Tema 4 *Clustering* y clasificación

Índice

Conceptos preliminares

- Dentro del aprendizaje automático, la clasificación (aprendizaje supervisado) y el clustering (aprendizaje no supervisado) son técnicas que se utilizan para resolver numerosas tareas que pretenden extraer conocimiento del contenido textual

- Cuando nos referimos a clustering o a clasificación de información textual no estructurada ambas tareas intentan realizar algún tipo de organización lógica de dicha información para facilitar posteriores análisis, o son las técnicas de base para dichos análisis

- Ambas tareas tienen como fase inicial la representación del contenido que van a procesar

Conceptos preliminares

- Diferencias entre clasificación automática y *clustering:*

  - ***Clustering*** (agrupamiento): las clases (clusters, grupos) no se conocen de antemano (aplicación de técnicas no supervisadas)

  - **Clasificación**: las clases están previamente definidas (aplicación de técnicas supervisadas)

Índice

- Conceptos preliminares
- ***Clustering***
- Clasificación
- Herramientas
- Bibliografía

Clustering

- Se parte de un conjunto de objetos (documentos) descritos por un conjunto de variables (rasgos, características)

- **Objetivo** del *clustering*: **organizar el conjunto de objetos en grupos** (clusters) tales que los objetos pertenecientes al mismo cluster son más "similares" entre sí de lo que lo son con respecto a los de otros clusters

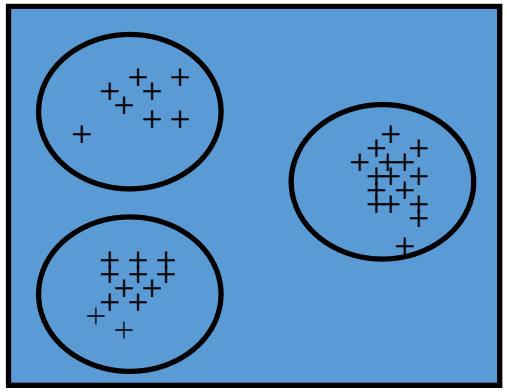
- **Las características de los grupos no se conocen a priori**, hay que descubrir patrones, de ahí que sea una **técnica de aprendizaje no supervisado**

Clustering

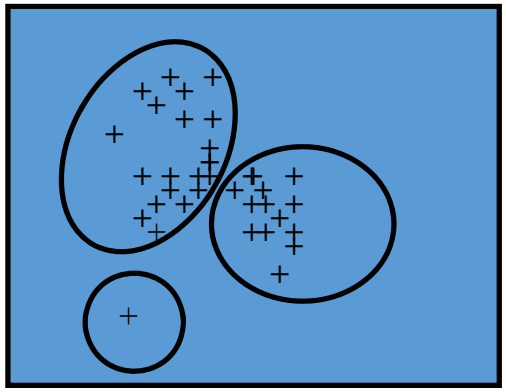
Característica deseable para el clustering:

- Los objetos del mismo cluster deben ser similares entre si (cohesión) y marcadamente diferentes de los de las otras clases (aislamiento)

la cumple



no la cumple



- Los clusters de la figura de la izquierda con cohesionados y están aislados del resto, no ocurre lo mismo en la figura de la derecha

Clustering

- Los mismos documentos se pueden agrupar según diferentes criterios por lo que pueden dar lugar a diferentes agrupaciones (agrupación temática, por extensión, por estructura del contenido, grupos de granularidad alta -pocos grupos-, granularidad baja -muchos grupos con objetos muy cohesionados- …)

- Es importante seleccionar el conjunto de variables (rasgos) relevantes para el objetivo final del clustering

- Entrada clustering:
  - los *n* objetos que se quieren agrupar
  - opcionalmente el valor del nº de clusters o grupos (*k*)

- Salida clustering:
  - los *k* clusters de objetos y/o
  - una organización jerárquica de los objetos

Clustering

Entrada Clustering. Formatos:

- matriz de rasgo-documento (término-documento)
  - o $MR_{n \times p}$ | *n:* nº de objetos (documentos), *p*: nº variables (rasgos)
  - o MR(i,j) valor de la jota-ésima variable del objeto i-ésimo

- matriz de diferencias o matriz de distancias entre objetos (documentos)
  - o $MD_{n \times n}$ / *n:* nº de objetos
  - o MD(i,j) valor de la diferencia entre los objetos *i* y *j*
    - se puede utilizar una matriz de similitud
    - se puede obtener directamente o a partir de la matriz de rasgos utilizando una medida de similitud o distancia entre vectores

Clustering

- En una agrupación temática los textos se suelen caracterizar por las palabras que contienen
  - También podrían utilizarse como rasgos otras unidades lingüísticas como: entidades nombradas, sintagmas, …

- Para la representación de un documento en esta tarea hace falta:
  - Método para calcular el peso (valor) de cada rasgo
  - Función de similitud o distancia

- Algunos algoritmos requieren unas medidas concretas de similitud o distancia, otros no

Métodos de *clustering*: tipos

Tipos de clustering:

- **Agrupamiento duro** (hard): cada objeto pertenece o no a un clúster, pero solo se asigna a uno. También se denomina clustering exclusivo (*exclusive clustering*). Por ejemplo, la salida del algoritmo K-means

- **Agrupamiento suave** (soft): en lugar de colocar cada objeto en un clúster separado, se asigna una probabilidad de que ese punto de datos esté en esos clústeres, o se indica su pertenencia a varios. También se denomina clustering solapado (*overlapping clustering*). Por ejemplo, la salida del algoritmo K-means fuzzy o borroso

Métodos de *clustering*: tipos

De acuerdo a la estructura de clusters que se obtiene, los métodos de clustering pueden ser:

- **No jerárquicos**: la salida son los *n* objetos organizados en *k* grupos sin existir ninguna relación entre los grupos

- **Jerárquicos**: los *n* objetos se presentan organizados en una jerarquía de grupos. Se puede construir la jerarquía de arriba abajo (métodos divisivos) o de abajo a arriba (métodos aglomerativos)

- **Basados en modelos**: los datos se modelan utilizando un modelo estadístico estándar para trabajar con diferentes distribuciones. La idea es encontrar el modelo que mejor se adapte a los datos

Métodos de *clustering*: tipos

Hay numerosos algoritmos de clustering y diferentes propuestas para clasificarlos. Una posible clasificación es la siguiente:

- **Modelos de conectividad**: se basan en la noción de que los puntos de datos más cercanos en el espacio de datos presentan más similitudes entre sí que los puntos de datos más lejanos. A este grupo pertenecen los algoritmos jerárquicos. Estos modelos pueden seguir dos enfoques:

  - Enfoque de abajo a arriba: comienzan con la clasificación de todos los puntos de datos en grupos separados y luego los agregan a medida que disminuye la distancia
  - Enfoque de arriba abajo: todos los puntos de datos se clasifican como un solo grupo y luego se dividen a medida que aumenta la distancia
  - La elección de la función de distancia es subjetiva
  - Son fáciles de interpretar pero carecen de escalabilidad para manejar grandes conjuntos de datos

Métodos de *clustering*: tipos

- **Modelos de centroides:** son algoritmos iterativos de agrupación en los que la noción de similitud se deriva de la proximidad de un punto de datos al centroide (elemento representativo) de los clusters

  - Se debe proporcionar de antemano el número de clusters, lo que hace que sea importante tener un conocimiento previo del conjunto de datos.
  - Estos modelos se ejecutan de forma iterativa para encontrar el óptimo local
  - El algoritmo de agrupación de K-Means entra en esta categoría

Métodos de *clustering*: tipos

- **Modelos distribucionales:** se basan en la noción de cuán probable es que todos los puntos de datos del cluster pertenezcan a la misma distribución (por ejemplo: Normal, Gaussiana)

  - Estos modelos pueden sufrir de overfitting (sobreentrenamiento): cuando se ajustan demasiado a un conjunto particular de datos, y por lo tanto puede fallar cuando se intenten ajustar a datos adicionales o en predecir observaciones futuras
  - Un ejemplo popular de estos modelos es el algoritmo de maximización de expectativas (*Expectation-maximization*) que utiliza distribuciones normales multivariantes
  - Más información: https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm
  - Software: https://scikit-learn.org/stable/modules/mixture.html

Métodos de *clustering*: tipos

- **Modelos basados en densidad:** buscan en el espacio de datos áreas de densidad variada de puntos de datos en el espacio de datos. Aísla varias regiones de densidades diferentes y asigna los puntos de datos dentro de estas regiones en el mismo cluster

  - Ejemplos de este tipo de algoritmos son DBSCAN y OPTICS
  - Software DBSCAN:

    https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html
  - Más información DBSCAN: https://en.wikipedia.org/wiki/DBSCAN

Métodos de *clustering*: tipos

Algoritmos no jerárquicos

- Suponemos conocido el valor de *k* (nº de clusters)

- Objetivo: encontrar una partición en la que los objetos sean similares a los de su mismo cluster y lo sean menos de los de otros clusters

- Enfoque general algoritmos de partición: definir una medida de competencia de una partición y buscar una partición de objetos que optimice dicha medida

- Para un cluster (grupo) de objetos hay que definir medidas de:
  - Su heterogeneidad o falta de cohesión ($H(C_r)$ medida de heterogeneidad del cluster C_r)
  - Su separación del resto de clusters ($I(C_r)$ medida de separación del cluster C_r)

Métodos de *clustering*: tipos

Algoritmos no jerárquicos

- Dadas las medidas de heterogeneidad y separación de cada cluster serán **particiones óptimas** aquellas que **minimizan** las medidas de **heterogeneidad** y **maximizan** medidas de **separación** o combinan ambas

- El problema de encontrar una partición óptima de *n* objetos en *c* clusters puede requerir un alto coste computacional dependiendo del criterio que se utilice

- Algunos planteamientos cuando no se conoce el número de clusters son problemas NP-completos

Métodos de *clustering*: tipos

Algoritmos no jerárquicos. Partición: Algoritmos iterativos de reubicación

- Dado un criterio de competencia de una partición de *n* objetos en *c* clusters disjuntos hay dos fases:

  1. Una partición inicial
  2. Un conjunto de transformaciones para cambiar una partición

- La partición se modifica hasta que ninguna de las posibles transformaciones mejora el criterio de competencia de una partición

Métodos de *clustering*: tipos

Algoritmos no jerárquicos. Partición: Algoritmos iterativos de reubicación

1. Partición inicial. Posibilidades:

- Asignación de objetos a clusters de manera aleatoria
- Utilizar clustering jerárquico previo
- Utilizar una representación espacial, especificar *c* "puntos representativos, centros o centroides" y asignar cada objeto al centroide más cercano.

2. Transformaciones. Por ejemplo:

- Modificación iterativa de una partición asignando cada objeto al grupo cuyo centroide es más cercano y recalculando el centroide del grupo

Métodos de *clustering*: tipos

Algoritmos no jerárquicos. Partición: Algoritmos iterativos de reubicación

Esquema general:

1. Seleccionar C representantes (centroides) generando una partición inicial en c grupos
2. Para cada documento D_i, asignarlo al cluster con el centroide más similar
3. Recalcular todos los centroides
4. Repetir 2 y 3 hasta que no haya modificaciones en la partición en toda una iteración o hasta que se itere un determinado número de veces

Métodos de *clustering*: tipos

Algoritmos no jerárquicos. Partición: Algoritmos iterativos de reubicación

- Los algoritmos que involucran la identificación de un centroide para cada cluster reciben el nombre genérico de ***k*-means** (k-medias) o ***c*-means**
- Centroide: la posición media de todos los puntos en un cluster:
  - Centroide teórico: posición o punto medio
  - Centroide real: objeto del cluster más próximo al centroide teorico
- Hay variaciones de unos algoritmos *k*-means a otros
- Ver animaciones y videos algoritmo k-means:

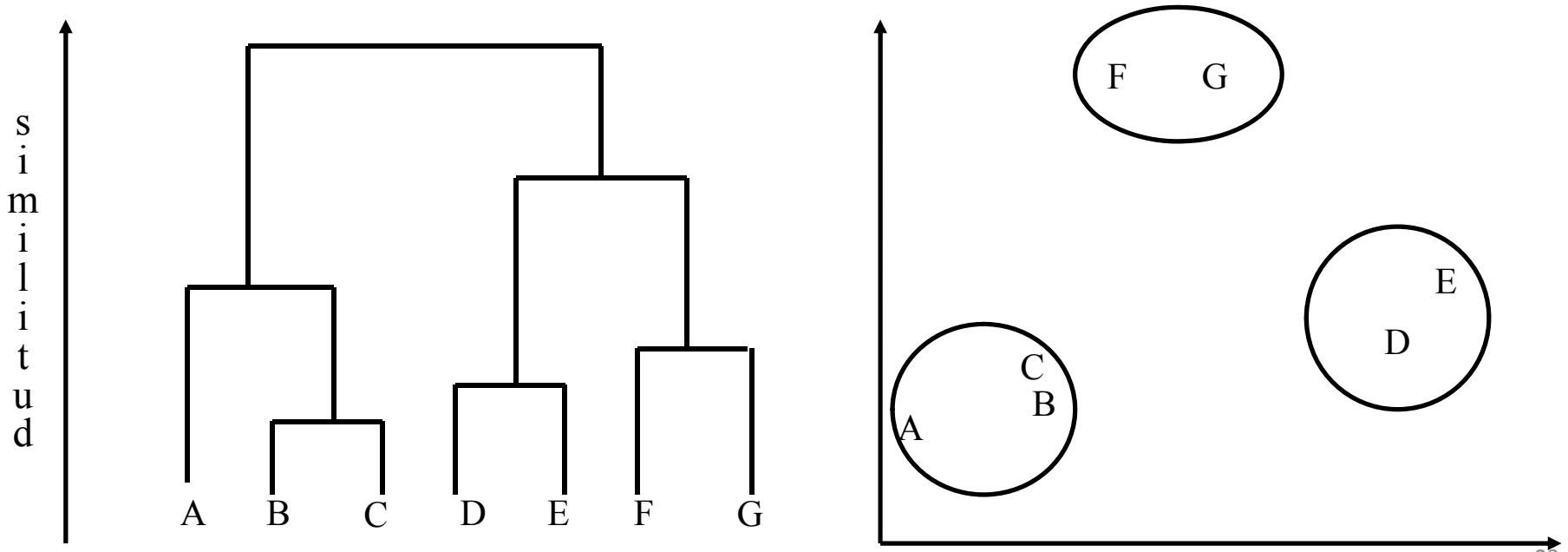
  (http://alekseynp.com/viz/k-means.html)

  (https://www.youtube.com/watch?v=BVFG7fd1H30)
- Hay numerosos tipos de algoritmos no jerárquicos

Métodos de *clustering*: tipos

Algoritmos jerárquicos

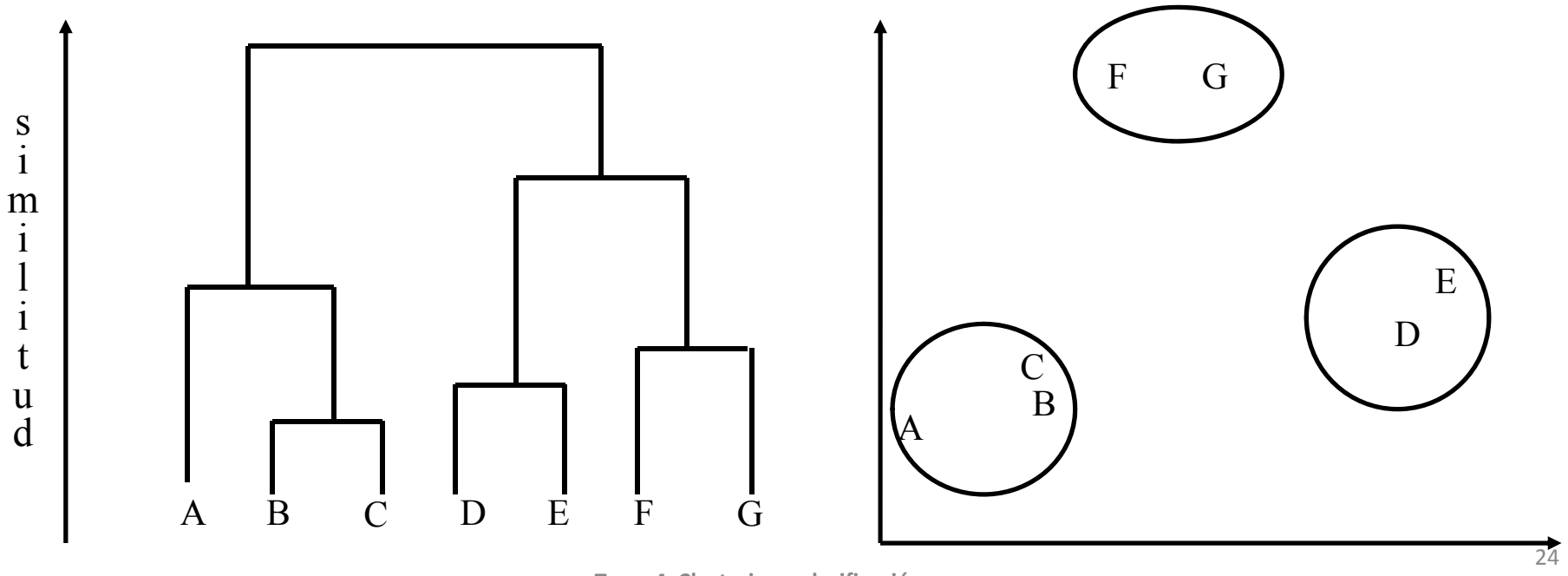
- La salida del clustering es una jerarquía de clusters
- A la jerarquía (árbol) se le denomina **dendograma**
- Permite explorar los clusters a diferente niveles de granularidad o de abstracción. Pueden ser aglomerativos o divisivos



Métodos de *clustering*: tipos

Algoritmos jerárquicos

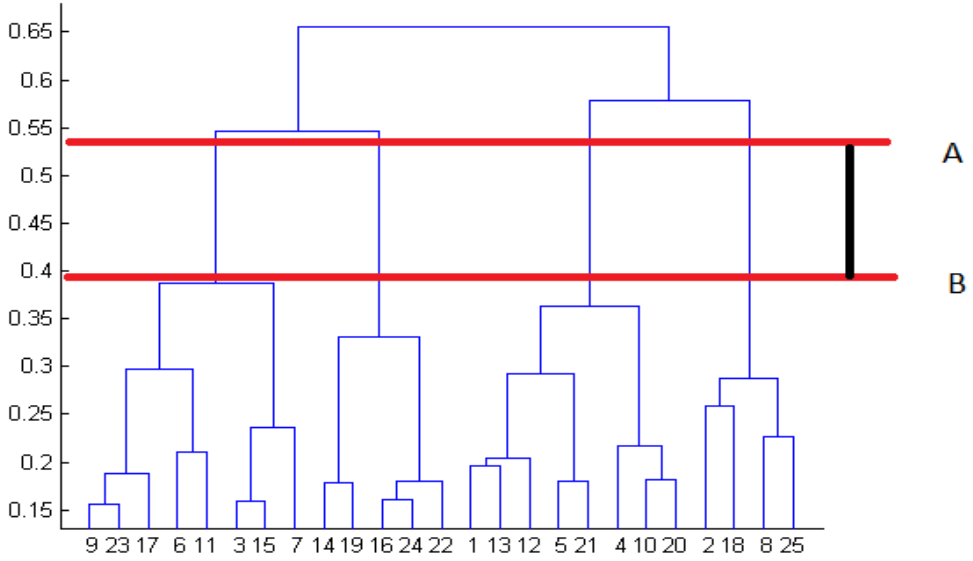
- En el eje X se representan los objetos, el eje Y corresponde a la distancia/similitud en el espacio de datos, por lo que la altura en el dendograma es significativa



Métodos de *clustering*: tipos

Algoritmos jerárquicos

- La decisión del número de clusters se puede elegir observando el dendrograma
- Puede calcularse como el número de líneas verticales en el dendrograma cortadas por una línea horizontal que puede cruzar la distancia máxima verticalmente sin intersectar una agrupación



En el ejemplo la mejor elección del número de clusters será 4

La línea horizontal roja en el dendrograma cubre la distancia vertical máxima AB

Ejemplo sacado de https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/

Métodos de *clustering*: tipos

Algoritmos jerárquicos: aglomerativos

- Comienzan con tantos clusters como objetos
- En cada paso, los clusters más similares se unen (la similitud depende del criterio, medida)
- La unión de clusters continúa hasta que todos los objetos pertenecen a un único cluster
- Los algoritmos más utilizados unen pares de clusters dando lugar a árboles binarios
- La distancia entre objetos individuales se generaliza a la distancia entre subconjuntos
- Esta medida de proximidad se denomina: métrica de enlace o de conexión (*linkage metric*)

Métodos de *clustering*: tipos

Algoritmos jerárquicos: aglomerativos

- Las principales métricas de enlace o de conexión son:
  - Enlace simple (*single link*)
  - Enlace promedio (*average link*)
  - Enlace completo (*complete link*)

$$d(C_1, C_2) = \text{operación}\{d(x,y) \mid x \in C_1, y \in C_2\}$$

- Enlace simple (*single link*): operación = mínimo
- Enlace promedio (*average link*): operación=promedio
- Enlace completo (*complete link*): operación= máximo

Métodos de *clustering*: tipos

Algoritmos jerárquicos: aglomerativos

Esquema general:

1. Calcular la matriz de distancias y tratar cada objeto como un cluster
2. Encontrar el par de clusters (objetos al principio) más similares, los une en uno y actualiza la matriz de distancias para reflejar esta operación
3. Si solo queda un cluster parar, sino continua con el paso 2

Para más información*: Hierarchical Agglomerative Clustering* (*HAC)* https://nlp.stanford.edu/IR-book/html/htmledition/hierarchical-agglomerative-clustering-1.html

Métodos de *clustering*: tipos

Algoritmos jerárquicos: divisivos

- Comienzan con un único cluster que contiene a todos los objetos
- En cada paso, se divide un cluster
- La división de clusters continúa hasta que todos los clusters contienen un sólo objeto
- Los algoritmos más utilizados dividen un cluster en dos dando lugar a árboles binarios (pueden no ser adecuados)
- La agrupación de arriba hacia abajo es conceptualmente más compleja que la agrupación de abajo hacia arriba, ya que necesita un segundo algoritmo de agrupación para llevar a cabo las divisiones
- Tiene la ventaja de ser más eficiente si no generamos una jerarquía completa hasta las hojas de documento individuales
- Los aglomerativos toman decisiones de agrupación basadas en patrones locales sin tener en cuenta inicialmente la distribución global. Los divisivos disponen de la información completa sobre la distribución global cuando se toman decisiones de partición de alto nivel

Métodos de *clustering*: tipos

Métodos jerárquicos. Algoritmos divisivos

Esquema general:

1. Se selecciona un cluster para dividir. Posibilidades:
   - Seleccionar el cluster con más objetos
   - Seleccionar el cluster que al dividirlo genere una solución que optimice una función criterio
2. Se realiza la partición de dicho cluster normalmente en 2
3. Si quedan clusters con más de 1 objeto continúa el paso 1

Métodos de *clustering*: tipos

Topic Model

- Es un modelo estadístico para descubrir los “temas” sobre los que versa una colección de documentos y permite realizar clustering
- Se asume que si un documento trata sobre un tema habrá palabras que aparecerán con mayor o menor frecuencia según estén relacionadas o no con dicho tema
- Como un documento puede tratar varios temas, podrá versar un X% sobre un tema, un Y% sobre otro, …
- Dada una colección de documentos, este modelo permite determinar los temas de los que tratan y para cada documento su concreta porporción de dichos temas

Métodos de *clustering*: tipos

Topic Model: Latent Dirichlet Allocation (LDA) [Blei et al. 2003]

- Es un modelo estadístico generativo
- En LDA cada documento se ve como una mixtura de varios temas y el modelo es capaz de asignárselos
- Una palabra puede aparecer en varios temas con diferente probabilidad y con diferentes palabras vecinas
- Requiere que el número del grupos sea conocido
- En http://videolectures.net/mlss09uk_blei_tm/ hay una video-clase de uno de los creadores del modelo explicándolo
- Información de la implementación de LDA en Gensim: https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/

Métodos de *clustering*: evaluación

Medidas de evaluación de clustering

Hay dos tipos de medidas de evaluación:

- **Internas o intrínsecas**: se basan en los propios objetos agrupados. Estos métodos suelen asignar mejor puntuación al algoritmo que produce clústeres con alta similitud dentro de un clúster y baja similitud entre clústeres

- **Externas o extrínsecas**: se basan en determinar cómo se ajustan los clusters creados por el algoritmo a un conjunto de clusters referencia de evaluación, benchmark o gold standard. Este gold standard idealmente lo realizan jueces humanos, más de uno, con un buen nivel de acuerdo entre ellos. Entonces podemos calcular un criterio externo que evalúa cómo de bien la agrupación se ajusta a las clases del gold standard

Métodos de *clustering*: evaluación

Medidas de evaluación de clustering: Internas o intrínsecas

- Permiten escoger el mejor algoritmo de clustering a partir de la información de los propios datos, así como el número de clúster óptimo sin ningún tipo de información adicional
- Una desventaja de este tipo de evaluación es que las puntuaciones altas en una medida interna no necesariamente resultan en aplicaciones efectivas para el usuario
- Esta evaluación está sesgada hacia algoritmos que utilizan el mismo modelo o función de optimización, por ejemplo k-means
- Ejemplo de medidas:
  - Medidas que calculan la cohesión y separación de clusters, como el índice **Silhouette** (https://en.wikipedia.org/wiki/Silhouette_(clustering))
  - Índice **Davies–Bouldin (**https://en.wikipedia.org/wiki/Davies%E2%80%93Bouldin_index**)**

Métodos de *clustering*: evaluación

Medidas de evaluación de clustering: Externas o extrínsecas

- Las colecciones de referencia o gold estándar son escasas y no cubren todos los campos de aplicación del clustering debido al esfuerzo que supone realizar una agrupación manual

- Algunas medidas son:

  - **Precisión, cobertura** (recall) y la media armónica entre ellas **medida-F** (ver capítulo 1)
  - El cálculo de la precisión y la cobertura se puede realizar de dos maneras diferentes:
    - ***Microaveraging***: los cálculos se basan en la suma de todos los valores individuales
    - ***Macroaveraging***: los cálculos se basan en la suma de los valores por categoría, obteniendo después la media total
  - **Índice de Rand** ajustado (Adjusted Rand index): es una función que mide la similitud de las dos asignaciones, ignorando las permutaciones (https://en.wikipedia.org/wiki/Rand_index)

Métodos de *clustering*: herramientas

Algunas otras herramientas disponibles para clustering:

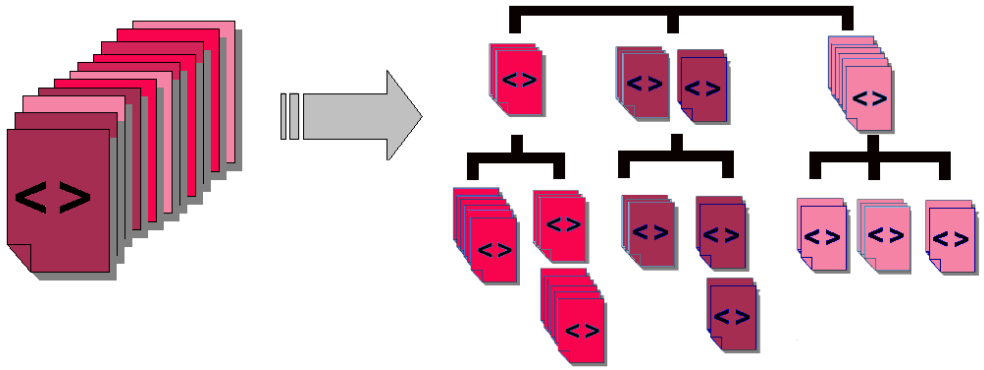
- CLUTO (http://glaros.dtc.umn.edu/gkhome/views/cluto/)
- Text-Garden (http://ailab.ijs.si/dunja/textgarden/)
- Weka (https://www.cs.waikato.ac.nz/ml/weka/)
- jLDADMM: un paquete Java para topic model en textos normales y cortos (https://sourceforge.net/projects/jldadmm/)
- Twitter-LDA (https://github.com/minghui/Twitter-LDA)

Índice

- Conceptos preliminares
- *Clustering*
- **Clasificación**
- Herramientas
- Bibliografía

Clasificación

La **clasificación automática de documentos** es una tarea en la que un documento, o una parte del mismo, se etiqueta como perteneciente a un determinado conjunto, grupo o categoría predeterminada



Clasificación

- Un sistema de clasificación debe asignar un valor booleano ***T*** (true) o ***F*** (false) para cada par (d_j , c_i), siendo d_j un documento de la colección y c_i una de las categorías entre las que se desee clasificar $\mathbf{C} = \{c_1 , c_2 , .. , c_j\}$

- La clasificación desde los años 90 utiliza el paradigma del Aprendizaje Automático Supervisado que trata de obtener las características que debe cumplir un documento para que sea clasificado en una u otra categoría, basándose para ello en una colección inicial de documentos preclasificados

Clasificación

- Un ***descriptor de clase*** contiene las características que debe cumplir un documento para ser clasificado en una determinada clase

- Una vez que se han representado los documentos y se han obtenido los *descriptores de clase* para cada una de las clases consideradas (por medio del proceso de aprendizaje), el sistema utilizará estos descriptores para crear un clasificador, siendo capaz de clasificar nuevos documentos

- Se debe disponer de un conjunto de documentos de ejemplo para cada una de las categorías consideradas

- Cuanto mayor sea el conjunto de datos etiquetados mayor será la información potencial disponible y, previsiblemente, mejor resultará el aprendizaje

Clasificación

- En el **aprendizaje semisupervisado** se utilizan datos no etiquetados para refinar una estimación inicial obtenida a partir de los datos etiquetados con la categoría correspondiente de los que se disponga

- Las técnicas de *aprendizaje supervisado* y s*emisupervisado* centran su aprendizaje en el corpus del que se dispone, que se divide al menos en dos subcolecciones:

  - **Subcolección de entrenamiento**: documentos utilizados para analizar las características de cada clase (hallar los descriptores de clase) y poder crear el clasificador

  - **Subcolección de prueba o test**: para comprobar la calidad de los clasificadores generados

Clasificación

- Dentro del aprendizaje semisupervisado se podría ubicar la **supervisión distante** (*distant supervision*)

- En este tipo de supervisión se dispone de un conjunto de entrenamiento etiquetado generalmente pequeño, y un conjunto más grande de documentos no etiquetados

- Se dispone además de un operador (regla, heurística, base de conocimiento, …) que permite etiquetar una muestra de la colección no etiquetada asumiendo que habrá ruido (errores) en el etiquetado. El algoritmo de clasificación con ambos conjuntos etiquetados decide la etiqueta para un nuevo ejemplo

Clasificación

Aplicaciones de la clasificación automática supervisada en minería de textos:

- Reconocimiento y clasificación de entidades nombradas
- Etiquetado morfosintáctico y sintático
- Desambiguación del sentido de la palabras
- Análisis de sentimientos (identificación de la polaridad)
- Clasificación temática de documentos
- Detección de spam
- Filtrado de documentos
- …

Clasificación: tipos

Clasificación ***singlelabel*/ *multilabel***

- **Clasificación singlelabel** cuando cada uno de los documentos de la colección debe tener una y sólo una categoría asignada
- **Clasificación multilabel** cuando el número de categorías puede variar desde 0 hasta el número total de clases

Un caso concreto de clasificación *singlelabel* es la clasificación binaria, asignando un T o F a cada documento (una aplicación de análisis de polaridad positiva o negativa asignaría T a un documento si es positiva y F en caso de que no lo sea)

La clasificación binaria se puede utilizar también en *multilabel*, asignando un T o F para cada par documento-categoría

Clasificación: tipos

Clasificación ***hard / ranking classification***

- Para establecer si un documento pertenece a una categoría:
  - Se puede decidir directamente si pertenece o no a una categoría (***hard classification***) obteniéndose un conjunto no ordenado de los documentos que pertenecen a cada categoría
  - Se puede generar un ranking de categorías según la adecuación de cada documento a cada categoría (***ranking classification***)

Clasificación: tipos

Clasificación ***fastfeature / fullfeature classification***

- Dependiendo del tipo de información utilizada en el proceso de clasificación se distingue entre:
  - **Fastfeature:** cuando no se tiene en cuenta información externa al documento
  - **Fullfeature:** cuando se emplean características adicionales, externas al documento, como puede ser el texto de los enlaces salientes

Clasificación: tipos

Hay numerosos algoritmos de clasificación pero vamos a presentar solo los más conocidos:

- Clasificadores probabilísticos (Naïve Bayes)
- Árboles de decisión
- Máquinas de vectores de soporte (*Support Vector Machine*)
- Basados en redes neuronales artificiales
- Basados en aprendizaje profundo

Clasificación: tipos

Hay numerosos algoritmos de clasificación pero vamos a presentar solo los más conocidos:

- Clasificadores probabilísticos (Naïve Bayes)
- Árboles de decisión
- Máquinas de vectores de soporte (*Support Vector Machine*)
- Basados en redes neuronales artificiales
- Basados en aprendizaje profundo

Clasificación: algoritmos

Clasificador probabilístico Naïve Bayes

- Está basado en la teoría de la decisión de Bayes: la **teoría de las probabilidades condicionadas**

- El problema de la clasificación se reduce al cálculo de las probabilidades a posteriori de una clase dado un documento

$$P(c_k \mid \vec{d_j}) = \frac{P(\vec{d_j} \mid c_k)P(c_k)}{P(\vec{d_j})}$$

- La tarea de un clasificador Naïve Bayes es elegir la clase que haga mayor esta probabilidad

Clasificación: algoritmos

Clasificador probabilístico (Naïve Bayes)

- Deben estimarse primero otras cantidades como son:
  - La probabilidad a priori de cada clase, $P(c_k)$,
  - La probabilidad a priori del documento, $P(d_j)$;
  - La probabilidad condicionada de cada documento a una clase dada, $P(d_j | c_k)$.

$$P(c_k \mid \vec{d_j}) = \frac{P(\vec{d_j} \mid c_k)P(c_k)}{P(\vec{d_j})}$$

- La asunción del principio de independencia entre rasgos implica que la probabilidad de un documento dada una clase sea el producto de las probabilidades condicionada de cada rasgo presente en el documento

$$P(\vec{d_j} \mid c_k) = \prod_{i=1}^{N_j} P(\vec{t_i} \mid c_k)$$

donde N_j es el número de rasgos presentes en d_j

Clasificación: algoritmos

Clasificador probabilístico Naïve Bayes

- La diferencia entre dos algoritmos Naïve Bayes vendrá por la diferente función que se emplee para la estimación de la probabilidad de un rasgo a una clase: $P(t_i|c_k)$

- Algunas de las funciones más usadas en la literatura son las gaussianas y multinomiales

Clasificación: algoritmos

Árboles de decisión

- Un **árbol de decisión** es un árbol cuyos nodos internos representan términos (rasgos), las ramas salientes los pesos de éstos, y las hojas se corresponden con las categorías
- Se recorre el árbol de arriba a abajo para cada uno de los documentos, hasta llegar a una de las hojas, es decir, hasta asignar una categoría.
- En estas estructuras de árbol las hojas representan etiquetas de clase y las ramas representan conjunciones de características que conducen a esas etiquetas de clase
- Los árboles de decisión en los que la variable objetivo puede tomar valores continuos (normalmente números reales) se denominan **árboles de regresión**
- Algunos algoritmos: ID3 (Iterative Dichotomiser 3), C4.5 (sucesor de ID3), CART (Classification And Regression Tree), Chi-square automatic interaction detection (CHAID), MARS

Clasificación: algoritmos

Máquinas de vectores de soporte (*Support Vector Machine - SVM*)

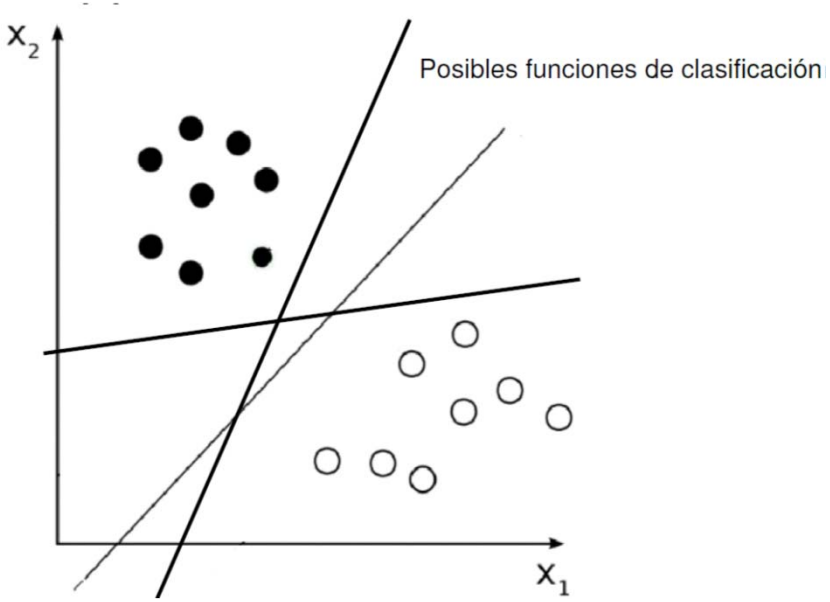
- Thorstem Joachims utilizó los SVM por primera vez en clasificación automática
- Se construyen en espacios de dimensión T, siendo |T| el número de términos existentes en nuestra colección
- El objetivo es obtener una hipersuperficie que separe el conjunto de ejemplos positivos de los negativos para cada categoría y en ese espacio de dimensión |T|
- Para establecer estas hipersuperficies de decisión se utiliza una colección de ejemplos de entrenamiento, denominados **vectores de soporte**
- De todas las posibles hipersuperficies de decisión se escoge aquella que tenga mayor margen sobre las clases.

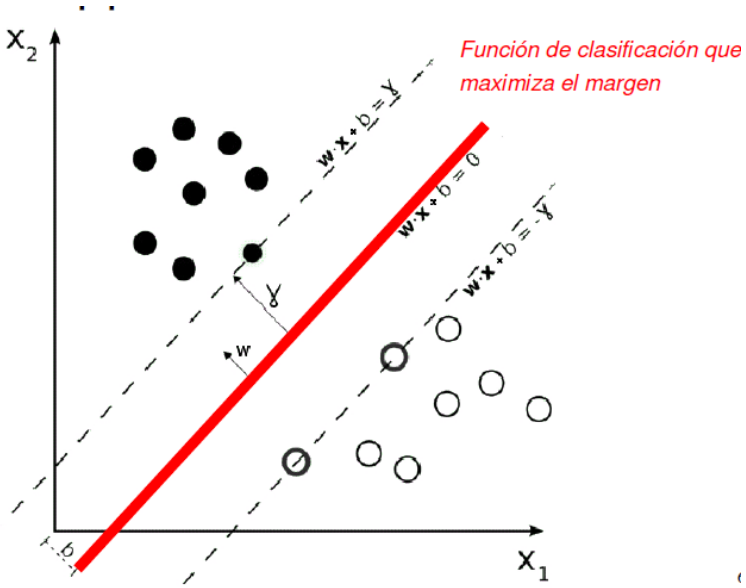
Clasificación: algoritmos

Máquinas de vectores de soporte (*Support Vector Machine - SVM*)

- Para el caso de un problema con dos categorías separables linealmente, la función que decida la clasificación de cada instancia será del siguiente tipo:

$$f(x) = w \cdot x + b$$





Clasificación: algoritmos

Máquinas de vectores de soporte (*Support Vector Machine*)

- Aunque el anterior ejemplo muestre un problema lineal, las máquinas de vectores de soporte pueden utilizarse también cuando las categorías no son linealmente separables utilizando una **función de kernel** que transforme el espacio |T| dimensional en un espacio con dimensión |T'| > |T| donde el problema sí resulte ser linealmente separable
- Entre las funciones de kernel más usadas podemos encontrar:

Lineal: $K(x_i, x_j) = x_i^T \cdot x_j$

Polinomial: $K(x_i, x_j) = (\gamma \cdot x_i^T \cdot x_j + r)^d, \gamma > 0$

Radial Basis Function (RBF): $K(x_i, x_j) = e^{-\gamma \cdot \|x_i - x_j\|^2}, \gamma > 0$

Sigmoidea: $K(x_i, x_j) = \tanh(\gamma \cdot x_i^T \cdot x_j + r)$

r,d y ɣ son parámetros configurables para cada kernel, siendo d un valor entero y el resto reales.

Clasificación: algoritmos

Basados en redes neuronales artificiales

- Son una clase de algoritmos de reconocimiento de patrones que se utilizan comúnmente para problemas de regresión y clasificación

- Están formadas por nodos, neuronas, que toman un valor real, lo multiplican por un peso y lo hacen funcionar mediante una función de activación no lineal. Al construir múltiples capas de neuronas, cada una de las cuales recibe parte de las variables de entrada, y luego transmite sus resultados a las capas siguientes, la red puede aprender funciones muy complejas

- Teóricamente, una red neuronal es capaz de aprender la forma de cualquier función, dada la suficiente potencia computacional

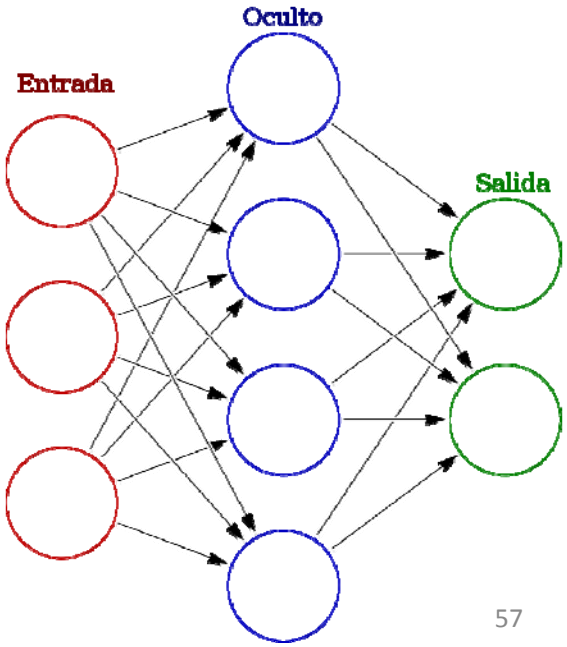
Clasificación: algoritmos

Basados en redes neuronales artificiales

- Las **redes neuronales** son un modelo computacional que consiste en un conjunto de unidades, llamadas neuronas, conectadas entre sí para transmitirse señales. La información de entrada atraviesa la red neuronal, se opera con ella y se producen unos valores de salida

- Cada neurona está conectada con otras a través de unos enlaces en los que el valor de salida de la neurona anterior es multiplicado por un valor de peso que pueden incrementar o inhibir el estado de activación de las neuronas adyacentes.
- A la salida de la neurona, puede existir una función limitadora o umbral, que modifica el valor resultado o impone un límite que se debe sobrepasar antes de propagarse a otra neurona. Esta función se conoce como función de activación.

(https://es.wikipedia.org/wiki/Red_neuronal_artificial)



Clasificación: algoritmos

Basados en redes neuronales artificiales

- Teóricamente, una red neuronal es capaz de aprender la forma de cualquier función, dada la suficiente potencia computacional

- Algoritmos populares de este tipo son:
  - Perceptron
  - Multilayer Perceptrons (MLP)
  - Back-Propagation
  - Stochastic Gradient Descent
  - Hopfield Network
  - Radial Basis Function Network (RBFN)

Clasificación: algoritmos

Basados en aprendizaje profundo (deep learning)

- Los métodos de Aprendizaje Profundo son una actualización moderna de las Redes Neurales Artificiales que explotan la facilidades de computación existentes en la actualidad

- Consisten en redes neuronales mucho más grandes y complejas de varias capas.

- Algoritmos populares de este tipo son:
  - Convolutional Neural Network (CNN)
  - Recurrent Neural Networks (RNNs)
  - Long Short-Term Memory Networks (LSTMs)
  - Stacked Auto-Encoders
  - Deep Boltzmann Machine (DBM)
  - Deep Belief Networks (DBN)

Clasificación: algoritmos

Basados en aprendizaje profundo (deep learning): *transformers*

- En el año 2017 apareció el modelo *Transformer* [Polosukhin et al. 2017] basado en redes profundas mucho más potentes y flexibles que las arquitecturas de redes profundas anteriores

- Un *transformer* es básicamente un modelo que recibe un texto y produce otro texto. La aplicación más natural de esta idea es la traducción automática, pero se puede generalizar para utilizarse en muchas otras tareas de PLN

- La arquitectura consiste en una secuencia de *encoders* y *decoders*, a través de los cuales van fluyendo representaciones intermedias del texto de entrada, hasta producir el texto de salida

- Los *encoders* y *decoders* son redes neuronales con una capa especial, denominada de auto-atención, que permite determinar qué partes de la frase son las más relevantes y aprender relaciones de contexto entre las palabras

Clasificación: algoritmos

Basados en aprendizaje profundo (deep learning): *transformers*

- Gracias a las diferentes capas de auto-atención, el procesamiento del texto se realiza de forma independiente al orden de las palabras, lo que supone un mecanismo muy potente para resolver fenómenos lingüísticos complejos como, por ejemplo, la correferencia

- A finales de 2019 Google publicó T5, acrónimo de "*Text-to-Text Transfer Transformer*", un *framework* basado en *transformers* preparado para adaptar cualquier tarea de PLN y resolverla desde la perspectiva *text-to-text* de los *transformers*.

- Además de la traducción automática (que de forma natural es *text-to-text*) pueden beneficiarse de la potencia de estos modelos pre-entrenados casi cualquier tarea de procesamiento de textos, incluidas las basadas en clasificación

Clasificación: algoritmos

Basados en aprendizaje profundo (deep learning): *transformers*

- Bert (Bidirectional Encoder Representations from Transformers) y GPT (Generative Pre-trained Transformer) son sistemas prentrenados basados en *transformers* con corpora enormes que se pueden ajustar a tareas específicas de PLN

- Los modelos GPT-n se encuentran basados en la arquitectura de *transformers:* GPT-2 (Radfors et al. 2019); GPR-3 (Brown et al. 2020)

Clasificación: evaluación

- Se compara la salida del sistema con un conjunto de referencia o gold standard
- A través de una tabla de contingencia se pueden definir las siguientes funciones de evaluación para todo sistema de clasificación

	Datos Positivos	Datos Negativos
Asignaciones Positivas	VP	FP
Asignaciones Negativas	FN	TN

VP (verdaderos positivos); VN (verdaderos negativos); FP (falsos positivos): FN (falsos negativos)

- La exactitud –*accuracy*- (Â) y el error (Ê)

$$\hat{A} = \frac{(TP+TN)}{(TP+TN+FP+FN)}$$

$$\hat{E} = 1 - \hat{A} = \frac{(FP+FN)}{(TP+TN+FP+FN)}$$

- También se utilizan la Precisión, Cobertura y medida-F

Índice

- Conceptos preliminares
- *Clustering*
- Clasificación
- **Herramientas**
- Bibliografía

Herramientas clustering y clasificación

- Lista de software para minería de textos: https://en.wikipedia.org/wiki/List_of_text_mining_software
- En Data Mining Free Tools hay una listado de herramientas de data mining que incuyen algoritmos de clasificación supervisada (https://sci2s.ugr.es/keel/links.php#sub10)
- Top 27 Free Software fot Text Analysis, text Mining, Text Analytics (https://www.predictiveanalyticstoday.com/top-free-software-for-text-analysis-text-mining-text-analytics/)
- Librerías para clustering y clasificación: Matlab, R, Weka, Gensim, scikit-learn
- Plataformas de deep learning:
  - TensorFlow (Google): https://www.tensorflow.org/
  - Keras API de redes neuronales de alto nivel, escrita en Python, capaz de funcionar sobre TensorFlow, CNTK o Theano. (https://keras.io/)
  - PyTorch (Facebook): https://pytorch.org/
- Transformers: https://huggingface.co/transformers/
- GPT-2: https://openai.com/blog/gpt-2-1-5b-release/

Índice

- Conceptos preliminares
- *Clustering*
- Clasificación
- Herramientas
- **Bibliografía**

Bibliografía

- [Blei et al. 2003] D. M. Blei; A. Y. Ng; Michael I. Jordan. Latent Dirichlet Allocation. Journal of Machine Learning Research 3 (2003) 993-1022.
- [Polosukhin et al. 2017] Polosukhin, Illia; Kaiser, Lukasz; Gomez, Aidan N.; Jones, Llion; Uszkoreit, Jakob; Parmar, Niki; Shazeer, Noam; Vaswani, Ashish (2017-06-12). "Attention Is All You Need". arXiv:1706.03762 [cs.CL].
- [Radfors et al. 2019] Alec Radford; Jeffrey Wu; Rewon Child; David Luan; Dario Amodei; Ilya Sutskever . Language Models are Unsupervised Multitask Learners. https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf
- [Brown et al. 2020] Brown, Tom B.; Mann, Benjamin; Ryder, Nick; Subbiah, Melanie; Kaplan, Jared; Dhariwal, Prafulla; Neelakantan, Arvind; Shyam, Pranav; Sastry, Girish; Askell, Amanda; Agarwal, Sandhini; Herbert-Voss, Ariel; Krueger, Gretchen; Henighan, Tom; Child, Rewon; Ramesh, Aditya; Ziegler, Daniel M.; Wu, Jeffrey; Winter, Clemens; Hesse, Christopher; Chen, Mark; Sigler, Eric; Litwin, Mateusz; Gray, Scott; Chess, Benjamin; Clark, Jack; Berner, Christopher; McCandlish, Sam; Radford, Alec; Sutskever, Ilya; Amodei, Dario. Language Models are Few-Shot Learners. (22 de julio de 2020) https://arxiv.org/abs/2005.14165