Universidad Rey Juan Carlos

Fundamentos matemáticos en clasificación y predicción de vulnerabilidades en ciberseguridad basada en técnicas de Machine Learning

Trabajo de Fin de Grado

Escuela Técnica Superior de Ingeniería Informática

Grado en Matemáticas

Curso 2016-2017

Sergio Michiels Mangas

Tutor:

Miguel Romance Del Río

Índice general

Capítulo 1

Presentación - Motivación práctica y descripción

En este proyecto se van a estudiar los fundamentos matemáticos de algunos algoritmos de aprendizaje automático no supervisado y sistemas de recomendación clásicos. Se enunciarán los algoritmos, la base matemática que llevan por detrás y se comentarán las variantes que existen de los mismos. Se mostrará cómo funcionan estos algoritmos, y en algunos casos se demostrarán los teoremas que los fundamentan. Se ha escogido este tema por los siguientes motivos:

- Su actualidad. El *Big Data* es un concepto que hace referencia al almacenamiento de grandes cantidades de datos y a los procedimientos y algoritmos utilizados para extraer patrones o información de dichos datos. Cada año se actualizan las previsiones sobre el volumen de datos que se podrán tratar en el futuro, y los números no paran de crecer a ritmo exponencial. En 2013 se hablaba de 4,4 billones de gigabytes, y esa cifra puede multiplicarse por diez en tan solo seis años [4]. La velocidad en el acceso, flujo y manejo de datos; la variabilidad de los datos, si están estructurados o no y necesitan preprocesamiento; y por supuesto el volumen de los propios datos, son las tres Vs del Big Data [3], y los desafíos a los que hoy en día y en el futuro habrá que hacerles frente.

- El aprendizaje automático es otro concepto que está muy relacionado con el Big data y también está muy presente hoy en día. Se entrará más en detalle en el capítulo 2, pero existen numerosos algoritmos de aprendizaje automático, supervisados y no supervisados, orientados a sets de datos más pequeños o no, etc. Además, hay muchísimas publicaciones sobre el diseño de nuevos algoritmos o pequeñas mejoras

para algoritmos ya existentes. Se dan conferencias de numerosos temas relacionados con ellos, incluso hay grandes empresas de todos los sectores que están interesadas en estos temas, como *Netflix* o *Amazon* [2]. Junto con el Big data, ambos son temas de investigación, donde encontrar y proponer nuevas alternativas para ciertos problemas siempre es bienvenido. Existen proyectos o concursos como el *Netflix Prize* [18] que incentivan la investigación y el desarrollo de nuevas técnicas de aprendizaje automático.

- La gran cantidad de algoritmos que existen. Se pueden hacer numerosas taxonomías sobre los algoritmos de aprendizaje automático: si necesitan datos ya etiquetados o no, si clusterizan un conjunto de datos de forma aglomerativa, divisiva, en clusters independientes, etc. Además, es interesante estudiar los algoritmos ya existentes y sus posibles mejoras, o incluso el desarrollo de nuevos algoritmos. En el *Netflix Prize* mencionado anteriormente se consiguió mejorar el algoritmo de Netflix en un 10 %, y el premio para los ganadores fue de un millón de dólares. Además, hay conferencias muy importantes a lo largo del planeta que tratan exclusivamente temas de Big data, minería de datos y Machine Learning, y numerosos artículos de investigación científica con nuevas tendencias o propuestas [26].

Considero que estos temas son muy interesantes y que tienen una gran demanda en el panorama profesional actual. Creo que no basta con saber utilizar una serie de algoritmos, sino que hay que entenderlos y saber cómo funcionan. Además, este Trabajo de Fin de Grado me servirá como base matemática para el otro Trabajo de Fin de Grado de mi doble titulación, en el que emplearé algunos de los algoritmos descritos aquí.

El proyecto se estructura de la siguiente manera:

- Primero se hará una breve introducción al Machine Learning.
- A continuación se enunciarán y estudiarán algunos algoritmos de clusterización no supervisada. K-means, clustering jerárquico y PCA.
- Después se hará una pequeña introducción sobre los sistemas de recomendación. Qué hacen, la importancia que tienen y cómo se clasfican.
- Luego se enunciará uno de los sistemas de recomendación más clásicos que existen, *Collaborative filtering* y algunos de sus diferentes enfoques, *user based filtering*, *item based filtering* y *personality diagnosis*.
- Finalmente se dedicará un apartado a las conclusiones y al trabajo futuro.

Capítulo 2

Introducción al Machine Learning

El aprendizaje es el proceso en el que se construye un modelo científico tras descubrir información o conocimientos a partir de una serie de datos. [10]. Generalmente, se denomina Machine Learning a la rama de la inteligencia artificial que busca estudiar y construir algoritmos que permitan a las máquinas aprender y hacer predicciones sobre datos a partir de un modelo construido con una serie de entradas.

La ciencia del aprendizaje y el Machine Learning tienen un papel muy importante en campos como la estadística, la medicina, la minería de datos y muchas otras disciplinas e ingenierías. Gracias al Machine Learning se pueden resolver problemas como [12] :

- Predecir si un paciente de un hospital que ha sufrido un infarto puede sufrir un segundo infarto.
- Estimar la cantidad de glucosa en sangre que tenga un paciente diabético a partir de un espectro infrarrojo.
- Identificar los factores de riesgo más importante para ciertas enfermedades.
- Identificar caracteres o números escritos a mano o en carteles.

Tradicionalmente [10], los algoritmos de Machine Learning se catalogan en varios grupos:

- **Basados en símbolos**: Suponen que toda la información se puede representar con símbolos, y que se pueden crear nuevos símbolos y conocimiento basado en dichos símbolos, tomando decisiones a partir de lógica e inferencia.
- **Basados en conexiones**: Intentan imitar los sistemas neuronales que tenemos en nuestro cerebro. Las decisiones se toman tras haber entrenado el sistema y una vez que se reconocen ciertos patrones.
- **Basados en comportamiento**: Se basan en la suposición de que existen soluciones para problemas de comportamiento, y toman las decisiones que creen mejor para resolver los problemas que se presentan.
- **Basados en el sistema inmunitario**: El sistema aprende de los encuentros o datos que obtuvo en el pasado y desarrolla la habilidad de detectar patrones similares en los datos que vaya recibiendo.

Ninguno de estos grupos es mejor que el otro, sino que depende de la situación concreta en la que nos encontremos. Por tanto, no es necesario seleccionar un método basándonos en estas pequeñas diferencias, y durante el proceso de Machine Learning, se crean modelos matemáticos para describir los datos que se nos presenten.

Los algoritmos de Machine Learning tienen que ser probados empíricamente, ya que su rendimiento depende principalmente del tipo de entrenamiento que se siga, de la propia métrica que elijamos para medir su rendimiento, y del problema a resolver. Se suelen evaluar comparando los resultados de emplear diferentes algoritmos sobre un mismo set de datos, o bien utilizando el mismo algoritmo en diferentes sets de ejemplo de los datos.

El objetivo es entrenar de la mejor manera posible el algoritmo para reducir el error que se pueda cometer en las clasificaciones de nuestro sistema. Además de contrastar algoritmos y procedimientos, la clave para un buen entrenamiento es la información, los datos.

Dependiendo de los datos de los que se disponga, de si están ya etiquetados o no, de su distribución, así como del resultado que se quiera obtener, podemos clasificar los algoritmos de Machine Learning en dos grupos:

- **Aprendizaje supervisado**: Los datos son pares de información-etiqueta, es decir, ya conocemos el grupo o el tipo al cual pertenece ese dato. Dichos datos se utilizan para entrenar una función, y se crea un modelo de aprendizaje que intenta que la función de salida pueda predecir al mínimo coste. Los métodos más populares son las SVMs (Máquinas de Soporte Vectorial), las redes neuronales o los árboles de decisión.

- **Aprendizaje no supervisado**: Los datos no poseen ninguna clase o etiqueta, sólo está la información. El objetivo de estos algoritmos es detectar las características clave de los datos y formar grupos (o clusters) de la manera más natural posible, dada una función de coste a minimizar. Los métodos más populares son *K-means*, clustering jerárquico y similares.

Concretamente en este Trabajo de Fin de Grado vamos a discutir algunos algoritmos de clustering pertenecientes al Machine Learning no supervisado así como las redes de recomendación, principalmente aquellos algoritmos con un enfoques más clásico.

Capítulo 3

Clustering en Machine Learning

Definición 3.0.1. *Un algoritmo de agrupamiento (en inglés, clustering), es un procedimiento de agrupación de una serie de vectores de acuerdo con un criterio, generalmente una función de distancia o similitud.*

El clustering es uno de los campos en los que se emplean algoritmos de Machine Learning, sobre todo algoritmos de aprendizaje no supervisado. El objetivo es clasificar una serie de datos en un número de clases, buscando patrones o similitudes en sus características sin una variable objetivo como referencia. Podemos acompañar esta definición con un ejemplo muy simple: clasificación de figuras geométricas planas. Estas figuras tendrán características como el color, el número de lados o vértices, etc. Un algoritmo de clustering buscará similitudes en esas características para formar grupos. Dependiendo de cómo se defina la función que determina la similitud entre figuras es posible que el algoritmo agrupe los elementos por colores similares, o por el número de lados. Estos algoritmos se diferencian de los algoritmos de aprendizaje supervisado en que no necesitamos una base de datos que ya estén clasificados para poder determinar las clases de futuros ejemplos. Los algoritmos de clusterización determinan las clases desde cero. En el ejemplo de las figuras geométricas, un algoritmo de aprendizaje supervisado necesita de un "experto" o alguien que de antemano proporcione una base, diciéndo a qué clases pertenecen ciertas figuras.

El clustering se puede aplicar en numerosos ámbitos: biología, medicina, marketing, etc, ya que el conocimiento de estos grupos o clusters puede permitir una descripción de un conjunto de datos multidimensional complejo.

Dentro del clustering podríamos decir que hay dos técnicas para el agrupamiento de los datos [9]:

- **Clusterización jerárquica**: Consiste en hacer una agrupación en la que los clusters forman conjuntos que se contienen unos a otros. El clustering jerárquico se puede representar como un árbol, y puede ser aglomerativo (a partir de grupos con un solo individuo ir agrupando hasta hacer un único cluster), o bien divisivo (a partir de un cluster que contiene todos los datos se van haciendo particiones hasta obtener clusters con un solo individuo).

- **Clusterización no jerárquica**: Consiste en una agrupación en la que el número de grupos se determina de antemano y los datos se van organizando y asignando a los grupos en función de su cercanía. Uno de los métodos más antiguos y conocidos de clusterización no jerárquica es *K-means*, el cual comentaremos y analizaremos en el siguiente apartado.

A continuación se procederá a tratar y describir los algoritmos clásicos de clustering no supervisado elegidos para este Trabajo de Fin de Grado: sus característcas, funcionamiento, variantes, así como su base matemática, teoremas o demostraciones que sean importantes para comprender el algoritmo. Los algoritmos son los siguientes:

- **K-means**: Un algoritmo heurístico que realiza una clasificación en un número determinado de clusters ya predefinido.

- **Clustering jerárquico**: Son técnicas de clusterización que no requieren de un parámetro con el número de clusters de manera previa, sino que generan un dendograma con los clusters.

- **Análisis de Componentes Principales**: Un algoritmo de reducción de la dimensionalidad que se basa en la descomposición de una matriz de covarianza o en la descomposición en valores singulares de una matriz con los datos. Se utiliza para comprimir información o facilitar la computación a otros algoritmos.

3.1. K-means

K-means [10] [12] [11] es un algoritmo de aprendizaje automático no supervisado que realiza una clasificación y agrupamiento sobre un set de datos en un número determinado de grupos ya predefinido. El algoritmo necesita minimizar una función que representa la similitud o la distancia entre dos observaciones.

Definición 3.1.1. *Sea V un espacio vectorial sobre un cuerpo K, y sea $\vec{x}$ un vector del espacio. Se dice que $\|\cdot\| : V \to \mathbb{R}$ es un operador que define la norma de $\vec{x}$, y la escribimos $\|\vec{x}\|$. La norma debe cumplir:*

- $\|\vec{x}\| = 0 \iff \vec{x} = \vec{0}$.
- $\forall \vec{x} \in V \ y \ \forall k \in K, \ \|k\vec{x}\| = |k| * \|\vec{x}\|$.
- $\forall \vec{x}, \vec{y} \in V$, *se cumple* $\|\vec{x} + \vec{y}\| \leq \|\vec{x}\| + \|\vec{y}\|$. *(desigualdad triangular)*

El algoritmo K-means es el siguiente:

Algoritmo 3.1.2. *Dado un conjunto de observaciones $(p_1, p_2, \ldots, p_n)$, donde cada observación es un vector real de dimensión d, K-means construye una partición de las observaciones en $k \leq n$ conjuntos $S = \{S_1, S_2, \ldots, S_k\}$ con el fin de minimizar la suma de los cuadrados dentro de cada grupo y minimizar una función de coste.*

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{p \in S_i} \|x - \mu_i\|^2. \tag{3.1}$$

El resultado obtenido es una partición del espacio en celdas de Voronoi.

Definimos las celdas de Voronoi del siguiente modo [7]:

Definición 3.1.3. *Sea $P = p_1, p_2, \ldots, p_n$ un conjunto de puntos en un espacio de dimensiónn d por coherencia con el algoritmo 3.1.2. Se define el diagrama de Voronoi de P, $Vor(P)$, como la subdivisión del plano en n regiones, $V(p_i)$, una para cada $p_i \in P$ cumpliendo la propiedad de proximidad, un punto q sólo pertenece a $V(p_i)$ si y sólo si $\|q - p_i\| < \|q - p_j\|, \ \forall p_j \in P, i \neq j$*

Existen diferentes implementaciones del algoritmo K-means, cada una con ciertos pasos diferentes. La versión estándar del algoritmo, también conocida como "*Algoritmo de Lloyd*" [12] es la siguiente:

Algoritmo 3.1.4. *Dado un conjunto inicial de k centroides*

$$(m_1^{(t)}, m_2^{(t)}, \ldots, m_k^{(t)})$$

el algoritmo continúa un número determinado de iteraciones repitiendo estos dos pasos:

- *Paso 1: asignación*
  A cada observación del set de datos se le asigna el centroide más cercano o más similar a él. Este paso es la construcción del diagrama de Voronoi del conjunto de puntos de los centroides, y por tanto, cada observación irá exactamente dentro de un cluster.

$$S_i^{(t)} = \{x_p : \|x_p - m_i^{(t)}\| \leq \|x_p - m_j^{(t)}\| \ \forall \ 1 \leq j \leq k\}. \tag{3.2}$$

- *Paso 2: actualización*
  Recalcular cada uno de los centroides como la media de todas las observaciones que fueron asignadas a cada uno de ellos. Lo que hacemos es desplazar cada uno de los centroides al " centro" de cada una de sus celdas de Voronoi.

$$m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j. \tag{3.3}$$

El algoritmo se considera que ha convergido cuando las asignaciones ya no cambian, es decir, cuando el diagrama de Voronoi ya es estable.

Este algoritmo es un algoritmo heurístico. Esto quiere decir que con los dos pasos que se ejecutan en cada iteración nos asegura que el error no va a aumentar en cada una de ellas, y por tanto la convergencia de su solución. Sin embargo, no hay ninguna garantía de que se converja al óptimo global. Incluso el resultado puede depender de los grupos iniciales. Para intentar resolver este problema se ejecuta el algoritmo varias veces con diferentes condiciones iniciales, seleccionando como final aquella en la que se obtenga un mejor resultado.

Existen variaciones del algoritmo dependiendo de la distancia que se define entre los puntos o bien dependiendo de cómo se inicializan los clusters en la primera iteración. Concretamente, existen dos variantes de K-means dependiendo de cómo se eligen los centroides iniciales:

- *K-means con método de inicialización de Forgy* [12]: Consiste en elegir aleatoriamente k observaciones del conjunto de datos y utilizarlas como centroides iniciales. Este método tiende a dispersar los centroides iniciales de los clusters.

- *K-means con inicialización con partición aleatoria* [14]: Consiste en asignar al inicio y de forma aleatoria un cluster para cada observación, para después proceder al paso 2 del *algoritmo de Lloyd* (algoritmo 3.1.4). Es decir, la situación inicial del algoritmo no tiene por qué representar un diagrama de Voronoi, ya que cuando se calculen los centroides de cada grupo es bastante probable que estén desperdigados por el espacio (debido a la asignación aleatoria inicial). Terminado el paso dos del algoritmo para generar estos centroides, ya en la siguiente iteración de K-means será cuando se construirá el diagram de Voronoi calculando para cada centroide cuáles son los datos más similares a sí mismo y se definirán propiamente los clusters.

Hay dos elementos clave para conseguir una buena implementación de K-means además de los ya mencionados. El número de clusters en los que clasificar los datos y la métrica para las distancias. Se suele seleccionar como métrica la distancia euclídea (3.7), aunque para comprobar la similitud entre dos vectores también se suelen utilizar la definición del coseno (3.5) o el coeficiente de correlación de Pearson (3.6).

- Distancia Euclídea

$$d(u,v) = \sqrt{\sum_{i=1}^{d}(u_i - v_i)^2}, \tag{3.4}$$

- Distancia del coseno

$$\cos(u,v) = \frac{\langle u,v\rangle}{|u||w|}, \tag{3.5}$$

- Distancia del coeficiente de correlación de Pearson

$$\rho_{u,v} = \frac{\sigma_{u,v}}{\sigma_u \sigma_u}, \tag{3.6}$$

  donde $\sigma_{u,v}$ es la covarianza de (u,v), σ_u es la desviación típica de la variable u, y σ_v es la desviación típica de la variable v.

Para la elección del número de clusters k, no existe ninguna manera para determinar el número perfecto. Salvo que tengamos algún tipo de certeza de que el número elegido es el adecuado antes de realizar la clusterización, no hay ningún método que garantice su optimalidad. Sin embargo, existen diferentes métodos para calcular un valor de k que tiene posibilidades de ser un valor adecuado.

En lo relativo a la complejidad, la complejidad de K-means (fijados el número de grupos k y la dimensión de los datos d) pertenece a $O(n^{dk+1}\log(n))$.

K-means es uno de los algoritmos de Machine Learning sobre clustering que está implementado y disponible para ser utilizado en la aplicación diseñada en mi Trabajo de Fin de Grado de informática.

3.2. Clustering jerárquico

Los algoritmos como K-means realizan una clasificación sobre un conjunto en un determinado número de clases ya predefinido por alguien. Estos algoritmos con completamente correctos y pueden dar lugar a una clasificación razonable, sin embargo, también es posible pensar que predeterminar el número de clases no es correcto, y que debería ser el propio algoritmo el que determine el número de clases o clusters. Para estos casos, una solución es el clustering jerárquico.

La clusterización jerárquica (HCA, *Hierarchical cluster analysis* en inglés), es un método de clusterización que busca realizar una jerarquía de grupos [12] [11].

Algoritmo 3.2.1. *Existen principalmente dos estrategias de clustering jerárquico:*

- **Aglomerativo***: Es un acercamiento ascendente. Cada observación comienza en su propio cluster y dichos clusters se van juntando según su similitud mientras se sube en la jerarquía.*
- **Divisivo***: Es un acercamiento descendente. Se comienza con un único grupo que contiene todos los datos y se realizan divisiones en clusters separando los datos según su similitud, mientras se baja en la jerarquía.*

Los algoritmos de clustering jerárquico no son recomendados para grandes conjuntos de datos, ya que el aglomerativo posee una complejidad $O(n^2 \log(n))$ y el divisivo de $O(2^n)$, que es incluso peor. Sin embargo, poseen la ventaja de que no hay que determinar de antemano el número de clusters en los que clasificar los datos, sino que una vez definido el dendograma que representa los clusters, podemos decidir dónde cortar y a partir de qué punto formar los clusters.

Para tomar la decisión de qué clusters deben ser combinados o dónde se debe dividir un cluster, se debe definir un concepto de similitud entre conjuntos de observaciones. Esto se consigue definiendo una métrica (es decir, una distancia entre observaciones), y un criterio de enlace (*linkage* en inglés) para especificar la similitud o diferencias entre conjuntos en función de las distancias entre los elementos de los mismos.

La principal ventaja de los métodos de clusterización jerárquica, es que cualquier definición de estas métricas es válida para ejecutar el algoritmo. Ni

siquiera las observaciones en sí son necesarias, ya que los algoritmos trabajan con matrices, concretamente con una matriz *nxn* de distancias para n puntos. Sin embargo, esta es también su desventaja, ya que además de la complejidad que comentamos antes, también se tiene que tener en cuenta el espacio en memoria disponible para almacenar dichas estructuras de datos.

3.2.1. Métricas

La elección de la métrica infuirá en la forma de los clusters, ya que ciertas observaciones pueden estar más cerca o lejos unas de otras dependiendo de la medida elegida. Las más comunes son:

- La distancia euclídea:

$$d(u, v) = \sqrt{\sum_{i=1}^{d}(u_i - v_i)^2}, \tag{3.7}$$

- La distancia Manhattan:

$$d(u, v) = \sum_{i} |u_i| - |v_i| \tag{3.8}$$

- La distancia Mahalanobis:

$$d(u, v) = \sqrt{(u - v)^T S^{-1}(u - v)}, \tag{3.9}$$

  donde S es la matriz de covarianza entre los elementos de los vectores.

Para observaciones cuyas características no son numéricas o son cadenas de texto, se pueden utilizar otras distancias, como la distancia *de Hamming* o la distancia *de Levenshtein*.

- La distancia *de Hamming* es una distancia definida en la Teoría de la Información [25] que mide la diferencia entre dos palabras de código válidas. Cuanto mayor sea dicha distancia, menor es la posibilidad de que una palabra de código válido se transforme en otra palabra por una serie de errores. Fue definida por el matemático Richard Hamming en 1950. El único requisito para poder utilizar la distancia *de Hamming* es que ambos elementos tengan la misma longitud. Podemos definirla como:

$$d(u, v) = \sum_{i=1}^{d} \left|\delta_{u_i}^{v_i}(u_i, v_i) - 1\right| \tag{3.10}$$

  donde $|\cdot|$ representa la función valor absoluto y $\delta_{u_i}^{v_i}$ la *Delta de Kronecker* [6].

- La distancia *de Levenshtein*, también conocida como "distancia de edición", mide la diferencia entre dos palabras formadas con un determinado alfabeto. Concretamente, mide el número de operaciones requeridas para transformar una palabra en la otra. Se entiende por operación la inserción, eliminación o sustitución de un carácter del alfabeto. Fue definida en 1965 por el científico ruso Vladimir Levenshtein, y es una de las distancias más utilizadas la Teoría de la Información [25] y se la considera una generalización de la distancia *de Hamming*, ya que la distancia *de Levenshtein* acepta palabras de distinta longitud. Sean $u = u_1, \ldots, u_n$ y $v = v_1, \ldots, v_m$ las palabras a comparar, podemos definir esta distancia empleando el algoritmo de *Wagner-Fischer* [27]:

$$d_{i0} = \sum_{k=1}^{i} w_{del}(v_k) \quad \text{para } 1 \leq i \leq m \tag{3.11}$$

$$d_{0j} = \sum_{k=1}^{j} w_{ins}(u_k) \quad \text{para } 1 \leq j \leq n$$

$$d_{ij} = \begin{cases} d_{i-1,j-1} & \text{para } u_j = v_i \\ \text{mín} \begin{cases} d_{i-1,j} + w_{del}(v_i) \\ d_{i,j-1} + w_{ins}(u_j) \\ d_{i-1,j-1} + w_{sub}(u_j, v_i) \end{cases} & \text{para } u_j \neq v_i \end{cases}$$

$$\text{para } 1 \leq i \leq m \text{ y } 1 \leq j \leq n$$

  Donde w_{del} representa el número de eliminaciones a realizar, w_{ins} el número de inserciones y w_{sub} el número de sustituciones.

La distancia más utilizada en los estudios de Machine Learning y Data Mining es la distancia euclídea.

3.2.2. Criterios de enlace

Como se especificó antes, los criterios de enlace se utilizan para determinar la similitud o distancia entre dos clusters, empleando como referencias las distancias entre ciertos puntos de los mismos.

Sean U y V dos clusters cualesquiera, y sea $U[i]$ el i-ésimo punto del cluster U, podemos enumerar algunos de los criterios más comunes utilizando esta notación:

- *Single-Linkage clustering*: Se eligen aquellos puntos de los clusters que más cerca están entre sí.

$$d(U, V) = \text{mín}(d(U[i], V[j])), \tag{3.12}$$

$\forall\, U[i] \in U$ y $\forall\, V[j] \in V$.

- *Complete-Linkage clustering*: Se eligen aquellos puntos de los clusters que más lejos están entre sí.

$$d(U, V) = \text{máx}(d(U[i], V[j])), \tag{3.13}$$

$\forall\, U[i] \in U$ y $\forall\, V[j] \in V$.

- *Average linkage clustering*: Cuando se emplea, también se denomina al algoritmo de clustering jerárquico *UPGMA algorithm.*

$$d(U, V) = \sum_{ij} \frac{d(U[i], V[j])}{|U||V|}, \tag{3.14}$$

$\forall\, U[i] \in U$ y $\forall\, V[j] \in V$, donde $|U|$ y $|V|$ son las cardinalidades de los clusters.

- *Centroid linkage clustering*: Cuando se emplea, también se denomina al algoritmo de clustering jerárquico *UPGMC algorithm.*

$$d(U, V) = \|U_c - V_c\|, \tag{3.15}$$

donde U_c y V_c son los centroides de los clusters U y V respectivamente, y $\|.\|$ representa la norma tal como se definió en el apartado 3.1.

- *Ward's criterion*:

$$d(U, V) = \sqrt{\frac{|V| + |S|}{T} d(V, S)^2 + \frac{|V| + |T|}{T} d(V, T)^2 - \frac{|V|}{T} d(S, T)^2} \tag{3.16}$$

donde:

  - S y T son los clusters que al unirse formaron el cluster U
  - $T = |V| + |S| + |T|$, donde $|\cdot|$ es la cardinalidad del conjunto.

A excepción de *Centroid linkage clustering*, todos los criterios aquí listados están disponibles para ser utilizados en la aplicación diseñada en mi Trabajo de Fin de Grado de informática.

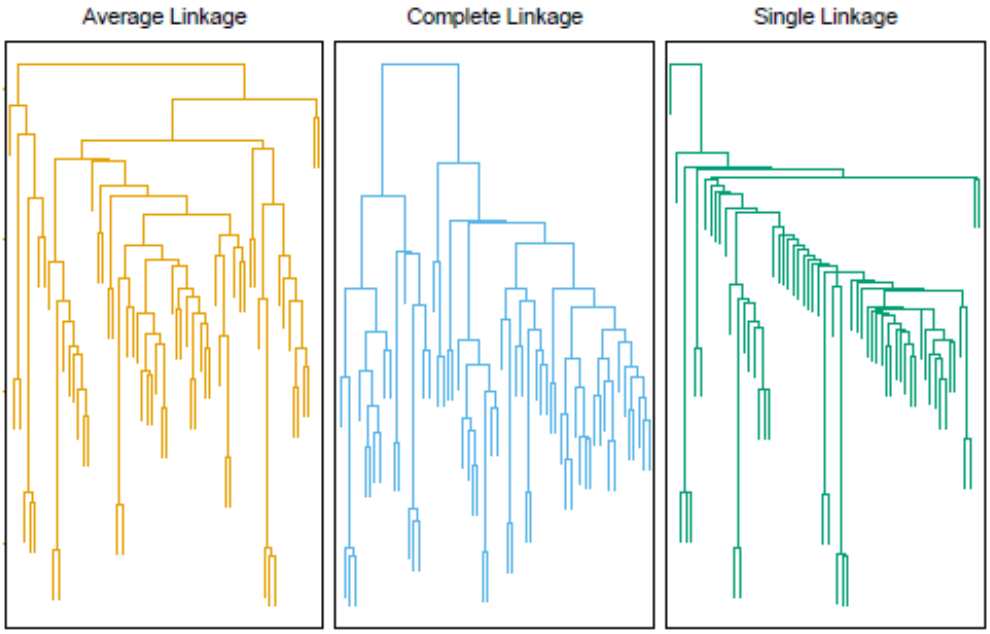

Figura 3.1: Comparación de dendogramas con diferente criterio de enlace (Fuente: [12])

3.3. Análisis de componentes principales

El Análisis de componentes principales (PCA, *Principal Component Analysis* en inglés), es un algoritmo que utiliza transformaciones ortogonales para convertir un conjunto de observaciones de variables posiblemente correlacionadas a un conjunto de valores con componentes sin correlación lineal, llamadas componentes principales. La transformación se realiza de tal manera que la primera componente tenga la mayor varianza, después la segunda componente con mayor varianza, y así sucesivamente. Al final se obtienen una serie de vectores que forman una base ortogonal [20].

PCA se puede realizar mediante una descomposición en autovalores de una matriz de covarianzas [10] o mediante una descomposición en valores singulares de una matriz de datos [12], tras haber realizado una normalización de los mismos en la matriz de datos original.

Intuitivamente, PCA se puede ver como intentar ajustar un elipsoide de dimensión n al conjunto de los datos, donde cada uno de los ejes representa una de las componentes principales [8]. Si uno de los ejes es pequeño, entonces la varianza de esa componente también lo es, y por tanto al omitir ese eje y su correspondiente componente de nuestra representación de los datos, sólo perdemos una pequeña parte de la información. Para encontrar estos ejes lo primero que hay que hacer es normalizar los datos alrededor del origen. Después calculamos la matriz de covarianza y sus autovalores y autovectores, para después ortogonalizarlos y hacerlos unitarios. Una vez hecho, cada uno de estos vectores se puede interpretar como uno de los ejes del elipsoide que se ajusta a los datos. Para calcular el porcentaje de varianza que retiene una componente, se divide su autovalor entre la suma del conjunto de todos los autovalores.

En la siguiente sección vamos a comentar el funcionamiento de PCA utilizando descomposición en valores singulares, ya que es el método que más suele usarse y que suele estar incluido en numerosas librerías.

3.3.1. Valores singulares

Antes de entrar en la enunciación y demostración del teorema de la descomposición en valores singulares, se va a definir brevemente el concepto de valor singular [8].

Para cualquier matriz M de $m \times n$, la matriz $M^T M$ $n \times n$ es simétrica y por tanto se puede diagonalizar ortogonalmente. Los autovalores de $M^T M$ son reales y no negativos, ya que para cualquier autovector u, se tiene:

$$M^T M u = \lambda u \tag{3.17}$$

si se multiplica a ambos lados por u, se tiene la igualdad

$$u^T M^T M u = \lambda u^T u \tag{3.18}$$

que indica que $\|Mu\|^2 = \lambda \|u\|^2$, por tanto $\lambda \geq 0$.

Por tanto tiene sentido tomar las $\sqrt{\lambda_i}$ si $\lambda_i, i = 1, \ldots, n$, son los autovalores de $M^T M$.

Definición 3.3.1. *Si M es una matriz $m \times n$, los valores singulares de M son las raíces cuadradas de los autovalores de $M^T M$, y se denotan mediante $\sigma_1, \ldots, \sigma_n$. Es una convención acomodar los valores singulares tal que $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n$.*

3.3.2. PCA con descomposición de valores singulares

Teorema 3.3.2. *Sea M una matriz $m \times m$ cuyas entradas pertenecen a $\mathbb{R}$ ó $\mathbb{C}$, entonces existe una factorización, llamada descomposición en valores singulares de M, de la forma*

$$M = U \Sigma V^T \tag{3.19}$$

donde:

- *U es una matriz con columnas ortogonales $m \times n$*
- *Σ es una matriz diagonal $n \times n$ con números reales no negativos en su diagonal*
- *V es una matriz ortogonal $n \times n$*

Los elementos σ_i de la diagonal de Σ son los denominados valores singulares de M. Dichos valores se suelen listar en orden descendiente [8].

Demostración. Sea $r < n$ el rango de M, entonces $\lambda_1 \geq \ldots \geq \lambda_r > \lambda_{r+1} = \ldots = \lambda_n = 0$ son los autovalores de $M^T M \in \mathbb{R}^{n \times n}$ ordenados de mayor a menor. Por tanto la matriz Σ debe tener valores en su diagonal siendo el resto de valores 0.

Sea $\{v_1, \ldots, v_n\}$ una base ortogonal de $\mathbb{R}^n$ formada por autovectores de $M^T M$, cada uno asociados en orden a un autovalor.

Sea el conjunto ortogonal $\{Mv_1, \ldots, Mv_n\}$. Dependiendo del valor de r, habrá más o menos ortogonales, ya que, sean v_i y v_j dos autovectores de $M^T M$, se cumple:

$$v_j^T M^T M v_i = v_j^T \lambda_i v_i = 0 \tag{3.20}$$

Por lo que Mv_i es ortogonal a Mv_j si no son nulos, puesto que v_i y v_j son ortogonales.

Si llamamos

$$u_1 = \frac{Mv_1}{\sigma_1}, \ldots, u_r = \frac{Mv_r}{\sigma_r}, u_{r+1} = \ldots = u_n = 0_{\mathbb{R}^m} \tag{3.21}$$

vemos que:

- $\{u_1, \ldots, u_r\}$ es un conjunto ortonormal. Entonces si $r < m$ podemos completar con $\{u_{r+1}, \ldots, u_m\}$ hasta formar una base ortonormal de $\mathbb{R}^m$.
- $$\begin{cases} Mv_1 & = & \sigma_1 u_1 \\ \vdots & & \\ Mv_r & = & \sigma_r u_r \\ Mv_{r+1} & = & 0_{\mathbb{R}^m} \\ \vdots & & \\ Mv_n & = & 0_{\mathbb{R}^m} \end{cases} \tag{3.22}$$
- Reescribiendo este último sistema de ecuaciones de manera matricial con las matrices $V = [v_1, \ldots, v_n] \in \mathbb{R}^{n \times n}$ ortogonal y

$$U\Sigma = \underbrace{[u_1, \ldots, u_m]}_{U \in \mathbb{R}^{m\times n} \text{ ortogonal}} \begin{bmatrix} \sigma_1 & 0 & \ldots & 0 & 0 & \ldots & 0 \\ 0 & \sigma_2 & \ldots & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \sigma_r & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 & 0 & \ldots & 0 \end{bmatrix} = \tag{3.23}$$

$$= [\sigma_1 u_1 \quad \ldots \quad \sigma_r u_r \quad 0 \quad \ldots \quad 0]$$

Luego claramente $MV = U\Sigma$ y, finalmente, como V es una matriz ortogonal, se puede decir que $M = U\Sigma V^T$, que es la ecuación de una descomposición en valores singulares.

Observación 3.3.3. *Los vectores de las columnas de U se denominan vectores singulares por la izquierda de M, mientras que los vectores de V se denominan vectores singulares por la derecha de M. Las matrices U y V no están determinadas en forma única por M, en cambio Σ sí, porque contiene los autovalores de M.*

Lema 3.3.4. *Sea M una matriz $m \times n$ con valores singulares $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r \geq \sigma_{r+1} \geq \ldots \geq \sigma_n$ si r es el rango de M. Sean $u_1, \ldots, u_r$ los vectores singulares por la izquierda de M, y sean $v_1, \ldots, v_r$ los vectores singulares por la derecha de M correspondientes a los valores singulares mencionados antes. Entonces:*

$$M = \sum_{i=1}^{r} \sigma_i u_i v_i^t \tag{3.24}$$

Demostración. La demostración del lema es trivial si observamos la descomposición de la ecuación (3.23). Si añadimos a esa descomposición los vectores de la matriz V, es claro que la matriz M puede escribirse como combinación lineal de matrices de rango 1 como se indica.

Una vez enunciado y demostrado el teorema de la descomposición en valores singulares de una matriz, podemos volver al tema original por el cual se incluye este algoritmo en este capítulo: el análisis de componentes principales en el Machine Learning.

PCA sirve para muchas cosas en el ámbito del Machine Learning y la minería de datos. Con PCA se puede hacer un análisis más profundo de los datos, analizar la importancia de sus variables, intentar realizar una representación gráfica de datos complejos... También se puede realizar una reducción de la dimensionalidad de los datos. En muchas ocasiones los datos poseen demasiadas variables para construir el mejor modelo posible. Con herramientas como PCA podemos realizar una selección de las mejores variables minimizando el error. PCA puede incluso utilizarse para comprimir (minimizando la pérdida de información) y descomprimir datos [15].

En este trabajo vamos a centrarnos en la reducción de la dimensionalidad de nuestros datos. El objetivo de esta reducción es transformar los datos de tal manera que "se baja" a un espacio en el que es más fácil trabajar y conservando la mayor parte de la información (retención de la variabilidad). No se puede aplicar PCA a cualquier set de datos ni de cualquier manera, es necesario realizar algunos pasos previos antes de aplicarlo, como la normalización de los datos ya mencionada al principio de este apartado.

En ciertos casos es probable que PCA no aporte un beneficio significativo a los resultados de nuestro algoritmo, o incluso que los empeore. En estas situaciones a veces puede funcionar probar diferentes normalizaciones o combinar PCA con algún otro algoritmo de reducción de la dimensionalidad o *whitening* [19]. Una transformación de *whitening* es una transformación lineal que convierte un vector de variables aleatorias dada una matriz de covarianzas en un nuevo set de variables cuya covarianza es la matriz de identidad.

Para reducir la dimensionalidad de un conjunto de vectores de dimensión n a dimensión k, podemos enunciar el siguiente algoritmo:

Algoritmo 3.3.5. *Nos quedamos con los k primeros autovectores unitarios de la matriz U, obteniendo así una matriz new_U* $\in \mathbb{R}^{n\times k}$*. Debe tenerse en cuenta que una reducción considerable en la dimensión de los datos puede conllevar una menor retención de la variabilidad. Si ahora multiplicamos los vectores de nuestros datos por dicha matriz transpuesta, obtenemos una nueva representación de los datos en dimensión k.*

$$new\_U^T x^{(i)} = \begin{bmatrix} \dots & u_1^T & \dots \\ \dots & u_2^T & \dots \\ & \vdots & \\ \dots & u_k^T & \dots \end{bmatrix} \begin{bmatrix} x_1^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} = \begin{bmatrix} z_1^{(i)} \\ \vdots \\ z_k^{(i)} \end{bmatrix} = z^{(i)} \tag{3.25}$$

Capítulo 4

Introducción a los sistemas de recomendación clásicos

Los sistemas de recomendación (Recommender Systems, RS, en inglés) son herramientas software y técnicas que proveen a un usuario sugerencias sobre items que pueden ser de su interés: como pueden ser noticias que leer, música que escuchar, productos a comprar o películas para ver [22].

Generalmente se suele denominar "item" al objeto que se recomienda al usuario, y cada sistema de recomendación suele ser específico para un tipo de item, para tener un diseño acorde con él, una interfaz gráfica o similar que sea adecuada, y para enfocarse en las técnicas concretas de recomendación para conseguir el objetivo propuesto. Los RS son principalmente personalizados, es decir, realizan sugerencias diferentes para cada usuario. Existen también sistemas de recomendación que realizan recomendaciones genéricas, se les suele ver en páginas de noticias, revistas o tiendas, mostrando los productos más de moda o los que la gente en general más consume. Este tipo de sistemas de recomendación son mucho más simples que los personalizados [1].

Los sistemas de recomendación han ayudado en estos últimos años con el problema de la gran cantidad de información que tienen disponible los usuarios. Cuando alguien solicita un producto, dependiendo del contexto y la necesidad, el sistema genera una serie de recomendaciones utilizando conocimientos acerca de otros usuarios, los items disponibles y previas compras/transacciones almacenadas en una base de datos o similar por parte de otros usuarios. Entonces el usuario responderá frente a la recomendación, bien aceptándola o no, proporcionándole al sistema un feedback que puede guardar y utilizar para futuras recomendaciones. [22]

Además, en estos últimos años se ha producido un aumento en el uso de sistemas de recomendación por parte de muchas empresas de diferentes campos [22]:

- Amazon, Youtube, Netflix, Yahoo, etc. Muchas empresas del sector multimedia han incorporado a sus sistemas un sistema de recomendación para proveer contenido a sus suscriptores. Incluso algunas como Netflix organizaron concursos con premios en metálico para que la gente aportara sus ideas para mejorar el sistema.

- Existen conferencias y eventos dedicadas a los sistemas de recomendación, como la *ACM Recommender Systems* (RecSys), que se creó en 2007.

- Existen cursos impartidos por instituciones de educación superior que se dedican en su totalidad a los sistemas de recomendación, además de la gran cantidad de artículos científicos que se publican en sitios como la *IEEE Intelligent Systems* o el *International Journal of Computer Science and Applications*.

Todo sistema de recomendación tiene una función "clave" que identifica qué items pueden serle útiles al usuario. Para implementarla, el sistema debe predecir si merece la pena recomendar cierto item u otro, calculando scores o realizando comparaciones, ya que la utilidad de un item puede depender de otras variables, lo que se conoce como contextual. Por ejemplo, el conocimiento del usuario, el momento en el que el usuario hace la petición, o la localización y disponibilidad de los items.

A continuación vamos a presentar una pequeña taxonomía de los sistemas de recomendación para a continuación detallar alguno de los más clásicos [22]:

- ***Content-based***: El sistema aprende a recomendar items que se parezcan a aquellos que le gustaron o compró en el pasado. Esta similitud se calcula a partir de las características que posean los items y una función.

- ***Collaborative filtering***: Este tipo de sistemas recomiendan al usuario los items que otros usuarios con gustos similares compraron en el pasado. La similitud entre gustos de usuarios se calcula en función de la similitud entre las valoraciones de items que han hecho cada uno de los usuarios. Estos sistemas tienen inconvenientes como falta de datos o datos dispersos (*sparse*).

- **Demográficos**: Este tipo de sistemas de recomendación se basan en el perfil demográfico de los usuarios.
- ***Knowledge-based***: Estos sistemas se basan en el conocimiento que tienen los usuarios acerca del dominio de los items en cuestión. Si los items cumplen o tienen ciertas características que el usuario necesita, sus preferencias, etc. Se suelen usar funciones para calcular la similitud entre lo que el usuario necesita y lo que ofrece cierto item.
- ***Community-based***: Este tipo de sistemas recomiendan items a un usuario basándose en las preferencias de sus amigos o conocidos, como la conocida frase: "Dime con quién andas y te diré quién eres". Se ha comprobado que la gente suele tener más en cuenta las recomendaciones de sus amigos o familiares que en personas desconocidas, por tanto, las recomendaciones se basan en valoraciones que han hecho los amigos del usuario. Estos sistemas han tenido su auge después del estallido de las redes sociales.

A continuación se va a comentar más en detalle uno de los sistemas más comunes y utilizados en los sistemas de recomendación, el *Collaborative filtering*.

Capítulo 5

Collaborative filtering

Los métodos colaborativos se basan en la asunción de que usuarios similares preferirán items similares, dicho de otra manera, que un usuario expresa preferencias similares para items similares. En vez de realizar análisis de contenidos o indexarlos, los métodos colaborativos dependen en su totalidad de que exista una comunidad de base que participe y valore los items.

Generalmente [1], los métodos de *Collaborative filtering* se clasifican en dos grupos, aunque también existen métodos híbridos, que combinan técnicas de ambos:

- Métodos basados en modelos: Utilizan los datos del sistema para construir un modelo probabilístico, como un modelo de clusters (agrupando usuarios similares en clases y estimando la probabilidad de que un usuario pertenezca a una clase) o una red Bayesiana, utilizando técnicas de Machine Learning. A partir de entonces se utiliza dicho modelo para realizar predicciones.

- Métodos basados en memoria: Almacenan las preferencias o valoraciones de los usuarios en memoria y acceden a la información necesaria en el momento para encontrar usuarios o items similares y hacer predicciones. Dentro de los algoritmos basados en memoria [23] se pueden encontrar dos grupos más, los métodos basados en usuarios (*user-based filtering*) y los basados en items (*item-based filtering*).

En los siguientes apartados se van a comentar las principales características de dos métodos basados en memoria así como de un método híbrido. Concretamente se detallará el funcionamiento del *user-based filtering*, del *item-based filtering* y del método híbrido *personality diagnosis*.

5.1. User-based Collaborative Filtering

Los sistemas de colaboración basados en usuarios pertenecen a los métodos basados en memoria, lo que quiere decir que operan sobre toda la matriz que relaciona usuarios e items, R, para hacer predicciones. La mayoría de estos sistemas se enfrentan principalmente a cálculos para determinar la similitud entre dos usuarios, utilizando grupos de vecindades construidos a partir de usuarios similares. En otras palabras, realizan operaciones con las filas de la matriz R.

Para detallar el funcionamiento de esos sistemas vamos a emplear un ejemplo como hilo de conducción. Existe un sistema de recomendación personalizado de música llamado RINGO [1] [24], que trabaja de la siguiente manera: Un nuevo usuario se "enfrenta" a la base de datos para determinar sus vecinos, que serán otros usuarios que en el pasado hayan tenido un gusto similar al usuario en cuestión (que hayan comprado la misma música, por ejemplo). Varios de los items que hayan valorado estos vecinos probablemente serán desconocidos para el usuario, por lo que se recomendarán de la siguiente forma:

- Se analizan y representan los datos de entrada.
- Se forma la vecindad del usuario.
- Se generan las recomendaciones.

A continuación se van a detallar en profundidad cada una de dichas fases.

5.1.1. Análisis y representación de los datos de entrada

Para representar la información de entrada, se necesita definir alguna estructura para guardar un set de valoraciones de los usuarios en algo que las relacione con los items en cuestión. Se suele definir una matriz R, donde cada par $R(u, i)$ representa la valoración del usuario u sobre el item i. Como los usuarios no tienen por qué dar valoraciones a todos los items que se encuentren en nuestra base de datos o sistema, es bastante probable que la matriz R sea lo que se denomina como "*matriz sparse*". Una *matriz sparse*, o matriz dispersa, es una matriz de gran tamaño en la que la mayor parte de sus elementos son cero. Esta dispersión de los valores es lo que hace que los algoritmos de recomendación a veces produzcan resultados incorrectos, por ello existen numerosas técnicas para intentar resolver esta situación [1] [22].

Por ejemplo [1]:

- Tomar el "*User Average scheme*": Consiste en calcular la valoración media que un usuario ha dado a todos los items que ha valorado, y sustituir los valores faltantes en la fila de la matriz por dicho promedio.
- Tomar el "*Item Average scheme*": Consiste en calcular la valoración media que un item ha recibido por parte de todos aquellos usuarios que lo hayan valorado, y sustituir los valores faltantes en la columna de la matriz por dicho promedio.
- El "*Composite scheme*": Es un método que intenta combinar la información de usuarios e items. La idea es utilizar la valoración del usuario u sobre el item i como base y añadir un término correctivo que se base en cómo ese item fue valorado por el resto de usuarios. Funciona de la siguiente manera [1]:
  - Una vez se localiza una entrada faltante en la matriz, se computa la media de las valoraciones del usuario en cuestión. La denominaremos $\bar{R}(u)$.
  - Se buscan las valoraciones que otros usuarios hayan hecho del item sin valorar. Sea L un set de l usuarios, $L = \{u_1, u_2, \ldots, u_l\}$, que han aportado una valoración para el item i, se puede calcular un término correctivo δ para cada usuario u de L, tal que $\delta_k = R(u_k, i) - \bar{R}(u_k)$.
  - Una vez computadas todas los términos correctivos de todos los usuarios de L, el *composite scheme* se calcula de la siguiente forma:

$$R(u,i) = \begin{cases} \bar{R}(u) + \frac{1}{l}\sum_{k=1}^{l} \delta_k & \text{si el usuario } u \text{ no ha valorado el item } i \\ R & \text{si el usuario } u \text{ ha valorado el item } i \text{ con } R \end{cases} \tag{5.1}$$

Existe una manera alternativa de aplicar el *Composite scheme*, y es realizando una transposición. Primero se calcula la media de las valoraciones que recibió el item i, $\bar{R}(i_k)$, y después se computan los términos correctivos, δ_k, de la misma manera que antes, pero ahora teniendo en cuenta todos los items $I = \{i_1, i_2, \ldots, i_l\}$ valorados por el usuario k. Una vez calculados todos los valores, se rellenan los huecos vacíos de la matriz utilizando el siguiente valor de $R(u, i)$:

$$R(u,i) = \bar{R}(i) + \frac{1}{l}\sum_{k=1}^{l} \delta_k \tag{5.2}$$

Una vez generada la matriz, se puede utilizar algún tipo de métrica para medir la similitud entre vectores de usuarios y a partir de ahí formar las vecindades de los usuarios, tema que se tratará en el siguiente punto.

5.1.2. Formación de las vecindades de los usuarios

En el siguiente paso del sistema de recomendación se debe calcular la similitud entre usuarios empleando la matriz R. De ahí se formarán grupos de vecinos, vecindades. Por ejemplo, usuarios similares al usuario u_a formarán un grupo basándose en esa similitud. Para formar estos grupos de usuarios hay que seguir dos pasos [1]:

- Primero, se calcula la similitud entre todos los usuarios utilizando una métrica para vectores.
- Segundo, se genera la vecindad en sí, donde las similitudes entre usuarios se procesan en orden para seleccionar aquellos usuarios que constituyen la vecindad del usuario en cuestión.

Para calcular la similitud entre dos usuarios u_a y u_b, podemos emplear por ejemplo la métrica del coeficiente de correlación de Pearson, que sigue de la siguiente manera:

Definición 5.1.1. *Sea un set de m usuarios, $U_m = \{u_1, u_2, \ldots, u_m\}$, que ha valorado $R(u_k, i_l)$, con $k = 1, 2, \ldots, m$ y $l = 1, 2, \ldots, n$, una serie de items $I_n = \{i_1, i_2, \ldots, i_n\}$. El coeficiente de correlación de Pearson viene dado por:*

$$sim(u_a, u_b) = \frac{\sum_{l=1}^{n}(R(u_a, i_l) - \bar{R}(u_a))(R(u_b, i_l) - \bar{R}(u_b))}{\sqrt{\sum_{l=1}^{n}\left(R(u_a, i_l) - \bar{R}(u_a)\right)^2 \sum_{l=1}^{n}\left(R(u_b, i_l) - \bar{R}(u_b)\right)^2}} \tag{5.3}$$

Otra manera de medir la similitud entre dos usuarios es con la métrica del coseno, que considera a cada usuario como vectores n dimensionales en un espacio de items, donde $n = |I_n|$. La similitud entre dos usuarios queda determinada por el coseno del ángulo entre dichos vectores:

$$sim(u_a, u_b) = \cos(\vec{u_a}, \vec{u_b}) = \frac{\sum_{l=1}^{n} R(u_a, i_l) R(u_b, i_l)}{\sqrt{\sum_{l=1}^{n} R(u_a, i_l)^2} \sqrt{\sum_{l=1}^{n} R(u_b, i_l)^2}} \tag{5.4}$$

En ciertos estudios de sistemas de recomendación [5] [1] se afirma que el coeficiente de correlación de Pearson suele dar mejores resultados que el coseno, pero ambos se pueden utilizar.

A partir de este punto en el proceso de recomendación, se selecciona a un usuario en concreto y se le denominará como *active user*. Este usuario será para el cual el sistema hará las recomendaciones y procederá a generar su vecindad. Para ello se computará una matriz de similitudes S, de tal manera que la i-ésima fila de la matriz represente la similitud del usuario u_i frente al resto de usuarios. Se pueden emplear numerosos esquemas y algoritmos de similitud con esta matriz para seleccionar aquellos usuarios más próximos al *active user*. Algunos enfoques son [1] [22]:

- *Aggregate neighborhood formation scheme*: en el que la vecindad está formada por aquellos usuarios más próximos al centroide de la vecindad a la cual pertenece el *active user*.
- *Center-based scheme*: En el que se eligen aquellos valores de la fila de la matriz que tienen un mayor valor de similitud.

5.1.3. Generación de recomendaciones

La generación de recomendaciones se basa en producir una serie de ratings o valoraciones, es decir, computar un valor numérico que represente la valoración del ususario u_a sobre el item i_l que pueda interesarle. Este valor debe estar en el rango permitido por el resto de usuarios en la matriz R, y se debe generar sólo con aquellas valoraciones de los vecinos del *active user*. En otras palabras, solo participará un subconjunto con k de los m usuarios de U_m que hayan valorado el item i_l.

Con todo esto, podemos definir una puntuación de predicción P_{u_a,i_j}:

$$P_{u_a,i_j} = \bar{R}(u_a) + \frac{\sum_{t=1}^{k}(R(u_t,i_j) - \bar{R}(u_t)) * sim(u_a,u_t)}{\sum_{t=1}^{k}|sim(u_a,u_t)|} \tag{5.5}$$

donde $U_k \subseteq U_l$.

Aquí, $\bar{R}(u_a)$ y $\bar{R}(u_t)$ son las valoraciones medias de los usuarios (u_a) y (u_t) respectivamente, mientras que $R(u_t,i_j)$ es la valoración del usuario u_t sobre el item i_j. De la misma manera, $sim(u_a,u_t)$ representa la similitud entre entre los usuarios u_a y u_t, utilizando la métrica del coeficiente de correlación de Pearson (5.3). El sistema terminará devolviendo varios items que reciban las mayores puntuaciones de recomendación.

Otra aproximación al problema de la recomendación es la denominada *top-N recommendations* [1]. En ella las recomendaciones formarán una lista de N items los cuales se espera que le gusten al *active user*. Para generar esta lista se clasifican los usuarios según la similitud con el *active user*. Se seleccinoan los k usuarios más similares, de una manera similar al algoritmo de Machine Learning supervisado *K-nn* [16]. Se crea un contador para la frecuencia de cada item y se aumenta según la valoración de dichos k usuarios. Se ordena la lista de frecuencias de los items y se ofrecen al *active user* los N primeros, los top-N.

5.2. Item-based Collaborative filtering

Item-based filtering es la otra aproximación al problema principal de la recomendación en el *Collaborative filtering*. Se basa en las relaciones entre items, y no entre las relaciones de los usuarios, como en *user-based*. Debido a que las relaciones entre usuarios son ciertamente dinámicas y cambian relativamente rápido, la generación de las matrices con los datos de los usuarios se complica, lo que conlleva a que que el *user-based Collaborative filtering* sea relativamente costoso computacionalmente hablando [1].

En el enfoque *Item-based* se trabaja con el conjunto de items I_{u_a} que el *active user* u_a haya valorado previamente, y se computarán similitudes con el item objetivo i_t. Entonces, se seleccionarán los k items más similares, $I_k = \{i_1, i_2, \ldots, i_k\}$, basándose en las similitudes que presentaron con el objetivo, $\{sim(i_t, i_1), sim(i_t, i_2), \ldots, sim(i_i, i_k)\}$. Las predicciones se pueden calcular utilizando una media ponderada sobre las valoraciones que el *active user* haya realizado sobre esos items similares al objetivo. Los pasos que se siguen para este enfoque de sistemas de recomendación son muy parecidos a los del *user-based*, en vez de calcularse similitudes entre usuarios que hayan valorado items en común, se calculan similitudes entre dos items i_t, i_j que hayan sido valorados por un usuario en común u_a.

Por tanto, las métricas del coseno y el coeficiente de correlación de Pearson quedan definidas así [1]:

$$sim(i_t, i_j) = \frac{\sum_{l=1}^{n}(R(u_l, i_t) - \bar{R}(i_t))(R(u_l, i_j) - \bar{R}(i_j))}{\sqrt{\sum_{l=1}^{n}\left(R(u_l, i_t) - \bar{R}(i_t)\right)^2 \sum_{l=1}^{n}\left(R(u_l, i_j) - \bar{R}(i_j)\right)^2}} \tag{5.6}$$

$$sim(u_a, u_b) = \cos(\vec{u_a}, \vec{u_b}) = \frac{\sum_{l=1}^{n} R(u_a, i_l) R(u_b, i_l)}{\sqrt{\sum_{l=1}^{n} R(u_a, i_l)^2} \sqrt{\sum_{l=1}^{n} R(u_b, i_l)^2}} \tag{5.7}$$

Una vez calculadas todas las similitudes entre todos los items en R, el paso final en el proceso de recomendación es calcular y aislar k de los n items, $I_k \subseteq I_n$, que más similares sean al item objetivo i_t sobre el cual se quiere hacer la predicción. Se formará un vecindad de items de la misma manera que en el *user-based filtering* y se procederá con la generación de recomendaciones. La predicción del item i_t para el *active user* u_a se calcula como la suma de las valoraciones dadas por el *active user* a los items del vecindario de i_t, I_k. Estas valoraciones se ponderarán según la similitud de cada uno de dichos items vecinos con i_t, $sim(i_t, i_j)$:

$$P_{u_a,i_j} = \frac{\sum_{j=1}^{k} R(u_a, i_j) * sim(i_t, i_j)}{\sum_{j=1}^{k} |sim(i_t, i_j)|} \tag{5.8}$$

donde $I_k \subseteq I_n$.

Existen autores [1] [13] que proponen que se puede establecer un perfil a largo plazo de los intereses de los usuarios observando su comportamiento o proporcionándole items determinados según sus tareas/intenciones. Utilizando la matriz de correlaciones entre items, y realizando un reajuste de los rankings en función de lo que busque el usuario, se pueden seleccionar items que concuerden con el perfil de un usuario de manera más sencilla. Como cumplen el perfil del usuario, serán útiles para el usuario y para que realice sus tareas.

5.3. Personality Diagnosis

Personality Diagnosis puede verse como un método de *Collaborative filtering* híbrido entre los basados en memoria y los basados en modelos. Su principal característica es que las predicciones tienen un significado semántico probabilístico, por ejemplo, podrían tenerse en consideración las preferencias de un usuario para estimar la probabilidad de que tenga la misma personalidad o gustos que otro usuario. La personalidad de un usuario se puede tomar como un vector con las valoraciones "de verdad" que ha hecho dicho usuario. Estas valoraciones "verdaderas" se diferencian del resto en una cantidad de ruido (Gaussiano). Dada la personalidad de un usuario, *personality diagnosis* estima la probabilidad de que un cierto usuario tenga la misma personalidad que otro, y consecuentemente, estima la probabilidad de que a dicho usuario le guste cierto item nuevo [1].

Como se dijo antes, la personalidad de cada usuario u_k viene dada por:

Definición 5.3.1. *Sean $k = 1, 2, \ldots, m$, $U_m = \{u_1, u_2, \ldots, u_m\}$, y un usuario u_k que tiene una serie de items $I_n = \{i_1, i_2, \ldots, i_n\}$ preferidos. Su personalidad queda determinada por el vector:*

$$\overset{true}{R}(u_k) = \{\overset{true}{R}(u_k, i_1), \overset{true}{R}(u_k, i_2), \ldots, \overset{true}{R}(u_k, i_n)\}. \tag{5.9}$$

$\overset{true}{R}(u_k, i_l)$, con $i_l \in I_n$ y $l = 1, 2, \ldots, n$ representa el verdadero rating del usuario u_k sobre el item i_l. Es importante distinguir entre la valoración verdadera y la valoración que se recibe en un principio, ya que la valoración verdadera esconde por debajo las preferencias del usuario que no son accesibles al diseñador del sistema de recomendación. Se asume también que las valoraciones recibidas (se denominará así a aquellas valoraciones que el usuario informa al sistema y por tanto no son las verdaderas) contienen ruido Gaussiano. Con esto se asume que el mismo usuario puede valorar de distinta manera items similares dependiendo del contexto. Por tanto, la valoración recibida de un usuario para el item i_l viene dada por una distribución normal independendiente, con media $\overset{true}{R}(u_k, i_l)$. En concreto:

$$Pr\big(R(u_k, i_l) = x \big| \overset{true}{R}(u_k, i_l) = y\big) \propto e^{-\frac{(x-y)^2}{2\sigma^2}}, \tag{5.10}$$

donde σ es un parámetro libre, x es la valoración recibida del usuario, y es la verdadera valoración del usuario u_k (aquella valoración que se habría informado si no hubiera ruido presente), y el símbolo $\propto$ significa proporcional.

Antes se mencionó que estas valoraciones verdaderas son representativas de la personalidad y los gustos de los usuarios. Con esto, se puede formular la probabilidad a priori $Pr\big(\overset{true}{R}(u_a) = v\big)$ de que el usuario u_a valore los items de acuerdo a un vector v dado por la frecuencia en que otros usuarios valoran de acuerdo a v. Entonces, en vez de contar las ocurrencias, se puede definir $\overset{true}{R}(u_a)$ como una variable aleatoria que puede tomar uno de m valores, $(R(u_1), R(u_2), \ldots, R(u_m))$, cada uno con probabilidad $\frac{1}{m}$:

$$Pr\big(\overset{true}{R}(u_a) = R(u_k)\big) = \frac{1}{m}. \tag{5.11}$$

Combinando las ecuaciones (5.10) y (5.11) y dadas las valoraciones del *active user*, se puede calcular la probabilidad de que el *active user* posea la misma personalidad o gustos que otro usuario cualquiera aplicando la regla de Bayes. Enunciamos entonces el Teorema de Bayes:

Teorema 5.3.2. *Sea* $\{A_1, A_2, \ldots, A_i, \ldots, A_n\}$ *un conjunto de sucesos mutuamente excluyentes y exhaustivos, y tales que la probabilidad de cada uno de ellos es distinta de cero. Sea* B *un suceso cualquiera del que se conocen las probabilidades condicionales* $P(B|A_i)$*. Entonces, la probabilidad* $P(A_i|B)$ *viene dada por la expresión:*

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{P(B)} \tag{5.12}$$

Se puede encontrar la demostración del teorema en numerosos documentos, como el siguiente de la Universidad de Pensilvania [21].

Empleando el teorema de Bayes y las ecuaciones (5.10) y (5.11) se obtiene:

$$\begin{aligned} &Pr\big(\overset{true}{R}(u_a) = R(u_k) | R(u_a, i_1) = x_1, \ldots, R(u_a, i_n) = x_n\big) \qquad (5.13)\\ &\propto \Big(\prod_{k=1}^{n} Pr\big(R(u_a, i_k) = x_k | \overset{true}{R}(u_a, i_k) = R(u_a, i_k)\big)\Big) \cdot Pr\big(\overset{true}{R}(u_a) = R(u_k)\big). \end{aligned}$$

Por lo tanto, calculando esta cantidad para cada usuario u_k se puede calcular la distribución de probabilidad para el *active user* sobre un item desconocido para él i_j. Esta distribución se corresponde con la predicción P_{u_a,i_j}, generada por el sistema de recomendación e igual a la valoración esperada del *active user* u_a para el item i_j:

$$\begin{aligned} P_{u_a,i_j} &= Pr\big(R(u_a,i_j) = x_j | R(u_a,i_1) = x_1, \ldots, R(u_a,i_n) = x_n\big) \qquad (5.14) \\ &= \sum_{k=1}^{m} Pr\big(R(u_a,i_j) = x_j | \overset{true}{R}(u_a) = R(u_k)\big) \\ &\quad \cdot Pr\big(\overset{true}{R}(u_a) = R(u_k) | R(u_a,i_1) = x_1, \ldots, R(u_a,i_n) = x_n\big). \end{aligned}$$

Este modelo se representa como un *Clasificador bayesiano ingenuo* [17] con un modelo clásico y la estructura siguiente:

- Primero se observan las valoraciones. Un enfoque sería ver las valoraciones como una serie de síntomas mientras que las personalidades son las enfermedades que llevan a esos síntomas. Empleando la ecuación (5.13) y las valoraciones, se calcula la probabilidad para cada tipo de personalidad o gustos de que sean la causa de dichos síntomas.

- Después se calculan las probabilidades de las valoraciones para items desconocidos utilizando la ecuación (5.14). La valoración más probable es devuelta como la predicción del sistema de recomendación.

Los resultados de *personality diagnosis* se pueden interpretar de varias maneras. Una de ellas es considerarlo como un método de clustering, viendo cada usuario como un cluster unitario. Ya que cada usuario se corresponde con una única personalidad, se intentará asignar al *active user* a uno de esos clusters [1]. Otra manera es imaginar que el *active user* está determinado por otro usuario cualquiera de manera uniforme tras añadirle una cantidad determinada de ruido a sus valoraciones. Conociendo las valoraciones recibidas del *active user* se puede inferir la probabilidad de que sea un usuario concreto y a partir de ahí calcular las probabilidades de las valoraciones para el resto de items.

Capítulo 6

Conclusiones y trabajo futuro

En este Trabajo de Fin de Grado se han presentado algunos de los algoritmos de clustering más conocidos en Machine Learning y redes de recomendación. Se ha analizado su funcionamiento, detallado alguna de sus variantes o enfoques e incluido la demostración de algún teorema.

El aprendizaje automático es un tema muy interesante y presente en la actualidad. No es suficiente con conocer un gran número de algoritmos para modelar o predecir ciertos datos, es necesario conocer cómo funcionan y su complejidad, para así distinguir en qué casos pueden ser unos mejores que otros, ya que así obtendremos mejores resultados. En mi Trabajo de Fin de Grado de Informática emplearé algunos de los fundamentos matemáticos de los algoritmos descritos en la primera mitad de este trabajo, porque aunque los sistemas de recomendación son otra aproximación al problema totalmente válida y podrían emplearse sin problemas, decidí centrarme en métodos de clusterización y Machine Learning clásico.

6.1. Logros conseguidos

Algunos de los logros y conocimientos adquiridos durante la realización de este trabajo son:

- Se han adquirido conocimiento sobre las diferentes taxonomías existentes de los algoritmos de Machine Learning, en función de la información que necesitan, los principios del algoritmos, etc.

- Se han visto aproximaciones matemáticas de los algoritmos, como relacionar K-means con Voronoi y sus características, o los sistemas de recomendación híbridos con la regla de Bayes y los clasificadores Bayesianos.
- Se han aprendido diferentes métricas nuevas y conceptos de similitud entre vectores de dimensión n, tengan elementos numéricos o no.
- Se han dado pequeñas pinceladas de conceptos nuevos para mí, como los Clasificadores bayesianos ingenuos, el ruido Gaussiano o el *Personality diagnosis*.

6.2. Trabajo futuro

Llegado este punto del trabajo, se pueden plantear una serie de posibles mejoras futuras que permitan extender los logros alcanzados y mejorar este trabajo:

- Realizar un análisis de algoritmos de aprendizaje automático supervisado. En este trabajo solo se estudian algoritmos de clusterización, es decir, de aprendizaje automático no supervisado. Estaría bien estudiar alguno de la otra clase, como las Máquinas de Soporte Vectorial, SVMs.
- Intentar explorar alguno de los enfoques más modernos de los sistemas de recomendación, aunque sea de manera muy breve, como los sistemas basados en grafos y redes complejas, así como la detección de comunidades.
- Estudiar algún otro sistema de recomendación clásico. *Collaborative filtering* es uno de los sistemas de recomendación más utilizados y conocidos, y aunque se han visto varios de sus enfoques en este trabajo, existen algunos otros sistemas ya enumerados en el capítulo 4, como el *content based filtering*, etc. Podría ser interesante analizar alguno otro para compararlo con el *collaborative filtering*.

Bibliografía

[1] Aristomenis S. Lampropoulos y George M. Tsihrintzis, *"Machine Learning Paradigms, Applications in Recommender Systems"*, Editorial Springer, 2015

[2] Big data en Amazon Web Services
`aws.amazon.com/es/training/course-descriptions/bigdata/`

[3] Big data: Volumen, Variabilidad y Velocidad
`news.microsoft.com/es-xl/big-data-volumen-variabilidad-y-velocidad`

[4] Big Data y la gestión del volumen de datos
`inndeavalencia.com/es/el-big-data-y-la-gestion-del-volumen-de-datos`

[5] J.S. Breese, D. Heckerman y C. Kadie, *"Empirical analysis of predictive algorithms for collaborative filtering"*, 14^{th} Conference on Uncertainty in Artificial Intelligence, 1998

[6] Delta de Kronecker
`es.wikipedia.org/wiki/Delta_de_Kronecker`

[7] Diagrama de Voronoi
`en.wikipedia.org/wiki/Voronoi_diagram`

[8] Documento de la Universidad Nacional de La Plata, *Descomposición en valores singulares (SVD) - Año 2010*
`www.mate.unlp.edu.ar/practicas/70_18_0911201012951`

[9] G. M. Downs y J. M. Barnard, Hierarchical and non-Hierarchical Clustering, 1995
`www.daylight.com/meetings/mug96/barnard/E-MUG95.html`

[10] S. Dua y X. Du, *"Data Mining and Machine Learning in Cybersecurity"*, Editorial CRC Press, 2011

[11] P. Flach, *"Machine Learning, The Art and Science of Algorithms that Make Sense of Data"*, Editorial Cambridge, 2012

[12] T. Hastie, R. Tibshirani y J. Friedman, *"The Elements of Statistical Learning"*, Editorial Springer, 2ª edición, 2009

[13] J.L. Herlocker y J.A. Konstan, *"Content-independent task-focused recommendation"*, IEEE Internet Comput., 2001

[14] Initializing the position of the clusters, K-means
`www.onmyphd.com/?p=k-means.clustering#h3_init`

[15] R. Islamaj Dogan, *Principal Component Analysis*, Documento del Departamento de Computer Science, Universidad de Maryland
`www.cs.umd.edu/~samir/498/PCA.pdf`

[16] Sergio Michiels Mangas, *"Clasificación y predicción de vulnerabilidades en ciberseguridad basada en técnicas de Machine Learning"*, Noviembre, 2016

[17] Naive Bayes classifier
`en.wikipedia.org/wiki/Naive_Bayes_classifier`

[18] Netflix Prize
`www.netflixprize.com/`

[19] PCA Whitening
`ufldl.stanford.edu/tutorial/unsupervised/PCAWhitening/`

[20] Principal Component Analysis
`en.wikipedia.org/wiki/Principal_component_analysis`

[21] Proof of Bayes Theorem
`hep.upenn.edu/~johnda/Papers/Bayes.pdf`

[22] F. Ricci, L. Rokach, B. Shapira, Paul B. Kantor, *"Recommender System Handbook"*, Editorial Springer, 2010

[23] B. Sarwar, G. Karypis, J. Konstan y J. Reidl, *"Item-based collaborative filtering recommendation algorithms"*, WWW10, 2001

[24] U. Shardanand, P. Maes, *"Social information filtering: algorithms for automating "word of mouth" "*, ACM Press/Addison-Wesley Publishing Co., New York, 1995

[25] Teoría de la información
es.wikipedia.org/wiki/Teor\%C3\%ADa_de_la_informaci\%C3\%B3n

[26] Upcoming conferences - IEEE Big Data
bigdata.ieee.org/conferences

[27] Wagner-Fischer algorithm
en.wikipedia.org/wiki/Wagner\%E2\%80\%93Fischer_algorithm