

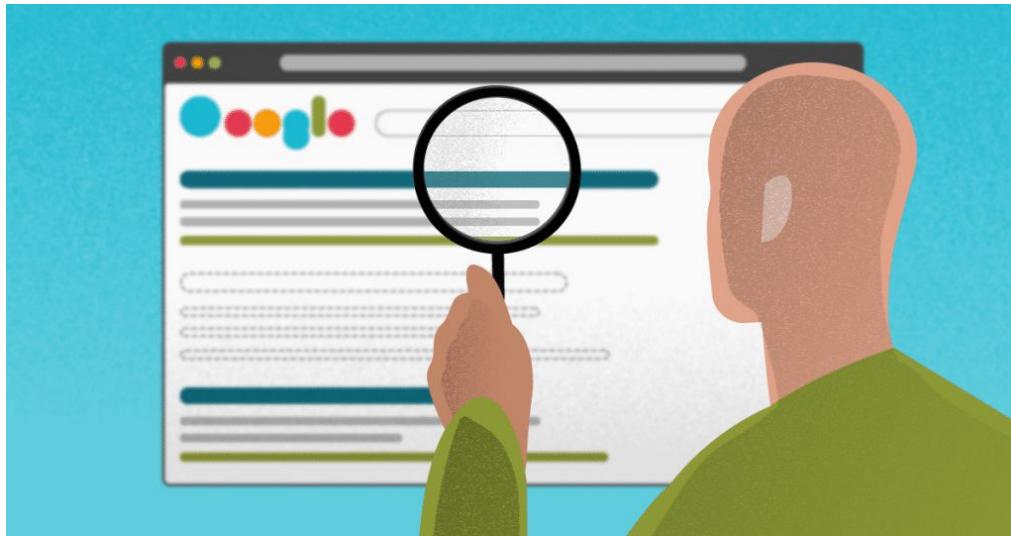
Datos Masivos I

3. Búsqueda de elementos similares

Profa. Alondra Berzunza

3. Búsqueda de elementos similares

Un problema fundamental de la minería de datos, es examinar los datos en búsqueda de elementos similares.



¿Cómo encontramos imágenes?

house

Page 2 of about 413,000,000 results (0.08 seconds)



Related searches: [house tv show](#) [greg house](#) [house clipart](#) [cartoon house](#) [house music](#)



Click the Small House In The
465 × 346 - 58k - jpg
[supercoloring.com](#)
[Find similar images](#)



The house ...
600 × 400 - 93k - jpg
[museumoffloridahistory.com](#)
[Find similar images](#)



This large house ...
500 × 375 - 43k - jpg
[glamro.gov.uk](#)
[Find similar images](#)



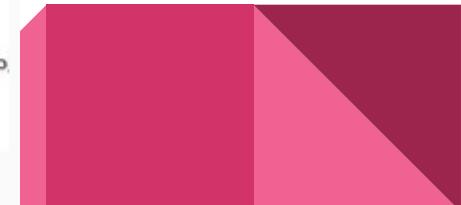
[HouseplanGuys.com](#), The largest
500 × 300 - 35k - jpg
[houseplanguys.com](#)
[Find similar images](#)



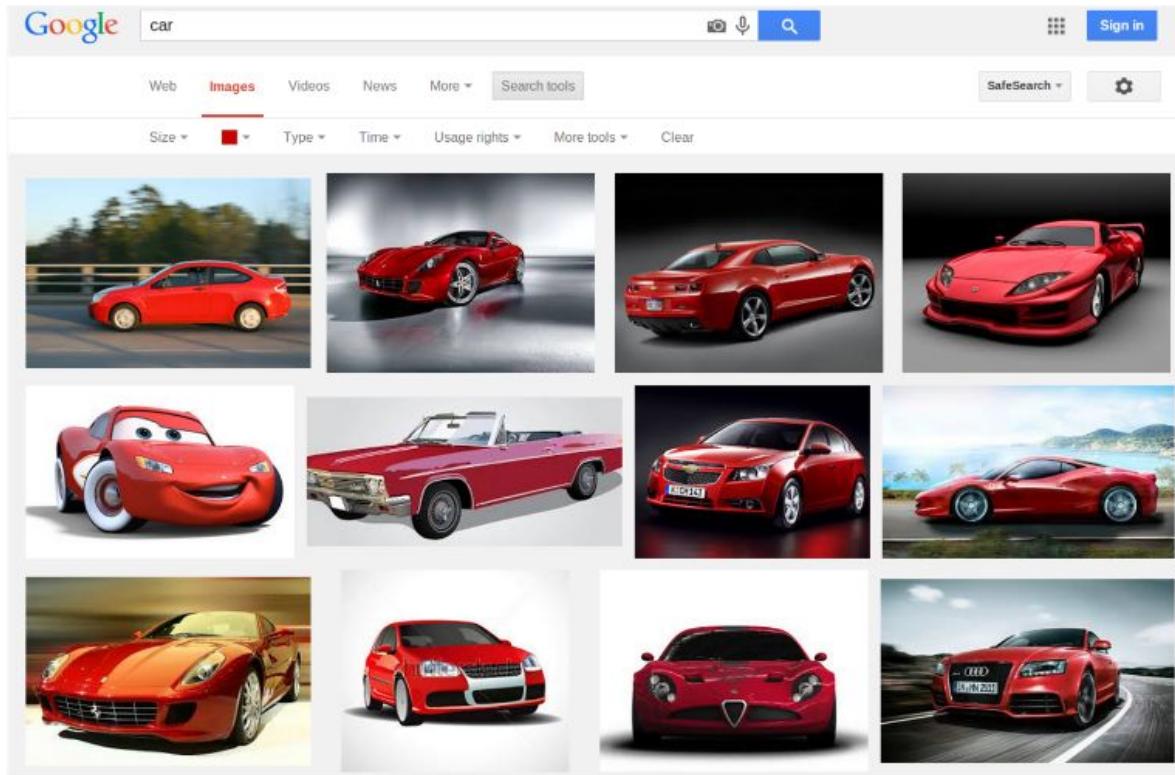
[House picture by atkinson_crystal](#)
800 × 600 - 299k - jpg
[s588.photobucket.com](#)
[Find similar images](#)



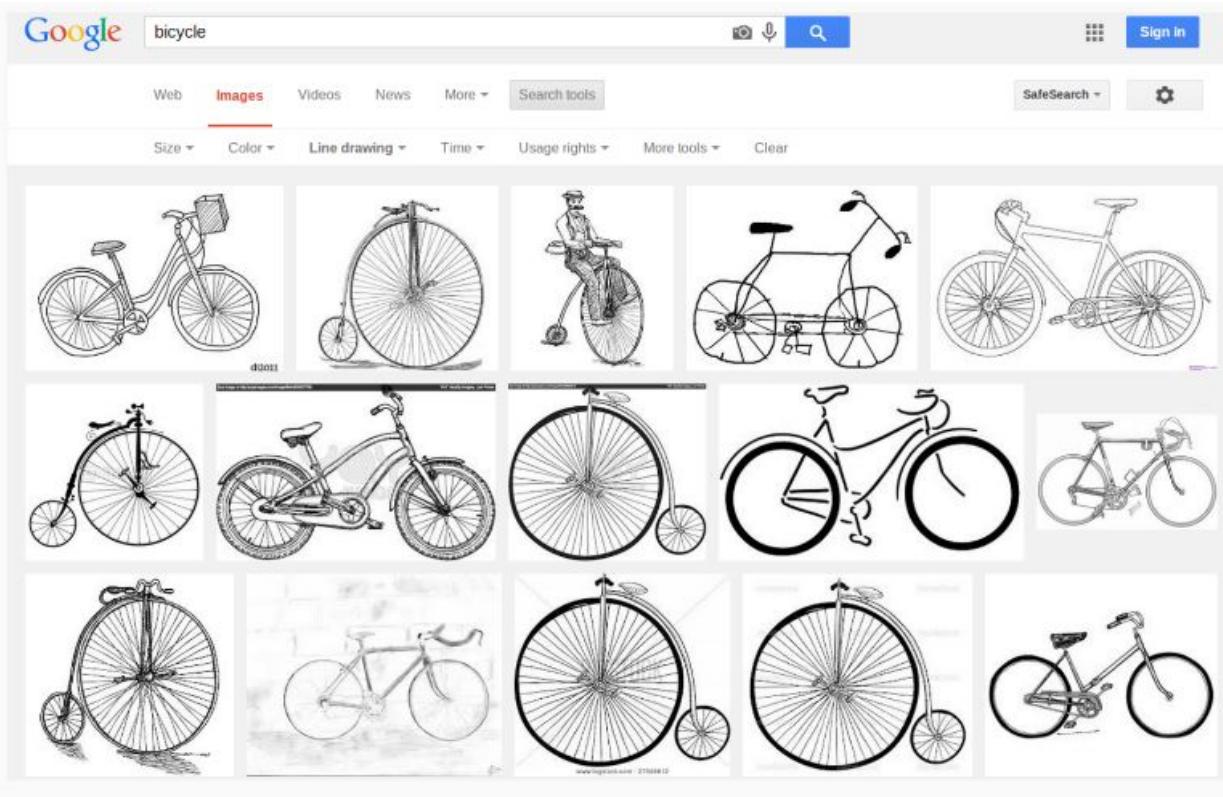
Barack & Michelle Obama P.
622 × 402 - 104k - jpg
[hiptics.com](#)
[Find similar images](#)



Búsqueda de imágenes por color



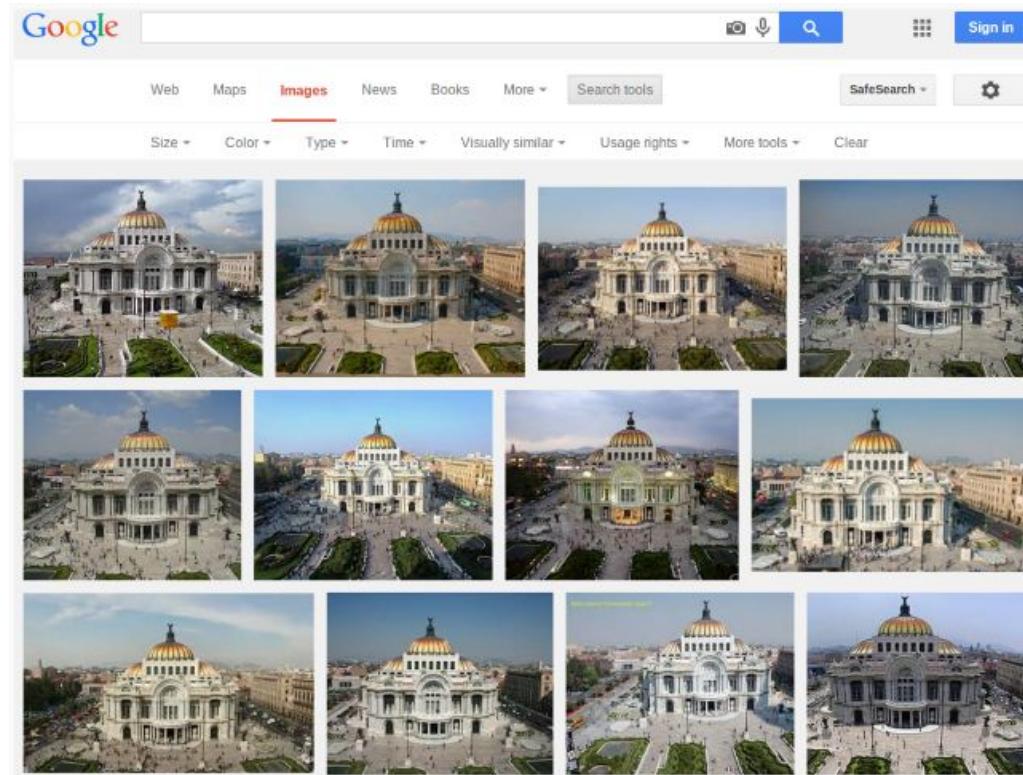
Búsqueda de imágenes por estilo



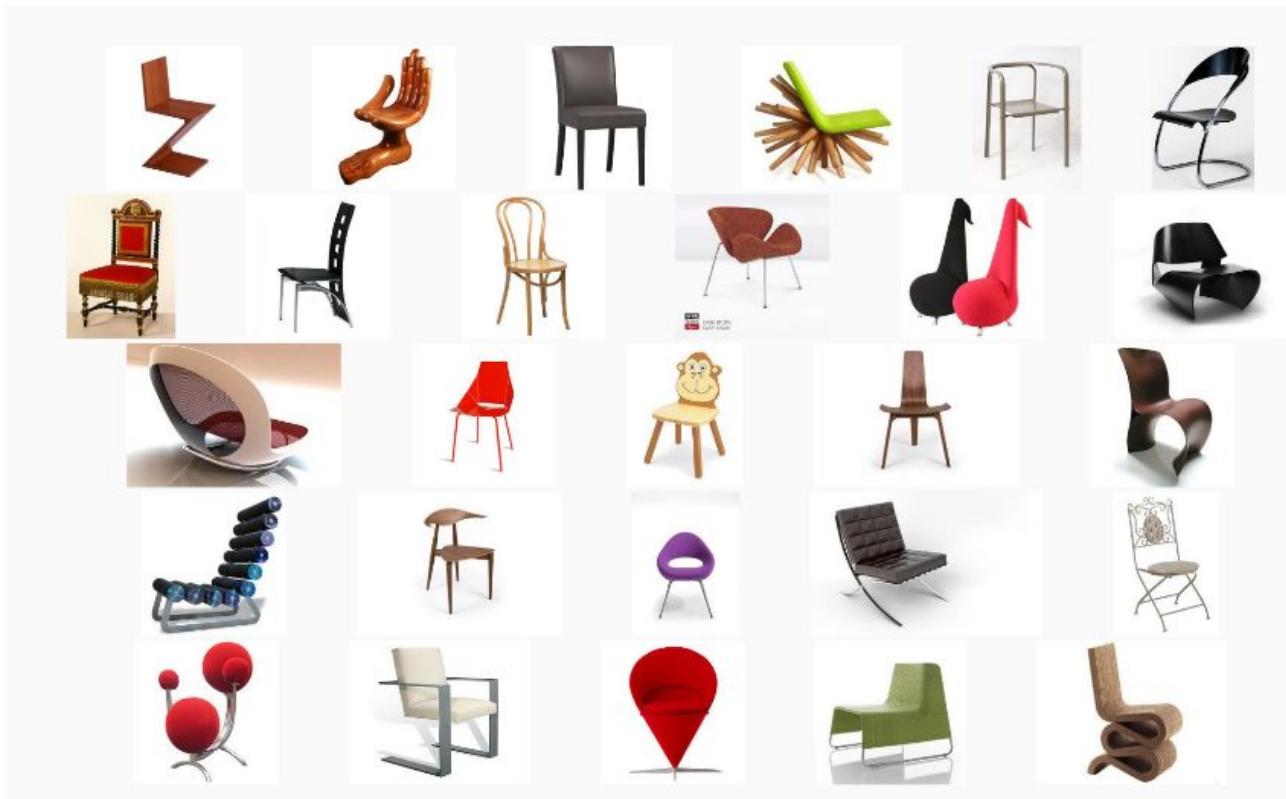
Búsqueda de imágenes con características específicas



Búsqueda de imágenes visualmente similares



Búsqueda de imágenes de la misma categoría

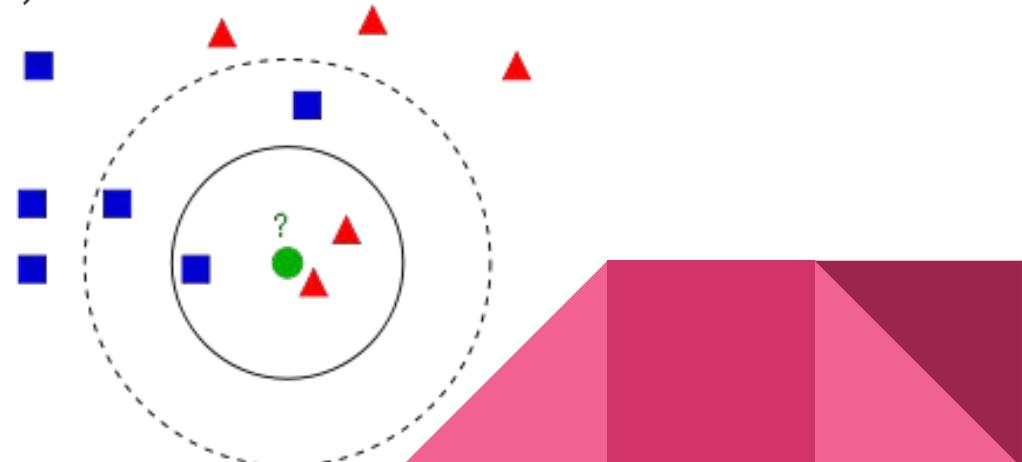


Medidas de similaridad

El problema del vecino más cercano

Tarea frecuente en análisis de datos (por ej. agrupamientos de clientes similares, búsqueda de documentos sobre el mismo tema, etc).

Usado por algunos métodos no paramétricos de aprendizaje de máquina (por ej. clasificador de k-vecinos más cercanos).



El problema del vecino más cercano

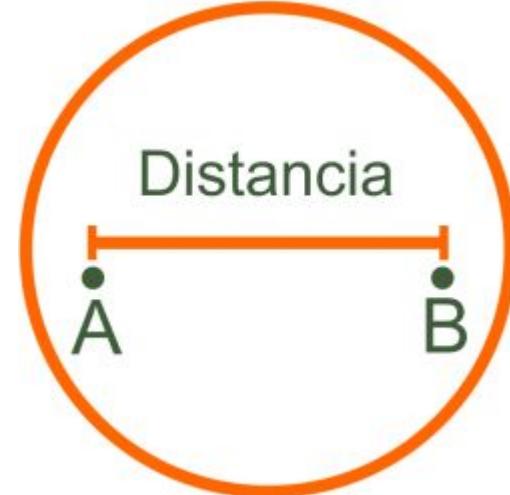
El problema es encontrar el par de objetos $(x^{(1)}, x^{(2)}) \in X$ que son más similares o que son más cercanos bajo algún criterio de similitud o distancia $M(x^{(1)}, x^{(2)})$

Usando fuerza bruta requería comparar todos los pares posibles en X lo cual es

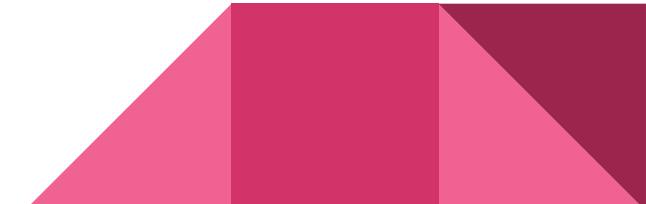
$$\frac{n}{2} = \lambda(n^2)$$

Función de Distancia

- Cuantifica la disimilitud entre dos objetos.
- Debe cumplir las siguientes propiedades:
 - No negativo: $dist(x, y) \geq 0$
 - Identidad de los indiscernibles: $dist(x, y) = 0 \Leftrightarrow x = y$
 - Simétrico: $dist(x, y) = dist(y, x)$
 - Desigualdad del triángulo: $dist(x, y) \leq dist(x, z) + dist(z, y)$



No siempre se mantienen las propiedades de las distancias en la percepción humana. Por ejemplo, la desigualdad del triángulo.



Distancia Euclídea

Representa el tamaño del segmento de línea que conecta dos puntos \mathbf{x}, \mathbf{y} .

$$dist(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{k=1}^L (y_k - x_k)^2}$$

Similitud y Distancia coseno

La similitud coseno compara la orientación de dos vectores mediante el coseno del ángulo entre ellos.

$$S_C(\mathbf{x}, \mathbf{y}) = \cos(\theta) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \cdot \|\mathbf{y}\|} = \frac{\sum_{i=1}^L x_i \cdot y_i}{\sqrt{\sum_{i=1}^L x_i^2} \cdot \sqrt{\sum_{i=1}^L y_i^2}}$$

Dos vectores con la misma orientación tienen una similitud de 1.

Comúnmente usada para comparar documentos de texto.

Distancia Hamming

Para un tamaño fijo T, es el número de elementos distintos de 2 vectores o cadenas.

Ejercicio:

Calcula la distancia de Hamming de los siguientes objetos.

1. 'asar' y 'azar'
2. 57941137 y 23111431
3. 10110010 y 01011110

Similitud y Distancia de Jaccard

Dados 2 conjuntos $\{c^{(1)}, c^{(2)}\}$ su similitud de Jaccard se define como:

$$J(c^{(1)}, c^{(2)}) = \frac{|c^{(1)} \cap c^{(2)}|}{|c^{(1)} \cup c^{(2)}|} \in [0, 1]$$

La distancia de Jaccard es:

$$dist_j(c^{(1)}, c^{(2)}) = 1 - J(c^{(1)}, c^{(2)})$$

Ejercicio:

Calcula la similitud de Jaccard de los conjuntos:

$$c^{(1)} = \{0, 3, 6, 7\} \quad c^{(2)} = \{2, 3, 5, 7\}$$

Generalización de Similitud de Jaccard para bolsas

$$J(B^{(1)}, B^{(2)}) = \frac{\sum_{w=1}^D \min(B_w^{(1)}, B_w^{(2)})}{\sum_{w=1}^D \max(B_w^{(1)}, B_w^{(2)})} \in [0,1]$$

Donde $B_w^{(1)}, B_w^{(2)}$ son las multiplicidades del elemento w en las bolsas $B^{(1)}, B^{(2)}$ respectivamente.

La distancia generalizada es $dist_J(B^{(1)}, B^{(2)}) = 1 - J(B^{(1)}, B^{(2)})$

Ejercicio:

Calcula la similitud generalizada de las siguientes bolsas

$$B^{(1)} = \{(0,2), (3,1), (6,1), (7,3)\}$$

$$B^{(2)} = \{(2,1), (3,2), (5,3), (7,1)\}$$

Traslape de dos Conjuntos

Número de elementos en común sobre mínimo de elementos de dos conjuntos.

$$ovr(c^{(1)}, c^{(2)}) = \frac{|c^{(1)} \cap c^{(2)}|}{\min(|c^{(1)}|, |c^{(2)}|)}$$

Ejercicio:

Calcula el traslape entre los siguientes pares de conjuntos.

$$c^{(1)} = \{0, 3, 6, 7\} \text{ y } c^{(2)} = \{2, 3, 5, 7\}$$

$$c^{(1)} = \{0, 3, 6, 7\} \text{ y } c^{(2)} = \{0, 3, 7\}$$

$$c^{(1)} = \{2, 3, 7\} \text{ y } c^{(2)} = \{2, 3, 4, 7\}$$



Árboles Kd

Árbol binario de k-dimensiones, que sirve para realizar la búsqueda del vecino más cercano.

Cada nivel del árbol se refiere a una dimensión.

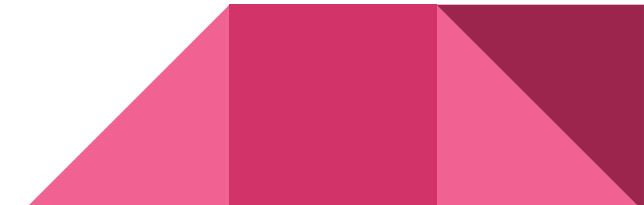
Para buscar puntos:

1. Se construye el árbol con el conjunto de puntos disponible
2. Dado un nuevo punto de consulta, se busca el punto más cercano recorriendo el árbol.



Construcción de un árbol

1. Elegir la dimensión de forma alternada (ej. para 2D, la raíz usa X, sus hijos Y, y así sucesivamente).
2. Inserta punto con valor en el criterio establecido (ej. la mediana) de la dimensión seleccionada, puntos menores son descendientes en su rama izquierda y mayores en su derecha.
3. Se repite 1 y 2 para los descendientes hasta que ya no haya más puntos.

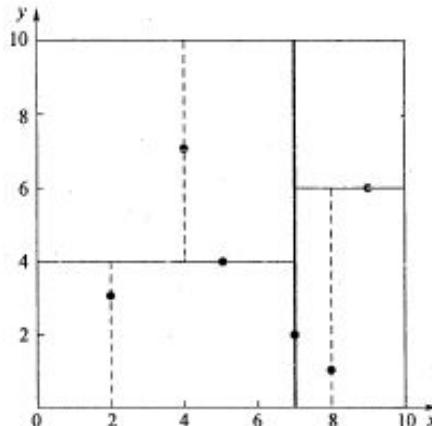


Construcción de un árbol

Supongamos que hay 6 puntos de datos bidimensionales

$$\{(2,3), (5,4), (9,6), (4,7), (8,1), (7,2)\},$$

y los puntos de datos están ubicados en pares dentro del espacio dimensional.



Construcción de un árbol

$$\{(2,3), (5,4), (9,6), (4,7), (8,1), (7,2)\}$$

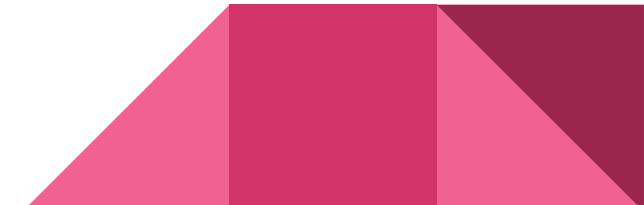
1. Determine el primer valor del dominio dividido. Calcule la varianza de los datos en x y de los datos en y.

Nos damos cuenta que el que tiene mayor varianza es x, por lo que el valor del dominio dividido se toma primero como 0, que es la dirección del eje x.

2. Determine el valor de dominio de datos de nodo. De acuerdo con los valores en la dirección del eje x 2,5,9,4,8,7.

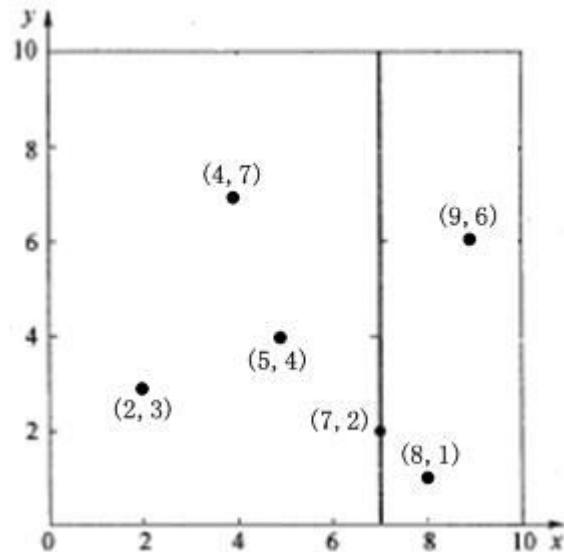
El valor medio es 7, por lo que los datos de nodo = (7,2).

De esta manera, el hiperplano de segmentación de este nodo es una línea recta $x = 7$ que pasa a través de (7, 2) y perpendicular a split = 0 (eje x);



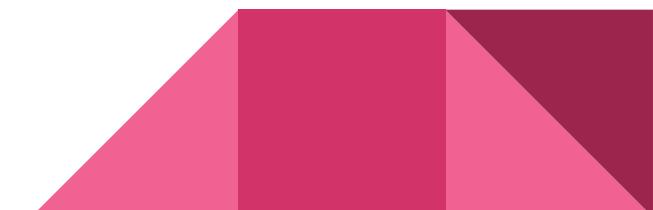
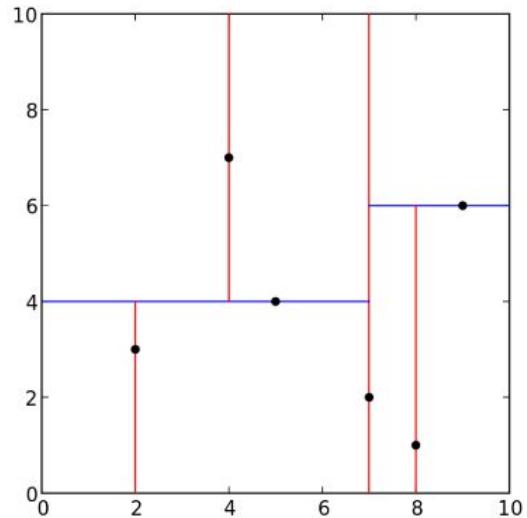
Construcción de un árbol

3. Determine los subespacios izquierdo y derecho. Dividir el hiperplano $x = 7$ divide todo el espacio en dos partes. La parte de $x \leq 7$ es el subespacio izquierdo, que contiene 3 nodos $\{(2,3), (5,4), (4,7)\}$; la otra parte es el subespacio derecho, que contiene 2 nodos $\{(9, 6), (8, 1)\}$.

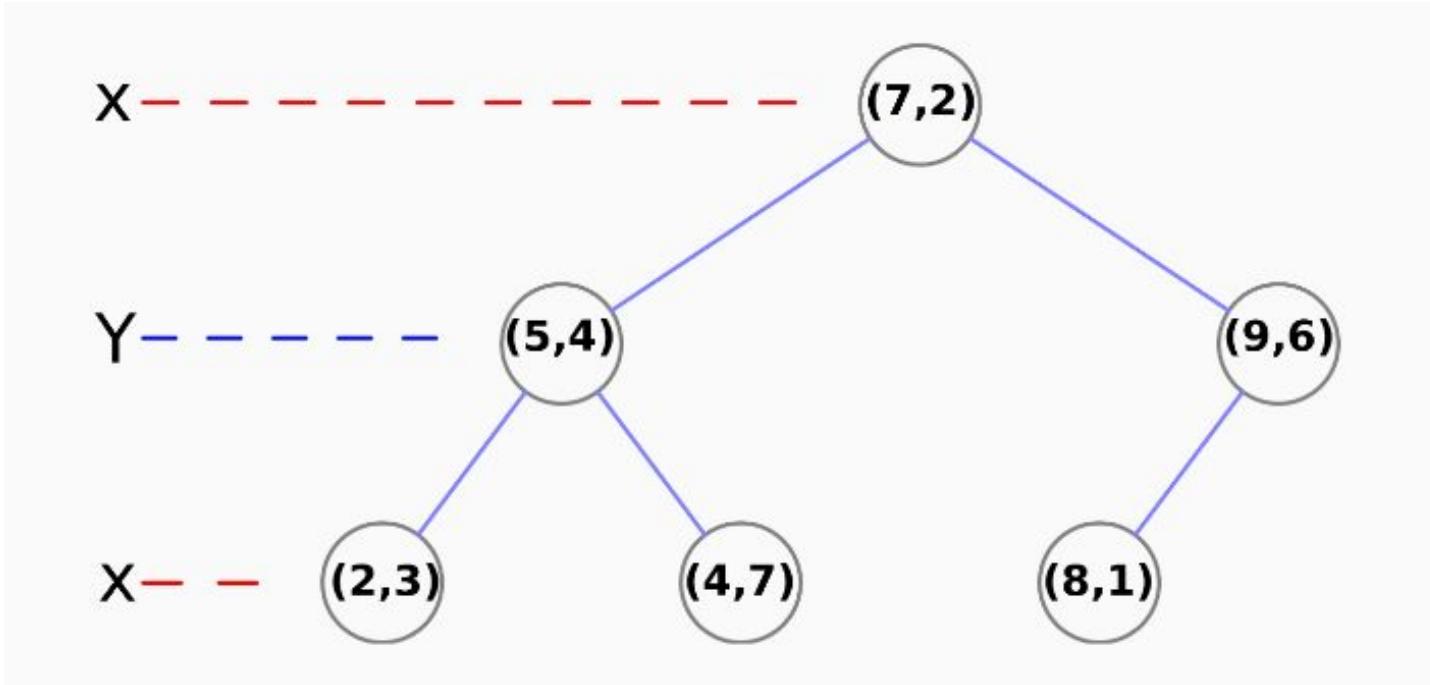


Construcción de un árbol

Como se indica en el algoritmo, la construcción del árbol Kd es un proceso recursivo. El mismo proceso se debe aplicar para los subespacios.



Construcción de un árbol



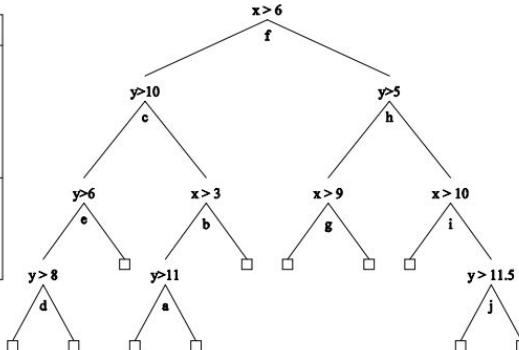
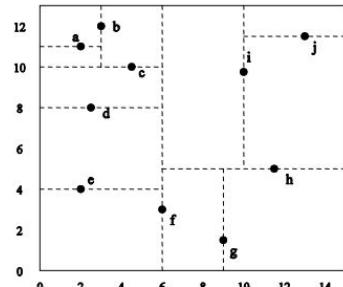
Inserción de puntos al árbol

1. Recorre el árbol a partir de la raíz moviéndose hacia el descendiente correspondiente.
2. Cuando se encuentra el nodo padre el punto a insertar, se agrega a la derecha o a la izquierda, dependiendo del valor de la dimensión de partición.
3. En caso de estar desbalanceado, se aplica un algoritmo de re-balanceo para evitar pérdida de rendimiento.

Búsqueda del vecino más cercano en árboles Kd

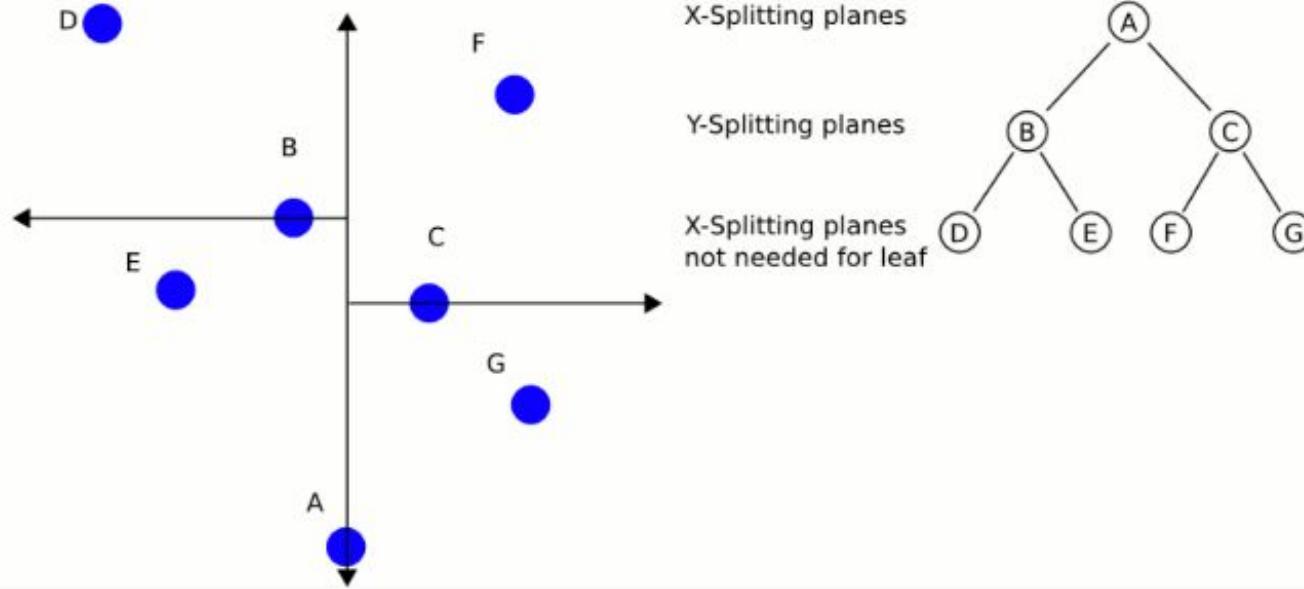
Se recorre el árbol a partir de la raíz y moviéndose hacia el descendiente correspondiente.

- Se mantiene el punto más cercano C_{\min} y quita los nodos del árbol que están más alejados de este.
- Recorre los subárboles/ subespacios restantes.

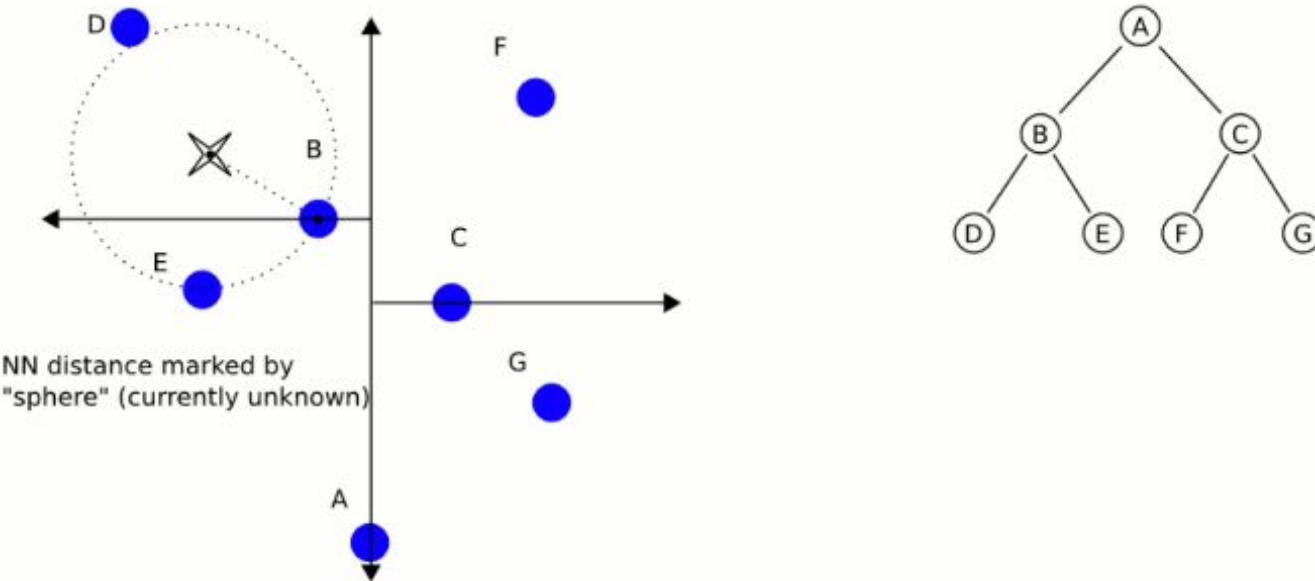


Sufre la maldición de la dimensionalidad

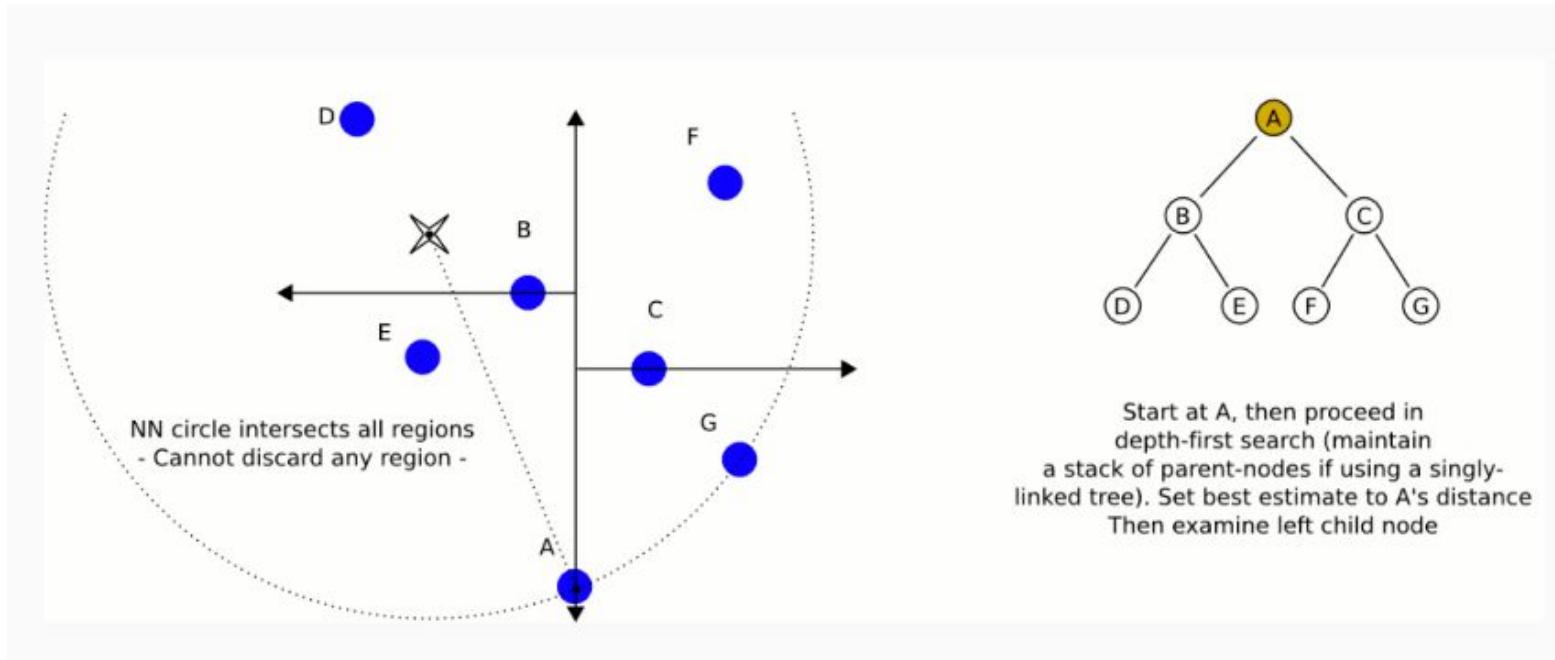
Búsqueda del vecino más cercano en árboles Kd



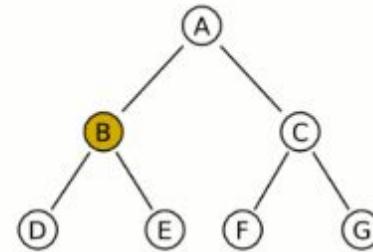
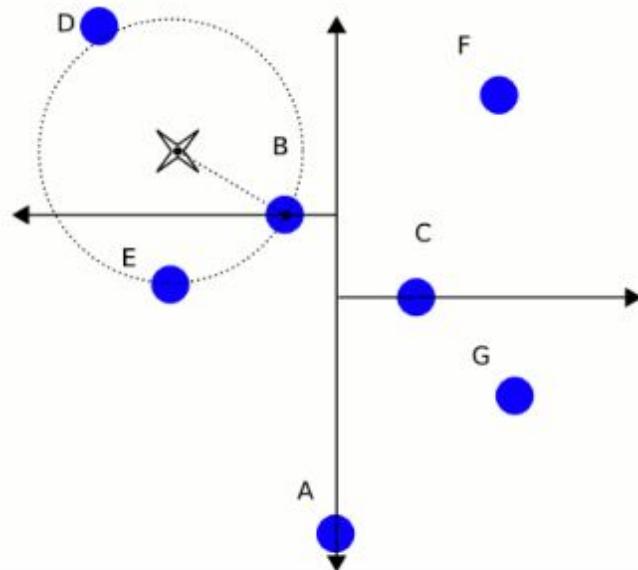
Búsqueda del vecino más cercano en árboles Kd



Búsqueda del vecino más cercano en árboles Kd

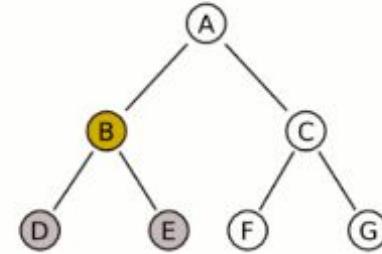
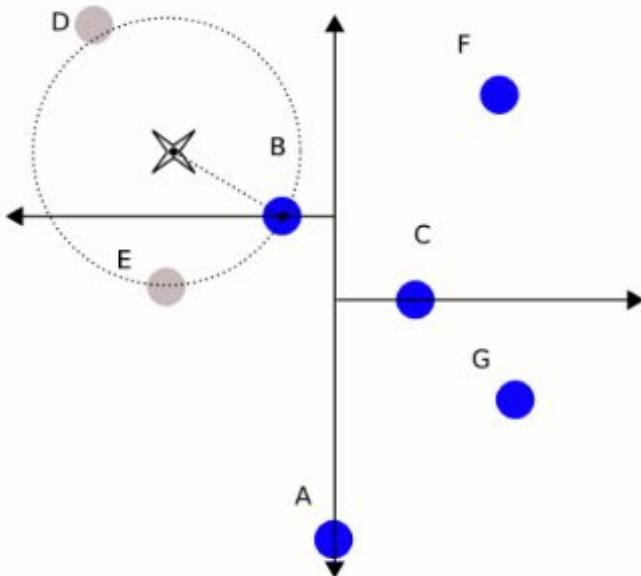


Búsqueda del vecino más cercano en árboles Kd



Calculate B's distance and compare against best estimate
- It is smaller distance, so update best estimate. Examine children (left then right)

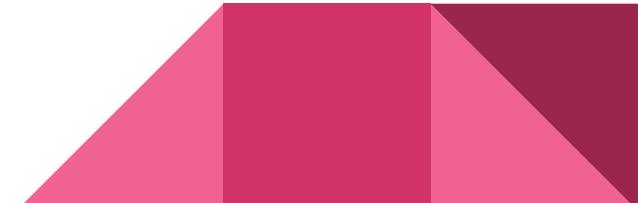
Búsqueda del vecino más cercano en árboles Kd



D & E Discarded as B
(already visited) is closer.
B is the best estimate for B's sub-branch
Proceed back to parent node

Ejercicio

- Construye el árbol binario Kd para el siguiente conjunto de puntos en 2D.
 $\{ (8.3, 3.0), (6.2, 2.4), (0.2, 4.3), (7.5, 1.6), (3.6, 0.2), (2.5, 8.8), (1.7, 5.1) \}$
- Dibuja la partición correspondiente del plano.
- Busca los vecinos más cercanos en X de los siguientes puntos:
 $\{ (9.8, 1.7), (3.3, 9.6), (8.1, 1.2) \}$



Maldición de la dimensionalidad

A medida que aumenta el número de dimensiones, las distancias se vuelven menos discriminativas.

En Machine Learning, el número de dimensiones se puede equiparar al número de variables o características que estemos utilizando.

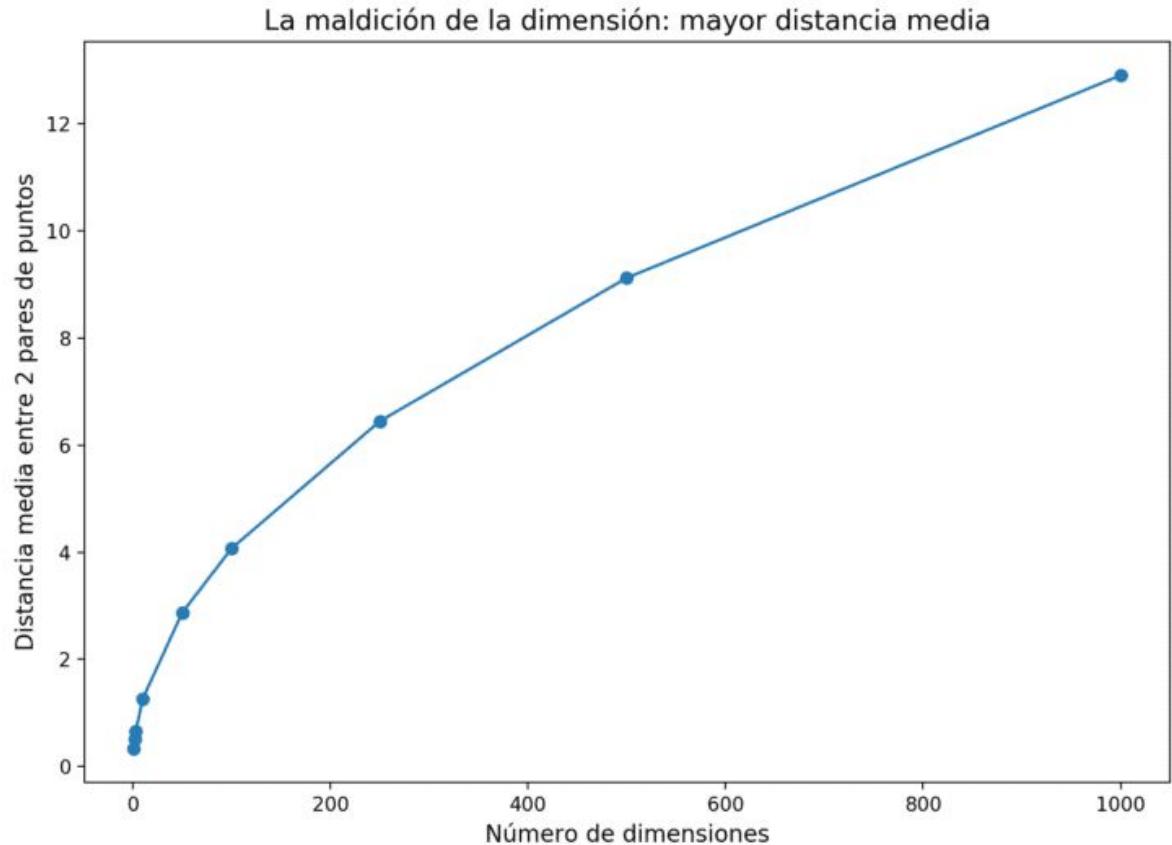
La maldición de la dimensión se «manifiesta» de dos maneras:

- La distancia media entre los datos aumenta con el número de dimensiones
- **La variabilidad de la distancia disminuye exponencialmente con el número de dimensiones**

Maldición de la dimensionalidad

A medida que el número de dimensiones aumenta, la distancia media entre ellos también aumenta.

Esto en sí mismo no es muy problemático para el aprendizaje automático.

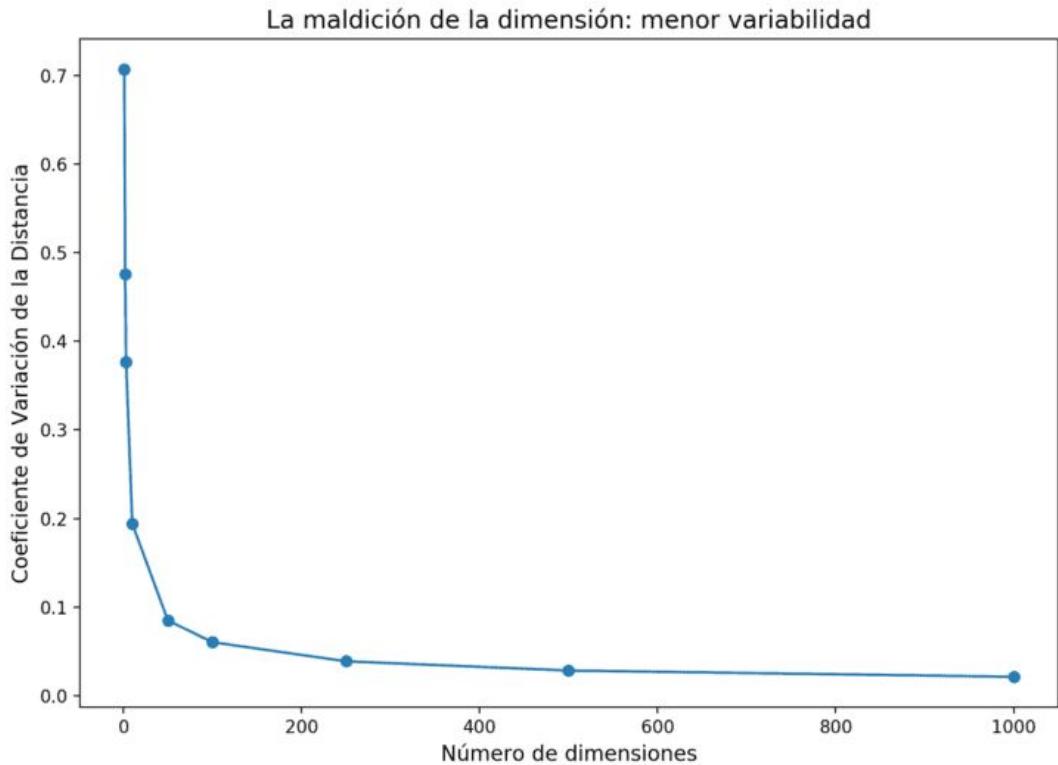


Maldición de la dimensionalidad

Lo verdaderamente importante es que a medida que el número de dimensiones aumenta, la variabilidad de las distancias entre los datos disminuye exponencialmente.

El gráfico muestra el coeficiente de variación para varias dimensiones (hasta 1000). El coeficiente de variación se utiliza para medir la variabilidad.

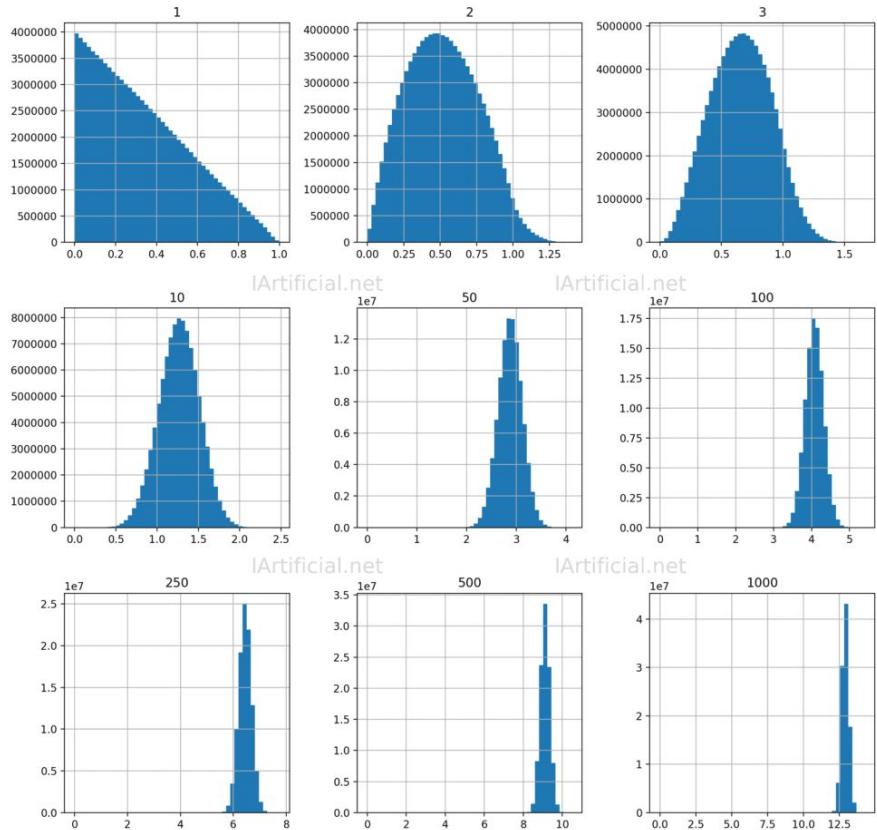
El problema es que cuando hay un gran número de atributos (features), los datos están todos a casi la misma distancia. Es decir, no hay variabilidad entre sus distancias.



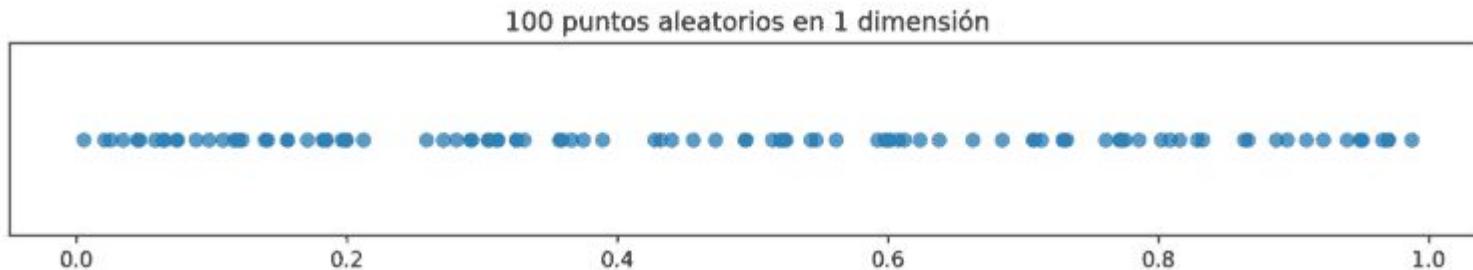
Maldición de la dimensionalidad

La maldición de la dimensión es un concepto un tanto abstracto, por eso es necesario observar lo que sucede cuando se aumenta el número de variables.

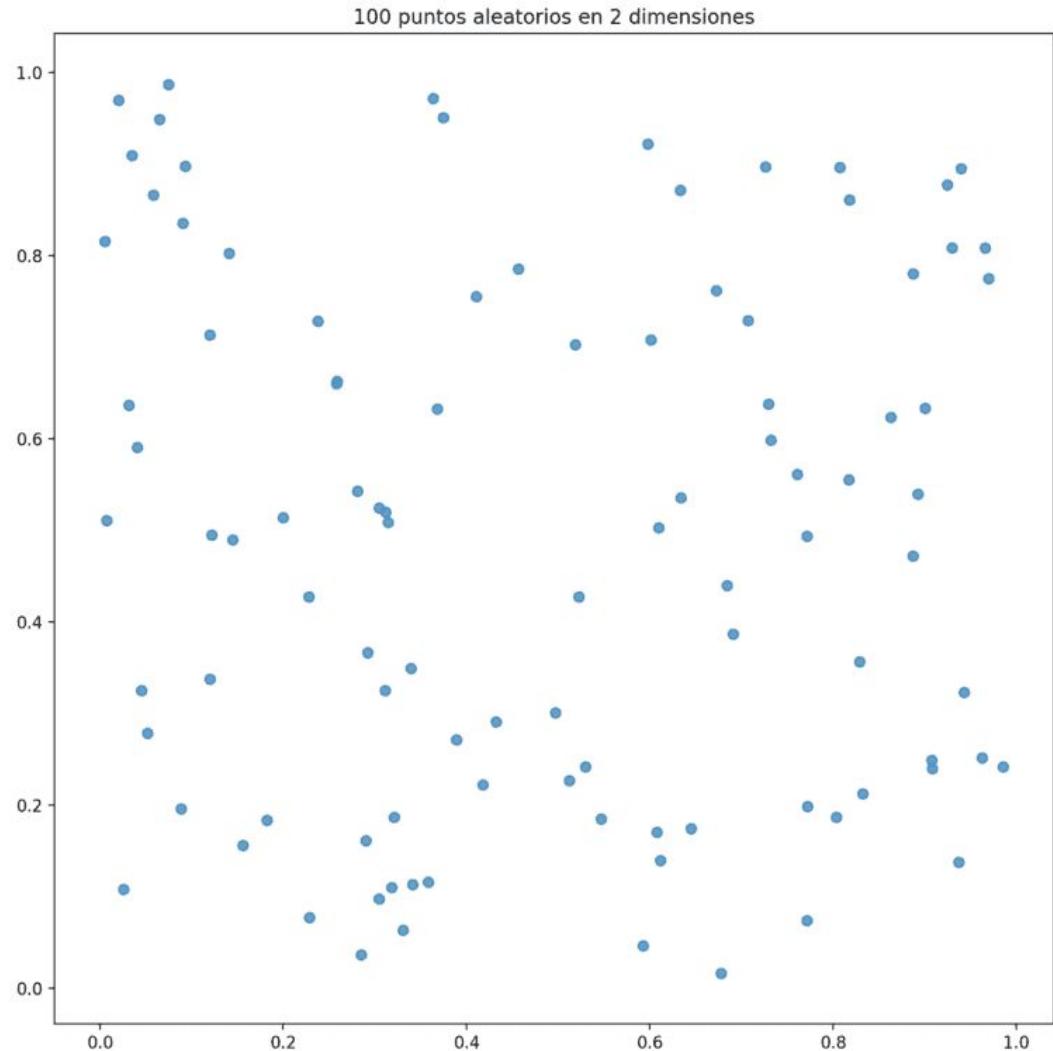
La figura muestra la distribución de distancias euclídeas entre 10000 puntos aleatorios en el rango $[0, 1]$ en varias dimensiones: 1, 2, 3, 10, 50, 100, 250, 500 y 1000.



Maldición de la dimensionalidad



La maldición de la dimensión sólo afecta a las técnicas basadas en distancias.



Maldición de la dimensionalidad

Para mitigar el efecto de la maldición de la dimensión tenemos dos opciones:

- reducir el número de dimensiones
- aumentar la cantidad de datos

Ejercicio

Una vez resuelto el problema de la visualización de las columnas de cada tabla/archivo ¿qué solución propondrías para realizar un modelo predictivo con cantidades masivas de datos?

Similitud en documentos

La forma más efectiva de representar documentos como conjuntos con el fin de identificar documentos léxicamente similares, es construir a partir de del documento, el conjunto de cadenas **cortas** dentro de él.

Entonces los elementos comunes en sus conjuntos serán las oraciones con palabras similares, incluso si aparecen en diferente orden.

Un documento es una cadena de caracteres.

SinContraste

MINION + TIMES

SinContraste

HELVETICA + GOTHAM

SinContraste

CLARENDOON + ROCKWELL

ConContraste

MINION + HELVETICA

ConContraste

HELVETICA + DIDOT

ConContraste

CLARENDOON + DIN

Similitud en documentos

Podemos tomar un conjunto de documentos:

d1: Ella toma café y él toma mate

d2: Ella toma notas mientras toma café

Bolsas de palabras

d1: {café, ella, él, mate, toma, y}

d2: {café, ella, mientras, notas, toma}

Matriz documento-término

	café	ella	él	mate	mientras	notas	toma	y
d1	1	1	1	1	0	0	1	1
d2	1	1	0	0	1	1	1	0

Similitud en documentos

Si quitamos stopwords

d_1 Ella $\underbrace{\text{toma}}_{w_4}$ $\underbrace{\text{café}}_{w_1}$ y él $\underbrace{\text{toma}}_{w_4}$ $\underbrace{\text{mate}}_{w_2}$

d_2 Ella $\underbrace{\text{toma}}_{w_4}$ $\underbrace{\text{notas}}_{w_3}$ mientras $\underbrace{\text{toma}}_{w_4}$ $\underbrace{\text{café}}_{w_1}$

Matriz documento-término

	café	mate	notas	toma
d1	1	1	1	1
d2	1	1	0	0

Bolsas de palabras sin stopwords

$d_1: \{1,2,4\}$

$d_2: \{1,3,4\}$

Similitud en documentos

Matriz documento-término sin stopwords con frecuencias

	café	mate	notas	toma
d1	1	1	1	1
d2	1	1	0	0

Resolver el problema de similitud

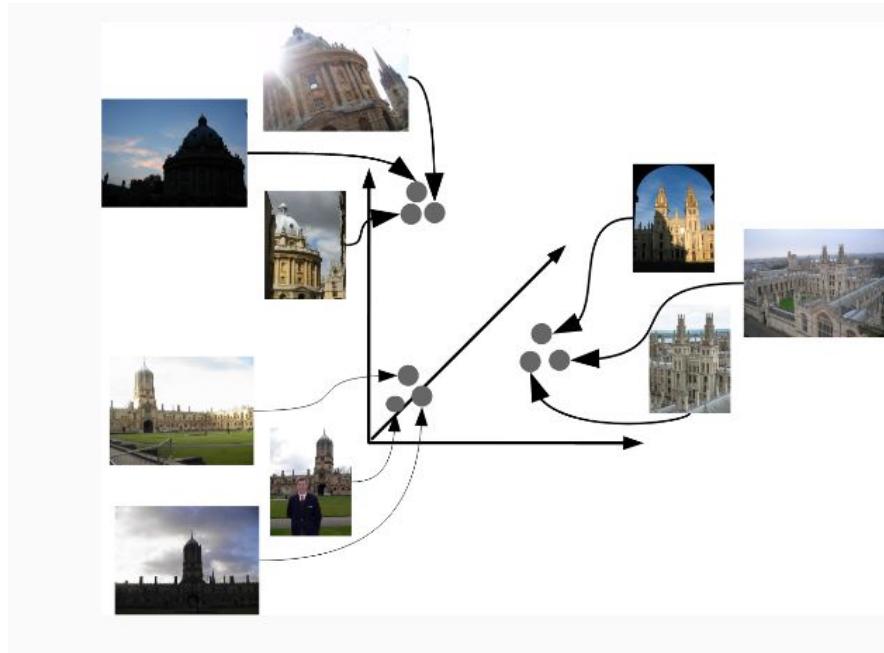
Bolsas de palabras sin stopwords con frecuencia

d1: {1,2,4,4}

d2: {1,3,4,4}

Similitud de imágenes

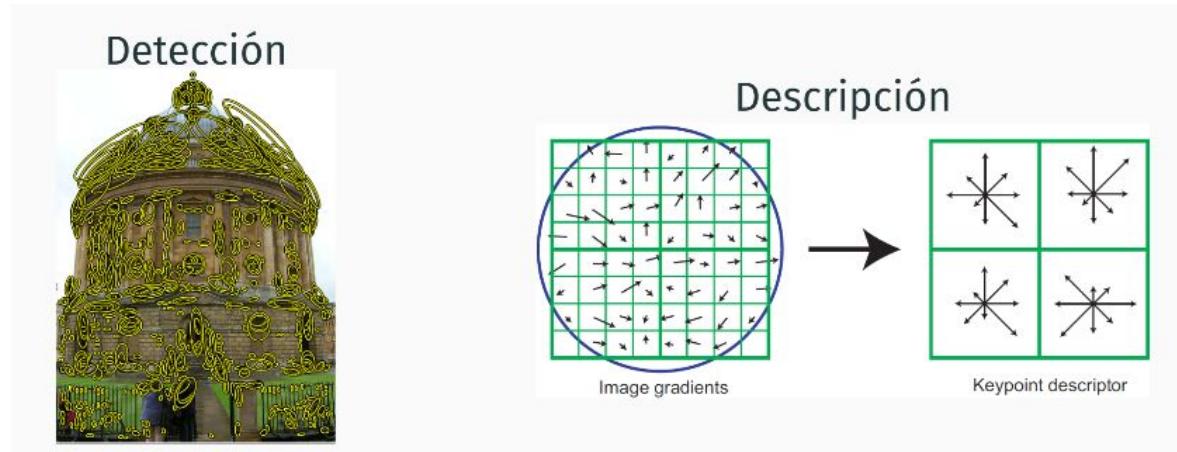
Buscamos mapear las imágenes a una representación compacta, discriminatoria, descriptiva, robusta y rápida de obtener



Similitud de imágenes

Dos tareas fundamentales:

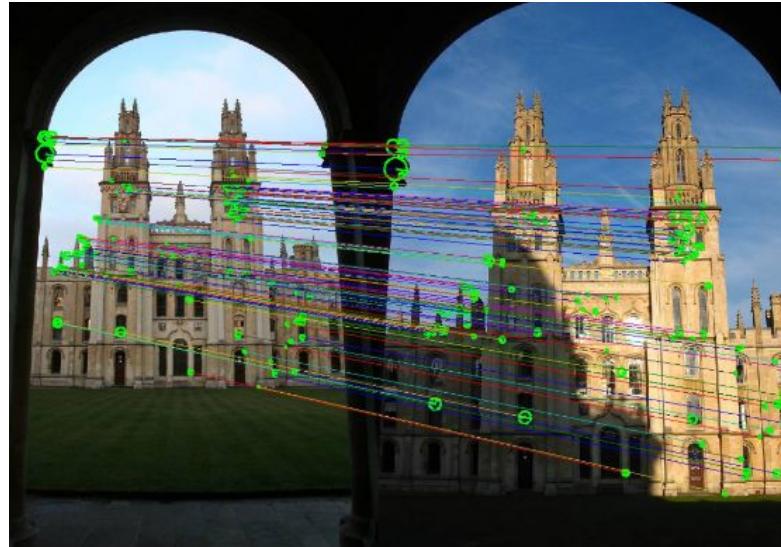
1. Detección de regiones de interés.
2. Descripción de cada región.



Similitud de imágenes

Imagen -> conjunto de vectores característicos

Buscar características similares



¿Cómo buscamos imágenes/ documentos similares?

- Filas y columnas de la matriz documento-término son usualmente dispersas y se representan por conjuntos o bolsas.
- Se deben comparar solo las que comparten al menos una característica/palabra.
- Ordena por valor de distancia o similitud.

Índice Inverso

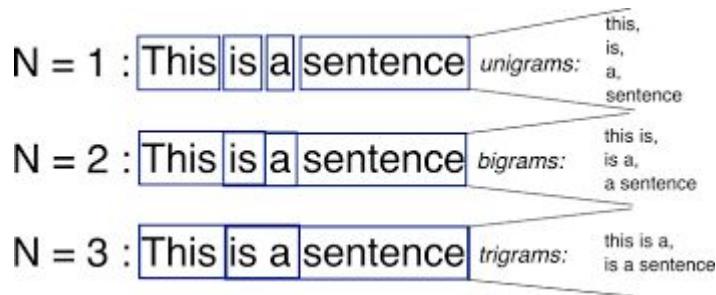
Búsqueda por índice inverso.

1. Recupera los conjuntos o bolsas de documentos donde ocurren las palabras de D.
2. Calcula la distancia o similitud entre D y cada elemento en la lista.
3. Ordena d de acuerdo a las distancias o similitudes calculadas



N-gramas

Son una secuencia de **n** objetos que pueden ser símbolos o palabras.



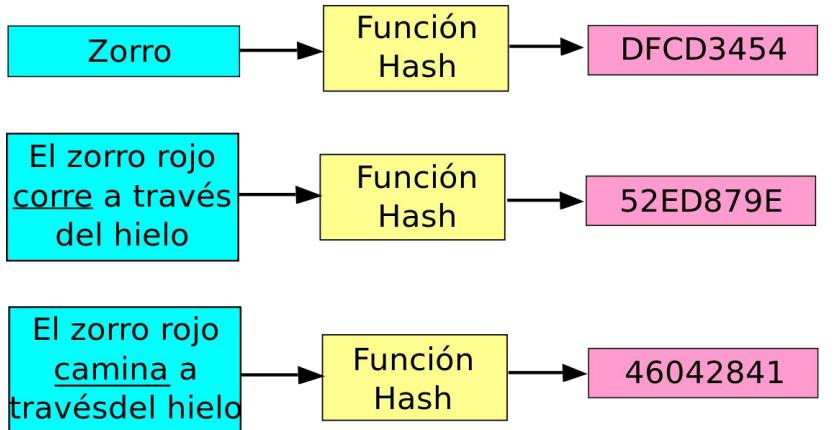
Ejemplo: Ella toma café y él toma mate

Ejercicio: “Los negocios del futuro deben autocomprender y autodecidir en tiempo real”

Funciones hash sensibles a la localidad

Funciones Hash

Entrada



Valor Hash

Una función criptográfica hash, es un algoritmo matemático que transforma cualquier bloque arbitrario de datos en una nueva serie de caracteres con una longitud fija.

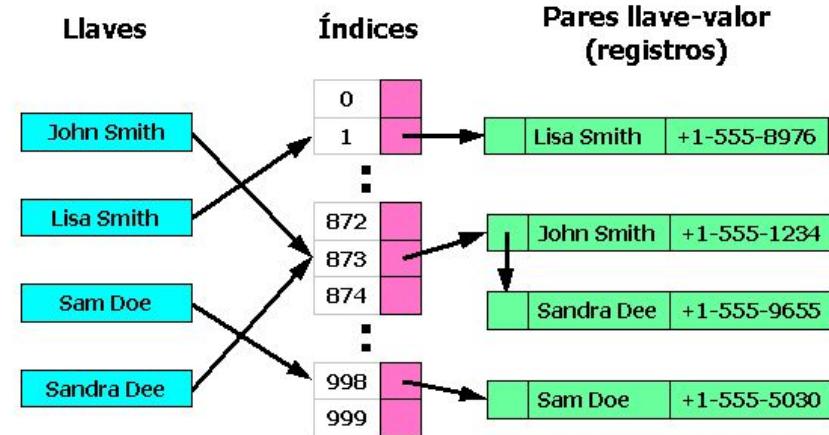
Independientemente de la longitud de los datos de entrada, el valor hash de salida tendrá siempre la misma longitud.

Tablas Hash

Una tabla o mapa hash es una estructura de datos que asocia llaves o claves con valores. La operación principal que soporta es la **búsqueda**.

Permite el acceso a los elementos almacenados a partir de una clave generada usando el nombre o un ID.

Funciona transformando la clave con una función hash en un hash*.



* un número que la tabla hash utiliza para localizar el valor deseado.

Propiedades de las funciones Hash

Es deseable que las funciones hash h tengan las siguientes propiedades.

- Uniformidad: Si todos los índices son igualmente probables.

$$P_{h \in \mathcal{H}}[h(x) = i] = \frac{1}{m}, \text{ para cada } x \text{ y cada } i$$

- Universal: Si la familia de funciones minimiza el número de colisiones.

$$P_{h \in \mathcal{H}}[h(x) = h(y)] \leq \frac{1}{m}, \text{ para cada } x \neq y$$

- Casi-Universal

$$P_{h \in \mathcal{H}}[h(x) = h(y)] \leq \frac{2}{m}, \text{ para cada } x \neq y$$

Tablas Hash

Las tablas hash se suelen implementar sobre arreglos de una dimensión, aunque se pueden hacer implementación multi-dimensiones basadas en varias claves.

Las tablas hash también permiten operaciones de inserción y eliminación.

- Búsqueda: Regresa una “cubeta” correspondiente al índice $h(x)$ en la tabla.
- Insertar: Agrega un objeto x , este se mapea a la cubeta $h(x)$.
- Eliminar: Quita el objeto x de la cubeta $h(x)$.

Familia H por método de División

Comúnmente se busca diseñar funciones hash h que sean independientes de los objetos a almacenar.

Un método general para crear funciones hash se conoce como el *método de división* y consiste en usar el residuo de la división del objeto y el tamaño de la tabla m , esto es,

$$h(x) = x \mod m$$

Es conveniente usar un número primo para m .

¿Cómo dirías que es la probabilidad de que en un grupo de 23 personas dos de ellas celebren su cumpleaños el mismo día?

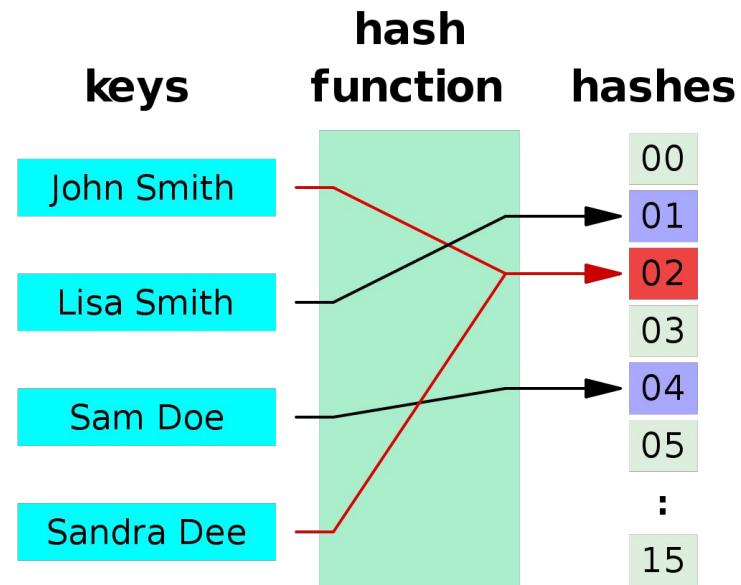
La paradoja del cumpleaños establece que si hay 23 personas reunidas hay una probabilidad del 50,7% de que al menos dos personas de ellas cumplan años el mismo día. Para 60 o más personas la probabilidad es mayor del 99%. Obviamente es casi del 100% para 366 personas (teniendo en cuenta los años bisiestos).

Colisiones

Una colisión hash es una situación que se produce cuando dos entradas distintas a una función hash producen la misma salida.

Al diseñar una función hash se busca que sea rápida y fácil de calcular, pero a la vez que tenga la mínima cantidad de colisiones.

En muchas aplicaciones queremos reducir colisiones de objetos diferentes. Sin embargo, no es posible evitarlas por completo, por lo que es necesario emplear estrategias para lidiar con ellas.



Para evitar colisiones podemos aumentar el tamaño de la tabla. Suponiendo que la función es uniforme.

Podemos calcular la probabilidad de que existan colisiones como el complemento de la probabilidad que no haya colisión.

$$\begin{aligned} \Pr[h(x) = h(y)] &= 1 - P[h(x) \neq h(y)] \\ &= 1 - \frac{m}{m} \times \frac{m-1}{m} \times \cdots \times \frac{m-n+1}{m} \end{aligned}$$

Y aproximarla de la siguiente forma:

$$P[h(x) = h(y)] \approx 1 - e^{\frac{-n^2}{2m}}$$

Colisiones

Asimismo, podemos calcular el número esperado de colisiones para valores dados de y de la siguiente manera:

$$\mathbb{E}[\text{colisiones}] = n - m \cdot \left(1 - \frac{m-1}{m}\right)^n$$

Encadenamiento

Una de las estrategias más comunes para resolver colisiones es el encadenamiento y consiste en que cada cubeta de la tabla almacena una lista enlazada con todos los objetos que han sido mapeados al índice correspondiente.

- Búsqueda: Examina si un objeto x se encuentra en la lista enlazada de la cubeta correspondiente a $h(x)$.
- Insertar: Agrega x , a la lista enlazada de la cubeta correspondiente $h(x)$.
- Eliminar: Busca el objeto x de la cubeta $h(x)$ y lo elimina de la lista enlazada.

Direccionamiento abierto

Otro método común para resolver colisiones.

Se sondean cubetas hasta encontrar alguna libre o la que contiene el objeto que se busca. Si no hay ninguna disponible, decimos que la tabla está llena y es necesario aumentar su tamaño para almacenar más objetos.

Hay varias formas de realizar el sondeo: lineal, cuadrático o usando el doble mapeo.

Sondeo lineal:

$$h(x, i) = (h'(x) + i) \mod m$$

Familias por método de multiplicación

Otra estrategia para diseñar funciones *hash* es el de multiplicación. En este método primero se multiplica el objeto x por una constante $a \in (0, 1)$ y se extrae la parte fraccional de $a \cdot x$. Posteriormente, se multiplica este valor por m y se toma la función piso $\lfloor \cdot \rfloor$

$$h(x) = \lfloor m \cdot (x \cdot a \mod 1) \rfloor$$

donde a es una constante.

Aunque a puede tomar cualquier valor en el intervalo $(0, 1)$, hay algunos valores que funcionan mejor que otros. Donald Knuth recomienda $a \approx \frac{\sqrt{5}-1}{2}$.

Familias de funciones Hash Universal

Una familia de funciones universal es

$$\mathcal{H}_{a,b}(x) = ((a \cdot x + b) \mod p) \mod m$$

donde $p \geq n$ es un número primo muy grande, $a \in \{1, 2, \dots, p-1\}$ y $b \in \{0, 1, 2, \dots, p-1\}$

Funciones hash para cadenas

Si deseamos almacenar cadenas de símbolos, es necesario convertirla a un solo número. La forma más simple es sumar sus códigos pero esto genera colisiones en palabras que usan los mismos símbolo. Una alternativa es la suma de los códigos multiplicados por un constante elevada a la posición:

$$f(s) = s_1 \cdot g^1 + s_2 \cdot g^2 + \cdots + s_d \cdot g^d$$

Ejercicio

Realiza lo siguiente:

Implementa una tabla que realice sondeo cuadrático. Elige un número primo grande para m y $c_1 = c_2 = 1$ y prueba la tabla con el conjunto de 200 números.

$$h(x, i) = (h'(x) + c_1 \cdot i + c_2 \cdot i^2) \mod m$$

Desarrolla un programa que cuente el número de ocurrencias de todas las subcadenas de longitud entre 5 a 10 (sin considerar espacios en blanco ni signos de admiración/interrogación ni caracteres especiales) usando funciones y tablas hash.

Funciones hash sensibles a la localidad

Funciones hash sensibles a la localidad

En la práctica cuando trabajemos con datos masivos, hasta las tareas simples se vuelven complicadas.

La operación de consulta es la más común.

Para conjuntos de datos pequeños las consultas pueden resolverse con búsqueda lineal para datos masivos, el costo de tiempo es muy grande; por lo anterior, se requieren técnicas similares a **índices** para acelerar el proceso de búsqueda.

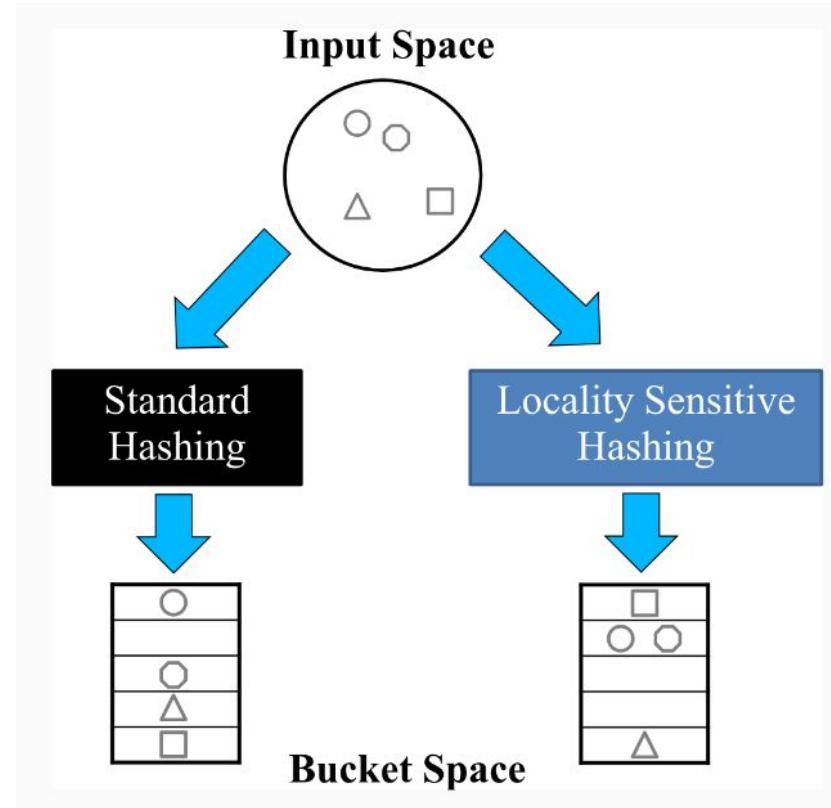
Funciones hash sensibles a la localidad

El algoritmo tradicional de hash, crea una tabla hash a través de una función hash. La clave es encontrar una función hash adecuada y asignar los datos originales al segmento correspondiente evitando lo más posible las colisiones.

El algoritmo LSH se basa en la suposición de que si dos datos son similares en el espacio de datos original, también tendrán un alto grado de similitud después de ser mapeados por la función hash; por el contrario, si no son similares, luego Todavía no tienen similitudes después del mapeo.

El hash sensible a la localidad es un procedimiento para encontrar pares similares en un gran conjunto de datos.

Funciones hash sensibles a la localidad



Ejemplo

Supongamos que tenemos un conjunto de datos que consta de documentos (artículos de noticias, páginas web, etc).

- En la primer parte del algoritmo, éstos son realmente documentos.
- En la segunda mitad, un documento será un vector D-dimensional.

Hay dos posibles problemas a los que nos podemos enfrentar:

1. Tenemos una cantidad razonable de documentos, pero los documentos son muy largos.
2. Tienen documentos de una longitud razonable, pero demasiados.

Ejemplo

Necesitamos definir una métrica de similitud.

Entonces dada una lista de N documentos, representamos por un vector binario de dimensión D , encontremos todos los pares de documentos con similitud Jaccard por encima de algún umbral t .

Si tenemos una matriz muy grande, ¿cómo resolvemos el problema sin escribir explícitamente esa matriz (N, D) ?

Minhasing

Método inventado por Andrei Broder en 1997. Originalmente se creó para detectar y eliminar páginas web duplicadas.

- Genera una permutación aleatoria del conjunto universo.
- Asigna a cada conjunto su primer elemento bajo la permutación.

$$h(\mathcal{C}^{(1)}) = \min(\pi(\mathcal{C}^{(1)}))$$

La probabilidad de que la función minihash para una permutación de filas produzca el mismo valor para dos conjuntos es igual a la similitud de Jaccard para esos conjuntos.

$$P[h(\mathcal{C}^{(1)}) = h(\mathcal{C}^{(2)})] = \frac{|\mathcal{C}^{(1)} \cap \mathcal{C}^{(2)}|}{|\mathcal{C}^{(1)} \cup \mathcal{C}^{(2)}|} \in [0, 1]$$

Minhasing

No es factible permutar explícitamente una matriz característica grande.

Una permutación aleatoria de miles de millones de documentos llevaría mucho tiempo y la clasificación necesaria de esas filas, aún más.

Es posible simular el efecto de una permutación aleatoria mediante una función hash aleatoria que asigna números de fila a tantos cubos como filas hay.

1. Calculamos h_1, \dots, h_n
2. Para cada columna,
 - si tenemos cero no hacemos nada
 - si tenemos uno, asignamos el valor más pequeño que se tenga de los valores de h

Ejercicio

¿Qué harías si analizas tus datos con los que vas a realizar un modelo y al remover outliers te das cuenta de que tienes más outliers, pero son demasiados?

Minhasing

Tuplas de valores Minihash

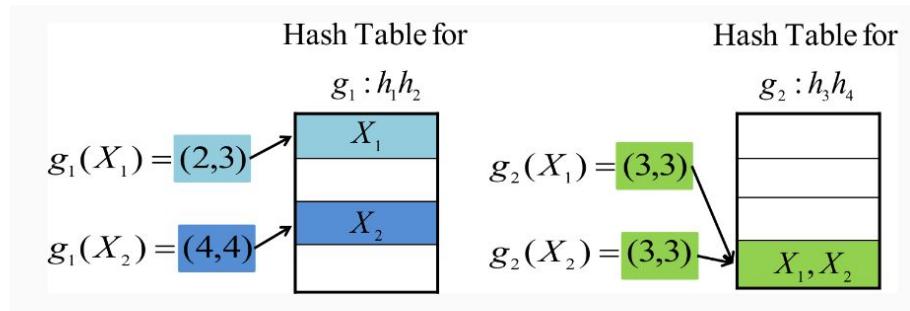
$$g_1(\mathcal{C}^{(1)}) = (h_1(\mathcal{C}^{(1)}), h_2(\mathcal{C}^{(1)}), \dots, h_r(\mathcal{C}^{(1)}))$$

$$g_2(\mathcal{C}^{(1)}) = (h_{r+1}(\mathcal{C}^{(1)}), h_{r+2}(\mathcal{C}^{(1)}), \dots, h_{2 \cdot r}(\mathcal{C}^{(1)}))$$

...

$$g_l(\mathcal{C}^{(1)}) = (h_{(l-1) \cdot r + 1}(\mathcal{C}^{(1)}), h_{(l-1) \cdot r + 2}(\mathcal{C}^{(1)}), \dots, h_{l \cdot r}(\mathcal{C}^{(1)}))$$

Conjuntos con tuplas idénticas se almacenan en el mismo registro.



Minhasing

- La probabilidad de que dos conjuntos sean idénticos es:

$$P[g_k(\mathcal{C}^{(1)}) = g_k(\mathcal{C}^{(2)})] = sim(\mathcal{C}^{(1)}, \mathcal{C}^{(2)})^r$$

- La probabilidad de que no tengan ninguna tupla idéntica de l posibles es:

$$P[g_k(\mathcal{C}^{(1)}) \neq g_k(\mathcal{C}^{(2)})] = (1 - sim(\mathcal{C}^{(1)}, \mathcal{C}^{(2)})^r)^l, \forall k$$

- Por lo tanto la probabilidad de que dos conjuntos tengan al menos una tupla idéntica es:

$$P_{colision}[\mathcal{C}^{(1)}, \mathcal{C}^{(2)}] = 1 - (1 - sim(\mathcal{C}^{(1)}, \mathcal{C}^{(2)})^r)^l$$

Minhash con Pyspark

¿Qué tenemos que saber antes implementar Minhash en PySpark?

Vectores dispersos

Los vectores dispersos pueden ayudarnos a representar matrices dispersas. Una matriz dispersa es una matriz donde la mayoría de sus entradas son 0, es por esto que, para ahorrar espacio, es más eficiente especificar, por renglon, que columnas el valor no es cero y que valor tiene.

Por ejemplo, un renglon de una matriz dispersa puede ser el siguiente

(1, 0, 0, 0, 0, 3, 0, 0, 2, 0)

Notemos que este vector puede ser definido especificando los siguientes parámetros:

1. El tamaño del vector.
2. Las entradas del vector que son distintas de cero (Índices desde cero).
3. Los valores correspondientes a dichas entradas.

CountVectorizer

CountVectorizer es una clase que nos ayuda a extraer el vocabulario de un conjunto de documentos y tambien nos ayuda a generar representaciones dispersas sobre el vocabulario.

La clase CountVectorizer nos ayuda a encontrar la matriz de representación convertida en vectores dispersos dados los documentos