MATLAB aplicado al Big Data

Tomás Robles Valladares

Ramón Alcarria Garrido

Borja Bordel Sánchez

Diego Martín de Andrés

MATLAB aplicado a la ingeniería telemática

Departamento de Ingeniería de Sistemas Telemáticos (UPM)

PROGRAMA

- Introducción al Big Data
- Manejo de variables de gran tamaño en MATLAB
- Computación y memoria distribuidas en MATLAB
- Procesado de grandes conjuntos de datos e imágenes de gran tamaño con MATLAB
- Reconocimiento de patrones en MATLAB

INTRODUCCIÓN AL BIG DATA

- El Big Data es un concepto que hace referencia a la acumulación de grandes cantidades de datos y a los procedimientos usados para encontrar patrones repetitivos dentro de esos datos
- El término fue utilizado por primera vez por Michael Cox y David Ellsworth en 1997
  - Artículo original disponible en: https://www.evl.uic.edu/cavern/rg/20040525_renambot/Viz/parallel_volviz/paging_outofcore_viz97.pdf

INTRODUCCIÓN AL BIG DATA

"Visualization provides an interesting challenge for computer systems: data sets are generally quite large, taxing the capacities of main memory, local disk, and even remote disk. We call this the problem of **big data**. When data sets do not fit in main memory (in core), or when they do not fit even on local disk, the most common solution is to acquire more resources."

INTRODUCCIÓN AL BIG DATA

- No existe una cantidad de datos fija a partir de la cual se hable de Big Data
  - El término hace referencia a una cantidad de datos que supera la capacidad del software habitual para ser adquiridos y procesados en un tiempo razonable
  - El límite entre el Big Data y el procesado de datos ordinario es, por tanto, subjetivo
  - De hecho, el límite del procesamiento ordinario ha ido creciendo a lo largo de los años

INTRODUCCIÓN AL BIG DATA

- Desde su origen, se han creado múltiples herramientas para afrontar el reto del Big Data
  - Hadoop
  - NoSQL
  - Business intelligence
  - Machine Learning
  - MapReduce
  - …

INTRODUCCIÓN AL BIG DATA

- En Big Data se distinguen tres tipos de datos
  - Datos estructurados: Datos que tienen bien definida su longitud y su formato (p.ej. fechas, números o cadenas de caracteres). Se almacenan en tablas.
  - Datos no estructurados: Datos que carecen de un formato específico. Se almacenan tal y como fueron recolectados. P. ej. documentos multimedia, e-mails o documentos de texto.

INTRODUCCIÓN AL BIG DATA

  - Datos semiestructurados: Datos que contienen marcadores para separar los diferentes elementos, aunque no se limitan a unos campos determinados. Son ejemplos los ficheros HTML, XML o los objetos JSON.

- Básicamente, el reto del Big Data puede entenderse como la suma de dos problemas
  - El almacenamiento de grandes cantidades de datos
  - El procesado de dichos datos

INTRODUCCIÓN AL BIG DATA

- La tecnología NoSQL trata de abordar el problema del almacenamiento de grandes cantidades de datos
- Se trata de sistemas de gestión de bases de datos que difieren de las tecnologías clásicas en aspectos tan importantes como el uso de SQL como lenguaje de consultas.

INTRODUCCIÓN AL BIG DATA

- CouchDB o MongoDB son las soluciones más conocidas
- En MATLAB, sin embargo, se trabaja con otras dos tecnologías
  - *MapReduce*, una tecnología de Google que permite seleccionar un subconjunto de datos, agruparlos o reducirlos y cargarlos en otra colección
  - *Hadoop*, una tecnología de Apache diseñada para almacenar y procesar grandes cantidades de datos.

INTRODUCCIÓN AL BIG DATA

- En Big Data, en líneas generales se emplean cuatro técnicas básicas de procesado de datos
  - Asociación: Permite encontrar relaciones entre diferentes variables
  - Minería de datos: Tiene como objetivo encontrar comportamientos predictivos. Engloba un gran conjunto de técnicas que combinan métodos estadísticos y de *machine learning*

INTRODUCCIÓN AL BIG DATA

  - Agrupación: El propósito es encontrar similitudes entre grupos de individuos, y el descubrimiento de los propios grupos una vez identificadas las cualidades que los definen.
  - Análisis de texto: Esta metodología, como su nombre indica, permite extraer información de los datos textuales

- En MATLAB la mayor parte de rutinas, algoritmos y herramientas realizan minería de datos

MANEJO DE VARIABLES DE GRAN TAMAÑO EN MATLAB

- Como se ha visto, la característica principal del Big Data es el manejo de variables que exceden los recursos de memoria física e, incluso, de disco
- En esta sección nos centraremos en abordar el caso de variables que exceden la memoria física del sistema pero que pueden quedar alojadas en el disco

MANEJO DE VARIABLES DE GRAN TAMAÑO EN MATLAB

- En MATLAB, desde 2014, se han implementado o mejorado varias soluciones para abordar el manejo de variables de gran tamaño
  - Modificación del entorno MATLAB para poder acceder a todos los recursos del sistema
  - Capacidad para mapear variables en memoria por bloques
  - Capacidad para trabajar directamente sobre variables alojadas en disco

MANEJO DE VARIABLES DE GRAN TAMAÑO EN MATLAB

- La primera de las acciones para poder trabajar con variables de gran tamaño fue ampliar la memoria disponible en MATLAB
- En la versión tradicional de MATLAB, con una arquitectura de 32bits, sólo se disponía de 2GB de memoria
- Las nuevas versiones de 64 bits permiten hacer uso de toda la memoria física del sistema

MANEJO DE VARIABLES DE GRAN TAMAÑO EN MATLAB

- El problema radica en que la mayoría de los usuarios de MATLAB emplean PCs convencionales para ejecutar la aplicación
- La media de memoria física total disponible se sitúa en el entorno de los 4GB
  - Una cantidad inferior a la capacidad de un DVD convencional (4,7GB)

MANEJO DE VARIABLES DE GRAN TAMAÑO EN MATLAB

- Desde un principio, se demostró que eran necesarios mecanismos adicionales
- Especialmente para procesar video, imágenes o, en general, datos desestructurados de tipo audiovisual o tipo “documento enriquecido”
  - Documentos PDF con metainformación
  - Reconocimiento del habla
  - Etc.

MANEJO DE VARIABLES DE GRAN TAMAÑO EN MATLAB

- Desde 2006, MATLAB incluye una funcionalidad que permite mapear en memoria bloques de archivos de gran tamaño
  - Función *memmapfile*
- El objetivo es sólo mantener variables que se ajusten a la memoria disponible
  - El resto de los datos permanecerá en disco

MANEJO DE VARIABLES DE GRAN TAMAÑO EN MATLAB

- El procedimiento es directamente aplicable a conjuntos de datos
- El resultado es una forma eficiente de acceder a conjuntos de Big Data alojados en disco, que son demasiado grandes para ajustarse a la memoria, o que tardarían demasiado tiempo en cargarse

MANEJO DE VARIABLES DE GRAN TAMAÑO EN MATLAB

- La función debe ser invocada de la forma

```
mapa = memmapfile('Nombre', 'Atributo1', valor, … , Atributon', valor);
```

- Una vez obtenido el mapa en memoria, el array de datos queda accesible de la forma

```
Mapa.Data
```

MANEJO DE VARIABLES DE GRAN TAMAÑO EN MATLAB

- Algunos de los atributos más importante son:
  - *Offset*, para indicar el byte desde el que se debe comenzar a mapear
  - *Format*, para indicar el formato de los datos mapeados
    - Incluye tipo, tamaño del array y nombre de los campos (cuando proceda)
    - Solo admite como valores tipos de datos básicos o arrays de celdas

MANEJO DE VARIABLES DE GRAN TAMAÑO EN MATLAB

- *Writable*, para indicar si el bloque puede ser modificado o sólo consultado

- Mapeado un boque en memoria, sus atributos pueden ser modificados de forma dinámica haciendo

```
Mapa.Atributo = nuevo_valor;
```

MANEJO DE VARIABLES DE GRAN TAMAÑO EN MATLAB

- Ejemplo: mapear todo el fichero *datos.dat*, que almacena datos tipo *double*

```
mapa = memmapfile('datos.dat', 'Format', 'double');
```

- Ejemplo: mapear desde el byte 2048 el fichero *datos.dat*, que almacena datos tipo *double*

```
mapa = memmapfile('datos.dat', 'Offset', 2047, 'Format', 'double');
```

MANEJO DE VARIABLES DE GRAN TAMAÑO EN MATLAB

- Ejemplo: mapear los bytes del 2048 al 2148 del fichero *datos.dat*, que almacena datos tipo *double*

```
mapa = memmapfile('datos.dat', 'Offset', 2047, 'Format',{'double', [100 1], 'datos'});
```

- A los datos se accede como

```
Mapa.Data.datos
```

MANEJO DE VARIABLES DE GRAN TAMAÑO EN MATLAB

- Ejemplo: mapear los bytes del 4000 al 12000 del fichero *datos.dat*, que almacenan dos estructuras de dos campos donde el primer campo es un array de 50 posiciones de tipo *uint16*, y el segundo una matriz de dimensiones 5x10 de tipo *uint64*

```
mapa = memmapfile('datos.dat', 'Offset', 3999, 'Format',…
{'uint16', [50 1], 'campo1';'uint64', [5 10], 'campo2'}, 'Repeat', 2);
```

MANEJO DE VARIABLES DE GRAN TAMAÑO EN MATLAB

- El acceso a los datos se haría de la forma

```
datos = mapa.Data;
v1 = datos(1).campo1;
v2 = datos(2).campo1;
…
```

- Es preciso, por tanto, conocer previamente la estructura del fichero
  - Si no la conocemos, puede mapearse como datos binarios (sin especificar el atributo *Format*)

MANEJO DE VARIABLES DE GRAN TAMAÑO EN MATLAB

- La función *memmapfile* tiene un funcionamiento similar a la función *mmap* de Linux
  - Aunque permite trabajar a alto nivel si conocer direcciones de memoria
- Es un método muy eficiente y que permite escribir código muy legible y elegante

MANEJO DE VARIABLES DE GRAN TAMAÑO EN MATLAB

- Presenta, sin embargo, algunas desventajas
  - A nivel de usuario, MATLAB presenta los datos binarios como *uint8*. Esto puede ser problemático, por ejemplo, en fichero tipo “mapa de bits”
  - El tamaño máximo de bloque esta limitado a 2 GB en arquitecturas de 32 bits, y a 256 TB en las de 64 bits

MANEJO DE VARIABLES DE GRAN TAMAÑO EN MATLAB

- El mapeado sólo es exitoso cuando el orden de los bytes del fichero coincide con el nativo del sistema donde se ejecuta MATLAB
  - Big-endian, little-endian, etc.
  - Para averiguar este punto existe la función *computer*

```
str = computer
archstr = computer('arch')
[str,maxsize] = computer
[str,maxsize,endian] = computer
```

MANEJO DE VARIABLES DE GRAN TAMAÑO EN MATLAB

- Hay casos en los que ni siquiera se puede mapear un bloque en memoria con *memmapfile*
  - Por ejemplo, por que un solo dato es mayor que la memoria disponible
- Desde 2011, MATLAB incluye la función *matfile*, que permite leer y modificar variables directamente sobre el disco, sin cargarlas en memoria

MANEJO DE VARIABLES DE GRAN TAMAÑO EN MATLAB

- Su sintaxis es más sencilla que *memmapfile*

```
matObject = matfile('fichero', 'Writable', isWritable);
```

- Sólo admite el atributo *Writable* que indica si la variable puede ser modificada o sólo leída
- Si conocemos el nombre de las variables éstas pueden accederse como

```
var =  matObjet.nombre;
```

MANEJO DE VARIABLES DE GRAN TAMAÑO EN MATLAB

- Si no conocemos el nombre de las variables, podemos conocer cuáles les ha asociado MATLAB

```
varlist = who(matObject);
```

- El resultado es un array columna de cadenas de caracteres

COMPUTACIÓN Y MEMORIA DISTRIBUIDAS EN MATLAB

- Una de las primeras soluciones al problema del Big Data es agrupar varios equipos entre los que se distribuya el trabajo de almacenamiento y procesado
  - Como resultado, cada equipo concreto realizará un trabajo adecuado a sus capacidades
- MATLAB incluye varias herramientas para facilitar este tipo de soluciones

COMPUTACIÓN Y MEMORIA DISTRIBUIDAS EN MATLAB

- Una primera solución son las funciones multi-hebra
- Como vimos en el Tema 2, MATLAB admite programación concurrente
- Lanzando cada hebra en un procesador diferente puede lograrse un procesado compartido

COMPUTACIÓN Y MEMORIA DISTRIBUIDAS EN MATLAB

- Muchas funciones nativas de MATLAB admiten opciones para configurar una ejecución distribuida
  - *fft(), inv(),* etc.
- En aquellos casos en los que no se soporte de forma nativa, el *Parallel Computing Toolbox* permite hacer concurrencia explícita
  - Requiere conocimientos avanzados

COMPUTACIÓN Y MEMORIA DISTRIBUIDAS EN MATLAB

- Una mejora de las ejecuciones multi-hebra es la asignación de parte de trabajado a la Unidad de Procesamiento Gráfico (GPU)
  - Si se dispone de alguna
- Algunas funciones hacen uso de estos dispositivos de forma implícita
  - *filter()*

COMPUTACIÓN Y MEMORIA DISTRIBUIDAS EN MATLAB

- En el resto de casos, el *Parallel Computing Toolbox* incluye herramientas para realizar asignaciones de trabajo a la GPU
  - Requiere un uso avanzado que no vamos a ver
- Si no se desea amplear los *toolboxes* de MATLAB, el entorno admite integración idrecta con el *kernel CUDA* de nVidia
  - *Compute Unified Device Architecture*

COMPUTACIÓN Y MEMORIA DISTRIBUIDAS EN MATLAB

- Cuando se dispone de un *cluster* de ordenadores conectados en red y configurados para realizar computación distribuida, MATLAB puede ejecutarse en forma de servidor de trabajos
  - Se debe instalar el complemento *MATLAB Distributed Computing Server*

COMPUTACIÓN Y MEMORIA DISTRIBUIDAS EN MATLAB

- El servidor debe ejecutarse sobre un planificador apropiado (no incluido en MATLAB)
  - *IBM® Platform LSF*
  - *Microsoft® Windows® HPC Server*
  - …
- Los script y funciones que ejecute el servidor deben hacer uso de la programación concurrente de M

COMPUTACIÓN Y MEMORIA DISTRIBUIDAS EN MATLAB

- Para el caso del Big Data, se ha optimizado el *MATLAB Distributed Computing Server* para su uso junto con la funcionalidad *mapreduce*, de forma que pequeños programas de análisis puedan ser escalados directamente a trabajos sobre clusters Hadoop
  - Más adelante revisaremos estos conceptos

COMPUTACIÓN Y MEMORIA DISTRIBUIDAS EN MATLAB

- Si no se dispone de un cluster de ordenadores propio, *MATLAB Distributed Computing Server* incluye herramientas para enviar trabajos a la plataforma *Elastic Computing Cloud (EC2)* de Amazon
  - http://es.mathworks.com/products/parallel-computing/parallel-computing-on-the-cloud/

COMPUTACIÓN Y MEMORIA DISTRIBUIDAS EN MATLAB

- Tanto *MATLAB Distributed Computing Server* como el *Parallel Computing Toolbox* permiten, también, trabajar sobre datos distribuidos
- En concreto permiten trabajar con matrices y arrays multidimensionales distribuidos en la memoria de un cluster de ordenadores
- De esta manera se aborda el problema de trabajar con variables que exceden la capacidad del disco

COMPUTACIÓN Y MEMORIA DISTRIBUIDAS EN MATLAB

- En MATLAB hay básicamente tres formas de crear un array distribuido
  - Crear un array que es distribuido desde su origen
  - Dividir un array de gran tamaño en varias partes
  - Agrupar como un array distribuido varios conjuntos de datos independientes
- En el primer caso, MATLAB gestiona el proceso. En los dos restantes es precio declarar un bloque de programación concurrente.

COMPUTACIÓN Y MEMORIA DISTRIBUIDAS EN MATLAB

- En MATLAB ,muchas funciones de creación de arrays admiten como atributo un objeto *codistributor*
  - http://es.mathworks.com/help/distcomp/using-matlab-functions-on-codistributed-arrays.html
- Dicho objeto encapsula el número de dimensiones de distribución….
  - Siempre menor que el número de dimensiones del array

COMPUTACIÓN Y MEMORIA DISTRIBUIDAS EN MATLAB

- … la dimensión sobre la que se distribuye,…
  - En el caso bidimensional, filas o columnas
- …e incluso el patrón de distribución que se debe seguir
  - En cuántas partes y de qué dimensiones deben ser las partes

COMPUTACIÓN Y MEMORIA DISTRIBUIDAS EN MATLAB

- Ejemplo: Crear una matriz 10x10 de “unos”, distribuida en columnas (primera dimensión)

```
codist = codistributor (‘1d’, 1);
resultado = ones(10, cosdist);
```

- Ejemplo: Crear una matriz 10x10x10 de “unos”, distribuida de forma bidimensional

```
codist = codistributor (‘2dbc’);
resultado = ones([10 10 10], cosdist);
```

COMPUTACIÓN Y MEMORIA DISTRIBUIDAS EN MATLAB

- Ejemplo: Crear una matriz 10x10 de “unos”, distribuida en columnas en tres partes, de tal forma que la primera parte tenga 3 columnas, 5 la segunda y 2 la tercera

```
codist = codistributor (‘1d’, 1, [3 5 2], [10 10]);
resultado = ones(10, cosdist);
```

COMPUTACIÓN Y MEMORIA DISTRIBUIDAS EN MATLAB

- En otros casos, ya existe un array de gran tamaño que se desea distribuir entre varias máquinas
- El código que permite su distribución es el siguiente

```
spmd
    array_distribuido = codistributed(array);
end
```

COMPUTACIÓN Y MEMORIA DISTRIBUIDAS EN MATLAB

- En cada máquina se puede acceder al fragmento del array asignado como

```
getLocalpart(array_distribuido);
```

- La función *codistributed* admite como segundo argumento un objeto *codistributer* donde pueden indicarse la dimensión de distribución, el patrón, etc.

COMPUTACIÓN Y MEMORIA DISTRIBUIDAS EN MATLAB

- En el caso de que se tenga un array local en cada máquina, y con ellos se quiera construir un array distribuido puede escribirse

```
spmd
  array_distribuido = codistributed.build(array_local);
end
```

- Es condición imprescindible que todas las partes se llamen igual en todas las máquinas

COMPUTACIÓN Y MEMORIA DISTRIBUIDAS EN MATLAB

- La función *codistributed.build* admite como segundo argumento un objeto *codistributer* donde pueden indicarse la dimensión de distribución, el patrón, etc.

PROCESADO DE GRANDES CONJUNTOS DE DATOS […] CON MATLAB

- Una vez hemos logrado acceder, almacenar y/o cargar los datos que van a ser procesados, debe aplicarse el algoritmo correspondiente
- En grandes conjuntos de datos, el tiempo requerido puede ser inasumible
- MATLAB implementa varias herramientas para aliviar este problema
  - Muchas especializadas en el tratamiento de imágenes

PROCESADO DE GRANDES CONJUNTOS DE DATOS […] CON MATLAB

- La primera alternativa es transformar el programa en un algoritmo concurrente que haga uso de un cluster de ordenadores
- En otros capítulos ya hemos hablado de estas herramientas
  - *MATLAB Distributed Computing Server*
  - Arrays distribuidos
  - …

PROCESADO DE GRANDES CONJUNTOS DE DATOS […] CON MATLAB

- Un procedimiento muy general de procesamiento es la utilidad *mapreduce*
- Esta tecnología de Google se encuentra integrada en MATLAB desde finales de 2014
- Aunque se abstrae de muchos de los detalles originales su uso con conjuntos realmente grandes y desestructurados es muy complejo

PROCESADO DE GRANDES CONJUNTOS DE DATOS […] CON MATLAB

- Se basa en el siguiente esquema
  - Una primera función *mapper* analiza todos los datos, extrae de ellos la información relevante y adecuada, y a la agrupa asociando a cada grupo una clave
  - Una segunda función *reducer* se aplica a cada conjunto asociado a una clave, y reduce todo el conjunto a uno de menor tamaño, compendiando información

PROCESADO DE GRANDES CONJUNTOS DE DATOS […] CON MATLAB

- Por ejemplo: Dado un texto describiendo diferentes ciudades y sus niveles de contaminación queremos conocer el nivel medio de polución en cada ciudad
  - La función *mapper* analiza línea a línea el texto y crea conjuntos numéricos indicando los niveles de polución, a los que se asocia como clave el nombre de la ciudad
  - La función *reducer* calcula la media de los niveles

PROCESADO DE GRANDES CONJUNTOS DE DATOS […] CON MATLAB

- Una vez creadas las funciones *mapper* y *reducer* se puede emplear la utilidad como

```
salida = mapreduce(datos, mapper, reducer);
```

- MATLAB se encarga de conectar ambas funciones y de gestionar los conjuntos de claves
- **mapper** y **reducer** son manejadores que apuntan a dichas funciones

PROCESADO DE GRANDES CONJUNTOS DE DATOS […] CON MATLAB

- Todas las funciones *mapper* deben seguir este esquema

```
function mapper(datos, info, intermKVStore)
    %Procesado
    add(intermKVStore,‘clave', datoSeleccionado);
end
```

- **intermKVStore** es un iterador apuntando a los conjuntos de datos intermedios entre las funciones *mapper* y *reducer*

PROCESADO DE GRANDES CONJUNTOS DE DATOS […] CON MATLAB

- Todas las funciones *reducer* deben seguir este esquema

```
function reducer(intermKey, intermValIter, outKVStore)
        %Procesado
        add(outKVStore,‘clave',datoReducido);
end
```

- **intermVallter** es un iterador apuntando a los datos asociados a la clave **intermKey**
- **outKVStore** es un iterador apuntando a los conjuntos de datos reducidos de salida

PROCESADO DE GRANDES CONJUNTOS DE DATOS […] CON MATLAB

- Uno de los usos más comunes de esta herramienta es la creación de aplicaciones para su despliegue en Hadoop
- Para ello basta emplear *MATLAB Coder*, integrando en la aplicación un fichero de configuración para Hadoop
- Al margen de esta utilidad genérica existen otras desarrolladas para aplicaciones específicas

PROCESADO DE GRANDES CONJUNTOS DE DATOS […] CON MATLAB

- Un primer caso particular, es el procesamiento de imágenes
- Cuando se trabaja con imágenes de tamaño muy grande, aplicar los algoritmos de análisis más pesados puede ser un gran desafío
- Desde 2009 MATLAB incluye en el *Image Processing Toolbox* la función *blockproc* que permite procesar por bloques una imagen

PROCESADO DE GRANDES CONJUNTOS DE DATOS […] CON MATLAB

- Su uso es muy simple

```
Imagen_procesada = blockproc(imagen_cruda,tamañoBloque,funcion)
```

- **tamañoBloque** es un vector de dos posiciones indicando las filas y columnas de las que se compone un bloque
- **funcion,** es un manejador apuntando a la función que se pretende aplicar a la imagen

PROCESADO DE GRANDES CONJUNTOS DE DATOS […] CON MATLAB

- El segundo caso particular de interés es el procesado de video
- Desde Mayo de 2015 MATLAB incluye el *Vision HDL Toolbox*
- Este Toolbox proporciona soporte para crear algoritmos que procesen datos que se adquieren en *streaming* de píxeles (video)

PROCESADO DE GRANDES CONJUNTOS DE DATOS […] CON MATLAB

- Esta pensado para exportar los algoritmos a FPGAs y ASICs que permitan procesado en tiempo real.
- Soporta múltiples interfaces, tamaños y velocidades de fotogramas, incluyendo el vídeo de alta definición (1080p).
- Los algoritmos de procesamiento de este *Toolbox* utilizan una arquitectura que se ajusta a las implementaciones de HDL.

PROCESADO DE GRANDES CONJUNTOS DE DATOS […] CON MATLAB

- Este tecnología es muy eficiente en el uso de la memoria, aunque exige grandes capacidades de procesamiento
- Sin embargo, se trata de una tecnología aún no probada por usuarios a gran escala, por lo que no vamos a ver en detalle su utilización en este curso

RECONOCIMIENTO DE PATRONES EN MATLAB

- El aprendizaje automático resulta útil para extraer información y desarrollar modelos predictivos con conjuntos de Big Data
- Por otro lado, reconocer patrones en grandes conjuntos de datos es la aplicación más inmediata del Big Data
- En MATLAB hay herramientas tanto para reconocer patrones como para realizar aprendizaje automático

RECONOCIMIENTO DE PATRONES EN MATLAB

- Reconocer un patrón es identificar a que clase de entre un conjunto conocido pertenece un vector de observaciones
- MATLAB, básicamente, implementa cuatro procedimientos de reconocimiento de patrones
  - Análisis discriminante
  - K-vecinos más cercanos
  - Máquinas de vectores de soporte
- Suelen usarse junto con “selección de características”

RECONOCIMIENTO DE PATRONES EN MATLAB

- “Selección de características” describe un conjunto de herramientas empleadas para reducir las entradas de un algoritmo a un tamaño más apropiado para su procesamiento y análisis
- Permite en un vector de observaciones, seleccionar aquellas más discriminativas según un determinado criterio

RECONOCIMIENTO DE PATRONES EN MATLAB

- En MATLAB, se puede realizar selección de características con la función *rankfeatures()*
- No vamos a profundizar porque no es, en sentido estricto, un algoritmo de reconocimiento de patrones
- Se combina con ellos para agilizar y eliminar ruido, pero no es imprescindible

RECONOCIMIENTO DE PATRONES EN MATLAB

- El análisis discriminante es una técnica estadística multivariante cuya finalidad es describir las diferencias significativas entre grupos
- Para ello se observan varias variables discriminantes
- Más concretamente, se comparan y describen las medias de las variables clasificadoras

RECONOCIMIENTO DE PATRONES EN MATLAB

- MATLAB compara los datos dados con los diferentes grupos definidos y elige aquel grupo con el que menos diferencias existan
- Uso básico

```
resultado = classify (datos, grupos, nombreGrupos)
```

RECONOCIMIENTO DE PATRONES EN MATLAB

- MATLAB toma cada fila de la matriz **datos** como una evaluación, y averigua cual de las filas de **grupos** es a la que se parece más
- A cada fila de **grupos** se le asocia un nombre que se escribe en la equivalente fila de **nombreGrupos**
- La salida es una vector con los nombres de los grupos identificados

RECONOCIMIENTO DE PATRONES EN MATLAB

- Por tanto, **datos** y **grupos** deben tener le mismo número de columnas
- Y **grupos** y **nombreGrupos** el mismo número de filas
- Es muy recomendable que, para cada grupo, haya varias muestras de referencia en la variable **grupos**

RECONOCIMIENTO DE PATRONES EN MATLAB

- En un uso avanzado, puede elegirse la función de discriminación empleada

```
resultado = classify (datos, grupos, nombreGrupos, 'funcion')
```

- Se puede escoger entre*: linear, diaglinear, quadratic, diagquadratic* and *mahalanobis*

RECONOCIMIENTO DE PATRONES EN MATLAB

- El método de los K-vecinos más cercanos (o k-nn) es un método de clasificación que permite estimar la función de densidad de las una o varias variables para cada clase de un conjunto
- Este es un método de clasificación no paramétrico calcula la probabilidad de que un elemento pertenezca a una clase, a partir de los patrones ejemplo de la clase

RECONOCIMIENTO DE PATRONES EN MATLAB

- De nuevo, MATLAB compara los datos dados con los diferentes grupos definidos y elige aquel grupo para el que la probabilidad de pertenencia es mayor
- Uso básico

```
resultado = knnclassify (datos, grupos, nombreGrupos)
```

RECONOCIMIENTO DE PATRONES EN MATLAB

- El significado y estructura de los parámetros son los mismos que para el caso anterior
- En un uso avanzado, puede elegirse el número de vecinos empleados y el tipo de distancia considerada

```
resultado = knnclassify (datos, grupos, nombreGrupos, k, ‘distancia’)
```

RECONOCIMIENTO DE PATRONES EN MATLAB

- El tipo de métrica puede escogerse entre: *euclidean, cityblock, cosine, correlation* y *hamming*
- En este caso, el número de patrones por cada clase puede ser cualquiera
  - Incluyendo una única muestra

RECONOCIMIENTO DE PATRONES EN MATLAB

- Las máquinas de vectores de soporte (SVMs) son un conjunto de algoritmos de aprendizaje supervisado desarrollados por Vladimir Vapnik en los laboratorios AT&T.
- Dado un conjunto de muestras de entrenamiento, se entrena una SVM que deduzca la clase de una nueva muestra.

RECONOCIMIENTO DE PATRONES EN MATLAB

- El uso de este procedimiento requiere dos llamadas a función
- Una primera para la creación y entrenamiento

```
maquinaSVM = svmtrain (grupos, nombreGrupos)
```

- Una segunda para realizar el reconocimiento

```
resultado = svmclassify(maquinaSVM, datos)
```

RECONOCIMIENTO DE PATRONES EN MATLAB

- El significado y estructura de los parámetros son los mismos que para los casos anteriores
- Es muy recomendable que, para cada grupo, haya varias muestras de referencia en la variable **grupos**
  - Fase de entrenamiento