

Fernando Sancho Caparrini

[Inicio](#) [Docencia](#) [Cursos](#) [Investigación](#) [Proyectos](#) [Publicaciones](#) [Últimas Entradas](#) [Por Temas](#)

Buscar palabras usadas en entradas y páginas de este sitio
Introduzca aquí la[s] palabra[s] a buscar:

Introduca términos de búsq



LOS MÁS LEÍDO

TOP 10

1. Aprendizaje Supervisado y No Supervisado (76140 vistas)
2. Introducción a la Lógica Difusa (37482 vistas)
3. Aprendizaje Inductivo: Árboles de Decisión (37209 vistas)
4. Redes Neuronales: una visión superficial (34832 vistas)
5. Algoritmos de hormigas y el problema del viajante (32992 vistas)
6. Introducción al Aprendizaje Automático (28297 vistas)
7. Sistemas Complejos, Sistemas Dinámicos y Redes Complejas (27886 vistas)
8. Entrenamiento de Redes Neuronales: mejorando el Gradiente Descendente (26570 vistas)
9. Sistemas Basados en Reglas (25651 vistas)
10. PSO: Optimización por enjambres de partículas (24124 vistas)

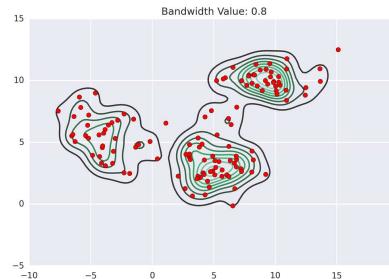
NETLOGO BOOK

Algoritmos de Clustering

« Planificación: Fundam... » || Inicio || » Variational

Última modificación: 20 de Diciembre de 2020, y ha ter.

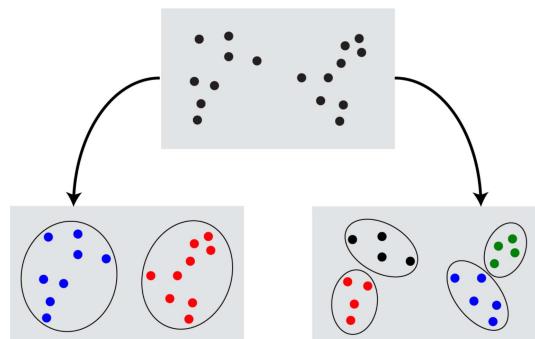
Etiquetas utilizadas: [algoritmos](#) || [aprendizaje automático](#) || [inteligencia artificial](#) || [machine learning](#)



Como hemos visto ya en [otras entradas](#) un algoritmo de **clustering** tiene como objetivo agrupar los objetos de un dataset según su **similaridad**, de forma que los objetos que hay dentro de un mismo grupo (**cluster**) sean más similares que aquellos que caen en grupos distintos.

Desde un punto de vista intuitivo, este problema tiene un objetivo muy claro: agrupar adecuadamente los datos no etiquetados. A pesar de su intuición, la noción de "clúster/agrupamiento" no puede ser demasiado precisa, una de las causas por las que se ha propuesto un rango tan amplio de algoritmos de clustering. Una lectura interesante [leer este blog](#).

Are these data better described by 2 or 4 clusters?



Para resolver este problema se han desarrollado muchos algoritmos que se diferencian entre sí en función de lo que entiende por cluster (que, en esencia, viene dado por cómo definimos que dos objetos son más o menos similares) y en función de la eficiencia computacional a la hora de conseguir la agrupación final.

Axiomatización del Clustering

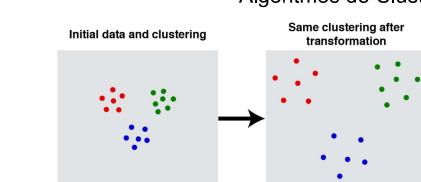
Normalmente, para poder hablar de similaridad se suele acudir a algún tipo de **distancia** (aunque no necesariamente se conformamos con algo que no llega a cumplir todas las propiedades de una **distancia/métrica** y trabaja con **pseudo-métricas**), con el fin de poder asociar la similitud de los objetos analizados con la distancia que existe entre ellos. Por ejemplo, si podemos describir el objeto por medio de un vector numérico de propiedades, es habitual tratar con métricas vectoriales (como la **distancia euclídea**) para medir la similitud entre los objetos.

Jon Kleinberg (J. Kleinberg, “An impossibility theorem for clustering” en Proceedings of the 15th International Conference on Neural Information Processing Systems, NIPS’02, pp. 463–470, MIT Press, 2002) propone tres axiomas que destacan las características que debe presentar un problema de agrupamiento y que pueden considerarse independientemente del algoritmo utilizado para encontrar la solución. Estos axiomas son (se puede ver [introducción aquí](#)):

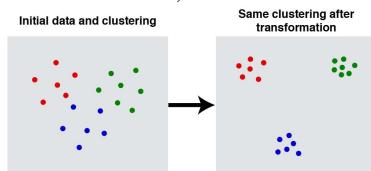
1. **Invariancia por escala:** Un algoritmo de clustering no debe dar resultados distintos si se aplica al mismo conjunto de puntos.



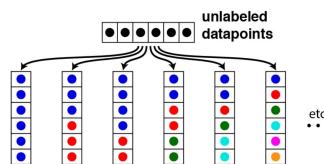
In english and spanish, the new book to learn NetLogo. 16 chapters going from the basics to advanced features. With exercises proposed in every chapter and full examples.



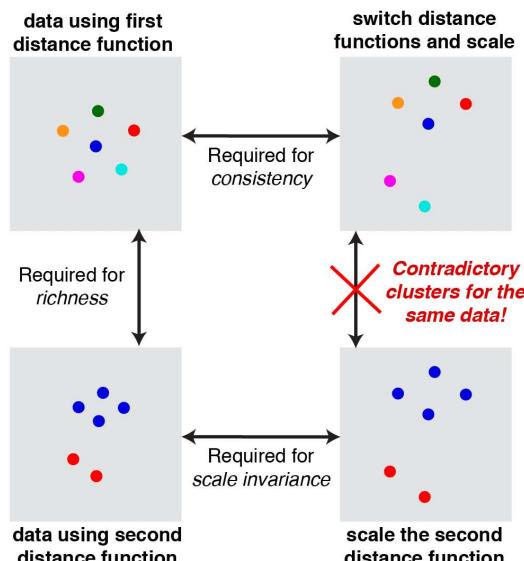
- 2. Consistencia:** Un algoritmo de clustering no debe dar resultados distintos si las distancias dentro de los clústeres se reducen, o si las distancias entre clústeres se aumentan.



- 3. Riqueza:** La función de agrupación debe ser lo suficientemente flexible como para producir una partición/agrupación arbitraria del conjunto de datos de entrada.

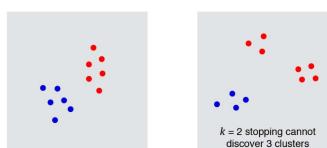


El resultado más interesante del trabajo de Kleinberg es que, pese al claro carácter intuitivo de estos axiomas, puede construir una función de agrupamiento que verifique los tres simultáneamente.

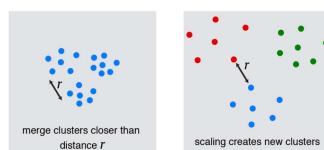


Y ya que los tres axiomas no pueden verificarse simultáneamente, se pueden diseñar algoritmos de clustering que violan uno de los axiomas mientras se verifican los otros dos. Kleinberg ilustra este punto describiendo tres tipos particulares de jerárquico (que veremos más abajo, como ejemplo de algoritmo de clustering):

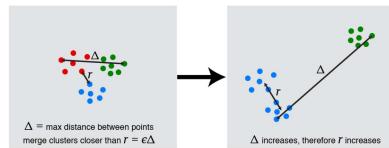
- Condición de parada del k -clúster:** Dejar de unir clústeres cuando ya hay k clústeres (viola la riqueza, ya que el algoritmo nunca devolvería un número de clústeres diferente a k).



- Condición de parada de la r -distancia:** Dejar de unir clústeres cuando el par más cercano de puntos es más lejos que r (viola la invariancia de escala, ya que cada clúster contendrá una sola instancia cuando la escala es grande, y un solo clúster con todos los datos cuando la escala es muy pequeña).



- **Condición de detención de la escala:** Dejar de unir clústers cuando el par de clústers más cercanos sea más lejos que una fracción de la distancia máxima por pares (se satisface la invariancia de escala, pero no la consistencia).



Clasificación de Algoritmos de Clustering

Según la forma en que los clusters se relacionan entre sí y con los objetos del dataset, podemos establecer una primera división entre los diversos algoritmos existentes:

- **Clustering Duro:** donde cada objeto pertenece a un solo cluster (por lo que los clusters pasarían a ser así como una partición del dataset).
- **Clustering Blando (o difuso):** donde los objetos pertenecen a los clusters según un grado de pertenencia (con un grado de pertenencia).

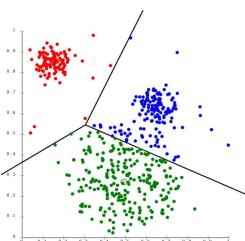
Pero a veces también podemos encontrar una clasificación más fina atendiendo a cómo se relacionan con los clusters:

- **Partición estricta:** cada objeto pertenece exactamente a un cluster.
- **Partición estricta con outliers:** puede haber objetos que no pertenecen a ningún cluster (los outliers).
- **Clustering con superposiciones:** un objeto puede pertenecer a más de un cluster.
- **Clustering Jerárquico:** Los clusters se ordenan jerárquicamente de forma que los objetos que pertenecen a un cluster también pertenecen a su cluster padre.

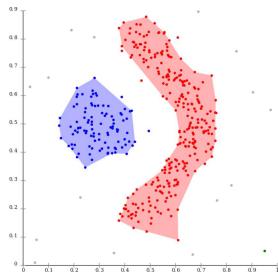
Algunos representantes

Entre la diversidad de aproximaciones existentes en la literatura, en esta entrada nos vamos a centrar en las más representativas ([se pueden encontrar implementaciones específicas de los algoritmos para NetLogo](#)). Concretamente, los algoritmos que veremos a continuación son:

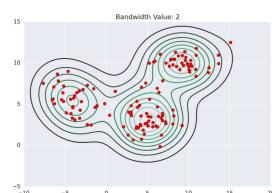
1. **K-medias:** modelo de centroides.



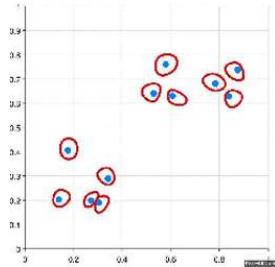
2. **DBSCAN:** modelo de densidad discreto.



3. **Mean Shift:** modelo de gradientes en densidad.



4. **AGNES:** modelo jerárquico.



K-medias

Este algoritmo intenta encontrar una partición de las muestras en K (parámetro del algoritmo) agrupando que cada ejemplo pertenezca a una de ellas, concretamente a aquella cuyo centroide esté más cercano. Para que la clasificación separe lo mejor posible los ejemplos no se conoce a priori, completamente de los datos con los que trabajemos. Este algoritmo intenta minimizar la varianza total del decir, si c_i es el centroide de la agrupación i -ésima, y $\{x_{ij}\}$ es el conjunto de ejemplos clasificados en esa agrupación, entonces intentamos minimizar la función:

$$\sum_i \sum_j d(x_{ij}, c_i)^2$$

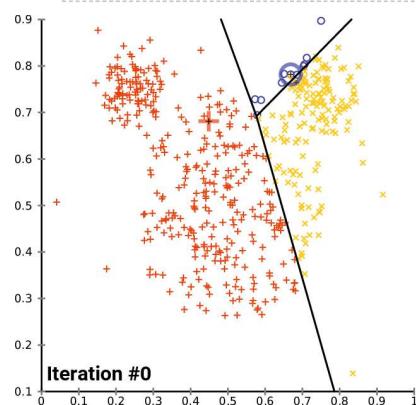
que, a veces suele denominarse **función potencial**.

Aunque podríamos utilizar cualquier algoritmo de optimización para realizar esta tarea, en este caso se trata de un algoritmo recursivo que converge a una solución local (un mínimo local de la función potencial) de forma relativamente eficiente:

```

K-MEANS(D,K) [D: Dataset, K > 0]
  C = {c1,...,ck} ⊆ D (cualesquiera)
  Asignar Di = {P ∈ D: d(P,ci) < d(P,cj), j ≠ i}, i = 1...K
  Mientras algún ci cambie:
    Calcular ci = centroide(Di), i=1...K
    Recalcular Di, i = 1...K
  Devolver {D1,...,DK}

```



El algoritmo anterior es relativamente eficiente, y normalmente se requieren pocos pasos para que el resultado sea estable. Pero en contra, es necesario determinar el número de agrupaciones a priori, y el sistema es sensible a la posición inicial de los K clusters seleccionados, haciendo que no consigan un mínimo global, sino que se quedan en un mínimo local (algo muy común cuando se trabaja con un problema de optimización no convexo).

Por desgracia, no existe un método teórico global que permita encontrar el valor óptimo de grupos iniciando en una posición arbitraria. La elección de las posiciones en las que debemos situar los centros, por lo que se suele hacer una aproximación experimental realizando el algoritmo con diversos valores y posiciones de centros. En general, un valor elevado de K hace que el error sea menor pero a cambio disminuye la cantidad de información que la agrupación resultante proporciona.

DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) es un algoritmo de agrupamiento basado en densidades, desarrollado en 1996, no paramétrico, basado en densidades, y que marca como **outliers** aquellos puntos que no están dentro de la nube de puntos. Constituye uno de los algoritmos de agrupamiento más usados y citados, y es relativamente eficiente cuando se usan estructuras de datos adecuadas para su implementación (pudiendo ejecutarse en tiempo $O(n^2)$ en el peor caso).

El algoritmo recibe como dato de entrada dos valores que, juntos, determinan la densidad mínima: ε (radio de los entornos que se mirarán alrededor de los puntos del conjunto), y $MinPts$ (que establece mínimo de puntos que debe haber en el entorno para que se considere que se supera la densidad mínima).

Con estos parámetros prefijados, el algoritmo opera como sigue:

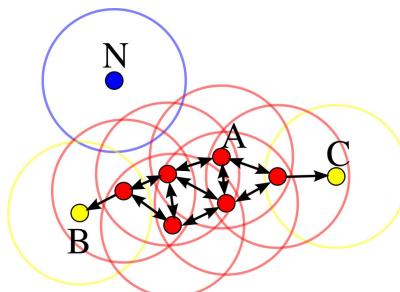
```
DBSCAN(D, ε, MinPts) [D: Dataset; ε > 0; MinPts > 0]
    C = ∅
    Para cada P ∈ D no visitado:
        Marcar P como visitado
        N = E(P, ε)
        Si tamaño(N) < MinPts:
            marcar P como OUTLIER
        si no:
            C = C ∪ {creaCluster(P, N, ε, MinPts)}
    Devolver C
```

Donde:

```
creaCluster(P, N, ε, MinPts) [P ∈ D; N entorno de P; ε > 0; MinPts > 0]
    Cl = {P}
    Para cada P' ∈ N:
        Si P' no ha sido visitado:
            marcar P' como visitado
            N' = E(P', ε)
            Si tamaño(N') ≥ MinPts:
                N = N ∪ N'
                Si P' no está en ningún cluster:
                    Cl = Cl ∪ {P'}
    Devolver Cl
```

y

$$E(P, \varepsilon) = \{P' \in D : d(P, P') < \varepsilon\}$$



Esencialmente, el algoritmo comienza por un punto con suficiente densidad y va construyendo un cluster de puntos cercanos relevantes al mismo. Cuando acaba de construir un cluster, salta a otro punto con suficiente densidad y construye otro cluster independiente.

En comparación con K-medias, DBSCAN presenta las siguientes ventajas:

- No es necesario especificar del número de clusters deseado.
- Puede encontrar clusters con formas geométricas arbitrarias. Gracias al parámetro $MinPts$, se permite la posibilidad de que aparezcan clusters distintos conectados por una delgada línea de puntos.
- Tiene capacidad de detectar outliers (y ruido).
- Solo necesita dos parámetros y no importa el orden de recorrido de los puntos para la construcción de los clusters (solo puede haber indeterminismo en los puntos fronterizos entre dos clusters distintos, pero pueden acabar asignados a uno u otro, pero no es fácil que ocurra, y el número de puntos involucrados es relativamente bajo).
- Usando estructuras de datos adecuadas se puede dar una implementación bastante más rápida.

A cambio, presenta algunas desventajas, que no son específicas, sino comunes a muchos algoritmos de clustering:

- El uso de distancias equivalentes a la euclídea puede dar resultados indeseados en dimensiones altas.
- Los valores de $(MinPts, \varepsilon)$ se pueden adaptar para tratar conjuntos de densidades específicas, pero si hay conjuntos con mucha diferencia en sus densidades, no hay combinación de valores que pueda

adecuadamente para todos ellos a la vez.

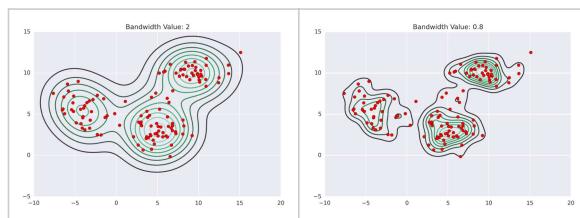
Además, se puede usar en datasets que no se representan necesariamente en espacios vectoriales, ya que tener definida una distancia entre puntos, no de la representación completa.

Mean Shift

Esta aproximación se basa en el concepto de **Kernel de Estimación de Densidad (KDE)**. La idea es que el conjunto de puntos que se quiere procesar proviene del muestreo de una distribución de probabilidad dada. El KDE proporciona un método para estimar la distribución subyacente. Funciona poniendo una función (una densidad) en cada uno de los puntos que representa una ponderación. Hay muchas posibles funciones que usar como kernel puntuales, pero quizás la más habitual es el kernel **Gaussiano**.

$$K(x) = e^{-\frac{(x-x_i)^2}{2h^2}}$$

Cambiando la función kernel (o los parámetros de un kernel específico) se obtiene una estimación de la función de densidad. Por ejemplo, si los kernels son muy concentrados en cada punto (en el kernel G se consigue haciendo $h \approx 0$ muy pequeño), entonces la función de densidad obtenida separará los puntos en pequeños picos que los aislan; si los kernels son muy suavizados (en el kernel Gaussiano se consigue haciendo $h \approx 10$ muy grande), entonces la función de densidad se asemejará a una gran meseta que conecta los diversos puntos entre sí.



Una vez obtenida la función de densidad, asociada a una posible distribución de probabilidad que ha generado el dataset, se puede usar esta densidad para establecer los distintos clusters que se asocian a la distribución. Para saber qué puntos están en el mismo cluster, el algoritmo **Mean Shift** aplica un proceso de ascenso degradante para ver cuáles acaban en el mismo máximo (de esta forma, se asocia cada cluster con las diversas cuencas de atracción de la función de densidad).

Así pues, el algoritmo se podría describir brevemente como:

```

MEAN-SHIFT(D,K) [D: Dataset; K: kernel]
     $f(x) = \sum_{P \in D} K_P(x)$  (función de densidad centrada en P a partir de K)
     $M = \emptyset$ 
    Para cada  $P \in D$ :
         $M = M \cup \text{Max}(f,P)$  (aplicando  $\text{grad}(f)$  desde P)
    Para cada  $m \in M$ :
         $C_m = \{P \in D: \text{Max}(f,P) = m\}$ 
    Devolver  $\{C_m: m \in M\}$ 

```

El hecho de que Mean Shift no haga suposiciones acerca del número de clusters, o de la forma que tienen, convierten en un método ideal para trabajar con un número arbitrario de clusters y de formas arbitrarias. De hecho, el algoritmo que se ha explicado es realmente un algoritmo de optimización (los clusters se identifican como los máximos de la función de densidad), el uso de las diversas cuencas de atracción como clusters del dataset proporciona una ingeniosa (y fructífera) identificación entre conceptos provenientes de mundos intuitivamente diferentes.

AGNES

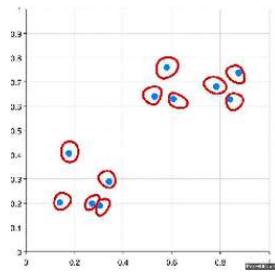
Vamos a ver el que se conoce como algoritmo **AGNES (Agglomerative Nesting)**, que es un algoritmo de clustering **aglomerativo**, es decir, va construyendo una jerarquía creciente de clusters desde los clusters más pequeños posibles, los puntos aislados, hasta el mayor cluster posible, que agrupa todos los puntos del dataset.

En general, este algoritmo sigue los siguientes pasos:

```

AGNES(D,N) [D: Dataset, N > 0]
     $Clas = \{\{P\} : P \in D\}$  (un cluster unitario para cada  $P \in D$ )
    Mientras  $|Clas| > N$ :
        Sean  $C_1, C_2$  en  $Clas$ :  $d(C_1, C_2) = \min \{d(C_i, C_j) : C_i, C_j \in Clas\}$ 
         $C_{12} = C_1 \cup C_2$ 
         $Clas = Clas - \{C_1, C_2\} \cup \{C_{12}\}$ 
    Devolver  $Clas$ 

```

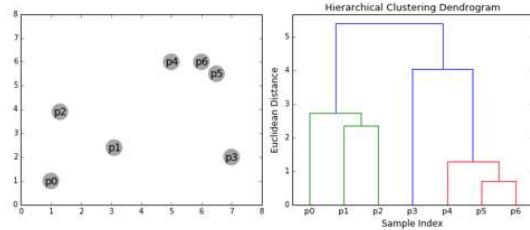


La única diferencia que se puede introducir en este algoritmo es acerca de cómo medir la distancia entre los puntos para reconocer los más cercanos. Es lo que se conoce como **Métodos de Conexión**, y supuesto que tenemos una medida de distancia, d , definida entre los puntos del dataset, las distancias entre clusters más habituales son:

1. **Conexión completa:** $d_{max}(C_1, C_2) = \max\{d(x_1, x_2) : x_1 \in C_1, x_2 \in C_2\}$.
2. **Conexión simple:** $d_{min}(C_1, C_2) = \min\{d(x_1, x_2) : x_1 \in C_1, x_2 \in C_2\}$.
3. **Conexión media:** $d_{mean}(C_1, C_2) = \text{mean}\{d(x_1, x_2) : x_1 \in C_1, x_2 \in C_2\}$.
4. **Conexión centroide:** $d_{cent}(C_1, C_2) = d(c_1, c_2)$, donde $c_i = \text{centroide}(C_i)$.

Se debe tener en cuenta que no hay ninguna elección mejor que otra, y que diferentes distancias ofrecerán agrupaciones resultantes.

Debido a la complejidad de representar simultáneamente toda la jerarquía de clusters que aparece en estos algoritmos, se han ideado formas esquemáticas para mostrarlos atendiendo al orden y dando una idea de la jerarquía que hay entre los diversos clusters que aparecen. La más común de ellas es la que se conoce como dendograma, que nos permite hacerse una idea de todas las agrupaciones de forma bastante sencilla:



En este tipo de representaciones, además, la altura de los bloques representa la distancia entre los clusters que se unen. En cada altura, el número de ejes verticales indica cuántos clusters hay en ese nivel jerárquico.

Este tipo de algoritmos tienen ventajas claras sobre el algoritmo base de K-medias, como por ejemplo que no necesita que los datasets se representen en espacios vectoriales, ya que solo precisa de la existencia de una medida de distancia entre puntos, no de proyectarlos completamente. Por contra, no es especialmente adecuado cuando el dataset es muy grande.

Existen variantes similares que siguen el proceso contrario: partiendo del cluster formado por todos los puntos del dataset, van dividiendo sucesivamente los clusters presentes para separar aquellos en los que se encuentre una mayor disimilaridad entre sus elementos.

TAMBIÉN TE PUEDE INTERESAR

- [1. Aprendizaje Supervisado y No Supervisado \(7614 vistas\)](#)
- [2. Introducción a la Lógica Difusa \(37482 vistas\)](#)
- [3. Aprendizaje Inductivo: Árboles de Decisión \(3720 vistas\)](#)
- [4. Redes Neuronales: una visión superficial \(34832 vidas\)](#)
- [5. Algoritmos de hormigas y el problema del viaje del vendedor \(27886 vistas\)](#)
- [6. Introducción al Aprendizaje Automático \(28297 vidas\)](#)
- [7. Sistemas Complejos, Sistemas Dinámicos y Redes \(27886 vistas\)](#)
- [8. Entrenamiento de Redes Neuronales: mejorando el Algoritmo Descendente \(26570 vistas\)](#)
- [9. Sistemas Basados en Reglas \(25651 vistas\)](#)
- [10. PSO: Optimización por enjambres de partículas \(24561 vistas\)](#)

1 Comentario Añade un comentario si lo deseas...  Política de privacidad de Disqus **Favorite** Tweet Compartir**Ordenar por** 

Únete a la conversación...

[INICIAR SESIÓN CON](#)[O REGISTRARSE CON DISQUS !\[\]\(e1c624d4757f08486e89482c18364c17_img.jpg\)](#)

Nombre

**Mirlo T** • hace 8 meses • edited

Muchas gracias por tan valioso aporte teorico. felicitaciones

 |  • Responder • Compartir  [Suscríbete](#)[Añade Disqus a tu sitio web](#)[Añadir](#)[Do Not Sell My Data](#)[« Planificación: Fundam...](#) « || [Inicio](#) || » [Variational](#)Magazine Theme for PivotX  by Windmill Web Work  in 2009, released under the Simple Public License .