INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO

Tesis de Mario

TESIS

QUE PARA OBTENER EL TÍTULO DE

Licenciado en Matemáticas Aplicadas

PRESENTA

MARIO HUMBERTO BECERRA CONTRERAS

ASESOR

Fernando Esponda Darlington

CIUDAD DE MÉXICO 2017

"Con fundamento en los artículos 21 y 27 de la Ley Federal de Derecho de Autor y como titular de los derechos moral y patrimonial de la obra titulada "**Tesis de Mario**", otorgo de manera gratuita y permanente al Instituto Tecnológico Autónomo de México y a la biblioteca Raúl Baillères Jr., autorización para que fijen la obra en cualquier medio, incluido el electrónico, y la divulguen entre sus usuarios, profesores, estudiantes o terceras personas, sin que pueda percibir por tal divulgación una prestación"

0

Fecha

Firma

Resumen

Este documento presenta una plantilla para usar en las tesis y tesinas del ITAM. Se provee de manera gratuita y sin ninguna responsabilidad bajo la licencia *creative commons BY-SA 3.0*.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

A Long Tail is just culture unfiltered by economic scarcity.

– Chris Anderson

1.1. Motivación

Con la cantidad de información que se genera hoy en día y la cantidad de servicios *online* que tenemos, para explotar el fenómeno conocido como *long tail* en *marketing* los proveedores de estos servicios quieren personalizar el contenido que ofrecen a sus usuarios; en particular quieren predecir la respuesta del usuario a distintas opciones. A este tipo de sistemas se les conoce como **sistemas de recomendación**, y tal vez el más conocido es el de *Netflix*; sin embargo, existen más, como la sección de 'Qué ver' en *YouTube* o los destinos que nos recomiendan en *Airbnb* o lo productos que nos recomiendan en *Amazon.*

Los sistemas de recomendación se han vuelto más populares últimamente por varias razones, pero todo se podría resumir en un solo concepto: *long tail* o cola larga. Este concepto rompe la creencia del principio de Pareto que afirma que el 80 % de las consecuencias (ganancias) provienen del 20 % de las causas (productos). Este principio se ha usado como regla de dedo en distintas disciplinas, pero en ciertos productos no se cumple, por ejemplo, las rentas de películas en

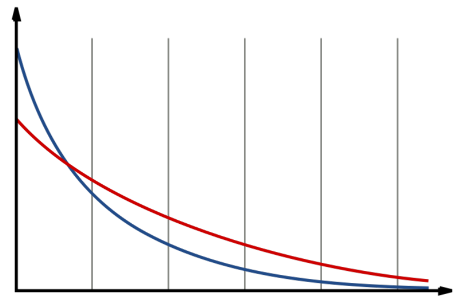

Figura 1.1: Una distribución de cola larga (roja) con una de cola no larga (azul).

Netflix, las cuales tienen una cola larga. En una distribución de cola larga, es menos del 80 % de las ganancias el que proviene del 20 % de los productos, esto significa que se pueden tener ganancias a partir de productos no tan populares. Se puede ver una imagen de una distribución de cola larga en la figura 1.1.

Con el concepto de cola larga en mente, podemos imagina por qué un sistema de recomendación puede servir para ciertos tipos de productos como *Netflix* o *Amazon*. Estas empresas tienen miles y miles de distintos productos que ofrecer, y muchos de ellos no son populares o conocidos por la mayoría de los usuarios, entonces, mediante un sistema de recomendación, se les pueden ofrecer productos no conocidos por ellos pero que pueden ser de su agrado. Esto le funcionó bastante bien a *Netflix*, pues en un principio, el 30 % de sus rentas provenía de estrenos, comparado con el 70 % de *Blockbuster*; y esto era en parte gracias al sistema de recomendación, y a que *Netflix*, a diferencia de *Blockbuster*, no tenía el espacio limitado para guardar películas físicamente.

Un ejemplo extremo de la aplicación de los sistemas de recomendación y de la cola larga es el libro *Touching the Void*. Este libro no fue muy popular cuando acababa de salir, pero años después un libro parecido llamado *Into Thin Air* fue publicado. Ambos eran vendidos por *Amazon*, y su sistema de recomendación encontró algunas personas que compraron ambos libros y empezó a recomendar *Touching the Void* a usuarios que habían comprado o considerado comprar *Into Thin Air*, y eventualmente *Touching the Void* se volvió popular.

1.2. Objetivo

1.3. Alcance

Capítulo 2

Marco teórico

2.1. Aprendizaje estadístico

El aprendizaje estadístico se refiere a un conjunto de herramientas para modelar y entender conjuntos de datos complejos. Es un área relativamente moderna, y tiene componentes de computación, en particular del aprendizaje de máquina. Aunque estas disciplinas van de la mano, no son totalmente equivalentes, teniendo ciertas diferencias filosóficas en cuanto a qué se hace en cada una.

Con la enorme cantidad de datos a los que se tiene acceso ahora, el aprendizaje estadístico se ha vuelto un campo muy demandado y fructífero en muchas áreas. El progreso que se ha logrado en los últimos años se ha debido en gran parte al desarrollo del poder de cómputo, sin el cual muchas de las técnicas modernas sería imposible tener un resultado.

Usualmente se divide al aprendizaje estadístico en dos vertientes: aprendizaje supervisado y aprendizaje no supervisado. El primero involucra construir un modelo estadístico para predecir o estimar una variable llamada *de respuesta* basado en una o más variables llamadas *de entrada*; mientras que en el aprendizaje no supervisado se tienen variables de entrada pero no de salida, se desea aprender relaciones y estructura de los datos.

Un ejemplo de aprendizaje supervisado es estimar el ingreso de un hogar a partir de variables acerca de la zona en donde se habita, tipo de casa en la que se vive, número de carros, etc. Y un ejemplo de aprendizaje no supervisado sería encontrar grupos de los hogares que surjan naturalmente a partir de los datos, de tal forma que dentro de cada grupo exista una gran similitud (definida *a priori*) y entre los grupos haya diferencias. Gran parte del problema aquí radica en definir la medida de similitud.

2.1.1. Aprendizaje supervisado

Como se mencionó, en el aprendizaje supervisado, se tiene una variable respuesta, la cual será denotada como el vector Y de dimensión n, y cada variable de entrada se denomina X_i, con i desde 0 hasta p. Cada X_i, al igual que Y, es un vector de dimensión n, por lo que el conjunto de datos $\{X_1, ..., X_p\}$ se puede escribir como una matriz $X = [X_1, ..., X_p]$ de dimensiones $n \times p$. Se asume que existe una relación entre Y y X que se puede escribir de forma general como

$$Y = f(X) + \varepsilon.$$

Aquí, f es una función de X fija pero desconocida, la cual tiene información sistemático de X acerca de Y, y ε es un término de error aleatorio. La esencia del aprendizaje supervisado es poder encontrar o aprender f a partir del *conjunto de entrenamiento* denominado $\mathcal{L}$, tal que $\mathcal{L} = \{(x_1, y_1), ..., (x_n, y_n)\}$. En este caso, cada y_i es un escalar y cada x_i es un vector de dimensión p. Con $\mathcal{L}$ se construye un predictor $\hat{f}$, que es una estimación de f, de tal forma que se tiene una estimación $\hat{Y}$ de Y aplicando X a $\hat{f}$, es decir

$$\hat{Y} = \hat{f}(X).$$

La estimación $\hat{f}$ se hace seleccionando f de una familia de funciones $\mathcal{F}$ de tal forma que $y_i \approx \hat{f}(x_i)$ para cada $(x_i, y_i) \in \mathcal{L}$. Por ejemplo, en regresión lineal se resuelve el problema de mínimos cuadrados

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2,$$

donde $\mathcal{F}$ es la familia de funciones de la forma $f(X) = X\beta$ con $\beta \in \mathbb{R}^p$. Así, $\hat{f}(X) = X\hat{\beta}$, donde $\hat{\beta}$ resuelve el problema de mínimos cuadrados. En este caso, se quiere encontrar el vector de parámetros β tal que la suma de residuales al cuadrado sea mínima.

Si la familia de funciones $\mathcal{F}$ es muy inflexible (como regresión lineal), entonces las predicciones tenderán a ser malas pues no se capta bien la relación entre X y Y, sin embargo, esto puede ser deseable en algunos casos, como cuando se pretende interpretar la relación entre X y Y.

Es prudente introducir el concepto de *función de pérdida* o de error. Sea $\hat{Y} = \hat{f}(X)$, la predicción de Y; en el caso de regresión lineal, $\hat{Y} = X\hat{\beta}$. En este mismo caso la función de pérdida fue la de pérdida cuadrática, definida como $L_{\mathcal{L}}(\hat{Y}, Y) = (\hat{f}(X) - Y)^2$, o sea, la diferencia entre la estimación de Y y la verdadera Y al cuadrado, con los datos del conjunto $\mathcal{L}$. Existen distintos tipos de funciones de pérdida, pero muchas veces se utiliza la pérdida cuadrática debido a las propiedades diferenciables que tiene.

En general, cuando uno entrena un modelo para predecir, no se desea minimizar el error de entrenamiento $L_{\mathcal{L}}$, sino el error de predicción, esto es, el error de cualquier observación futura (X^0, Y^0). Esto quiere decir que no se quiere minimizar $L_{\mathcal{L}}(f, \hat{f})$, sino el error esperado de predicción, definido como $\mathbb{E}\left[L(\hat{f}(X^0), Y^0)\right]$.

Debido a que se quiere minimizar el error esperado de predicción, es común separar el conjunto de datos (X, Y) en dos, el conjunto de entrenamiento, presentado anteriormente como $\mathcal{L}$, y un *conjunto de validación* o *conjunto de prueba* denotado como $\mathcal{T}$. Para hacer esto, del conjunto original de datos, se toma una muestra aleatoria para entrenar (este es el conjunto de entrenamiento $\mathcal{L}$) y el resto es el conjunto de validación $\mathcal{T}$, con el cual se prueba el poder predictivo del modelo. A $L_{\mathcal{T}}$ se le conoce como el error de validación o error de prueba, y es una estimación del error de predicción. Si $\mathcal{T}$ tiene m elementos, y m es grande, entonces el error de prueba se aproxima al error esperado de predicción.

Notar que el error de entrenamiento no aproxima el error de predicción porque el error de entrenamiento depende de $\mathcal{L}$. De hecho la predicción del error de predicción utilizando el error de entrenamiento está sesgada hacia abajo, especialmente para modelos complejos.

I. Regresión y clasificación

El aprendizaje supervisado, a su vez, puede ser dividido en dos tipos de problemas dependiendo del tipo de variable respuesta. Cuando la variable respuesta es categórica, se dice que es un problema de clasificación, en otro caso, se dice que es un problema de regresión. Un ejemplo de problema de regresión es modelar el precio de una casa, mientras que un ejemplo de problema de clasificación es modelar el género de una persona.

2.1.2. Aprendizaje no supervisado

El aprendizaje no supervisado describe un problema más retador en cuanto a que para cada $i = 1, .., n$, se tiene un vector de observaciones x_i para el cual no se tiene una respuesta asociada y_i. Es por esto que se le llama *no supervisado*, pues no hay una variable respuesta en el análisis. Uno de los principales análisis que se hace en el aprendizaje no supervisado es *análisis de conglomerados*, en el cual a cada una de las observaciones $x_1, ..., x_n$ se clasifica en un grupo. Existen muchos métodos de análisis de conglomerados, sin embargo, no es el objetivo de este trabajo estudiarlas.

2.1.3. Regularización

2.1.4. Evaluación de desempeño

2.2. Optimización de funciones de pérdida

2.2.1. Descenso en gradiente

2.2.2. Descenso en gradiente estocástico

2.3. Sistemas de recomendación

El objetivo de los sistemas de recomendación es predecir las respuestas de los usuarios a distintas opciones. Dos clasificaciones muy amplias de los sistemas de recomendación son: basados en contenido y filtrado colaborativo. Existen dos grandes tipos de sistemas de recomendación de acuerdo a las técnicas que usan.

- Basados en contenido: A partir de características y propiedades de los productos (por ejemplo, género, actores, país de origen, año, etc.) intentamos predecir el gusto por el producto construyendo variables derivadas del contenido de los artículos (como qué actores salen, año, etc.).
- Colaborativos: A partir de usuarios y productos se construyen medidas de similitud en el sentido de que les han gustado los mismos productos o que les gustaron a las mismas personas. Así, los productos recomendados a un usuario son los que le gustaron a otros usuarios.

En este trabajo se supone que se tiene una matriz de calificaciones R, tal que las filas representan usuarios que calificaron a los productos representados en las columnas de la siguiente forma:

	P_1	P_2	P_3	$\cdots$	P_n
U_1	1	2	3	$\cdots$	-
U_2	4	-	4	$\cdots$	-
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
U_m	-	-	1	$\cdots$	3

donde cada P_i es el producto i-ésimo que se ofrece y U_i es el usuario i-ésimo que ha calificado el producto correspondiente. Los espacios donde hay guiones representan datos faltantes. Es usual que este tipo de matrices tengan la mayor parte de las entradas faltantes, pues no todos los usuarios han visto todas las películas, o calificado todos los productos. El objetivo del sistema de recomendación es llenar los espacios faltantes mediante una predicción y recomendar los productos que tienen una predicción alta. Notar que aquí se considera que ya se tiene la matriz de calificaciones, lo cual a veces, en la práctica, puede ser un trabajo difícil y haya que pedir a usuarios que califiquen productos.

2.3.1. Basados en contenido

Para este tipo de recomendaciones se construye un **perfil** para cada producto. Como se había mencionado, este perfil puede contener el género, los actores, país de origen, etc. Esto puede ser más sencillo en el caso de películas pues existen fuentes de información sobre las películas como *Internet Movie Database (IMDb)*; pero con documentos o imágenes, por ejemplo, no siempre se tiene esta información, así que es necesario la construcción de variables a través de *tags*, palabras clave y, en el caso de texto, tal vez un modelo de lenguaje.

Una vez construidos los perfiles, el proceso que queda es de alguna forma emparejar los gustos del usuario obtenidos de la matriz de utilidad con los perfiles de los productos.

Un ejemplo muy sencillo es el siguiente. Imaginemos dos películas con 5 actores cada una y que dos actores están en las dos películas, y que además las calificaciones promedio de cada película son 3 y 4. Entonces los perfiles de dos películas se ven como:

	Actor 1	Actor 2	Actor 3	Actor 4	Actor 5	Actor 6	Actor 7	Actor 8	Calif
P_1	0	1	1	0	1	1	0	1	3
P_2	1	1	0	1	0	1	1	0	4

Con esta información se puede obtener una medida de similitud entre las películas. Una medida muy utilizada es la distancia coseno, definida como

$$sim(P_1, P_2) = \frac{P_1 \cdot P_2}{||P_1|| \times ||P_2||} \tag{2.1}$$

Dos películas son más similares si tienen una distancia coseno menor.

Supongamos ahora que los usuarios U_k y U_j solo han calificado dos película. Si el usuario U_k calificó con 2 y 4 las películas P_1 y P_2 respectivamente y si el usuario U_j calificó con 4 y 4 las películas P_1 y P_2 respectivamente, entonces se puede ver esto en forma matricial.

	P_1	P_2
U_k	2	4
U_j	4	4

Tiene sentido centrar las calificaciones por usuario (restarle la media) para reducir un poco la heterogeneidad de la escala, entonces tenemos

	P_1	P_2
U_k	-1	1
U_j	0	0

Así, para el usuario U_k tenemos que los perfiles para cada película se ven como

	Actor 1	Actor 2	Actor 3	Actor 4	Actor 5	Actor 6	Actor 7	Actor 8	Calif
P_1	0	-1	-1	0	-1	-1	0	-1	2
P_2	1	1	0	1	0	1	1	0	4

Y para el usuario U_j tenemos que los perfiles para cada película se ven como

	Actor 1	Actor 2	Actor 3	Actor 4	Actor 5	Actor 6	Actor 7	Actor 8	Calif
P_1	0	4	4	0	4	4	0	4	4
P_2	4	4	0	4	0	4	4	0	4

Entonces, para el perfil general de cada usuario, se promedian por película los perfiles de cada uno y obtenemos

	Actor 1	Actor 2	Actor 3	Actor 4	Actor 5	Actor 6	Actor 7	Actor 8	Calif
U_k	0.5	0	-0.5	0.5	-0.5	0	0.5	-0.5	3
U_j	2	4	2	2	2	4	2	2	4

Con esto podemos estimar qué tanto le va a gustar a un usuario una película calculando la distancia coseno entre el perfil del usuario y el perfil de la película. Esta forma es muy rudimentaria y sencilla, pero puede servir como base para otros modelos.

2.3.2. Colaborativos

I. Modelo base

En esta sección se construye un modelo base muy sencillo para hacer predicciones, y el cual sirve como comparación con otros modelos más complejos. Para este modelo se supone que los artículos y los usuarios tienen sesgos, o sea, hay usuarios que tienden a calificar más alto, o a ser más estrictos y calificar más bajo; al mismo tiempo, puede haber artículos que tiendan a recibir calificaciones más altas; y que gran parte de la calificación observada se debe a efectos asociados a estos sesgos.

Sea μ la media general de todos los artículos y todos los usuarios, entonces la predicción del modelo base es

$$\hat{r_{ij}} = \mu + a_i + b_j$$

donde a_i es la desviación del usuario i respecto a la media general y b_j es la desviación del artículo j respecto a la media general.

Una forma de estimar los sesgos es calculando

$$a_i = \frac{1}{M_i} \sum_t r_{it} - \mu,$$

y

$$b_j = \frac{1}{N_j} \sum_s r_{sj} - \mu,$$

donde donde M_i es el número de artículos calificados por el usuario i y N_j es el número de calificaciones que tiene el artículo j.

Otra forma un poco más precisa es resolver el problema de mínimos cuadrados

$$\min_b \sum_{(i,j)\in A} (r_{ij} - \mu - a_i - b_j)^2 + \lambda \left(\sum_i a_i^2 + \sum_j b_j^2 \right)$$

donde r_{ij} es la calificación del usuario i del artículo j, A es el conjunto de usuarios y artículos para los cuales se conoce la calificación r_ij, $|A|$ es la cardinalidad de A (o sea, el total de artículos calificados por todos los usuarios) y λ es un término de regularización que evita el sobreajuste.

A este sencillo modelo se le pueden agregar otros términos de regularización que ayuden a disminuir el ruido causado por artículos y usuarios con pocas calificaciones, al encoger las calificaciones a la media. Una vez que se calcularon a_i y b_j usando cualquiera de los métodos mencionados, se hace la predicción

$$r_{ij} = \mu + \frac{M_i}{\gamma + M_i} a_i + \frac{N_j}{\gamma + N_j} b_j \tag{2.2}$$

donde M_i es el número de artículos que ha calificado el usuario i, N_j es el número de calificaciones que tiene el artículo j, y γ es el parámetro de regularización.

II. Métodos de vecindario

Este tipo de modelos se centran en la similitud de los usuarios y los artículos. Los usuarios son similares si los vectores que los representan (es decir, las filas en la matriz de calificaciones) están cercanos de acuerdo a alguna similitud definida. Una recomendación para un usuario i se hace al ver a los usuarios más parecidos a i y recomendando artículos que hayan sido del agrado de estos usuarios parecidos.

El primer problema con estos métodos surge al definir una medida de similitud entre artículos o usuarios. Dos medidas muy utilizadas son la similitud de Jaccard y la similitud coseno. En el caso de una matriz de calificaciones binaria, la similitud Jaccard tendría más sentido, pero si son calificaciones explícitas conviene más utilizar similitud coseno. Una desventaja de utilizar la similitud coseno es que los elementos faltantes son tratados como 0, lo cual tiene el efecto de tratar la falta de calificación como disgusto hacia el artículo. Una forma de solucionar esto es centrando las calificaciones por usuario al restar la media, de esta forma el faltante al ser tratado como cero tiene el efecto de que al usuario ni le gusta ni le disgusta el artículo.

Una forma de hacer predicciones de calificaciones para un usuario i y un artículo j es encontrar los un vecindario de los n usuarios más parecidos a i y tomar el promedio de las calificaciones del artículo j. O de forma dual, se podría encontrar un vecindario con los n artículos más parecidos a j y tomar el promedio de las calificaciones que el usuario i ha dado a esos artículos. Aquí se hace una vertiente en los métodos de vecindario, dependiendo de si se centran en las similitudes de los usuarios o de los artículos.

Basados en usuarios

Este tipo de algoritmos pretenden imitar las recomendaciones de boca en boca al asumir que los usuarios con preferencias similares calificarán los artículos de forma similar. De esta manera, para hacer una predicción para el usuario i y el artículo j habría que encontrar un vecindario de i, denotado como $N(i)$, y tomar el promedio de las calificaciones en el vecindario:

$$\hat{r_{ij}} = \frac{\sum_{a \in N(i)} r_{aj}}{|N(i)|}. \tag{2.3}$$

Por ejemplo,

PONER EJEMPLO DE CUADERNO!!!.

Este ejemplo está ignorando el hecho de que algunos usuarios en el vecindario son más similares a otros, por lo que es una buena idea tener un ponderador de similitud. Así, en lugar de 2.3, la predicción es

$$\hat{r_{ij}} = \frac{\sum_{a \in N(i)} s_{ia} r_{aj}}{\sum_{a \in N(i)} s_{ia}}, \tag{2.4}$$

donde s_{ia} es la similitud entre el usuario i y el usuario a perteneciente al vecindario de i.

Basados en artículos

Este tipo de algoritmos asumen que los usuarios prefieren artículos que son similares a otros artículos que les gustaron. Muchas veces este tipo de sistemas producen información más confiable porque es más sencillo encontrar artículos del mismo género que encontrar usuarios a los que solo les gustan artículos de un solo género. En este caso, para hacer una predicción para el usuario i y el artículo j habría que encontrar un vecindario de j, denotado como $N(j)$, y tomar el promedio (ponderado por nivel de similitud para mejor estimación) de las calificaciones que el usuario ha hecho a los artículos en $N(j)$, es decir

$$\hat{r_{ij}} = \frac{\sum_{a \in N(j)} s_{ja} r_{ia}}{\sum_{a \in N(j)} s_{ja}}, \tag{2.5}$$

donde s_{ja} es la similitud entre el artículo j y el artículo a perteneciente al vecindario de j.

Por ejemplo,

PONER EJEMPLO DE CUADERNO!!!.

III. Métodos de factorización de matrices

Modelo básico de factorización de matrices

La idea de la factorización de matrices proviene de los modelos de factores latentes, en los cuales se supone que hay factores ocultos o latentes que caracterizan a los usuarios y a los productos. Por ejemplo, se podría suponer que las películas se pueden resumir en dos factores: una dimensión de seriedad-comedia y una dimensión de fantasía-realidad, ambas numéricas, de tal forma que en la primera dimensión un valor más alto significa que una película es más seria y más bajo significa que es más inclinada a comedia. Para la segunda dimensión, similarmente, un valor más alto significa que la película está más inclinada a fantasía y un menor valor significa que está más apegada a la realidad. Estos factores latentes también se pueden pensar para los usuarios, en el sentido de que un usuario tiene un valor más alto en la primera dimensión si le gustan más las películas serias y un valor más alto en la segunda dimensión si le gustan las películas de fantasía.

Teniendo esto en cuenta, los usuarios y las películas se pueden representar en vectores. El vector correspondiente a un usuario que le gustan mucho las películas serias de fantasía se podría ver así:

$$u_1 = \begin{bmatrix} 3 & 4 \end{bmatrix}^T.$$

Y un usuario que le gustan las comedias pero es indiferente entre si es de fantasía o realidad se podría ver así:

$$u_2 = \begin{bmatrix} -2 & 0 \end{bmatrix}^T.$$

De la misma forma se podría pensar en dos películas, una seria con fantasía, como *El Señor de los Anillos*, y una apegada a la realidad de comedia, como *Mean Girls*. Sus vectores correspondientes se podrían ver así:

$$p_1 = \begin{bmatrix} 2 & 4 \end{bmatrix}^T$$

$$p_2 = \begin{bmatrix} -3 & -3 \end{bmatrix}^T.$$

Entonces una forma de modelar la calificación que daría cada usuario a cada película sería pensando en la combinación lineal de los usuarios y las películas. Es decir, la calificación que daría el usuario u_1 a la película p_1 sería el producto punto de los vectores: $u_1^T p_1 = 22$. Si se hace esto para todos los usuarios y películas, se tiene el siguiente sistema lineal:

$$R = UP^T,$$

donde

$$U = \begin{bmatrix} u_1^T \\ u_2^T \end{bmatrix} = \begin{bmatrix} 3 & 4 \\ -2 & 0 \end{bmatrix}, \quad P = \begin{bmatrix} p_1^T \\ p_2^T \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ -3 & -3 \end{bmatrix}$$

y

$$R = \begin{bmatrix} u_1^T p_1 & u_1^T p_2 \\ u_2^T p_1 & u_2^T p_2 \end{bmatrix} = \begin{bmatrix} 22 & -21 \\ -4 & 6. \end{bmatrix}$$

Aquí se puede ver que el usuario u_1, a quien le gustan las películas serias de fantasía califica más alto *El Señor de los Anillos* que *Mean Girls*, mientras que el usuario u_2 hace lo contrario, pues le gustan las comedias.

Ahora, en la vida real no se conocen los valores de las matrices U y P, sino que se tiene acceso a ciertas entradas de la matriz R, y el reto está en tener una buena estimación de las matrices U y P. Este problema es una instancia de la descomposición en valores singulares, con la peculiaridad de que no se tienen todas las entradas de la matriz R, sino solo una pequeña proporción de ellas.

Entonces, si se tiene acceso a la matriz R, se busca encontrar dos matrices U y P de rango k tales que $R \approx UP^T$. De esta forma, la aproximación de la calificación que da el usuario i al artículo j sería

$$r_{ij} = u_i^T p_j. \tag{2.6}$$

Hay distintas formas de elegir qué función de pérdida utilizar para saber qué tan cerca el producto de matrices está de R, pero muchas veces se utiliza la raíz del error cuadrático medio (RMSE por *root-mean-squared-error*), el cual se calcula de la siguiente forma:

$$RMSE = \left(\frac{1}{|A|} \sum_{(i,j) \in A} (r_{ij} - u_i^T p_j)^2 \right)^{\frac{1}{2}}, \tag{2.7}$$

donde r_{ij} es la calificación del usuario i del artículo j, A es el conjunto de usuarios y artículos para los cuales se conoce la calificación $r_i j$, $|A|$ es la cardinalidad de A (o sea, el total de artículos calificados por todos los usuarios), u_i es la i-ésima fila de la matriz U y p_j es la j-ésima fila de la matriz P. Es decir, el RMSE es la raíz cuadrada del promedio de las desviaciones de la aproximación de R y la R verdadera.

Otra opción muy utilizada es el error absoluto medio (MAE por *mean-absolute-error*), definido como la suma de las desviaciones absolutas de la aproximación de R a R, esto es

$$MAE = \frac{1}{|A|} \sum_{(i,j) \in A} |r_{ij} - u_i^T p_j|.$$

El RMSE puede ser preferible en situaciones en las que pequeños errores de predicción no son importantes, pues el RMSE penaliza errores más fuertemente que el MAE.

Por lo tanto, cuando se quiere minimizar el RMSE, el problema a resolver es

$$\min_{p,u} \sum_{(i,j)\in A} \left[(r_{ij} - u_i^T p_j)^2 + \lambda(\|p_j\|^2 + \|u_i\|^2) \right] \tag{2.8}$$

donde λ es un término de regularización que evita el sobreajuste.

Agregando sesgos

Al modelo simple enunciado en la sección anterior se le puede agregar la idea del modelo base: que existen sesgos en los artículos y en los usuarios. Entonces al modelo 2.6 se le agregan los sesgos, de tal forma que se tiene el nuevo modelo con sesgos

$$r_{ij} = \mu + a_i + b_j + u_i^T p_j, \tag{2.9}$$

con μ, a_i y b_j definidos anteriormente. Estos parámetros se pueden estimar resolviendo el problema

$$\min_{p,u,b,a} \sum_{(i,j)\in A} \left[\left(r_{ij} - \mu - a_i - b_j - u_i^T p_j \right)^2 + \lambda \left(\|p_j\|^2 + \|u_i\|^2 + a_i^2 + b_j^2 \right) \right] \tag{2.10}$$

Modelo de optimización

Sea L la función a optimizar en el problema 2.10, es decir,

$$\begin{aligned} L &= \sum_{(i,j)\in A} \left[\left(r_{ij} - \mu - a_i - b_j - u_i^T p_j \right)^2 + \lambda \left(\|p_j\|^2 + \|u_i\|^2 + a_i^2 + b_j^2 \right) \right] \\ &= \sum_{(i,j)\in A} \left[\left(x_{ij} - a_i - b_j - u_i^T p_j \right)^2 + \lambda \left(\|p_j\|^2 + \|u_i\|^2 + a_i^2 + b_j^2 \right) \right] \\ &= \sum_{(i,j)\in A} \left(x_{ij} - a_i - b_j - u_i^T p_j \right)^2 \\ &\quad + \lambda \left(\sum_j \sum_{m=1}^{k} p_{jm}^2 + \sum_i \sum_{m=1}^{k} u_{im}^2 + \sum_i a_i^2 + \sum_j b_j^2 \right). \end{aligned}$$

Para utilizar descenso en gradiente estocástico, se define la función de pérdida l_{ij} como

$$l_{ij} = \left(x_{ij} - a_i - b_j - u_i^T p_j\right)^2 + \lambda \left(\sum_{m=1}^{k} p_{jm}^2 + \sum_{m=1}^{k} u_{im}^2 + a_i^2 + b_j^2 \right).$$

Notar que $L = \sum_{(i,j) \in A)} l_{ij}$.

Las derivadas parciales de l_{ij} son

$$\frac{\partial l_{ij}}{\partial u_{im}} = -2e_{ij}p_{jm} + 2\lambda u_{im} \qquad \frac{\partial l_{ij}}{\partial p_{jm}} = -2e_{ij}u_{im} + 2\lambda v_{jm}$$

$$\frac{\partial l_{ij}}{\partial a_i} = -2e_{ij} + 2\lambda a_i \qquad \frac{\partial l_{ij}}{\partial b_j} = -2e_{ij} + 2\lambda b_j$$

donde $e_{ij} = x_{ij} - a_i - b_j - u_i^T p_j$ es el residual de la calificación del usuario i del producto j.

Entonces, para resolver 2.10, se inicia con un unas matrices P y Q, y los vectores a_i y b_j para todos los usuarios i y artículos j, con valores arbitrarios. Y se prosigue a ajustar U, P, a y b iterativamente usando la dirección contraria al gradiente en cada punto hasta cierto criterio de paro.

Algoritmo 1: Algoritmo de descenso en gradiente estocástico para 2.10

```
Iniciar vectores a, b y matrices P y Q
mientras no se cumpla el criterio de paro hacer
    para todo (i, j) ∈ A hacer
        para todo m ∈ 1, . . . , k hacer
            u_im ← u_im − γ (−2e_ij p_jm + 2λu_im)
            p_jm ← p_jm − γ (−2e_ij u_im + 2λv_jm)
        fin
        a_i ← a_i − γ (−2e_ij + 2λa_i)
        b_j ← b_j − γ (−2e_ij + 2λb_j)
    fin
fin
```

2.3.3. Evaluación de modelos

Una primera medida para evaluar un sistema de recomendación que ya fue presentada anteriormente es el RMSE. Esta medida da una idea de el desempeño del modelo para todos los artículos ya conocidos, sin embargo, no es suficiente para ofrecer recomendaciones que le vayan a gustar al usuario. La mayoría de las veces se busca recomendar cierto número de artículos a un usuario, digamos N, y se desea que estos N artículos sean de agrado del usuario, o a veces los usuarios están interesados en descubrir cosas nuevas. Es por esto que también se debe de evaluar cómo los sistemas se desempeñan en este tipo de situaciones.

Una forma de evaluar este tipo de recomendaciones es tomar esto como un problema de clasificación binario, en el cual se tiene un acierto si a un usuario se le recomendó un artículo en los mejores N (top-N) y este usuario lo vio/compró, y no se acierta en otro caso. Con este tipo de problemas se pueden obtener las medidas típicas en un problema de clasificación binario (precisión, tasa de verdaderos positivos, tasa de falsos negativos, etc).

En [2] se muestra una forma de evaluar de esta forma un sistema de recomendación, la cual consiste en seleccionar cada elemento i calificado con la calificación máxima por cada usuario u en el conjunto de prueba:

1. Seleccionar 1000 artículos adicionales que no fueron calificados por el usuario u. Se asume que la mayoría de ellos no van a ser de interés para el usuario u.

2. Calcular la calificación de acuerdo al modelo para el artículo i y cada uno de los 1000 artículos adicionales.

3. Se forma una lista ordenada de acuerdo a las calificaciones predichas por el modelo para los 1001 artículos. Sea p el lugar que ocupa el artículo de prueba i entre todos los elementos de la lista. El mejor resultado corresponde al caso en que el artículo de prueba i está antes de todos los artículos aleatorios (esto es, $p = 1$).

4. Se forma una lista de las mejores N recomendaciones al seleccionar los N artículos mejor calificados en la lista. Si $p \leq N$ entonces se tiene un acierto, en otro caso se tiene un desacierto. La probabilidad de obtener

aciertos aumenta si N aumenta, y si $N = 1001$, siempre se tiene un acierto.

Capítulo 3

Resultados

Los algoritmos presentados en este trabajo fueron implementados en el lenguaje para cómputo estadístico `R` [7] y fueron probados con un conjunto de películas calificadas de *MovieLens*, una página que ofrece recomendaciones de películas, los cuales están disponibles al público. *MovieLens* es un proyecto de *GroupLens*, un equipo de investigación de la Universidad de Minnesota orientado al cómputo social. Algunas de las funciones utilizadas fueron implementadas en el lenguaje `C++` y compiladas en `R` con el paquete `Rcpp` [3] [4].

3.0.1. Análisis exploratorio de datos

I. *MovieLens*

El conjunto de datos de *MovieLens* consiste en 20,000,263 calificaciones de 26,744 películas hechas por 138,493 usuarios. Todos los usuarios habían calificado al menos 20 películas. Las calificaciones están en una escala de 0.5 a 5, con intervalos de medio punto, es decir, se puede poner $0.5, 1, \ldots, 4.5, 5$ como calificación [5]. La distribución de las calificaciones se puede ver en la figura 3.1; mientras que en la figura 3.2 se puede ver el histograma de los promedios por película.

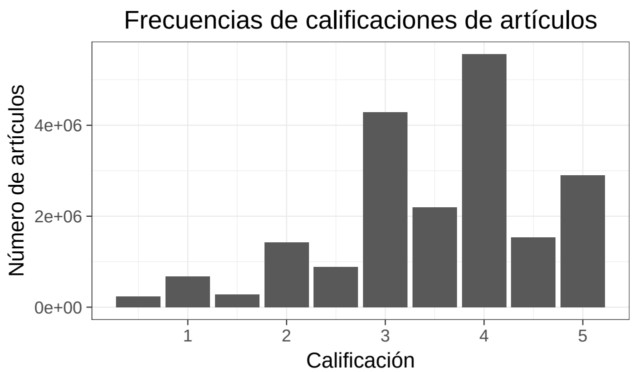

Figura 3.1: Frecuencia de calificaciones del conjunto de datos *MovieLens*.

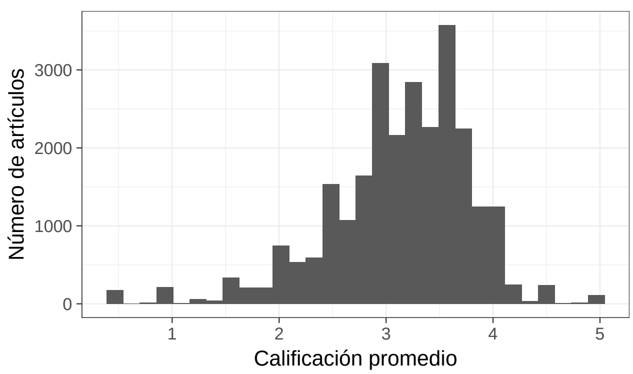

Figura 3.2: Histograma del promedio de calificaciones por película del conjunto de datos *MovieLens*.

En la sección 1.1 se introduce el concepto de cola larga. En la figura 3.3 se puede ver cómo este fenómeno ocurre en los datos de *MovieLens*, donde es claro que muy pocas películas tienen una gran cantidad de calificaciones, mientras que muchos artículos tienen pocas calificaciones.

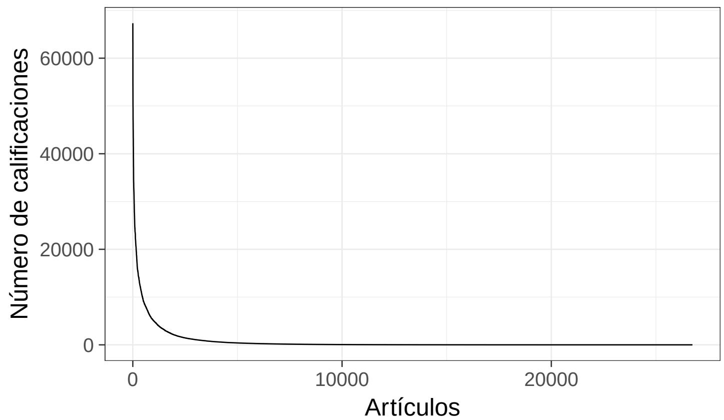

Figura 3.3: Cola larga del conjunto de datos *MovieLens*.

II. *BookCrossing*

El conjunto de datos de *BookCrossing* consiste en 1,149,780 calificaciones de 271,379 libros hechas por 278,858 usuarios. Los datos se obtuvieron durante cuatro semanas de agosto a septiembre de 2004. Las calificaciones están en una escala de 1 a 10, con intervalos de un punto, es decir, se puede poner $1, 2, \ldots, 9, 10$ como calificación. Se tiene como calificación especial el 0, el cual es solamente una calificación implícita, es decir, si leyó o no el libro [8]. Debido a que para usar el método presentado en este trabajo se necesitan calificaciones explícitas, se filtraron estas del conjunto de datos original. Con esta operación, quedaron 383,852 calificaciones de 153,683 libros hechas por 68,092 usuarios. La distribución de las calificaciones filtradas se puede ver en la figura 3.4; mientras que en la figura 3.5 se puede ver el histograma de los promedios por libro.

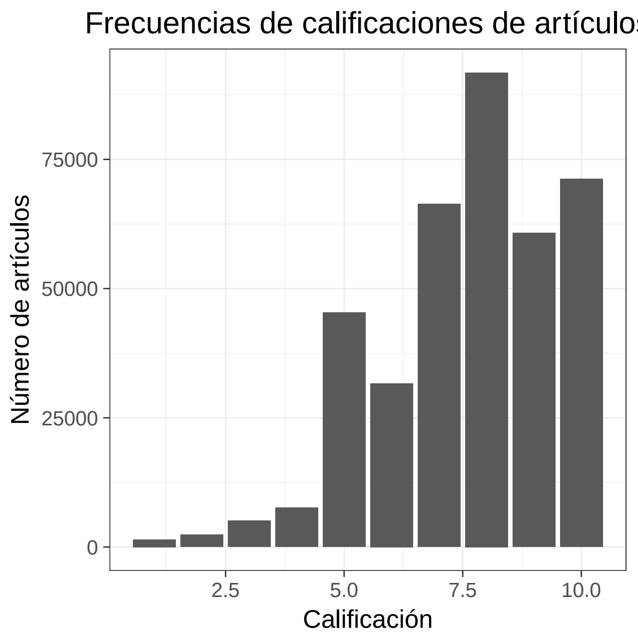

Figura 3.4: Frecuencia de calificaciones del conjunto de datos *BookCrossing*.

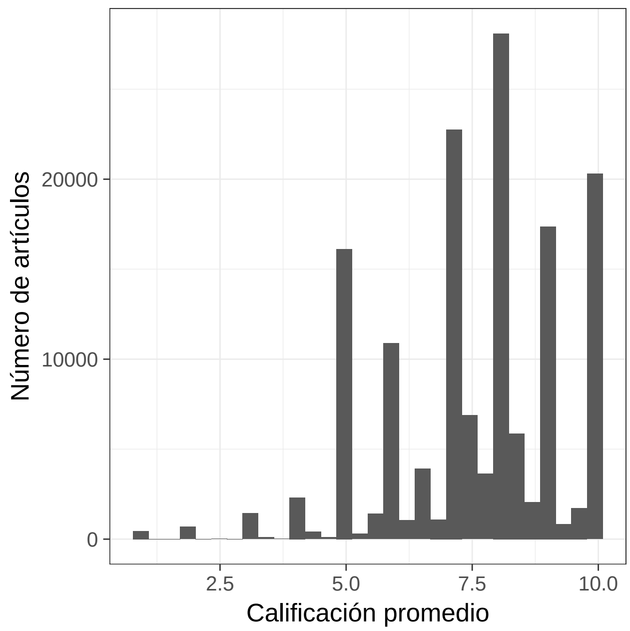

Figura 3.5: Histograma del promedio de calificaciones por película del conjunto de datos *BookCrossing*.

En la figura 3.6 se puede ver la cola larga en los datos de *BookCrossing*, donde nuevamente es claro que muy pocas películas tienen una gran cantidad de calificaciones, mientras que muchos artículos tienen pocas calificaciones.

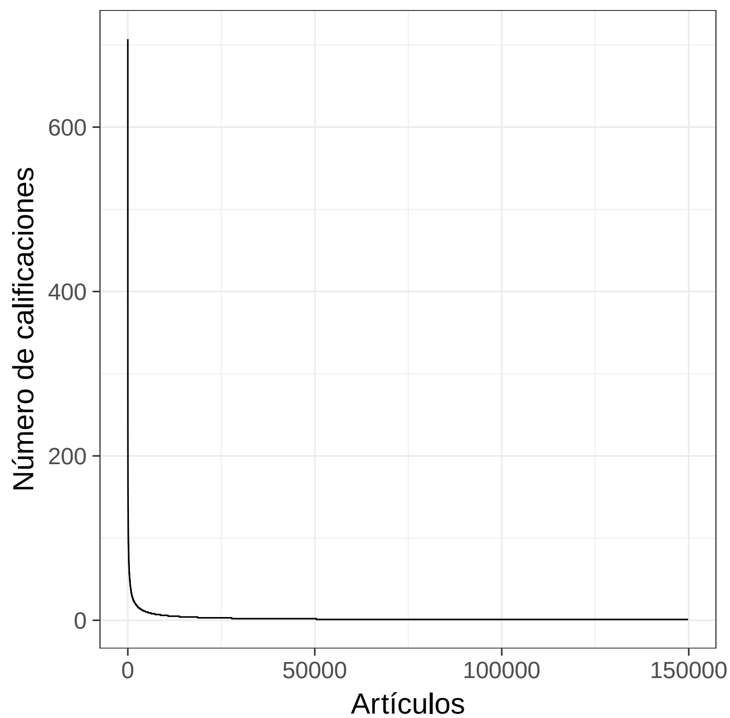

Figura 3.6: Cola larga del conjunto de datos *BookCrossing*.

3.0.2. Comparación de modelos

Para cada conjunto de datos se calculó un modelo base como el descrito en la sección 2.3.2.I en la ecuación 2.2 y un modelo de factorización como descrito en la sección 2.3.2.III. Cada uno de los conjuntos de datos se dividió en tres subconjuntos: un conjunto de entrenamiento, un conjunto de prueba y un conjunto de validación. Los parámetros de los modelos fueron estimados usando el conjunto de entrenamiento. En el caso del modelo de factorización, el conjunto de validación fue utilizado para estimar el error de predicción y poder utilizarlo como criterio de paro en el algoritmo de optimización y así evitar el sobreajuste. Para el modelo base, el conjunto de validación no fue utilizado. La medida de error utilizada fue la raíz del error cuadrático medio (RMSE), definido en la ecuación 2.7. De aquí en adelante, cuando se hable de error, es con referencia a la raíz del error cuadrático medio.

I. *MovieLens*

Para *MovieLens*, el número de usuarios, artículos y calificaciones en cada uno de los conjuntos se puede ver en la tabla 3.1.

Tabla 3.1: Número de usuarios, calificaciones y artículos en los conjuntos de *MovieLens*.

Conjunto	Número de artículos	Número de usuarios	Número de calificaciones
Entrenamiento	26,247	138,493	18,029,206
Validación	6,256	41,483	1,469,158
Prueba	2,895	20,676	501,899

En la figura 3.7 se pueden ver los errores del modelo base en el conjunto de prueba, de acuerdo al valor del parámetro de regularización γ, definido en la ecuación 2.2. Se puede ver que con $\gamma \approx 19$ se minimiza el valor de la raíz del error cuadrático medio en el conjunto de prueba.

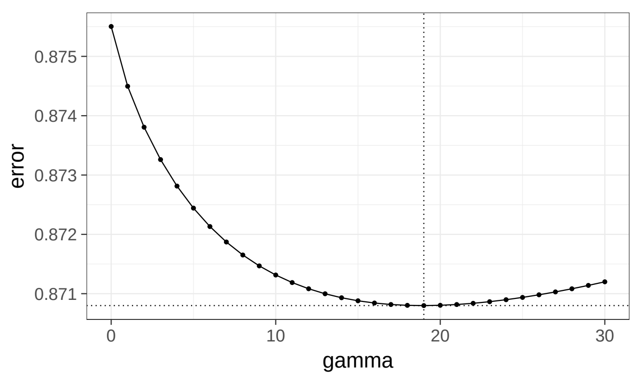

Figura 3.7: Errores del modelo base en el conjunto de prueba de *MovieLens*, de acuerdo al valor del parámetro de regularización γ.

En la figura 3.8 se pueden ver los errores de entrenamiento y de validación del modelo de factorización con distintos valores en el número de dimensiones latentes, tasa de aprendizaje y parámetro de regularización λ. Como es de esperarse, el error de entrenamiento es menor en cada uno de los casos. También se podría esperar que mientras más dimensiones latentes haya, el error de entrenamiento sea menor, debido a que hay más dimensiones para aproximar la matriz que se quiere aproximar; y es justo lo que sucede: los modelos con más dimensiones latentes tienen menor error de entrenamiento. Sin embargo, se sabe que menor error de entrenamiento no implica necesariamente menor error de validación; por lo que se busca el modelo con menor error de validación. En este caso, todos los modelos tienen un error de validación relativamente parecido, siendo el de 500 dimensiones latentes, tasa de aprendizaje de 0.001 y $\lambda = 0.01$ el modelo con menor error (0.763325), siguiendo el de 200 dimensiones latentes, tasa de aprendizaje de 0.001 y $\lambda = 0.01$ con un error de 0.765166.

Aquí se podría tomar una decisión en cuanto a qué modelo utilizar: el modelo con 500 dimensiones latentes tiene menor error, pero tiene más del doble de dimensiones latentes que el de 200, el cual tiene un desempeño muy parecido, por lo que tal vez se preferiría utilizar el modelo con 200 dimensiones latentes debido a que es más simple y tiene casi el mismo desempeño. El modelo que se utilizó como final fue el de 200 dimensiones latentes, con un error en el conjunto de prueba de 0.7651675 (muy parecido al del conjunto de validación).

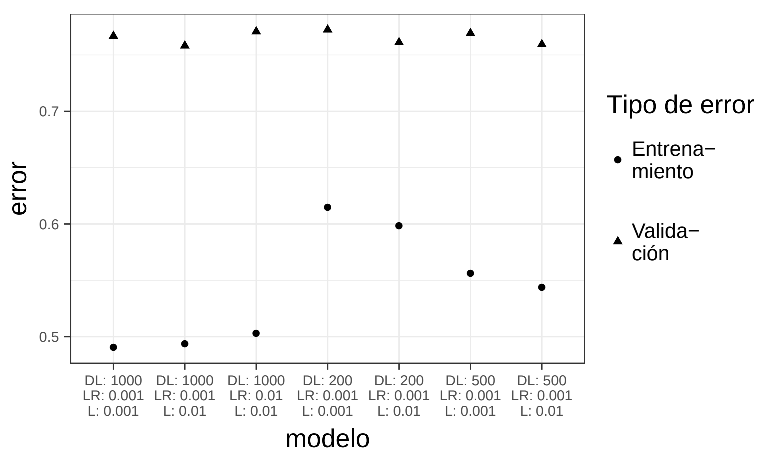

Figura 3.8: Errores del modelo de factorización en el conjunto de prueba de *MovieLens*. LR es la tasa de aprendizaje, DL es el número de dimensiones latentes y L es el parámetro de regularización λ.

En la figura 3.9 se pueden ver los errores de entrenamiento y de validación del modelo de factorización final (i.e., 200 dimensiones latentes, tasa de aprendizaje de 0.001 y $\lambda = 0.01$) en cada iteración del algoritmo de descenso en gradiente estocástico. Se puede ver cómo ambos errores disminuyen con cada iteración, sin embargo, el error de entrenamiento disminuye mucho más rápido. Si no se hubiera tenido como criterio de paro el error de validación, posiblemente el error de entrenamiento hubiera continuado disminuyendo, pero el de validación hubiera empezado a aumentar, debido al sobreajuste.

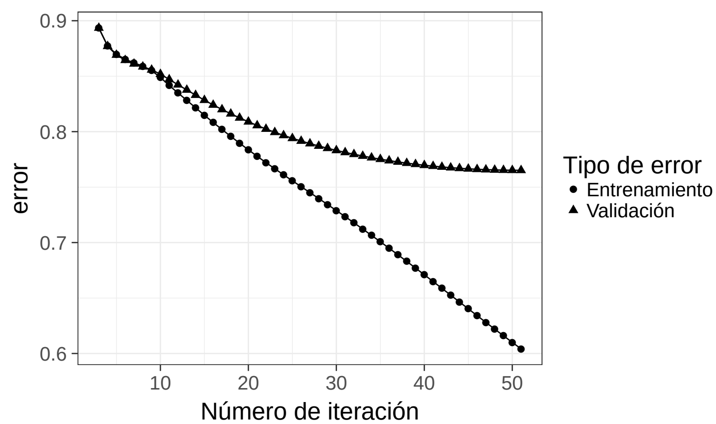

Figura 3.9: Errores de entrenamiento y validación del modelo de factorización en cada iteración del algoritmo de optimización para *MovieLens*.

Como se menciona en 2.3.3, además del RMSE, una forma de evaluar un sistema de recomendación es como un problema de clasificación con la tarea de recomendar los mejores N artículos. En las figuras 3.10 y 3.11 se puede ver el resultado de esta tarea comparando el modelo base y el modelo de factorización como función de N, el número de artículos recomendados. Se puede observar que el modelo de factorización es superior que el modelo base para todas las N para las que se calcularon las medidas. Por ejemplo, para $N = 10$, el recall del modelo de factorización es de 0.429, lo cual significa que el modelo tiene una probabilidad de 0.429 de poner en el top-10 de recomendaciones una película que atraiga a un usuario; mientras que el recall del modelo base en $N = 10$ es de 0.059, lo cual significa que la probabilidad mencionada es de 0.059.

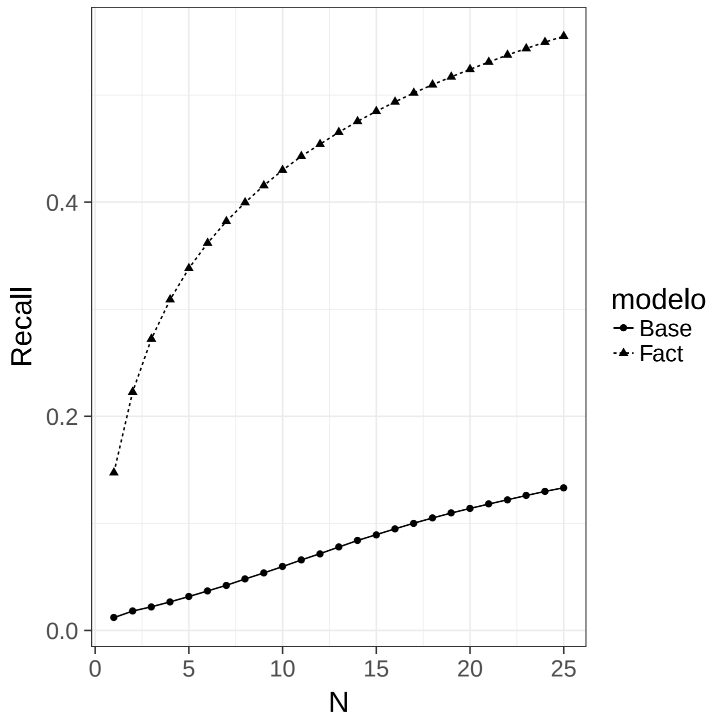

Figura 3.10: Recall para cada valor de N en la tarea de las mejores N recomendaciones para *MovieLens*.

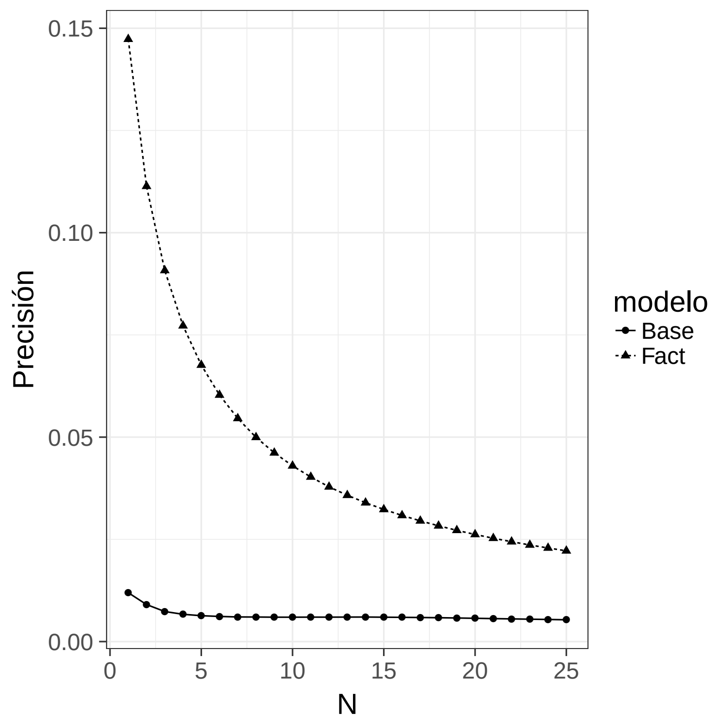

Figura 3.11: Precisión para cada valor de N en la tarea de las mejores N recomendaciones para *MovieLens*.

Como se mencionó antes, cada artículo es expresado como una combinación lineal de distintos factores latentes, por lo que artículos y usuarios son mapeados a un mismo espacio latente. Por ende, cada artículo tiene un vector representante en un espacio euclidiano, por lo que se pueden calcular distancias y encontrar vecinos con estas distancias. Con esta idea, además de recomendar directamente a un usuario artículos personalizados, se pueden encontrar los artículos más parecidos a un determinado artículo mediante la norma euclidiana de la diferencia o la similitud coseno.

Para el conjunto de datos de *MovieLens* se calculan los vecinos más cercanos para algunas películas selectas. Se utilizó la distancia euclidiana y se calculó con el paquete `RANN` [1] de `R`. Estos vecinos se pueden ver en las tablas 3.2 y 3.3, y se puede apreciar que en cierta forma, los vecinos más cercanos de cada película sí son parecidos.

Tabla 3.2: Algunas películas y sus vecinos más cercanos.

Película	**Película más cercana**	**Distancia**
Aladdin (1992)	Lion King, The (1994)	1.74
	Beauty and the Beast (1991)	1.79
	Little Mermaid, The (1989)	2.09
	Princess and the Frog, The (2009)	2.33
	Tarzan (1999)	2.42
	Mulan (1998)	2.43
	Oliver & Company (1988)	2.47
	Great Mouse Detective, The (1986)	2.47
	How to Train Your Dragon 2 (2014)	2.49
	Anastasia (1997)	2.49
Hannibal (2001)	Red Dragon (2002)	2.58
	Hitcher, The (2007)	2.88
	Venom (2005)	2.89
	Quiet, The (2005)	2.89
	Hannibal Rising (2007)	2.89
	Mindhunters (2004)	2.9
	187 (One Eight Seven) (1997)	2.91
	Raven, The (2012)	2.91
	Taking Lives (2004)	2.93
	Don't Say a Word (2001)	2.93
Harry Potter and the Sorcerer's Stone (2001)	HP and the Chamber of Secrets (2002)	0.47
	HP and the Prisoner of Azkaban (2004)	1.11
	HP and the Goblet of Fire (2005)	1.13
	HP and the Order of the Phoenix (2007)	1.33
	HP and the Half-Blood Prince (2009)	1.59
	HP and the Deathly Hallows: Part 2 (2011)	1.83
	HP and the Deathly Hallows: Part 1 (2010)	1.87
	Narnia: The Voyage of the Dawn Treader (2010)	2.76
	Narnia: Prince Caspian (2008)	2.88
	The Lion King 1 1/2 (2004)	2.91

Tabla 3.3: Algunas películas y sus vecinos más cercanos.

Película	**Película más cercana**	**Distancia**
Lion King, The (1994)	Aladdin (1992)	1.74
	Beauty and the Beast (1991)	2.1
	Mulan (1998)	2.4
	Little Mermaid, The (1989)	2.42
	Tarzan (1999)	2.44
	Oliver & Company (1988)	2.44
	Princess and the Frog, The (2009)	2.48
	Frozen (2013)	2.49
	Land Before Time, The (1988)	2.49
	How to Train Your Dragon 2 (2014)	2.55
Pulp Fiction (1994)	Reservoir Dogs (1992)	2.26
	Inglorious Bastards (1978)	3.03
	Goodfellas (1990)	3.15
	An Evening with Kevin Smith 2 (2006)	3.19
	Black Mirror (2011)	3.2
	Ricky Gervais Live 3: Fame (2007)	3.23
	Nightcrawler (2014)	3.24
	Inside Llewyn Davis (2013)	3.25
	Flirting (1991)	3.25
	Generation Kill (2008)	3.25
Toy Story (1995)	Toy Story 2 (1999)	1.43
	Monsters, Inc. (2001)	1.91
	Toy Story 3 (2010)	2
	Bug's Life, A (1998)	2.02
	Finding Nemo (2003)	2.03
	Incredibles, The (2004)	2.26
	Up (2009)	2.47
	For the Birds (2000)	2.5
	Wreck-It Ralph (2012)	2.5
	Partly Cloudy (2009)	2.5

II. *BookCrossing*

Para *BookCrossing*, el número de usuarios, artículos y calificaciones en cada uno de los conjuntos se puede ver en la tabla 3.4.

Tabla 3.4: Número de usuarios, calificaciones y artículos en los conjuntos de *BookCrossing*.

Conjunto	Número de artículos	Número de usuarios	Número de calificaciones
Entrenamiento	140,807	64,459	351,217
Validación	13,072	5,332	21,224
Prueba	5,065	2,286	7,435

Capítulo 4

Conclusiones

Bibliografía

[1] Sunil Arya y col. *RANN: Fast Nearest Neighbour Search (Wraps Arya and Mount's ANN Library)*. R package version 2.5. 2015. URL: https://CRAN.R-project.org/package=RANN.

[2] Paolo Cremonesi, Yehuda Koren y Roberto Turrin. «Performance of Recommender Algorithms on Top-n Recommendation Tasks». En: *Proceedings of the Fourth ACM Conference on Recommender Systems*. RecSys '10. Barcelona, Spain: ACM, 2010, págs. 39-46. ISBN: 978-1-60558-906-0. DOI: 10.1145/1864708.1864721. URL: http://doi.acm.org/10.1145/1864708.1864721.

[3] Dirk Eddelbuettel. *Seamless R and C++ Integration with Rcpp*. ISBN 978-1-4614-6867-7. New York: Springer, 2013.

[4] Dirk Eddelbuettel y Romain François. «Rcpp: Seamless R and C++ Integration». En: *Journal of Statistical Software* 40.8 (2011), págs. 1-18. URL: http://www.jstatsoft.org/v40/i08/.

[5] F Maxwell Harper y Joseph A Konstan. «The movielens datasets: History and context». En: *ACM Transactions on Interactive Intelligent Systems (TiiS)* 5.4 (2016), pág. 19.

[6] Jure Leskovec, Anand Rajaraman y Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 2014. (Visitado 27-05-2015).

[7] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2016. URL: https://www.R-project.org/.

[8] Cai-Nicolas Ziegler y col. «Improving recommendation lists through topic diversification». En: *Proceedings of the 14th international conference on World Wide Web*. ACM. 2005, págs. 22-32.