Machine Learning en Spark con MLlib

Rafael Caballero Roldán

+ Spark ML lib

Spark

Una breve introducción a Spark

Introducción

Spark SQL y SparkML

Vectores locales

Matrices locales
Matrices distribuidas

¿Qué es Spark?

El lema de Spark: "*Análisis rápido y distribuido de Big Data*"

Rápido

Análisis rápido y eficiente de Big Data

Memoria

Los computes de hacen en memoria, evitando el cuello de botella de acceso a disco

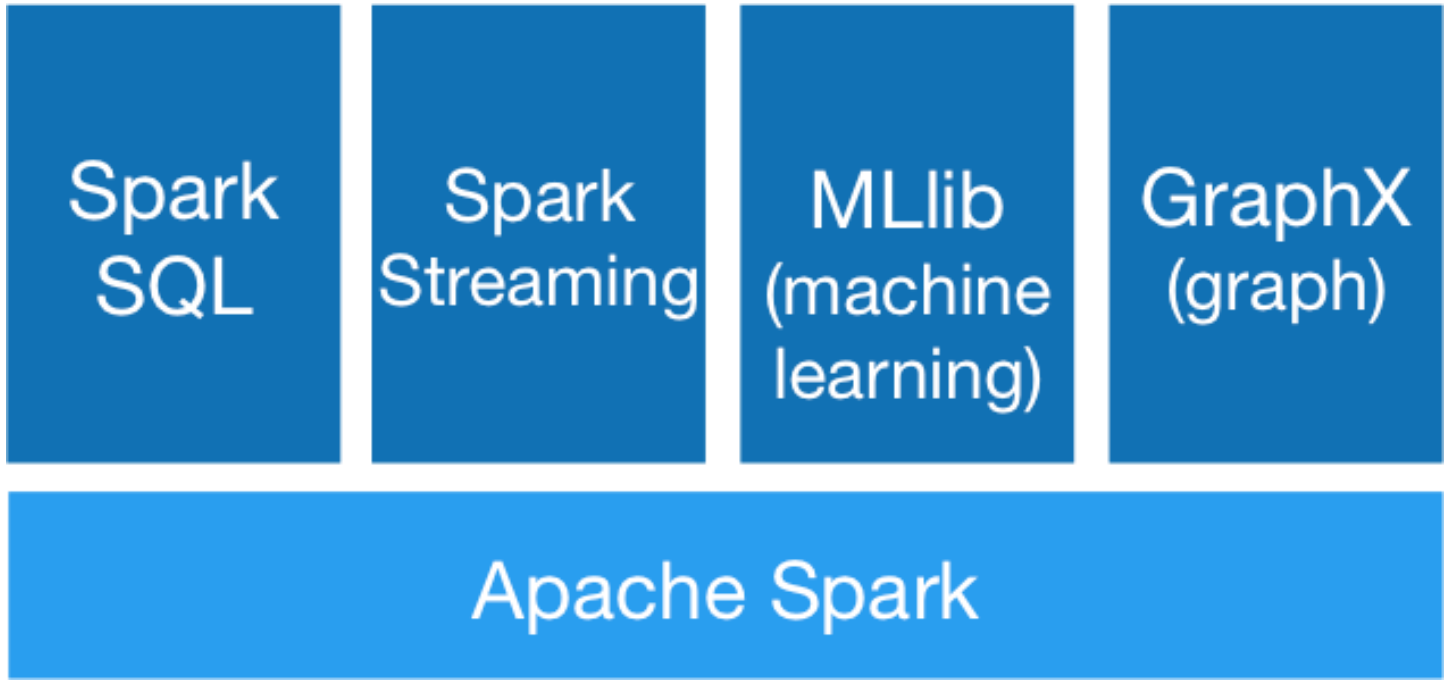
Distribuido

Al contrario que otros entornos (R, Mathlab) está pensado para funcionar de forma eficiente en entornos distribuidos

Análisis

Incorpora herramientas de análisis como MLlib

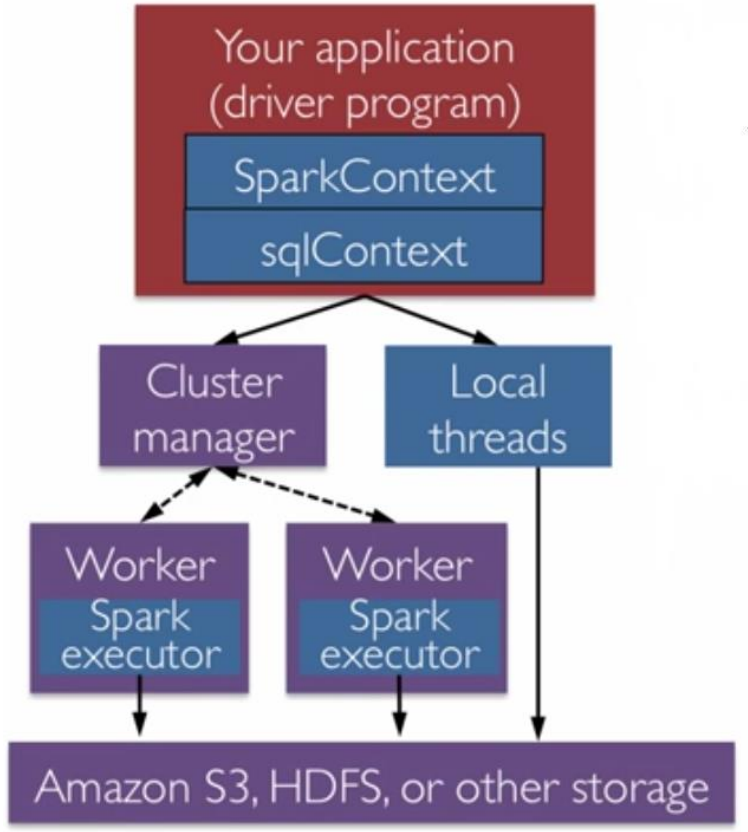
Componentes en Spark



Componentes en Spark

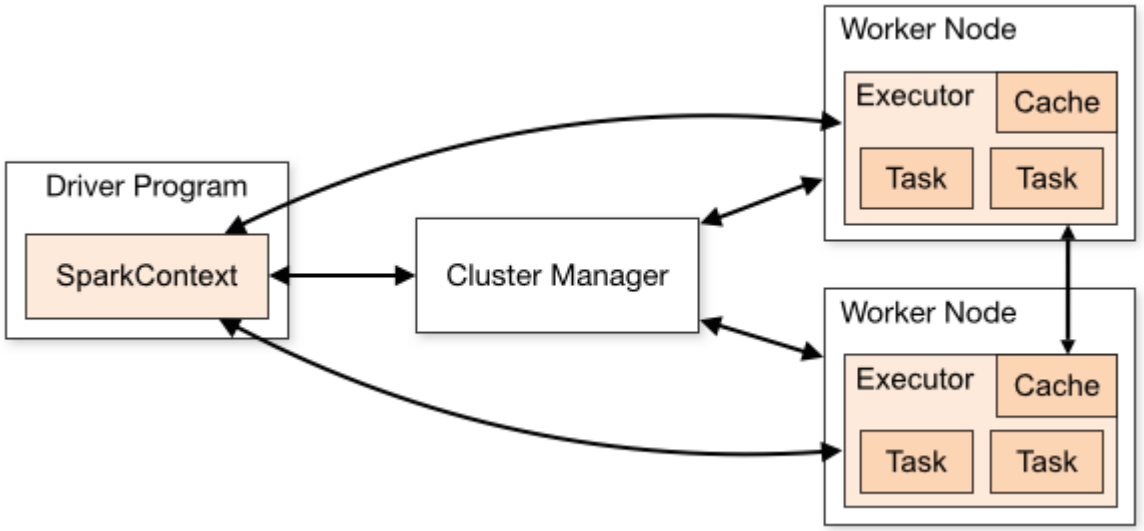
En realidad un programa se divide en ejecución en dos: driver y worker

- El driver se ejecuta en una sola máquina
- El worker se distribuye por las máquinas
- Si estamos en modo local, los worker se distribuyen por threads de la máquina
- Los Data Frames se reparten por todos los workers
- La interacción la lleva a cabo el SparkContext, que define la conexión entre el programa y el clúster



Componentes en Spark (II)

Arquitectura en clúster



- Cada *SparkContext* tiene sus propios procesos JVM (aislamiento para bien y para mal)
- El SparkContext también debe aceptar conexiones de los drivers
- A cada driver se le asigna una web UI, habitualmente en el Puerto 4040. Se puede monitoriar el comportamiento en http://<driver-node>:4040

RDDs y Dataframes

Dos estructuras de datos básicas en Spark

Signficado

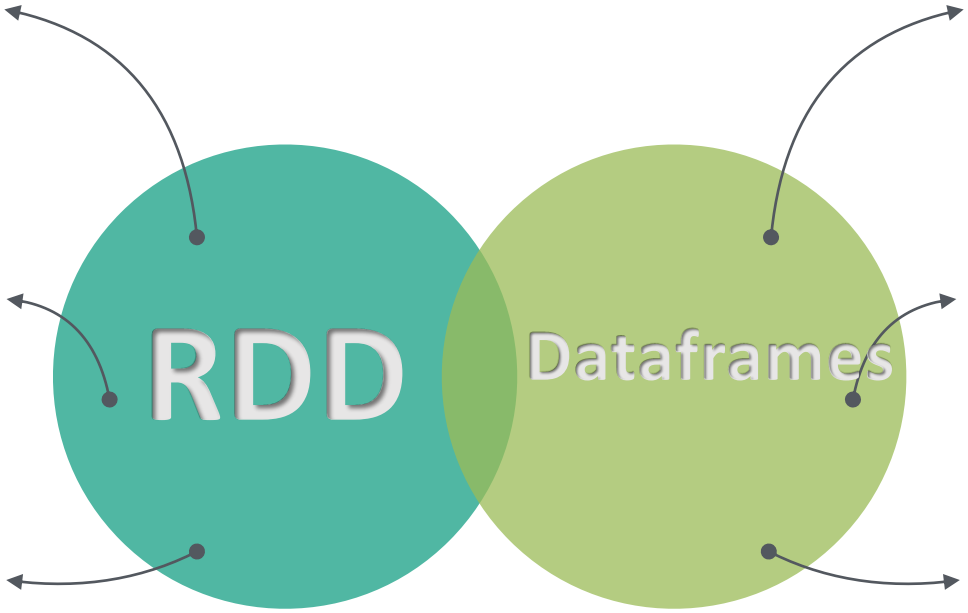
Los Resilient Distributed Dataset (*RDD*) son colecciones distribuidas de datos

Programación funcional

Son inmutables; solo se pueden usar para generar otras nuevas con acciones y y transformaciones

Pereza

Las transformaciones se ejecutan cuando son demandadas por una acción para obtener un valor



RDD en tabla

Se trata del mismo principio, pero dotando de estructura de tabla a colecciones RDD

Columnas

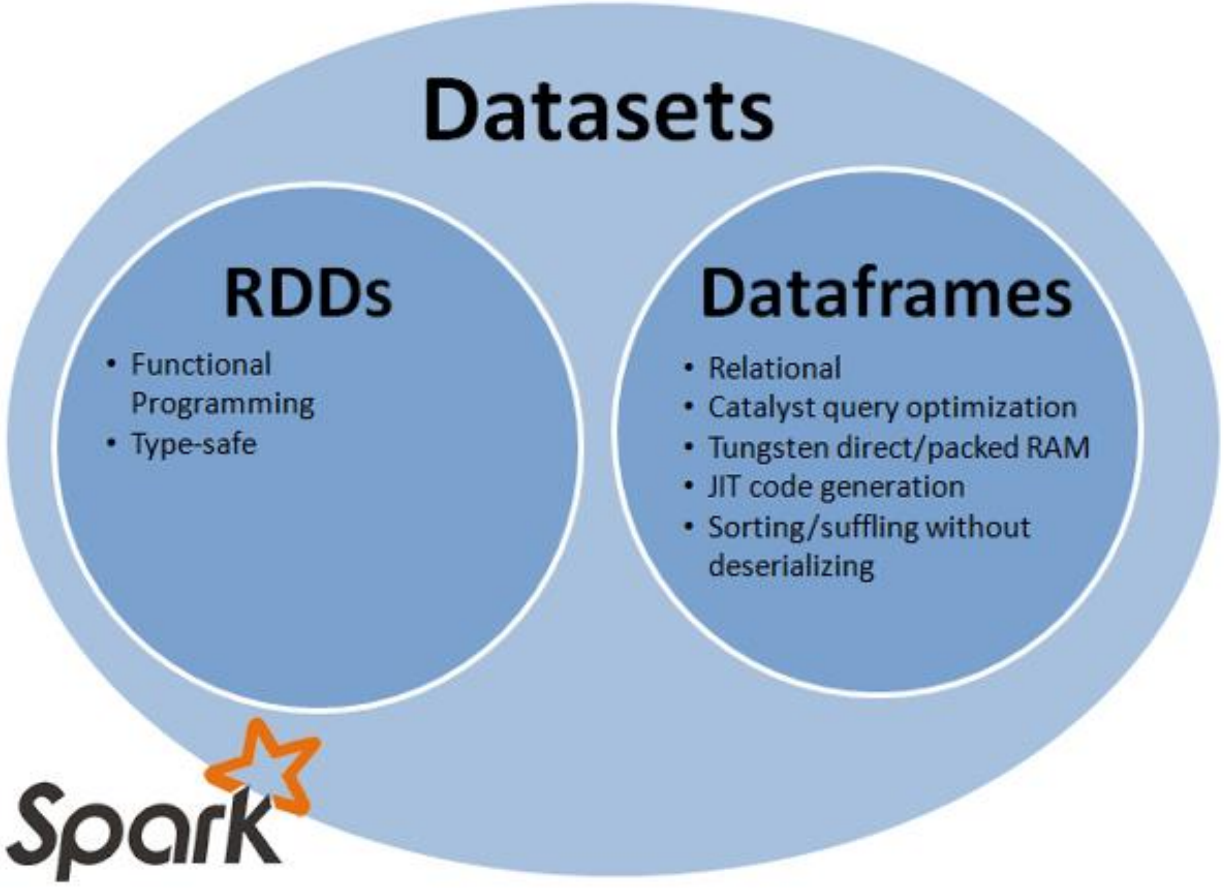
Los datasframes se basan en columnas con nombres

Eficiencia

Al tener esquema no se envían solo los datos, el esquema solo se comparte inicialmente

Datasets

La nueva estructura en 1.6



Evolución estructuras en Spark

Resilient Distributed Dataset	DataFrames	Datasets	Datasets 2.0
RDD			
Acciones	Tablas	Combina ambos modelos	Dataset = Dataset of Row
Transformaciones	Mismo esquema	Flexibilida RDD	Abstracción de Structured Streaming
Funcional	Estilo relacional	Eficiencia de DataFrames	SparkSession reemplaza SparkContext
Pereza	Similar a R	Muy orientado a Scala	
Poca eficiencia	Efiiciente		
Spark 1.0	Spark 1.3	Spark 1.6	Spark 2.0

Ejemplo

Programa Java que lee un fichero de texto y cuenta el número de líneas que contienen la letra ‘a’.

Introducimos:

- los objetos SparkSession
- la transformación filter
- la acción count

Transformacionnes y acciones

Transformaciones

map(*func*)	Devuelve un nuevo conjunto distribuido obtenido al aplicar la función a cada elemento del conjunto original
filter(*func*)	Devuelve el conjunto que se obtiene al eleminar aquellos elementos del conjunto original para los que la función devuelve false
flatMap(*func*)	Como map, pero la función puede devolver varios elementos por cada uno de entrada
Otras	mapPartition, union, intersection, sample,distinct ….

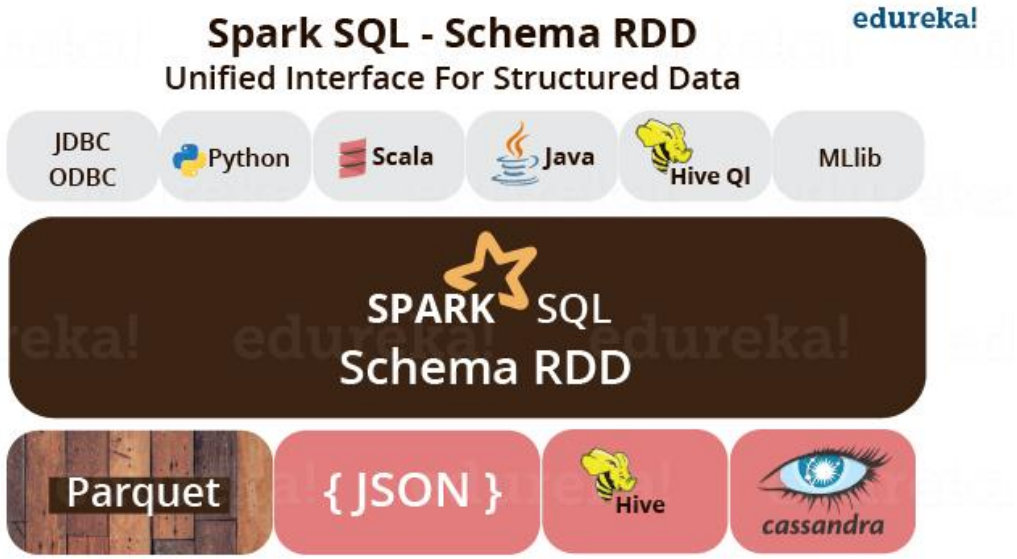
Acciones

reduce(*func*)	Combina elementos. Func recibe dos elementos y devuelve uno
collect()	Devuelve todos los elementos como un array
count()	Número de elementos en el conjunto
Otras	first(), take(n), saveAsTextFile(file), foreach

Spark SQL

Spark incluye un componente que permite tratar datos estructurados por columnas

- ✓ Los RDD admiten pocas operaciones básicas
- ✓ DataFrame: los datos se organizan en columnas
- ✓ Las columnas tienen tipos fijos
- ✓ Esto permite definer un “esquema”
- ✓ Spark permite hacer consultas sobre los datasets utilizando un lenguaje similar a SQL



+ **Spark ML lib**

Panorámica

ML lib es parte de la distribución de Apache Spark

A su vez, Spark es parte del ecosistema Hadoop

…y todo esto surge en el contexto del término de moda: Big Data

Introducción

Spark SQL y SparkML

Vectores locales

Matrices locales
Matrices distribuidas

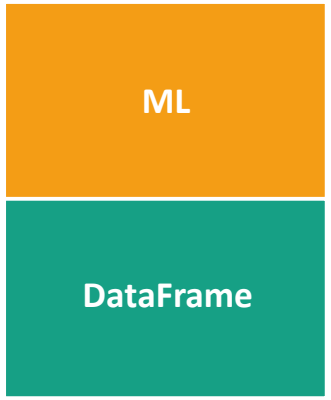
Spark : dos librerías de Aprendizaje Automático

Spark MLlib

- La primera librería, iniciada con Spark 0.8 (Sept. 2013)

Spark ML

- Aparece en Spark 1.2



MLlib

- Fácil de utilizar
- Basada en RDDs
- Madura, con muchos algoritmos implementados
- Difícil de evolucionar
- Estructura RDD demasiado básica
- Obsoleta a partir de 2.0

Spark ML

- Más compleja
- Basado en DataFrames como estructura de datos y en estimadores y transformaciones como algoritmos básicos
- Permite el desarrollo de “pipelines” de ML
- Librería principal a partir de 2.0

Test

¿Cuáles de las siguientes afirmaciones pueden ser ciertas en un programa Spark típico basado en RDDs/Dataframes/Datasets?

A. El programa puede tener cero acciones
B. El programa puede tener una sola acción
C. El programa puede tener una sola transformación
D. El programa puede tener cero transformaciones

+ Spark ML lib

Tipos de datos

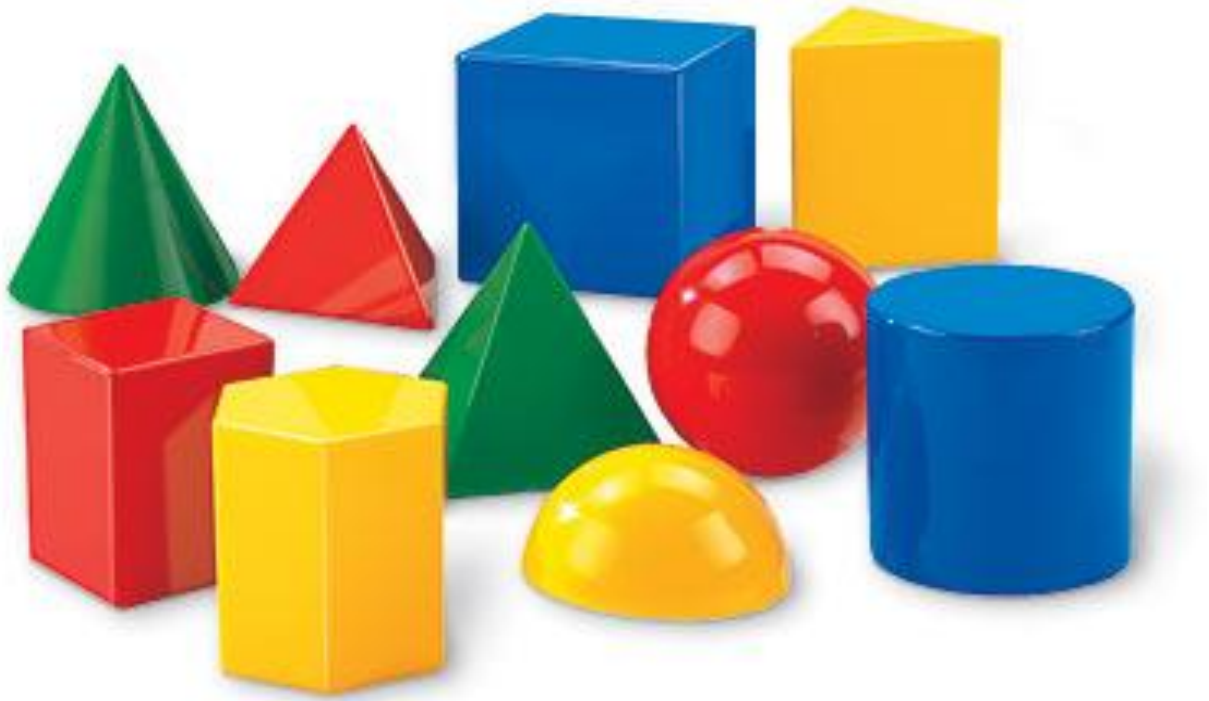
Los tipos de datos fundamentales en Mllib y sus características

Introducción

Spark SQL

Vectores locales

Matrices locales
Matrices distribuidas



Vectores locales en **MLlib**

- Se almacenan de forma local, no distribuidos (ojo al tamaño).
- Índices: Enteros que empiezan en 0. Valores: tipo double

0	1	2	3	4	5
13.5	14.2	22.0	-13.33	24.555	6.0

- 2 tipos
  - Densos: vectores con muchos valores útiles
  - Dispersos: vectores con “huecos”
  - Etiquetados: con un valor (etiqueta) asociado

Ejemplo

Ejemplos de métodos aplicados a los vectores densos

Nota: los mismos métodos son aplicables a los vectores dispersos, ambos son a efectos de Spark el mismo tipo

¿Cuál ocupará menos?

- Supongamos un vector de tamaño N, con M valores útiles, el resto reconocibles por una marca
- Supongamos tamaño de int 4 bytes y el de double 8 bytes
- Vector denso ocupará aprox. N x 8 bytes
- Vector disperso ocupará aprox. M x 4 + M x 8 (y un poquito más al tener que manejar dos estructuras)
- Vector disperso merecerá la pena N x 8 > M x 4 + M x 8
- Vector disperso merece la pena si M/N < 2/3 → menos de dos tercios de ocupación

Vectores y matrices dispersas: casos de uso

- One-hot encoding
  - Elementos de tipo no numérico se codifican de la forma 0000010000 indicando la posición en la lista completa
  - Ejemplo: “España”, “Francia”, “Alemania” → 100, 010, 001

- Matrices dispersas
  - Típicas en recomendadores
  - Ejemplo: premio Netflix (480.000 usuarios, 17770 películas, 100.000.000 de opiniones → densidad 1.17%

Vectores etiquetados (Labeled Points)

- Vectores “etiquetados” con un valor
  - La etiqueta es un double
  - Muy útiles para aprendizaje supervisado → la “etiqueta” es el valor asociado

Ejemplo

Creación de LabeledPoint desde:

- Vectores densos
- Vectores dispersos
- Carga desde fichero en forma de JavaRDD

Acceso a los componentes:

- Etiqueta
- Vector asociado

+ Spark ML lib

Tipos de datos

Los tipos de datos fundamentales en Mllib y sus características

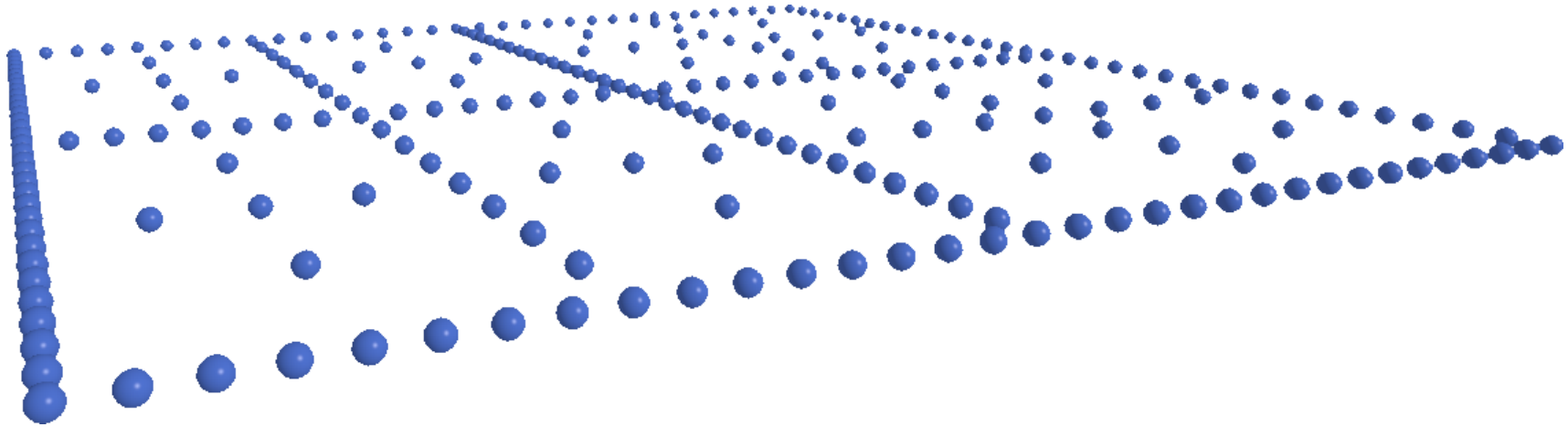
Vectores locales

Matrices locales

Matrices distribuidas

Matrices locales

- Similares a los vectores locales
  - Densos : cantidad elevada de valores inútiles
  - Dispersos : Pocos valores útiles (baja densidad)



+ Spark ML lib

Tipos de datos

Los tipos de datos fundamentales en Mllib y sus características

Vectores locales

Matrices locales

Matrices distribuidas

Matrices distribuidas

- Sintaxis : índices fila y columna son long. Elementos double
- Distribuidas : Varios RDDs, nodos distintos
- Tipos :
  - RowMatrix: matriz distribuida orientada a filas que son vectores locales
  - IndexRowMatrix: matriz distribuida orientada a filas con etiquetas
  - CoordinateMatrix: Matriz distribuida almacenada como una lista COO (row, column, value) con valores almacenados en un RDD
  - BlockMatrix: Los valores asociados cada pareja (fila, columna) son a su vez matrices

RowMatrix

- Creación : Bien a partir de fichero o de colecciones java “paralelizadas”
- Operaciones : Multiplicación, reducción de dimensiones, etc.
  Ojo: algunas operaciones, como la multiplicación
- Importante : Cuidado con intentar “recoger” la matriz entera, puede que no quepa en memoria → mejor usar las operaciones de la matriz o convertir a RDD

IndexedRowMatrix

- Cada fila lleva asociado un índice (tipo Long)
  Es una variante de matriz dispersa en la que solo existen algunas filas
  Las filas son de nuevo vectores, densos o dispersos
- Operaciones : Las mismas que la matriz, añadiendo la posibilidad de extraer el índice
- Construcción : A través de un JRDD<IndexedRow> rows
- Se puede convertir en una RowMatrix (método toRowMatrix())

Ejercicio: IndexedRowMatrix

- Entrada: Queremos crear una IndexedRowMatrix a partir del fichero /usr/local/spark/data/mllib/vectors/vectores2.txt
  - Formato: Cada línea es de la forma: Long:Vector
- Cómo: IndexedRow no tiene método parse. Copiar y modificar leeVectores (ejemplo anterior), usar str.split(“:”) y hacer parsing del Long y del Vector. El método devolverá un JavaRDD<IndexedRow>.
- Salida : Escribir el IndexedRowMatrix en la carpeta /usr/local/spark/data/mllib/vectors/indexedRow

CoordinateMatrix

- En lugar de una matriz distribuida de filas dispersas (IndexedRowMatrix) tenemos una matriz realmente dispersa

- Se almacena un RDD de valores de la forma (fila:Long, columna:Long, valor:double). Cada elemento se llama MatrixEntry

- Se puede convertir en un IndexedRowMatrix (método toIndexedRowMatrix())

BlockMatrix

- Basado en un RDD<MatrixBlock>
- Un MatrixBlock, es ((Int, Int), Matrix) con Matrix, una matriz local
- Creación: desde IndexedRowMatrix o desde CoordinateMatrix llamando a toBlockMatrix()
- Divide la matriz en bloques (por defecto de 1024x1024)
- Todo tipo de operaciones: producto por otra BlockMatrix, traspuesta, etc.

+ Spark ML lib

Estadísticas básicas

Los métodos de machine learning se basan y se miden por medio de estadísticas clásicas y básicas que también están representadas en Spark ML lib

Media y varianza

Estimadores

Estimadores en Spark ML

Correlaciones en Spark ML

+ Spark ML lib

Estadísticas básicas

Los métodos de machine learning se basan y se miden por medio de estadísticas clásicas y básicas que también están representadas en Spark ML lib

Media y varianza

Estimadores

Estimadores en Spark ML

Correlaciones en Spark ML

Media

Partimos de una población (variable aleatoria) numérica con N individuos

$$x_1, x_2, x_3, \dots, x_n$$

Por ejemplo: edad, altura, número de clicks en una página web.

Media: la idea es resumir toda la población en un solo “representante”

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

Media, mediana, moda (medidas de posición)

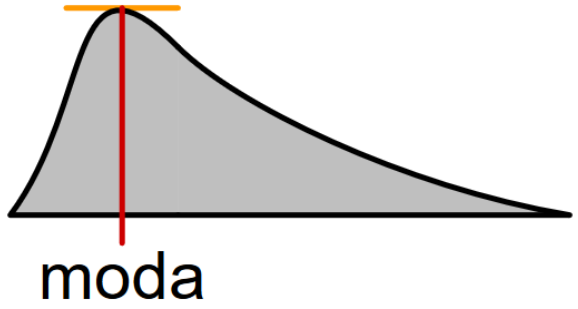
Mediana: valor que deja la mitad de la población a cada lado

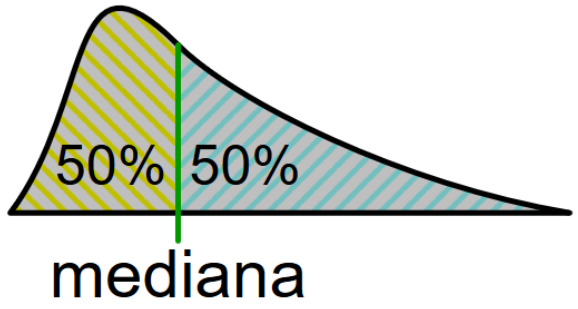
Moda: Valor más repetido

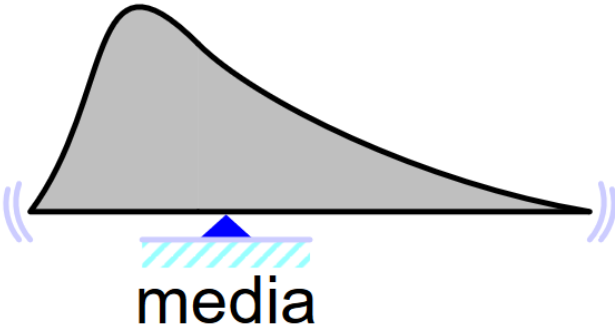
Ejemplo: 3, 4, 4, 4, 98, 101, 102, 103, 104

Media: (3+4+4+4+98+101+102+103+104)/9 = 58.11

Moda: 3 4 4 4 98 101 102 103 104
Mediana: 3 4 4 4 98 101 102 103 104







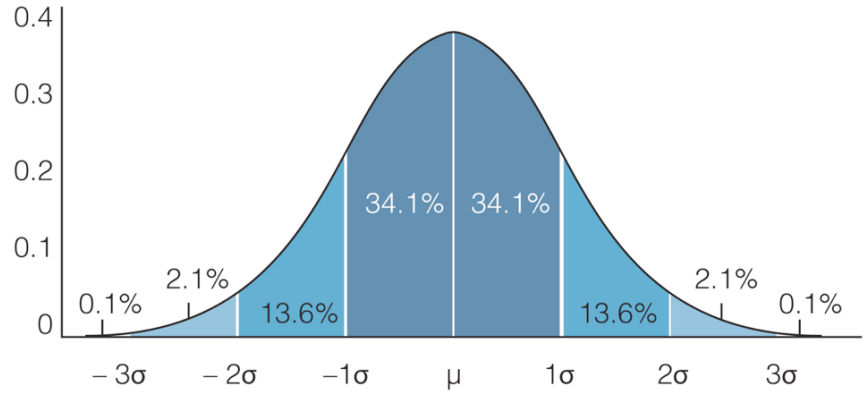
Varianza

Indicador de dispersión ¿están los datos muy agrupados alrededor de la media?

Varianza: Cuadrado de las distancia media a la media

$$\sigma^2 = \frac{\sum_{i=1}^{n} \left(x_i - \bar{x}\right)^2}{n}$$

Distribución normal



μ = Expected Value
-1σ to 1σ = 1 Standard Deviation (ie: ~2/3 of the time, your results/variance will fall within this range)
-2σ to 2σ = 2 Standard Deviations (ie: 95% of the time, your results/variance will fall within this range)
-3σ to 3σ = 3 Standard Deviations (ie: 99.7 of the time, your results/variance will fall within this range)

+ Spark ML lib

Estadísticas básicas

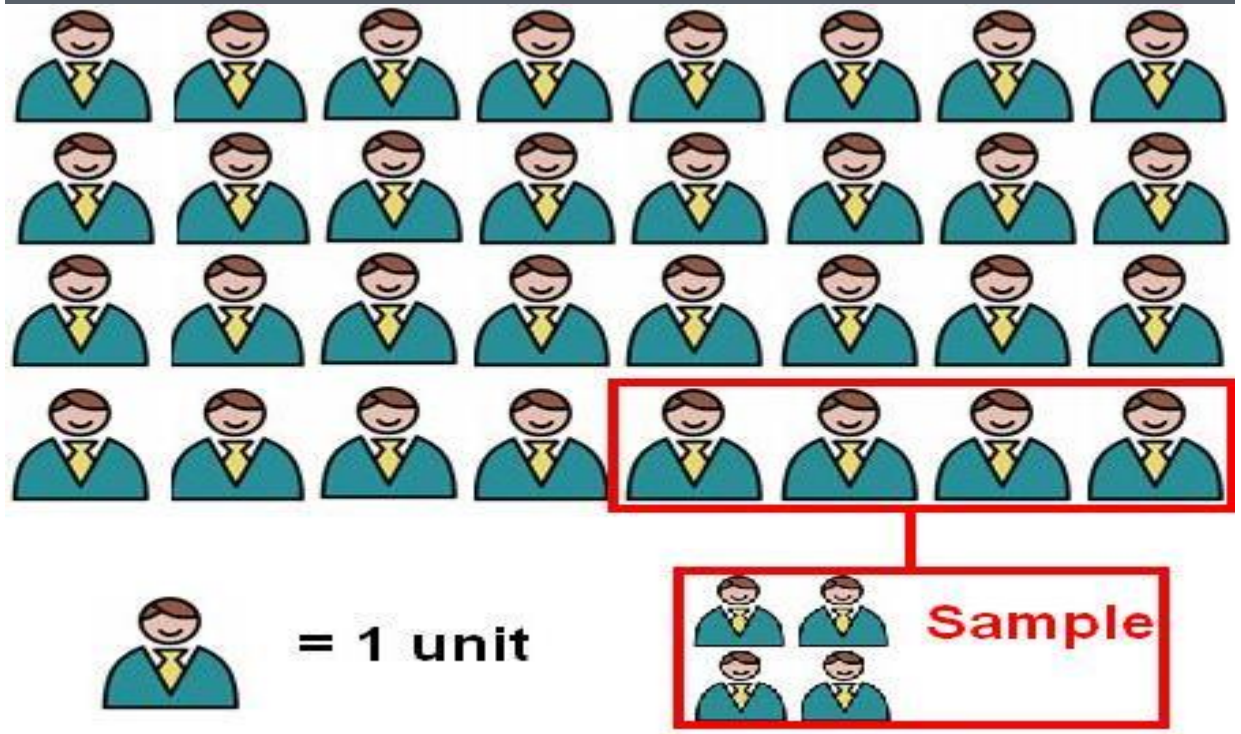
Los métodos de machine learning se basan y se miden por medio de estadísticas clásicas y básicas que también están representadas en Spark ML lib

Media y varianza

Estimadores

Estimadores en Spark ML

Correlaciones en Spark ML



Estimadores: definición

Por desgracia normalmente no tenemos la población total, solo una muestra

$$Y_1 , Y_2, \dots x_1, x_2, x_3, \dots , x_n \dots Y_{N-1} , Y_N$$

Se trata de estimar averiguar parámetros como la media o la varianza a partir de esta muestra

La función que aproxima el parámetro se llama estimador

En ocasiones el estimador no da valores sino un intervalo de confianza con cierto nivel de verosimilitud

Estimadores: Sesgo y error

Supongamos que tenemos un parámetro p, y un estimador e . Entonces

- Sesgo(e) = E[e-p], donde E es la esperanza (imaginar E como la media)
- Si sesgo(e) = 0 el estimador se dice centrado, y en otro caso sesgado
- Error de e en x, x una muestra → e(x) – p
- MSE(e), error cuadrado medio → $E[(e(X)-p)^2]$

Estimador para la media

La misma fórmula para la media de la población y de la muestra

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

Se trata de un estimador centrado, además con un coste lineal

Estimador para la varianza

Podemos usar la misma fórmula para la varianza de la población y de la muestra

$$\sigma^2 = \frac{\sum_{i=1}^{n} \left(x_i - \bar{x}\right)^2}{n}$$

En este caso no se puede usar la media de la población, que no se conoce, sino que habrá que usar la media de la muestra

¿Se trata de un estimador centrado ? Probar el programa Varianza.java y discutir el resultado

+ Spark ML lib

Estadísticas básicas

Los métodos de machine learning se basan y se miden por medio de estadísticas clásicas y básicas que también están representadas en Spark ML lib

Media y varianza

Estimadores

Estimadores en Spark ML

Correlaciones en Spark ML

+ Spark ML lib

Estadísticas básicas

Los métodos de machine learning se basan y se miden por medio de estadísticas clásicas y básicas que también están representadas en Spark ML lib

Media y varianza

Estimadores

Estimadores en Spark ML

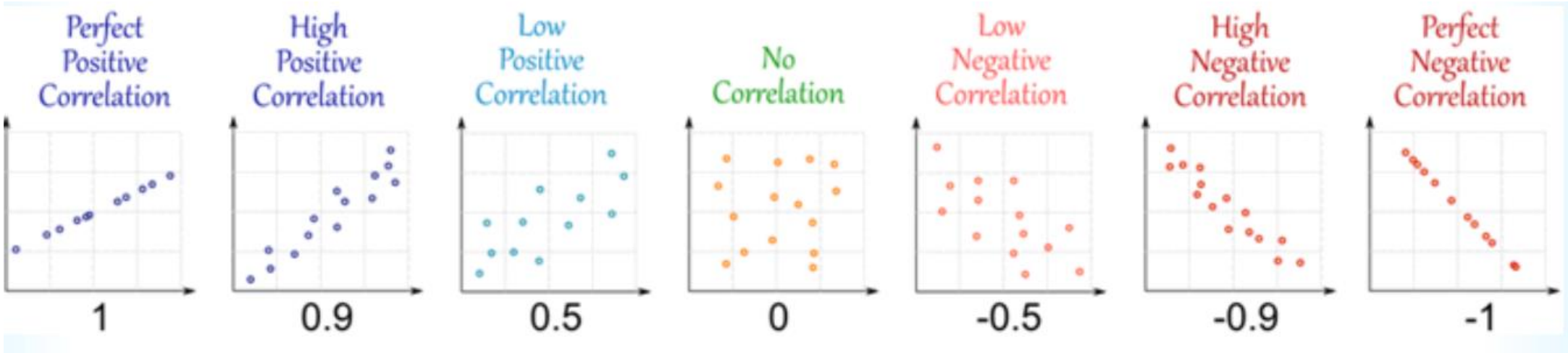
Correlaciones en Spark ML

Correlación lineal

- Dadas dos variables X, Y, ver si se cumple $Y = aX + b$
- El grado de correlación se representa por un valor $r \in [-1,1]$
- Cuanto más cerca este $|r|$ de 1 mayor correlación
- Si $r \approx 1$ se tiene que X crece a la vez que Y ($a>0$)
- Si $r \approx -1$ se tiene que X crece cuando Y disminuye y viceversa

Correlación lineal

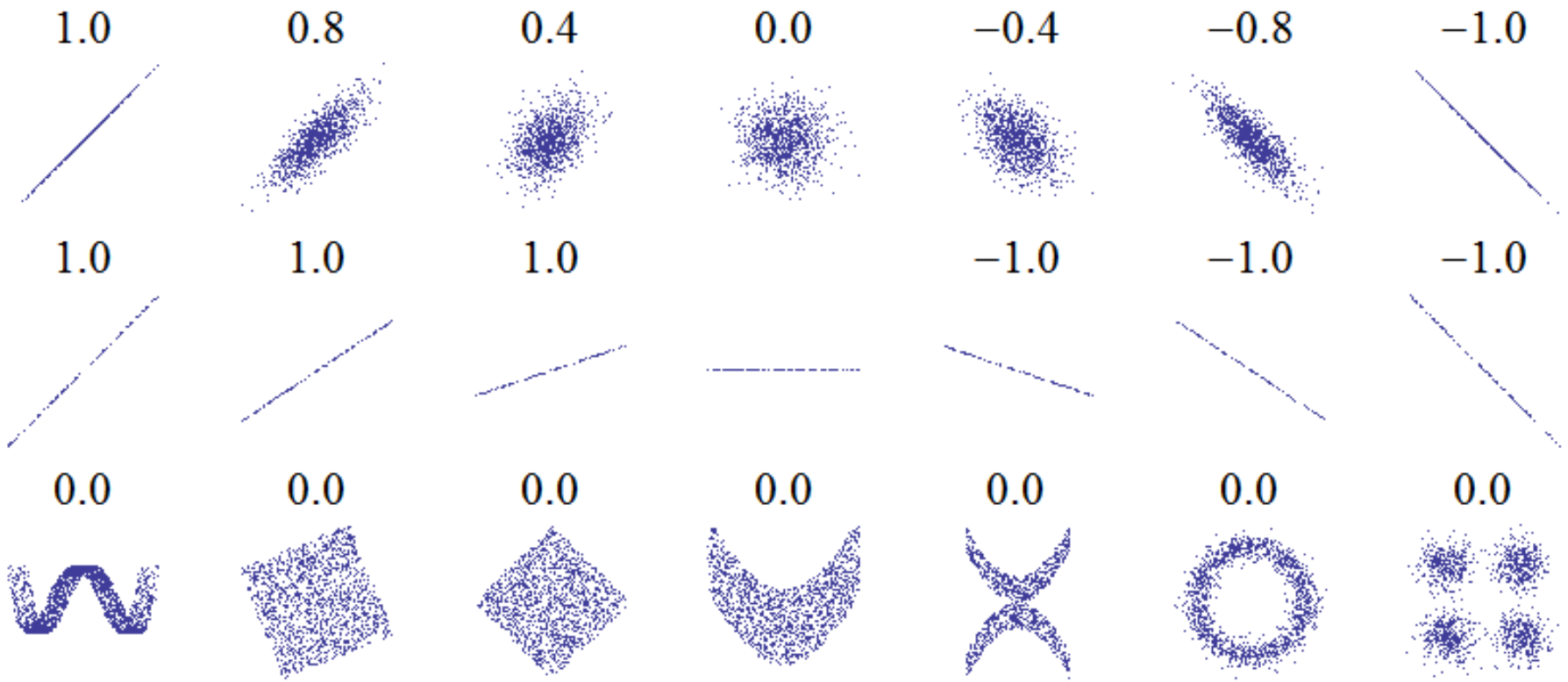
De forma gráfica



Fuente https://www.mathsisfun.com/data/correlation.html

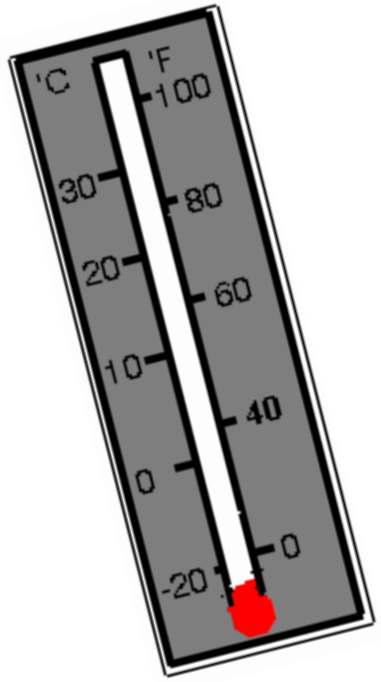
Correlación lineal: limitaciones

Ojo: ¡Sólo correlación lineal!



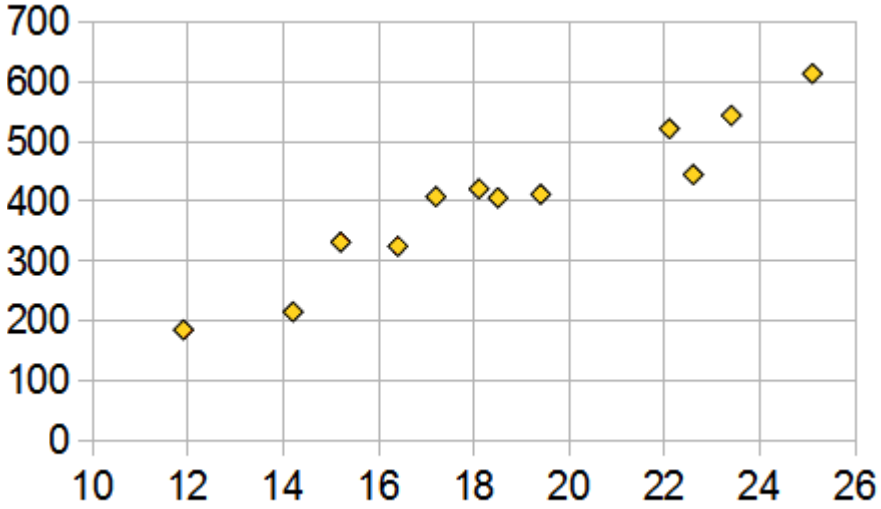
Ejemplo

Correlación de dos variables



Ejemplo: heladería

Temperatura frente a ventas	
Temperatura °C	Ventas de Helados
14,2°	215
16,4°	325
11,9°	185
15,2°	332
18,5°	406
22,1°	522
19,4°	412
25,1°	614
23,4°	544
18,1°	421
22,6°	445
17,2°	408



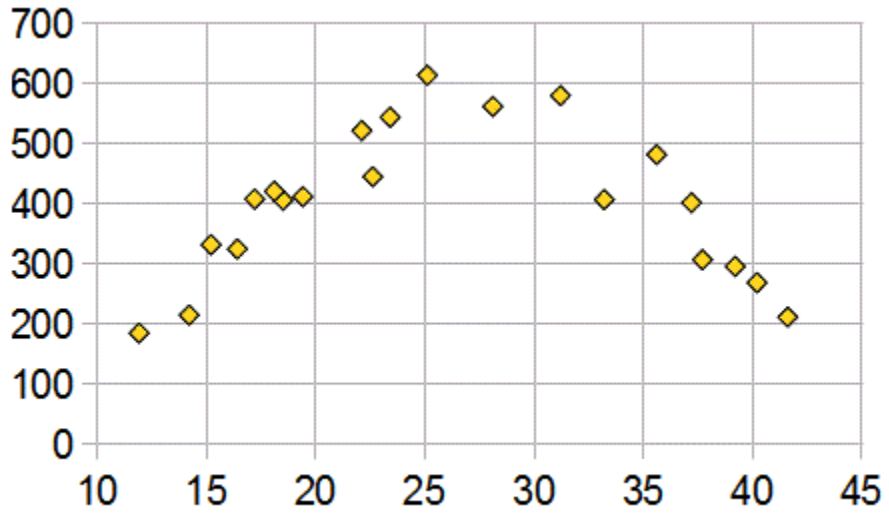
Relación aparentemente lineal
Queremos confirmarlo

Fuente https://www.mathsisfun.com/data/correlation.html

Ejemplo de correlaciones (Pr.EstadisticasBasicas.Correlaciones)

```
SparkSession spark = SparkSession.builder().appName("Correlaciones").master("local[2]").getOrCreate();

JavaSparkContext jsc = new JavaSparkContext(spark.sparkContext()) ;
JavaDoubleRDD x = jsc.parallelizeDoubles(Arrays.asList(14.2, 16.4, 11.9, 15.2, 18.5,
                                                       22.1, 19.4, 25.1, 23.4, 18.1, 22.6, 17.2 ));

JavaDoubleRDD y = jsc.parallelizeDoubles(Arrays.asList(215.0, 325.0, 185.0, 332.0, 406.0,
                                                       522.0, 412.0, 614.0, 544.0, 421.0, 445.0, 408.0 ));

Double correlation = Statistics.corr(x.srdd(), y.srdd());

System.out.println("Correlation is: " + correlation); // 0.9575066230015948
spark.stop();
```

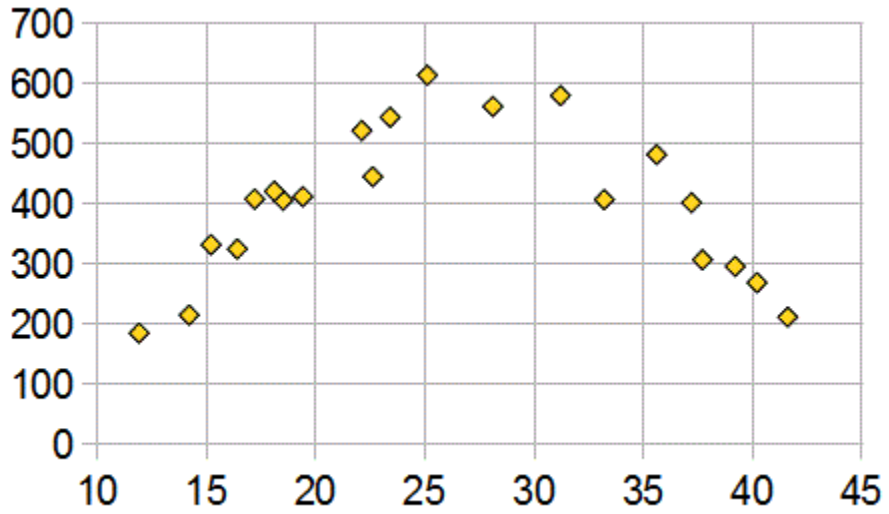
Más avisos



¿Cómo será r?

Fuente https://www.mathsisfun.com/data/correlation.html

Linealidad, siempre linealidad

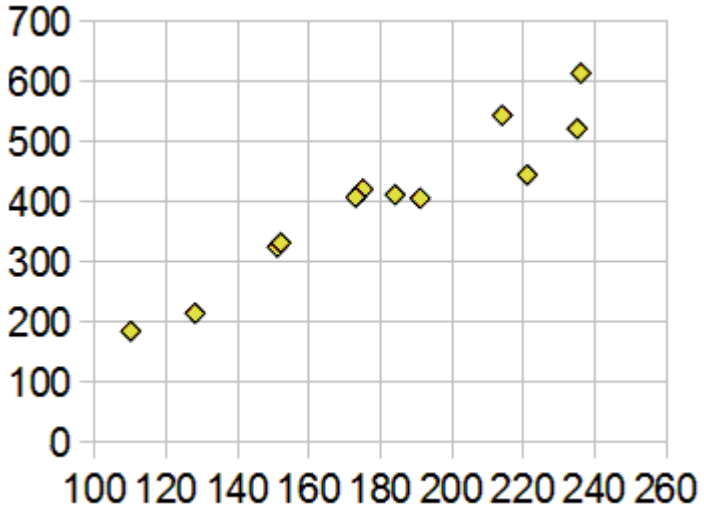


Sale un valor **próximo a cero**; aunque sin duda están relacionadas pero no de una forma lineal

Fuente https://www.mathsisfun.com/data/correlation.html

Cuidado con sacar consecuencias

La correlación no supone necesariamente una relación causa-efecto directa



Fuente https://www.mathsisfun.com/data/correlation.html

Matriz de correlaciones

- Tenemos varias variables, y queremos buscar relaciones entre ellas
- Partimos del fichero "correlaciones.txt"
- Obtendremos un RDD<Vector> data, usando el método leeVectores
- Obtenemos e imprimimos la matriz de correlaciones con:

```
Matrix correlMatrix = Statistics.corr(data.rdd());
System.out.println(correlMatrix.toString());
```