Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA AGRONÓMICA Y BIOCIENCIAS

NEKAZARITZAKO INGENIARITZAKO ETA BIOZIENTZIETAKO GOI MAILAKO ESKOLA TEKNIKOA

ESTUDIO SOBRE UN SISTEMA DE RECOMENDACIÓN AVANZADO: APREDIZAJE COLABORATIVO DE MÉTRICAS

presentado por

María Canta Belategui (k)

aurkeztua

GRADO EN CIENCIA DE DATOS
GRADUA DATUEN ZIENTZIAN

dirigido por

Mikel Sesma Sara (k)

zuzendua

Junio, 2023 / *2023, ekaina*

RESUMEN

Un Sistema de Recomendación es una herramienta tecnológica que permite a los usuarios seleccionar los elementos más adecuados entre una amplia variedad de opciones. A través del análisis de patrones y comportamientos, estos sistemas son capaces de detectar las necesidades y preferencias de cada cliente de forma individualizada y ofrecer recomendaciones que se ajusten a sus gustos y preferencias.

Inicialmente, los Sistemas de Recomendación clásicos fundamentaban sus bases en preferencias anteriores del usuario o en gustos de usuarios similares. A medida que la incipiente sobrecarga de información ha ido creciendo, se han adoptado numerosos enfoques para solucionar los problemas asociados a estas técnicas y se han desarrollado una amplia variedad de algoritmos para mejorar la precisión y diversidad de las recomendaciones. En este escrito, se realiza una revisión exhaustiva de la literatura existente, con el fin de identificar los conceptos y las ideas clave. Asumiendo las limitaciones de los diseños clásicos, se propone una métrica de recomendación avanzada, *Aprendizaje Métrico Colaborativo*, que considere no solo la precisión, sino también la diversidad y novedad de las recomendaciones. Este algoritmo de recomendación mejorado es implementado y evaluado en el conjunto de datos *MovieLens*, demostrando su eficacia en la personalización de las recomendaciones y la mejora de la satisfacción del usuario.

Palabras Clave

Sistemas de Recomendación, aprendizaje automático, métricas de similitud, sugerencia personalizada, usuario, ítem, *Aprendizaje Colaborativo de Métricas*, películas.

ABSTRACT

A Recommender System is a technological tool that allows users to select the most appropriate items from a wide variety of options. Through the analysis of patterns and behaviors, these systems are able to detect the needs and preferences of each customer individually and offer recommendations that match their tastes and preferences.

Initially, classical Recommender Systems were based on the user's previous preferences or similar users' likes. As the burgeoning information overload has grown, numerous approaches have been adopted to solve the problems associated with these techniques, and a wide variety of algorithms have been developed to improve the accuracy and diversity of recommendations. In this paper, a comprehensive review of the existing literature is conducted to identify key concepts and ideas. Assuming the limitations of classical designs, an advanced recommendation metric, Collaborative Metric Learning, is proposed to consider not only the accuracy, but also the diversity and novelty of recommendations. This improved recommendation algorithm is implemented and evaluated on MovieLens dataset, demonstrating its effectiveness in personalizing recommendations and improving user satisfaction.

Key Words

Recommender System, machine learning, metric, similarity, personalized suggestion, user, item, Collaborative Metric Learning, films.

Índice

1. Introducción y objetivos

La nueva era de la información ha supuesto una auténtica revolución digital, en la que la generación, recopilación y comunicación de datos es constante. Este tráfico de información destaca por dar origen a un depósito de datos complejo y de gran volumen, vertiendo la posibilidad de acceder rápidamente a una amplia gama de recursos. Mientras que hace varios años la principal dificultad era obtener la suficiente información para poder tomar decisiones, en la actualidad nos encontramos con el desafío contrario: un exceso de la misma, que puede resultar abrumador y enredar aún más el proceso de selección.

Este volumen de conocimiento provoca que, en gran medida, la satisfacción del usuario esté directamente ligada con su capacidad de gestionar, analizar y sintetizar la información proveniente del gran número de estímulos que le rodean. Sin embargo, en númerosas ocasiones, este cúmulo de información provoca una sobreexposición excesiva que propicia situaciones de incertidumbre en las que el usuario se ve ahogado en un sinfín de opciones. La habilidad de poder elegir entre todas ellas la que es, a priori, la mejor, se ha sustentado hasta hace poco en experiencias propias y recomendaciones de amigos, conocidos o expertos. No obstante, este contexto de saturación de la información es el que ha impulsado la aparición de herramientas de automatización que permiten aunar el fundamento principal de estas situaciones ordinarias.

Un Sistema de Recomendación es una herramienta que utiliza técnicas inteligentes con el objetivo de ofrecer sugerencias personalizadas a los usuarios sobre productos, servicios o contenidos que puedan ser de su interés [25]. Estos sistemas se basan en el análisis de patrones y preferencias del usuario, así como en la información obtenida a partir de otros usuarios con perfiles similares. El propósito principal es realizar recomendaciones que se adapten a las necesidades y gustos individuales de cada integrante, esto es, guiar al usuario de forma personalizada hacia selecciones acertadas en un amplio rango de opciones posibles. Estos "consejeros virtuales" tienen como característica fundamental que están enfocados a usuarios individuales, fusionando preferencias explícitas e implícitas para hacer frente a la sobrecarga de información. Son ampliamente utilizados en plataformas digitales como sitios web de comercio electrónico, redes sociales o plataformas de streaming de video y música, entre otros [25].

El desarrollo de estos sistemas inteligentes se remonta a finales del siglo XX [25] , cuando el auge y la popularización del uso de Internet provocó también la propagación de los Sistemas de Recomendación. Desde entonces, [27] asegura que ha sido un asunto fundamental y de gran interés en el campo de las Ciencias de la Computación, en el que se han ido desarrollando diferentes metodologías y algoritmos para hacer frente a este exceso de contenido.

En el núcleo de los Sistemas de Recomendación tradicionalmente encontramos dos enfoques de diseño: los *basados en contenido* y los de *filtrado colaborativo*.

Los sistemas de recomendación basados en contenido son aquellos que utilizan información sobre los elementos (ítems) del propio usuario para recomendar otros objetos similares. Para ello, se genera un perfil de usuario [25] a partir de las características de los elementos preferidos por éste, utilizado para recomendarle contenido parecido. En otras palabras, se basa en la idea de que si a un usuario le gusta un ítem en particular, es probable que también le gusten los ítems similares. En lo que al filtrado colaborativo refiere, [27] manifiesta que estos sistemas se basan en la idea de que si a un grupo de usuarios con intereses similares le ha gustado un elemento en particular, entonces es probable que a otro usuario con intereses parecidos también le guste ese elemento. Se centran, por tanto, en las valoraciones de otros usuarios. Por consiguiente, mientras que el foco principal de los Sistemas de Recomendación basados en contenido es aprender automáticamente las características de los elementos, el filtrado colaborativo se centra en determinar métricas de similitud que permitan calcular las distancias entre diferentes usuarios e ítems.

Convencionalmente han sido estos dos enfoques los que han fundamentado el núcleo y las bases de los Sistemas de Recomendación. Sin embargo, la incremental sobrecarga de información ha situado este ámbito en primer plano de estudio, favoreciendo el desarrollo de herramientas más avanzadas. En [27] se describe su trayectoria de desarrollo, que ha recorrido desde combinaciones de metodologías tradicionales (sistemas híbridos), hasta sistemas inteligentes basados en redes neuronales profundas o Inteligencia Artificial.

En el preámbulo de avanzar hacia la mejora continua se propone la técnica *Aprendizaje Métrico Colaborativo*, a partir de ahora CML. Esta técnica propone un enfoque de aprendizaje automático que se utiliza para aprender métricas de distancia personalizadas a partir de datos de múltiples fuentes. La principal novedad que aporta este diseño es la posibilidad de añadir características propias de los objetos a recomendar, lo que hace que se aúnen las dos vertientes tradicionales de los sistemas de recomendación: los basados en contenido y los basados en filtrado colaborativo [13]. En este diseño, de acuerdo con [11], los algoritmos de aprendizaje buscan aprender una función que pueda medir la similitud o la distancia entre pares de objetos, bien usuarios o bien ítems, de manera que objetos similares tengan una distancia menor que objetos disímiles.

Por ende, a raíz de la incrememtal importancia y popularización de los Sistemas de Recomendación, la utilización de estos "asistentes personales" ha abarcado un campo interdisciplinar. En este abanico de posibles, hay numerosos y muy diversos rasgos de estos sistemas inteligentes que pueden afrontarse con distinto nivel de complejidad. A lo largo de las líneas de este estudio se pretende realizar un profundo recorrido de las técnicas tradicionales y

proponer nuevas arquitecturas de diseño que mejoren las recomendaciones alcanzadas. En virtud de ello, el propósito principal del presente trabajo es:

- Estudiar e implementar una herramienta de recomendación avanzada, CML, con su respectiva aplicación, evaluación y comparación con el resto de técnicas desarrolladas.

Asimismo, a lo largo de las líneas de este estudio encontraremos objetivos más específicos:

- Implementar y desarrollar los sistemas clásicos de recomendación, presentando sus ventajas y limitaciones.
- Investigar y comprender los fundamentos teóricos del Aprendizaje Métrico Colaborativo y cómo se diferencia de los enfoques tradicionales de Sistemas de Recomendación.
- Desarrollar e implementar un algoritmo de Aprendizaje Métrico Colaborativo para Sistemas de Recomendación utilizando el conjunto de datos *MovieLens*.
- Evaluar el rendimiento de la técnica de Aprendizaje Métrico Colaborativo.
- Analizar los resultados mediante métricas de evaluación convenientes y comparar la precisión obtenida con los enfoques tradicionales basados en contenido y filtrado colaborativo.

2. Preliminares

2.1. Conceptos previos

La Inteligencia Artificial (IA) y el aprendizaje automático son técnicas que están transformando vertiginosamente el mundo de los Sistemas de Recomendación. La IA es un campo de las Ciencias de la Computación y la Informática que se enfoca en crear sistemas que puedan realizar tareas que normalmente requerirían la inteligencia humana, como el razonamiento, el aprendizaje y la percepción [27]. Por otro lado, el aprendizaje automático es una rama de la IA que se enfoca en enseñar a las máquinas a aprender de los datos y mejorar con la experiencia [10]. Con el objetivo de automatizar prácticas habituales de decisión, ambos conceptos han supuesto un auténtico revuelo de los sistemas inteligentes.

En este sentido, el aprendizaje automático es una técnica clave de los Sistemas de Recomendación, utilizado para crear modelos que puedan predecir las preferencias de los usuarios y proporcionar recomendaciones personalizadas en tiempo real. Dentro de este ámbito se puede hacer una importante distinción: aprendizaje supervisado y no supervisado. El aprendizaje supervisado implica entrenar un modelo de aprendizaje automático utilizando datos etiquetados, es decir, datos que ya tienen una respuesta conocida. El objetivo del modelo es aprender la relación entre las características de los datos de entrada y la salida deseada. Una vez que se entrena el modelo, se puede utilizar para predecir el resultado esperado para nuevos datos de entrada. En contraste, el aprendizaje no supervisado implica entrenar un modelo utilizando datos no etiquetados, es decir, datos que no tienen una respuesta esperada conocida. Tiene como propósito descubrir patrones y estructuras ocultas en los datos y agruparlos en categorías o clústeres [28]. Asimismo, tanto en sistemas basados en contenido, como en herramientas de filtrado colaborativo se utilizan técnicas de aprendizaje no supervisado para encontrar usuarios similares y hacer predicciones de valoración.

En lo que al aprendizaje supervisado respecta, la regresión y la clasificación son planteamientos comúnmente utilizados para generar estas recomendaciones individualizadas. Mientras que la regresión se aplica para predecir una variable continua, como por ejemplo, la valoración que un usuario le dará a una película, la clasificación se utiliza para predecir una variable discreta, tal como si a un usuario le gusta, o no, un ítem [28]. En general, la elección entre regresión o clasificación dependerá del tipo de problema de recomendación que se esté abordando y del tipo de variable objetivo que se quiera predecir. Estas herramientas más sencillas son implementadas en los Sistemas de Recomendación iniciales, de carácter más básico y, por tanto, más sencillos. Más aún, existen otras técnicas de aprendizaje automático más complejas; filtrado colaborativo, procesamiento del lenguaje natural o redes neuronales profundas [27], que se irán incorporando a lo largo del escrito.

Adicionalmente, algoritmos de aprendizaje automático se emplean también en procesos de optimización [11], tal que se pueda encontrar la combinación óptima de parámetros que maximicen la precisión y la eficiencia del Sistema de Recomendación. En [11] el algoritmo implementado es AdaGrad, que optimiza la función de pérdida que mide la discrepancia entre las observaciones reales y las predicciones del modelo. En lugar de tener una tasa de aprendizaje global para todos los parámetros, AdaGrad adapta la tasa de aprendizaje para cada parámetro de forma individual, utilizando información sobre la frecuencia y magnitud de las actualizaciones anteriores. Por tanto, a través del aprendizaje automático se construye y optimiza un modelo de recomendación con el propósito de obtener predicciones lo más precisas posible.

En suma, el aprendizaje automático y la IA son técnicas clave utilizadas en los Sistemas de Recomendación para mejorar la experiencia del usuario y proporcionar recomendaciones personalizadas con una potencial mejora de selección de contenidos y búsqueda efectiva.

2.2. Técnicas de aprendizaje

Un aspecto primordial de los Sistemas de Recomendación es la extracción de características, proceso de indentifiación y representación de las propiedades relevantes de los usuarios y los elementos [28]. En gran medida, la efectividad y precisión del proceso de recomendación residirá en la rigurosidad de este procedimiento. Los resultados obtenidos a través de esta extracción se utilizan para construir vectores de características, denominados *embeddings*, que permiten representar usuarios y elementos en un espacio vectorial común. Estos embeddings se utilizan para capturar las características relevantes de los usuarios y elementos, y para encontrar patrones de interacción entre ellos [28]. En los modelos basados en contenido, los embeddings se utilizan para representar las características de los ítems [25], mientras que en los modelos colaborativos se usan para representar las preferencias de los usuarios y para encontrar patrones en las interacciones entre los usuarios y los elementos. Para la representación de estas relaciones se utiliza una matriz de interacciones usuario-ítem definida a partir de las valoración que un usuario hace a un elemento [27]. En ambos casos, no hace falta recurrir a técnicas de procesamiento de la información para obtener los vectores de características.

En técnicas de recomendación más avanzadas, como la propuesta en este documento, se emplean herramientas de Deep Learning para aprender y extraer características complejas de los datos. El Deep Learning es una técnica de aprendizaje automático que se enfoca en el uso de algoritmos de redes neuronales artificiales para aprender a partir de datos. Una red neuronal artificial está compuesta por una red de nodos interconectados que procesan

información de entrada y producen una salida. Cada nodo está asociado con una función matemática y se llama neurona artificial [28]. En el contexto de los Sistemas de Recomendación, especialmente en la metodología CML [11], el Deep Learning ha demostrado ser muy efectivo para modelar patrones de comportamiento de los usuarios y mejorar la precisión de las recomendaciones. Aplicado fundamentalmente al modelado de funciones de similitud, se pueden aprender automáticamente características complejas de los datos que pueden ser difíciles de capturar con técnicas más simples. La red neuronal implementada es el perceptrón multicapa (MLP, por sus siglas en inglés) [2].

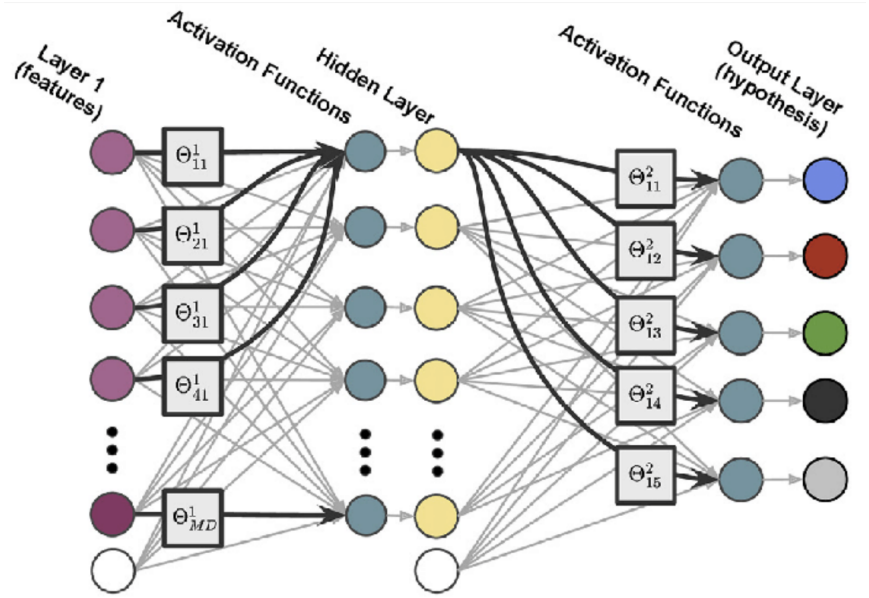


Figura 1: Modelo de perceptrón multicapa (MLP) utilizado en clasificación multiclase de redes neuronales artificiales [26].

El MLP consta de varias capas de neuronas, incluyendo una capa de entrada, una o varias capas ocultas y una capa de salida. Cada neurona en una capa está conectada a todas las neuronas en la capa siguiente, y cada conexión entre neuronas tiene un peso asociado que se puede ajustar durante el entrenamiento de la red. Durante este entrenamiento, el MLP se alimenta con un conjunto de datos y el proceso de aprendizaje ajusta los pesos de las conexiones entre neuronas para minimizar la diferencia entre las salidas de la red y los resultados esperados, lo que permite que los embeddings se adapten a los patrones de interacción entre los usuarios y los elementos. Una vez que se completa el entrenamiento, el MLP se utiliza para hacer predicciones o clasificaciones sobre nuevos datos [2].

Por último, mencionar que el MLP tiene ciertos hiper-parámetros que se pueden definir, bien por parte del implementador, o bien ajustar mediante un proceso de validación cruza-

da [23]; número de capas ocultas o número de neuronas por capa, entre otros. Esta técnica consiste en dividir el conjunto original de entrenamiento en k subconjuntos llamados pliegues (*folds*), donde cada uno se utiliza alternativamente como conjunto de validación y el resto como un nuevo conjunto de entrenamiento. De esta manera, se llevan a cabo k procesos de entrenamiento independientes en los que se evalúa el modelo utilizando diferentes combinaciones de parámetros de entrenamiento y validación. Así es posible seleccionar el valor óptimo de los hiper-parámetros, dando lugar a un modelo eficaz, robusto y bien generalizado.

3. Sistemas de Recomendación Clásicos

Un Sistema de Recomendación es una herramienta de filtrado de la información que tiene como objetivo recomendar elementos relevantes y personalizados a los usuarios, utilizando como base su historial de interacciones con el sistema, preferencias y patrones de comportamiento [25]. Para construir una sugerencia, los Sistemas de Recomendación requieren analizar información proveniente de diversas fuentes, detalladas en [3]. Por una parte, los ítems constituyen el conjunto de elementos objeto de recomendación. Estos elementos suelen estar representados por medio de atributos o características que ayudan al sistema a entender su naturaleza y a realizar recomendaciones más precisas y relevantes. Los usuarios componen el conjunto de personas que utilizan el sistema para obtener recomendaciones e interactúan con el conjunto de elementos anterior. En este marco aparece el tercer componente esencial de los Sistemas de Recomendación, el registro de interacciones entre usuarios e ítems. Denominadas transacciones, constituyen un pilar fundamental para aprender las preferencias del usuario y hacer recomendaciones personalizadas.

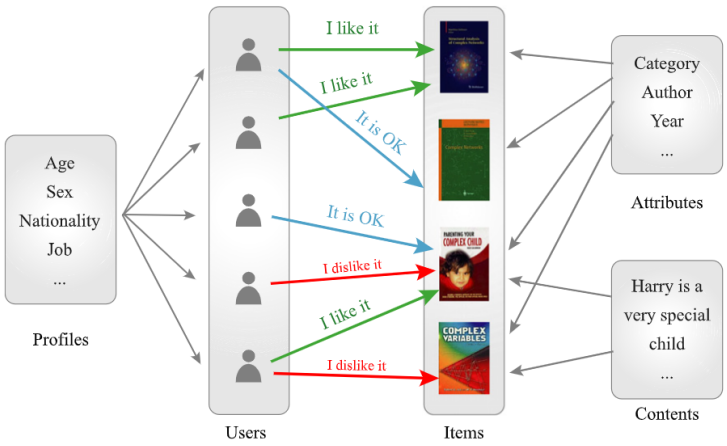


Figura 2: Grafo bipartido que representa la relación entre usuarios e ítems en un Sistema de Recomendación [19].

En un Sistema de Recomendación es necesario representar estas preferencias abstractas en vectores de características. De este modo surgen el perfil de usuario, el perfil de ítem y las valoraciones, que conforman la formalización matemática de las propiedades de los ítems y los usuarios [11].

- Perfil usuario: se aprende a partir de la información que el sistema recopila sobre el usuario. Se representa con un vector $u_i \in \mathbb{R}^r$, donde i refiere a cada usuario de la plataforma.

- Perfil item: descripción detallada de las características de cada elemento. Se representa con un vector $v_j \in \mathbb{R}^r$, donde j representa cada ítem del conjunto de datos.

- Valoración: se expresa mediante el valor r_{ij} y denota la valoración del usuario i al elemento j.

En lo que al tipo de información respecta, [11] desarrolla la idea de que los Sistemas de Recomendación tratan dos tipos diferentes de datos. Por una parte está la información *explícita*, que se puede obtener directamente de las interacciones de los usuarios con el sistema, como las valoraciones que estos dan a los ítems. Por otra, la *implícita*, información que se infiere a partir de datos pero no está explícitamente declarada. Es decir, esta información se encuentra "oculta" y debe ser deducida a través de análisis o procesamiento de datos.

3.1. Basado en Contenido

Los Sistemas de Recomendación basados en contenido son una técnica utilizada para hacer recomendaciones personalizadas a los usuarios en función de sus preferencias y comportamientos previos. Tal como se ha señalado en líneas anteriores, esto se logra mediante la creación de un perfil de usuario a partir de las características de los elementos consumidos y la aplicación de algoritmos de recomendación para encontrar elementos similares [14]. El perfil de cada ítem, v_j, recoge un despliegue de información implícita sobre cada elemento [16] . Conociendo la valoración que un usuario propicia a cada ítem, r_{ij}, el perfil de usuario se construye de acuerdo a la siguiente expresión:

$$u_i = \frac{\sum_{j \in S_i} r_{ij} v_j}{\sum_{j \in S_i} r_{ij}},$$

donde S_i es el conjunto de películas valoradas por el usuario i.

En otras palabras, el perfil de usuario es la suma ponderada de las características de las películas que ha valorado, donde el peso es la valoración que el usuario ha dado a cada película. Este vector se normaliza dividiéndolo por la suma de las valoraciones del usuario, de modo que la ponderación no se vea afectada por el número de valoraciones que este ha hecho. Este perfil de usuario es el que se utiliza posteriormente para calcular la similitud entre los elementos y hacer las recomendaciones. Con tal objetivo, se calcula la distancia entre el perfil de usuario y el de ítem. En este caso, se selecciona la similitud del coseno para medir la semejanza entre los perfiles. De acuerdo a [15], la expresión es como sigue:

$$d_c(u, v) = \frac{u \cdot v}{\|u\|_2 \|v\|_2}, \tag{1}$$

que toma valores en el intervalo [-1,1]. Valores próximos a 1 indican mayor similitud entre elementos.

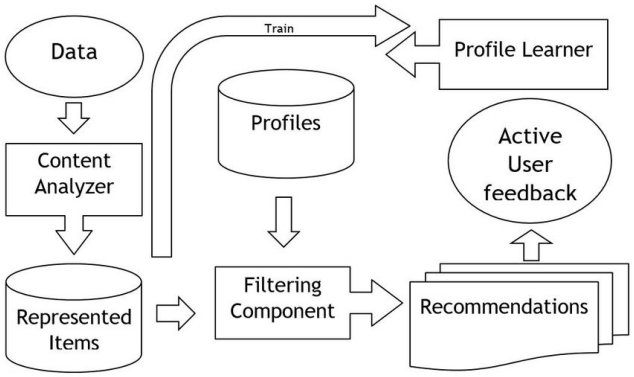


Figura 3: Representación de un Sistema de Recomendación basado en contenido mediante una arquitectura de alto nivel. Extracción de características de los ítems, entrenamiento del perfil usuario y recomendaciones en función de la similitud encontrada entre vectores [1].

Así pues, para cada usuario se calcula la similitud de su perfil con el de todas las películas que no ha valorado y se genera un ranking en el que los ítems de menor distancia, o más similares, ocupan las primeras posiciones, esto es, son los más recomendados. Sin embargo, se observa que el Sistema de Recomendación basado en contenido no devuelve la valoración de un usuario para una película, sino que simplemente recoge la similitud de este con el ítem. Por este motivo, y con el objetivo de poder implementar métricas de predicción posteriores, se establece el siguiente sistema de conversión, en el que a cada ítem recomendado se le atribuye una valoración proporcional a su grado de similitud. Considerando que el rating toma valores en el intervalo $[0, 5]$ la expresión es la siguiente.

$$\hat{r}_{ij} = 5 \cdot \frac{d_c(u_i, v_j)}{\max(d_c(u_i, v_j))},$$

donde $\hat{r}_{ij}$ representa el rating estimado.

3.2. Filtrado Colaborativo

El filtrado colaborativo es una técnica utilizada en Sistemas de Recomendación que tiene como objetivo predecir las preferencias de los usuarios en función de las valoraciones de otros usuarios o elementos. Esta herramienta se basa en la idea de que los usuarios o elementos que han tenido intereses similares en el pasado, es probable que también tengan preferencias similares en el futuro. Si bien el filtrado colaborativo devuelve una lista con los ítems más recomendados, también proporciona el rating que un usuario daría a un elemento concreto [15].

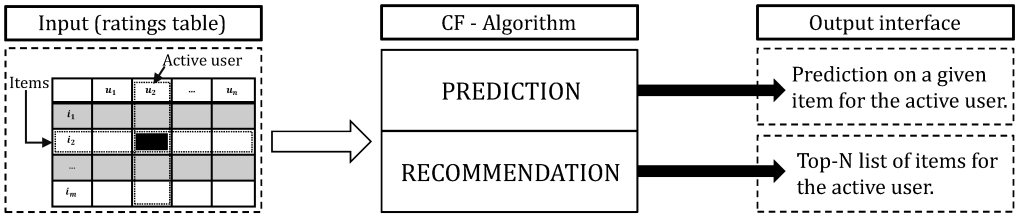


Figura 4: Diagrama de flujo de un Sistema de Recomendación basado en filtrado colaborativo. Recolección de datos de las interacciones usuario-ítem, construcción de una matriz de preferencias y generación de recomendaciones a partir de la similitud entre usuarios o elementos [21].

Un elemento esencial del filtrado colaborativo es la matriz de interacción usuario-elemento. Explicado en [15] , las filas representan a los usuarios y las columnas a los elementos, luego cada celda representa la interacción entre el usuario e ítem correspondientes. Generalmente tiene estructura dispersa, ya que es común que los usuarios interactúen con una pequeña fracción de los elementos disponibles, omitiendo un gran número de ítems.

3.2.1. Usuario - Usuario

El propósito principal del filtrado colaborativo usuario-usuario es encontrar elementos que puedan ser de interés para un usuario en función de las preferencias de otros usuarios similares. Para llevar a cabo este tipo de filtrado, se utiliza la matriz de interacciones mencionada, a partir de la cual se calcula una medida de similitud que permite obtener un conjunto M constituido por los k usuarios más cercanos sobre el usuario en estudio [5]. En este caso la medida de similitud seleccionada es la distancia coseno (1). Con ese propósito, en [5] se explica cómo se forman los vectores de usuarios, que en este caso son los vectores de valoraciones centralizados.

$$u_i = (r_{i1} - \bar{r_i}, r_{i2} - \bar{r_i}, \ldots, r_{in} - \bar{r_i}),$$

donde r_{ij} es la valoración del usuario u_i al elemento j, $\bar{r_i}$ es el promedio de todas las valoraciones del usuario i y n es el número total de elementos.

La fórmula indica que u_i es el usuario "objetivo" para el que queremos predecir su posible interés en un determinado ítem. Es un vector centralizado que, para cada elemento j, considera la diferencia entre la valoración de dicho usuario y su promedio de evaluaciones. Más aún, si el usuario i no ha valorado el elemento j, el valor u_{ij} correspondiente se reemplazará por cero. De esta manera, se pueden comparar directamente las preferencias de distintos usuarios en un espacio común y utilizar esta información para realizar recomendaciones personalizadas.

Finalmente, como se propone en [15], las valoraciones de los k usuarios más cercanos se utilizan para generar una predicción del rating que el usuario objetivo daría a un elemento en particular.

$$\hat{r}_{ij} = \frac{\sum_{w \in M} d_c(u_i, u_w) r_{wj}}{\sum_{w \in M} d_c(u_i, u_w)}$$

La implementación de esta media ponderada consiste en considerar las valoraciones de los usuarios más similares al usuario u_i, ponderando cada ítem de forma proporcional a la proximidad entre u_i y u_w, calculada mediante la distancia coseno (1). Este valor se normaliza para que la ponderación sea justa y no esté influenciada por la cantidad de usuarios similares seleccionados [15].

3.2.2. Item - Item

El filtrado colaborativo ítem-ítem presenta un enfoque en el que se utilizan las similitudes entre los elementos para hacer predicciones de calificación para los usuarios. A diferencia del filtrado colaborativo usuario-usuario, en este caso se calcula la similitud entre los ítems, en lugar de entre los usuarios [15].

La idea detrás del filtrado colaborativo ítem-ítem es que si a un usuario le gusta un elemento específico, entonces es probable que también le gusten elementos similares. Por lo tanto, se pueden hacer recomendaciones para un usuario basándose en elementos parecidos que ha valorado positivamente en el pasado. Para ello, lo primero que se necesita es determinar los vectores que representan a los ítems. De acuerdo con [15],

$$v_j = (r_{1j} - \bar{r_j}, r_{2j} - \bar{r_j}, \ldots, r_{mj} - \bar{r_j}),$$

donde v_j es el vector que representa al ítem j, r_{ij} es la calificación dada por el usuario i al ítem j, $\bar{r_j}$ es el promedio de las valoraciones de todos los usuarios para el ítem j y m representa al número total de usuarios.

En otras palabras, el vector de un ítem es una lista de las valoraciones centralizadas de todos los usuarios para ese elemento. Tal y como se sabe, no todos los usuarios valoran todos los ítems. Como consecuencia, para aquellos usuarios i que no han valorado el ítem j, se les asigna el valor 0 en la posición i del vector v_j.

Por último, cabe mencionar la importancia de estos vectores para calcular la similitud entre los ítems. Con ese fin, [5] propone la similitud del coseno (1), que permite construir un conjunto Q con los q elementos más próximos al ítem en estudio. De esta manera, para calcular la valoración de un usuario a un ítem no calificado, se pondera el rating de un elemento similiar de forma proporcional a su similitud. De nuevo, se normaliza este valor para evitar la influencia del número q de ítems similares seleccionados.

$$\hat{r}_{ij} = \frac{\sum_{q \in Q} d_c(r_j, r_q) r_{iq}}{\sum_{q \in Q} d_c(r_j, r_q)}$$

Este enfoque es verdaderamente útil cuando se dispone de un gran número de usuarios y relativamente pocos elementos, ya que permite aprovechar las similitudes entre elementos dando lugar a recomendaciones más precisas. Al mismo tiempo, constituye un modelo más escalable que el descrito en la Sección 3.2.1, pues permite lidiar mejor con la sobrecarga de información.

3.2.3. Factores Latentes

Mientras que las técnicas de filtrado colaborativo basadas en usuarios e ítems se centran en herramientas de memoria, los Sistemas de Recomendación basados en filtrado de factores latentes implementan modelos que pueden analizar interacciones complejas entre los usuarios e ítems. Estos modelos matemáticos utilizan técnicas avanzadas de aprendizaje automático, como la descomposición matricial, para identificar patrones ocultos en las evaluaciones de los usuarios y elementos [20]. De esta manera, las recomendaciones obtenidas son más precisas y personalizadas.

La idea fundamental de los modelos basados en factores latentes radica en la creación de vectores de características para cada usuario e ítem. Denominados embeddings, estos se aprenden de la matriz de interacción (4), que contiene las evaluaciones históricas de los usuarios para cada ítem. Estas variables latentes permiten representar características sub-

yacentes de los usuarios y elementos de manera compacta y eficiente, lo que permite una mejor modelización de las relaciones en la matriz de interacción.

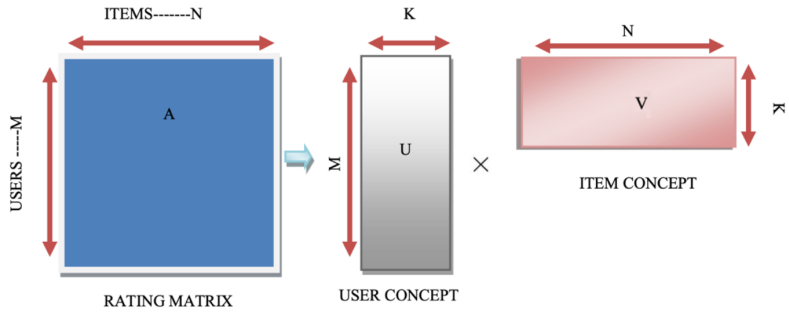


Figura 5: Descomposición matricial de la matriz de interaccion A en una matriz usuario U y una matriz ítem V, donde k es el número de factores latentes utilizados en la factorización [24].

En el contexto de los Sistemas de Recomendación, este enfoque de factorización de matrices es considerado el más preciso para abordar el problema de la escasez de datos en la matriz de interacción. La consecuencia de que todos los usuarios no valoren todos los ítems deriva en problemas de dispersión, situación que exige una reducción de la dimensionalidad para crear modelos más eficientes. En [20] se propone un método para resolver este problema de dispersión, reduciendo el tamaño de la matriz y reteniendo sólo las componentes más importantes de esta.

Como se observa en la Figura 5, el procedimiento consiste en obtener dos nuevas matrices de usuarios e ítems, U y V, cuyas filas y columnas representan los factores ocultos que determinan las preferencias de los usuarios.

$$A \approx U \times V,$$

donde A es la matriz de puntuaciones predichas, U es la matriz cuyas filas son los embeddings de los usuarios y V cuyas columnas son los embeddings de los elementos.

Estas matrices se aprenden a partir de los datos de entrenamiento mediante un proceso de optimización que minimiza el error cuadrático medio entre las valoraciones reales y las predichas [20]. Además, se utilizan regularizaciones λ_1 y λ_2 para evitar el sobreajuste del modelo. La función de pérdida aprendida en este proceso es:

$$\min_{U,V} \ \|A - U \times V\|_2 + \lambda_1 \|U\|_2 + \lambda_2 \|V\|_2.$$

Esta función de coste tiene tres términos. El primero representa la diferencia entre las valoraciones reales y las predichas, medida mediante la norma 2 o distancia euclídea[1]. El segundo y tercer término son las regularizaciones λ_1 y λ_2, que se aplican a las matrices U y V, respectivamente. Estas regularizaciones se utilizan para evitar el sobreajuste del modelo, lo que significa que el modelo se ajusta demasiado a los datos de entrenamiento y no generaliza bien a datos nuevos o desconocidos. Estas se modelan agregando términos adicionales a la función de pérdida para penalizar los valores extremos de los parámetros. El propósito de estas penalizaciones es limitar la complejidad del modelo y reducir la magnitud de los parámetros, en este caso, de las matrices. Al penalizar los valores más alejados se fomenta que los resultados sean más moderados, y por tanto, se ajusten mejor tanto a los datos de entrenamiento, como a datos nuevos.

Una vez que se han aprendido las matrices U y V mediante el proceso de optimización, estas se utilizan para hacer recomendaciones de elementos a los usuarios. Como ha sido mencionado, cada una de estas matrices contiene información oculta o subyacente, tanto de usuarios como de ítems. Estos vectores de factores latentes son valores aprendidos cuyo significado real se desconoce, es decir, son una representación abstracta de las características de los usuarios e ítems que el modelo ha considerado relevantes [20]. Esta abstracción matemática permite al modelo hacer predicciones precisas de las preferencias de los usuarios sobre los elementos. En concreto, se puede calcular la predicción de la valoración de un usuario para un ítem que no ha visto como el producto escalar entre la fila correspondiente de la matriz U y la columna correspondiente de la matriz V. Sin embargo, como se especifica en [22], es necesario recordar que no todos los usuarios valoran los elementos con el mismo criterio. Por este motivo, para poder considerar las diferentes formas de evaluación de los usuarios sobre los ítems, es común utilizar el término bias o sesgo. El bias se refiere a la corrección que se aplica a la predicción de la valoración de un usuario para un ítem, teniendo en cuenta su tendencia a valorar los elementos de cierta manera, o bien positiva, o bien negativa.

$$\hat{r}_{ij} = u_i \cdot v_j^T + u_0^{(i)} + v_0^{(j)},$$

donde u_i y v_j son los factores latentes, y $u_0^{(i)}$ y $v_0^{(j)}$ representan el sesgo de cada usuario y elemento, respectivamente.

En resumen, la reducción de la dimensionalidad es una técnica que se utiliza para extraer información subyacente importante de matrices dispersas y hacer recomendaciones a los usuarios. La implementación del proceso de optimización y las regularizaciones son importantes para evitar el sobreajuste del modelo y garantizar una buena generalización.

[1]La distancia euclídea de un espacio n-dimensional es $d_E(P,Q) = ||p_i - q_i|| = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$.

Finalmente, los factores latentes aprendidos a partir de los datos se utilizan para hacer predicciones de valoraciones y recomendaciones personalizadas de elementos a los usuarios.

3.3. Métricas de rendimiento en Sistemas de Recomendación

Las métricas de rendimiento son medidas cuantitativas que permiten evaluar la calidad de un modelo predictivo, además de permitir su comparación con implementaciones de otros modelos diferentes. Es claro que la elección de un criterio de evaluación alineado con el objetivo del problema es esencial para poder determinar adecuadamente el rendimiento de cada algoritmo. Es decir, para poder evaluar la bondad de las predicciones obtenidas por un Sistema de Recomendación, es imprescindible ceñirnos a la naturaleza de la técnica de filtrado implementada.

3.3.1. Evaluación de las predicciones

Las métricas de rendimiento de las predicciones tienen como objetivo determinar la precisión o exactitud de las recomendaciones realizadas por el sistema al evaluar ciertos elementos. Para ello, se compara la valoración del ítem obtenida a través del modelo, y la valoración original de dicho elemento. Al tratarse de un problema de regresión, la métrica más popular es el Error Cuadrático Medio (MSE, por sus siglas en inglés). Esta métrica mide la diferencia cuadrática promedio entre los valores predichos y los valores reales en un conjunto de datos [6].

$$MSE = \frac{\sum_{i \in U^t} \sum_{j \in I^t} (r_{i,j} - \hat{r}_{i,j})^2}{|R^t|}, \tag{2}$$

donde r_{ij} es la valoración del usuario i sobre el ítem j, $\hat{r}_{i,j}$ es la predicción de dicha valoración y R^t acumula todos los pares usuario-ítem que pertenecen al conjunto de *test*.

3.3.2. Métricas específicas para Sistemas de Recomendación

La métrica descrita anteriormente ofrece resultados representativos del funcionamiento de los Sistemas de Recomendación. No obstante, se centra en un único ítem para cada usuario, cuando verdaderamente el resultado de un Sistema de Recomendación es una lista de elementos ordenados de mayor a menor probabilidad de satisfacción. En este contexto nacen las técnicas de rendimiento específicas para Sistemas de Recomendación, en las que cada sugerencia se representa en forma de lista donde los elementos más recomendados ocupan posiciones superiores.

- Ganancia Acumulada Descontada Normalizada (nDCG): métrica de ranking que se utiliza en sistemas de recomendación para medir la calidad de las recomendaciones en función de su posición en una lista. La idea detrás de la nDCG es que una recomendación en una posición superior es más valiosa que una recomendación en una posición inferior [4]. La métrica toma en cuenta tanto la relevancia del elemento recomendado, como su posición en la lista, otorgando una puntuación más alta a los elementos relevantes que aparecen en posiciones superiores. Además, valora/penaliza más intensamente los aciertos/fallos producidos en las primeras posiciones del registro [12]. Antes de analizar la métrica en cuestión, es necesario introducir dos medidas auxiliares que se utilizan en su construcción. En primer lugar,

$$DCG_i = \sum_{j \in R_i^T} \frac{2^{rel_j} - 1}{log_2(j+1)},$$

donde R_i^T representa el conjunto de elementos recomendados al usuario i y rel_j es la relevancia o valoración del ítem en la posición j. Por tanto, la relevancia rel_j para cada usuario i será la recomendación obtenida por el modelo, $\hat{r}_{ij}$.

Esta expresión refleja la ganancia acumulada descontada para la lista de recomendaciones del usuario i. Se trata de una medida que evalúa la calidad de las recomendaciones devueltas por el sistema, tanto por la relevancia otorgada a cada ítem como por la posición que este ocupa en la lista. Además, en el denominador de esta fórmula se utiliza una función de descuento logarítmico para dar más peso a los elementos relevantes que aparecen al principio de la lista [12].

Más aún, [12] especifica la expresión ideal de la métrica recién vista.

$$IDCG_i = \sum_{j \in R_i^T} \frac{2^{rel_j} - 1}{log_2(j_i+1)},$$

donde R_i^T es el conjunto de elementos recomendados al usuario i, rel_j es la relevancia del ítem en la posición j y j_i representa la posición ideal del ítem j.

Esta fórmula, conocida como el Ideal DCG, representa la ganancia acumulada descontada perfecta, esto es, el mejor valor posible de DCG que se puede obtener a partir de una lista de recomendación dada [4]; los elementos más relevantes ocupando las posiciones más altas.

Por último, con todos los elementos explicados, se calcula la métrica $nDCG_i$ para cada usuario i.

$$nDCG_i = \frac{DCG_i}{IDCG_i},$$

El valor global de la métrica se obtiene calculando la media de todos los valores de $nDCG_i$ individualizados [4].

$$nDCG = \frac{\sum_{i \in U^t} nDCG_i}{|U^t|}, \tag{3}$$

donde U^t es el conjunto de usuarios de prueba.

- Precisión Media Media (MAP): métrica de evaluación comúnmente utilizada para evaluar el rendimiento de un modelo en términos de su capacidad para organizar los elementos recomendados en un orden que refleje su relevancia. Para cada usuario, se compara la lista de elementos recomendados con una lista de elementos relevantes, y se calcula la precisión media de la lista de ítems recomendados [17].

$$MAP = \frac{\sum_{i \in U^t} AP(i)}{|U^t|}, \quad \text{con} \quad AP(i) = \frac{\sum_{j \in I^t} \hat{r}_{ij} \cdot rel(j)}{|I^t|}, \tag{4}$$

donde U^t es el conjunto de usuarios de prueba, I^t es el conjunto de ítems de prueba, $\hat{r}_{ij}$ es la puntuación predicha para el ítem j por el modelo para el usuario i y $rel(j)$ es un indicador binario que indica si el ítem j es relevante o no para el usuario i.

La fórmula superior se compone de dos partes principales. El numerador lo constituye la precisión promedio para cada usuario i, $AP(i)$. Esta medida se calcula dividiendo la suma de las puntuaciones predichas $\hat{r}_{ij}$ de los ítems relevantes j para el usuario i, por el número total de ítems de prueba I^t. Este cociente se multiplica por el indicador $rel(j)$ correspondiente al ítem j. Este parámetro toma el valor 1 si el ítem j está en la lista de elementos relevantes para el usuario i, y 0 en caso contrario [17]. El umbral establecido para considerar un elemento importante es que la similitud coseno obtenida entre cada usuario e ítem sea superior a 0,5, es decir, $d_c(u, v) \geq 0{,}5$. Por otra parte, el denominador está compuesto por el número total de usuarios en el conjunto de prueba. De este modo, la precisión promedio $AP(i)$ se calcula para cada usuario i y se promedia para todos los usuarios con el objetivo de obtener el valor final de MAP.

Ambas métricas de evaluación toman valores en el intervalo $[0, 1]$, y cuanto mayor sea su valor, mejor será la recomendación. No obstante, aunque en un primer lugar estas dos métricas parezcan similares, realmente no lo son. Su diferencia fundamental radica en cómo ponderan y evalúan los resultados obtenidos. Mientras que la métrica nDCG se centra en

la posición y relevancia de las recomendaciones, la métrica MAP evalúa y compara la precisión media de la lista recomendada frente a la relevante. Debido a estas diferencias en su implementación, es posible que un Sistema de Recomendación obtenga una puntuación alta en nDCG, pero valores bajos en MAP, o viceversa. Un rendimiento elevado de la métrica nDCG indica que las recomendaciones están bien posicionadas en la lista y son relevantes, pero no garantiza necesariamente una buena organización en términos de relevancia. Por otra parte, un elevado valor de MAP indica que, en cuanto a relevancia, las recomendaciones están bien organizadas. De igual manera, este hecho no implica necesariamente un buen posicionamiento de las recomendaciones en el ranking.

En resumen, ambas métricas proporcionan perspectivas diferentes sobre la calidad de las recomendaciones. Así pues, es interesante considerarlas en conjunto para tener una evaluación completa y precisa del rendimiento del Sistema de Recomendación.

4. Collaborative Metric Learning

Esta sección está enfocada en estudiar en detalle la conexión entre el aprendizaje de métricas y el filtrado colaborativo, dando lugar a una herramienta de recomendación avanzada: el *Aprendizaje Métrico Colaborativo* o CML, desarrollado en detalle en [11].

El CML es un enfoque de recomendación que se basa en el aprendizaje de métricas para mejorar la calidad de las recomendaciones personalizadas. El principal propósito del CML es optimizar una métrica que refleje la similitud de los usuarios y los ítems en un espacio de características compartido, de manera que se puedan descubrir patrones complejos y representaciones más sofisticadas de los datos. En Sistemas de Recomendación clásicos, como hemos visto en la Sección 3, los modelos se limitan a calcular similitudes entre usuarios e ítems, sin aprender explícitamente las métricas implementadas. Es decir, estas métricas predefinidas, como la similitud coseno (1), son establecidas por adelantado por el diseñador del sistema y se calculan para cada par (usuario, ítem). Como consecuencia, el modelo no tiene en cuenta las características subyacentes de los datos y puede ignorar relaciones complejas entre usuarios e ítems.

En contraste, el CML propone un enfoque de aprendizaje conjunto de métricas que permite capturar tanto las similitudes entre los usuarios y los ítems, como las relaciones complejas y no lineales entre ellos. La parte "colaborativa" se refiere a la forma en la que se utiliza la información de las interacciones para aprender las métricas de similitud. Sin embargo, aunque se hable de aprendizaje de métricas, técnicamente lo que se aprenden son los vectores que representan a los usuarios y los elementos, no directamente la métrica en sí. En efecto, el proceso de entrenamiento está enfocado en aprender una transformación de estos vectores, lo que provoca que se redefina la distancia entre ellos. Es decir, lo que verdaderamente se aprende es la representación vectorial de los usuarios y elementos, pero se realiza de tal manera que maximiza la similitud entre los vectores de los usuarios e ítems que han sido evaluados positivamente, mientras minimiza la similitud de los restantes. De esta manera, el proceso de aprendizaje vectorial está diseñado para optimizar la métrica de similitud entre los vectores aprendidos, por lo que se cataloga como "aprendizaje de métricas", aunque verdaderamente lo que se aprenden son estas representaciones vectoriales, esto es, los embeddings.

En lo que al tratamiento de la información respecta, el CML modela el feedback implícito observado como un conjunto de pares usuario-ítem con relaciones positivas. De hecho, aunque la idea fundamental del modelo es aprender una métrica que minimice la distancia

con los elementos más similares, gracias a la desigualdad triangular[2], este proceso agrupará tanto usuarios a los que les gustan los mismos ítems, como ítems que gustan a usuarios próximos. Asimismo, los ítems vecinos más cercanos para un usuario concreto serán:

1. Elementos que previamente le gustaron al usuario.
2. Elementos que les gustaron a usuarios que comparten gustos similares con este.

Al aprender las métricas de similitud de forma conjunta, el modelo se puede focalizar no sólo en la similitud entre usuarios o elementos, sino que puede considerar ambas situaciones simultáneamente. De este modo, es posible extender las relaciones positivas conocidas a otros pares, como usuario-artículo, usuario-ítem e ítem-ítem, incluso cuando estas relaciones no se observan directamente en los datos. Todo ello hace que el CML sea capaz de producir recomendaciones más precisas y relevantes para los usuarios, al tiempo que mejora la escalabilidad y la personalización de los modelos de recomendación. Los próximos apartados se centrarán en el desarrollo de estas ideas en un contexto más formal, incorporando una explicación detalla del algoritmo CML, su proceso de aprendizaje, mapeo de características, entrenamiento y regularización.

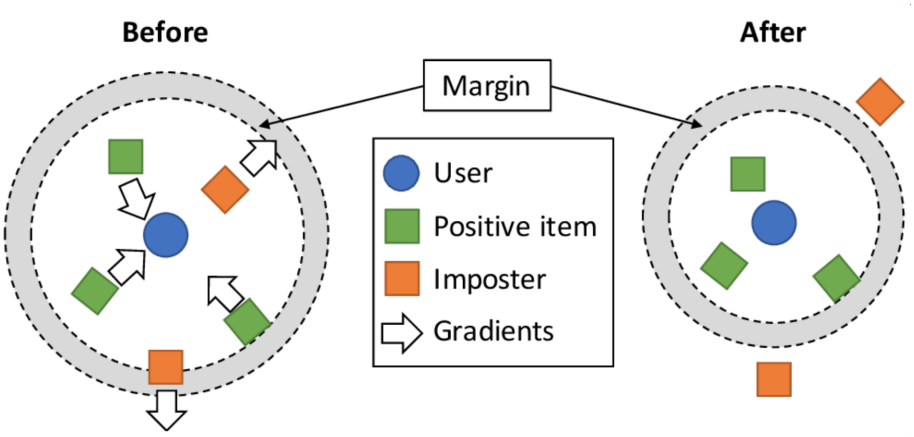


Figura 6: Diagrama del proceso de aprendizaje de métricas colaborativas de similitud. El gradiente acerca los elementos positivos al usuario y aleja los disímiles o "impostores" hasta superar el margen establecido [11].

[2]Dados tres objetos x, y, z, la suma por pares de dos distancias cualesquiera debe ser mayor o igual que la distancia por pares restante. Dado "x similar a y y z", esto acercará no sólo los pares indicados, sino también (y, z). Sea (y, z) limitada por la suma de las distancias entre (x, y) y (x, z): $d(y, z) \leq d(x, y) + d(x, z)$.

4.1. Aprendizaje de métricas de similitud

Esta subsección tiene como objetivo presentar las características más importantes del aprendizaje de métricas de similitud y el filtrado colaborativo, así como su enfoque complementario.

El algoritmo CML aprende una métrica que permite calcular la similitud entre usuarios o elementos en función de las interacciones pasadas entre ambos. Como concepto diferenciador respecto a los sistemas clásicos, este algoritmo busca aprender una función de similitud que transforme el espacio original de características en uno en el que los elementos similares estén más cerca entre sí que los diferentes.

Es importante mencionar que en el CML la información que se modela es implícita, es decir, se obtiene a partir del comportamiento y las interacciones de los usuarios con los elementos del sistema; "me gustas", clics, marcadores, etc. Este *feedback* es más abundante y está menos sesgado que la retroalimentación explícita, aunque presenta un inconveniente mayor: sólo se observa información implícita positiva. Las no interacciones entre los pares usuario-ítem pueden deberse a numerosos factores, como inatención, falta de interés o desconocimiento. Por este motivo, la información implícita no constatada no puede considerarse negativa, por lo que, como veremos en la próxima Sección 4.2, se han desarrollado diversas técnicas para abordar este problema y obtener información más completa.

A raíz de lo anteriormente expuesto, es necesario especificar un conjunto S que determine, por pares, si dos objetos son similares o no. Sea $X = \{x_1, x_2, \ldots, x_n\} \in \mathbb{R}^m$, la información se presenta por restricciones por pares:

$$S = \{(x_i, x_j) |\ x_i \text{ y } x_j \text{ son similares.}\}.$$

El conjunto S está compuesto por pares usuario-ítem que tienen relaciones positivas, mientras que un par $(u_i, v_j) \notin S$ implicará disparidad entre los objetos. El objetivo del CML es aprender una métrica conjunta usuario-ítem que codifique estas relaciones, haciendo especial hincapié en acercar los pares de objetos de S y alejar los restantes. El conjunto S se construye de acuerdo a [30] como sigue:

1. Se construye una matriz de interacción usuario-ítem $\mathbb{R}^{m \times n}$, donde m es el número de usuarios y n es el número de ítems. El elemento r_{ij} de la matriz $\mathbb{R}$ indica la interacción del usuario i con el ítem j, por ejemplo, el número de veces que el usuario ha visto el ítem o ha hecho clic en él.

2. Se aplica un algoritmo de factorización a la matriz de interacciones para obtener una representación latente de los usuarios y los ítems. Tal como se ha descrito en la Sección 3.2.3, se obtienen dos matrices de factores latentes $\mathbb{U} \in \mathbb{R}^{m \times k}$ y $\mathbb{V} \in \mathbb{R}^{n \times k}$, donde k es la dimensión latente y cada fila de $\mathbb{U}$ y columna de $\mathbb{V}$ representan los factores subyacentes de los usuario e ítems, respectivamente.

3. Finalmente, a partir de la representación latente se construye el conjunto S de pares (usuario, ítem) semejantes. En particular, se define una medida de similitud, s_{ij}, que representa los gustos o preferencias del usuario i sobre el ítem j. Se calcula de acuerdo a la similitud del coseno (1), $s_{ij} = \frac{u_i^\top v_j}{|u_i||v_j|}$, donde u_i y v_j son los vectores latentes del usuario i y del elemento j. Por último, se seleccionan los pares (usuario, ítem) con las k medidas de similitud más altas para formar el conjunto S.

4.2. Formulación del modelo

Una vez especificado el conjunto S de pares (usuario, ítem) similares, las preferencias de estos usuarios por los elementos van a estar determinadas por la distancia euclídea:

$$d(i, j) = ||u_i - v_j||.$$

La formalización del modelo busca aprender una representación vectorial de los usuarios y elementos que permita definir una métrica de similitud adecuada. En este sentido, se busca que los vectores de los pares usuario-elemento que tienen relaciones positivas estén más cercanos entre sí, y aquellos que son disímiles estén más lejos. En otras palabras, aunque técnicamente lo que se aprenden son las representaciones de usuarios y elementos, el objetivo es optimizar la métrica de similitud entre ellas a través de la minimización de la función de pérdida (5), que penaliza la distancia euclídea entre los vectores de los pares usuario-ítem que están relacionados positivamente.

$$L_m(d) = \sum_{(i,j) \in S} \sum_{(i,k) \notin S} w_{ij} \left[m + d(i, j)^2 - d(i, k)^2 \right]_+ , \tag{5}$$

donde j es un ítem valorado positivamente por el usuario i, y k es un elemento que no le gustó; w_{ij} es un peso que se asigna a cada par usuario-ítem (u_i, v_j), $m > 0$ es una constante que representa el tamaño del margen de seguridad y $[z]_+ = max(z, 0)$.

La función de pérdida busca minimizar la distancia entre los vectores usuario e ítem para los pares que han sido calificados positivamente y maximizarla para aquellos pares (u_i, v_j) que no pertenecen al conjunto S. Para lograr esto, se penaliza la diferencia entre la distancia

umbral m y la distancia euclídea de un par usuario-ítem positivo mediante la función $[z]_+$. Esto refleja que el usuario prefiere ese elemento si la distancia $d(i,j) \leq m$.

Por otro lado, la diferencia entre la distancia euclídea de un par usuario-ítem negativo y la misma distancia umbral se ignora, ya que los pares negativos no son relevantes para el aprendizaje del modelo. Al tratar con relaciones negativas en las que $d(i,k) \geq m$, la función $[z]_+$ devuelve el valor 0, por lo que no se agrega ninguna penalización. En suma, reflejado en la Figura 6, la función de pérdida (5) genera gradientes que acercan los elementos positivos al usuario y alejan los elementos impostores hasta que superan el margen de seguridad.

Como último detalle, los pesos w_{ij} permiten ajustar la contribución de cada par usuario-ítem en la función de pérdida, lo que puede ser útil para equilibrar la influencia de los pares de alta y baja calidad. A continuación, J denota el número total de elementos y $rank_d(i,j)$[3] es la posición en el ranking del ítem j para el usuario i. Como ha sido explicado en líneas previas, por la construcción de la función de pérdida esta ponderación se hará únicamente sobre pares con relación positiva, tal que:

$$w_{ij} = log(rank_d(i,j) + 1).$$

Este esquema penaliza más altamente un par usuario-ítem en una posición inferior que en una superior. Sin embargo, calcular $rank_d(i,j)$ en cada paso del descenso de gradiente es costoso. Por este motivo, se propone un procedimiento de muestreo secuencial que considera repetidamente un elemento negativo hasta encontrar un impostor, es decir, un elemento que siendo negativo, no se ha catalogado como tal (está dentro del margen de seguridad). Concretamente el proceso consiste en muestrear U elementos negativos para cada par (i,j), generalmente entre 10 y 20. Para cada uno de estos elementos se calcula su valor en la función de pérdida (5). A continuación, M representa el número de impostores encontrados en el subconjunto U, esto es, aquellos ítems que devuelven un valor distinto de 0 en la función de pérdida mencionada. Con todo ello, el $rank_d(i,j)$ se aproxima por el valor $\left[\frac{J \times M}{U}\right]$.

De esta manera, lo que se logra es suavizar la influencia de los pares con alta similaridad, mientras que se penalizan con mayor peso los pares más disímiles. Esto puede ser útil para evitar que los pares de baja calidad dominen la función de pérdida y afecten negativamente el rendimiento del modelo.

[3] $0 \leq rank_d(i,j) < J$, con $rank_d(i,j) = 0$ para el elemento más recomendado.

4.3. Mapeo de características

En un Sistema de Recomendación, los elementos (como películas o productos) se describen mediante ciertas características. La idea detrás de la integración de estas características en un modelo de recomendación es aprender una función de transformación f que proyecte estas propiedades en un espacio común de usuario-ítem.

En este enfoque, se considera que cada elemento j tiene un vector de características sin procesar $x_j \in R^m$, es decir, un vector de información que no ha sido tratado previamente. La función f aprende a proyectar estos vectores de características en un espacio común usuario-elemento de menor dimensión, donde se pueden calcular las similitudes entre los elementos y los usuarios. El vector resultante, $f(x_j)$, representa la ubicación del elemento j en el espacio compartido.

Por otro lado, el vector v_j representa la verdadera ubicación del elemento j en el espacio común. Este vector se ajusta durante el entrenamiento del modelo para maximizar la precisión de la recomendación. El objetivo es que el vector resultante $f(x_j)$ se acerque lo máximo posible a v_j en el espacio común usuario-elemento. La función de pérdida utilizada para ajustar la proyección de los ítems en este espacio compartido se define como:

$$L_f(\theta, v_*) = \sum_j ||f(x_j, \theta) - v_j||^2, \tag{6}$$

donde θ es un parámetro de la función f y v_* es el conjunto de vectores de elementos que se ajustan durante el entrenamiento.

El objetivo de esta función de pérdida es minimizar la distancia euclídea entre la ubicación proyectada de los elementos $f(x_j)$ y su verdadera ubicación v_j. Esto es, se penaliza la ubicación final del ítem j, v_j, cuando esta se desvía de $f(x_j)$, lo que contribuye a mejorar la precisión de las recomendaciones y a mantener una representación coherente de los elementos en el espacio común.

Específicamente, la función de transformación f es una red neuronal multicapa, MLP, descrita en la Sección 2.2. La idea es que la MLP aprenda una transformación no lineal de los vectores de características sin procesar, x_j, para mapearlos en un espacio común de usuario-elemento. La función de transformación f aprende a recoger las características que son más relevantes para las preferencias de los usuarios, a la vez que los elementos con características similares se agrupan juntos en el espacio común. Con tal motivo, la función f es informada por v_*, y viceversa.

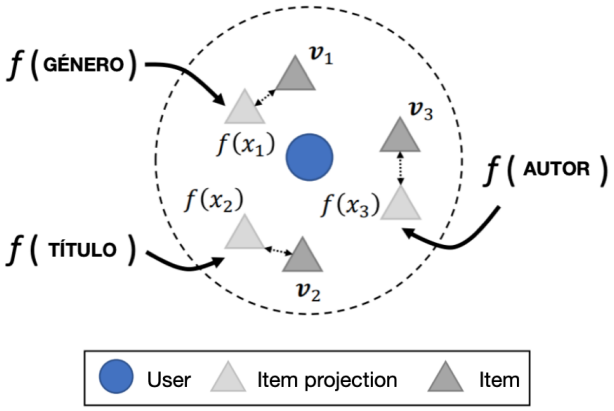


Figura 7: Proyección de las características de los elementos de un Sistema de Recomendación basado en películas en el espacio conjunto usuario-ítem. Las proyecciones de los elementos en el espacio buscan situarse cerca de su verdadera ubicación. Imagen inspirada en [11].

Si bien el proceso de entrenamiento será explicado en la siguiente sección, mencionar que durante el entrenamiento del modelo, la función de transformación f y los vectores de elementos v_j se ajustan simultáneamente para minimizar la función de pérdida L_f (6) y la función de pérdida de métrica L_m (5).

En resumen, la integración de las características de los elementos en un Sistema de Recomendación supone la proyección de estas en un espacio común de usuario-elemento. Este mapeo de características es necesario para mejorar la precisión de las recomendaciones, ya que se aprende una representación latente de los elementos y su relación con los usuarios. Además, el aprendizaje de la función de transformación f ayuda a mejorar la capacidad del modelo para hacer recomendaciones precisas y personalizadas, ya que captura las características relevantes de los elementos y penaliza su ubicación en el espacio común cuando se alejan de esta representación.

4.4. Regularización del espacio compartido

En lo que al aprendizaje automático respecta, la regularización se refiere a las técnicas que se utilizan para evitar el sobreajuste en los modelos. Este sobreaprendizaje ocurre cuando un modelo se ajusta demasiado bien a los datos de entrenamiento y, como resultado, se vuelve menos efectivo en la predicción de nuevos ejemplos. La regularización implica la inclusión de términos adicionales en la función de pérdida con el propósito de penalizar durante el entrenamiento valores extremos de los parámetros y evitar así el sobreajuste.

En el contexto del CML, un esquema de regularización es importante para garantizar que los vectores latentes de los usuarios y los ítems no estén demasiado correlacionados en el espacio conjunto en el que están proyectados. Esto es importante porque, en general, los modelos basados en vecinos más cercanos tienden a ser menos efectivos en espacios de alta dimensionalidad si los datos se extienden demasiado. En la implementación de este algoritmo se aplican dos técnicas de regularización diferentes: la primera refiere a una restricción adicional de los vectores u_* y v_*, mientras que la otra sugiere la regularización de la matriz de covarianza C. Ambas se aplican con el objetivo de garantizar la robustez de las métricas aprendidas. La primera restricción establece que la norma L^2 de todos los vectores u_* y v_* debe de estar dentro de la esfera unitaria:

$$||u_*||^2 \leq 1 \quad \text{y} \quad ||v_*||^2 \leq 1.$$

Cabe destacar que las ecuaciones anteriores se toman como restricciones del problema, no como regularizaciones. Esto significa que no forman parte del proceso de entrenamiento, sino que simplemente restringen que los vectores u_* y v_* se extiendan demasiado en el espacio conjunto. De hecho, si se utilizara una regularización L^2, los vectores se verían atraídos hacia el origen del espacio común usuario-ítem; las normas de los vectores irían reduciéndose y tendiendo a 0, lo que carece de sentido.

Por otra parte, el núcleo del esquema de regularización lo conforma la matriz de covarianzas C, utilizada para reducir la correlación entre las dimensiones de los vectores latentes. Sea y_*^n el vector latente de un objeto, bien un usuario, o bien un elemento, las covarianzas entre todos los pares de dimensiones i y j forman la matriz C:

$$C_{ij} = \frac{1}{N} \sum_n (y_i^n - \mu_i)(y_j^n - \mu_j),$$

donde $\mu_i = \frac{1}{N} \sum_n y_i^n$ y N es el tamaño del conjunto de datos, es decir, el número total de usuarios y elementos. Por último, n es el índice de la muestra de datos, esto es, toma valores desde 1 hasta N para indicar cada uno de los objetos.

En definitiva, la matriz C es simétrica y definida positiva, e indica la variación y relación entre las dimensiones de los vectores latentes. Además, se aplica por igual a los vectores latentes de los usuarios y de los ítems, ya que y_* es cualquier objeto de dicho espacio compartido. La regularización de la matriz C implica agregar un término adicional a la función de pérdida del modelo que penaliza la matriz por desviarse demasiado de la matriz identi-

dad. Esto se hace para evitar que las dimensiones de los vectores latentes estén altamente correlacionadas, lo que podría conducir a una menor eficacia del modelo. La función de pérdida L_c correspondiente es:

$$L_c = \frac{1}{N}(||C||_f - ||diag(C)||_2^2), \tag{7}$$

donde $||\cdot||_f$ es la norma Frobenius[4] y $||diag(C)||_2^2$ se refiere a la suma de los cuadrados de los elementos diagonales de la matriz C. En otras palabras, $||diag(C)||_2^2$ es la suma de los cuadrados de las varianzas de cada dimensión del espacio de características latentes.

En resumen, la regularización en el contexto del CML se utiliza para garantizar que los vectores latentes sean robustos y no se extiendan demasiado en el espacio conjunto, además de para reducir la correlación entre las dimensiones de estos vectores. Asimismo, la función de pérdida (7) fomenta que las dimensiones redundantes se desanexen y se utilice de manera más eficiente el espacio común usuario-ítem.

4.5. Proceso de entrenamiento

La función objetivo que propone el modelo CML trata de minimizar las funciones de pérdida definidas en apartados anteriores.

$$\min_{\theta, u_*, v_*} \quad L_m + \lambda_f L_f + \lambda_c L_c, \quad \text{sujeto a} \quad \|u_*\|^2 \leq 1 \quad \text{y} \quad \|v_*\|^2 \leq 1, \tag{8}$$

donde θ, u_* y v_* son los parámetros del modelo que se busca optimizar. Los hiper-parámetros λ_f y λ_c controlan la importancia relativa de las dos últimas funciones de pérdida. Se establecen antes del entrenamiento del modelo según la experiencia y conocimiento previo del problema en cuestión. De forma adicional y en líneas generales, las funciones de pérdida se resumen en:

- L_m es la función de pérdida definida para minimizar la distancia entre los pares de vectores usuario e ítem positivos, y maximizarla para aquellos pares disímiles. Esta función se define en la Ecuación (5).
- L_f es la función de pérdida (6) para el aprendizaje de métricas en el espacio compartido usuario-elemento. Esta función se encarga de minimizar la distancia entre las proyecciones de los vectores de características en el espacio compartido.

[4]La norma de Frobenius de una matriz se expresa como: $||C||_f = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|c_{ij}|^2}$

- L_c controla la regularización de los vectores de características de los usuarios y elementos. Se encarga de evitar que los vectores se extiendan demasiado en el espacio conjunto y se define en (7).

Durante el proceso de entrenamiento se utiliza el algoritmo Mini-Batch Stochastic Gradient Descent (SGD) para minimizar la función objetivo (8) propuesta por el modelo CML. La selección de esta estrategia se debe a que la cantidad de registros de usuarios y elementos puede ser muy grande, lo que hace que el uso de un algoritmo de descenso por gradiente tradicional sea lento y computacionalmente costoso. Para superar este problema, se utiliza el algoritmo Mini-batch SGD, que actualiza los parámetros utilizando solo un lote de registros en lugar de todos los datos de entrenamiento, reduciendo notablemente el tiempo de procesamiento [29]. En general, el mini-batch SGD ofrece un buen equilibrio entre velocidad de procesamiento y precisión del modelo para problemas de Sistemas de Recomendación.

Por otra parte, el grado de aprendizaje se controla mediante la técnica de optimización *AdaGrad* (Algoritmo de Gradiente Adaptativo) [18]. En el contexto del algoritmo Mini-batch SGD, en cada iteración se calcula el gradiente de un subconjunto de los datos de entrenamiento y se utiliza para actualizar los parámetros del modelo. Estos vectores de dirección se comportan de acuerdo a la Figura 6, donde se observa cómo los gradientes de los diferentes lotes pueden variar en magnitud y dirección. En este sentido, AdaGrad se encarga de ajustar la magnitud de la actualización (tamaño del paso de aprendizaje o *learning rate*) de manera adaptativa. Para el caso de la función objetivo (8), los parámetros que se busca optimizar son θ, u_* y v_*. Al utilizar AdaGrad, se adapta el tamaño del paso de aprendizaje para cada uno de estos parámetros, lo que permite al algoritmo converger más rápido y de manera más estable.

En definitiva, el proceso de aprendizaje puede dividirse en un conjunto de etapas o fases, que incluyen:

1. Seleccionar N pares positivos (usuario, ítem) del conjunto S.
2. Para cada par en N, seleccionar U elementos negativos y aproximar el ranking $rank_d(i, j)$ de cada par utilizando el método descrito en la Sección 4.2.
3. Construir los gradientes de la función objetivo y actualizar los parámetros del modelo utilizando una tasa de aprendizaje controlada por AdaGrad.
4. Aplicar la restricción de norma L_2 a u_* y v_*, tal que: $y' = \frac{y}{\text{máx}(||y||,1)}$
5. Repetir el proceso hasta que el modelo converja.

En suma, el proceso de entrenamiento trata de minimizar la función objetivo definida en la ecuación (8), utilizando el algoritmo de Mini-Batch SGD con los hiper-parámetros λ_f y λ_c para controlar la importancia relativa de las funciones de pérdida. La técnica de optimización AdaGrad se utiliza para controlar el tamaño del paso de aprendizaje de manera adaptativa para cada parámetro y la actualización de los parámetros se realiza mediante el método Mini-Batch SGD, que utiliza los gradientes para ajustar los parámetros en la dirección que minimiza la función de pérdida.

Finalmente, en el proceso de entrenamiento se aprenden los parámetros θ, u_* y v_*. El parámetro θ se refiere a los pesos que se asignan a las diferentes características de los usuarios e ítems en la función de pérdida, lo que determina su importancia relativa en la recomendación. Por ejemplo, en un Sistema de Recomendación de películas, las características de un ítem podrían ser el título, género o autor, entre otros. El parámetro θ controla cómo se combinan estas características para determinar la relevancia de una película para un usuario en particular.

Por su parte, los parámetros u_* y v_* capturan las relaciones no lineales entre los usuarios e ítems en el espacio compartido. En otras palabras, los vectores u_* y v_* permiten que el modelo encuentre patrones ocultos en los datos de entrenamiento que pueden ser útiles para hacer recomendaciones precisas. Una vez que se han aprendido estos vectores, se pueden utilizar para hacer predicciones precisas de las preferencias de los usuarios sobre los elementos.

4.6. Predicciones con CML

Una vez entrenado el modelo y aprendidos los vectores u_* y v_* que representan a los usuarios y elementos, es posible utilizarlos para hacer predicciones de las preferencias de los usuarios sobre nuevos ítems. Dado un usuario i y un elemento j, la predicción de preferencia $\hat{r}_{ij}$ se calcula mediante la fórmula:

$$\hat{r}_{ij} = \langle u_i, v_j \rangle = u_i \cdot v_j, \tag{9}$$

donde u_i y v_j son los vectores que representan al usuario i y al elemento j, respectivamente. $\langle \cdot, \cdot \rangle$ denota el producto escalar entre los vectores.

En resumen, la predicción se obtiene proyectando el usuario y el elemento en el espacio latente, donde se evalúa su similitud. Es importante tener en cuenta que el proceso de aprendizaje del CML busca encontrar los vectores u_* y v_* que minimizan la función objeti-

vo definida en el proceso de entrenamiento (8). Por lo tanto, una vez que se han aprendido los vectores, se utilizan para calcular las predicciones de preferencia de los usuarios para elementos no vistos anteriormente y así generar recomendaciones personalizadas.

Este enfoque de recomendación aprovecha la capacidad del modelo para capturar relaciones latentes entre los usuarios y los elementos, proporcionando una forma efectiva de predecir y recomendar elementos relevantes. Al utilizar el producto escalar entre los vectores de usuarios y elementos, se establece una medida de similitud que refleja la afinidad entre ambos. Por lo tanto, cuanto mayor sea el valor de $\hat{r}_{ij}$, mayor será la predicción de preferencia del usuario i hacia el elemento j.

5. Comparación de modelos

Al tratar de evaluar el grado de interés de un usuario sobre un conjunto de elementos aparecen diversos problemas colaterales que afectan al rendimiento y efectividad del proceso de recomendación. A continuación, se enumeran las dificultades más relevantes propias de estas herramientas [15] [7].

1. Problema "Cold Start": situación en la que es complejo hacer recomendaciones para nuevos usuarios e ítems debido a la falta de información sobre estos. La ausencia de valoraciones produce una gran dificultad en buscar similaridades entre usuarios o elementos. Como resultado, no se pueden predecir los gustos de los nuevos usuarios ni estos pueden calificar nuevos ítems, lo que conduce a recomendaciones menos precisas.

2. Problema de dispersión: la matriz de interacciones se vuelve muy dispersa cuando gran parte de los usuarios no evalúa una cantidad considerable de los elementos. De este modo, especialmente para el filtrado colaborativo, es complicado generar grupos de usuarios/ítems que presenten características similares debido a la escasez de los datos.

3. Problema de escalabilidad: los Sistemas de Recomendación presentan problemas críticos relativos al manejo de grandes conjuntos de datos. Cuando el número de usuarios y elementos crece, los cálculos se vuelven más pesados y el algoritmo no puede generar resultados satisfactorios para volúmenes de datos grandes. Existen diversos métodos, como técnicas de reducción de la dimensionalidad o clustering, que tratan de reducir esta gran cantidad de información.

4. Falta de diversidad: los Sistemas de Recomendación clásicos tienden a recomendar los mismos tipos de artículos, generalmente por su popularidad, lo que limita la variedad de opciones que se presentan a los usuarios. Esto es, se tienen dificultades para ofrecer recomendaciones fuera de la zona de confort del usuario.

5. Falta de adaptabilidad: a menudo, los Sistemas de Recomendación clásicos utilizan información histórica de los usuarios para hacer recomendaciones, lo que puede llevar a recomendaciones inexactas o irrelevantes para el usuario en el presente. En general, estas herramientas fallan en la adecuación en tiempo real a los cambios en las preferencias, lo que podría reflejar sugerencias obsoletas.

6. Serendipia: se define como la aparición fortuita de un suceso poco probable. En el contexto de Sistemas de Recomendación, se refiere a una especie de sobre-especialización, en la que no se recomiendan ítems que se salgan del perfil del usuario.

	Problemas en Sistemas de Recomendación					
	"Cold Start"	**Dispersión**	**Escalabilidad**	**Diversidad**	**Adaptabilidad**	**Serendipia**
Contenido	M	N	S	N	N	N
FC usuarios	S	S	S	M	S	S
FC ítems	S	S	S	M	S	S
FC latente	M	S	M	M	N	M
CML	N	N	N	N	N	M

N = no afecta, S = Sí afecta, M = afecta en término medio.

Tabla 1: Comparación de Sistemas de Recomendación.

La Tabla 1 desglosa el comportamiento de cada algoritmo frente a los problemas más comunes de recomendación. Cada celda representa si dicho problema afecta en mayor o menor medida a la implementación. En términos generales, el Aprendizaje Colaborativo de Métricas, al aprender métricas de similitud más adaptativas, diversas y eficientes, puede mejorar la calidad y personalización de las recomendaciones en diferentes contextos.

6. Experimentación y resultados

La sección experimental de este trabajo tiene como objetivo evaluar el rendimiento de los Sistemas de Recomendación clásicos y el modelo de Aprendizaje Métrico Colaborativo, CML, en el conjunto de datos *MovieLens*. Más aún, evidenciar que esta herramienta de recomendación avanzada es capaz de realizar recomendaciones más precisas y personalizadas que sistemas clásicos, lo que deriva en métricas de evaluación con mejores resultados. Para terminar, se presentan los resultados de los experimentos y se realiza un análisis comparativo de los diferentes enfoques evaluados. Además, mostraremos la evolución de las diferentes métricas calculadas a través de una visualización adecuada que permita comparar gráficamente los resultados obtenidos. Finalmente, se incluye una representación visual de los usuarios y elementos en el espacio latente, lo cual proporciona una comprensión más profunda de cómo se relacionan y agrupan los objetos en dicho espacio.

6.1. Dataset

El conjunto de datos *MovieLens* es una base de datos pública de calificaciones de películas que ha sido ampliamente utilizada para evaluar Sistemas de Recomendación. Este conjunto de datos contiene información sobre las calificaciones de las películas realizadas por los usuarios, así como los detalles de las películas en sí. Cada usuario está identificado por un código *userId*, al igual que cada película con un atributo *movieId*. Para cada par usuario-elemento se tiene la valoración que este ha proporcionado a la película en una variable denominada *rating*. La películas están descritas por su título y por los géneros a los que pertenece, cuyos valores se almacenan en los atributos *title* y *genres*, respectivamente.

	Características del conjunto de datos					
	Dominio	**Usuarios**	**Items**	**Ratings**	**Concentración**[1]	**Características**
MovieLens	Películas	943	1682	100,000	25,49 %	Título, Género(s)

Concentración[1]: porcentaje de valoraciones que abarca el 5 % de los elementos más valorados.

Tabla 2: Estadísticas del dataset.

Este conjunto de datos contiene valoraciones y etiquetas proporcionadas por 943 usuarios sobre 1682 películas, lo que da lugar a 100,000 pares (usuario, ítem) diferentes. Hay que tener en cuenta que no todos los usuarios califican todos los elementos. La selección de usuarios se realizó de forma aleatoria y únicamente se incluyeron usuarios con más de 20 valoraciones publicadas [8].

6.2. Metodología de evaluación

El conjunto de datos original lo dividimos en dos subconjuntos, los datos de entrenamiento *train*, y el conjunto de prueba *test*. Esta división la hacemos en una proporción 80 % - 20 %, de tal forma que el conjunto *train* contenga todos los usuarios e ítems del dataset original, que no todos los pares de objetos. Este factor es importante para no obtener problemas de arranque en frío, descritos en la Sección 5. Asimismo, el conjunto *train* se utiliza para entrenar los modelos de recomendación, mientras que el conjunto *test* se emplea para evaluar su rendimiento y realizar comparaciones. Es decir, una vez que el modelo está entrenado, se evalúa en el conjunto de *test* y se comparan las predicciones obtenidas con los datos reales de este conjunto.

En lo que a la evaluación de estas predicciones refiere, se implementan las métricas de rendimiento descritas en la Sección 3.3, entre las que encontramos el Error Cuadrático Medio (MSE), la Ganancia Acumulativa con Descuento Normalizado (nDCG) y el Promedio de Precisión Media (MAP). El objetivo primordial es comprobar que el modelo obtenga un buen rendimiento, tanto en el conjunto de entrenamiento, como en el de *test*, lo que supone que está generalizando bien a nuevos datos y no se está sobreajustando. Para terminar, estas métricas de evaluación nos permitirán comparar diferentes modelos de recomendación y determinar cuál es el que mejor se ajusta a los objetivos establecidos. En particular, podremos evaluar la mejora que aporta el modelo CML en comparación con los Sistemas de Recomendación clásicos: basados en contenido y en filtrado colaborativo.

6.3. Resultados y discusión

6.3.1. Resultados de los Sistemas de Recomendación Clásicos

En primer lugar, vamos a estudiar las métricas de rendimiento obtenidas en Sistemas de Recomendación basados en contenido y filtrado colaborativo, es decir, las herramientas clásicas de recomendación.

	Contenido	**FC user-user**	**FC item-item**	**FC latente**
MSE	2,5661	1,0354	0,9671	0,8297
nDCG	0,8867	0,7058	0,9601	0,9105
MAP	0,4777	0,1935	0,1942	0,7111

Tabla 3: Métricas de evaluación para Sistemas de Recomendación clásicos.

Al evaluar los resultados de los Sistemas de Recomendación clásicos, podemos observar patrones distintivos para los diferentes enfoques utilizados. Por norma general, en la práctica suelen funcionar mejor las herramientas de recomendación que se centran en los ítems, ya que los usuarios suelen presentar gustos y preferencias más complejas. Por este motivo, el filtrado colaborativo basado en usuarios es el que proporciona peor rendimiento, ya que al modelo le cuesta reconocer las relaciones entre los usuarios.

Los enfoques basados en contenido y el filtrado colaborativo ítem-ítem presentan comportamientos aceptables. Cabe mencionar que el valor ligeramente superior obtenido para la métrica MSE en el Sistema de Recomendación basado en contenido está probablemente relacionado con el método de conversión construido para obtener valoraciones a partir de la similitud. Asimismo, el enfoque de filtrado colaborativo latente muestra un desempeño competitivo en términos de MSE, nDCG y MAP, lo que indica una buena capacidad para ofrecer recomendaciones precisas. En gran medida, este comportamiento se debe a la capacidad que presenta el modelo para capturar relaciones no lineas. La factorización matricial permite, además de reducir la complejidad computacional y abordar problemas de escalabilidad, identificar características subyacentes en los datos. Estos beneficios potenciales se suman a la habilidad de aprender patrones ocultos e inferir recomendaciones basadas en similitudes latentes, lo que puede ser útil en escenarios en los que no se dispone de suficientes valoraciones específicas de un usuario o elemento.

6.3.2. Resultados del CML

A continuación obtendremos las métricas para el Aprendizaje Métrico Colaborativo. Es importante mencionar que cuando se implementa la red neuronal multicapa, MLP, es necesario seleccionar el número de capas ocultas y las neuronas que las van a conformar. Estos valores constituyen hiper-parámetros del modelo, esto es, valores configurables que afectan a su comportamiento y rendimiento. Con el objetivo de obtener el modelo que optimice las métricas de evaluación seleccionadas, se implementa la técnica de validación cruzada sobre el conjunto *train*, de manera que se seleccione el número de neuronas óptimo para la capa oculta implementada. Definiendo en $k = 5$ el número de divisiones seleccionado para la validación cruzada, el número de neuronas óptimo para la capa oculta es 20. De este modo, la red neuronal está compuesta por una capa oculta de 20 neuronas y la tasa de aprendizaje inicial seleccionada es 0,0001. Como ha sido mencionado, el optimizador AdaGrad adapta este valor en cada iteración en base a los gradientes obtenidos. Por último, el conjunto de entrenamiento se divide en lotes de 256 que evaluarán el rendimiento de la red neuronal durante el proceso de aprendizaje.

Más aún, como ha sido mencionado en apartados anteriores, el objetivo principal del CML es aprender una representación latente de los usuarios y los ítems en un espacio de menor dimensionalidad. Por ello, esta red neuronal utiliza *embeddings* de tamaño 100, lo que permite trasladar las características iniciales a un espacio conjunto de menor dimensión, en el que la similitud entre los vectores de usuarios y los vectores de ítems refleja las preferencias de los usuarios hacia los ítems. Antes de realizar una comparativa conjunta de los algoritmos, estudiaremos la evolución de la solución en función de los hiper-parámetros seleccionados para el entrenamiento de esta red neuronal.

	Número de iteraciones					
	0	**10**	**20**	**30**	**40**	**50**
MSE	0,340428	0,417361	0,436702	0,416621	0,377657	0,345914
nDCG	0,891388	0,912446	0,918820	0,922329	0,918962	0,915041
MAP	0,59963	0,376861	0,397230	0,574769	0,736495	0,845873

Tabla 4: Métricas de evaluación para el Aprendizaje Métrico Colaborativo.

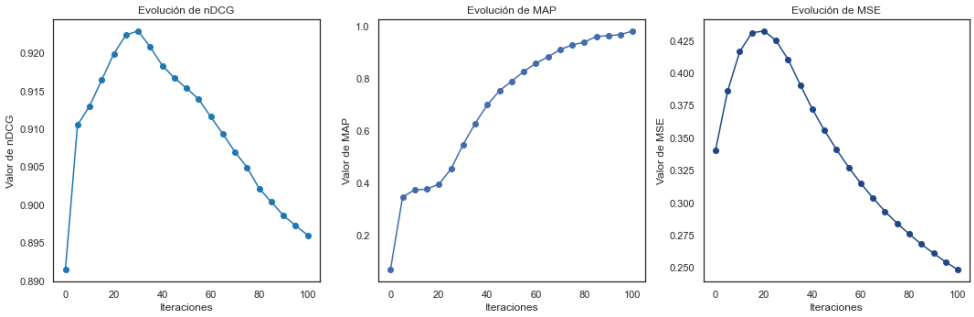


Figura 8: Evolución de las métricas para el Aprendizaje Métrico Colaborativo en 100 iteraciones.

En la Figura 8 se analiza la evolución de las métricas para el Sistema de Recomendación CML en 100 iteraciones. Se observa que, tras varias iteraciones, el valor de la métrica nDCG comienza a disminuir, al contrario de lo que ocurre en MAP. En lo que al MSE refiere, podemos ver un aumento significativo de sus valores, seguido de una importante disminución. La Tabla 4 recoge los resultados hasta la iteración 50. A partir de este momento parece que el modelo está sobreaprendiendo y probablemente no generalice bien a nuevos datos de entrada, lo que justifica la decisión de detener el entrenamiento en ese punto. En general, estos resultados muestran que el Sistema de Recomendación tiene un rendimiento eficiente en términos de precisión y relevancia de las recomendaciones.

Cabe recordar que no hay una relación predefinida entre el comportamiento de las métricas propias de los Sistemas de Recomendación (nDCG y MAP). Es posible que un sistema tenga un alto nDCG pero un MAP relativamente bajo, o viceversa. Esto puede ocurrir cuando el sistema logra un buen ordenamiento de recomendaciones relevantes, pero no necesariamente las presenta en las primeras posiciones del ranking.

6.3.3. Desempeño y comparativa global

Finalmente, esta sección está enfocada a estudiar y comparar el desempeño de los algoritmos de recomendación estudiados. La tabla siguiente muestra los resultados obtenidos para cada una de las implementaciones descritas.

	Contenido	**FC user-user**	**FC item-item**	**FC latente**	**CML**
MSE	2,5661	1,0354	0,9671	0,8297	**0,3459**
nDCG	0,8867	0,7058	0,9601	0,9105	**0,9150**
MAP	0,4777	0,1935	0,1942	0,7111	**0,8459**

Tabla 5: Comparación de las métricas obtenidas para cada implementación.

En suma, los Sistemas de Recomendación basados en factores latentes y el modelo CML parecen ofrecer un rendimiento más sólido en términos de precisión de las valoraciones y ordenamiento de las recomendaciones en comparación con los sistemas basados en contenido y los sistemas de filtrado colaborativo básicos. Estas implementaciones aprovechan modelos avanzados y técnicas de aprendizaje para capturar relaciones complejas en los datos y proporcionar recomendaciones más precisas y relevantes.

Especialmente destaca el rendimiento sobresaliente del algoritmo CML, superior al resto en todas las evaluaciones. No obstante, el filtrado colaborativo latente ofrece también métricas muy competitivas. Por este motivo, habría que analizar si el coste y la complejidad computacional que supone implementar el Aprendizaje Métrico Colaborativo compensa la mejora en precisión y personalización de las recomendaciones.

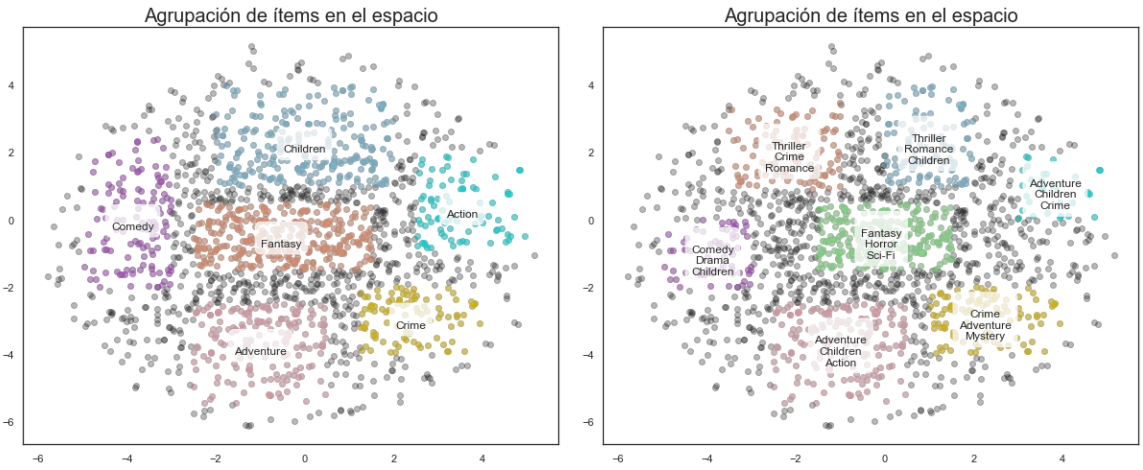
6.4. Visualización de los embeddings

Para terminar con la parte experimental, es interesante mostrar la agrupación de las películas en el espacio compartido usuario-ítem. Como ha sido mencionado en la Sección

4.3, los usuarios y elementos se mapean a un espacio conjunto utilizando una representación en forma de embedding. Para este problema concreto, el foco de atención lo atraen las películas, ya que almacenan información adicional sobre los géneros a los que pertenecen. De esta manera, la representación de los ítems aprendida por el modelo puede capturar similitudes y diferencias entre las películas en función de sus características.

El objetivo principal de visualizar estos embeddings es comprender las relaciones entre películas y encontrar agrupaciones o clústeres que puedan tener significado en término de los géneros en los que se catalogan. Primero que nada, es importante mencionar que la dimensión de los embeddings obtenidos a través de nuestro modelo es de 100 elementos. Antes de poder visualizar estos vectores, es necesario aplicar un proceso de reducción de la dimensionalidad para quedarnos con 2 únicas componentes, las necesarias para mantener la máxima información posible en la visualización. Para ello, aplicamos el algoritmo de reducción de la dimensionalidad t-SNE: *t-Distributed Stochastic Neighbor Embedding*, que permite visualizar los embeddings reducidos por grupos manteniendo la máxima información posible.

Tras reducir la dimensión de los embeddings a un espacio bidimensional, se obtiene la agrupación de los ítems en el espacio en función de los géneros a los que pertenecen.



(a) Clustering en 1 género.

(b) Clustering en 3 géneros.

Figura 9: Agrupación de ítems en el espacio en función de los géneros a los que pertenecen.

El algoritmo t-SNE es apropiado en este caso por su capacidad para mantener la esctructura de vecindad en los datos, es decir, trata de conservar las relaciones de proximidad local. Esto es especialmente importante para que observaciones próximas en el espacio original, es decir, similares en cuanto a géneros, lo sean también en el espacio reducido. De esta manera, se mantienen las agrupaciones y estructuras en los datos permitiendo obtener resultados concluyentes en este espacio bidimensional.

Los gráficos de dispersión de la Figura 9 muestran la concentración de los ítems en el espacio en función de los géneros en los que se catalogan. Al observar los clusters en los gráficos, es posible identificar conjuntos de películas que comparten características similares en términos de género. Por ejemplo, en la Figura 9a los ítems se estructuran según el género más dominante. En la segunda imagen, Figura 9b, podemos observar agrupaciones relacionadas con los tres géneros que más representan a cada grupo de elementos. En general, se observa que los géneros más prevalentes son aquellos más frecuentes en la clasificación de películas.

Por último, mencionar que si en el gráfico de dispersión se observan agrupaciones cercanas de películas, pongamos de comedia, significa que el modelo ha aprendido a capturar las similitudes entre este tipo de películas en términos de sus embeddings. Igualmente, agrupaciones separadas indican que el modelo es capaz de distinguir las características específicas de cada género, como en el caso de películas de acción o de crímenes.

7. Conclusiones

El presente trabajo muestra de manera concluyente las ventajas significativas que se obtienen al implementar una herramienta de recomendación avanzada: el Aprendizaje Colaborativo de Métricas o CML.

La implementación del CML en el conjunto de datos *MovieLens* ha demostrado un impacto positivo indudable en la personalización y adecuación de las recomendaciones. Los resultados obtenidos sustentan la afirmación de que el CML es una estrategia de recomendación superior a los sistemas clásicos, obteniendo resultados motivadores en las métricas evaluadas. Asimismo, al permitir la interpretación de los embeddings en un espacio compartido, se ofrece una mayor comprensión de los factores subyacentes que influyen en las recomendaciones realizadas. Finalmente, la utilización de las características propias de los ítems permite una mejor captura de la información relevante de los elementos. Esta característica particular del CML es especialmente valiosa en la interpretación de los resultados alcanzados, y por ende, en la toma de decisiones.

Por último, es importante destacar que la calidad del Sistema de Recomendación está estrechamente ligada a la calidad del conjunto de datos utilizado para su evaluación; un dataset consistente brindará resultados más eficientes en los algoritmos evaluados. Si bien no existe una técnica de recomendación universalmente reconocida como la mejor, cada una posee fortalezas y debilidades que pueden variar según el contexto de uso. En la actualidad, técnicas de recomendación avanzadas ofrecen recomendaciones más personalizadas y precisas, pero son también computacionalmente más costosas de implementar. Por ello, es fundamental considerar aspectos como la escalabilidad, adaptabilidad y eficiencia al seleccionar un Sistema de Recomendación para garantizar su rendimiento óptimo.

8. Líneas futuras

Perspectivas de trabajo futuras sobre el presente escrito incluyen la ampliación del conjunto de características consideradas para usuarios e ítems. En el caso de las películas, se sugiere la incorporación de información adicional relacionada con los actores, directores, sinopsis u otros atributos relevantes. Más aún, de los usuarios se podrían estudiar características demográficas, económicas o sociales. La inclusión de esta información complementaria en el contexto del Aprendizaje Métrico Colaborativo permitiría capturar aspectos más detallados y completos, además de facilitar una mejor comprensión de las bases en las que se sustentan las recomendaciones.

De forma adicional, otra línea de estudio prometedora para la mejora de los Sistemas de Recomendación se encuentra en la aplicación de técnicas de Deep Learning y el uso de redes neuronales profundas, como las redes neuronales convolucionales (CNN) o recurrentes (RNN) [28]. Estos enfoques avanzados han demostrado su sobresaliente desempeño en numerosas tareas de procesamiento de la información, incluida la recomendación. Las CNN, por ejemplo, son adecuadas en el procesamiento de datos con estructura espacial, como imágenes o secuencias de palabras. En el contexto de nuestra experimentación, una CNN podría identificar patrones visuales en carteles de películas o extraer características textuales de la sinopsis. En lo que a las RNN refiere, estas estructuras son especialmente efectivas para el procesamiento de datos secuenciales, como las interacciones entre usuarios y elementos a lo largo del tiempo. En el marco de la recomendación, esta característica permite modelar la evolución de las preferencias y el comportamiento del usuario a medida que este interactúa con diferentes elementos, lo que facilita la generación de recomendaciones adaptadas a los cambios en los gustos y preferencias.

En este sentido, futuras investigaciones podrían centrarse en abordar estos desafíos y explorar nuevas formas de combinar las fortalezas del Aprendizaje Colaborativo de Métricas con las ventajas de las técnicas del Deep Learning. Estudios recientes presentan el algoritmo *Neural Collaborative Filtering* [9], que ha demostrado ser exitoso en la generación de recomendaciones precisas y personalizadas, fundamentalmente en situaciones en las que el contexto y las características específicas de los usuarios y/o elementos son relevantes.

Bibliografía

[1] Mohammed Alshammari. *An explainable recommender system based on semantically-aware matrix factorization.* PhD thesis, 08 2019.

[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[3] Lorena Chavarría Báez, Rosaura Palma Orozco, and Elena F. Ruiz Ledesma. Los sistemas de recomendación en la toma de decisiones. pages 7–11, 2013. null ; Conference date: 09-07-2013 Through 12-07-2013.

[4] R. Chen, Qingyi Hua, Yan shuo Chang, Bo Wang, Lei Zhang, and Xiangjie Kong. A survey of collaborative filtering-based recommender systems: From traditional methods to hybrid methods based on social networks. *IEEE Access*, 6:64301–64320, 2018.

[5] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, page 39–46, New York, NY, USA, 2010. Association for Computing Machinery.

[6] Félix Hernández del Olmo and Elena Gaudioso. Evaluation of recommender systems: A new approach. *Expert Syst. Appl.*, 35:790–804, 2008.

[7] Oscar Escamilla González and Sergio Marcellin Jacques. Estado del arte en los sistemas de recomendación. *Res. Comput. Sci.*, 135:25–40, 2017.

[8] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

[9] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.

[10] Matthew Helm, Andrew Swiergosz, Heather Haeberle, Jaret Karnuta, Jonathan Schaffer, Viktor Krebs, Andrew Spitzer, and Prem Ramkumar. Machine learning and artificial intelligence: Definitions, applications, and future directions. *Current Reviews in Musculoskeletal Medicine*, 13, 02 2020.

[11] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge J. Belongie, and Deborah Estrin. Collaborative metric learning. *Proceedings of the 26th International Conference on World Wide Web*, 2017.

[12] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, oct 2002.

[13] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 426–434, 2008.

[14] Manoj Kumar, D.K. Yadav, Ankur Singh, and Vijay Kr. Gupta. Article: A movie recommender system: Movrec. *International Journal of Computer Applications*, 124(3):7–11, August 2015. Published by Foundation of Computer Science (FCS), NY, USA.

[15] Pushpendra Kumar and Ramjeevan Singh Thakur. Recommendation system techniques and related issues: a survey. *International Journal of Information Technology*, 10:495–501, 2018.

[16] Urszula Kuzelewska. Clustering algorithms in hybrid recommender system on movielens data. *Studies in Logic, Grammar and Rhetoric*, 37:125 – 139, 2014.

[17] Tie-Yan Liu. Learning to rank for information retrieval. In *Foundations and Trends® in Information Retrieval*, volume 3, page 12. Now Publishers Inc., 2009.

[18] Agnes Lydia and Sagayaraj Francis. Adagrad—an optimizer for stochastic gradient descent. *Int. J. Inf. Comput. Sci*, 6(5):566–568, 2019.

[19] Linyuan Lü, Matúš Medo, Chi Ho Yeung, Yi-Cheng Zhang, Zi-Ke Zhang, and Tao Zhou. Recommender systems. *Physics Reports*, 519(1):1–49, oct 2012.

[20] Chih-Chao Ma. A guide to singular value decomposition for collaborative filtering. 2008.

[21] Luong Vuong Nguyen, Min-Sung Hong, Jason J. Jung, and Bong-Soo Sohn. Cognitive similarity-based collaborative filtering recommendation system. *Applied Sciences*, 10(12):4183, Jun 2020.

[22] Carlos Ordoñez and Marcelo Romero. Sistemas de recomendación basados en descomposición de valores singulares. *Revista de la Facultad de Ingeniería*, 29(3):41–56, 2014.

[23] Ll Pérez-Planells, Jesús Delegido, Juan Pablo Rivera-Caicedo, and Jochem Verrelst. Análisis de métodos de validación cruzada para la obtención robusta de parámetros biofísicos. *Revista de teledetección*, (44):55–65, 2015.

[24] Zeeshan Rasheed and M Pushpalatha. An overview of data mining techniques in recommender systems. *International Journal of Computer Science and Mobile Computing*, 4:609–615, 2015.

[25] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Recommender Systems Handbook*, volume 1-35, pages 2–3. 10 2010.

[26] Alex Witsil and Jeffrey Johnson. Volcano video data characterized and classified using computer vision and machine learning algorithms. *Geoscience Frontiers*, 11, 02 2020.

[27] Qian Zhang, Jie Lu, and Yaochu Jin. Artificial intelligence in recommender systems. *Complex and Intelligent Systems*, 7, 11 2020.

[28] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. *Deep Learning Based Recommender System: A Survey and New Perspectives*, volume 52. Association for Computing Machinery, New York, USA, feb 2019.

[29] Peilin Zhao and T. Zhang. Accelerating minibatch stochastic gradient descent using stratified sampling. *ArXiv*, abs/1405.3080, 2014.

[30] Xiao Zhou, Danyang Liu, Jianxun Lian, and Xing Xie. Collaborative metric learning with memory network for multi-relational recommender systems. IJCAI'19, page 4454–4460. AAAI Press, 2019.