detectores de CMS siguen formas cilíndricas. Para conseguir esta forma los detectores de CMS se dividen en dos partes, el barril o barrel y las tapas o endcaps. Los principales detectores de CMS son (de posición más interna a más externa) el tracker, calorímetro electromagnético y hadrónico y cámaras de muones [5].

- Tracker. Es el encargado de detectar de manera precisa (con un error inferior a 10μm) la posición de las partículas. Midiendo la posición de la partícula en detectores sucesivos se reconstruye la trayectoria de la partícula por lo que se conoce su curvatura y así su momento. El tracker está formado por sucesivas capas de detectores de tipo pixel y de tipo strip, ambos basados en detectores de silicio. Al ser la parte más interna de los detectores, el tracker recibe la mayor cantidad de partículas y así la mayor dosis de radiación por lo que sus materiales han de ser muy resistentes a sus efectos.

- Calorímetro electromagnético o ECAL, encargado de medir la energía de los fotones y electrones (magnitud muy relacionada con el descubrimiento y caracterización del bosón de Higgs). El ECAL utiliza una tecnología de centelleadores de tungstanato de plomo, de manera que con un fotodetector unido a cada cristal convierte la energía depositada por la partícula en una señal eléctrica que es amplificada y analizada posteriormente.

- Calorímetro hadrónico o HCAL, mide la energía de los hadrones, partículas formadas por quarks, y proporciona de manera indirecta información sobre partículas hadrónicas. HCAL es un calorímetro de muestreo (sampling calorimeter) lo que quiere decir que ha sido diseñado de forma que mide la energía depositada, la posición y el momento de llegada de las partículas usando sucesivas capas de material absorbente y

centelleador que produce un rápido pulso de luz cuando una partícula pasa por él. Este pulso es recolectado en los fotodetectores donde produce una señal eléctrica. Las sucesivas capas son necesarias principalmente por las grandes cascadas hadrónicas que se producen cuando una de estas partículas entra en contacto con algún material, se necesita al menos un metro de material para detenerlas completamente.

- Cámaras de muones. La detección de muones es una de las tareas más importantes en CMS. Dado que estas partículas penetran varios metros de material sin casi pérdidas de energía, no han sido frenadas en ninguno de los calorímetros. Por tanto, las cámaras de muones se pueden situar en la parte más externa de CMS donde solo los muones producirán una señal clara. En las cámaras de muones se usan tres tipos de detectores: tubos de deriva, para mediciones de la trayectoria en el barril; cámaras de tiras catódicas, para medir la trayectoria en las tapas; y cámaras de tira resistiva, para recibir una señal rápida que pueda ser usada en el trigger.

Con la información de estos detectores se reconstruyen las distintas colisiones individuales protón-protón. En particular, combinando la información del tracker con el resto de detectores es posible reconstruir la trayectoria de las partículas cargadas. Este proceso es llevado a cabo por algoritmos de reconstrucción que agrupan y ajustan estadísticamente los puntos de detección en trayectorias compatibles con el campo magnético. Estas trayectorias reconstruidas se llaman comúnmente trazas.

1.4 Reconstrucción de trazas y vértices en CMS

Las partículas cargadas dejan pequeños depósitos de energía a medida que su trayectoria pasa a través del tracker y el sistema de muones. Estos depósitos son reconstruidos como trazas, parametrizadas a través de su posición, momento, dirección y sentido de movimiento. Con un algoritmo iterativo se consigue reconstruir esta trayectoria. En la primera iteracicón se seleccionan las trazas más sencillas de ajustar y se dejan para la última iteracción las más complejas donde la mayoría de los hits ya están asignados. En cada iteración el algoritmo sigue distintas fases. Empieza por predecir la posición de la partícula en la siguiente subcapa de detectores a partir de la información en la primera subcapa. Esta fase se conoce como "seeding". Una vez se tienen suficientes hits en las subcapas más internas del tracker, se tiene una semilla de traza, con esto se entra en la siguiente fase de la reconstrucción el "pattern recognition". Con esta semilla se parametriza la trayectoria de la partícula y se extrapola la posición en la siguiente subcapa. Para ello se utiliza el Kalman Filter [6] que es capaz de extrapolar la posición al siguiente detector asociándole un error. Si esta extrapolación coincide con un nuevo hit en esta siguiente subcapa, se añade este nuevo punto a los ya disponibles de manera que se puede encontrar con mayor precisión el siguiente hit. Esto se conoce como "fitting". Una vez se ha repetido este proceso con todas las subcapas se realiza un último ajuste y, si fuese necesario, se descarta algún hit discordante de la trayectoria. Este último paso que sirve para afinar la trayectoria se conoce como "smoothing". Con todo esto ya se tiene la trayectoria de la partícula, es decir su posición, momento, dirección y sentido. El momento se calcula a partir de la curvatura de la trayectoria ya que el campo magnético es conocido.

El punto que se toma como referencia para parametrizar las trazas es el punto de mayor proximidad a la línea del haz, conocido como "POCA" (Point of Closest Approach). Por

tanto, las variables que definen una traza son: $(x, y, z)_{\text{POCA}}$, los ángulos ϕ y η[1] , y la variable q/p_t siendo q la carga de la partícula y p_t su momento.

Con todo esto se tiene la información en z de todas las trazas por lo que es posible inferir donde se encuentran los vértices, es decir, los puntos de colisión protón-protón de donde han surgido todas estas partículas. Para ello, se utiliza un algoritmo llamado "Determinstc Annealing" [7], el cuál tiene una inspiración termodinámica, en el que se construye una función de likelihood que involucra la distancia entre cada traza y el vértice propuesto. Minimizando de manera iterativa este algoritmo se encuentra la asignación de cada traza a su respectivo vértice. Este algoritmo a parte de las posiciones de las trazas tiene en cuenta el error en su detección y su momento transverso para así asignar un peso a cada traza. De manera que las trazas más importantes son aquellas con menor error en la detección y mayor momento transverso.

1.5 The High Luminosity Large Hadron Collider

Para inicios de 2029 está previsto el arranque del High Luminosity Large Hadron Collider (HL-LHC) [8], el cuál es un acelerador construido en el mismo túnel que el LHC, y cuya principal diferencia reside en el aumento del número de colisiones por segundo. El HL-LHC tendrá una luminosidad 10 veces más grande que la del LHC, dando lugar a un promedio de colisiones espúreas de entre 140 a 200, cuando en el LHC es de tan sólo 40. Las colisiones espúreas se relacionan con el pile-up, siendo este una medida del número de colisiones simultáneas que se producen en el acelerador. Por tanto, a mayor número de colisiones espúreas se tiene un mayor pile-up y viceversa.

El incremento de luminosidad del HL-LHC dotará a sus diferentes experimentos de una mayor cantidad de colisiones a estudiar. Los objetivos de los experimentos pasan por la observación y medida de fenómenos del modelo estándar, como por ejemplo la producción doble de bosones de Higgs, así como las búsquedas exhaustivas de partículas más allá del modelo estándar (BSM, por sus siglas en inglés).

1.6 The MIPs Timing Detector

El MIPs Timing Detector (MTD) [9] es un nuevo detector que se va a introducir en el experimento CMS para así adaptarlo a las nuevas condiciones de alta luminosidad que se tendrán en el HL-LHC. Este detector medirá el tiempo de paso de las partículas con una precisión de entre 30-60 ps, es decir, añadirá información temporal a las trazas.

El MTD, al igual que el resto de detectores de CMS, tienen una estructura cilíndrica por lo que se subdivide en dos partes: el barril (Barrel Timing Layer, BTL) basado en tecnología de cristales centelleadores y las tapas (Endcap Timing Layer, ETL), basado en sensores de silicio de tipo Low-Gain Avalanche Diodes, LGADs. Con ambas partes trabajando en sincronía el detector será capaz de detectar partículas con una pseudorapidez $|\eta| \leq 3$. La figura 3 muestra una imagen esquemática del MTD.

[1] o pseudorapidez que se define como $\eta = -\ln(\tan\theta/2)$ siendo θ el ángulo polar medido desde el eje z tomado como la dirección en la que circulan las partículas.

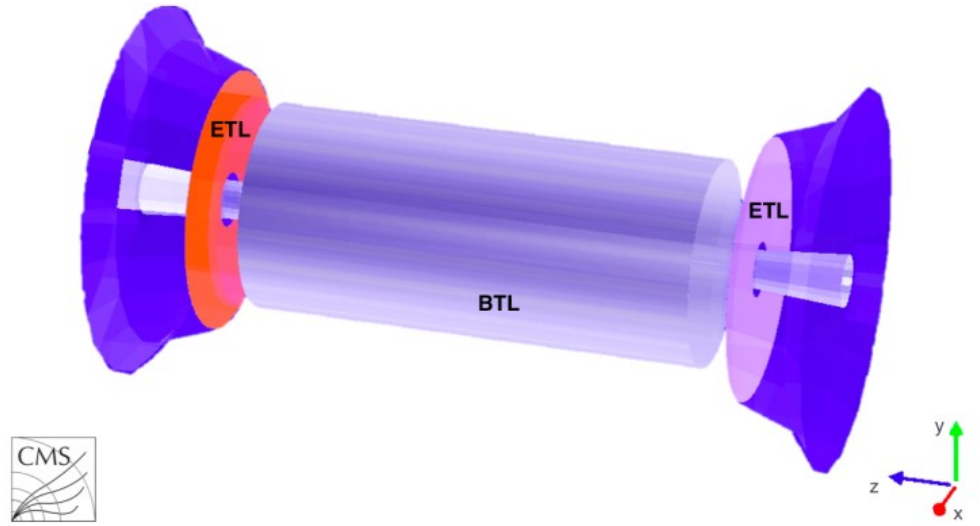


Figura 3: Vista esquemática del MTD implementado para las simulaciones. Se observa el barril de este detector en gris y las tapas en azul. Fuente: [9].

Al realizar una medida temporal de cada partícula se ha de añadir esta información a la traza. Para ello, se va a extrapolar cada trayectoria hasta el MTD y así medir el tiempo de paso de la partícula en este punto en relación al momento de cruce de los haces. Esta medida temporal no se corresponde con el tiempo en POCA. Conociendo con exactitud el momento de la partícula y su masa o su energía se puede reconstruir la trayectoria a la inversa (hacia el pasado) y así obtener su tiempo en POCA. Sin embargo, la identificación de las partículas individuales no es sencilla en CMS. En el caso de electrones y muones sí se hace ya que son partículas cargadas, dejan una traza en el tracker, y, además se detectan en el ECAL y en las cámaras de muones, respectivamente. Por otro lado, los hadrones no se distinguen entre sí. Principalmente se producen $\pi^{\pm}$ (piones), p^{+} (protones) y $k^{\pm}$ (kaones), siendo los piones el hadrón más probable. Por tanto, en un primer análisis se toma que la masa de todos los hadrones es $m_{\pi^{\pm}} = 139.57071 \pm 0.00053$ MeV [10]. Con esto se tiene la información necesaria para obtener el tiempo en POCA y así poder reconstruir la traza.

El principal uso del MTD consistirá en detectar el tiempo de paso de las partículas en relación con el momento en el que ambos haces de protones se cruzaron en el acelerador. De esta manera se añade a cada partícula la dimensión temporal. Con esta información se facilita la discriminación entre los distintos vértices; y, a su vez, es más sencillo establecer cuál es el vértice primario. El vértice primario es el de mayor interés físico ya que es la principal fuente de sucesos con una alta energía del centro de masas. Por otra parte, distinguir mejor los vértices entre sí se traduce en una reducción efectiva del pile-up, lo que genera un impacto y una mejora de calidad de los resultados a todos los niveles del programa de física de CMS.

1.7 Objetivos

Es necesario realizar un estudio de nuevos algoritmos de clustering ya que aplicando el algoritmo "Deterministic Annealing", que es utilizado actualmente en el LHC, se han obtenido resultados muy pocos satisfactorios en dos dimensiones (z-t). Al añadir la información temporal de las trazas se esperaba poder mantener la calidad de los resultados pese al aumento de la luminosidad en el HL-LHC; pero, en las simulaciones realizadas no se mejoran los resultados de una manera significativa con la información temporal. Por tanto, es necesario estudiar distintos tipos de algoritmos que mejoren el desempeño de asignación de trazas a vértices para así mantener la calidad exigida al experimento CMS.

El objetivo de este trabajo es realizar un estudio sobre distintos algoritmo clásicos de clustering para ver su utilidad en la asignación de vértices y trazas en el futuro HL-LHC. Para ello se han seleccionado seis algoritmos (K-Means, MeanShift, DBSCAN, Agglomerative Hierarchical Clustering, EM-GMM y BIRCH) ya existentes y basados en distintos principios, para así estudiar su desempeño al ser aplicados en este problema. Para comparar los algoritmos entre sí ha sido necesario establecer una métrica que evalúe su desempeño de una manera objetiva.

2 Algoritmos de Clustering

El clustering es un problema enmarcado en el contexto del machine learning y que consiste en la agrupación de conjuntos de datos no etiquetados en función de sus similitudes. Es decir, a partir de un conjunto de datos inicial se forman distintos subconjuntos. Los datos dentro de cada subconjunto comparten alguna propiedad, o en general son más parecidos entre sí que al resto de datos en otros subconjuntos. Estos subconjuntos o clusters sirven para identificar patrones o para obtener resultados de segmentación, principalmente. En este capítulo se muestra una descripción de diferentes estrategias de clustering ampliamente utilizadas en la actualidad.

2.1 K-Means

El algoritmo K-Means se basa en minimizar la distancia de los puntos a su respectivo centro o centroide. Para ello es necesario introducir previamente el número de centros deseados, posteriormente se calcula la distancia de cada punto a los distintos centros y se asigna cada punto al cluster perteneciente al centro más cercano. A continuación, se calcula una nueva posición de los centros como la media aritmética de los puntos pertenecientes a cada cluster. Con esto se obtiene la posición de los centros de la siguiente iteración. Este proceso se repite hasta que se alcanza la convergencia o se llega al número máximo de iteraciones.

Este algoritmo tiende a minimizar la inercia, la cuál coincide con la suma al cuadrado de las distancias al centroide, de la siguiente manera:

$$\sum_{i=0}^{n} \min_{u_j \in C}(||x_i - u_j||^2) \tag{1}$$

Siendo x_i las coordenadas de cada punto, u_j el centroide calculado de cada cluster y C el conjunto de todos los clusters.

Para el correcto funcionamiento de este algoritmo se asume que los conjuntos de datos están distribuidos de manera isótropa y tienen tamaños y varianza similares, todo esto se trata con más detalle en [11].

Con todo esto, este algoritmo presenta principalmente dos grandes ventajas, su baja complejidad y que su tiempo de computación escala como $O(n)$ es decir, de manera lineal con el número de punto a clusterizar. Y, por tanto, es un algoritmo rápido y que no requiere de una gran potencia de computación.

Por otra parte, una de sus principales desventajas reside en el hecho de que el algoritmo no es capaz de seleccionar automáticamente el número de clusters. Por lo tanto, es necesario proveer al algoritmo esta información, lo cuál es difícil en muchos problemas, como el que se va a tratar en este trabajo. Además, al partir de una primera distribución aleatoria de centros no siempre alcanza la misma configuración final para el mismo conjunto de datos, es decir, es posible alcanzar distintos mínimos locales de la inercia. Para subsanar esto el algoritmo implementa un número de distintas inicializaciones hasta encontrar la más óptima, aunque esto implica un escalado en los tiempos de computación.

Si se cumplen las condiciones para que el algoritmo funcione adecuadamente se obtienen resultados como los mostrados en la figura 4.

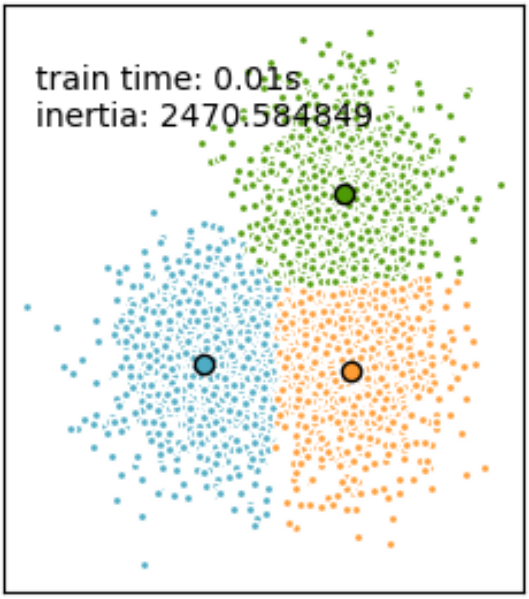


Figura 4: Resultado de aplicar el algoritmo K-Means a un conjunto de datos sencillo formado por 3 clusters diferentes. Los puntos resaltados se corresponden con el centroide de cada cluster. Fuente: [12].

2.2 Mean-Shift

Este algoritmo se basa en identificar zonas con una alta densidad de puntos para localizar en ellas el centro de los clusters. Para ello, se selecciona un punto inicial aleatorio y un circulo de radio r a su alrededor. Posteriormente, se calcula la posición media de los puntos que se encuentran dentro de este círculo. De forma trivial se puede demostrar que si la distribución de puntos dentro del círculo no tiene una densidad constante, de manera general, esta posición media no coincidirá con el punto central inicial. Este proceso se repite con el nuevo punto central y se genera un nuevo círculo hasta que se alcanza la convergencia. Esta técnica de actualizar la posición se conoce como "hill climbing", la cuál para un centroide dado con coordenadas $\vec{x}$ en la iteración t actualiza su posición como:

$$\overrightarrow{x^{t+1}} = \overrightarrow{x^t} + \overrightarrow{m(x^t)} \tag{2}$$

siendo $\vec{m}$ el vector que tiene en cuenta el desplazamiento del centroide dado en la iteración t. Para calcular este vector se define $N(\overrightarrow{x})$ como el número de elementos que conforman el conjunto de los puntos que se encuentran separados del punto $\overrightarrow{x}$ una distancia menor o igual que r, es decir, los que se sitúan dentro del círculo. Entonces:

$$\overrightarrow{m(x)} = \frac{1}{N(\overrightarrow{x})} \sum_{\overrightarrow{x_j} \in N(x)} \overrightarrow{x}_j - \overrightarrow{x} \tag{3}$$

Esto conlleva a una de sus principales ventajas: no es necesario inicializar el número de clusters, ya que el propio algoritmo ajusta este parámetro en función de los datos. Aún así, la selección del radio r de este círculo no es algo trivial y, para ello, se utiliza un parámetro llamado "bandwidth" o ancho de banda. Este parámetro selecciona el radio máximo del círculo. Para seleccionar el "bandwidth" se tienen funciones encargadas de seleccionar el valor más óptimo en función del conjunto de datos y la métrica utilizada.

Por otra parte, su mayor inconveniente se encuentra en que el tiempo de cómputo del algoritmo escala como $O(T*n*\log(n))$, donde T es el número total de puntos y n el número de muestras, que en principio coincide con T. Esto genera que su tiempo de ejecución se incremente respecto a otros algoritmos con un escalado menor.

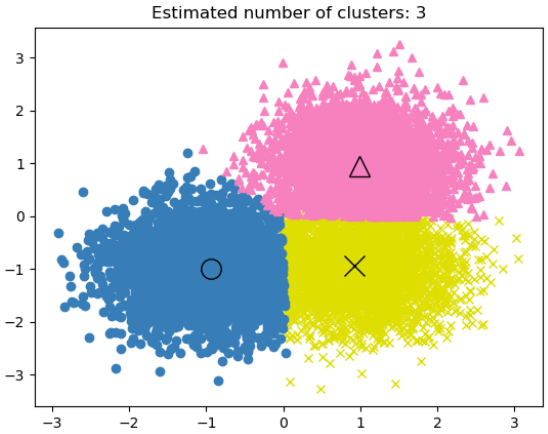


Figura 5: Resultado de aplicar el algoritmo MeanShift a un conjunto de datos sencillo. Se observa que el algoritmo reconstruye los 3 clusters distintos. La figura geométrica situada en cada cluster se corresponde con su centroide. Fuente: [13].

En la figura 5 se muestra un ejemplo del desempeño del algoritmo MeanShift al aplicarse en un conjunto de datos problema sencillo.

2.3 DBSCAN

El algoritmo DBSCAN (por sus siglas en inglés, Density-Based Spatial Clustering of Applications with Noise) separa los cluster en función de la densidad de puntos. Para ello, el algoritmo encuentra las zonas de más alta densidad y expande los cluster alrededor de estas zonas. De esta manera se consigue que cada cluster tenga una forma propia y, en general, diferente al resto, algo que no sucede con otros algoritmos.

Este algoritmo elige un punto arbitrario como origen y calcula los vecinos de dicho punto con radio ϵ. Si se encuentran los suficientes vecinos (esto se establece por el otro parámetro del algoritmo, $minPoints$) comienza el proceso de formación de un cluster. Para ello, añade todos estos puntos a un cluster y se repite el proceso con los puntos nuevos hasta que se completa la formación del cluster, es decir, se han visitado todos los puntos del cluster y no es posible añadir ningún punto extra. En cambio, si no se encuentran suficientes vecinos, se guardan los puntos como ruido. Y se repite el proceso partiendo de un nuevo punto no visitado anteriormente.

Este algoritmo presenta varias ventajas. En primer lugar, el algoritmo no requiere una estimación del número inicial de clusters, al contrario que K-Means, por ejemplo. En segundo lugar, el algoritmo es capaz de identificar los valores atípicos o outliers como ruido. Y, de manera menos interesante para el problema a tratar, pero importante de destacar, identifica clusters de tamaños y formas arbitrarios.

De forma similar, su principal desventaja se encuentra en que el algoritmo no agrupa de manera adecuada conjuntos de datos formados por cluster de muy distinta densidad entre sí. Esto se debe a que cuando se varía la densidad de puntos en cada cluster se

necesitan valores de ϵ y $minPoints$ diferentes para cada uno. A priori, esto no ha de influir gravemente en el problema estudiado[2], ya que los clusters tienen densidades de puntos similares.

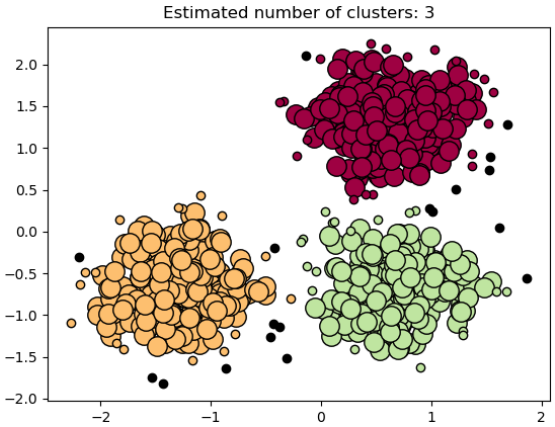


Figura 6: Resultado de aplicar el algoritmo DBSCAN a un conjunto de datos sencillo. Se observa que el algoritmo reconstruye los 3 clusters distintos. Además, con puntos más grandes se muestran los datos iniciales que el algoritmo a tomado como inicializaciones del cluster, con puntos más pequeños los valores que pertenecen al cluster pero no sirven para inicializarlo en el algoritmo y en negro los valores considerados como ruido. Fuente: [14].

En la figura 6 se muestra como funciona este algoritmo para un conjunto de datos sencillo. DBSCAN es capaz de identificar los outliers como ruido (dibujados en negro en la figura) así como cada uno de los clusters que componen el conjunto de datos. El algoritmo no detecta automáticamente los centroides de los clusters aunque estos son sencillamente calculables como el valor medio de la posición de los puntos que pertenecen a dicho cluster.

2.4 Agglomerative Hierarchical Clustering

Los algoritmos de clustering jerárquicos trabajan uniendo o separando clusters anidados de manera consecutiva. Esto es fácilmente visualizable representando los clusters en un esquema de árbol o dendograma, como se muestra en la figura 7, siendo la raíz el único cluster que engloba todo y las hojas, la parte superior del algoritmo, cada punto por separado. Existen dos formas de trabajo para este algoritmo bottom-up o top-down, la primera de ellas empieza desde la raíz del esquema y va separando los clusters progresivamente hasta llegar al resultado deseado. Y, la segunda comienza con cada punto por separado y va uniendo los dos más cercanos en cada iteración hasta llegar al resultado. Para todo esto, es necesario definir una métrica en los clusters. Usualmente se trabaja con la media de la distancia de los puntos de un cluster a otro. Pese a la importancia que tiene la métrica en otros algoritmos, en este en concreto, el resultado es casi independiente de la métrica escogida ya que se acaba recuperando la estructura jerárquica.

El algoritmo concreto utilizado en este estudio trabaja en formato bottom-up uniendo consecutivamente las muestras más próximas. Además se tienen en cuenta tres tipos de métrica: “Ward” la cuál minimiza la suma cuadrática de distancias entre todos los clusters.

[2]Estudiando a fondo la distribución de los datos se observa que sí influye más de lo esperado. Ver apartado 6.1

"Complete linkage" minimiza la máxima distancia entre pares de clusters. "Average linkage" minimiza la distancia entre todos los puntos de cada par de clusters y "Single linkage" minimiza la menor distancia entre cada par de clusters.

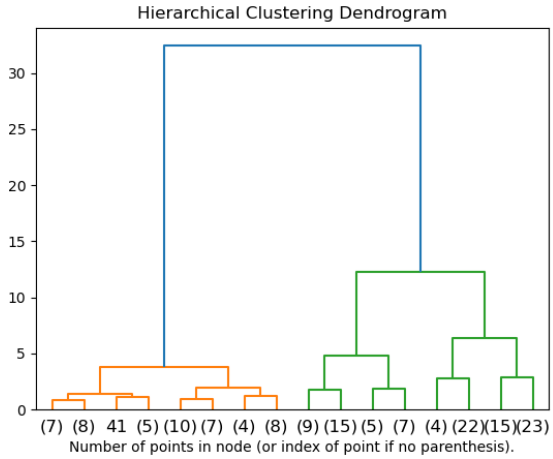


Figura 7: Dendograma correspondiente a aplicar el algoritmo agglomerative herarchical clustering a un conjunto de datos. Se ha truncado el esquema de manera que solo se muestren los tres niveles superiores. Fuente: [15].

Las principales ventajas de este algoritmo se encuentran en que no es necesario especificar el número de clusters deseado. No es vital definir una métrica correcta ya que esta tiene poca importancia a la hora de los resultados y es un algoritmo muy útil para recuperar estructuras jerárquicas.

Por otra parte, su principal desventaja se encuentra en su costo computacional ya que este crece como $O(n^3)$ lo que genera que al aumentar el número de puntos iniciales dispare el tiempo de ejecución.

2.5 Expectation Maximization - Gaussian Mixture Models (EM-GMM)

En este algoritmo se asume que los datos siguen un gaussian mixture model (modelo de mezcla gaussiana), es decir, están distribuidos siguiendo una o un número finito de distribuciones gaussianas, lo que da una mayor flexibilidad que restringirse únicamente a la media como con el algoritmo K-Means. Posteriormente se le aplica un algoritmo de optimización, en concreto, expectation maximization (maximización de expectativas).

De esta manera, cada cluster queda caracterizado por su valor medio y una desviación estándar en torno a él. Con todo esto se calcula la probabilidad de que cada punto pertenezca a cada cluster y se asigna al cluster más probable. A continuación, se recalculan los valores de media y desviación estándar de cada cluster; para ello se realiza una suma repesada, siendo el peso la probabilidad de que cada punto pertenezca a dicho cluster. Se repite el proceso, se recalcula la probabilidad de que cada punto pertenezca a cada cluster y se recalculan los valores de cada cluster hasta que se alcanza la convergencia.

La principal ventaja de este tipo de algoritmos es que se adapta a distintas formas de los cluster de manera más eficiente, principalmente a formas elípticas. Como se muestra en la figura 8 el algoritmo ajusta un conjunto de datos formado por un primer conjunto circular

(naranja) y otro conjunto elíptico (azul). Además, soporta que un mismo punto pertenezca a dos clusters con distintas probabilidades por tanto en caso de algún dato problemático es posible guardarlo como perteneciente a ambos clusters y realizar posteriormente un análisis más detallado y preciso.

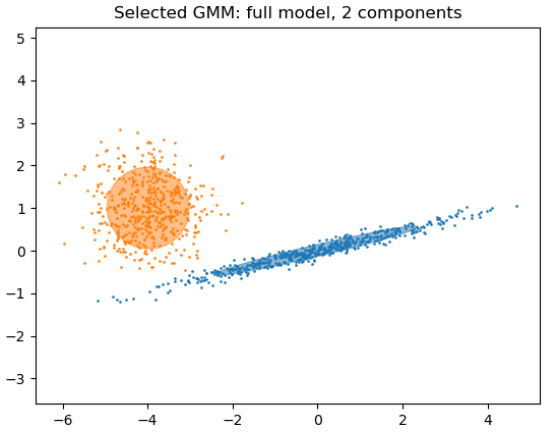


Figura 8: Ajuste a un conjunto de datos simulado formado por un círculo y un elipsoide utilizando el algoritmo EM-GMM. La zona sombreada representa la componente gaussiana de cada conjunto de datos, se obtiene a partir del algoritmo de ajuste como la matriz de covarianza. Fuente: [16].

Una de las desventajas de este algoritmo reside, una vez más, en el hecho de que es necesario conocer con antelación el número de clusters deseado. La complejidad computacional de este algoritmo depende tanto del número de datos n, el número de cluster deseado k y el número de dimensiones m, siendo este $O(n, k, m^3)$.

2.6 BIRCH

BIRCH son las siglas de Balanced Iterative Reducing and Clustering using Hierarchies (Reducción y agrupamiento iterativo equilibrado mediante jerarquías). Este algoritmo tiene como primer objetivo reducir el número de puntos iniciales, para ello genera nuevos datos que resumen este primer conjunto de datos. Para realizar esta reducción recurre a un CFT (Clustering Feature Tree o árbol de características de agrupamiento). Con este CFT se realiza el clustering uniendo los nodos cercanos. De esta manera, en la parte más ramificada de este árbol se tiene cada punto individual y estos primeros "clusters" se van uniendo por proximidad hasta llegar a un único "mega-cluster".

Este árbol tiene en sus nodos tres datos: N,LS y SS. N es el número de puntos de la muestra, LS es el vector suma de las posiciones de los puntos y SS el cuadrado de las dimensiones de los puntos muestra. Posteriormente, se agrupan estos nodos de manera jerárquica; entonces se tiene que los agrupamientos similares se encuentran en nodos próximos. En este momento, el algoritmo escoge el número de clusters más óptimo.

Las principales ventajas de este algoritmo se encuentran en su bajo coste computacional, ya que crece como $O(n)$. Esto ocurre ya que es un algoritmo local, es decir, no necesita escanear todo el conjunto de datos para tomar una decisión. Su otra gran ventaja es no necesitar el número de clusters sino que lo proporciona el algoritmo en base a un parámetro llamado *threshold*. Este parámetro es el radio máximo del subcluster obtenido al unir un

subcluster y la muestra más cercana. Si se obtiene un radio superior no se realizará la unión.

2.7 Comparación de los algoritmos

Como se ha visto con anterioridad cada algoritmo tiene una serie de ventajas y desventajas, siendo cada uno de ellos de más adecuada aplicación en función del problema a tratar y del conjunto de datos al que se aplique.

En la figura 9 se observa como varía el resultado de aplicar cada algoritmo al mismo conjunto de datos en función de las características de este. La primera fila se corresponde con un conjunto de datos formado por dos círculos concéntricos. Solamente los algoritmos DBSCAN y Agglomerative Hierarchical Clustering reconstruyen de manera correcta los conjuntos de datos iniciales. Esto se debe a que estos algoritmos no tienden a minimizar ningún tipo de distancia sino que agrupan los puntos cercanos en el mismo cluster. Algo similar ocurre en la segunda fila aunque en este caso solo lo reconstruye de manera correcta el algoritmo DBSCAN.

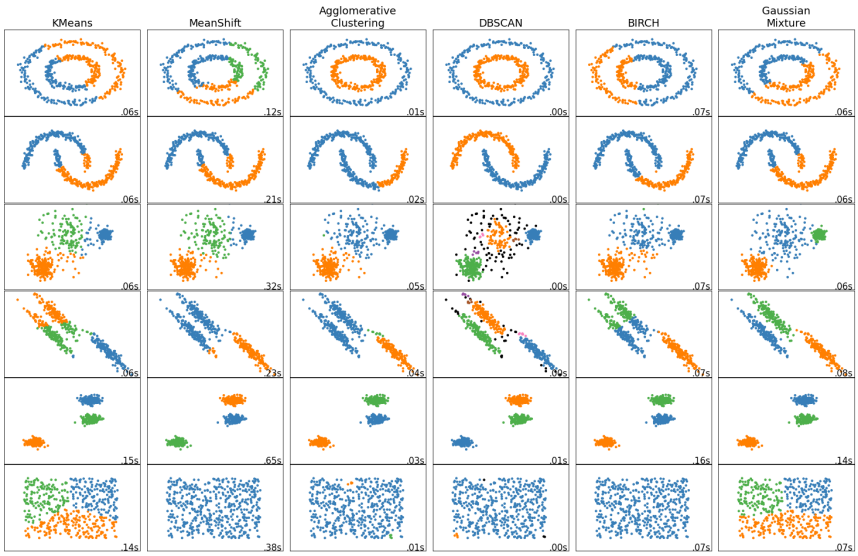


Figura 9: Resultado de aplicar los distintos algoritmos de clustering a distintos conjuntos de datos sencillos. Además se muestra el tiempo que se ha tardado en ejecutar dicho algoritmo.

En la tercera fila de figura 9 se muestran tres burbujas de puntos con distintas varianzas. Aquí algoritmos como EM-GMM realizan un muy buen clustering ya que se adaptan a las distintas formas y densidad de datos. En cambio, el algoritmo DBSCAN selecciona una gran cantidad de valores como ruido (marcado en negro) del conjunto de datos más disperso. De manera intermedia se sitúa el algoritmo K-Means que una vez introducido el número correcto de clusters genera un buen ajuste aunque pierde algún punto del conjunto de datos situado en la mitad.

En la cuarta fila se han generado conjuntos de datos distribuidos de manera anisótropa. El algoritmo que mejor ajusta a esta forma de los datos es EM-GMM ya que al estar implementado siguiendo un modelo de mezcla gaussiana le permite ajustarse a formas más elípticas. En la quinta fila se observa como todos los algoritmos reconstruyen de manera idónea las tres burbujas de datos.

La última fila se ha generado con puntos distribuidos de manera uniforme por lo tanto no se debería encontrar ningún cluster. Los algoritmos que necesitan el número de cluster como parámetro inicial (K-Means y EM-GMM) fracasan al intentar encontrar un número de cluster no correcto. El algoritmo BIRCH y el algoritmo MeanShift funcionan perfectamente en este caso. Por último, los algoritmos Agglomerative Hierarchical Clustering y DBSCAN funcionan de manera muy adecuada salvo por unas pequeñas fluctuaciones.

3 Validación de los algoritmos

Esta sección busca comprobar la eficacia de cada uno de los algoritmos mostrados en el capítulo anterior, usando un conjunto de datos de vértices y trazas generados con un modelo simplificado. El objetivo fundamental es comprobar que los algoritmos funcionan correctamente. Adicionalmente, estos datos han servido para optimizar los parámetros inicales de cada algoritmo y medir su desempeño final.

Con este objetivo, se han programado en lenguaje python, varios programas encargados de generar estos conjuntos de datos, de su análisis y también de evaluar los resultados obtenidos. Estos programas se encuentran en el repositorio creado a tal efecto (ver [17]).

3.1 Generación de vértices y trazas aleatorios

Para generar los vértices y las trazas de manera aleatoria se generan primero los vértices y, luego, a su alrededor las trazas pertenecientes a cada vértice. Los vértices se han generado siguiendo una distribución gaussiana en el eje z y otra en el eje t. En el eje z esta gaussiana toma como parámetros de media $\mu_z = 0$ cm y de desviación estándar $\sigma_z = 5$ cm; a su vez en el eje t estos parámetros son $\mu_t = 0$ ps y $\sigma_t = 200$ ps. Son seleccionados estos valores ya que son los valores típicos esperados en el HL-LHC. Con estos parámetros se generan 200 vértices aleatorios. Los resultados de ejecutar este código se muestran en la figura 10[3].

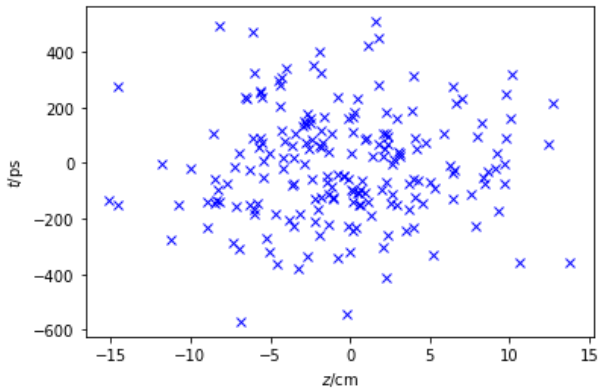


Figura 10: La posición de los vértices generados de manera aleatoria.

Posteriormente se generan alrededor de cada vértice las trazas necesarias. Para definir el número de trazas en torno a cada vértice se uso, al igual que en el caso anterior, una distribución gaussiana con un valor medio de 70 trazas en cada vértice y una desviación estándar de 10. Con esto se sabe cuántas trazas se sitúan en cada vértice. Para colocar en su posición cada traza se usan coordenadas polares centradas en la posición del vértice. Para definir la coordenada radial se usa una nueva gaussiana con parámetros de radio medio, $\mu_r = 0$ cm, y desviación estándar, $\sigma_r = 0.05$ cm. Para el ángulo se usa una distribución uniforme para generar un número aleatorio en el intervalo $[0, 2\pi)$ de manera que se obtiene el ángulo en radianes. Con todo esto ya se tiene la posición de una traza, se repite este proceso para todos las trazas de todos los vértices. Una vez se tienen todas las posiciones es posible representarlas en el plano obteniendo la figura 11.

[3]Destacar que si se repite este mismo código no se obtiene el mismo resultado ya que la función numpy.random [18] utilizada genera números aleatorios.

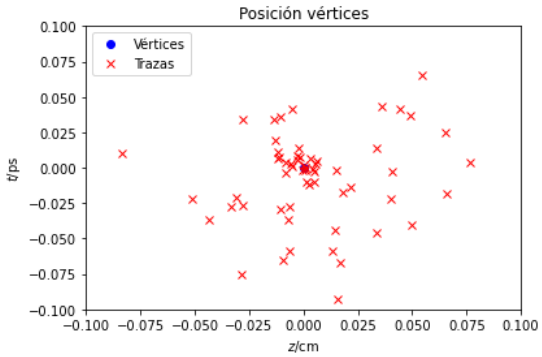


Figura 11: La posición de las trazas generados de manera aleatoria en relación a su vértice.

Por último, cada conjunto de dato se ha renormalizado, de manera que sean independientes de las unidades temporales y longitudinales utilizadas. Para ello se ha utilizado la media (μ) y la desviación estándar (σ) del conjunto de trazas cómo se muestra en la ecuación (4). Los algoritmos han sido ejecutados para estos nuevos conjuntos de datos renormalizados más realistas para su evaluación.

$$\begin{aligned} z' &= \frac{z - \mu_z}{\sigma_z} \\ t' &= \frac{t - \mu_t}{\sigma_t} \end{aligned} \tag{4}$$

3.2 Evaluación de resultados: la métrica

Para poder comparar los distintos algoritmos entre sí de una manera objetiva es necesario definir una métrica, es decir, definir un criterio de comparación.

Teniendo en cuenta el objetivo final de aplicación de estos algoritmos, que no es otro que el experimento CMS, es imprescindible que los algoritmos puedan reconstruir los sucesos con una alta precisión, que no cometan errores en la asignación de un número elevado de trazas y que reconstruyan una gran mayoría (siendo deseable que fueran todos) los vértices correctamente. Además es necesario que el algoritmo sea capaz de converger rápidamente ya que, en caso contrario, la alta luminosidad y el alto flujo de datos, el algoritmo sería impracticable.

Por todo ello, se ha decidido que para evaluar los resultados hay que tener en cuenta los siguientes parámetros:

- Fracción de trazas correctamente asignadas.
- Fracción de vértices correctamente reconstruidos.
- Número de clusters generados/Número de vértices.
- Tiempo de ejecución.
- Distancia media entre los centroide y los vértices simulados.
- La nota y la nota ajustada.

La fracción de trazas correctamente reconstruidas se refiere al número de trazas que el algoritmo ha asignado de manera correcta a su cluster correspondiente, dividido entre el número total de trazas. De manera simétrica se puede calcular la fracción de trazas mal asignadas como 1 menos el valor anterior.

La fracción de vértices correctamente reconstruido hace referencia al número de vértices en los que la totalidad de las trazas asignadas realmente pertenecen a ese vértice entre el número total de vértices. Al igual que el parámetro anterior, existe una relación directa entre la calidad del algoritmo y este número. El valor máximo de este parámetro es 1, significando esto que todos los vértices simulados han sido correctamente reconstruidos.

El número de clusters/número de vértices solo tiene sentido analizarlo en los algoritmos que lo seleccionan automáticamente (como MeanShift, DBSCAN, Aglomerative Hierarchical Clustering y BIRCH), ya que en el resto, el número de clusters será un valor introducido manualmente.

El tiempo de ejecución, que debido a la alta luminosidad de las condiciones reales de trabajo, se necesita que sea lo más bajo posible lo cual evitará que el algoritmo se sature. De manera obvia el tiempo de ejecución depende de muchos factores como el ordenador utilizado, la memoria disponible en el mismo, etc. Para obtener una estimación lo más objetiva posible todos los algoritmos se han ejecutado en el mismo ordenador en circunstancias lo más parecidas posibles, sin ninguna otra aplicación en segundo plano y con un caché limpio para evitar las posibles interferencias, aunque no se ha profundizado en el estudio de qué otras variables pueden interferir y de qué manera.

La distancia se corresponde con la distancia euclídea que existe entre el centroide del cluster y la posición real del vértice. Idealmente, esta distancia debería de ser 0. Además, esta distancia cumple un doble papel, a parte de ser un estimador de la calidad del ajuste sirve para realizar una asignación entre vértices y clusters. Ya que, al realizar el ajuste no existe una correspondencia entre las etiquetas de los vértices y las de los clusters. Por tanto, para realizar esta asignación entre vértices y clusters se sigue un criterio de proximidad. Es decir, se asigna a cada vértice su centroide más próximo de manera que así se obtenga esta correspondencia entre ambos conjuntos.

Por último la nota y la nota ajustada son dos funciones que se importaron del paquete sklearn [19] y que comparan dos listas de datos con etiquetas cambiadas y desordenadas. Para la nota se usó la función *rand_score* la cuál calcula el número de puntos similares en las dos listas ignorando las permutaciones. Esta implementación se basa en el rand index [20]. La nota es calculada de la siguiente manera:

$$RI = \frac{\text{Nº de pares iguales}}{\text{Nº total de pares}} \tag{5}$$

Siendo RI (Rand Index) la nota calculada. De esta manera se tiene que $RI \in [0, 1]$ siendo 0 indicativo de que no existe ninguna coincidencia entre ambas listas y 1 una coincidencia perfecta. Por ejemplo, tomando una lista como $\text{lista}_1 = [0, 0, 1, 1]$ y $\text{lista}_2 = [1, 1, 0, 0]$. Se tiene que la nota o el $RI = 1$ ya que ambas listas coinciden. En cambio al introducir una $\text{lista}_3 = [1, 1, 0, 2]$ este valor decae hasta $RI = 0.8\widehat{3}$. El número total de pares se calcula como $\binom{4}{2} = 6$, siendo 4 el tamaño de la lista. El número de pares iguales es 5 en este caso. Tomando la lista1 y la lista3 y nombrando a los elementos como [A,B,C,D] se tiene que el par AB pertenece al mismo cluster en sendas listas (en la lista1 ambos elementos pertenecen al cluster 0 y en la lista3 al cluster 1). También se tiene que los elementos AC,

AD,BC y BD pertenecen a clusters distintos en ambas listas. Con esto se obtienen los 5 pares iguales.

La nota ajustada utiliza los mismos principios que la nota, pero corrige por el factor de aleatoriedad de las permutaciones. Los detalles de la implementación se hallan en la ecuación (6). Esto se implementó usando la función *adjusted_rand_score*.

$$ARI = \frac{RI - \text{Expected\_}RI}{\text{máx}(RI) - \text{Expected\_}RI} \tag{6}$$

Siendo ARI el valor de la nota ajustada (Adjusted Rand Index). De esta manera se consigue que dos listas con elementos seleccionados de manera aleatoria tengan un valor negativo, castigándose así esta aleatoriedad manteniéndose un resultado cercano a 1 para las listas más coincidentes. Por tanto, se tiene que $ARI \in [-1, 1]$. La principal ventaja de la nota ajustada frente a la nota se encuentra en su mayor poder de discriminación al tener en cuenta y penalizar el factor aleatorio.

3.3 Resultados

Antes de proceder con la evaluación de resultados, los algoritmos han sido aplicados a una serie de conjuntos de datos con el objetivo de optimizar sus parámetros. Posteriormente se aplicaron los algoritmos a 100 conjuntos de datos generados de manera aleatoria (con los valores especificados en el apartado 3.1). Y, se evaluaron los resultados de los algoritmos usando las métricas ya explicadas. Por último, se calculó la media y la desviación estándar de cada uno de estos valores para así tener suficiente estadística y no obtener resultados sesgados. Los resultados obtenidos se muestran en la tabla 1.

3.3.1 K-Means

El algoritmo K-Means proporciona los resultados mostrados en la tabla 1 cuando se aplica a 100 conjuntos de datos generados aleatoriamente como se explica en apartado 3.1 con unos parámetros iniciales de 200 clusters. En este algoritmo no se estudia el número de clusters ya que este es siempre el mismo, porque lo toma como un parámetro fijo introducido manualmente como se discutió en el apartado 2.1.

Para ver que este ajuste es el mejor de todos se probó a variar el número de clusters manteniendo el conjunto de datos constante. Evaluando estos resultados se encuentra un máximo cuándo el número de clusters es igual al número de vértices simulados (200 en este caso) como se observa en la figura 12. Además, también se observa que el valor absoluto de la pendiente antes del máximo es mayor que después del máximo. Por tanto, en caso de no conocer con exactitud el número de vértices es mejor introducir un número de clusters sobreestimado que subestimado.

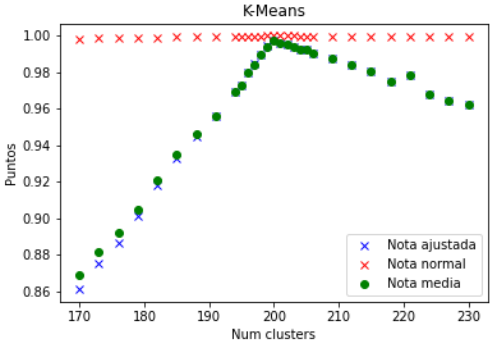


Figura 12: Evolución de la nota y la nota ajustada al variar el número de cluster que toma como parámetro el algoritmo K-Means al ser aplicado sobre el mismo conjunto de datos simulado aleatoriamente.

3.3.2 MeanShift

El algoritmo MeanShift proporciona los resultados mostrados en la tabla 1 al ser aplicado a 100 conjuntos de datos generados de manera aleatoria, tal y como se explicó en la sección apartado 3.1. Para obtener estos resultados los parámetros iniciales que se toman son: $n\_samples = 299$, $quantile = 0.01$ y $min\_bin\_freq = 1$; además se establece $bin\_seeding = True$ para optimizar el tiempo de ejecución. En la tabla 1 se observa que el número de clusters generado ya no es constante, como sí sucedía en el algoritmo K-Means. Esto se debe a que no es un parámetro del algoritmo si no que es calculado por el algoritmo. Además, este número de clusters es ligeramente inferior al número de vértices utilizados, por tanto $Nº\ Clusters/Nº\ Vértices < 1$.

Pese a no inicializar como parámetro el número de clusters este algoritmo si inicializa otros parámetros como el *quantile* o *n_samples* para calcular el *bandwidth*. El *quantile* define como se toma la distancia, por ejemplo si se establece igual a 0.5 se toma la mediana de todos los pares de distancia. En cambio *n_samples* define el número de muestras aleatorias a tomar para el calculo del *bandwidth*. En la figura 13 se observa que existe una discontinuidad entre el número de trazas correctas y de vértices correctos al estudiar su evolución con *n_samples*. Esta discontinuidad se produce en el mismo punto para distintos conjuntos de datos con esta estructura. Esto se debe a que si se aumenta en exceso este parámetro el algoritmo sufrirá overfitting, es decir, dejará de estudiar las características generales del conjunto de datos para el *bandwidth* y usará características concretas, lo que resulta en que el algoritmo selecciona un *bandwidth* demasiado pequeño. Además, como estas muestras se seleccionan de manera aleatoria, una vez se selecciona un número representativo de ellas es lógico que la discontinuidad se produzca en el mismo punto.

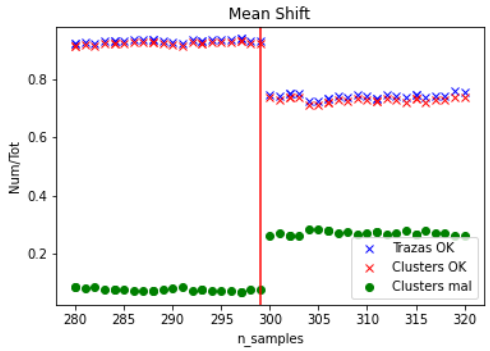


Figura 13: Evolución del número de trazas bien asignadas, el números de clusters bien reconstruidos y los mal reconstruidos respecto al total de ellos al variar el parámetro *n_samples* en el algoritmo MeanShift. La línea roja con ecuación $n\_samples = 299$ muestra el valor elegido de este parámetro como el mejor ajuste.

3.3.3 DBSCAN

El algoritmo DBSCAN se basa en dos parámetros para estimar los clusters: ϵ que estima un radio alrededor del cluster y *minPoints* que calcula si el punto analizado tiene los suficientes vecinos para formar un cluster a su alrededor. Con un valor demasiado pequeño de ϵ no se reconstruyen los vértices, ya que la dispersión de las trazas es mayor que esta distancia. En cambio, con un valor excesivamente elevado se llegan a unir en un mismo cluster las trazas de dos vértices distintos. Si el parámetro *minPoints* se establece demasiado alto, mayor que el número de trazas por vértice, no se formarán los clusters por insuficiencia de trazas. En la figura 14 se muestra como varía la estimación de trazas y clusters correctos frente al total en función de estos parámetros. Se observa que con un valor excesivamente alto de ϵ o de *minPoints* no se consigue una buena asignación.

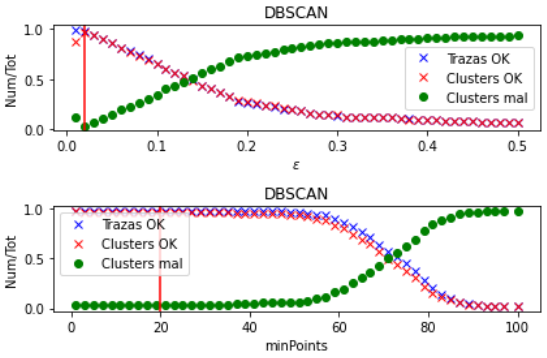


Figura 14: Número de trazas bien asignadas, número de clusters bien reconstruidos y los mal reconstruidos respecto al total de ellos al variar el parámetro ϵ y *minPoints* en el algoritmo DBSCAN. Las líneas rojas verticales (con ecuaciones $\epsilon = 0.02$ y $minPoints = 20$) muestran los valores elegidos de sendos parámetros como el mejor ajuste.

El algoritmo DBSCAN proporciona los resultados mostrados en la tabla 1 cuando es aplicado a 100 conjuntos de datos generados de manera aleatoria y los parámetros iniciales del algoritmo son $\epsilon = 0.02$, $min\_samples = 20$ y $leaf\_size = 12$.

3.3.4 Agglomerative Hierarchical Clustering

En este algoritmo es necesario definir una distancia a partir de la cuál no se unirán más clusters, este parámetro se conoce como *distance threshold*. Dadas las características de la relación entre vértices y trazas la mejor métrica para este caso es "Ward", la cuál computa la suma cuadrática de distancias entre todos los clusters y la minimiza. Para la *distance threshold* sucede lo mismo que en el algoritmo DBSCAN, pese a esto, el valor óptimo no coincide. En la figura 15 se muestra cómo varía el número de clusters frente a este parámetro. Se observa que para *distance threshold* demasiado elevadas el número de clusters decae, es decir, se han unido demasiadas ramas del diagrama, demasiados clusters.

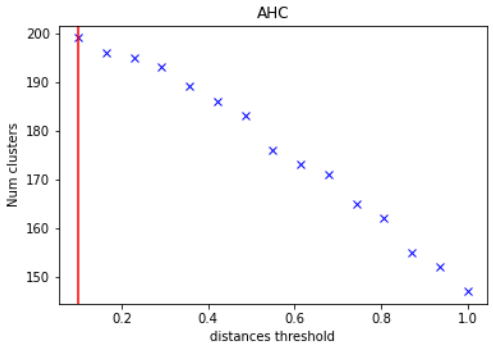


Figura 15: Evolución del número de clusters total al variar el parámetro *distance threshold* en el algoritmo Agglomerative Hierarchical Clustering. La línea roja vertical con ecuación *distance_threshold* = 0.1 muestra el valor seleccionado como mejor ajuste de este parámetro.

El algoritmo Agglomerative Hierarchical Clustering proporciona los resultados mostrados en la tabla 1 al ser ser aplicado con el parámetro de *distance threshold* = 0.1 y un *linkage* = *Ward* sobre 100 conjuntos de datos generados aleatoriamente. Además para que funcione de manera adecuada se ha de establecer *n_clusters* = *None* para que no se seleccione el número de clusters de manera manual.

3.3.5 EM-GMM

El algoritmo EM-GMM proporciona los resultados mostrados en la tabla 1 cuando se ejecuta este algoritmo sobre 100 conjuntos de datos creados según lo expuesto en 3.1 y tomando como parámetro del algoritmo *num_clusters* = 200, es decir igual al número de vértices simulados. En este algoritmo, al igual que en el K-Means, el parámetro principal que recibe es el número de clusters, por tanto este valor se mantiene constante. Proporcionando una gráfica similar a la mostrada en la figura 12 al estudiar como varía la nota y la nota ajustada en función del número de clusters.

3.3.6 BIRCH

Con el algoritmo BIRCH se tienen dos parámetros principales *threshold*, qué marca la distancia a partir de la cuál los clusters serán considerados independientes; y *branching*, el cuál considera el número máximo de subclusters en cada nodo.

Al igual que en los casos anteriores si se toma una distancia demasiado pequeña se tiene un gran número de clusters independientes y si se establece un valor demasiado elevado los

clusters obtenidos son combinaciones de los vértices reales. Por tanto, es necesario situar este parámetro en un valor intermedio. Como se muestra en la figura 16 existe un máximo para la nota y la nota ajustada para *threshold* ≈ 0.02. Por este motivo se toma este valor el apropiado para conseguir el funcionamiento óptimo del algoritmo.

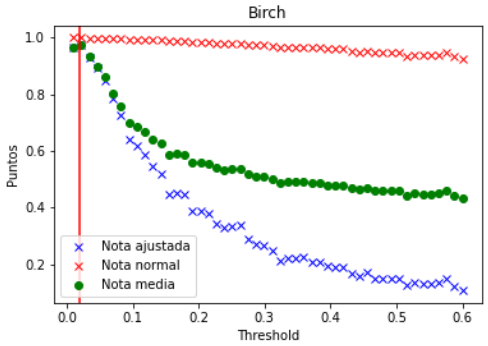


Figura 16: La nota y la nota ajustada frente al parámetro *threshold* en el algoritmo BIRCH. La línea vertical roja con ecuación *threshold* $= 0.02$ se corresponde con el valor elegido de dicho parámetro como el mejor ajuste.

El algoritmo BIRCH proporciona los resultados mostrados en la tabla 1 cuando se ejecuta con los parámetros de *threshold* $= 0.02$ y *branching* $= 70$ y, al igual que en el algoritmo Agglomerative Hierarchical Clustering es necesario establecer *n_clusters* $=$ *None*.

3.4 Comparación de los algoritmos

Variable	K-Means	MeanShift	DBSCAN	AHC	EM-GMM	BIRCH
Trazas OK/Totales	0.991 ± 0.004	0.92 ± 0.03	0.970 ± 0.012	0.996 ± 0.003	0.986 ± 0.006	0.975 ± 0.010
Vértices OK/Totales	0.966 ± 0.015	0.91 ± 0.03	0.997 ± 0.004	0.976 ± 0.012	0.982 ± 0.007	0.960 ± 0.014
Nº Clusters/Nº Vértices	–	0.91 ± 0.03	0.967 ± 0.012	0.999 ± 0.003	–	0.981 ± 0.012
Tiempo de ejecución/s	3.5 ± 0.5	2.0 ± 0.7	0.7 ± 0.2	7.4 ± 0.7	4.1 ± 0.7	0.7 ± 0.2
Distancia	0.00086 ± 0.00011	0.0023 ± 0.0007	0.00108 ± 0.00017	0.00071 ± 0.00007	0.00103 ± 0.000015	0.00191 ± 0.00018
Nota	0.99988 ± 0.00006	0.9991 ± 0.0003	0.99966 ± 0.00013	0.999945 ± 0.00004	0.99983 ± 0.00007	0.99972 ± 0.00012
Nota Ajustada	0.987 ± 0.006	0.91 ± 0.03	0.967 ± 0.013	0.995 ± 0.004	0.983 ± 0.007	0.973 ± 0.011

Tabla 1: Evaluación de los resultados proporcionados por los distintos algoritmos estudiados al ser aplicados a 100 conjuntos de datos generados de manera aleatoria.

Con todo esto se observa como el algoritmo Agglomerative Hierarchical Clustering es el que mejor predice el número de clusters. Pese a esto su tiempo de ejecución es el mayor de todos, siendo mucho mayor que el de otros algoritmos como BIRCH o DBSCAN que también predicen de manera adecuada el número de clusters. De estos tres algoritmos Agglomerative Hierarchical Clustering es el que mejores resultados en Trazas OK, vértices OK , notas y distancia proporciona. Teniendo BIRCH y DBSCAN un desempeño similar entre ellos. Con estos dos algoritmo se obtienen resultados de similar calidad pero con un tiempo de ejecución medio aproximadamente 10 veces inferior.

Los algoritmos K-Means y EM-GMM son los que no predicen el número de clusters automáticamente, además se fundamentan en principios similares. Comparando los resultados entre ambos algoritmos se encuentran resultados similares siendo el K-Means mejor en asignación de trazas, tiempo, distancia, nota y nota ajustada levemente mejor y EM-GMM en reconstrucción correcta de vértices.

El algoritmo que proporciona los peores resultados para este tipo de conjunto de datos es MeanShift que reconstruye un menor número de vértices correctamente y además no predice de manera adecuada el número de clusters.

Por último, la distancia media entre los centroides y los centros de los vértices es similar para todos los algoritmos, no es un elemento diferencial en ninguno de ellos. Siendo el algoritmo DBSCAN el mejor con un valor de $distancia_{\text{DBSCAN}} = 0.00108 \pm 0.00017$ y el peor BIRCH con $distancia_{\text{BIRCH}} = 0.00191 \pm 0.00018$.

4 Aplicación a datos CMS

En esta sección, los algoritmos van a ser ejecutados sobre un dataset de vértices y trazas correspondiente con una simulación de un proceso de producción de partes top-antitop en el experimento CMS. Esta simulación ha sido generada a nivel del proceso cuántico por el programa Madgraph [21], posteriormente la hadronización de los quarks resultantes del proceso de producción ha sido realizada utilizando Pythia [22], la simulación de la propagación de las trazas por los detectores usando el programa GEANT4 [23] y finalmente para la reconstrucción de trazas y vértices se han utilizado los algoritmos estándar de la colaboración CMS.

Estos conjuntos de datos han sido renormalizados al igual que los generados de manera aleatoria según la ecuación (4). Además, se ha añadido a la evaluación los resultados de clustering proporcionados por los algoritmos estándar de CMS. De esta manera se tienen unos valores de comparación. Para la evaluación de los resultados en esta parte se añadió un nuevo parámetro que es el número de vértices que faltan. Este parámetro se refiere a la cantidad de vértices simulados que produjeron trazas las cuáles no depositaron ninguna señal en los detectores por lo que no se tiene ninguna traza en ellos. Por tanto, el número de clusters ideal es siempre levemente inferior al número de vértices simulados ya que existen ciertos vértices que no tienen asociada ninguna traza.

4.1 Diferencias entre datos aleatorios y datos simulados

Los datos simulados por CMS tienen ciertas peculiaridades respecto de los datos aleatorios. En la práctica el MTD no va a ser capaz de asignar una medida temporal a cada partícula. Esto es debido a problemas de eficiencia y falta de hermeticidad en el MTD. Por tanto, en los datos simulados existen algunas trazas sin coordenada temporal, para subsanar esta falta de información se le asigna un valor de $t = 0$ a las trazas no detectadas. Esto genera que al analizar en conjunto las trazas haya una mayor cantidad de trazas en $t = 0$ de las esperadas y de las generadas en los conjuntos de datos aleatorios.

Por otra parte, como se discutió en la apartado 1.6, se asume que todos los hadrones detectados en el MTD son piones y se reconstruye su tiempo en POCA a partir de la masa de esta partícula. Esto genera que, aunque la mayoría de las veces esta asignación sea correcta, exista un número notable de casos en los que esta reconstrucción temporal proporciona valores muy dispares de los correctos. Esto genera que al propio error del MTD haya que superponerle un error en la reconstrucción para subsanar esta disparidad.

Con todo esto se tiene que los datos generados de manera aleatoria, tienen una distribución más homogénea y con una menor incertidumbre que los datos simulados.

4.2 Evaluación Algoritmo CMS

Con la simulación de datos medidos en CMS se aplica el algoritmo "Deterministic Annealing" para obtener la asignación de vértices y trazas oficial que produciría CMS. Por último se evalúa esta asignación con las métricas elegidas para el resto de algoritmos, de esta manera se tiene una comparativa cuantitativa de la calidad. Se han obtenido los resultados mostrados en la tabla 2.

Representando la posición de los centroides y de los vértices obtenidos a partir de la reconstrucción oficial de CMS se obtiene la figura 17.

Variable	Valor
Trazas OK/Totales	0.49 ± 0.03
Vértices OK/Totales	0.034 ± 0.010
Nº Clusters/Nº Vértices	1 ± 0
Vértices Faltan	13 ± 3
Tiempo de ejecución/s	≈ 120
Distancia	0.091 ± 0.012
Nota	0.9888 ± 0.0013
Nota Ajustada	0.63 ± 0.03

Tabla 2: Resultados de evaluar el clustering que se realiza en CMS siguiendo el proceso de "Deterministic Annealing" para 28 conjuntos de datos generados a partir de una simulación precisa de las señales de los detectores en este experimento. El tiempo de ejecución no se ha medido en las mismas condiciones que el resto de algoritmos, pese a esto se conoce una aproximación de su valor.

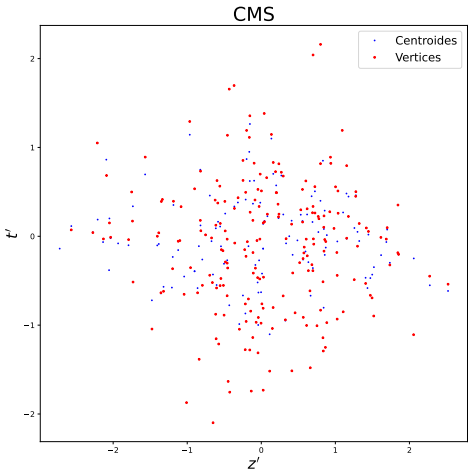


Figura 17: Distribución de vértices en cruces rojas (valores reales) y centroides en puntos azules (obtenidos en la reconstrucción oficial de CMS) en el espacio z'-t'.

4.3 Evaluación algoritmos seleccionados

Aplicando los algoritmos estudiados a 28 conjuntos de datos simulados a partir de un proceso de producción top-antitop y evaluando su desempeño con la métrica seleccionada se obtiene la tabla 3.

Variable	K-Means	MeanShift	DBSCAN	AHC	EM-GMM	BIRCH
Trazas OK/Totales	0.254 ± 0.018	0.23 ± 0.02	0.20 ± 0.03	0.251 ± 0.018	0.26 ± 0.02	0.23 ± 0.03
Vértices OK/Totales	0.022 ± 0.009	0.018 ± 0.014	0.008 ± 0.005	0.021 ± 0.009	0.015 ± 0.008	0.018 ± 0.010
Nº Clusters/Nº Vértices	1.05 ± 0.07	1.1 ± 0.2	1.00 ± 0.07	1.01 ± 0.06	1.05 ± 0.07	1.08 ± 0.08
Vértices Faltan	13 ± 3	13 ± 3	13 ± 3	13 ± 3	13 ± 3	13 ± 3
Tiempo de ejecución/s	1.8 ± 0.4	1.9 ± 0.7	0.16 ± 0.06	0.43 ± 0.18	5.9 ± 1.1	0.22 ± 0.09
Distancia	1.00 ± 0.10	1.13 ± 0.15	0.10 ± 0.02	0.99 ± 0.11	1.01 ± 0.10	1.09 ± 0.11
Nota	0.9815 ± 0.0016	0.965 ± 0.005	0.85 ± 0.06	0.9811 ± 0.0013	0.978 ± 0.003	0.9696 ± 0.0015
Nota Ajustada	0.311 ± 0.017	0.22 ± 0.02	0.05 ± 0.02	0.304 ± 0.016	0.27 ± 0.04	0.24 ± 0.02

Tabla 3: Evaluación de los resultados proporcionados por los distintos algoritmos estudiados al ser aplicados 28 conjuntos de trazas generados a partir de una simulación precisa de las señales en los detectores del experimento CMS.

4.3.1 K-Means

Para aplicar el algoritmo K-Means es necesario reajustar los parámetros iniciales. Lo ideal sería tomar un número de clusters igual al número de vértices. Pero, cómo este parámetro no se conoce de manera exacta se toma un número de clusters igual a 200 como primera aproximación. Aplicando este algoritmo a los conjuntos de datos simulados de manera fiel al experimento CMS se obtienen los resultados de tabla 3. Se observa que el número de clusters seleccionado es levemente superior al de vértices simulados, ya que $Nº\ Clusters/Nº Vértices > 1$. Por tanto, la estimación realizada de introducir 200 clusters como parámetro inicial es correcta ya que sobreestima levemente el número de vértices lo qué proporciona mejores resultados como se ha observado en la figura 12.

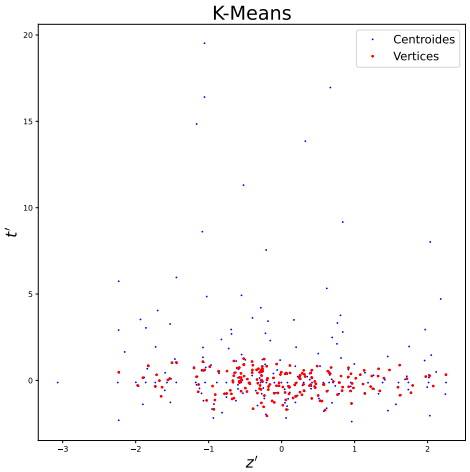


Figura 18: Distribución de vértices en cruces rojas (valores reales) y centroides en puntos azules (valores obtenidos por el algoritmo de clustering K-Means) en el espacio z'-t'.

Para estudiar la dispersión entre los centroides y vértices se representaron ambos en la figura 18. Se observa que el algoritmo sitúa ciertos centroides muy separados de sus correspondientes vértices. Existe una mayor dispersión temporal en los centroides que en los vértices, esto se debe a que hay trazas con una alta dispersión temporal. Entonces el algoritmo sitúa ciertos centroides con un alto valor temporal. Además, comparando con la asignación de centroides que realiza el algoritmo "Deterministic Annealing" (figura 17) se

encuentra que el algoritmo K-Means sitúa los centroides con una dispersión mucho mayor, lo que repercute en un aumento de la distancia media entre vértices y centroides en este algoritmo.

4.3.2 MeanShift

Para aplicar el algoritmo MeanShift de manera que proporcione los mejores resultados posibles es necesario realizar un pequeño reajuste de los parámetros iniciales respecto de los aplicados. Es necesario tomar $n\_samples = 328$, $quantile = 0.0118$ y $min\_bin\_freq = 1$. De esta manera se maximiza el número de trazas OK y se consigue un número de clusters muy cercano al número de vértices reales. Esto se observa en figura 19 donde el máximo de las trazas bien asignadas se encuentra para $quantile = 0.0118$, aunque parezca que con valores inferiores se puedan conseguir resultados mejores dado que este parámetro se tiene que encontrar en el intervalo [0,1] con valores inferiores se obtiene un $bandwith = 0$, no válido para el algoritmo (este tiene que pertenecer al intervalo $(0, \infty)$). Con estos parámetros iniciales se obtienen los resultados mostrados en la tabla 3.

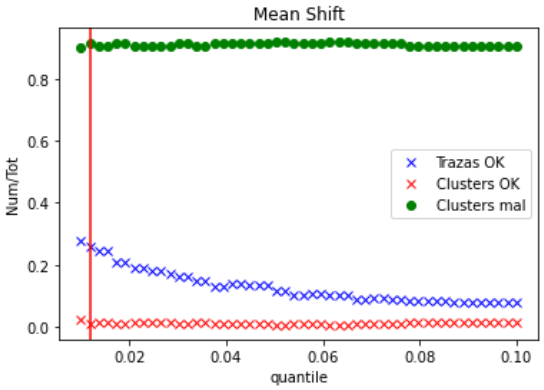


Figura 19: Evolución del número de clusters correcta e incorrectamente reconstruidos y trazas bien asignadas respecto del total al variar el parámetro inicial *Quantile* del algoritmo MeanShift. La línea vertical con ecuación $quantile = 0.0118$ representa el valor seleccionado como el mejor ajuste.

En la figura 20 se observa como es la distribución de centroides dados por este algoritmo frente a la de los centroides reales. Se observa que la distribución de centroides es similar la distribución del algoritmo K-Means (figura 18). Aún así, se observa como su distancia aumenta levemente, esto se debe a que el valor media de distancia entre un centroide y su vértice más cercano es mayor en este caso.

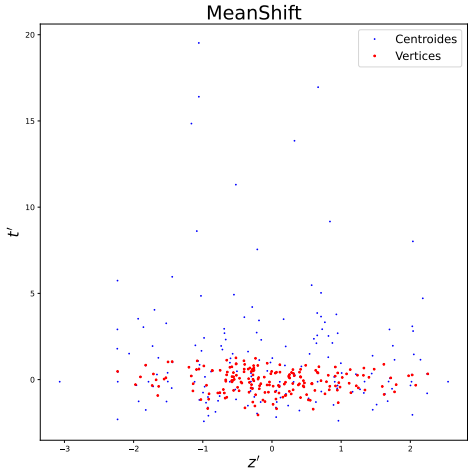


Figura 20: Distribución de vértices en cruces rojas (valores reales) y centroides en puntos azules (valores obtenidos por el algoritmo de clustering MeanShift) en el espacio z'-t'.

4.3.3 DBSCAN

El algoritmo DBSCAN proporciona los resultados mostrados en la tabla 3. Para obtener estos valores se ha aplicado el algoritmo DBSCAN con unos parámetros iniciales $\epsilon = 0.04$ y $min\_samples = 2$.

En este algoritmo se observa una gran disminución de la distancia respecto de los otros algoritmos, teniendo un valor $distancia_{\text{DBSCAN}} = 0.10 \pm 0.02$. Siendo esta distancia similar a la asignada por la reconstrucción oficial de CMS $distancia_{\text{CMS}} = 0.091 \pm 0.012$. Esto se traduce que en la figura 21 donde se compara la distribución de vértices y centroides esta se encuentra con una menor dispersión entre los centroides.

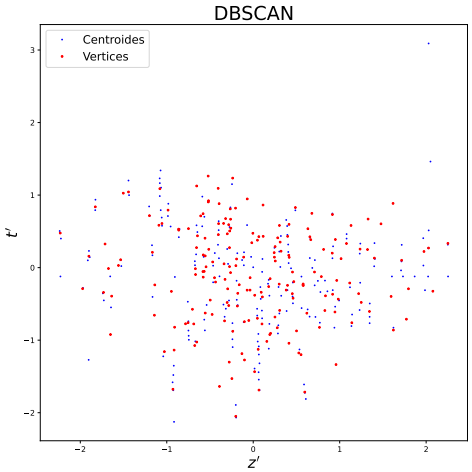


Figura 21: Distribución de vértices en cruces rojas (valores reales) y centroides en puntos azules (valores obtenidos por el algoritmo de clustering DBSCAN) en el espacio z'-t'.

4.3.4 Agglomerative Hierarchical Clustering

El algoritmo Agglomerative Hierarchical Clustering toma el parámetro *distance threshold* = 0.5. Aplicando dicho algoritmo a los conjuntos de datos simulados fielmente al detector CMS se obtienen los resultados representados en la tabla 3. Se observa que al igual que el algoritmo K-Means sitúa un cuarto de las trazas en el cluster correcto ($Trazas\ OK/Totales_{\mathrm{AHC}} = 0.251 \pm 0.018$). Con la ventaja de que este algoritmo no necesita tomar como parámetro el número de vértices y, además, es más rápido.

4.3.5 EM-GMM

El algoritmo Expectation Maximization-Gaussian Mixture Models toma como parámetro el número de clusters, al igual que el algoritmo K-Means. Por tanto, se recurre al mismo método para establecer este parámetro, se sobreestima levemente el número de vértices tomando como valor 200. Con esto se obtiene los resultados representados en la tabla 3. Se observa que el algoritmo proporciona unos resultados similares al algoritmo K-Means en trazas bien asignadas aunque su rendimiento es notablemente inferior en los vértices reconstruidos de manera adecuada.

4.3.6 BIRCH

El último de los algoritmos estudiados toma dos parámetros iniciales *threshold* y *branching*. Variando el valor de *threshold* se obtienen distintos números de clusters como se muestra en la figura 22. Se observa como al aumentar esta distancia, decae el número de clusters. Encontrándose el valor óptimo en un valor de *threshold* = 0.12. De manera similar, se estudia el comportamiento de *branching*, obteniéndose un valor óptimo en *branching* = 80.

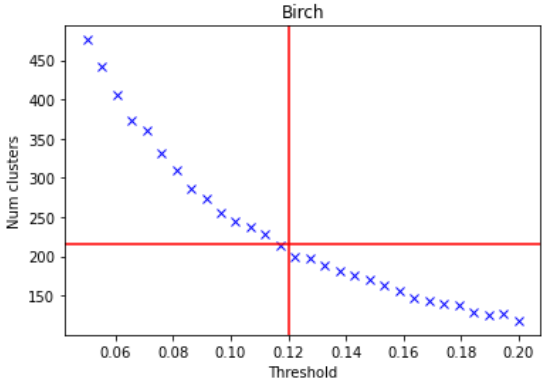


Figura 22: Variación del número de clusters proporcionado por el algoritmo BIRCH al variar el parámetro *threshold*. La línea roja horizontal con ecuación *num_clusters* = 215, marca el número de clusters ideal que se corresponde con el número de vértices y la línea vertical con ecuación *threshold* = 0.12 el valor seleccionado como mejor aproximación de este parámetro.

Aplicando el algoritmo BIRCH a 28 conjuntos de datos similares a las mediciones que se realizarán en el HL-LHC con los parámetros iniciales óptimos se obtienen los resultados representados en tabla 3. Se observa que este algoritmo es el que más vértices reconstruye de manera adecuada. Pese a esto no es el algoritmo que más trazas asigna de manera

correcta. Además, este algoritmo también sitúa los centroides muy separados de los vértices, $distancia_{\text{BIRCH}} = 1.09 \pm 0.11$.

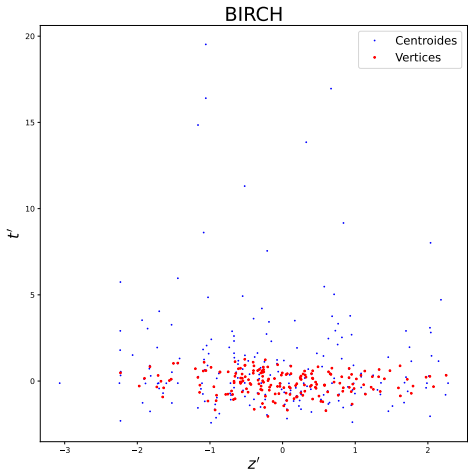


Figura 23: Distribución de vértices en cruces rojas y los centroides proporcionados por el algoritmo BIRCH en puntos azules en el espacio z-t.

Observando la dispersión de los centroides frente a los vértices en la figura 23 se observa que sigue una distribución similar a la proporcionada por el algoritmo K-Means (figura 18) o al algoritmo MeanShift (figura 20). Lo cuál tiene sentido ya que el valor de distancia proporcionado por estos algoritmos es similar al obtenido en BIRCH.

4.4 Comparación de los algoritmos

Comparando los algoritmos entre sí se observa que los algoritmos que más trazas asignan de manera correcta son el K-Means, el Agglomerative Hierarchical Clustering y el Expectation Maximization-Gaussian Mixture Model, acertando los tres algoritmos aproximadamente el 25 % de las trazas. Entre estos algoritmos el EM-GMM acierta levemente más vértices de manera correcta. Pese a esto, este algoritmo tiene una mayor distancia media, es decir sitúa los centroides más separados de los vértices. Pese a esto no existe una gran discrepancia entre los algoritmos en este parámetro ya que el algoritmo que menos trazas asigna de manera correcta es el DBSCAN con un valor de $Trazas\ OK/Totales_{\text{DBSCAN}} = 0.20 \pm 0.03$. Los algoritmos estudiados aciertan un número de trazas muy inferior al algoritmo "Deterministic Annealing", $Trazas\ OK/Totales_{\text{CMS}} = 0.49 \pm 0.03$.

El algoritmo que más vértices reconstruye de manera correcta son K-Means y Agglomerative Hierarchical Clustering con valores de $Vértices\ OK/Totales_{\text{K-Means}} = 0.022 \pm 0.009$ y de $Vértices\ OK/Totales_{\text{AHC}} = 0.021 \pm 0.009$, respectivamente. Seguidos por MeanShift ($Vértices\ OK/Totales_{\text{MeanShift}} = 0.018 \pm 0.014$) y BIRCH (Vértices OK/Totales$_{BIRCH} = 0.018 \pm 0.010$). El algoritmos K-Means reconstruye de manera adecuada un número levemente inferior de vértices de manera adecuada que el algoritmo de CMS, reconstruyendo este algoritmo $Vértices\ OK/Totales_{\text{CMS}} = 0.034 \pm 0.010$.

Todos los algoritmos reconstruyen un número total de vértices similar al de clusters, siendo el algoritmo MeanShift el que tiene una mayor discrepancia, construyendo aproxima-

damente 1.1 clusters por cada vértice simulado. Siendo el algoritmo DBSCAN el algoritmo más preciso al estimar el número de vértices. El algoritmo DBSCAN también sobresale a la hora de estimar la posición de los centroides, siendo el algoritmo con menor distancia entre centroides y vértices ($distancia_{\text{DBSCAN}} = 0.10 \pm 0.02$). Esto se observa de manera cualitativa al comparar la figura 21 con las figuras 17, 18, 20 y 23 donde se representan las posiciones de los vértices y centroides para cada uno de los algoritmos. Además, el valor de la distancia que proporciona el algoritmo DBSCAN es similar, aún siendo superior, al valor que proporciona la reconstrucción oficial de CMS, siendo esta $distancia_{\text{CMS}} = 0.091 \pm 0.012$.

Comparando los algoritmos estudiados con los resultados proporcionados por el algoritmo "Deterministic Annealing" se observa que ninguno de los algoritmos implementados se acerca ni al número de vértices reconstruidos de manera adecuada ni al número de trazas asignadas adecuadamente. El algoritmo K-Means es el algoritmo estudiado que más se acerca al implementado por CMS en cuánto a vértices más reconstruidos y EM-GMM es el que más sobresale en en trazas bien reconstruidas. El algoritmo DBSCAN el que peor desempeño tiene en ambas métricas.

El número de clusters seleccionado por el algoritmo implementado en CMS es muy preciso. Los algoritmos estudiados seleccionan en todos los casos valores muy cercanos pero sobreestiman este número.

La distancia a la que "Deterministic Annealing" sitúa los centroides de los vértices reales es aproximadamente de veces inferior que los algoritmos estudiados, salvando el DBSCAN que tiene un rendimiento similar a CMS en esta estadística. También es destacable la reducción del tiempo de ejecución de los algoritmos estudiados frente al "Deterministic Annealing" ya que este tarda $t_{\text{CMS}} \approx 120$s mientras que el algoritmo más lento de los estudiados es Expectation Maximization-Gaussian Mixture Models con $t_{\text{EM-GMM}} = 5.9 \pm 1.1$s.

Por último, la nota y, sobretodo, la nota ajustada es bastante superior a la proprorcionada por los algoritmos implementados, siendo K-Means el más cercano en nota ajustada (con un valor aproximado de la mitad de la nota ajustada por el algoritmo "Deterministic Annealing") y DBSCAN el más lejano.

5 Influencia del momento transverso en el clustering

Los resultados obtenidos no mejoran el algoritmo "Deterministic Annealing". Conociendo que este algoritmo tiene en cuenta el momento transverso de las trazas se decide añadir la información de esta variable en los algoritmos estudiados. Como con la implementación utilizada no se puede establecer un peso a cada traza en función del momento transverso, se establece un filtro a las trazas que usan los algoritmos de clustering en función del momento transverso de las mismas. Además, se ha decidido tomar este filtro ya que el problema de la incorrecta asignación de masa desaparece a alto momento transverso. Por tanto, con este filtro se contrarresta este problema en la reconstrucción temporal de las trazas.

También hay que tener en cuenta que si el filtro establecido es demasiado estricto, si el p_t establecido como valor mínimo es demasiado alto, el número de trazas resultantes será insuficiente para aplicar los algoritmos de clustering. Por esto mismo se deciden aplicar 3 condiciones distintas no demasiados restrictivas: $|p_t| \geq 0.5$ GeVGeV, $|p_t| \geq 1.0$GeV y $|p_t| \geq 1.5$ GeV.

5.1 K-Means

En la tabla 4 se muestran los resultados de aplicar los filtros en el momento de las trazas y el algoritmo K-Means. Al comparar con los resultados mostrados en la tabla 3 se observa una clara mejoría respecto de los resultados de este mismo algoritmo sin el filtro de las trazas. Siendo, incluso, mejores que los resultados del algoritmo "Deterministic Annealing" (tabla 2) en la reconstrucción de vértices. Aunque estos resultados son claramente inferiores en la asignación adecuada de trazas.

Variable	$\|p_t\| \geq 0.5$ GeV	$\|p_t\| \geq 1.0$ GeV	$\|p_t\| \geq 1.5$ GeV
Trazas OK/Trazas totales	0.29 ± 0.02	0.30 ± 0.03	0.32 ± 0.04
Vértices OK/Vértices totales	0.038 ± 0.015	0.04 ± 0.02	0.044 ± 0.015
Número de Clusters/Número de vértices	1.04 ± 0.07	1.04 ± 0.07	1.04 ± 0.07
Tiempo de ejecución/s	1.6 ± 0.4	1.4 ± 0.4	1.4 ± 0.5
Distancia	0.81 ± 0.07	0.22 ± 0.03	0.141 ± 0.012
Nota	0.9820 ± 0.0013	0.9793 ± 0.0019	0.977 ± 0.003
Nota Ajustada	0.318 ± 0.016	0.304 ± 0.013	0.26 ± 0.03
Trazas totales	2290 ± 190	850 ± 90	380 ± 50

Tabla 4: Resultados obtenidos por el algoritmo K-Means al filtrar las trazas en función de su momento transverso $|p_t|$.

5.2 MeanShift

Estudiando los resultados obtenidos al aplicar el algoritmo MeanShift se obtiene la tabla 5. En dicha tabla se observa una clara mejoría en los resultados respecto del mismo algoritmo sin el filtrado previo de las trazas (tabla 3). Pese a esta mejoría los resultados son peores en asignación de trazas y similares en reconstucción de vértices (tomando el mejor resultado del algoritmo MeanShift) que los proporcionados por el algoritmo aplicado en CMS sin el filtrado de trazas.

Variable	$\|p_t\| \geq 0.5$ GeV	$\|p_t\| \geq 1.0$ GeV	$\|p_t\| \geq 1.5$ GeV
Trazas OK/Trazas totales	0.272 ± 0.017	0.29 ± 0.04	0.33 ± 0.05
Vértices OK/Vértices totales	0.031 ± 0.011	0.039 ± 0.015	0.043 ± 0.014
Número de Clusters/Número de vértices	0.97 ± 0.12	0.76 ± 0.09	0.56 ± 0.05
Tiempo de ejecución/s	2.5 ± 1.8	1.3 ± 0.5	0.9 ± 0.3
Distancia	0.95 ± 0.10	0.26 ± 0.05	0.16 ± 0.02
Nota	0.968 ± 0.004	0.969 ± 0.004	0.967 ± 0.004
Nota Ajustada	0.25 ± 0.02	0.29 ± 0.03	0.31 ± 0.03
Trazas totales	2290 ± 190	850 ± 90	380 ± 50

Tabla 5: Resultados obtenidos por el algoritmo MeanShift al filtrar las trazas en función de su momento transverso $|p_t|$.

5.3 DBSCAN

En la tabla 6 se muestran la evaluación de los resultados proporcionados por el algoritmo DBSCAN. En este algoritmo, al contrario que en el resto, los resultados no mejoran mientras se hace más estricto el filtro. Sino que se alcanza un máximo para $|p_t| \geq 0.5$GeV y la calidad de los resultados decae llegando a ser similar para un $|p_t| \geq 1.5$GeV que para cuando no se aplica ningún filtro.

Variable	$\|p_t\| \geq 0.5$ GeV	$\|p_t\| \geq 1.0$ GeV	$\|p_t\| \geq 1.5$ GeV
Trazas OK/Trazas totales	0.24 ± 0.03	0.21 ± 0.03	0.19 ± 0.04
Vértices OK/Vértices totales	0.019 ± 0.011	0.013 ± 0.008	0.008 ± 0.004
Número de Clusters/Número de vértices	0.83 ± 0.05	0.55 ± 0.05	0.28 ± 0.04
Tiempo de ejecución/s	0.11 ± 0.04	0.032 ± 0.014	0.0146 ± 0.0018
Distancia	0.091 ± 0.016	0.110 ± 0.008	0.114 ± 0.012
Nota	0.90 ± 0.03	0.896 ± 0.013	0.84 ± 0.02
Nota Ajustada	0.09 ± 0.03	0.07 ± 0.02	0.048 ± 0.015
Trazas totales	2290 ± 190	850 ± 90	380 ± 50

Tabla 6: Resultados obtenidos por el algoritmo DBSCAN al filtrar las trazas en función de su momento transverso $|p_t|$.

5.4 Agglomerative Hierarchical Clustering

El algoritmos Agglomerative Hierarchical Clustering proporciona los resultados mostrados en la tabla 7. Al igual que el resto de algoritmos (exceptuando DBSCAN) su desempeño mejora al aplicar los distintos filtros. Pese a esta mejoría, los resultados obtenidos no mejoran la asignacióin de trazas obtenida por el algoritmo "Deterministic Annealing" cuando se ejecuta sin aplicar ningún filtro a las trazas (tabla 2). Sí mejoran levemente la correcta reconstrucción de vértices de este mismo algoritmo.

También es importante destacar que el número de clusters reconstruidos por este algoritmo decae drásticamente cuánto más estricto es el filtro aplicado. Esto se debe a la reducción del número de trazas utilizadas.

Variable	$\|p_t\| \geq 0.5$ GeV	$\|p_t\| \geq 1.0$ GeV	$\|p_t\| \geq 1.5$ GeV
Trazas OK/Trazas totales	0.29 ± 0.02	0.28 ± 0.03	0.30 ± 0.06
Vértices OK/Vértices totales	0.034 ± 0.014	0.035 ± 0.016	0.039 ± 0.009
Número de Clusters/Número de vértices	0.82 ± 0.05	0.53 ± 0.04	0.27 ± 0.03
Tiempo de ejecución/s	0.21 ± 0.09	0.04 ± 0.02	0.0083 ± 0.0018
Distancia	0.89 ± 0.08	0.29 ± 0.06	0.18 ± 0.03
Nota	0.9803 ± 0.0016	0.974 ± 0.002	0.967 ± 0.003
Nota Ajustada	0.32 ± 0.02	0.32 ± 0.02	0.34 ± 0.03
Trazas totales	2290 ± 190	850 ± 90	380 ± 50

Tabla 7: Resultados obtenidos por el algoritmo Agglomerative Hierarchical Clustering al filtrar las trazas en función de su momento transverso $|p_t|$.

5.5 EM-GMM

El algoritmo Expectation Maximization-Gaussian Mixture Models proporciona los resultados de la tabla 8 cuando los datos iniciales son filtrados en función del momento transverso de las trazas. Se observa que estos resultados mejoran el número de vértices reconstruidos de manera adecuada respecto al mismo algoritmo sin filtrado (tabla 3). Sin embargo, el número relativo de trazas bien reconstruidas se mantiene aproximadamente constante.

Variable	$\|p_t\| \geq 0.5$ GeV	$\|p_t\| \geq 1.0$ GeV	$\|p_t\| \geq 1.5$ GeV
Trazas OK/Totales	0.31 ± 0.02	0.30 ± 0.04	0.33 ± 0.04
Vértices OK/Totales	0.032 ± 0.013	0.041 ± 0.018	0.043 ± 0.015
Nº Clusters/Nº Vértices	1.04 ± 0.07	1.04 ± 0.07	1.04 ± 0.07
Tiempo de ejecución/s	4.4 ± 1.1	1.7 ± 0.7	0.4 ± 0.3
Distancia	0.81 ± 0.07	0.22 ± 0.03	0.141 ± 0.012
Nota	0.977 ± 0.003	0.970 ± 0.011	0.976 ± 0.003
Nota Ajustada	0.27 ± 0.03	0.24 ± 0.06	0.26 ± 0.03
Trazas totales	2290 ± 190	850 ± 90	380 ± 50

Tabla 8: Resultados obtenidos por el algoritmo EM-GMM al filtrar las trazas en función de su momento transverso $|p_t|$.

5.6 BIRCH

El algoritmo BIRCH proporciona los resultados mostrados en la tabla 9 cuando se aplican los filtros sobre el momento transverso de las trazas. Se observa que el algoritmo reconstruye menos clusters cuánto más estricto es este filtro pero los reconstruye de manera más correcta. Además, el número de trazas bien asignadas aumenta con según lo hace la restricción del filtro.

Variable	$\|p_t\| \geq 0.5$ GeV	$\|p_t\| \geq 1.0$ GeV	$\|p_t\| \geq 1.5$ GeV
Trazas OK/Totales	0.27 ± 0.02	0.29 ± 0.03	0.31 ± 0.04
Vértices OK/Totales	0.033 ± 0.013	0.041 ± 0.017	0.041 ± 0.012
Nº Clusters/Nº Vértices	0.88 ± 0.09	0.65 ± 0.06	0.43 ± 0.03
Tiempo de ejecución/s	0.20 ± 0.07	0.12 ± 0.05	0.13 ± 0.04
Distancia	0.98 ± 0.10	0.28 ± 0.06	0.17 ± 0.02
Nota	0.970 ± 0.003	0.969 ± 0.003	0.965 ± 0.004
Nota Ajustada	0.26 ± 0.03	0.28 ± 0.03	0.32 ± 0.03
Trazas totales	2290 ± 190	850 ± 90	380 ± 50

Tabla 9: Resultados obtenidos por el algoritmo BIRCH al filtrar las trazas en función de su momento transverso $|p_t|$.

5.7 Comparación de los algoritmos

Todos los algoritmos mejoran su desempeño y la calidad de sus resultados al aplicar un filtro al momento transverso de las trazas. Siendo el filtro más óptimo $|p_t| \geq 1.0$ GeV ya que filtrando con mayores momentos decae demasiado el número de trazas y no se reconstruye el suficiente número de clusters, el número de clusters entre el número de vértices es mucho menor a 1. Todos los algoritmos mejoran su desempeño al aplicar el filtro $|p_t| \geq 0.5$ GeV respecto de no aplicar ninguno y al aplicar el filtro de 1.0 GeV también se encuentra una mejoría notable en la mayoría de algoritmos (excepto DBSCAN). Sin embargo, al seleccionar trazas únicamente con $|p_t| \geq 1.5$ GeV los algoritmos no mejoran de manera notable la calidad de sus resultados.

La mejoría que sufren los algoritmos estudiados consigue que alguno de ellos proporcione resultados superiores al algoritmo "Determinstc Annealing" cuando no se filtran los datos en este, en lo que se refiere a la reconstrucción correcta de vértices. Por ejemplo, el algoritmo K-Means con el filtro adecuado reconstruye de manera adecuada un número superior de vértices ($\textit{Vertices OK/Totales}_{\text{K-Means } |p_t| \geq 1.0\text{GeV}} = 0.04 \pm 0.02$ y $\textit{Vertices OK/Totales}_{\text{CMS}} = 0.034 \pm 0.010$). El algoritmo de CMS sin el filtrado de las trazas siempre asigna un número superior de trazas correctamente que los algoritmos estudiados con el filtrado de las mismas.

Comparando los algoritmos entre sí aplicando el filtro de $|p_t| \geq 1.0$GeV se observa que los algoritmos K-Means y EM-GMM proporcionan un número de trazas bien asignadas superior al resto aunque no existe una gran discrepancia con el resto de algoritmo (salvo el DBSCAN). Y, el algoritmo BIRCH junto con el EM-GMM reconstruye un número de vértices correctos superior al resto de algoritmos.

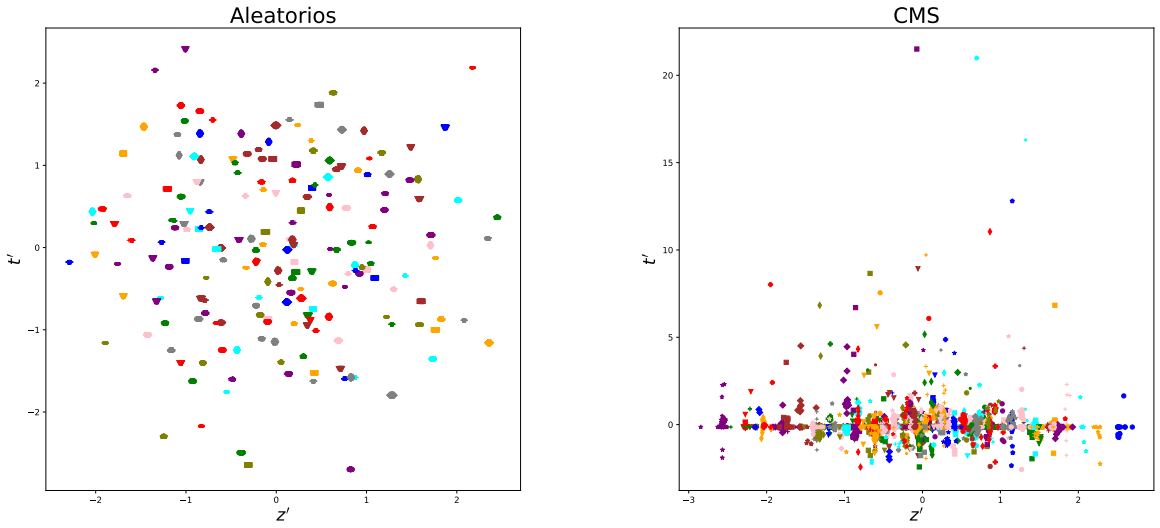
En lo relativo al tiempo de ejecución se observa que todos los algoritmos ven reducido este mismo cuánto más estricto es el filtro aplicado. Esto se debe a que su coste computacional crece con el número de datos, al filtrar las trazas se reduce el número de datos inicial por lo que su tiempo de ejecución se reduce.

Por último, la distancia no es un factor que sirva de comparación entre distintos filtros ya que depende de la distribución de datos iniciales que varía con cada filtro. Esto se debe a la renormalización que se ha aplicado ya que la media y la desviación estándar depende del conjunto de datos (ecuación (4)).

6 Conclusión

6.1 Discusión

En general se observa un mal desempeño de los algoritmos al ser aplicados a datos reales obtenidos en simulaciones de CMS reconstruyendo de manera adecuada vértices completos. Lo que es más llamativo se observa un gran descenso de la calidad al comparar el desempeño de estos mismos algoritmos en los conjuntos de datos generados de manera aleatoria. Esto se debe a la diferencia en estructura que tienen estos datos.



(a) Datos generados aleatoriamente

(b) Datos generados a partir de la simulación CMS

Figura 24: Trazas generadas y al vértice al que pertenecen en función de su posición. Todas las trazas pertenecientes al mismo vértice se representan con el mismo marcador.

En la figura 24 se observa cualitativamente como la densidad de trazas es similar en casi toda la figura 24a donde las trazas se han generado aleatoriamente; en cambio, en la figura 24b se observa como existe una zona de muy alta densidad entorno a $t = 0$ y que esta densidad decae hacia tiempos más altos.

De manera cuantitativa, la dispersión temporal en los datos generados aleatoriamente es $\sigma_{t'} \approx 4$ps, en cambio en la simulación de CMS se observa una dispersión $\sigma_{t'} \approx 20$. Es decir, utilizando un número de trazas y vértices similar los datos obtenidos a partir de la simulación de CMS están mucho más concentrados en una zona de alta densidad que los datos generados de manera aleatoria.

Por todo esto, se observa que los datos generados aleatoriamente no son una representación realista de los simulados por CMS. Y, por tanto, no son unos conjuntos de datos válidos para evaluar, ni para estimar el desempeño, de los algoritmos al ser aplicados al problema real. Aunque si son válidos para entender el funcionamiento y la implementación de los distintos algoritmos en un caso más sencillo.

Además, estas discrepancias en la densidad y distribución de los datos explica el bajo desempeño de los algoritmos al realizar el clustering de los datos de CMS en comparación con los datos generados aleatoriamente.

Por otra parte, estos algoritmos no son los más adecuados para realizar el clustering de estos datos, ya que la estructura de los datos no es isótropa en todo el espacio (condición para el correcto desempeño de K-Means y EM-GMM) sino que existe una alta densidad de trazas centrada $t = 0$ que decae al aumentar t. Así mismo, ciertos vértice generan alguna traza con un alto tiempo, por lo que la densidad no es similar para todos los vértices ni se tiene un círculo de radio constante que agrupe todas las trazas siempre, esto perjudica gravemente al algoritmo DBSCAN. El algoritmo Aglomerative Hierarchical Clustering al igual que BIRCH se basan en estructuras jerárquicas, cosa que en estos conjuntos de datos no existe, cada vértice es un proceso independiente del resto aunque los principios físicos que lo generen sean los mismos.

6.2 Conclusión

Con todo esto, se ha demostrado que los algoritmos estudiados (K-Means, MeanShift, DBSCAN, Agglomerative Hierarchical Clustering, Expectation Maximization-Gaussian Mixture Models y BIRCH) no suponen una mejora en calidad respecto del algoritmo "Deterministic Annealing" existente actualmente en el experimento CMS. Aunque mejoran la correcta asignación de trazas a sus correspondientes vértices, reconstruyen de manera adecuada un número mucho inferior de vértices.

Pese a esto ciertos algoritmo realizan un mejor clustering para objetivos concretos, por ejemplo el algoritmo DBSCAN estima de manera muy precisa la posición de los vértices.

Para intentar mejorar el desempeño de los algoritmos se realizó el estudio de la influencia del momento transverso de las trazas en los resultados obtenidos. Realizando el pre-filtrado de las trazas se obtiene que los resultados mejoran notablemente, en general. Esta mejoría se podría beneficiar de un reajuste de los parámetros iniciales para la estructura de los datos después del filtrado. Pese a esta mejoría ningún algoritmo tiene un desempeño similar al "Deterministic Annealing" en cuánto al número de vértices correctamente reconstruidos.

En conclusión, se recomienda un estudio más profundo de como afecta el momento transverso y otro tipo de filtros a las trazas al desempeño de los algoritmos. Debido a que se ha observado una mejoría de los resultados obtenidos por estos algoritmos al aplicar el filtro al momento. Ya que una adecuada comprensión de esta relación será de gran ayuda en la implementación y optimización de un algoritmo para el problema estudiado.

Por todo esto, no se recomiendo el uso de estos algoritmos (ni de algoritmos similares) para resolver el problema del clustering de las trazas en el experimento CMS en el futuro HL-LHC.

Aun así, teniendo en cuenta la calidad de los resultados y, sobretodo, su rápida ejecución se recomienda realizar un estudio en profundidad de las posibles aplicaciones de este tipo de algoritmos en la fases iniciales del trigger en el experimento CMS.

Referencias

[1] J. Hollar, A. Nayak, B. Calpas, N. Almeida, P. Bargassa, A. David Tinoco Mendes, P. Faccioli, P. G. Ferreira Parracho, M. Gallinaro, P. Q. Ribeiro, *et al.*, "Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC," 2012.

[2] "Modelo estándar." `https://es.wikipedia.org/wiki/Modelo_est%C3%A1ndar_de_la_f%C3%ADsica_de_part%C3%ADculas`. Acceso: 1 de julio de 2023.

[3] L. Evans and P. Bryant, "LHC machine," *Journal of instrumentation*, vol. 3, no. 08, p. S08001, 2008.

[4] "Mapa del LHC." `https://cds.cern.ch/record/2693837/files/Poster-2019-858.pdf`. Acceso: 1 de julio de 2023.

[5] S. Chatrchyan, G. Hmayakyan, V. Khachatryan, A. Sirunyan, W. Adam, T. Bauer, T. Bergauer, H. Bergauer, M. Dragicevic, J. Eroe, *et al.*, "The CMS experiment at the CERN LHC," *Journal of instrumentation*, vol. 3, 2008.

[6] R. L. Stratonovich, "Optimum nonlinear systems which bring about a separation of a signal with constant parameters from noise," *Radiofizika*, vol. 2, no. 6, pp. 892–901, 1959.

[7] E. Chabanat and N. Estre, "Deterministic annealing for vertex finding at CMS," 2005.

[8] G. Apollinari, I. Béjar Alonso, O. Brüning, P. Fessia, M. Lamont, L. Rossi, and L. Tavian, "High-Luminosity Large Hadron Collider (HL-LHC). Technical Design Report V. 0.1," tech. rep., Fermi National Accelerator Lab.(FNAL), Batavia, IL (United States), 2017.

[9] J. N. Butler and T. Tabarelli de Fatis, "A MIP timing detector for the CMS phase-2 upgrade," tech. rep., Fermi National Accelerator Lab.(FNAL), Batavia, IL (United States), 2019.

[10] S. Lenz, G. Borchert, H. Gorke, D. Gotta, T. Siems, D. Anagnostopoulos, M. Augsburger, D. Chatellard, J. Egger, D. Belmiloud, *et al.*, "A new determination of the mass of the charged pion," *Physics Letters B*, vol. 416, no. 1-2, pp. 50–55, 1998.

[11] P. R. y Arturo Amo, "Demonstration of k-means assumptions." `https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_assumptions.html#sphx-glr-auto-examples-cluster-plot-kmeans-assumptions-py`. Acceso: 1 de julio de 2023.

[12] "Comparison of the k-means and minibatchkmeans clustering algorithms." `https://scikit-learn.org/stable/auto_examples/cluster/plot_mini_batch_kmeans.html#sphx-glr-auto-examples-cluster-plot-mini-batch-kmeans-py`. Acceso: 1 de julio de 2023.

[13] "A demo of the mean-shift clustering algorithm." `https://scikit-learn.org/stable/auto_examples/cluster/plot_mean_shift.html#`

sphx-glr-auto-examples-cluster-plot-mean-shift-py. Acceso: 1 de julio de 2023.

[14] "Demo of dbscan clustering algorithm." https://scikit-learn.org/stable/auto_examples/cluster/plot_dbscan.html#sphx-glr-auto-examples-cluster-plot-dbscan-py. Acceso: 1 de julio de 2023.

[15] "Plot hierarchical clustering dendrogram." https://scikit-learn.org/stable/auto_examples/cluster/plot_agglomerative_dendrogram.html#sphx-glr-auto-examples-cluster-plot-agglomerative-dendrogram-py. Acceso: 1 de julio de 2023.

[16] "Gaussian mixture model selection." https://scikit-learn.org/stable/auto_examples/mixture/plot_gmm_selection.html#sphx-glr-auto-examples-mixture-plot-gmm-selection-py. Acceso: 1 de julio de 2023.

[17] "Repositorio de código." https://github.com/antoniohockey13/TFG. Acceso: 1 de julio de 2023.

[18] "numpy.random." https://numpy.org/doc/1.16/reference/routines.random.html. Acceso: 1 de julio de 2023.

[19] "scikit-learn." https://scikit-learn.org/stable/modules/classes.html#module-sklearn.cluster. Acceso: 1 de julio de 2023.

[20] L. Hubert and P. Arabie, "Comparing partitions," *Journal of classification*, vol. 2, pp. 193–218, 1985.

[21] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H.-S. Shao, T. Stelzer, P. Torrielli, and M. Zaro, "The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations," *Journal of High Energy Physics*, vol. 2014, no. 7, pp. 1–157, 2014.

[22] T. Sjöstrand, S. Ask, J. R. Christiansen, R. Corke, N. Desai, P. Ilten, S. Mrenna, S. Prestel, C. O. Rasmussen, and P. Z. Skands, "An introduction to pythia 8.2," *Computer physics communications*, vol. 191, pp. 159–177, 2015.

[23] S. Agostinelli, J. Allison, K. a. Amako, J. Apostolakis, H. Araujo, P. Arce, M. Asai, D. Axen, S. Banerjee, G. Barrand, *et al.*, "Geant4—a simulation toolkit," *Nuclear instruments and methods in physics research section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 506, no. 3, pp. 250–303, 2003.

[24] J. MacQueen, "Some methods for classification and analysis of multivariate observations university of california," *Los Angeles*, 1967.

Índice de figuras

Índice de tablas